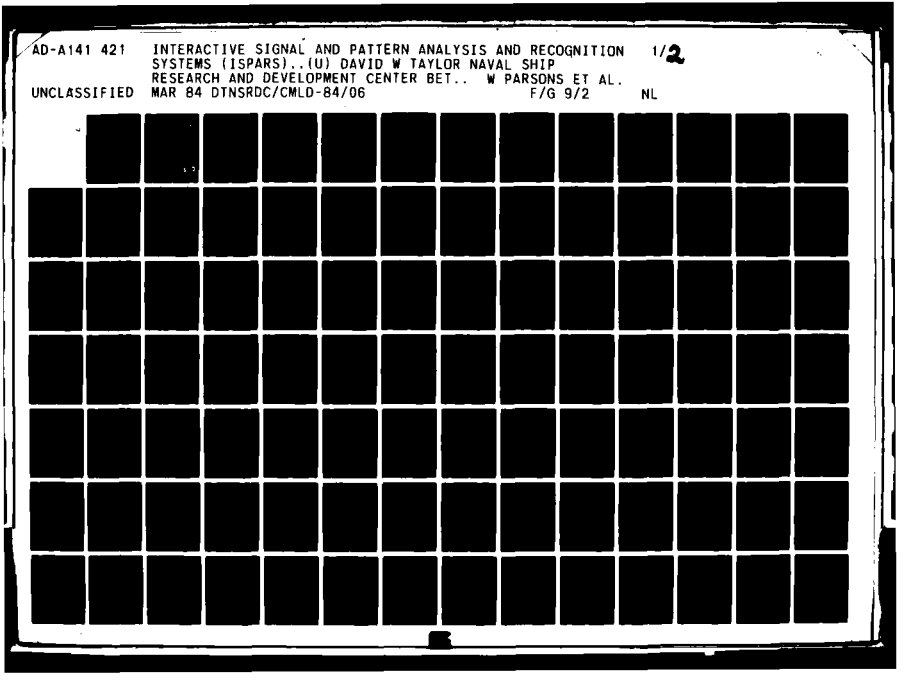
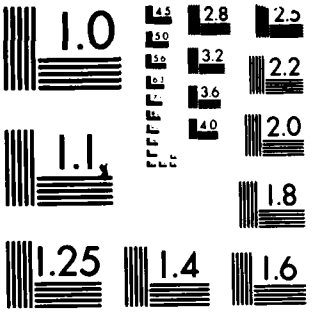
AD-A141 421 INTERACTIVE SIGNAL AND PATTERN ANALYSIS AND RECOGNITION 1/2
SYSTEMS (ISPARS)..(U) DAVID W TAYLOR NAVAL SHIP
RESEARCH AND DEVELOPMENT CENTER BET.. W PARSONS ET AL.
UNCLASSIFIED MAR 84 DTNSRDC/CMLD-84/06 F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

(12)

DTNSRDC/CMLD-84/06

DAVID W. TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER

Bethesda, Maryland 20084

AD-A141 421

INTERACTIVE SIGNAL AND PATTERN ANALYSIS
AND RECOGNITION SYSTEMS (ISPARS)
USERS' GUIDE

by

William Parsons
Joseph Garner
James Carlberg
Sidney Berkowitz

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

COMPUTATION, MATHEMATICS, AND LOGISTICS DEPARTMENT
RESEARCH AND DEVELOPMENT REPORT

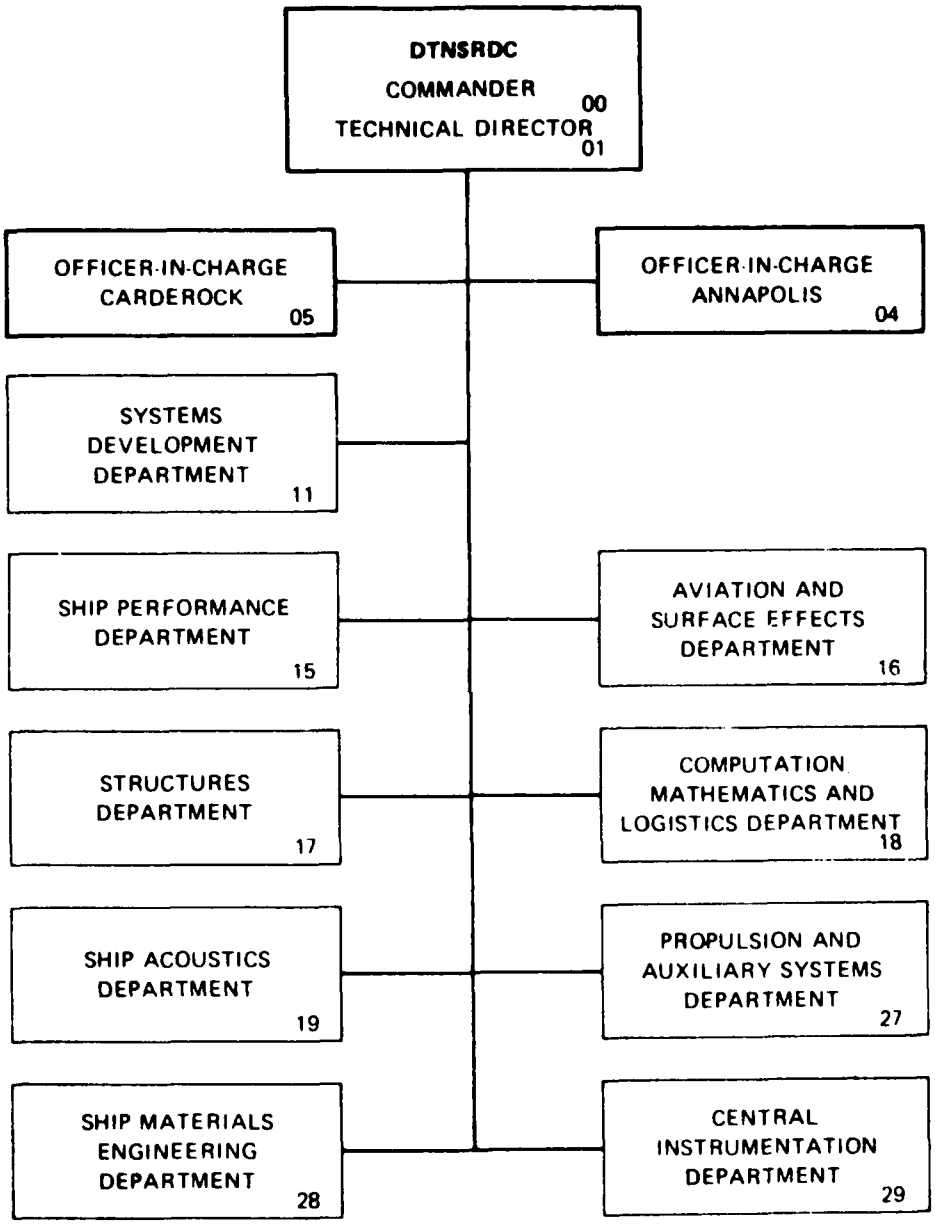
March 1984

DTNSRDC/CMLD-84/06

INTERACTIVE SIGNAL AND PATTERN ANALYSIS
AND RECOGNITION SYSTEMS (ISPARS)
USERS' GUIDE

NDW-DTNSRDC 5602/30 (2-80)
(supersedes 3960/46)

MAJOR DTNSRDC ORGANIZATIONAL COMPONENTS



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: DTNSRDC/CMLD-84/06	2. GOVT ACCESSION NO.: A141421
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): INTERACTIVE SIGNAL AND PATTERN ANALYSIS AND RECOGNITION SYSTEMS (ISPARS) USERS GUIDE	5. TYPE OF REPORT & PERIOD COVERED: Interim Report Sep 79 - Sep 80
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Parsons, W. Carlberg, J. Garner, J. Berkowitz, S.	8. CONTRACT OR GRANT NUMBER(s):
9. PERFORMING ORGANIZATION NAME AND ADDRESS: David Taylor Naval Ship Research and Development Center Bethesda, MD 20084	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: (See Reverse Side)
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Sea Systems Command Materials and Mechanics Division Washington, D.C. 20362	12. REPORT DATE: March 1984
	13. NUMBER OF PAGES: 135
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Signal Analysis
Pattern Recognition
Spectral Analysis
Interactive Graphics
Waveform Processing

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report is a functional description and user's guide to the interactive Signal and Pattern Analysis and Recognition Systems (ISPARS). ISPARS is a loosely integrated interactive graphics software system, installed on a PDP11/45 computer system, for processing sampled signal waveforms and determining the pattern classes into which waveforms may be separated.

ISPARS capabilities for signal analysis include segmentation, spectral decomposition, transformation (Filtering), normalization, and characterization of signals. Signal analysis is viewed as a means of eliminating redundant

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(Block 10)
Program Element 61153N
Project SR01403
Task Area SR0140301
Work Unit 1808-010

(Block 20 continued)

information in the signal and thus extracting significant features. These features form the input to the pattern analysis procedure which permits the construction of algorithms for clustering and classifying feature strings. ISPARS affords the user a convenient, flexible means of visualizing and exploring a sampled data stream by employing interactive displays of complex transformations, easy parameter modification, convenient data stream control, and a simple, concise command language.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This report is a functional description and user's guide to the Interactive Signal and Pattern Analysis and Recognition System (ISPARS). ISPARS is a loosely integrated interactive graphics software system, installed on a PDP 11/45 computer system, for processing sampled signal waveforms and determining the pattern classes into which waveforms may be separated.

ISPARS capabilities for signal analysis include segmentation, spectral decomposition, transformation (filtering), normalization, and characterization of signals. Signal analysis is viewed as a means of eliminating redundant information in the signal and thus extracting significant features. These features form the input to the pattern analysis procedure which permits the construction of algorithms for clustering and classifying feature strings. ISPARS affords the user a convenient, flexible means of visualizing and exploring a sampled data stream by employing interactive displays of complex transformations, easy parameter modification, convenient data stream control, and a simple, concise command language.

ADMINISTRATIVE INFORMATION

This work was completed in the Computer Science and Information Systems Division of the Computation, Mathematics, and Logistics Department under the sponsorship of NAVSEA 03F, Task Area SR0140301, Task 15321, Element 61153N.

OVERVIEW

This report is a functional description and users guide to the Interactive Signal and Pattern Analysis and Recognition System (ISPARS). As the name suggests, ISPARS deals with two broad areas of data analysis: the processing of sampled signal waveforms and the determination of pattern classes into which waveforms may be separated. Signal analysis is concerned with the segmentation, spectral decomposition, transformation (filtering), normalization, and characterization of signals. It is intended to aid the physical scientist or engineer in gaining insight into the basic physical processes which generate the composite signal and to suggest invariants which are characteristic of the overall physical process. As such, signal analysis is a means of eliminating redundant information in the signal and thus of extracting significant features. These features may then form the input to a pattern analysis procedure which permits the construction of algorithms for clustering and

classifying test feature strings, for reducing their dimensionality, and for evaluating the accuracy of and confidence in the classification algorithm itself.

Aside from the pragmatic value of determining a recognition scheme for applications - such as target identification, fault isolation, and performance prediction - the classification procedure offers the analyst insight into the relationships among various partitions of the data. Through such analysis one can, for example, formulate discriminant hypersurfaces, spot faulty data, construct subclasses reflecting multi-mode distribution clusters, and - for stochastic processes - determine syntaxes for parsing signal feature waveform strings. Since signal and pattern analysis are well-established technological areas supported by a wealth of literature, it would be difficult to build a system offering a comprehensive suite of techniques drawn from those areas. Moreover, there are ancillary techniques - such as high-speed analog-to-digital conversion, hardware associative memory, and speech/signal synthesis - which are part of the particular PDP 11/45 installation on which ISPARS resides but which are not described here as part of the system itself. Nevertheless, ISPARS complements those techniques and offers a resource of analysis techniques which is effective in the sense that the visual perception of the user is brought to bear in extracting meaning from the data in a way not possible with non-graphics-oriented methods.

ISPARS, then, is an interactive graphics software system which affords the user a convenient, flexible means of visualizing and exploring a sampled data stream in a variety of forms. From the user's viewpoint, ISPARS has a number of properties which are crucial to an effective signal/pattern analysis:

- o interactive displays - which portray complex transformations and projections, track through a data stream (in some cases automatically), react to parameter modification, indicate local variable values under potentiometer or light pen control, and permit the drawing or emphasis of data partititions.
- o data stream control - which affords an unbroken view of the time series and its transformations in a manner transparent to storage/communications.

- o linguistic interface - which activates the system through a simple, concise command language and its associated interactive command string interpreter, thus avoiding to a great extent program calls.

This report is intended as a user's guide to ISPARS and contains functional descriptions of the system components and information needed to use the program. The detailed descriptions of the subroutines, files, and linking procedures used in program development are documented in a forthcoming report devoted to program descriptions.[1*]

SIGNAL ANALYSIS SUBSYSTEMS

The signal analysis system consists currently of three modules: VU, SELECT, and WATERFALL.

The VU module is devoted to basic transformations of the raw data stream, including movement along the time series fast Fourier transforms (FFT), power/phase spectra, smoothing, peakpicking, and potentiometer-controlled intensity identification. Peaks of the power spectrum picked in VU can be made input to SELECT, where time slice selection may precede frequency tracking. Values of frequency, phase, amplitude, and bandwidth may be monitored under potentiometer control for either frequency or phase track displays, which can be scrolled either in time or, in the case of frequency tracks, in frequency. In WATERFALL, one may have an overall, dynamically moving view of the time-dependent power spectra from several angles prior to processing by VU or by normalization routines.

PROGRAM VU

Introduction

VU is an interactive, general purpose computer program for examining and analyzing digitized signals. The program currently runs on a PDP 11/45 minicomputer system with a VT11 processor supported by the RT-11 operating system. The digitized signals are stored on disk or magnetic tape as an infinite data stream. The graphics display monitor is used to display the signal data and results of data analysis such as spectral decomposition,

---

* A complete list of references is given on page 125.

peakpicking, and smoothing. The spectral decompositions may be produced by FFT and power spectra using user selected windowing options. Peakpicking routines locate and mark peaks in a spectral decomposition, and smoothing algorithms remove noise in a specified data buffer. VU can display a marker whose position is under control of a potentiometer on the PDP's laboratory peripheral system. This marker can be used to the identify amplitude and/or frequency of a data point displayed on the graphics screen.

The RT-11 commands for running VU and examples of the graphic displays generated by these commands are presented in the section on VU Functions.

Theory of Operation

Command Concept. VU is composed mostly of FORTRAN routines[2] and some assembly language routines. Graphic displays are generated using FORTRAN callable system routines.[3] The routines are linked as a standard RT-11 overlay structure with a root segment and 42 overlay segments divided into four overlay regions. The RT-11 overlay scheme is described in section 6 of the RT-11 System Reference Manual.[4] A separate report[1] contains the RT-11 BATCH commands to link VU.

VU operation is controlled by using a command language. The user types on the PDP 11's terminal command strings composed of a one or two-character command code followed by an optional parameter string. The parameter string is preceded by an equals sign (=). Command codes effect movement of data, scaling of data, spectral decomposition, peakpicking, parameter changes, changing input streams, and exiting the program. An example of a command string is

```
J1=2573
```

which means the first point displayed will be the 2573rd point of the raw data. The complete list of VU commands is given in the section on VU Operating Instructions, and a complete description of each command is given in the section on VU Functions.

Data Manipulation. Most commands will alter the graphics display. VU starts by displaying the first 512 points of raw data. As the user moves through the data, new displays are created showing the user the current location in the

raw data. Move commands include forward (F, F1=), back (B, B1=), or jump (J1=). The user can specify the number of data points displayed (S1=) and the vertical scale (S).

Spectral Analysis. Spectral decompositions are performed on the points displayed. Spectral decompositions currently include a fast Fourier transform (T) and a power spectrum (P). The decompositions can be applied with or without a window. Currently a modified Bessel function is available as the window. Power spectrum options allow the user to display the results using either a linear or logarithmic scale. A peakpicker routine can be applied (P5) to the power spectrum results and peaks will be identified on the screen with blinking X's. The frequency and amplitude of any designated spectrum point may be displayed on the screen. Major peaks of consecutive power spectra can be stored on a file for later examination by program SELECT, described in a later section of this report.

Input-Output. A hardcopy of any display can be generated with a "G" command. The hardcopy is produced on an X-Y plotter by routines that interpret the contents of the display buffer. The hardcopy reproduces only points and vectors; text is not reproduced.

VU reads raw data from a user specified file. The user specifies the RT-11 standard file as soon as VU is initiated. The data on file are multichannel, multiplexed digital values. Once the file specification is accepted, VU will ask the user to enter the number of channels and the channel to be displayed. All commands will process the data file associated with the specified channel. When the user desires to look at a new file, an "R" command is entered at the terminal. VU will ask the user to specify the file and channel to be observed. The user can change the number of channels specified. This option is currently used to observe every n^{th} point of raw data.

The PDP 11/45 is connected to an SDS 910 computer through a special purpose interface. The SDS has a conversion system that generates a digital tape from up to 24 multiplexed analog channels. The PDP/SDS interface allows the transfer to the PDP of the conversion data from tape to a user-specified PDP file.

The VU command language has two submodes. When a submode is entered, the commands assume a new meaning until a submode is exited. One submode allows

the user to change VU parameters. The parameters are displayed on the screen with the current value. The user changes a parameter by entering the appropriate command string with the desired parameter value.

The other submode enables a smoothing process. The smoothing process converts an array of data into a series of rises, levels, and falls. The smoothing mode has commands for data movement, for display points, for input/output, and for changing parameters. Most of these commands provide capabilities similar to those for VU in its normal mode.

Smoothing

The smoothing programs are a subsection of program VU consisting of subroutines designed to approximate a time ordered series of numbers by a sequence of simple line segments. The input to the smoothing section may be either digital acoustic data or frequency track data generated by program SELECT. The smoothing routines are coded in PDP-11 FORTRAN for the PDP 11/45 computer and use the VT-11 Graphics Processor for displaying results. The smoothing process is command driven and affords the user a convenient means of observing intermediate smoothing results.

The smoothing process contained in VU has two key subroutines: KLASIT, which obtains a first pass approximation of the level changes in the raw data, and WAMSER, which attempts to fit the piecewise linear approximation, generated by KLASIT, more closely to the original curve shapes. Since the output of each of these routines is a smoothed signal, the following discussion will occasionally refer to the "KLASIT smoothed data" or the "WAMSER smoothed data" to conveniently distinguish between them. The basic goal of the smoothing module is to produce a good piecewise linear approximation to the input signal such that changes in level in the approximation are not affected by the noise in the raw data. Consequently, not only does KLASIT ignore transitions attributed to the noise, but the entire smoothing process is built on a careful effort to estimate the true noise level of the input signal. To explain the overall smoothing process, the KLASIT and WAMSER approximations will first be described separately, and then the complete smoothing schema will be tied together.

KLASIT Approximation. Subroutine KLASIT examines the original raw data and generates a sequence of rise, fall, or level linear forms to approximate the raw data. Each rise or fall has an associated endpoint and each level, as well as the first and last points of the data, has an associated value equal to the amplitude of the raw data at those points.

A sequence of points in the raw data, each of which is not more than NABR amplitude units away from an initial (base) point, is approximated by a level form. A point more than NABR amplitude units away from the base point but surrounded by points within NABR units of the base point is considered isolated and is included in the level. To qualify as a bona fide level the set of points must also have a minimum duration. Should a jump occur before the duration threshold is exceeded, the points are included in the rise or fall instead of constituting a level.

The process of identiying rises and falls resembles a peak-picking algorithm. A base point in the input vector is initialized and not updated until there is a "significant" change (i.e., the change in amplitude from the base point exceeds a threshold). The indices of the current local maximum and local minimum are retained at all times. At every significant change, the base point is updated to the point at which the change occurred. The direction of the change determines the state of the approximating form, i.e., a rise or a fall. The endpoint of any form is the point at which it changes direction, begins a level, or (in the case of levels) ceases to be a level. For each form, its type (rise, fall, level), endpoint, and amplitude at the endpoint are stored, and the resulting sequence of triplets represents the KLASIT approximation to the raw data.

WAMSER Approximation. Subroutine WAMSER examines the results of the KLASIT smoothing process and generates a second order approximation to the change in intensity levels. Each rise or fall form in the KLASIT approximation is compared to the original data to refine the endpoints of the transition and determine its inflection. KLASIT data consist of rise, fall, or level forms with no juxtaposition of the same type form, but WAMSER takes each rise or fall and supplies added information, which has the effect of breaking the form into two segments of the same type to better approximate the curve of the original data.

The WAMSER smoothing begins with the identification of transition endpoints. For consecutive non-level forms (e.g., a rise followed by a fall), the start of the second transition (and the endpoint of the first transition) is that point at which the direction changes, the same point provided by the KLASIT data. If a rise or fall is preceded by a level, the KLASIT data are changed to make the transition begin within the level and at the end of the first series of points of minimum length RUN (a program parameter) which contains only "isolated" jumps, if any, in the direction of the transition. An isolated jump is a change in a point whose two successors are at the same level as its predecessors. Similarly, if a level follows a rise or fall, the transition associated with the rise or fall ends within the level region at the endpoint (working backwards) of a similar "run" of points in the level. Figure 1 shows some sample data, the KLASIT approximation, and the transition endpoints determined by WAMSER.

Once the transition endpoints have been determined, a transition decay time is computed to give some idea of the shape of the transition curve. This decay time is the number of samples required for the transition to achieve a set percentage (a program parameter) of the total change in intensity level. Figure 2 illustrates how decay time distinguishes two different types of rise transitions.

The effect of WAMSER on levels may be to shorten their duration because the endpoints of the rise/fall may be extended into the level. A non-level form, in addition to possibly expanding in duration, may also have a decay time less than its total duration. Such an occurrence effectively defines two non-level forms that better describe the transition (Figure 2). The resulting WAMSER description of the original data is a sequence of triplets, each of which identifies a rise, level, or fall form by its endpoint, amplitude at the endpoint, and decay time (decay time for levels is zero).

Smoothing Schema. The smoothing process (Figure 3) requires that an estimate of the noise be used in approximating the raw data so that legitimate changes in amplitude can be recognized. A change in amplitude at a point in the raw data is not considered legitimate if the change does not exceed the noise level at that point. Since the noise estimate can have a critical effect on the result, special pains are taken to obtain a good estimate of the noise. A

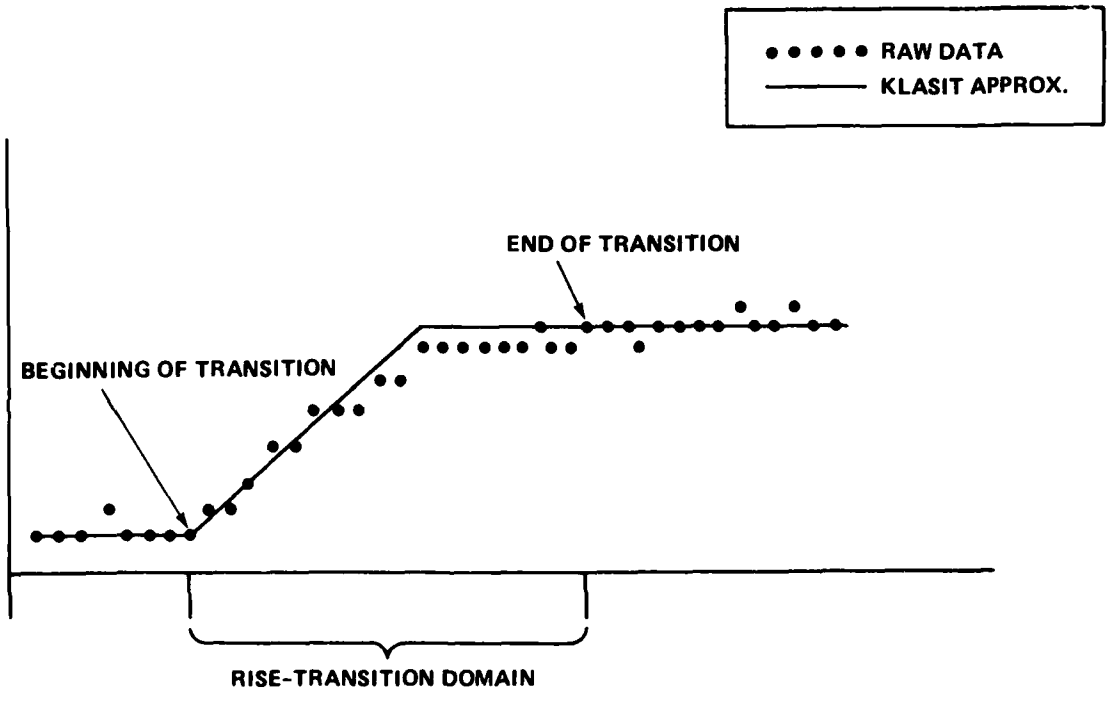


Figure 1 - WAMSER Determined Transition Bounds Compared to KLASIT Approximation. (RUN parameter = 4).

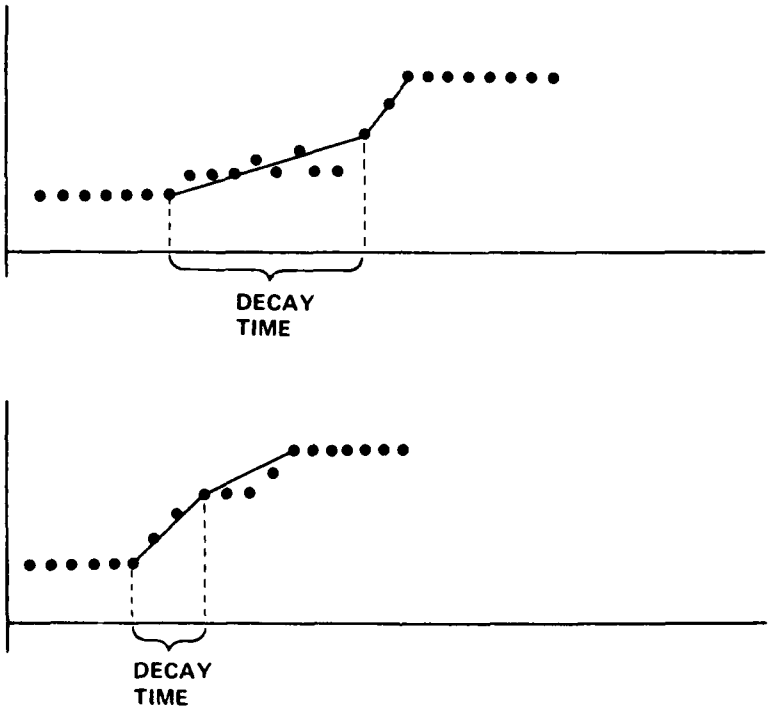


Figure 2 - Characterization of Transition Shape Using Decay Time.

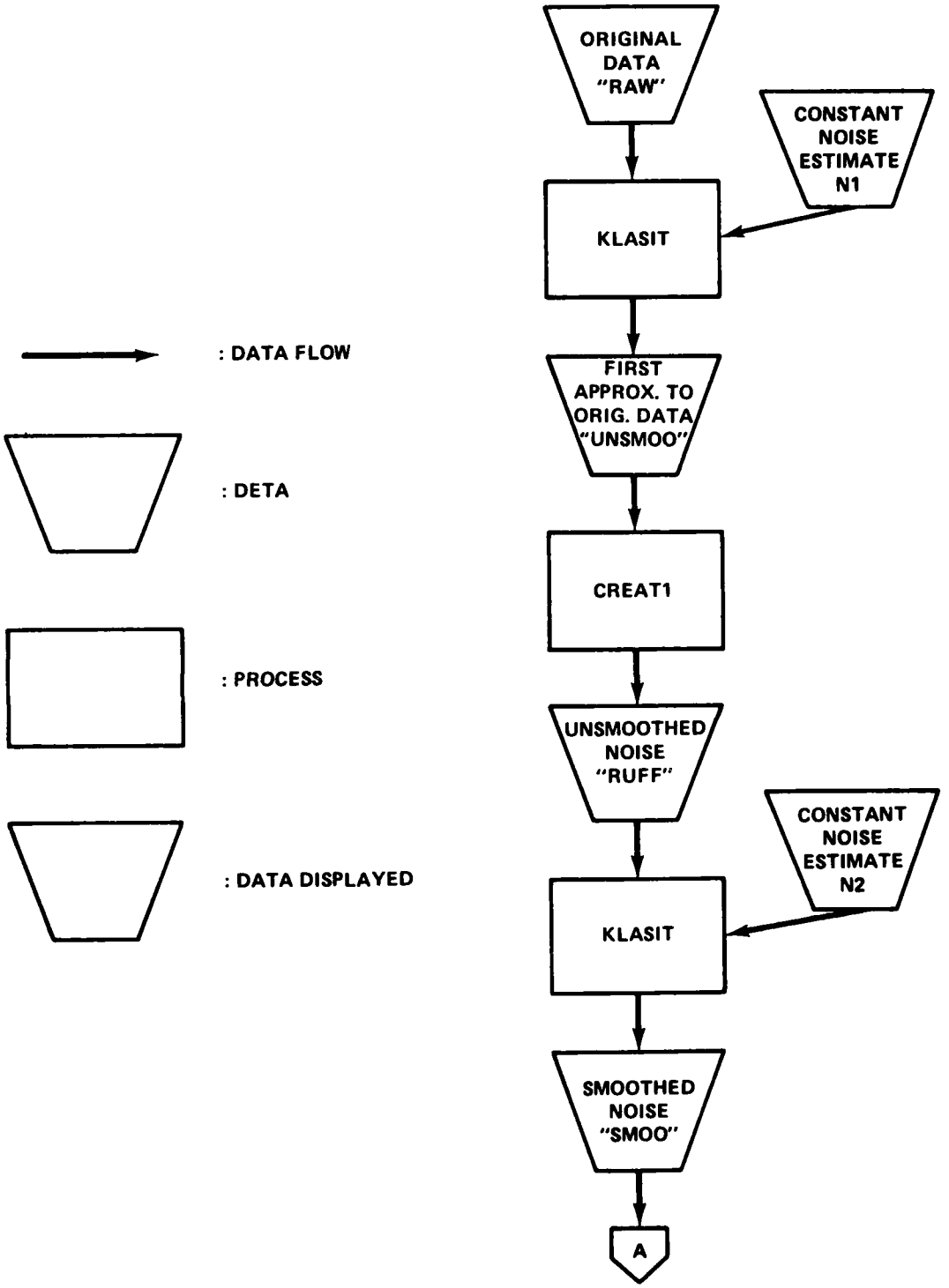


Figure 3 - Smoothing Procedure Schematic

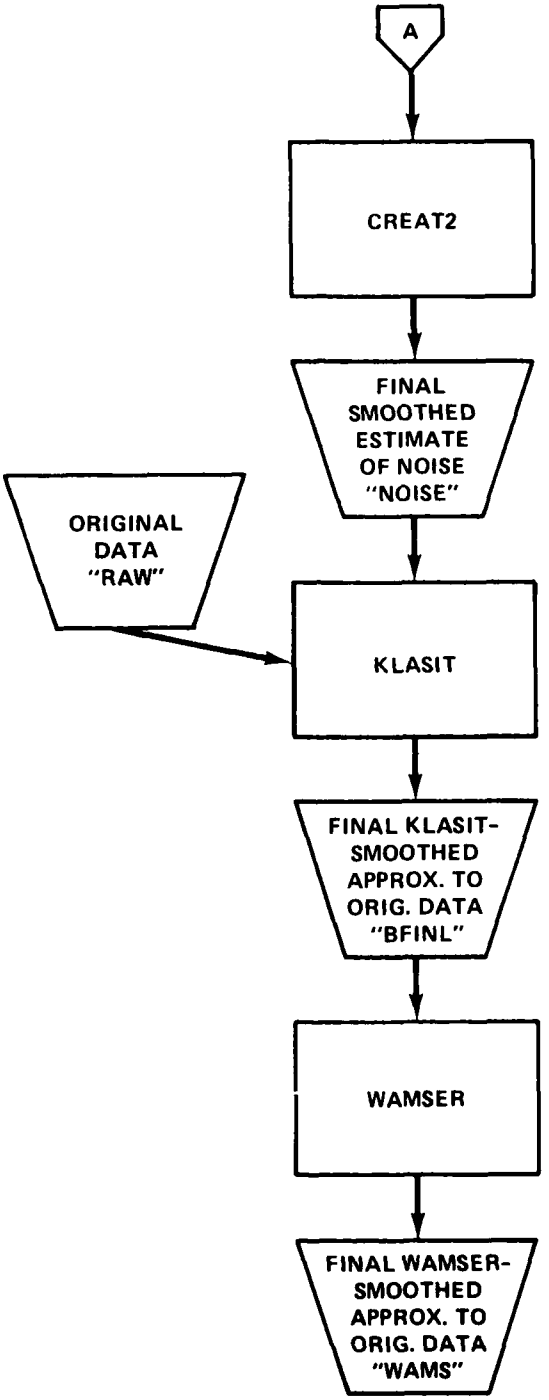


Figure 3 - Continued

first approximation of the raw data is obtained by assuming a constant noise level and employing the first-level smoothing procedure, KLASIT (cf. Figure 3). The KLASIT approximation is a piecewise linear estimate of the input signal consisting of the simple linear forms: rises, falls, and levels. The first approximation of the raw data is input to a noise computation procedure, CREAT1, which assigns a noise value for each point of each linear form. All level forms and all long rises and long falls, where "long" is determined by a user-controlled parameter, are assigned a constant zero noise. Short rises and falls ("short" determined by a separate user-controllable parameter) are assigned constant noise values equal to the full absolute change in amplitude over the rise or fall. Rises or falls of intermediate length are given constant noise levels equal to a percentage of the absolute change in amplitude; the percentage decreases linearly (from 100 percent to 0 percent) as the duration of the form increases from short to long. The output of the first noise computation, called the "unsmoothed noise," may be displyed on the same screen as the original data for visual comparison (Figure 4).

The unsmoothed noise itself is also input to KLASIT to be smoothed, assuming a constant noise-within-the-noise estimate. The resulting sequence of linear forms approximating the unsmoothed noise is converted to a digital signal by subroutine CREAT2 and divided by two to account for positive and negative values. The resulting signal is used as the "final smoothed estimate of the noise" contained in the original data (Figure 5). The smoothed noise may be displayed along with the unsmoothed noise by a separate user command.

The final application of KLASIT (cf. Figure 3) is to smooth the original data using the smoothed noise as the noise estimate input to the procedure. The "final KLASIT-smoothed approximation the original data" is displayed on the same screen as the original data for comparison (Figure 6). This result gives a piecewise linear first order approximation to the time-ordered intensity levels of the original signal with noise removed.

The finishing touches of the smoothing schema are performed by the WAMSER procedure on the final KLASIT-smoothed approximation. WAMSER refines the linear forms supplied by KLASIT to better approximate the local shape of the original data curve. The result, the "WAMSER-smoothed approximation to the original data," may be displayed in the center of the screen so that it can be visually compared with both the KLASIT-smoothed approximation and the original

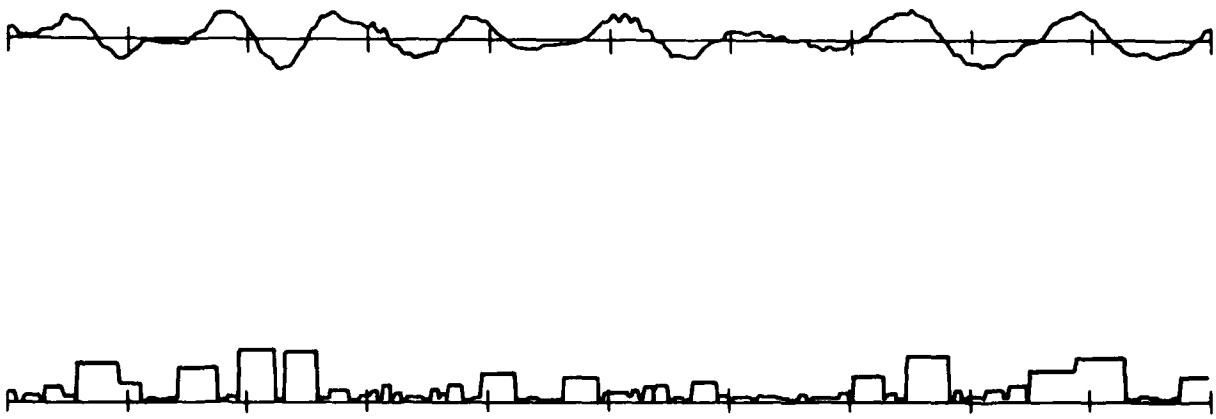


Figure 4 - Original Data (Top) and Unsmoothed Noise (Bottom).

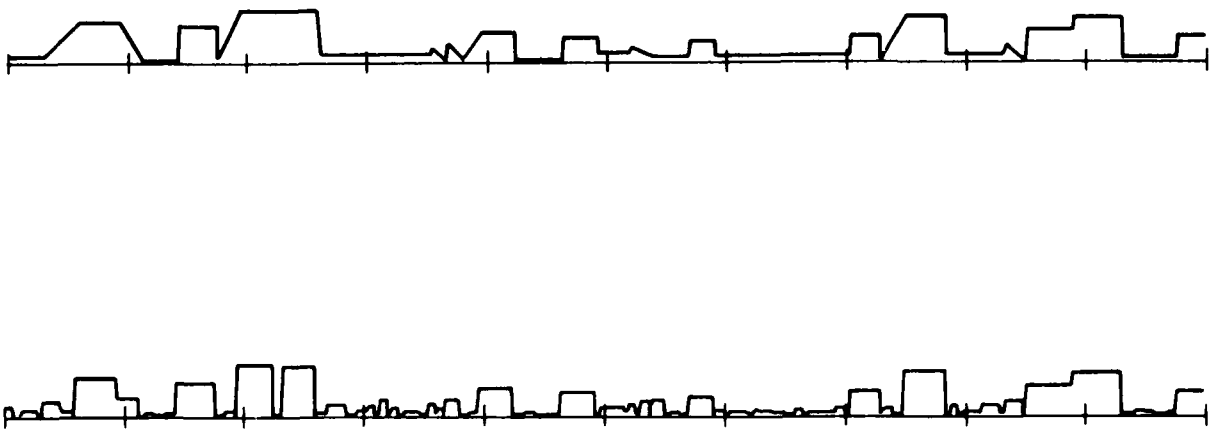


Figure 5 - Final Smoothed Estimate of Noise (Top). Unsmoothed Noise (Bottom).

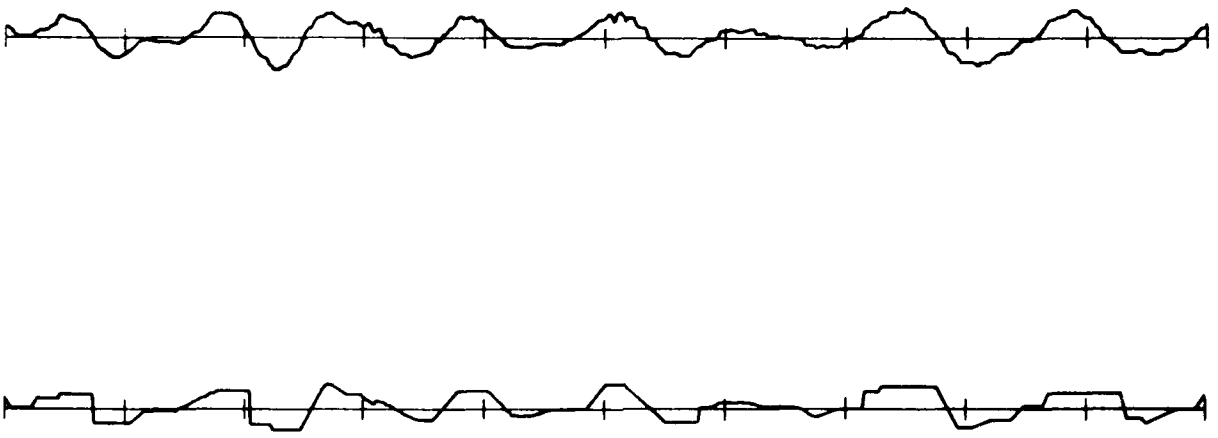


Figure 6 - Original Data (Top). KLASIT-Smoothed Approximation to Original Data (Bottom).

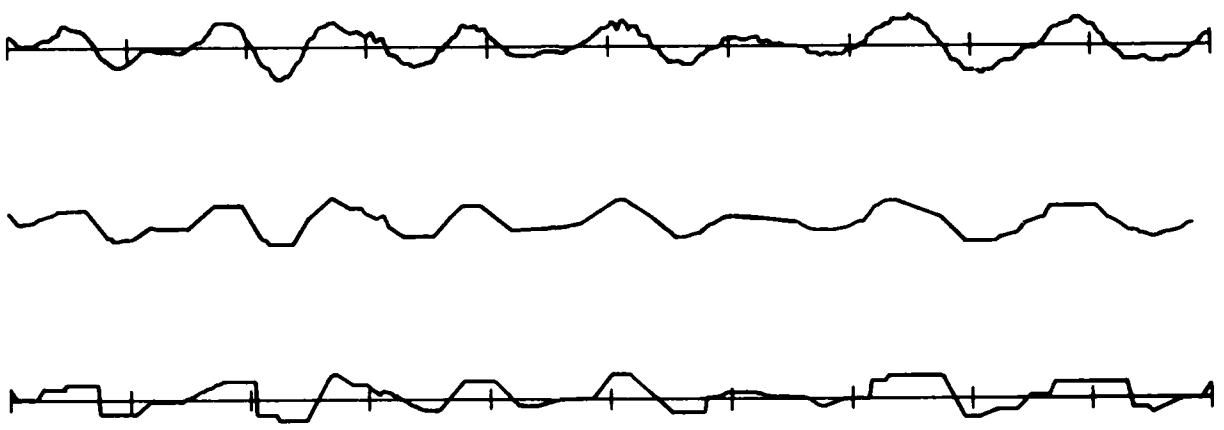


Figure 7 - Original Data (Top). WAMSER - Smoothed Approximation to Original Data (Middle). KLASIT - Smoothed Approximation to Original Data (Bottom).

data on the same display (Figure 7). The user may also superimpose either the WAMSER approximation or the KLASIT approximation on the original data to further evaluate results. The commands needed to direct the program to any of these results are explained in detail in the section on Functions.

Smoothing Parameters. The smoothing module, since it is embedded in the VU program, operates within some of the natural bounds of VU. Raw data are displayed in buffers of length PTSDPY by the VU program and all stages of the smoothing process operate on and display buffers of the same size. Other VU parameters, such as the file name containing the raw data, the sampling rate, and the starting point within the input file, are implicit to the smoothing process.

Certain parameters are peculiar to the smoothing routines. In KLASIT, these parameters include a noise estimate, a threshold for significant change in amplitude, a minimum duration for level forms, and a minimum duration for peaks. The effects of these parameters are explained in the section on Functions and in the subroutine descriptions.[1] These parameters have a significant effect on the KLASIT smoothing results and are alterable by special user commands also detailed in the Functions section. Furthermore, since KLASIT is applied several times in the smoothing process (Figure 3) with different input and for different purposes, a separate parameter set is needed for each KLASIT application. The VU program is structured to use a separate default set of parameter values for each application of KLASIT. Furthermore, default sets assume different values for frequency track data and raw signal data. The default values are included in the display of each KLASIT smoothing result and may be changed using the change parameter functions/commands.

The WAMSER smoothing process employs three parameters: a minimum duration for levels and an amplitude change threshold, both used to determine transition endpoints, and a percentage parameter used to determine decay time. These parameters are also explained in detail in the next section on Functions.

Functions

The keyboard commands for operating program VU are described in the following pages in a standard format which features the command name and its form followed by a description of the results obtained by using the command. Examples and sample displays are included where appropriate, as well as discussions of related parameters whenever user controlled parameters can affect results.

Function: DATA DISPLAY

Command: RUN DPO: VU <CR> (<CR> is the "RETURN" key)

Description:

This function loads the VU program into memory and directs the operating RT-11 system to transfer control to the program. The command is typed when the operating system is in the keyboard monitor mode. The program first asks the user to enter the name of the data file to be observed. If the specified file cannot be found, the program generates an error message and then asks the user to reenter the file name. The error message is generated vocally by VOTRAX, an electronic voice synthesizer. When the specified file has been successfully looked up, the program asks the user to enter the number of multiplexed data channels embedded in the file. If the number of channels is greater than one, the program asks the user to enter the channel to be observed and then displays the first 512 points of the input data for the specified channel. The program then enters the command mode and the user has control of program execution. Figure 8 shows a sample user/program dialog at the terminal with the program responses underlined, and Figure 9 shows a sample display.

```
.LOAD DP
.RUN DPO:VU
ENTER DATA FILE:  *DEV:FILNAM.EXT
ENTER # CHANLS:  12
STATE CHNL TO BE OBSERVED:  3
LAST SAMPLE IS #  191744
```

Figure 8 - Example of Initial Terminal Dialog

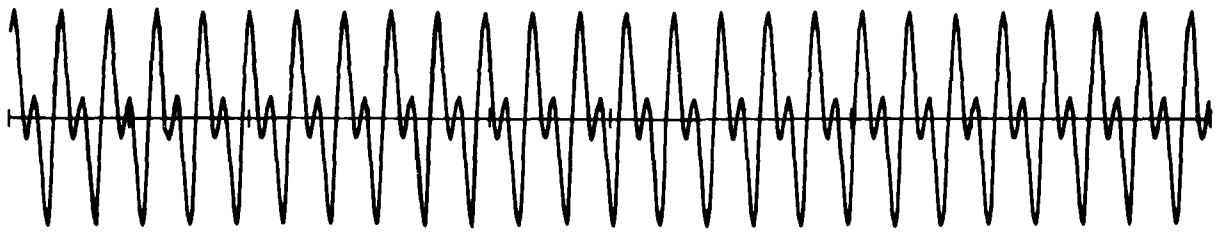


Figure 9 - Initial Display

Function: REENTER PROGRAM

Command: R<CR>

Description:

This function enables the user to reenter the program and to specify a new input stream. The current data stream file is closed and freed. This command executes the same procedure as the RUN DP0:VU<CR> command except that the number of points displayed remains as the user last specified.

Function: EXIT

Command: E<CR>

Description:

This function exits the program. The data stream file is closed and freed, and the system prints

TYPE<CR> TO EXIT

When the user types a carriage return, the screen is erased and control is returned to the RT-11 keyboard monitor.

Function: TAPE TRANSFER

Command: T1<CR>

Description:

This function creates a user defined data file from a 910 magnetic tape using the DR11C[4] (SDS-910 to PDP 11/45 interface.) The 910 tape must be in the First Pass format. First Pass is the analog-to-digital conversion program for the SDS-910 computer system. The First Pass format contains digitized multiplexed channel data with the first two words in each record as identification, i.e., run identification and number of multiplexed channels digitized. The DR11C interface drivers provide the capability of manipulating the 910 tape using parameters that specify forward or backward file skip, forward or backward record skip, and forward data point skip.

The program asks the user to specify the file to be created and to specify the tape manipulation parameters. First, the user is asked to specify the name of the file to be created. Note, the file name is an RT11 standard specification and must be followed by an "=" sign since the file is being created. If the file cannot be created as specified, an error message is typed and the user is asked to reenter the file name. Once the file name has been successfully specified, the program asks the user to enter the number of multiplexed channels in the data. The number of channels is accepted with an I4 format but this number must not exceed 24.

The program now asks the user to specify the tape file, tape records, and data points to skip prior to data transfer. Files and records can be skipped either forward or backward. A negative number indicates a backward skip. Each of these skip parameters is accepted with an I4 format.

The interface handler now moves the tape to the specified point and transfers the data. The run ID and the number of channels are read from the tape. The run ID is printed on the terminal for the user's information. Figure 10 shows a sample of the user/program dialog. If the number of channels specified by the user does not agree with the number of channels on the tape, the user may have made a typing error or the tape may not be in the desired position. The program gives the user the option of aborting or continuing. The program responds

The # OF CHNLS IS __ and not __. CONTINUE?

The user types N<CR> if he wishes to abort and Y(CR) if he wishes to continue.

```
T1

ENTER NAME OF NEW FILE, FOLLOWED BY = SIGN:  *DP2:SINE1K.DAT=

ENTER # OF CHANNELS:  1

ENTER # OF TAPE FILES TO SKIP:  0

ENTER # OF RECORDS TO SKIP:  0

ENTER # OF DATA PTS TO SKIP:  0

RUNID=  2
ERROR CONDITION; IERR=  3
END OF FILE; 188 PTS WERE SENT

SIZE OF FILE=  3124 BLOCKS
```

Figure 10 - Tape Transfer User/Computer Dialog

Data are read from tape until an error condition occurs. Two of these errors are related to the PDP data file. If a file write error occurs, the program responds

ERROR IN DISK WRITE, CODE = _____

and the program stops. If the data fill the allocated file space, data transfer is terminated, the file is closed, and control is returned to the VU executive.

Tape related errors are divided into four categories: beginning of tape, end of tape, end of file, and parity errors. The parity errors are the only true tape errors. Beginning of tape, end of tape, and end of file are really indicators that terminate data transmission. When one of these tape conditions occurs, an error message indicating the condition and the number of points transferred prior to the condition is printed at the terminal. Data transmission is terminated, the file is closed, and control is returned to the VU executive. The new data file is not displayed. The user must use the reenter (R) function to process the new data.

Function: HORIZONTAL DISPLAY SCALING

Command: S1 = XXXX <CR?

Description:

The number of data points displayed (horizontal scale) on the graphics console becomes XXXX. XXXX must be greater than 0 and less than 1025. The program starts by displaying 512 points as default.

Parameter: PTSDPY is the number of data points displayed across the graphics console.

Function: VERTICAL DISPLAY SCALING

Command: S<CR>

Description:

The user has control of the amplitude (vertical) scale. The command will modify the current scale by the desired scale factor. After the user enters the command, the program responds

ENTER REAL SCALE FACTOR:

The user then enters the desired scale factor in F10.2 format followed by a carriage return. The display will be modified accordingly.

Parameter:

ZFACT is the scale factor.

ZR is the raw data scale modifier by which all data points are modified before being displayed. ZR is initialized to 20 and when modified becomes ZR=ZR/ZFACT.

Function: MOVE TO END OF DATA

Command: F<CR>

Description:

This function displays the last PTSDPY points of the data stream. PTSDPY is initially 512 and can be set with an S1 command. The index relative to the beginning of the data stream for the specified channel is displayed for the first point displayed on the screen. If a transform (i.e., power spectrum, FFT, etc.) of the previously displayed data was displayed, then the transform of the current data display will be completed and displayed beneath the data.

Parameters:

ZNUM is the index, relative to the beginning of the data stream, of the first data point displayed for the specified channel. This function modifies ZNUM by

$$ZNUM = ZLAST - PTSDPY + 1$$

where ZLAST is the index to the last point in the data stream for the specified channel.

Function: MOVE FORWARD RELATIVELY

Command: F1 = XXXX <CR>

Description:

This function makes the first displayed point the XXXXth data point beyond the current first displayed point. XXXX is entered in F10.0 format. The index of the firist displayed point and any transforms are displayed as described for the "F" command.

Parameter:

XNUM is as described in the "F" command, except that this function modifies ZNUM by

$$ZNUM = ZNUM + XXXX$$

Function: MOVE FORWARD RELATIVELY AND SET MOVE DEFAULT

Command: F2 = XXXX <CR>

Description:

This function will perform an F1 = XXXX command and set the default move command to F1 - XXXX.

BLNK is the default command and is set BLNK = "F"

ZBLNK is the default move modifier and is set ZBLNK = XXXX

Parameters:

ZNUM is as described in the "F1 = XXXX" command.

Function: MOVE TO THE BEGINNING OF DATA

Command: B<CR>

Description:

This function displays the first PTSDPY points of the data stream. Beyond the actual points displayed, this function performs like the "F" command.

Parameters:

ZNUM is as described in the "F" command, except that this command sets

ZNUM = 1

Function: MOVE BACKWARD RELATIVELY

Command: B1 - XXXX<CR>

Description:

This function makes the first displayed point the XXXX point before the currently displayed first point. Beyond the actual points displayed, this function performs like the "F1 = XXXX" command.

Parameter:

ZNUM is as described in the "F" command, except that this command modifies ZNUM by

ZNUM = ZNUM-XXXX

If ZNUM becomes less than one, then ZNUM=1.

Function: MOVE BACKWARD RELATIVELY AND SET MOVE DEFAULT

Command: B2 = XXXX<CR>

Description:

This fuction performs a "B1 = XXXX" command and sets the default move command to "B1 = XXXX."

Parameters:

ZNUM is as described in the B1 - XXXX command.

ZBLNK and BLNK are as described in the "F2 = XXXX" command except that

BLNK = "B"

Function: MOVE RELATIVELY BY DEFAULT

Command: <SPACE><CR> or <CR>

Description:

This function executes the last relative default move.

Parameters:

BLNK contains the command, "B" if backward, or "F" if forward

ZBLNK contains the number of points to move

ZNUM is as previously described in "F" except that if BLNK contains "B", then

ZNUM = ZNUM - ZBLNK

and if BLNK contains "F", then

ZNUM = ZNUM + ZBLNK

Function: JUMP TO ABSOLUTE DATA POINT

Command: J1 = XXXX <CR>

Description:

This function makes the first point of the displayed data the XXXXth data point in the data stream for the specified channel. XXXX is entered with format F10.0.

Parameter:

ZNUM is as described in the "F" command except that it is set by

ZNUM = XXXX

Function: LOCATE BEGINNING EVENT AND MOVE DISPLAY

Command: L<CR>

Description:

This function is used to determine the beginning and end of an event in the input stream. The event beginning is found by searching the entire input stream of the specified channel for the first point whose absolute value is at least the value of parameter AMPBEG. The display is modified so that PTSDPY points of the event are shown. This function continues the search for the event end by locating the last data stream point whose value exceeds the absolute value of AMPBEG. The number of data stream points between the first and last points is the event duration. The duration is converted to seconds, assuming the data stream was digitized at 18,000 samples per second. This duration value is printed at the terminal.

Parameter:

AMPBEG is the absolute value that must be equalled or exceeded before a data stream region may be called an event.

Function: CHANGE NUMBER OF CHANNELS

Command: C<CR>

Description:

This function allows the user to change the number of channels specification associated with the input data stream. This option is currently used to decimate the data and thus allows the user to observe every nth point of the raw data, depending on how many channels the user specifies. After the user enters the command, the program responds

ENTER # CHANLS:

The user then enters the desired number of channels specification. If the user wants to display every other point, then the channel specification will be twice the actual number of channels multiplexed in the data stream. When the user enters the new channel specification, the program responds (if the number of channels specified is greater than one)

STATE CHNL TO BE OBSERVED:

The user enters in I4 format the channel to be observed. Note: The user can use this command to change the channel to be observed.

Parameters:

NCHAN is an integer indicating the number of multiplexed channels in the input data stream.

OCHAN is an integer that indicates the channel being observed and processed according to user commands.

Function: FFT DISPLAY

Command: T<CR>

Description:

This function operates like a toggle switch in that the Fourier transform display can be turned on and off. If the transform is not displayed (off) and the user enters the T command, a complex fast Fourier transform[5] of the displayed input stream will be calculated and displayed (on). If the transform is displayed (on) and the user types "T", the Fourier transform will be erased (off). If a move function is requested and the transform is displayed, the original transform will be erased along with the data display and the transform of the newly displayed input stream will be computed and displayed.

The number of points used in computing the fast Fourier transform is the number of points displayed (PTSDPY). The maximum allowable size is 1024. The transform is defined via the following operation:

$$A_r = \sum_{K=0}^{N-1} X_k \exp(-2\, jrK/N),\ r=0,\ldots,\ N-1$$

where

A_r is the r^{th} coefficient of the Fourier transform and X_k is the K^{th} complex point of a time series which consists of N samples. The real part, which consists of PTSDPY/2 points, is displayed near the middle of the screen below the data stream display. The imaginary part, also PTSDPY/2 points, is displayed in the lower quarter of the graphics screen. Figure 11 shows an example of the FFT display.

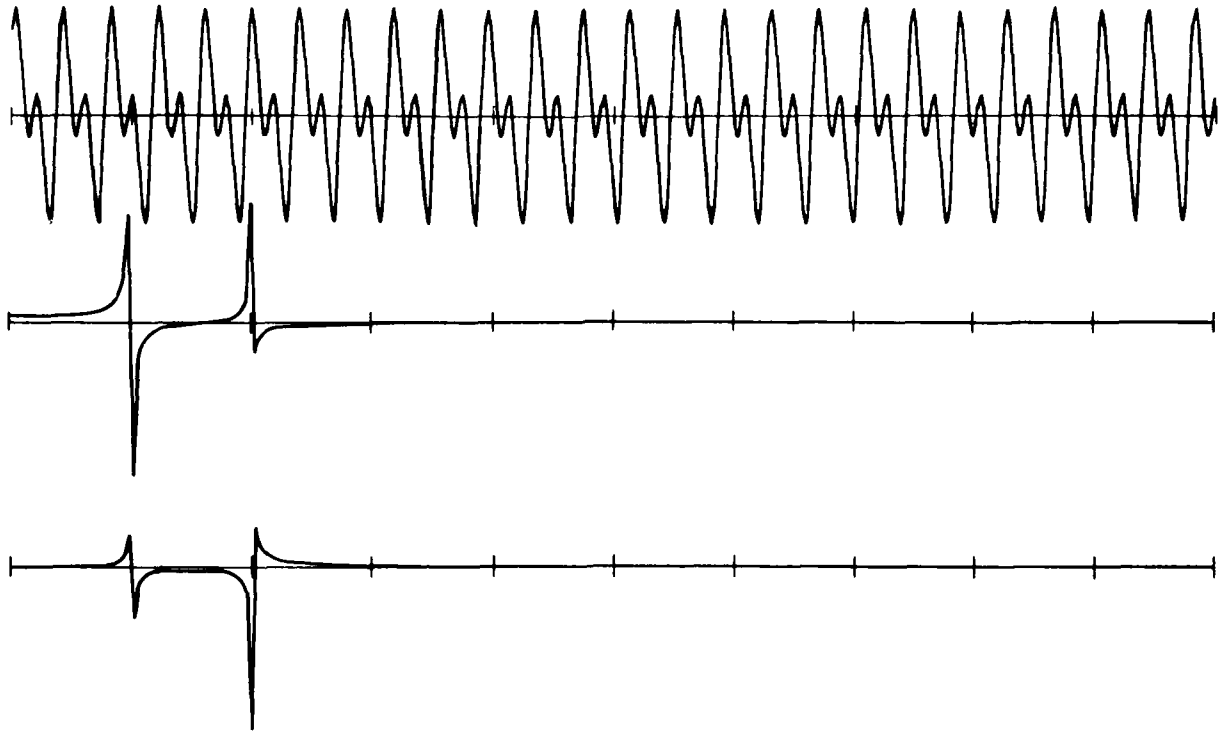


Figure 11 - FFT Display

Function: POWER AND PHASE SPECTRA

Command: P<CR>

Description:

This function calculates and displays either the power spectrum or the phase spectrum, depending on the value of a user alterable parameter. As with the FFT display, this function operates like a toggle switch in that, if the spectrum is not displayed and the user enters the P command, the appropriate spectrum is calculated and displayed. If the spectrum is displayed when the P command is typed, the displayed spectrum will be erased.

The power spectrum is generated if the units digit (base 10) of parameter LOGPOW is either zero or one. The power spectrum is calculated by performing a sum of the squares of the Fourier coefficients. Thus, the fast Fourier transform must be generated before the power spectrum is calculated. The spectrum is displayed with a linear scale if the units digit of LOGPOW is zero and with a logarithmic scale if the units digit is one.

The phase spectrum is generated if the units digit (base 10) of parameter LOGPOW is two. The phase spectrum is computed from the real cosines and the imaginary sines of the Fast Fourier transform.

A window option is available to smooth the effects of a sharp transition at the beginning and end of the data buffer. The window is applied to the data before the Fast Fourier transform is generated. Currently a Kaisser-Bessel window is used. The tens digit (base 10) of LOGPOW selects the Kaiser-Bessel window option if it is not equal to 1. If the tens digit of LOGPOW is 1, then a rectangular window (unmodified data) is used to generate the Fast Fourier transform.

The spectra are displayed on the lower half of the screen. Figure 12 shows an example of a power spectrum display. A pointer is also displayed. This pointer is a vertical line with the lower end touching the spectrum display. The horizontal position of the pointer is controlled by potentiometer knob associated with channel 0. Note: No phone plug is to be inserted in channel 0 because the phone plug disables the output voltage of the potentiometer. The frequency and amplitude of the spectrum point at the lower end of the pointer are displayed on the screen. The frequency is calculated assuming the data stream was digitized with a sampling rate of 18,000 samples per second.

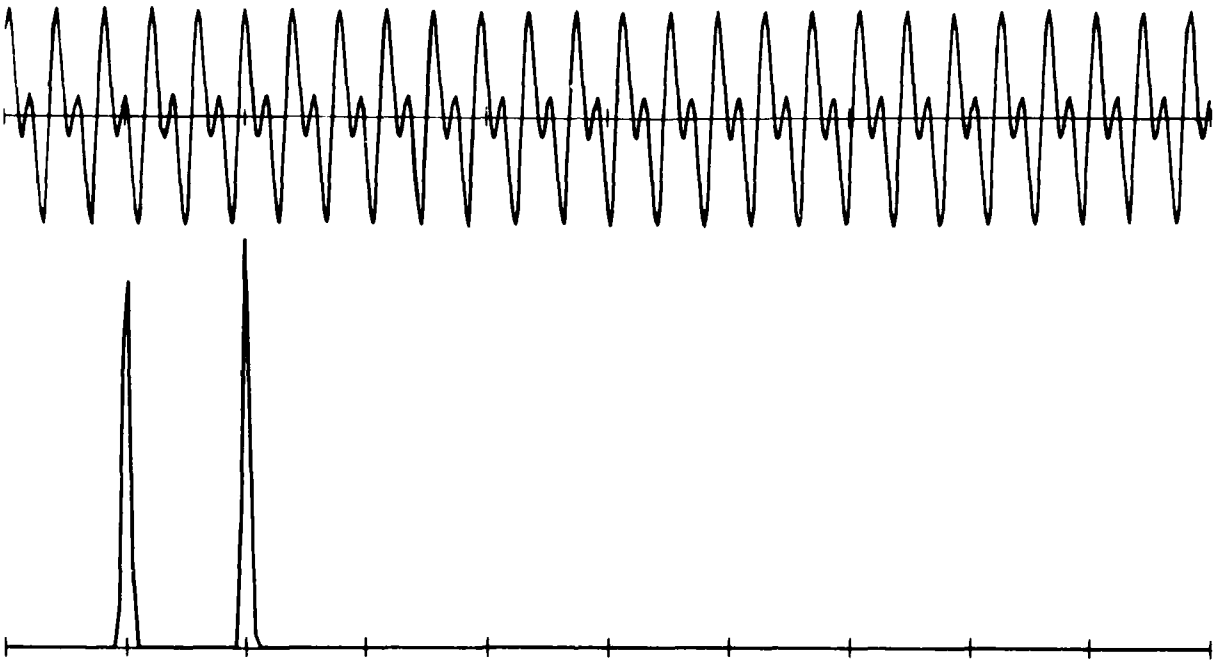


Figure 12 - Power Spectrum Display

Parameters:

LOGPOW controls the spectrum type and the windowing. The units digit controls the type of spectrum generated.

0 - Power spectrum, linear scale

1 - Power spectrum, logarithmic scale

2 - Phase spectrum

The ten digit controls the windowing.

1 - Rectangular window

other - Kaiser-Bessel window

As an example, LOGPOW = 21 means that a power spectrum displayed with a logarithmic scale will be generated and a Kaiser-Bessel window will be used on the data in calculating the Fast Fourier transform. LOGPOW can be set by the user in the change parameter function (X). LOG is initially equal to zero.

ZPSCAL scales the spectrum display. All spectrum points are multiplied by ZPSCAL before being displayed. ZPSCAL can also be set by the user in the change parameter function (X). Initially it is read from an existing disk file (DP0:OPARAM.DAT) of defaults. The user can change this parameter and save the change as the new default.

DCHEK(POW) indicates the state of the spectrum display. The display is either on, off, or erased.

Function: POWER AND PHASE SPECTRA WITH PEAKPICKING

Command: P5<CR>

Description:

This function calculates and displays the spectra as described for the "P" function. This function also selects peaks in the spectrum and displays the results as blinking X's on top of the peaks in the spectrum display. If peaks are already being displayed when the user invokes this function, then only the peaks are erased. If the "P" function is invoked when a spectrum and peaks are being displayed, then both will be erased. Several parameters control the definition of a peak. The user can set these with the change parameter function (X).

Figure 13 shows an example of a power spectrum with peaks selected.

Parameters:

PDRAW indicates whether the peaks are on, off, or erased.

The follow parameters can be set by the user with the change parameter function (X). Initially, the default values are read from an existing disk file (DP0:OPRAM.DAT). The user can also change these defaults.

AMPMIN specifies the minimum amplitude a spectral point must exceed before it can be considered a peak. AMPMIN is given as a percentage of the average value of the spectrum.

BANK determines what is called a significant change. If the change is not significant, then the spectral point will not be considered a peak. The significant change must be greater than

$$\frac{\text{(value at this index)} + \text{(Value at last base)}}{2} * \text{BAND}$$

PERHI is the percentage of the maximum (peak value) required for the start or end of a peak.

PERLO is the percentage of the maximum required before a slope comparison with FACTOR can be made.

FACTOR is the minimum slope required for the start or end of a peak if (PERLO* maximum) is exceeded.

BWIDTH is the maximum width of a single peak in hertz. Any peak exceeding this width is resolved into two peaks.

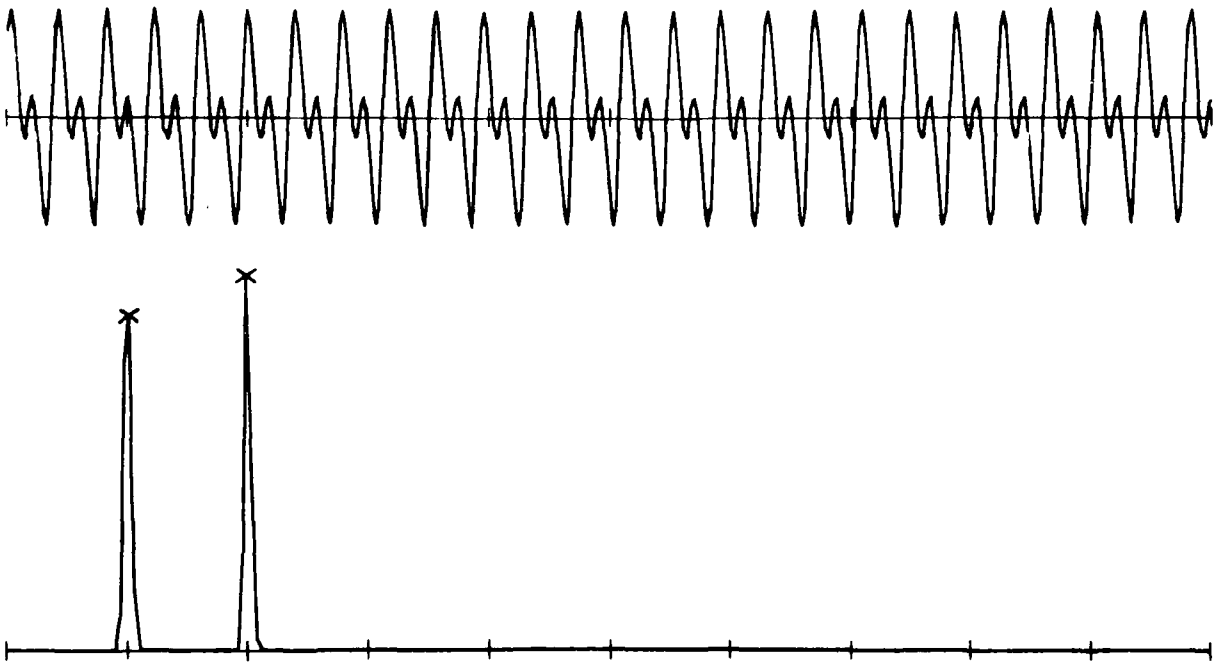


Figure 13 - Power Spectrum with Peaks Selected

SFREQ is the frequency at which peakpicking will begin.

FMERGE specifies the maximum frequency separation in hertz betwen two peaks before the peaks are candidates for merging.

HMERGE is the height threshold which specifies when two peaks can be merged. HMERGE is the percent below the high peak that the low peak must be before merging is possible.

Function: SPECTRAL SLICES

Command: Z<CR>

Description:

This function performs peakpicking on a user specified number of consecutive power spectra and places the results in a file (DP0:MORPK2.DAT) to be examined with program SELECT which tracks the peaks. The peakpicking is done as described in the "P5" function. The user specifies the point in the data stream at which the spectra will start, the number of data points between spectra, and the number of power spectra for peakpicking. When the user types "Z<CR>", VU responds with

FIRST SAMPLE =

The user now enters the index to the data point at which the first spectrum begins. If zero is entered, the spectra generation begins with the first point displayed. VU responds

DURATION OF SHIFT =

The user enters the number of data points between the beginnings of consecutive power spectra. VU then responds

OF WINDOWS =

The user enters the number of consecutive power spectra from which peakpicking results will be generated. The number of windows should be less than 1024, because 1024 is the number of records allocated to store the peakpicking results. Currently, a maximum of 10 peaks will be selected for each window. The three user specified parameters are entered in F9.0 format. When peakpicking is finished, the program responds

**********PEAKPICKING COMPLETED**********

and the user can enter a new command.

Parameters:

The parameters are described under the "P5" function parameters.

Function: SAVE DISPLAY

Command: T5<CR>

Description:

This function stores up to five display buffers on disk. These stored buffers can be read and displayed by a separate program called MANYVU. The first time T5 is executed, the display buffer is written on File DP0: TEST00.DPY. The second time T5 is executed, the display buffers are written on files DP0:TEST02.DPY, DP0:TEST03.DPY, DP0:TEST04.DPY.

Function: PRINT ORIGINAL DATA

Command: Q7 = XXXX <CR>

Description:

This function prints out XXXX values of the data stream. The first value printed is the first displayed point. XXXX is entered in F10.0 format. The maximum number of values printed is the number of points displayed (PTSDPY).

Function: GENERATE HARDCOPY

Command: G<CR>

Description:

This function generates a hardcopy of the graphic data (points and vectors) displayed on the graphics screen. The hardcopy is produced on an X-Y plotter. Alphanumeric information is not reproduced.

Function: ENTER/EXIT SMOOTHING (KLASIT) MODE

Command: K

Description: The "K" command controls entering or leaving the smoothing mode of program VU. Upon entering, the FFT and power spectrum displays are erased and the user is requested to enter a track number. A positive track number refers to a frequency track stored on disk file DP0:TRAK2.DAT, created by program SELECT, and identifies the particular frequency versus time data to be considered the original data to be smoothed. If a frequency track is chosen by the user, the current data display is erased and the selected track is read in and displayed. The program then waits for further KLASIT level commands. When tracks are being smoothed and the "K" command is entered to exit the smoothing mode, all subpictures are erased and the original scale and parameters associated with the original data are restored to the values they had when smoothing was entered. Finally, the original data are reread and displayed.

A zero track number entry at the beginning of the smoothing mode indicates that the data currently being displayed are to be smoothed, and the program awaits further KLASIT-level commands. Upon exit from this mode, all displays but that of the original data are erased and the display of the original data is turned on.

The VU commands controlling movement of data scale, hard copy, and print, which have been explained in the preceding pages, are also valid in the smoothing module.

Parameters:

TRAKNUM - parameter requested and read from the keyboard; if positive, it identifies the frequency track to be smoothed; if zero, it signifies that the original data are to be smoothed.

Function: EXIT KLASIT

Command: E

Description:

The "E" command exits the KLASIT smoothing mode and performs the same functions that the "K" command does.

Function: DISPLAY ORIGINAL DATA ONLY

Command: D

Description:

The "D" command displays the original data and turns off or erases all other displays. (The parameter displays are erased and any other displays that are on are turned off.) The original data is that amount of data (limit = 1024, default = 512) stored in the original data program arrays by command options in program VU.

Function: DISPLAY ORIGINAL DATA AND UNSMOOTHED NOISE

Command: D1

Description:

The "D1" command displays the original data at the top of the screen, computes the unsmoothed noise from the original data, and displays the unsmoothed noise at the bottom of the screen. In addition, the parameters used to compute the unsmoothed noise are displayed with their current values. All other displays are turned off.

The computation of the unsmoothed noise begins with an approximation to the original data produced by subroutine KLASIT using the number one parameter set--in particular, using parameter N1 as a constant noise as the level threshold. The approximation generated by this KLASIT application is used to create the unsmoothed noise. A zero noise level is associated with all level forms and with any rise or fall of long duration (greater than the Q1 parameter). Short rises or falls (duration less than the Q0 parameter) are assigned a noise level equal to the absolute change in level. Rises or falls of intermediate duration are assigned a noise level equal to the absolute change in level multiplied by the fraction (Q1-duration)/(Q2-Q0).

Parameters:

N1 - constant noise level estimate used by KLASIT to approximate original data; a constant threshold for levels

L1 - minimum duration threshold for level form

I1 - threshold for significant change in level

P1 - a duration threshold below which peaks are considered too narrow to be significant in KLASIT

Q0 - a lower duration threshold which identifies short rises or falls

Q1 - an upper duration threshold which identifies long rises or falls

Function: DISPLAY UNSMOOTHED NOISE AND SMOOTHED NOISE

Command: D2

Description:

The "D2" command displays the unsmoothed noise at the bottom of the screen, the smoothed noise at the top of the screen, and the parameter values used to obtain the smoothed noise. All other displays are turned off.

If the smoothed noise has not yet been computed for the current set of original data displayed, the "D2" command also initiates the computation. In fact, if the unsmoothed noise has not also been computed, it will be computed first as if a "D1" command had been entered. The smoothed noise is obtained by applying the KLASIT procedure to the unsmoothed noise and, in so doing, uses parameter set two. The smoothed noise, therefore, is in the form of rise, fall, and levels approximations to the unsmoothed noise.

Parameters:

(Parameter set one applied to computation of the unsmoothed noise is described under command "D1")

N2 - a constant noise level estimate used by KLASIT to construct the smoothed approximation to the unsmoothed noise

L2 - minimum duration threshold for a level form

I2 - threshold for significant change in amplitude; signals start of rise or fall

P2 - duration threshold for narrow peaks--a true peak must be longer than P2 to be included in approximation

Function: DISPLAY ORIGINAL DATA AND FINAL SMOOTHED DATA

Command: D3

Description:

The "D3" command displays the original data at the top of the screen, the final smoothed data at the bottom of the screen, and the final set of parameter values used to obtain the final result. All other displays are turned off.

The "D3" command also initiates all computations needed to obtain the final smoothed data, whatever the state of the program. "D3" can be entered immediately after entering the KLASIT smoothing mode, in which case the following sequence of computations is made:

- The unsmoothed noise is obtained
- The smoothed noise is computed by applying KLASIT to the unsmoothed noise
- A smoothed noise array is computed from the KLASIT smoothed noise
- The final smoothed data are obtained by applying KLASIT to the original data using the smoothed noise array as the noise estimate.

If "D3" is entered after a "D2" command, only the last two computation steps are executed since the smoothed noise was produced by the "D2" command.

The smoothed noise array is obtained from the KLASIT smoothed noise waveforms by assigning half the amplitude of a level to all time points in a level, assigning half the amplitude to the endpoints of rise/fall forms, and performing a straight line interpolation between the half amplitudes of the endpoints. This smoothed noise estimate is input to KLASIT to compute the final smoothed waveform approximation to the original data.

Parameters:

(Parameter sets one and two are applied to computations of the unsmoothed noise and the smoothed noise waveforms are described under the "D1" and "D2" commands.)

L3 - minimum duration threshold for a level form

I3 - threshold for significant change in amplitude; signals start of rise/fall

P3 - minimum duration (width) of peaks described by rise/fall combinations

Function: SUPERIMPOSE (MERGE) BOTTOM DISPLAY OVER TOP DISPLAY

Command: M

Description:

The "M" command causes the bottom display to be superimposed on the top display for "D1", "D2", or "D3" screens. If the display is already overlaid, this command removes the overlaid picture. If any of the displays involved are not turned on or have not been created, the "M" command will have no effect.

Function: DISPLAY/ERASE WAMSER RESULTS

Command: W

Description:

The "W" command effects further smoothing of the final (KLASIT) smoothed waveform data by subroutine WAMSER and displays the result in the center of the screen. If the WAMSER display is already on, this command erases it. The "W" command may be entered at any time after the "D" command, and the necessary KLASIT stage smoothing will be performed and the final "D3" type KLASIT display created. The KLASIT smoothed data consist of a series of rise, fall, and/or level waveforms defined by type, amplitude at end of form, and duration. Subroutine WAMSER further smooths these data by taking each rise or fall form and comparing it to the original data to see how long it takes to achieve 40 percent of its amplitude change. This time is called a decay time and is added to the waveform data. The decay time allows each rise or fall segment to be approximated by two line segments which better represent the inflection of the curve for the given time interval. WAMSER further refines the end points of the rise/fall transitions.

Parameters:

TEN - percentage factor of the total change in amplitude which determines decay time for a transition

BAND - minimum amplitude change for transition start point in a "level" form if the RUN parameter is unsatisfied

RUN - minimum length of level before start of transition

Function: DISPLAY WAMSER RESULTS AND SUPERIMPOSE/ERASE ON RAW DATA

Command: W1

Description:

The "W1" command performs the same function as the "W" command with the added feature of superimposing the WAMSER results on the original data. If the display is already in the "W" command state, "W1" simply displays the overlay. If the "W1" overlay is already on, another "W1" erases the overlay only.

Function: DISPLAY WAMSER RESULTS AND SUPERIMPOSE ON SMOOTHED RESULT

Command: W2

Description:

The "W2" command performs the same function as the "W" command with the added feature of superimpoing the WAMSER results on the smoothed results at the bottom of the screen. If the "W2" display is already on, the overlay only is erased. If the "W" display is on, "W2" causes only the overlay to be displayed.

Function: CHANGE SMOOTHING PARAMETER VALUE

Command: Command format must be

xy = z

where x = N, L, I, P, or Q

y = 0, 1, 2, or 3

z = new parameter value

Description:

A change parameter command of the form xy = z changes the value of the parameter xy; e.g., "L1 = 10" changes the value of the L1 parameter used in "D1" command operations to ten. Only those parameters used in "D" level commands ("D1", "D2", or "D3") can be changed; invalid commands are rejected.

The procedure for all parameter changes is the same. The current display and program state are determined, and the proper parameter value is changed. If the parameter is currently being displayed, the new value is displayed immediately. All displays affected by the parameter change are erased and the current display and program state are regenerated using the new parameter value.

Function: ENTER/EXIT CHANGE PARAMETER MODE

Command: X<CR>

Description:

This command has two functions. First, if VU is not in the change parameter mode, the "X" will put it in this mode, and then a new set of commands is accepted. These new commands specify the parameter to be changed and the new value of the parameter. When the change parameter mode is entered, the display is erased and each parameter with its current value is displayed. The command associated with each parameter is shown enclosed in parenthesis next to the parameter. Figure 14 shows an example of this display. When the user changes a parameter, the new value is shown on the screen. Each change command is described as a function. Two commands not shown in the figure involve saving the current parameters as default and restoring parameter values to the last saved default values.

```
PEAKPK PARAMETERS:                    POWER SPECTRUM SCALE FACTOR:
  AMPMIN (A1) =       1                     ZPSCAL (Z1) = 5,000
  BAND   (B1) =   1.000                     LOGPOW (L1) = 1
  PERHI  (P1) =   0.910
  PERLO  (P2) =   0.800
  FACTOR (F1) =   2.300
  FMERGE (F3) = 150.000               STARTING AMPLITUDE:
  HMERGE (H1) =   1.000                AMPBEG (A2) = 325
  BWIDTH (B1) = 400.000
  SFREQ  (S1) =       1
```

Figure 14 - Change Parameter Display

These default commands are X2 and X1 and the default values are stored on a file called DP0:OPARAM.DAT.

The second function of the "X" command is to exit the change parameter. The displayed parameters are erased. The same input data stream segment is displayed as was displayed when the change parameter function was entered. No other display (e.g., power spectrum) is regenerated.

Function: CHANGE PARAMETER VALUES TO DEFAULT

Command: X1<CR>

Description:

This function is invoked only when VU is in the change parameter mode. All parameters are restored to default values. The default values are read from disk file DP0:OPARAM.DAT.

Function: CHANGE PARAMETER DEFAULT VALUES

Command: X2<CR>

Description:

This function is invoked only when VU is in the change parameter mode. The current parameter values are stored as the new default values on disk file DP0:OPARAM.DAT.

Function: CHANGE PARAMETER AMPMIN

Command: A1 = XXX<CR>

Description:

This function is invoked only when VU is in the change parameter mode. This function changes the value of the AMPMIN parameter to XXX. XXX should be between zero and 100. AMPMIN is a peakpicking parameter and is described in the P5 function.

Function: CHANGE PARAMETER AMPBEG

Command: A2 = XXXX<CR>

Description:

This function is invoked only when VU is in the change parameter mode. This function changes the value of AMPBEG to XXXX. AMPMIN is a 16-bit integer used and described in the locate (L) function.

Function: CHANGE PARAMETER BAND

Command: B1 = XXXX<CR>

Description:

This function is invoked only when VU is in the change parameter mode. This function changes the value of BAND to XXXX. BAND is a peakpicking parameter and is described in the P5 function.

Function: CHANGE PARAMETER BWIDTH

Command: B2 = XXXX<CR>

Description:

This function is invoked only when VU is in the change parameter mode. This function changes the value of BWIDTH to XXXX. BWIDTH is a peakpicking parameter and is described in the P5 function.

Function: CHANGE PARAMETER FACTOR

Command: F1 = XXXX<CR>

Description:

This function is invoked only when VU is in the change parameter mode. This function changes the value of FACTOR to XXXX. FACTOR is a peakpicking parameter and is described in the P5 function description.

Function: CHANGE PARAMETER FMERGE

Command: F3 = XXXX<CR>

Description:

This function is invoked only when VU is in the change parameter mode. This function changes the value of FMERGE to XXXX. FMERGE is a peakpicking parameter and is described in the P5 function.

Function: CHANGE PARAMETER HMERGE

Command: H1 = XXXX<CR>

Description:

This function is invoked only when VU is in the change parameter mode. This function changes the value of HMERGE to XXXX. HMERGE is a peakpicking parameter and is described in the P5 function description.

Function: CHANGE PARAMETER LOGPOW

Command: L1 = XXXX<CR>

Description:

This function is invoked only when VU is in the change parameter mode. This function changes the value of LOGPOW to XXXX. LOGPOW specifies the spectrum type and is described in the power spectrum (P) function description.

Function: CHANGE PARAMETER PERHI

Command: P1 = XXXX<CR>

Description:

This function is invoked only when VU is in the change parameter mode. This function changes the value of PERHI to XXXX. PERHI is a peakpicking parameter and is described in the P5 function.

Function: CHANGE PARAMETER PERLO

Command: P2 = XXXX<CR>

Description:

This function is invoked only when VU is in the change parameter mode. This function changes the value of PERLO to XXXX. PERLO is a peakpicking parameter and is described in the P5 function description.

Function: CHANGE PARAMETER SFREQ

Command: S1 = XXXX<CR>

Description:

This function is invoked only when VU is in the change parameter mode. This function changes the value of SFREQ to XXXX. SFREQ is a peakpicking parameter and is described in the P5 function.

Function: CHANGE PARAMETER ZPSCAL

Command: Z1 = XXXX<CR>

Description:

This function is invoked only when VU is in the change parameter mode. This function changes the value of ZPSCAL to XXXX. ZPSCAL controls the scaling of the spectral display. This parameter is described in the spectra (P) function description.

VU OPERATING INSTRUCTIONS

Program Execution

A sample user/program terminal dialog is shown in Figure 15. PDP generated responses are underlined. The RT-11 operating system must be in the keyboard monitor, indicated by the period (.) printed as a prompt character. VU is an overlayed program currently residing on DP0, and thus the DP devide handler must be memory resident. The "LOAD DP" command loads the device handler in memory; the "RUN" command loads the VU program and begins execution.

The user must now specify the input data file to be analyzed. The input data file used as the example is SS900.DAT on device DP2. Note: The DAT extension is a VU default and does not have to be entered. This file contains one channel of digitized data. Since it has only one channel, VU will not ask the user to specify the channel to be observed. This input file contains 2048 points. VU now displays across the screen the first 512 values from the input data file.

```
.LOAD DP

.RUN DPOVU

ENTER DATA FILE:  *DP2:SS900

ENTER # CHANLS:  1

LAST SAMPLE IS #    2048
J1=1000
S2=256
T

S

ENTER REAL SCALE FACTOR:  1.5
T
P5
B1=500
Z

FIRST SAMPLE = 100

DURATION OF SHIFT = 50

# OF WINDOWS = 20

**************PEAKPICKING COMPLETED**************
G
P
B
S1=1024
P
E

STOP --

PAUSE --
TYPE <CR> TO EXIT


.
```

Figure 15 - Sample VU Session

The program is now ready to accept user commands. The first command (J1=1000) moves the displayed data so that the 1000th data value is the first point displayed. The next command (S1=256) changes the horizontal scale to display 256 points across the screen. The FFT of the displayed data is generated and displayed next (T). The S command indicates the user wants to change the vertical scale of the input data. The user enters the desired scale factor after a prompt message from VU. The next three commands erase the FFT display (T), generate and display the power spectrum with peaks selected (P5), and move the data display back 500 points (B1 = 500). Note: Since the spectrum was displayed when the move command was given, the spectrum for the new data display will be calculated and displayed.

The next command (Z) creates a file of peaks from a series of spectral slices. The user must specify the first point for the first spectrum (Note: If zero, then the first display point is used), the number of points between each succeeding sepctral slice (DURATION OF SHIFT), and the number of spectral slices (# OF WINDOWS). The user enters these parameters as the program asks for them. The program outputs a message when the process is complete. The display does not change.

The remaining commands complete the example. A hardcopy of the display is produced (G), the spectrum and peaks are erased (P), the displayed data are moved to the first data point (B), the horizontal scale is changed to display 1024 points (S1=1024), a power spectrum is generated (P), and the program is exited (E). Upon exiting, the system goes into a pause state. The user must type a carriage return to return to the RT-11 keyboard monitor. At this time the display is erased.

Summary of VU Commands

Tables 1, 2, 3, and 4 give a summary of the VU commands. Table 1 presents an alphabetized ordering of the VU functions and the command that performs the function. The detailed description of the function is given on the indicated page. Table 2 presents an alphabetized list of the VU primary commands and the function associated with each. Table 3 presents an alphabetized list of the change parameter commands and the function associated with each. Table 4 presents an alphabetized list of the smoothing commands and the function associated with each.

TABLE 1 - VU FUNCTIONS

FUNCTION	COMMAND	PAGE
Change number of channels	C	
Change parameters (enter change parameter mode)	X	
Erase displayed Fast Fourier transform	T	
Erase displayed peakpicking results only	P5	
Erase displayed spectrum and peakpicking results (if displayed)	P	
Exit program. Return to operating system	E	
Fast Fourier transform generation	T	
Hardcopy of current display	G	
KLASIT mode (smoothing)	K	
Locate beginning and end of an event	L	
Move display to beginning of the input data	B	
Move display back by a specified number of sample points	B1=XXXX	
Move display by a specified number of sample points and change the default move command to B1=XXXX	B2=XXXX	
Move display by default command (initial default is F1=512.)	<BLANK> or <CR>	
Move display forward to the end of data	F	
Move display forward by a specified number of sample points	F1=XXXX	
Move display forward by a specified number of sample points and change the default move command to F1=XXXX	F2=XXXX	
Move display to the specified sample point	J1=XXXX	
Peakpicking results displayed with the generated spectrum	P5	
Print specified number of input data values	Q7-XXXX	
Reenter VU program	R	
Scale the vertical axis	S	
Scale the horizontal axis. Display specified points	S1-XXXX	
Spectral slices with peakpicking results	Z	
Store current display buffer on disk	T5	
Tape transfer from SDS 910 to PDP11	T1	

TABLE 2 - VU COMMANDS

COMMAND	FUNCTION
B	Move display back to beginning of the input file
B1=XXXX	Move display back by XXXX sample points
B2=XXXX	Move display back by XXXX sample points and change the default move command to B1=XXXX
<BLANK> or <CR>	Move display by default command (initial default is F1-512)
C	Change number of channels
E	Exit program. Return to operating system
F	Move display forward to the end of data
F1=XXXX	Move display forward by XXXX sample points
F2=XXXX	Move display forward by XXXX sample points and change the default move command to F1=XXXX
G	Generate a hard copy on the x-y plotter of the current display
rJ1=XXXX	Move the display to the XXXXth sample point
K	Enter smoothing mode (KLASIT)
L	Locate beginning and end of an event
P	Either generate and display the power spectrum of the currently displayed input data or, if spectrum is displayed, erase spectrum and displayed peak indicators
P5	Either generate and display power spectrum with peakpicking results or, if spectrum with peakpicking results is displayed, erase peakpicking results leaving only the power spectrum display
Q7=XXXX	Print out first XXXX value of the displayed input data
R	Re-enter the VU program. A new input file is requested
S	Change the vertical scale
S1=XXXX	Change the horizontal scale to display XXXX points
T	Either generate and display the Fast Fourier transform of the currently displayed input data or, if the transform is currently displayed, erase the transform
T1	Transfer data from the SDS 910 tape to the PDP
T5	Save the current display buffer on disk. Up to five display buffers can be saved
X	Enter change parameter mode
Z	Produce spectral slices and apply peakpicking to the slices. Store the peak results on file

TABLE 3 - CHANGE PARAMETER MODE COMMANDS

COMMAND	FUNCTION
A1=XXXX	Change peakpicking parameter AMPMIN
A2=XXXX	Change locate parameter AMPBEG
B1=XXXX	Change peakpicking parameter BAND
B2=XXXX	Change peakpicking parameter BWIDTH
F1=XXXX	Change peakpicking parameter FACTOR
F3=XXXX	Change peakpicking parameter FMERGE
H1=XXXX	Change peakpicking parameter HMERGE
L1=XXXX	Change spectrum parameter LOGPOW
P1=XXXX	Change peakpicking parameter PERHI
P2=XXXX	Change peakpicking parameter PERLO
S1=XXXX	Change spectrum parameter ZPSCA1
X	Exit change parameter mode. Return to VU mode
X1	Change current parameter values to the default values
X2	Change the default values to the current parameter values
Z1	Change power spectrum scale factor

TABLE 4 - SMOOTHING MODE COMMANDS

COMMAND	FUNCTION
B	Move display back to beginning of the input file
B1=XXXX	Move display back by XXXX sample points
B2=XXXX	Move display back by XXXX sample points and change the default move command to B1=XXXX
<BLANK> or <CR>	Move display by default command (initial default is F1=512)
D	Display original data only
D1	Display original data and unsmoothed noise
D2	Display unsmoothed noise and smoothed noise
D3	Display original data and final smoothed noise
E	Exit program. Return to operating system
F	Move display forward to the end of data
F1=XXXX	Move display forward by XXXX sample points
F2=XXXX	Move display forward by XXXX sample points and change the default move command to F1=XXXX
G	Generate a hard copy on the X-Y plotter of the current display
I1=XXXX	Change parameter I1
I2=XXXX	Change parameter I2
I3=XXXX	Change parameter I3
J1=XXXX	Move display to the XXXXth sample point
K	Exit smoothing mode. Return to VU mode
L1=XXXX	Change parameter L1
L2=XXXX	Change parameter L2
L3=XXXX	Change parameter L3
M	Superimpose (merge) bottom display on top display
N1=XXXX	Change parameter N1
N2=XXXX	Change parameter N2
P1=XXXX	Change parameter P1
P2=XXXX	Change parameter P2
P3=XXXX	Change parameter P3
Q0=XXXX	Change parameter Q0
Q1=XXXX	Change parameter Q1

TABLE 4 - SMOOTHING MODE COMMANDS (Cont'd)

COMMAND	FUNCTION
Q7=XXXX	Print out first XXXX value of the displayed input data
S1=XXXX	Change the horizontal scale to display XXXX points
W	Display/Erase WAMSER results
W1	Either display WAMSER results and superimpose on raw data or, if already displayed, erase WAMSER superimposed display
W2	Either display WAMSER results and superimpose on smoothed results or, if already displayed, erase WAMSER superimposed display

PROGRAM SELECT

Introduction

Program SELECT is an interactive graphics program for the examination, selection, and tracking of peaks in power spectra. SELECT operates on data supplied to it by program VU in the form of lists of frequency bin numbers associated with peaks in individual power spectra. For the sake of convenience, each list of numbers associated with a single power spectrum will be referred to as a "spectral slice." SELECT allows the user to examine the ensemble of spectral slice data and apply line following techniques to track frequency peaks throughout the ensemble. SELECT is coded in PDP-11 FORTRAN for the PDP 11/45 computer and uses the VT11 Graphics Processor.

Theory of Operation

Command Concept. SELECT is a command driven program which reads data from a specific file and awaits user directives input at the keyboard. The functions initiated by user commands include the selection of spectral slices for tracking, optional displays, movement of displays, tracking of spectral data, alteration of program parameters, and reinitializations. PDP 11/45 console potentiometers and LED's can be enabled to ascertain frequency values of any portion of a display. In addition, an option is provided for the generation of a hard copy of any display.

Firgure 16 is a typical display of spectral slice data; the position of peaks is indicated by the "X" markers. The time difference between spectral slices depends on the manner in which the power spectra were generated in VU. SELECT gives no consideration to the real time increments between slices, either in the displays or in the tracking procedure; it assumes, therefore, either the distribution of spectra in time is uniform or that time is an independent variable.

The displays of spectral data are limited to a maximum of 30 spectral slices. Different spectral slices may be displayed by moving the display ahead in time (up), thereby "rolling" the first slices off the screen and adding later slices at the bottom of the screen. The display can also be moved backward (down) and reset to the beginning of the data.

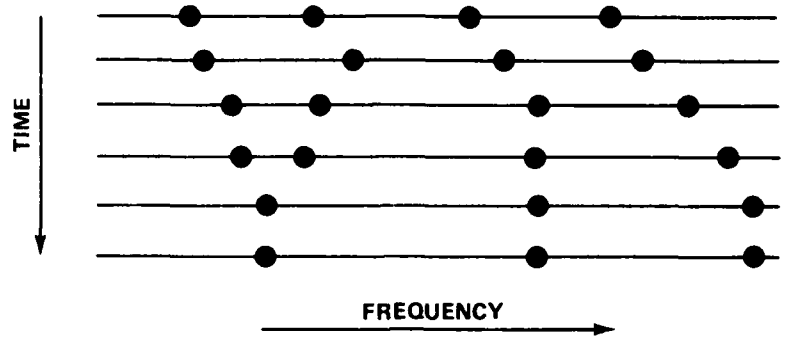


Figure 16 - Typical Display of Spectral Slice Input to SELECT

The maximum number of spectral slices that can be examined by SELECT is 1023. The user can display and review any part of the input and can select (mark) any spectral slices he wishes, not exceeding 512 in number, to be set aside and analyzed further by the tracking procedure of SELECT. These operations are initiated by the "select," "move," and "track" commands.

All commands available to the user are explained in detail in the next section of this report entitled "Functions."

Tracking. The tracking procedure is an application of line following approximation techniques to the collection of data representing the peak frequencies in the selected spectral data. The tracking procedure connects a peak in one spectral slice with one in the next slice (or, if unable to do so, a peak in a later slice) which is close in frequency and time to the last selected peak. Time here is used in the sense of difference in spectral slice index. The result is a track (or time-ordered sequence) of frequency peaks characterized by small gradual changes in frequency value. The tracking procedure generates as many tracks as possible without duplication up to a limit of 100 tracks and stores all tracks on a disk file.

The "track" command initiates the tracking procedure but does not change the display. A separate keyboard directive is required to display the tracks. Figure 17 shows the tracking results from the data displayed in Figure 16.

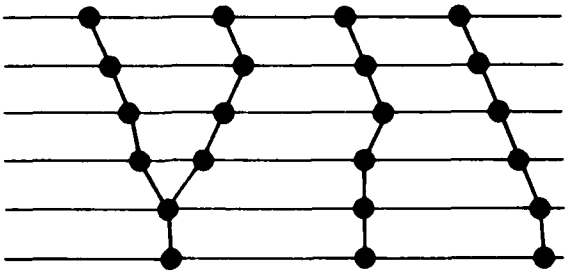


Figure 17 - Display of Tracks

Parameters. The results of the tracking procedure are sensitive to certain parameters used within the tracking algorithm. These parameters refer to the minimum number of similar frequency peaks needed to start a track, the time gap signaling the end of a track, and the frequency gap threshold for choosing close peaks. These parameters and their current values, as well as some other program parameters specifically related to certain commands, may be reviewed and changed by entering the change parameter mode of the program. In this mode, parameters, current values, and the appropriate command to change the values are displayed on the screen. The significance of all parameters is discussed in more detail in the section on "Functions" that follows.

Files and User Cautions. Program SELECT deals with three separate disk files, DP0:MORPK2.DAT, DP0:TRKABL.DAT, and DP0:TRAKED.DAT. These files are described in detail in Appendix B, but a brief summary and word of caution in their use is presented here to avoid possible confusion over what is being displayed by SELECT during various stages of execution.

The typical procedure is to run program VU as described in the preceding sections and generate spectral data on DP0:MORPK2.DAT. This file is input to SELECT. In SELECT, the user selects, for tracking, spectral slices which are stored on DP0:TRKABL.DAT. The "tracking" command directs SELECT to process the contents of DP0:TRKABL.DAT and store the tracks on the third file DP0: TRAKED.DAT. The "display tracks" command reads tracks from DP0:TRAKED.DAT and displays them. As long as this natural ordering of SELECT functions is followed, there should be no confusion.

SELECT, however, is designed in such a way that the user may run SELECT and perform this sequence of operations in one session and return at a later time and go directly to the display of tracks generated at the last session without again performing either the selection or tracking phase. This is possible because the track results are retained on file. On the other hand, if the user has new input data to SELECT and performs the selection phase but either fails to move his data selected for tracking to DP0:TRKABL.DAT or fails to enter a "Track" command, then the tracks displayed will not be new tracks but rather the tracks generated when SELECT was last executed. In summary:

- The user must be aware that the use of new data requires new movement of slices to DP0:TRKABL.DAT and new tracking.
- If more than one person uses SELECT, or if SELECT operates on more than one kind of data, tracks may be confused. To avoid confusion, files should be copied after execution of SELECT and recopied before the next run in which they are to be examined.

Functions

Function: PROGRAM START AND INITIAL DISPLAY

Command: RUN SELECT (entered when PDP 11/45 system is in keyboard monitor)

Description:

This statement begins execution of program SELECT, sets up initializations, creates the initial display, and then awaits further commands from the user. The initial display consists of the respresentations of spectral slices numbers 1 through 30 as they appear on the SELECT input file DP0: MORPK2.DAT generated by program VU. The data for a spectral slice consist of the frequency bin numbers which contained peaks in the corresponding power spectrum. The data are represented and displayed by SELECT as a solid horizontal line called a "base line," with small x's placed at the frequency value of each peak. At most, 30 slices are displayed, progressing in time from top to bottom of the screen. A sample spectral data display is shown in Figure 18.

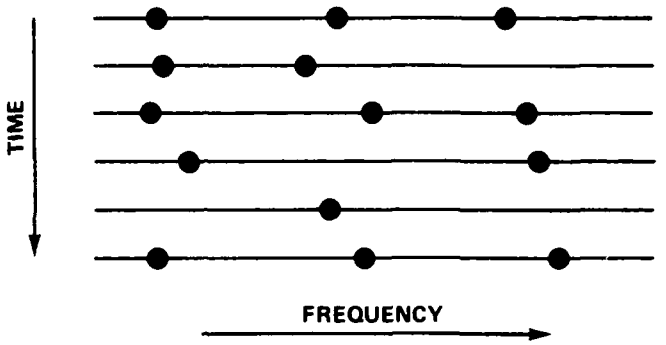


Figure 18 - Sample Spectral Data Display

Function: SELECT SPECTRAL SLICE ON DISPLAY

Command: An

n is a positive integer $\leq$ k, where k $\leq$ 30 is the number of slices currently displayed.

Description:

The "A" command selects a spectral slice from among those currently being displayed and "marks" it for later input to the tracking procedure. If the designated slice is already marked, it is unmarked and, hence, not included as input to the tracker. The display of the selected spectral slice (assuming it was previously unmarked) is changed to show a dashed or broken "base line" for the slice.

Example:

Given the display shown in Figure 19

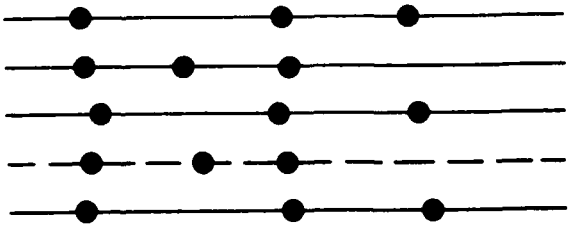
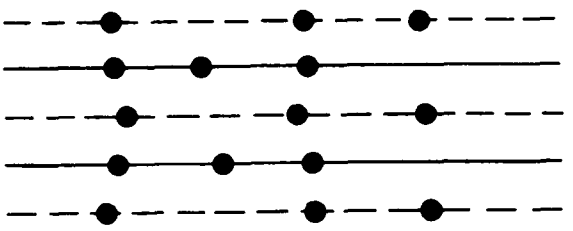


Figure 19 - Marked Spectral Data Display

the sequence of directives A1, A3, A4, A5 changes the display to that shown in Figure 20.



Firgure 20 - Altered Marked Display

Slices number 1, 3, and 5 are now marked for future tracking.

Function: SELECT BLOCKS OF SPECTRAL SLICES

Command: Cn m

n and m are positive integers, $n \leq m \leq k \leq 99$, where k is the total number of spectral slices on file.

Description:

The "C" command performs the same function as the "A" command except that it acts on blocks of spectral slices rather than on a single slice. The numeric designators in the "C" command are absolute spectral slice indices within the input file and are not restricted to the bounds of the current display.

Example

In the example given for the "A" command, the same result could be achieved by

C 1 5

A2

Function: SELECT ALL SPECTRAL SLICES

Command: C 3

Description:

The "C3" command performs the same function as the "A" or "C" commands, but does so on the entire input file of spectral slices, DP0:MORPK2.DAT.

Function: MOVE SELECTED SPECTRAL SLICES TO TRACKER INPUT FILE

Command: M

Description:

The "M" command copies those spectral slices that have been selected or marked for tracking from the file, DP0:MORPK2.DAT, of spectral data input to program SELECT to the file, DP0:TRKABL.DAT, which serves as input to the tracking procedure. The selected spectral slices are copied in the order of their occurrence on DP0:MORPK2.DAT to preserve time ordering. The existing display is erased and the selected spectral data are displayed in its place, starting at the first selected slice occurring after the start of the preceding display. The slice numbers of the first and last slice displayed are shown at the upper and lower right-hand side of the screen, respectively.

Function: DISPLAY CONTENTS OF TRACKER INPUT FILE

Command: M1

Description:

The "M1" command erases the current display and displays the contents of DP0:TRKABL.DAT, the file containing the spectral slices selected for input to the tracking procedure.

Function: MOVE DISPLAY UP

Command: Un

n is a non-negative integer $\leq$99.

Description:

The "U" command advances the current display ahead in the file being displayed by the number of spectral slices indicated by n. The display will not go past the end of data and, at the end of the file, will display the last 30 slices regardless of the value of n. If n is zero, the display is moved forward 30 slices.

Function: MOVE DISPLAY DOWN

Command: Dn

n is a non-negative integer $\leq$99.

Description:

The "D" command is similar to the "U" command except that it moves the display down or backwards in time. If n is zero, the display is moved to the beginning of the file being displayed.

Function: AUTOMATIC MOVE UP OR DOWN

Command: <blank>, or <CR>

Description:

The blank and the carriage return are automatic move commands applicable to whatever file is currently being displayed. The effect of the command is to move the display up or down a certain number of slices in the same manner as the "U" or "D" command. Two program parameters specify whether the move is up or down and the amount of the shift. These parameters are retained on disk file which contains a working set of parameter values and a default set, each of which is accessible for modification (cf. X and X1 commands).

Parameters:

CMAND1 - working parameter storing the character "U" or "D" as desired to specify the direction of the automatic move. Fifth word on DP0:TPARAM.DAT.

NUM1 - working parameter storing the integer number of slices to be shifted in the automatic move. Sixth word in DP0:TPARAM.DAT.

Function: BACK TO INPUT FILE

Command: B

Description:

The "B" command displays the input file, DP0:MORPK2.DAT, of spectral slice peaks in place of the current display. The new display will start at the same point as the last display of DP0:MORPK2.DAT. (The starting point distinguishes this command from the re-enter command.)

Function: RE-ENTER PROGRAM

Command: R

Description:

The "R" command erases the current displays, rereads the SELECT input data, and displays the first 30 spectral slices of DP0:MORPK2.DAT.

Function: IDENTIFY SPECTRAL SLICE

Command: Wn

n is positive integer ≤ 30

Description:

The "W" command allows the user to specify a spectral slice on the current display by its relative position on the display and to ascertain its absolute index in the original input file, DP0:MORPK2.DAT. Program SELECT will echo the number n on the printer and print beside it the absolute index of the designated slice.

Function: GENERATE HARD COPY OF DISPLAY

Command: G

Description:

The "G" command produces a hard copy of the current display on the Hewlett-Packard X-Y Plotter interfaced to the PDP-11.

Function: EXIT PROGRAM

Command: E

Description:

The "E" command terminates the program.

Function: POTENTIOMETER ENABLE/DISABLE FOR FREQUENCY MEASURE

Command: P

Description:

The "P" command, when first entered, displays a cursor on the frequency axis of the display which is movable by using the channel potentiometer on the PDP-11/45 system console. The frequency value at the position of the cursor is displayed in the LED's on the PDP-11/45 console.

The second entry of the "P" command disables the potentimeter and LED's.

Parameters:

ZNERVL - frequency interval between frequency bins. Maximum detectable frequency is 9000 Hz with 18-KHz sample rate.

ISTART - The start (min) frequency of peaks in the spectral slice data input to SELECT. Seventh word on DPO:TPARAM.DAT. Subject to user change up to a maximum of 7000 Hz (cf. "X" command)

Function: CHANGE WORKING PARAMETERS, ENABLE/DISABLE

Command: X

Description:

The "X" command has two functions.

If the system is not in the change parameter mode, the "X" command erases the current display and displays the set of parameters for SELECT, their current working values, and the commands needed to change the working values (Figure 21); the system is then in the change parameter mode and awaits a command to change a parameter or exit the mode.

If the system is currently in the change parameter mode, the "X" command saves the new working parameters on the parameter file, DP0:TPARAM.DAT, erases the display, and returns to the SELECT command mode to await further direction.

```
                    TRACKER PARAMETERS:
                            TRACKS (T1) = 80
                            FRQGAP (F1) =  2
                            KNTLIM (K1) =  3
                            TIMGAP (T2) =  2

                            ISTART (I1) =  1
                    CURRENT CARRIAGE RETURN DEFAULT:
                           CAMAND1 (C)  =  U
                           NUM1    (N1) = 30
TYPE X TO RETURN
TYPE X1 TO RETURN ADN CREATE NEW DEFAULT VALUES
```

Figure 21 - Change Parameter Display

Function: CHANGE DEFAULT PARAMETERS, ENABLE/DISABLE

Command: X1

Description:

"X1" has two functions: to get into or to get out of the change parameter mode.

If the system is not in the change parameter mode, the "X1" command erases the current display, reads the default parameters into the working set of parameters, displays the new working set and the commands needed to change them, and awaits a command to change a parameter or exit the change parameter mode.

If the system is currently in the change parameter mode, the "X1" command copies the new working parameters to the default parameters, saves both (equal) sets in the parameter file DPO:TPARAM.DAT, erases the display, and returns to the SELECT command mode.

Example:

(Display is the same as that for "X" command).

Function: CHANGE DIRECTION OF AUTOMATIC MOVE

Command: C

Description:

The "C" command changes the direction of the automatic move initiated by the <blank> or <CR> commands by changing the working parameter controlling this direction. This command may be entered only when the display is in the change parameter mode.

Parameter:

CMAND1 cf ("Automatic Move Up or Down")

Function: CHANGE MAGNITUDE OF AUTOMATIC MOVE

Command: N1 = n

n is an integer

Description:

This command assigns the value n to the working parameter that specifies the number of spectral slices by which the display is to be moved by an automatic move command. (cf. "Automatic Move Up or Down"). Valid only in change parameter mode.

Parameter:

NUM1 - cf ("Automatic Move Up or Down")

Function: CHANGE LOWER FREQUENCY BOUND OF DISPLAY

Command: I1 = n

n is a positive integer $\leq$9000

Description:

This command assigns the vlaue n to the working parameter that specifies the lower bound for frequency on all displays. Valid only in change parameter mode.

Parameter:

ISTART - working parameter containing the lower frequency bound of displays. The seventh value in DPO:TPARAM.DAT

Function: CHANGE FREQUENCY GAP THRESHOLD

Command: F1 = n

n is an integer

Description:

This command assigns the value n to the working parameter that stores the frequency gap threshold used in the tracking procedure. (cf. "Track Spectral Data"). Used only in change parameter mode.

Function: CHANGE TIME GAP THRESHOLD

Command: T2 = n

Description:

This command, in the change parameter mode, assigns the value n to the working parameter that stores the time gap threshold used in the tracking procedure. (cf. "Track Spectral Data.")

Parameter:

TIMEGAP - (cf. Tracking)

Function: CHANGE STABLE START THRESHOLD

Command: K1 = n

n is integer

Description:

This command assigns the value n to the working parameter that stores the size threshold for a stable sequence of peaks at which to begin generating a track. Valid only from change parameter mode.

Parameter:

KNTLIM - (cf. Tracking)

Function: CHANGE MAXIMUM NUMBER OF TRACKS

Command: T1 = n

n is integer ≤ 100

Description:

This command changes the maximum number of tracks that the tracking procedure will compute and store. Valid only in change parameter mode.

Parameter:

TRACKS - (cf. Tracking)

Function: TRACK SPECTRAL DATA

Command: T

Description:

The "T" commands performs tracking on the spectral slice data the user has selected for tracking, i.e., the contents of the file DP0:TRKABL.DAT. The results of the tracking procedure are stored on file DP0:TRAKED.DAT, up to a maximum of 100 tracks. There is no change in the display.

The spectral slice input data to the tracking procedure consists of a list of frequency bin numbers representing the frequencies of peaks in the power spectrum. The tracking procedure of SELECT examines this ensemble of peaks and applies a line following technique to the data in an attempt to identify time ordered sequences of smoothly changing peaks; these sequences are called tracks. Peaks in successive spectral slices belong to the same track if they are "close" in (frequency) value. Special consideration is given to determining the start of a track. The spectral ensemble is reviewed in a search for areas of "stable peaks" in which, over a number of consecutive spectral slices, a constant peak or a peak with no more than a unit change in frequency bin number exists. A region of stable peaks which exceeds a minimum length criterion forms the part of a track from which the complete track is subsequently generated. The tracking procedure begins in the stable region and proceeds iteratively, first forward through later spectral slices and then backward through earlier spectral slices, seeking to add the "closest" peaks (one per spectral slice) to the track.

The tracking algorithm attempts to add a peak to the track from each successive spectral slice. When no peak satisfies the frequency gap threshold, successive slices are examined to find a close peak which will resume the track. The resulting gap in the track is filled by a straight line interpolation of the peak bin value over the index (time) gap. The track ends when the index gap exceeds a user supplied parameter value called the time gap threshold.

Tracks are stored in vectors equal in length to the number of spectral slices contained in the file to be tracked. An element of the track vector is the frequency bin number of the peak in the corresponding spectral slice which belongs to the track. The portion of the set of spectral slices which has no members in the track (occurring both at the beginning and end) is indicated by

a negative one value in the track vector. Each track vector is stored on file DP0:TRAKED.DAT and the tracking procedure is recycled until no more tracks can be generated or until a user supplied limit (or program limit = 100) is reached.

New tracks are always begun in a different stable region, and duplicate tracks are not permitted. However, intersecting tracks can and do arise and no checks are made for them.

Parameters:

KNTLIM - The minimum length (number of spectral slices) for a stable region in which to start tracking.

FRQGAP - The frequency gap threshold; the maximum allowable change in bin numbers for continuation of a track.

TIMGAP - The time gap threshold; the maximum number of spectral slices that can be skipped and still continue the track.

TRACKS - The maximum number of tracks that the tracking procedure is to generate.

Several parameters are under user control but are transparent to the operation of program SELECT. These are parameters to program VU which affect the spectral slice data input to SELECT. Some of these parameters are

signal sample rate

window size of the FFT

real time interval between spectral slices

parameters for frequency peakpicking

The obvious dependency of the tracking parameters on these VU parameters (e.g., the effect of FRQGAP depends on the frequency resolution of the power spectrum) should be an important consideration for the user.

Function: DISPLAY/ERASE TRACKING RESULTS

Command: T1

Description: The "T1" command displays the results on DPO:TRAKED.DAT of the tracking procedure or erases the tracks if they are already displayed. (Note: Since file contents are never purged, if the "T1" command is not preceded by a "T" command when new data are examined, "T1" will display old tracking results from the last execution of SELECT.)

Example:

Given the display shown in Figure 22 of selected spectral slices which have been moved from DPO:MORPK2.DAT to DPO:TRKABL.DAT., the "T" command will

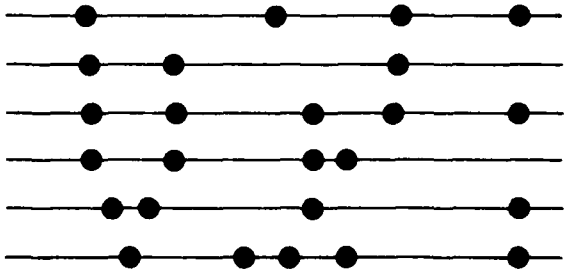


Figure 22 - Selected Spectral Data Display

not change. When the "T1" command is entered after "T", the display shown in Figure 23 will appear.

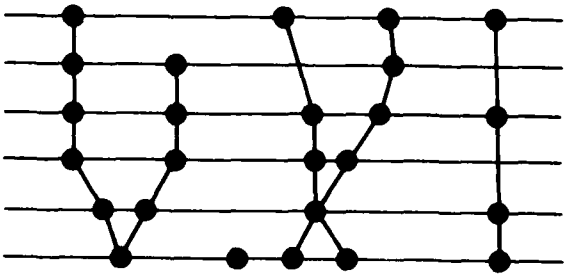


Figure 23 - Display of Tracked Data

An entry of "T1" now simply erases the tracks only and returns to the preceding display of selected slices.

Function: DISPLAY/ERASE ONLY TRACKING RESULTS

Command: T2

Description:

The "T2" command first erases the display and then displays only the tracks on file DP0:TRAKED.DAT. If tracks are already displayed, "T2" erases the tracks.

Example:

After "T" has been entered, the effect of the "T2" command on the display of selected slices shown in the first example under the "T1" command, is shown in Figure 24.

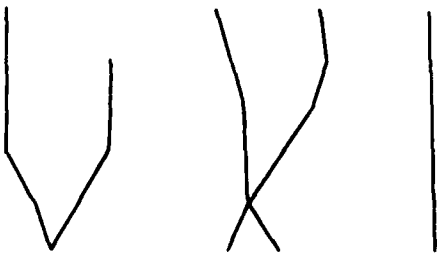


Figure 24 - Display of Tracks Without Spectral Data

Operating Instructions

Program Execution. Program SELECT is initiated from the keyboard by typing

RUN SELECT

If this command is preceded by

* GT OFF
* GT ON /L:3

the terminal print is disabled and output is displayed in a three-line buffer at the top of the CRT screen without interfering with program displays.

Input to program SELECT is assumed always to reside on the disk file DP0:MORPK2.DAT. The output files are also always the same, DP0:TRKABLE.DAT and DP0:TRAKED.DAT. The user must therefore be aware of what is on these files and whether or not he has stored new tracker input data (DP0:TRKABL.DAT)

before he does tracking. Since the file names are always the same, tracker results may be examined as often as desired without having to track the data over again.

After the RUN SELECT command, the only keyboard input is through user commands summarized here and explained in the preceding section.

Summary of User Commands. User commands are listed below in alphabetical order by command name. The symbol initiating the command is listed to the right of the name. Each command is explained in detail in the FUNCTIONS section of this report.

Automatic move up or down	<blank> or <cr>
Back to input file	B
Change default parameters, Enable/Disable	X1
Change direction of automatic move	C
Change frequency gap threshold	F1-n
Change lower frequency bound of display	I1=n
Change magnitude of automatic move	N1=n
Change maximum number of tracks	T1=n
Change stable start threshold	K1=n
Change time gap threshold	T2=n
Change working parameters, Enable/Disable	X
Display contents of tracker input file	M1
Display/Erase tracking results	t1
Display/Erase only tracking results	T2
Exit program	E
Generate hard copy of display	G
Identify spectral slice	W n
Move display down	D
Move display up	U
Move selected spectral slices to tracker input file	M
Potentiometer Enable/Disable	P
Program start	RUN SELECT
Select all spectral slices	C3
Select blocks of spectral slices	C n m
Select spectral slice display	A n
Track spectral data	T

WATERFALL DISPLAY PROGRAMS

Overview

Waterfall display programs display a series of consecutive spectra on the PDP-11 graphic display terminal. Each power spectrum is slightly displaced from the previous spectrum, producing a cascading or waterfall appearance. The waterfall programs allow the user to display, move through, and produce hardcopy of the spectra. The user adjustable parameters determine the vertical scale, the spacing between adjacent spectra, and the number of spectra displayed.

Two versions of waterfall programs have been written. First is the conventional display program, WFALL, which produces a display with frequency on the horizontal axis and time and amplitude on the vertical axis. This first program allows the user to specify a display with perspective, i.e., each succeeding spectrum has a horizontal and vertical offset from the previous spectrum. The second version, RWFALL, displays a rotated waterfall where the horizontal axis is time and the vertical axis is frequency and amplitude.

The waterfall programs, like the VU program, are command driven. Each command consists of a single character followed by an equal sign (=) followed by a value. The commands allow the user the following options:

- jump to a specified spectrum and display it first
- move continuously through the spectra at a specified rate (continuously add a new spectrum and erase the oldest)
- stop continuous movement
- hard copy current display on X-Y plotter
- change user adjustable parameters

The user adjustable parameters control the rate at which a spectrum display moves, the number of spectra displayed, and the vertical scale.

Program POWGEN is the power spectra generator to generate the data for the display programs. It creates a file of consecutive power spectra from a data file. This program is also command driven, allowing the user to specify where in the data file the spectra should start and end and to specify the number of points in each spectrum.

Conventional Waterfall Display (WFALL)

WFALL displays as many as 20 successive power spectra. Each spectrum can have up to 512 points. Each succeeding power spectrum is displayed, displaced from the previous spectrum by connecting vectors between amplitudes of adjacent frequencies. Frequency is displayed on the horizontal scale and time on the vertical scale. The amplitude is scaled by a user defined parameter and is displayed using either a linear or logarithmic scale. The next spectrum is displayed above the previous spectrum and can be horizontally offset to the left, right, or with no horizontal offset. The horizontal offset gives the display an appearance of perspective. The distance above the previous spectrum is a user specified parameter. Figure 25 shows a WFALL display example. The example shows 20 spectra with 256 points per spectrum, displayed with a linear scale and with a spacing of 20 points between spectra using a right perspective offset.

WFALL reads the spectrum from a user specified file. The first block of this file contains parameter information, which includes the number of points in each power spectrum, the maximum amplitude value, the number of power spectra, and the point on the data file at which the power spectrum generation began. The remaining blocks contain the spectra. The spectra file can be generated with a program called POWGEN discussed at the end of this Waterfall section.

WFALL moves through the spectra file. Motion is produced by erasing the oldest spectrum and displaying a new spectrum above the last. All spectra are continuously moved down until each reaches the horizontal position of the previous spectrum. Then the oldest displayed spectrum is erased and a new one is generated. Once started, this motion continues until stopped by the user. Motion starts when the user types <CR>. Motion stops when a second <CR> is typed or when any command is entered.

The WFALL commands are presented in Tables 5 and 6. Table 5 shows the WFALL primary commands. In addition to the display motion (<CR>) the primary commands allow for

- jumping the display to the specified spectrum (J)
- jumping the display to the first spectrum in the file (B)
- creating a display hardcopy (H)
- setting perspective to left, right, or straight (P)

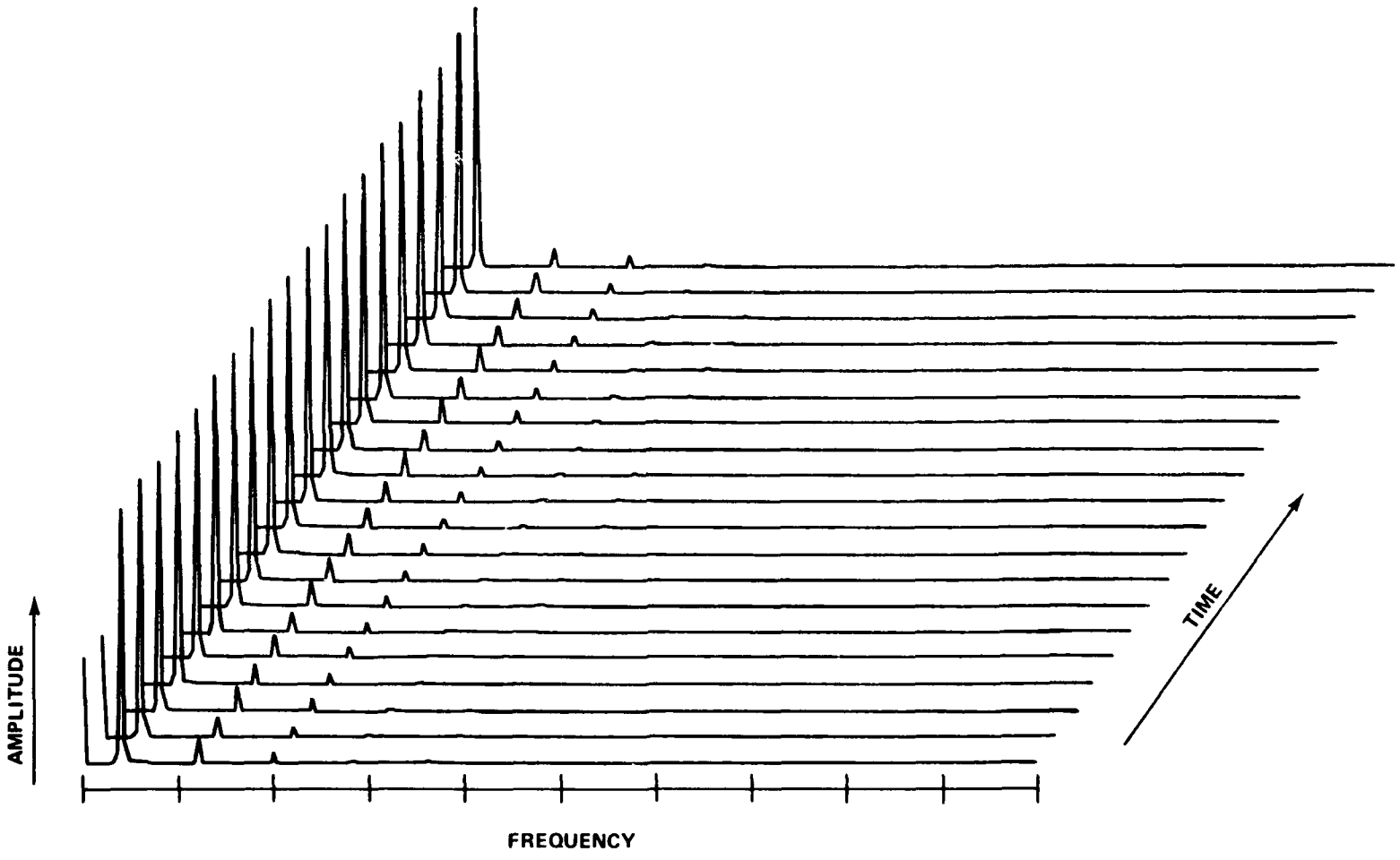


Figure 25 - WFALL Display Linear Scale

- entering the change parameter mode (X)
- exiting the program (E)

Table 6 shows the WFALL change parameter commands and the default values for each parameter. These commands generally affect the spectra display and are used to

- specify the spacing between the spectra (I)
- specify the number of spectra displayed (N)
- specify the scale of the display (S)
- specify the scale type as either linear or logarithmic (Z)
- specify the motion time constant (T)
- exit change parameter mode (X)

If the user enters an invalid command, WFALL will respond with "?????."

TABLE 5 - WFALL PRIMARY COMMANDS

COMMAND	FUNCTION
B	Move back to beginning of file and display the first NPWRS spectra. If the display is in motion, stop the motion.
<CR>	Start/Stop display motion. If display is stationary, start motion; otherwise, stop motion.
E	Exit program
H	Create hardcopy of the current display. If the display is in motion, stop the motion.
J=XXXX	Jump to the XXXXth spectrum on file and display the first NPWRS spectra from there. If the display is in motion, stop the motion.
P=XX	Generate the display with perspective (default is straight, i.e., P=0) P=0 straight perspective, i.e., no horizontal offset between spectra (default) P=1 right perspective, i.e., each succeeding spectrum is offset to the viewer's right. P=-1 left perspective, i.e., each succeeding spectrum is offset to the viewer's left.
R	Reenter program. Define a new input file.
X	Enter/Exit change parameter mode. If WFALL is not in change parameter mode, enter that mode and stop display if display is in motion. If WFALL is in change parameter mode, exit that mode and generate the display using the new parameters.

TABLE 6 - WFALL CHANGE PARAMETER COMMANDS

COMMAND	FUNCTION
I=XXXX	Change the spacing between the spectra to XXXX. The default spacing is 20.
N=XXXX	Change the number of spectra displayed to XXXX. The default number is 20.
S=XXXX	Scale the spectrum amplitudes by XXXX prior to displaying the spectrum. The default is .01.
T=XXXX	Change the display motion time constant to XXXX. The default is 1.
X	Exit change parameter mode.
Z=XX	Generate each spectrum with logarithmic or linear scale. Z=0 linear scale (default) Z=1 logarithmic scale

Rotated Waterfall Display (RWFALL)

RWFALL generates a rotated waterfall display. Each spectrum can have up to 512 points but no more than 128 points of each can be displayed. The number of power spectra displayed depends on the number of points displayed per power spectrum. The product of the number of displayed spectra and the number of displayed points per spectra is 2048. Thus, if 64 power spectrum points are displayed, then 32 spectra are displayed. The rotated waterfall display is created by joining vectors between amplitude values of adjacent spectra for the same frequency. The values for each succeeding spectrum are displayed to the right of the previous spectrum. This line of interconnected vectors represents a time history of the frequency (referred to as a bin). The next higher frequency, or bin, is displayed above the previous bin by a user specified spacing. The display produced has time on the horizontal scale and frequency and amplitude on the vertical scale. The displayed amplitude is dependent on a user defined scale factor and is displayed using either a linear or logarithmic scale.

RWFALL reads the spectra from a user specified file. The first block of this file contains parameters which include the number of points per spectrum, the maximum amplitude, the number of spectra on file, and the point on the data file at which the spectrum generation began. The remaining blocks contain the spectra. The spectra file can be generated with program POWGEN.

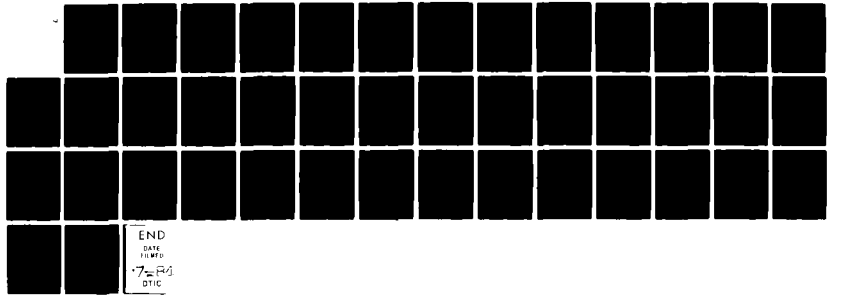
RWFALL moves through the spectra file. Motion is produced by deleting the oldest spectrum (left end of the screen) and displaying a new spectrum on the right of the screen. All spectra are continuously moved to the left until each reaches the vertical position of the previous spectrum. The oldest spectrum is deleted and a new one is generated. Once started this movement continues until stopped by the user. Motion starts when the user types <CR>. Motion stops when a second <CR> is typed or when any command is entered. Note: The motion does not necessarily stop immediately. It will stop only on spectra that are multiples of the number of displayed spectra divided by two. This fact is important only in realizing that motion may take up to half the screen to stop.

The RWFALL commands are presented in Tables 7 and 8. Table 7 shows the RWFALL primary comands. In addition to the display motion (<CR>), these primary commands allow for

AD-A141 421 INTERACTIVE SIGNAL AND PATTERN ANALYSIS AND RECOGNITION SYSTEMS (ISPARS)..(U) DAVID W TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER BET.. W PARSONS ET AL.

2/2

UNCLASSIFIED MAR 84 DTNSRDC/CMLD-84/06 F/G 9/2 NL

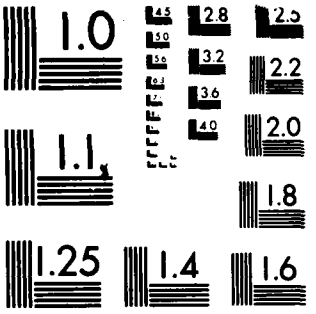


END

DATE FILMED

7-84

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

- jumping the display to a specified spectrum (J)
- jumping the display to the first spectrum in the file (B)
- creating a display hardcopy (H)
- changing the data file (R)
- entering the change parameter mode (X)
- exiting the program (E)

Table 8 shows the WFALL change parameter commands and the default values for each parameter. These commands generally affect the spectra display and are used to

- specify the first frequency bin displayed (A)
- specify the number of frequency bins to display (N)
- specify the spacing between adjacent bins (I)
- specify the scale of the display (S)
- specify the scale type as either linear or logarithmic (Z)
- specify the motion time constant (T)
- exit change parameter mode (X)

If the user enters an invalid command, RWFALL will respond with "??????".

TABLE 7 - RWFALL PRIMARY COMMANDS

COMMAND	FUNCTION
B	Move back to beginning of file and display the first spectrum. If the display is in motion, stop the motion.
<CR>	Start/Stop display motion. If display is stationary, start motion; otherwise, stop motion.
E	Exit program
H	Create hardcopy of the current display. If the display is in motion, stop the motion.
J=XXXX	Jump to the XXXXth spectrum on file and generate the spectrum display. If the display is in motion, stop the motion.
R	Reenter program. Define a new input file.
X	Enter/Exit change parameter mode. If RWFALL is not in change parameter mode, enter that mode and stop display if display is in motion. If RWFALL is in change parameter mode, exit that mode and generate the display using the new parameters.

TABLE 8 - RWFALL CHANGE PARAMETER COMMANDS

COMMAND	FUNCTION
A=XXXX	Change the first frequency bin displayed to XXXX. The default is frequency bin one.
I=XXXX	Change the spacing between adjacent frequency bins to XXXX. The default spacing is 4.
N=XXXX	Change the number of frequency bins displayed to XXXX. The default number is 128.
S=XXXX	Scale the spectrum amplitudes by XXXX prior to displaying the spectra. The default is 0.01.
T=XXXX	Change the display motion time constant to XXXX. The default is 1.
X	Exit change parameter mode.
Z-XX	Generate the display with either a linear or logarithmic scale. Z=0 linear scale (default) Z-1 logarithmic scale

Power Spectra Generator (POWGEN)

POWGEN is the power spectra generator. This program creates a file of consecutive non-overlapping power spectra from a waveform file. The power spectra file can be used as input to the waterfall display programs. Each power spectrum is generated from a sum of the squares of the real and imaginary coefficients of the Fast Fourier transform of the waveform. The Fourier transform is produced with twice as many points as the power spectrum.

POWGEN is a command driven program. The user must first specify the file names of the input and output. These file names are given using a standard RT-11 command string. Once the files have been accepted, POWGEN enters the command mode. Table 9 shows the acceptable commands. These commands allow the user to

- change the starting point for the first power spectrum (B)
- change the number of spectra generated (S)
- change the number of points per spectrum (N)
- specify identification information (I)

When the user has the desired parameters, the spectra generation is initiated by typing a <CR>. The spectra are written on the output file starting with the second block. The first block is written upon completion of the spectra generation. This block will contain the identification information, the number of points per spectrum, and the maximum value in the spectra file. The program cycles back on itself so the user can generate several spectra files. The program is terminated with a control C.

TABLE 9 - POWGEN COMMANDS

COMMAND	FUNCTION
B=XXXX	Use the XXXXth point of the input file as the starting point for spectrum generation. The default start is the first point.
<BLANK> or <CR>	Start power spectra generation. (Should be the last command)
I=40 char.	Enter 40 characters of ID information into the first block of the output file.
N=XXXX	Each power spectrum shall have XXXX points. XXXX should be a power of 2 and must be 512 or less. The default number of power spectrum points is 256.
S=XXXX	Set the limit to generate at most XXXX spectra on the output file. The default is to generate spectra until the end of the input is reached (S=-1)

PATTERN ANALYSIS SUBSYSTEMS

INTRODUCTION

Currently, there are two broad approaches to pattern recognition: statistical hypothesis testing and syntactic parsing. In the former, measurements are organized as fixed length vectors in a large but finite vector space. Transformations and projections of the space are then employed to extract significant feature axes in a lower-dimensional space for the sake of computational economy. Any of a wide spectrum of clustering or distance techniques may then be exploited to determine a discriminant surface between classes of feature vectors. This surface, a hypothesis test, is the realization of a pattern classification algorithm which may be validated against test data and assessed for predictive power and confidence of accuracy. The module which handles this type of pattern analysis is called WPS/OLPARS and is described in the next section.

In syntactic parsing, on the other hand, measurement strings are not necessarily of fixed dimension. Hence, the concepts of vector space and associated transformation and distance properties do not apply. Since the type of measurements handled by syntactic parsing can be thought of as time dependent--even if they actually exhibit an effective one-dimensional spatial dependence--the relevant statistical model is that of stochastic processes in a state or phase space. The state transitions may be represented in a graph, i.e., a set of vertices and edges. An algorithm for traversing an empirically derived graph for a given test string input and for accounting for possible anomalous deletions and insertions among the string symbols constitutes a pattern classification scheme and strongly resembles the parsing procedures of natural language translators or programming language compilers. Syntactic parsing based on a tree data structure--i.e., a set of finite, nonrecursive strings with possibly common prefixes--is handled by the WAVAN module. Candidates for such parsing might arise from the KLASIT waveform smoothing characterization in VU or from independent feature extraction routines. If the syntax is more complex--say, context-free--ISPARS provides a GENER8 module to transform a Backus-Naur Form description of the data set into a syntax graph and a RECOG module to parse the graph. Candidates for this more complex parsing might

arise from a contextual analysis of tactical scenarios to which a sound is related, or with embedded recursive waveforms arising from an intelligent source like speech.

The following sections briefly describe the software systems available for pattern analysis. Only a user-oriented summary containing appropriate references to detailed software documentation will be given here.

STATISTICAL HYPOTHESIS TESTING

Waveform Processing System; On-line Pattern Analysis and Recognition System (WPS/OLPARS)

The Waveform Processing System (WPS) is a large interactive graphics system developed by the Pattern Analysis and Recognition Corporation for Rome Air Development Center.[7] It is designed for the analysis and processing of waveforms, definition of waveform features, and the design of waveform classification logic. Originally, WPS was intended as a "front end" to the On-Line Pattern Analysis and Recognition System (OLPARS),[8] an interactive graphics system predating WPS, used to analyze vector data and develop classification logic. OLPARS is now a subsystem of WPS as is the On-Line Waveform Processing Language PARLAN,[7] which acts as an interface between the old WPS and OLPARS and also provides the user a means of creating his own waveform analysis programs.

WPS. The crucial characteristic of WPS is that it is an interactive graphics system; it communicates with the user through graphics displays and "menus" presented on a CRT screen. In addition, WPS has its own data management capability so that once data have been properly formated for input, the user need not be aware of data characteristics. The system can handle both time and spectral data and can distinguish between waveform and vector data. More than 500 analyst-selectable options (Figure 26) can be applied to any data set; the user need only identify the data set to be operated on. WPS is designed to use a foreground/background system so that many operations take place in the background while the analyst displays and analyzes data in the foreground.

The preprogrammed waveform processing options in WPS include single waveform display, edit modules, and multiple waveform displays. Algebraic/arithmetic options are available to normalize, smooth, demodulate, integrate, shift (in time or frequency), and add/subtract/multiply/divide waveforms. Spectral analysis transformations, such as FFT, FFT magnitude or phase, inverse FFT, log FFT magnitude, power spectrum, and cepstrum are also included. A segmentation module is included which facilitates waveform segmentation by providing a wide range of criteria that the user may select to mark the beginning and end of desired waveform segments. These criteria include cross correlation, convolution, rate of rise/fall, zero crossings, average power in time, and more.

PARLAN. Pattern Analysis and Recognition Language (PARLAN) is the key link between WPS and its major subsystem OLPARS, because only PARLAN can produce feature vectors from waveform data. PARLAN is a high level language based on FORTRAN and fully integrated with WPS and its data base. It originated as a feature definition language operating on segments of waveforms and has evolved to include general waveform segmentation and transformation capabilities in the form of high level primitives for these and other waveform processing operations. Since it is impossible for WPS to include preprogrammed options for all the waveform algorithms a user may require, PARLAN is designed with convenient data handling and enough low level capability to allow any user-designed waveform computation.

PARLAN programs may have as input/output both time and frequency domain waveforms, as well as vector data. Although a typical user would normally develop programs off-line, he may tune them, modify, and recompile "on-line" as part of a waveform analysis session. Although WPS has a number of pre-programmed options, the basic design and intention is that the user, through PARLAN, will interactively construct his own wave analysis program to obtain feature vectors for further analysis by OLPARS.

OLPARS. The On-Line Pattern Analysis and Recognition System (OLPARS) is a subsystem of WPS which was designed specifically to process vector data. OLPARS is an interactive graphics system used for pattern analysis and the construction of pattern classification logic. It provides a set of

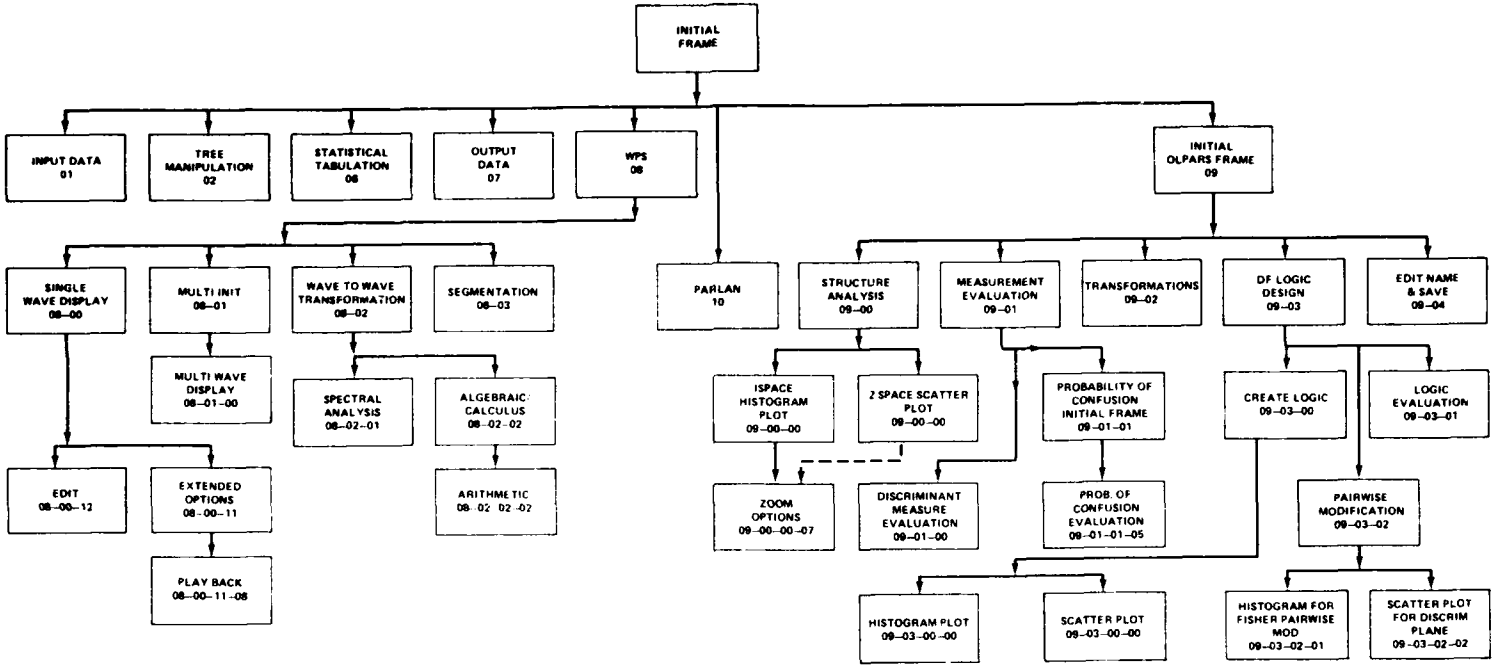


Figure 26 - Options Available in WPS/OLPARS

mathematical and graphical tools which allow an analyst to interact with a CRT and to display, transform, and manipulate multidimensional vector data to investigate its composition and distribution.[9] Like WPS, OLPARS is an interactive system in which the user is guided at each step by "menus" of possible operations. The data management facility of OLPARS allows the user to define class structures within data sets, change structures by partitioning classes, eliminate previous partitions, and delete "wild" points.

OLPARS uses a variety of projections and mappings into the two-dimensional CRT to allow the user to examine the structure of classes of n-dimensional vector measurements representing physical objects or phenomena. Typical projections include

- those which cluster classes
- projections to axes defined by the analyst (coordinate axes, eigenvectors, arbitary vectors)
- projections emphasizing class separation

The displays produced by these projections enable the analyst to recognize redundant and irrelevant features as well as especially significant features. By observing the data set displays in the projection space, the analyst can easily determine class separability and multi-modal class distributions. Sequential logic for class identification can be easily designed by repeated application of the maps and partitioning of the data into two groups within the projection space until all groups have been reduced to single classes (or single modes of a class). OLPARS performs all the mathematics and data set manipulations to transform the partitions into discriminant equations in the original n-space. The unique and appealing feature of OLPARS is its use of an interactive graphics system to analyze data. This feature provides the analyst with an immediate, visually intuitive grasp of his data without the burden of concentrating on the underlying mathematics.

Summary

WPS/OLPARS is a large, easily expandable interactive graphics system which allows the user to make visual decisions about the structure of either waveform data or vector data. He may visually restructure a segment the data, visually project or transform it, and visually eliminate irrelevant or

non-discriminatory data. WPS/OLPARS offers the tools for easy analysis of both waveform and vector data and efficient design of a logic network for classifying the data.

SYNTACTIC PARSING

WAVAN Module

Introduction. WAVAN is a syntactic pattern learning/classification subsystem for processing one-dimensional signal waveforms. These waveforms are essentially strings of symbols drawn from a given alphabet (like RISE, FALL, LEVEL), each with associated parameters (like intensity, frequency, duration, decay time) derived from such routines as KLASIT and WAMSR, discussed in this report in the Section on VU.

Figure 27 shows the subroutine control hierarchy of WAVAN. By way of initialization, WAVAN fetches the current data trees, if any exist, from disk by means of LVFDECH, and replaces the updated trees by means of LVDUMP. All graph (list, tree) manipulation is performed with the subroutines of GIRS -- described in the final section of this report -- which is linked with WAVAN. The routine SETLIB then initializes DEC PDP 11/45 SYSLIB functions to handle input-output file creation and assignment. Finally, WTWAV is called to assign weights to the input string on the basis of the associated parameters, and DESCEN effectively re-orders the string symbols by re-indexing them on the basis of descending weights. At this point, WAVAN is prepared to GROW (learn) the waveform in feature/waveform trees, shown in Figures 28 and 29, or to TEST (classify) the input waveform against feature/waveform data already stored for a predetermined set of files.

Theory of Operation. Currently, the GROW option is invoked by a call to WAVAN, and TEST is called separately. However, it is anticipated that both options will be implemented as functions within WAVAN and may then be reached by commands and parameter changes within SELECT. Details of GROW and TEST are given, from a user's viewpoint, in the sections which follow, and a more detailed system description of the routines is given in the ISPARS Program Description report.[1]

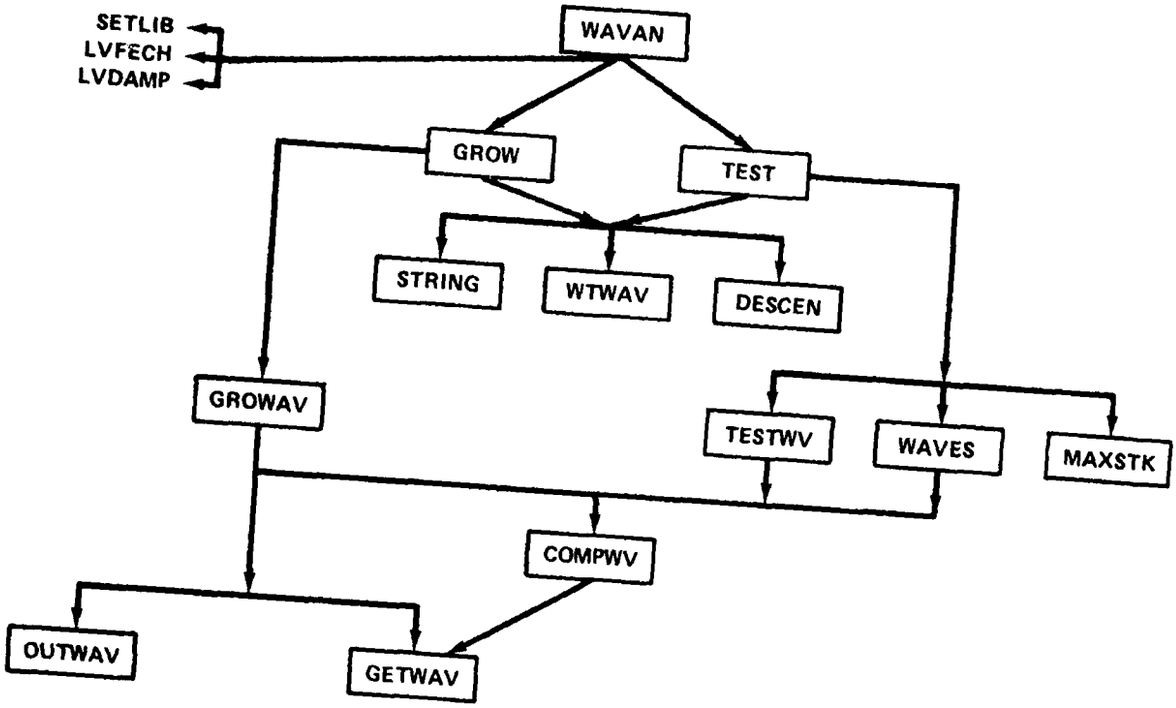


Figure 27 - WAVAN Subroutine Control Hierarchy with Modifications

The GROW Function.

Command: RUN DPO:WAVAN

Description: In the GROW option, a feature string re-indexed by weight is scanned and an accumulated confidence CRSA in the distinctive character of each substring is calculated. For each substring feature whose CRSA exceeds a threshold THRSHA, the feature effects a feature transition in the feature tree and the string is grown on the tree (Figure 30) and associated with a class identification number, the average value of CRSA, and the average temporal location (i.e., the unmapped index) of the feature. The total waveform and its associated parameters are then grown by GROWAV in a waveform dictionary tree (Figure 31) as a class entry. In order to determine whether the proposed string requires a new entry or whether it is similar enough to a previous entry to be averaged with that entry, COMPWV is called to perform a controlled expansion/contraction parse of the proposed string and to compare it with previous waveform entries of the subject class. Accesses from and to the actual waveform and parameter data on disk are performed by the routines GETWAV and PUTWAV, respectively.

Parameters:

[A(1), A(2)]	is the weight sub-vector of the feature confidence calculation. The weights are applied to the tree feature confidence ICRSC and the sample feature weight WT, respectively.
[A(5), A(10), A(11)]	is the weight sub-vector of the substring confidence. The weights are applied to the sample feature weight WT, the tree feature confidence ICRS, and the substring reliability IREL, respectively.
COEF( )	is the weight vector of the waveform match score. The weights are applied to level, decay, and difference level similarities.
GRWTHS( )	is a vector threshold on the match score variables mentioned for COEF. The thresholds regulate decisions on whether local expansion or contraction of the waveform is necessary for the match.

SCRTHS	is a threshold on the match score. New waveforms are merged with old if the match score is sufficiently low; otherwise, they are grown anew in the waveform tree.
THRSA	is the threshold of confidence in a feature that determines whether or not a feature transition is to be executed.

Input/Output:

CREATE	is input as .T. if a data tree create run is desired, .F. if update
KLAS( )	is the input matrix containing the feature string and associated measurement parameters
NFEAT	is the input number of features in KLAS
NWORD	is the input index of the waveform to be grown
TIMES	is the input number of the waveform to be learned
TREE	is the input/output random number associated with the root node of the feature tree
WORD( )	is the input/output random vector associated with the root nodes of the waveform trees. Each tree represents a signal string class.
WT ( )	is the input empirical weight vector associated with the feature string

The TEST Function.

Command: RUN DPO:TEST

Description: In the TEST option, a feature string re-indexed by weight is scanned and an accumulated confidence CRSA in the distinctive character of each substring is calculated. For each substring feature whose CRSA exceeds a threshold THRSHA, the feature tree (Figure 27) is traversed by that feature transition and its alternates, if any. All sink nodes of the transitions are stacked in descending CRSA order. For each node, an associated signal (or waveform) list is examined for matching wave candidates by computing reliabilities IRELT and comparing them with a threshold THRSHB. If no valid candidates are found, those signals with IRELT greater than a threshold THRESHC are

placed on a potential candidate list. If the valid candidate list length exceeds WTHRESH, an attempt to reduce the length will be made before outputting the list by comparing the input with the waveform trees of the candidate signals. The potential candidate list will be used to supplement the valid list if the length of the latter is less than WTHRESH. If there are no valid candidates at any stacked node, a brute force match of the input against the entire waveform tree is attempted.

Parameters:

[A(1), A(2)]	is the weight sub-vector of the feature confidence calculation. The weights are applied to the feature tree confidence ICRSA and the sample feature weight WT, respectively.
[A(6), A(7), A(8), A(9)]	is the weight and sub-vector of the substring confidence. The weights are applied to the tree feature confidence CRSA, the substring reliability IREL, the sample feature weight WT, and the index difference, respectively.
COEF( )	is the weight vector of the waveform match score. The weights are applied to level, decay, and difference level similarities.
GRWTHS( )	is a vector threshold on the match score variables mentioned for COEF. The thresholds regulate decisions on whether local expansion or contraction of the waveform is necessary for the match.
SCRTHS	is a threshold on the match score, used to reduce a candidate list whose length is greater than WTHRES.
THRSHA	is the threshold of confidence in a feature that determines whether a feature transition is to be executed.
THRSHB	is the threshold of confidence in the signal candidate that determines whether the candidate identifier is placed on the valid candidate list CAND.
THRSHC	is the threshold of confidence in the signal candidate that determines whether the candidate identifier, having failed to make the CAND list, qualifies for the TEMP list of potential candidates (THRSHC<THRSHB).

THRSHT	is a threshold on the match score, used to produce a candidate list from a brute-force search of the waveform dictionary.
WTHRESH	is the maximum number of signal candidates accepted for the final valid candidate list.

Input/Output:

CONMAT( )	is the output vector of valid candidate confidences
KLAS	is the input feature string and associated parameters
MATCH( )	is the output vector of valid candidate identifiers
MATES	is the ouput number of valid candidates
NFEAT	is the input number of features in KLAS
TIMES	is the input number of waveforms to be tested
TREE	is the input random number associated with the root node of the feature tree
WORD	is the input random vector associated with the root nodes of the waveform trees. Each tree represents a signal string class.
WT	is the input empirical weight vector associated with the feature string

Program GENER8.

Purpose. GENER8 is a fairly portable FORTRAN program with which one may conveniently implement translators for high-level computer languages. It has been used to generate syntax graphs for a FORTRAN auditing compiler and for a GIRL preprocessor. The following description is designed to introduce GENER8 to users not generally familiar with parsing and compilation. Each particular type of computer executes instructions in its "machine language," a means of communication dictated by the unique structure of the machine (Figure 28). It is beneficial to allow the programmer writing these instructions to do so in a (higher-level) language more closely resembling English (Figure 29). Doing

this necessitates the creation of a translator since the machine cannot directly understand a higher-level language. Since the computer can be instructed to manipulate the characters of a higher-level language, it is possible to instruct the computer to perform the translation itself (Figure 30).[10] GENER8 is used to create such translators.

Description. A language consists of a set of sentences which can be created by combining words or symbols of the language, called terminal symbols, in accordance with a set of rules which constitute the syntax of the language. A convenient method for specifying these rules is the Backus-Naur Form (BNF). In this form the terminal symbols are combined in syntactic rules, called productions, to form nonterminal symbols (designated by their enclosure in angle brackets <>) which in turn are used to form other nonterminals. One nonterminal, the distinguished entry symbol, must appear on the left hand side of at least one production, indicating the parent nonterminal from which all sentences are to be derived. As an example here is the BNF for a syntax describing the language of sums and products of nonegative integers.

```
<TERM>    ::= <PRODUCT>
<TERM>    ::= <PRODUCT> + <TERM>
<PRODUCT> ::= <PRIMARY>
<PRODUCT> ::= <PRIMARY> + <PRODUCT>
<PRIMARY> ::= <DIGIT>
<PRIMARY> ::= <DIGIT> <PRIMARY>
<DIGIT>   ::= 0|1|...|9
```

When more than one definition exists for a non-terminal, as is the case here for <TERM>, the syntax is called non-deterministic. In the sample syntax, there are also alternate forms of <PRODUCT> and <PRIMARY>. On the other hand, the ten alternaties of <DIGIT> are represented on one line with the alternates delimited by the bar "|".

The purpose of a translator is to effectively exhibit the syntactic structure, the parse, of any legal sentence of a given syntax in such a way that actions, called semantics, can be executed at appropriate junctures of the parse. Generally, the semantics consist of translations into a lower-level language which the computer is more readily equipped to use.

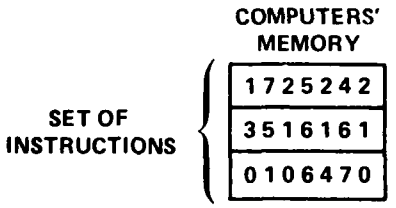


Figure 28 - Machine Language Instructions

```
READ, HOURS-WORKED, PAY-RATE
TOTAL PAY = HOURS-WORKED X PAY-RATE
PRINT TOTAL-PAY
```

Figure 29 - Higher-Level Language Instructions

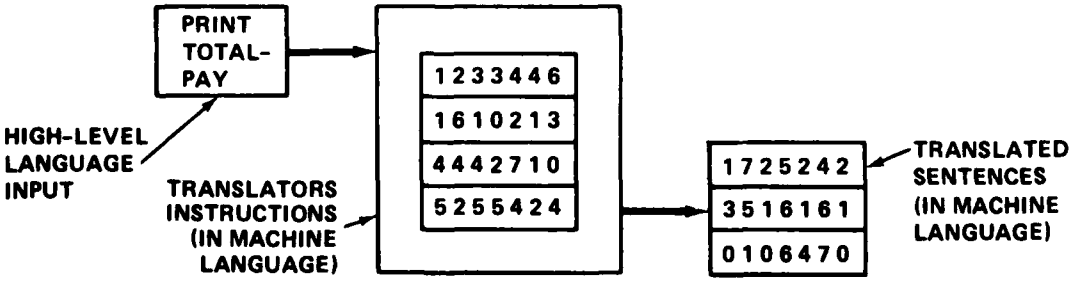


Figure 30 - Translating Higher-Level Sentences

An example of a parse for a specific <TERM> is shown in Figure 31.

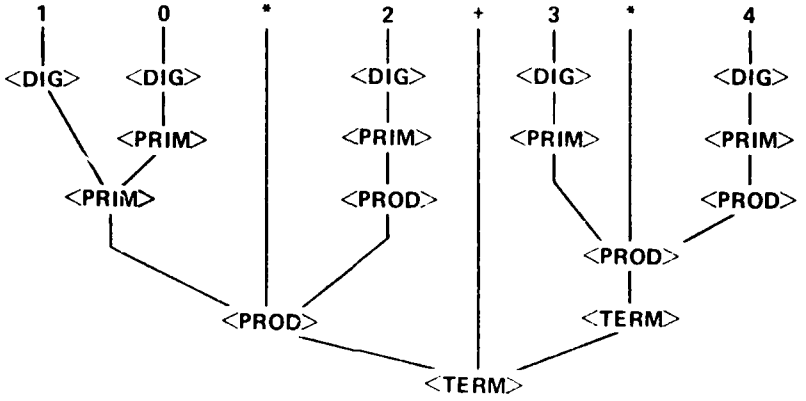


Figure 31 - A Sample Parse.

An instance of an illegal string in the sample snytax - that is, a string which does not parse to a <TERM> - is 10+, as the reader can easily verify.

In order to recognize strings which are legal (<TERMS> in the example) and to flag those strings which are not syntactically correct, a program called RECOG is available. This program is a syntax recognizer which requires that a suitable form of the syntax reside in memory. The required form is called a syntax graph (or recursive transition network). The nodes of the graph represent partial parse states in which the recognizer is said to reside at any instant. Each syntax graph must contain a start state, in which the recognizer resides before examining the input string, and at least one final stop state (represented by a pound sign "#") at which a valid string may end. Arcs from node to node are labeled by terminal symbols in the language. The recognizer will pass from one state to a second state if the terminal it is presently examining consititutes a label of the arc connecting the two states.

A state with a "stop" (#) is a final state. Figure 32 shows one possible syntax graph for <TERM>:

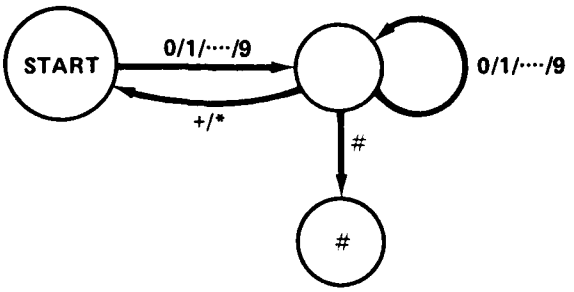


Figure 32 - A Sample Syntax Graph

This graph is derived from a syntax different from the previous syntax for <TERM>, namely:

<TERM> ::= <PRIMARY>¶<PRIMARY> + <TERM>¶<PRIMARY> * <TERM>
<PRIMARY> ::= <DIGIT> <DIGIT><PRIMARY>
<DIGIT> ::= 0¶1¶...¶9

The favorable characteristic of this graph is that it represents a minimum processing time recognizer for <TERM>. On the other hand, it consumes more memory than absolutely necessary and makes semantic interpretation of operator precedence difficult. Transformations of the graph for purposes of improving tradeoffs among these factors are discussed in the following paragraphs in connection with the use of pseudoterminals to conserve memory.

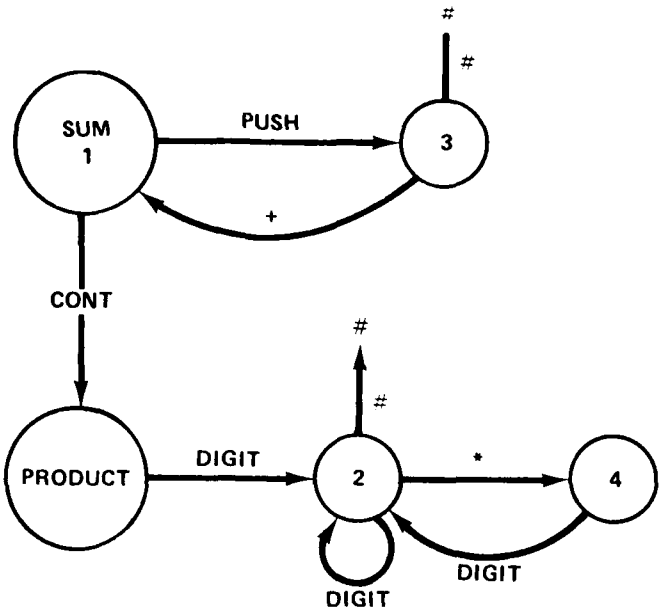
The specific function of GENER8 is to create a syntax graph for use by the recognizer RECOG in accepting or rejecting strings. GENER8 is written in the programming language GIRL/FORTRAN* for the PDP 11/45 computer and can be executed in batch or interactive mode. GENER8 is portable since GIRL pre-processors have been written for the IBM 360/370, the UNIVAC 1108, and the CDC

*GIRL stands for Graph Information Retrieval Language.[11] A syntax graph for use in accepting GIRL strings has been created, and a GIRL translator has been implemented using GENER8.

o0000. A knowledge of GIRL or FORTRAN is not required to generate syntax graphs using GENER8. However, in either batch or interactive mode, it is convenient (although not necessary) to make modifications in GIRL.

In some applications, merely accepting or rejecting a string is not enough. The semantics of the string must be used in the form of arithmetic computations or symbolic manipulations explicitly described or executed in a language understood by the computer at hand. Syntax graphs as produced thus far can be used to decide whether an input string is a valid statement in the language corresponding to the graph. In order to extend the recognizer to a translator the recognizer must interpret the semantics of the terminal symbols it is examining. This is accomplished by including in the evolved syntax graph "ACTIONS" at states requiring them. These semantic actions are pointers to a set of semantic routines which are called by the recognizer at the appropriate time. The "ACTION" pointers are arcs in the syntax graph which point to numbers: "1" for the first semantic routine, "2" for the second, and so on. The user then writes a FORTRAN program called SEMANT and, at labels "1", "2", etc., he places the corresponding FORTRAN code which will execute the desired semantic action. Figure 33 shows a syntax graph for a <SUM> grammar and the semantics routines necessary to evaluate these simple integer expressions. The numbers at the state nodes indicate the semantic action to be taken when the recognizer RECOG reaches that state.

Figure 34 is a graphic flow diagram of the overall process by which a syntax graph with possible semantic references is created. Problems which may be encountered are noted. The flow is as follows: The BNF of the syntax must be prepared as input. It is then entered in either the batch or conversational mode. If GIRL memory is exceeded and cannot be extended, then either the BNF must be rewritten and/or the syntax must be partitioned (by means of PSEUDOTERMINALS). At this point, either the graph has been created or there exists a GIRL program (compact option) which represents the graph. In the latter case, the program which represents the graph must be preprocessed, compiled, and executed to create the graph. If a translator is to be created, semantic "actions" must now be inserted into the syntax graph and semantics routines prepared. Finally, a lexical scan must be written to prepare input high-level language strings for the recognizer program (RECOG) to accept or translate.




```
      SUBROUTINE SEMANT (N, FAIL)
      LOGICAL *1 FAIL
      INTEGER STRING (100), SUM, PRIM, PROD, PNTR
      COMMON/PARMS/STRING, SUM, PNTR
C     PNTR AND SUM HAVE BEEN INITIALIZED (=0) PRIOR TO CALL
C     TO RECOG(NIZER). STRING CONTAINS THE INTEGER VALUES
C     ENCOUNTERED IN THE INPUT EXPRESSION, PLACE THERE BY
C     A LEXICAL SCANNER. FAIL IS A BOOLEAN VARIABLE WHICH
C     IS INITIALIZED AND CAN BE SET (= .TRUE.) TO CAUSE A
C     SEMANTIC FAILURE WHICH WILL FORCE THE RECOGNIZER
C     TO BACK UP.

          FAIL =. FALSE.
          GO TO (1,2,3,4),N
C     INITIALIZED FOR NEXT TERM (SUM HOLDS THE VALUE OF
C     PREVIOUS TERM)
        1 PROD = 1
          PRIM = 0
          RETURN
C     ESTABLISH VALUE OF INTEGER PRIMARY
        2 PNTR = PNTR +1
          PRIM = 10* PRIM + STRING (PNTR)
          RETURN
C     PRODUCT HAS BEEN FORMED. ADD IT TO SUM
        3 SUM = SUM + PROD*PRIM
          RETURN
C     FORM PARTIAL PRODUCT IN ANTICIPATION OF RECEIVING
C     NEXT FACTOR
        4 PROD = PROD*PRIM
          PRIM = 0
          RETURN
```

Figure 33 - Syntax Graph and Semantics for an Interpreter of Sum and Products of Non-negative Integer Constants

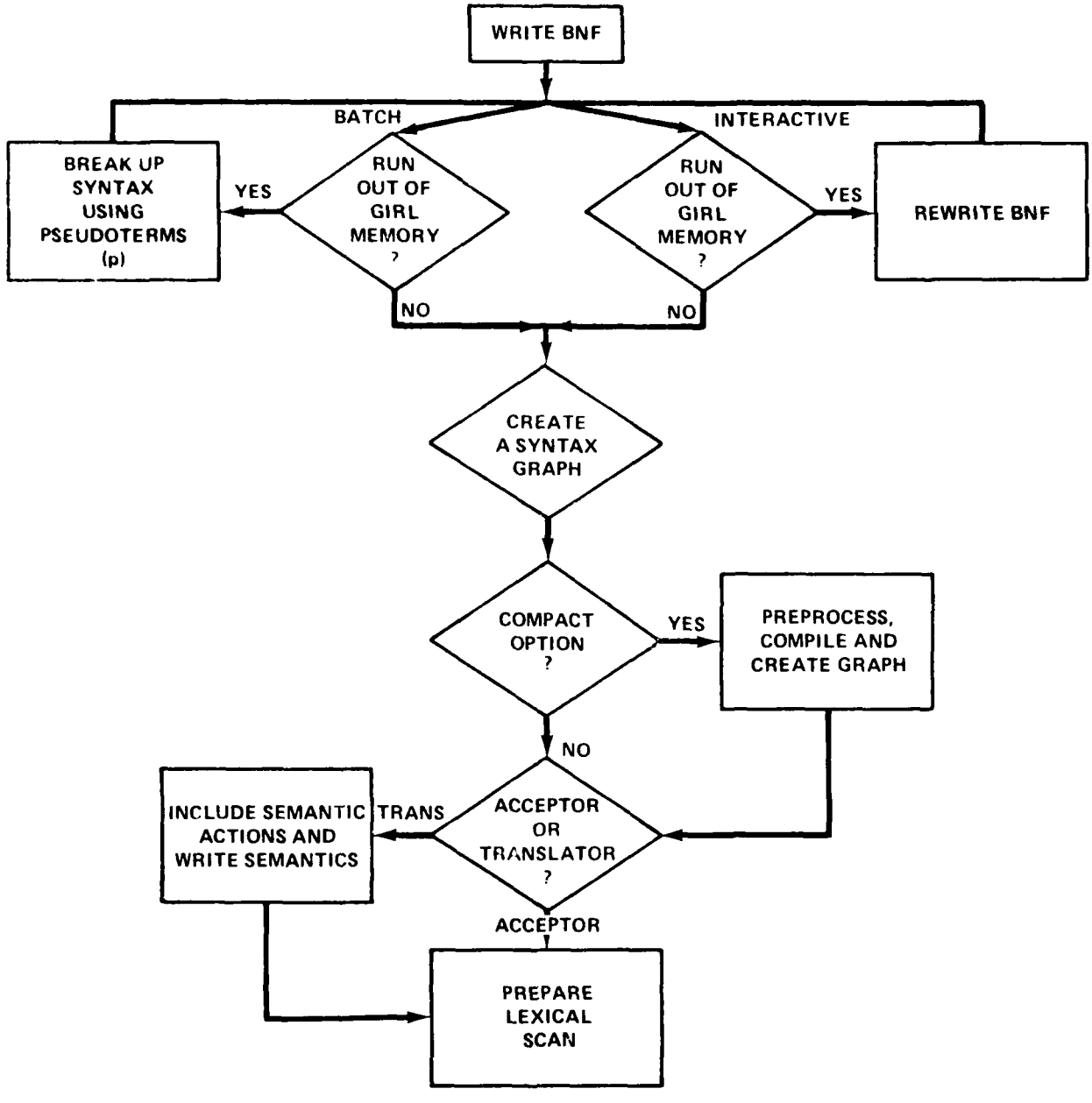


Figure 34 - Method for Creating and Executing a High-Level Language Translater

To conserve memory, the user may wish to manipulate the grammar by introducing "PSEUDO"-terminal symbols when GIRL memory size restrictions have been exceeded so that the complete syntax graph must be partitioned into subgraphs, each to be generated in a separate run from a subgrammar. Even for an in-core graph, memory requirements may be lowered by producing a smaller graph. The need for PSEUDO-terminals arises as follows: The subgrammar of a nonterminal (say T) is given by the productions defining T and the productions of all nonterminals in the productions defining T, and so on. If a subgrammar (i.e., its defining nonterminal symbol) appears in several productions of the grammar, the syntax graph generator will proliferate its subgraph throughout the graph at all nodes requiring it. In some cases this can result in the use of a large portion of graph memory for identical subgraphs. There exists, therefore, an option in GENER8 by which the user can choose either to have a subgrammar proliferated at any or all required nodes, or to generate only one subgraph and to make a closed reference using a PSEUDO-terminal to this subgraph at any node previously requiring its proliferation. In return for savings in memory this procedure will result in a slightly longer recognition time.

In order to discuss the implementation of the nonproliferation mechanism, we must extend the sample grammar thus far examined. The language generated by the sample grammar belongs to the class of languages called regular languages. Any such language requires a finite state automaton to recognize strings in the language. (The syntax graph and a small portion of RECOG constitute such a finite state automaton.) A more general grammar of interest to users of GENER8 generates languages called context-free. A context-free language requires a more complex automaton, i.e., a nondeterministic pushdown automaton, to recognize strings of the language. (See Hapcroft and Ullman[12] for a discussion of the relationships between classes of languages and the types of machines necessary to reco nize them.)

For non-proliferation of a subgrammar, the generator will save one copy of a subgraph and, at nodes at which copies of this subgraph would have been proliferated, a structure is set up as follows:

Line "CONT(inue)" to the copy of the subgraph (for which the pseudo terminal is the entry node)

Line "PUSH" to the next node (which is the return node from the subgrammar).

In the grammar defining <SUM>, for example, if the definition of <PRODUCT> were to be deferred, and <PRODUCT> were to be regarded as a terminal, or more appropriately a pseudo-terminal, the GIRL version of the BNF would have the form:

```
G  PLUS TYPE TERMIN
G  SUM TYPE NONTRM
G  PRODUCT TYPE PSEUDO
G  SUM ALTERN ($ DEFINS PRODUCT, $ DEFINS (PRODUCT,PLUS,SUM))
```

An example of the result of proliferation and non-proliferation using the <SUM> subgrammar is shown in Figures 35 and 36, respectively.

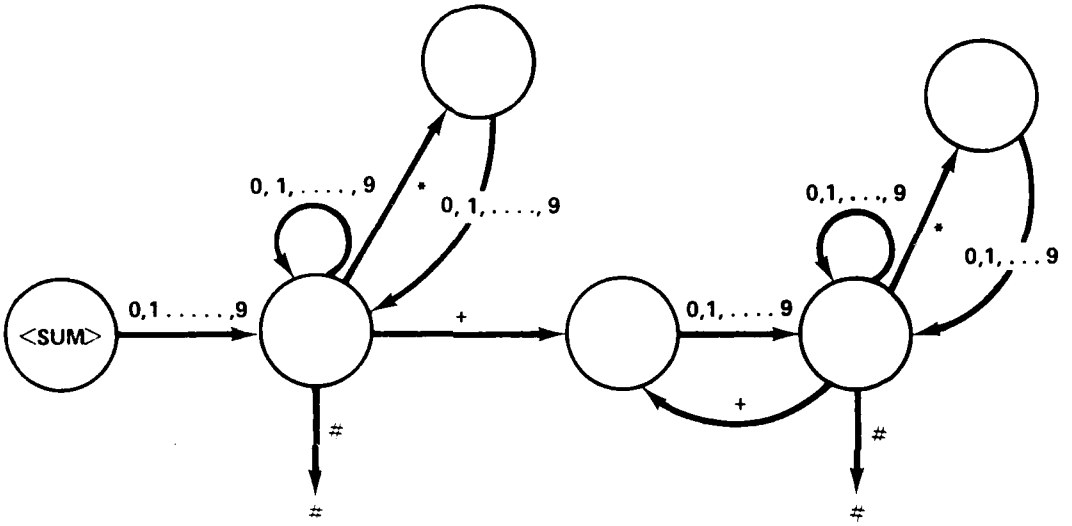
Making modifications to a language is sometimes a difficult and time-consuming effort. Use of the syntax graph generator and the GIRL/GIRS system allows language designers to easily picture (in graphic form) the operation of the language recognizer. Separation of the syntax description from the parser enables deletions or additions to the language to be readily accomplished through changes to the syntax graph. See Berkowitz[11] for a description of the GIRL language used to alter the syntax graph.

Figure 37 shows the graph modification and semantics added to the graph of Figure 33 to allow parenthesized expressions to be computed.

Input. The batch version of GENER8 requires the user to supply a valid GIRL/FORTRAN program of the form described in Figure 38. Lower case letters indicate card images that are either described after the setup or are user dependent. Upper case letters refer to character images. All system card images begin in Column 1. Those which are indented refer to FORTRAN card images whose code begins in Column 7. The card number is for reference only and need not be typed.

In the GIRL/FORTRAN program, GIRL statements are declared by punching a G in Column 1. Continuation cards are handled as in FORTRAN.

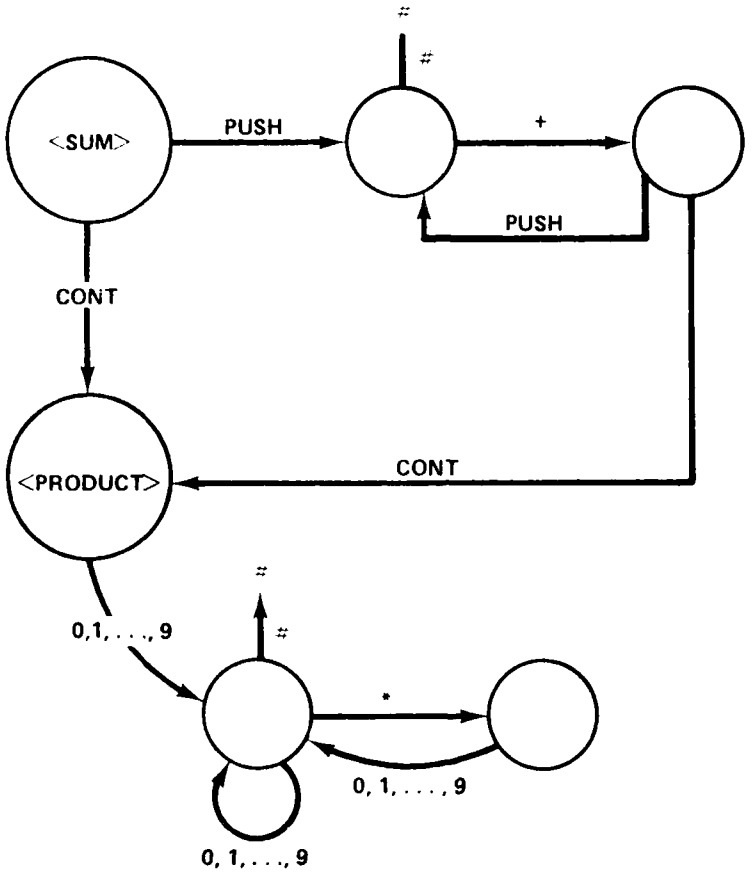
The conversational version of GENER8 requires the same information as the batch version but the user, working interactively with the program, is prompted to provide the necessary data. The same input data represented in cards 11, 13, and 14 of the batch version of GENER8 are requested by the



[NOTE: 66 LINKS ARE USED]

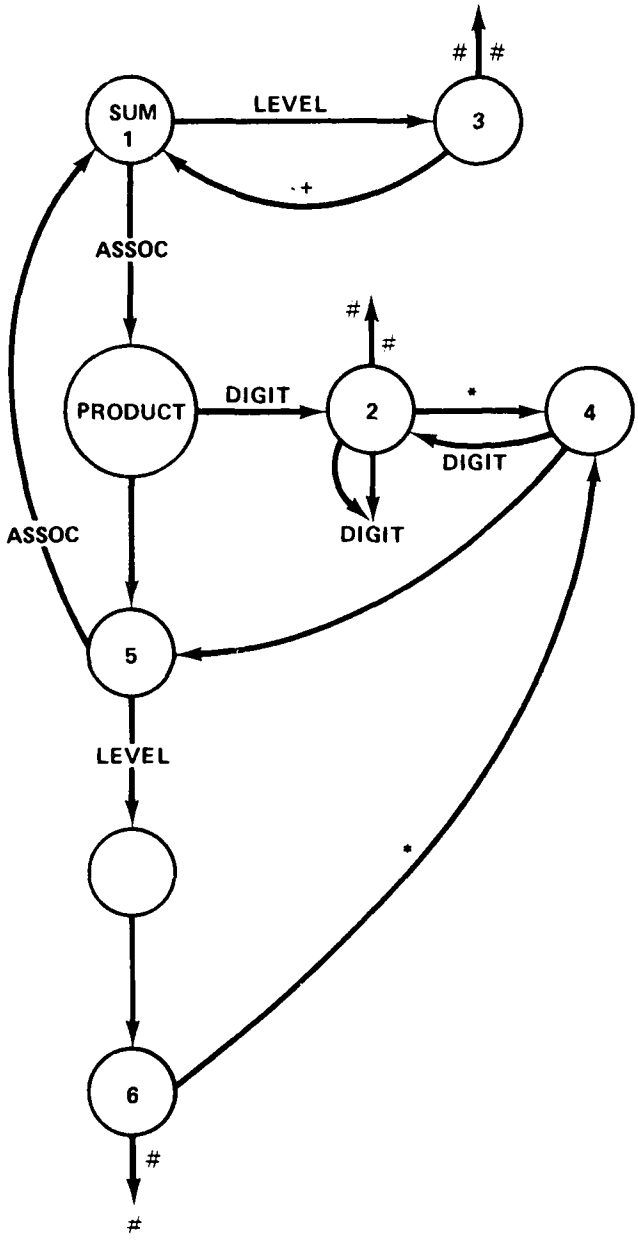
```
<SUM>     :: = <PRODUCT>|<PRODUCT>+<SUM>
<PRODUCT> :: = <PRIMARY>|<PRIMARY>*<PRODUCT>
<PRIMARY> :: = <DIGIT>|<DIGIT><PRIMARY>
<DIGIT>   :: = 0|1|...|9
```

Figure 35 - Proliferation of PRODUCT in the Grammar



[NOTE: 38 LINKS ARE USED]

Figure 36 - Breaking Grammar ⟨SUM⟩ to Save One Copy of ⟨PRODUCT⟩ Subgraph




```
C   STACK OLD PROD AND SUM WHILE
C   INTERPRETING SUM IN PARENTHESIS
    5 STKPTR=STKPTR+1
      STACK (STKPTR,1) = SUM
      SUM = 0
      RETURN
C   PARENTHESIZED TERM HAS BEEN
C   INTERPRETED. ONSTACK PREVIOUS
C   SUM AND PRODUCT
    6 PRIM=SUM
      SUM=STACK(STKPTR, 1)
      PROD=STKPTR-1
      RETURN
```

Figure 37 - Extending ⟨SUM⟩ to Include Parentheses

```
                                                                  Line No.
      nnnnnn PACK NOSAVE                                             1
      COMMON/GIRL/NTERMS,T1,T2,T3,...                                2
      COMMON/LINKS/ALTERN,DEFINS,TYPE,NONTRM,TERM,NULL,P1            3
      COMMON/PSU/PSEUDO,START,HOLLER,ACTION                          4
      COMMON/WRIT/WTEST,COMPOP                                       5
      non-DATA specification statements                              .
G     DEFINE ALTERN,DEFINS,TYPE,NONTRM,TERM,NULL,PSEUDO,             6
      *     HOLLER,ACTION,T1,T2,T3,...,N1,N2,N3,...                  .
      DATA statements if any                                         7
G     EXECUTE                                                        8
      CALL ASSIGN (99,'RKO:SYNTAX.GRF',14)                           9
      CALL ASSIGN (10,'RKO:COMPOP.GRL',14)                           10
      READ 1, NTERMS, P1,WTEST,COMPOP                                11
1     FORMAT(15,A1,2L7)                                              12
           or optional data assignments                              .
G     T1 TYPE TERMIN                                                 13
G     T2 TYPE TERMIN                                                 .
           .                                                         .
           .                                                         .
G     N1 TYPE NONTRM                                                 14
G     N2 TYPE NONTRM                                                 .
           .                                                         .
           .                                                         .
G     GIRL statements for language production rules                  15
           .                                                         .
           .                                                         .
      CALL GRAPH                                                     16
      User desired code (no END statement)                           .
G     COMPLETE                                                       17
      other GIRL/FORTRAN routines if any                             .
/     COMPLETE                                                       18
      purely FORTRAN routines if any
```

Figure 38 - Batch Mode GENER8 Program for PDP

NOTES:

CARD (1). The user supplies an integer in I6 format a in the first 6 columns equal to the expected memory size required to generate the syntax graph.

CARD (2). COMMON/GIRL/ contains a variable, e.g., NTERMS, to store the number of terminal symbols in the syntax and also a list of the user specified FORTRAN variables (here T1,T2,...) associated with each terminal symbol in the alphabet.

CARD (5). WTEST is specified as .TRUE. if a printed description of the syntax graph is desired. The description is used in drawing the graph. COMPOP is specified as .TRUE. if a printed description of the syntax graph is desired. The compact form is achieved by a GIRL program produced on RKO:COMPOP.GRL which creates a new graph devoid of the overhead (graph structures) used to produce the original graph. The new program calls several routines in the GENER8 package to resolve possible non-determinism in the compact graph. COMPOP must be .TRUE. if PSEUDO terminal symbols are used to reduce the memory requirements of the syntax graph as described on page 119.

CARD (6). The GIRL DEFINE statement lists certain reserved words, the variables T1,T2,..., assigned to the terminal symbols, and also the user-specified FORTRAN variables, N1,N2,N3n..., associated with the non-terminal symbols of the syntax.

CARD (11) P1 is set equal to the distinguised entry symbol of the language.

CARDS (13) and (14) Similar statements are required for each terminal and non-terminal FORTRAN variable, respectively.

CARDS (15) Each production in the syntax must be translated to a GIRL statement as in the following example:

The production <TERM> :: = <PRODUCT>¶<PRODUCT>+<TERM>

becomes

```
G    TERM ALTERN($ DEFINS PRODUCT,$ DEFINS(PRODUCT,PLUS,TERM))
```

That is, the definition of <TERM> has a list of two alternatives, of which the first contains a definens of one symbol: PRODUCT; and the second contains a definens of three symbols: PRODUCT, PLUS, and TERM.

Figure 38 - Batch Mode GENER8 Program for PDP (Cont'd)

program one by one. Interactive GENER8 then requests each production rule information by sequencing through the terminals, non-terminals, and pseudo-terminals and requesting the number of alternates and the definitions (DEFINS) of each. This version has the advantage of not requiring terminal symbols to be represented by FORTRAN variables (e.g., PLUS for "+").

Output. Output from both versions consists of a printed description of the syntax graph (if WRIT was .TRUE.), the syntax graph on disk file RKO:SYNTAX.GRF, and a GIRL program on disk file RKO:COMPOP.GRL (if COMPOP was .TRUE.) which when executed will result in a compact graph on disk.

If the printed output contains the message "USED LAST CELL OF AVAILABLE SPACE," then the GIRL memory specified on the GIRL option card (CARD (1) Figure 38) is too small and must be increased or the grammar must be split using pseudo-terminal symbols. Additional memory can be gained by the use of the virtual memory scheme for graph storage called Paged GIRS.

Subroutines Called. GENER8 is written in GIRL/(Graph Information Retrieval Language)/FORTRAN and uses a run time library called GIRS (Graph Information Retrieval System). The GIRL/GIRS system provides a simple yet elegant tool for the manipulation of graph structures such as syntax graphs. The conversational version of GENER8 eliminates the need for the user to have a knowledge of GIRL.

Operating Instruction. If the GIRL program outlined in Figure 38 is assumed to be stored on the disk file DP0:MAKGRF.GRL, the necessary steps for preprocessing, compiling, linking, and executing the program to obtain the syntax graph are contained in the following PDP 11/45 commands:

```
. RUN PREPOC
* DP0:MAKGRF
. R FORTRA
* DP0:MAKGRF=DP0MAKGRF
. R LINK
* DP0:MAKGRF=DP0:MAKGRF,GENER8,GIRS/F
. RUN DP0:MAKGRF
```

where

RKO:PREPOC.SAV is the GIRL preprocessor
DP0:GENER8.OBJ is the object file for the generation subroutine
DP0:GIRS.OBJ is the object file for the GIRS subroutine library.

Program RECOG

Purpose. The syntax graph-driven parser RECOG performs a syntactic analysis of the input string (a set of tokens provided by a lexical scan of the original string) to establish the legality of the string. The tokens are GIRS pseudo-random numbers which represent the terminal symbols in the grammar input to the syntax graph generator program GENER8, described just previously in this report. The parser RECOG in effect combines the terminal symbols represented by the tokens to form non-terminal symbols which correspond to the production rules in the original BNF of the grammar. This process is termed a "bottom-up analysis." In addition to its performance as a syntactic analyzer or recognizer, RECOG also executes semantics attached to nodes of the syntax graph, thus also acting as a translator.

Description. RECOG examines the lexical scan a token at a time and advances from state node to state node in the syntax graph when the token exists on a link between the nodes. Semantic routine references are executed when RECOG encounters them attached to a node by an ACTION pointer. The semantic references are to line labels of a FORTRAN subroutine SEMANT, to be provided by the user. When a node is reached at which a subgrammar would have been proliferated (see the description of GENER8 for details), the recognizer moves to the entry node (pointed to by the CONT link) of the one copy of the subgraph and the corresponding PUSH link represents the completion of the string after parsing the saved (CONT-linked) subgrammar. The recognizer places the PUSH-linked entry node on a push-down stack. When the saved substructure is correctly parsed (i.e., a final state in the subgraph is reached), the recognizer pops its stack and moves to the state which would have been entered when the substructure was parsed had the subgraph been proliferated. If the scan is incorrect, a backtracking mechanism tries another potential path in the syntax graph, if such exists.

Input. RECOG requires as data the syntax graph contained on disk file SYNTAX.GRF (as generated by GENRE8) which describes the grammar against which input statements are to be parsed. The graph and its associated parameters are read into GIRS memory and RECOG COMMON blocks by the following call to a GIRS input routine and the accompanying read statement:

```
CALL LVFECH (N)
READ (N) START,ASSOC,LEVEL,STOP,NODE,LINK,NTERMS,ACTION,
         STRING, (NNN(I),I=1,NTERMS)
```

where N is the logical unit number assigned to the file.

The statement to be parsed is input to RECOG in the form of an ordered list of terminal symbols organized by the lexical scanner into a GIRL list structure in GIRS memory. The Ith item from this list is retrieved by RECOG using the following GIRL statement:

```
G  STRING + STRING.I ' STJ
```

Output. RECOG indicates whether the tested statement is syntactically correct by the value of its only subroutine argument, a logical variable set to .TRUE. only if the statement parses. RECOG also implements semantic action routines supplied with the grammar to effectively translate the input statement.

Subroutines Called. RECOG is coded in GIRL/FORTRAN and uses a run time library called GIRS (Graph Information Retrieval System).

GRAPH INFORMATION RETRIEVAL LANGUAGE/GRAPH INFORMATION RETRIEVAL SYSTEM
GIRL/GIRS

Introduction. The Graph Information Retrieval System (GIRS) provides a convenient and efficient technique for the insertion, retrieval, modification, and deletion of data in a data base. This technique is based on a scheme of representing the various data items as nodes and establishing the relationships between the nodes by linking them together into node-link-node triples which are assembled into a graph that can be stored on disk. The Graph Information Retrieval Language (GIRL) is a high-level programming language designed to manipulate data in graph structures. GIRL statements are

interleaved with FORTRAN statements in GIRL programs and prior to execution are preprocessed into FORTRAN statements containing references to GIRS subroutines.

There are two versions of GIRS. The first contains the entire graph in a buffer in main memory and is referred to as "In-Core" GIRS. The second version, referrred to as "Out-Core" or "paged" GIRS, contains a graph which is physically divided such that only a portion of it resides in a main memory buffer and the rest is stored on disk to be brought in as needed. The general features and distinctions of In-Core and Out-Core GIRS will be summarized below. Detailed descriptions of the systems may be obtained from the design document by Berkowitz[13] or from the implementation and user's guides by Zaritsky.[14,15]

In-Core GIRS. GIRS has been implemented on the CDC 6700, the PDP 11/45, and the UNIVAC 1108 systems and can easily be adapted to other machines having FORTRAN IV compilers. It is a versatile scheme for both representation and manipulation of data relationships. Both versions provide several notable capabilities:

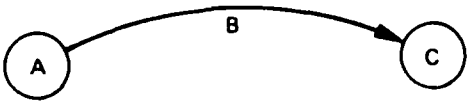
- Placement of a value at any location within a multivalued list (nondestructive insertion)
- Substitution of a value within any location within a multivalued list (destructive insertion)
- Deletion of a value within any location within a multivalued list (indexed* deletion)
- Retrieval of a value from any location within a multivalued list
- Retrieval of the index associated with a particular sink node value (submitted by the user) within a multivalued list
- Reduction of retrieval time needed for long lists through the use of internally saved indices
- Adjustment of the GIRS buffer size resulting in a more efficient use of computer space
- Convenient disk storage and retrieval of entire graphs

GIRS is a hashed-address associative memory scheme composed of several FORTRAN subroutines. GIRS enables data relationships to be expressed in an

*The index of a value is the position it occupies within the list.

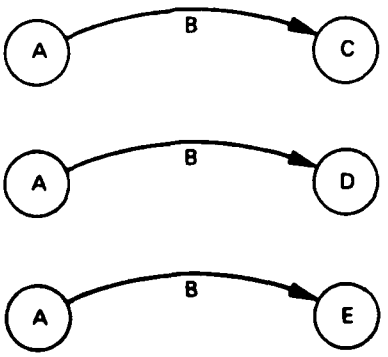
arbitrarily directed (plex) data structure known as a graph. After a graph has been created, GIRS can be called upon to store the graph on disk and fetch it from disk.

In GIRS, information is stored conceptually as a set of primitive structures called node-link-node triples:

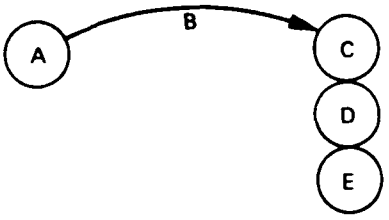


A Node-Link Node Triple

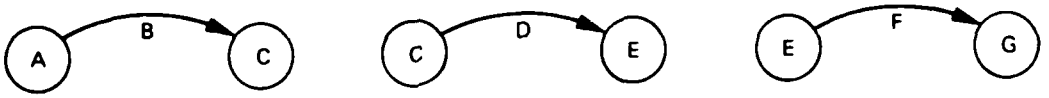
Links may be thought of as pointers, arcs, edges, associations, or functions. Nodes may also be thought of as beads or points or as arguments of those functions. The function in the triple A, B, C would be B(A). In this example, the node A is the source node (the argument) and the node (C) is the sink node (the value). Each triple represents a single relationship in which the source node is said to be associated with the sink node via a link. The collection of all of the relationships (triples) makes up the graph. Complex relationships can be expressed by combining these triples in various ways. For example, the triple can be a component of a list (also known as a multivalued list (MVL)),



alternately drawn as

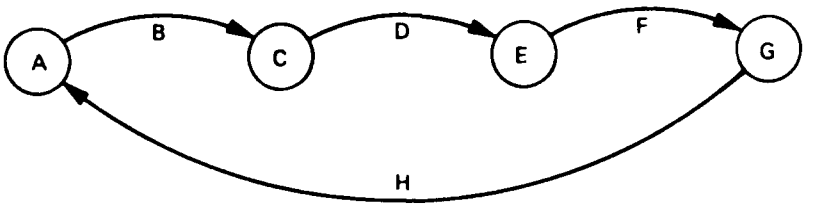


or it can be a component in a <u>string</u>



alternately drawn as

With the addition of the triple G,H,A this string would become a circuit.



A function can refer to either a single-valued list (SVL) or to an MVL.

A GIRS-type structure is useful for indicating relationships among data blocks. For example, if POINTERA were used to associate BLOCK1 with BLOCK2, the relationship would be represented conceptually by the structure

in which BLOCK1 is the source node, POINTERA is the link, and BLOCK2 is the sink node or value. Source nodes and links are represented by random numbers and sink nodes may be of the following types:

- Random numbers representing defined variables
- Integer data
- Hollerith data

The latter two types must always be terminal points in the graph.

Out-Core GIRS. Out-Core GIRS is an expansion of In-Core GIRS, and the previous description applies to both versions. However, if a user's needs should grow to the point that an In-Core buffer will not hold an entire graph, Out-Core, or Paged, GIRS should be considered.

There are several reasons for using the paged version of GIRS instead of either In-Core GIRS or some other data manipulation facility:

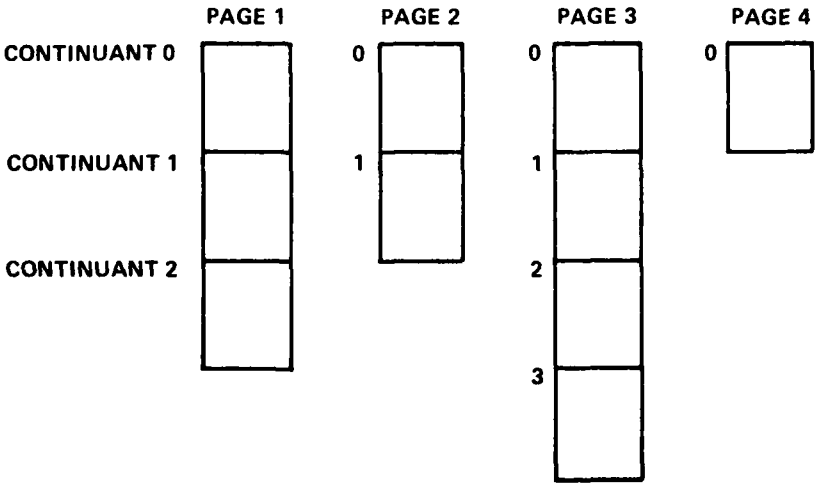
1. Large data storage capability.

2. Concurrent operation on more than one graph. Paged GIRS is ideal for shared and distributed data bases. Each user in a large data base might be assigned his or her own page to uniquely describe common data which might be stored elsewhere.[16]

3. The capability of having a user-embedded strategy. This capability allows for the inclusion of operations such as an inferential search mechanism to handle imprecise queries.

4. In the near future, a paged hardware associative memory[17] will be merged with Out-Core GIRS, enhancing the system greatly by adding high speed relational processing.

With O-Core GIRS, a graph may be segmented and placed onto as many as 63 logically and physically separate regions called pages. Pages can be extended in length as needed, (i.e., in the number of associations stored, but not in the number of addresses) by a specified increment, called a continuant. Each page, as requested, contains one or more continuants (logical records of uniform physical length) as illustrated in the following diagram:



Continuants may be used to further partition a graph or they may be used merely to hold an overflow of data from a previous continuant. It is the continuant which is swapped from disk to the GIRS buffer and back. The user determines the continuant size and also the number of continuants which will reside in the GIRS buffer. The continuant size determines the maximum number of nodes and links that may be defined for each page.

If a user chooses a buffer size to hold only one continuant and also requests just one page, the effect would be similar to In-Core GIRS except for the capability of automatic overflow to a new continuant.

A special language called GIRL (Graph Information Retreival Language)[18] is available to the user of GIRS. This programming language is designed to conveniently manipulate information in graph structures. As such, the language will play a key role in the construction of the organizational schemes found, for example, in information retrieval, pattern recognition problems, linguistic analysis, and process scheduling systems. The language is written to complement an algebraic language, in the sense that GIRL statements are distinguished from the statements of the algebraic language and the statements may be interleaved. The primary advantage of separating symbolic and numeric statements is that the programmer is afforded a linear, one-one trace of graph operations in the code description.

The function of GIRL is to insert, retrieve, delete, identify, and compare node-link-node triples. GIRL provides for indexed storage and retrieval, iteration, and labeled transfers and retains all the arithmetic power of the language within which it is embedded. Some typical GIRL code is:

A B C	Link the value C to the source node A with function B
A+B	Retrieve the value associated with source node A and function B
A-B	Remove A B C from the graph.
A+B.3'C = X /10/20	Retrieve the third item on the A+B list, name it "C," compare it to X, and go to statement 20 if equal or to statement 10 if not equal.

A version of GIRL has been implemented and operational since 1969. It has been used in syntactic parsing, pattern recognition, sparse matrix computation, information retrieval, network design, and an auditing compiler.

REFERENCES

1. Parsons, W. et al, "Program Descriptions for Interactive Signal and Pattern Analysis and Recogntion System (ISPARS)," Report DTNSRDC/CMLD-83/28 (November 1983).

2. "PDP-11 FORTRAN Language Reference Manual" Digital Equipment Corporation (June 1974).

3. FORTRAN/RT-11 Extensions Manual," Digital Equipment Corporation (June 1975).

4. "RT-11 System Reference Manual," Digital Equipment Corporation (June 1975).

5. "DR11-C General Device Interface Manual," Digital Equipment Corporation (June 1973).

6. "Lab Applications-11 System Reference Manual," Digital Equipment Corporation (October 1974).

7. Sanyal, P.K. et al, "The Waveform Processing System (WPS)," Rome Air Development Center Technical Report TR-76-224, Vol. I (Oct 1976), Vol. II (Sep. 1976), Vol. III (July 1976), Vol. IV Feb. 1977).

8. Sammon, J.W., "The On'Line Pattern Analysis and Recognition System - OLPARS," Rome Air Development Center Technical Report TR-68-263 (1968).

9. Sammon, J.W., "Interactive Pattern Analysis and Recognition," IEEE Transactions on Computers, Vol. 19 (July 1970).

10. Gries, D., "Compiler Construction for Digital Computers," John Wiley & Sons, New York (1971).

11. Berkowitz, S., "Graph Information Retrieval Language; Programming Manual for FORTRAN Complement," Naval Ship Research and Development Center Report 4237, June 1973.

12. Hopcroft, J. and J. Ullman, "Formal Languages and Their Relation to Automata," Addison-Wesley, Reading, Mass. 1969.

13. Berkowitz, S., "Design Trade-offs for a Software Associative Memory," David Taylor Naval Ship Research and Development Center Report 3531 (May 1973).

14. Zaritsky, I., "GIRS (Graph Information Retrieval System) Users Manual," David Taylor Naval Ship Reserach and Development Center Report 79/036 (April 1979).

15. Zaritsky, I., "Paged GIRS (Graph Information Retrieval System) Users Manual," David Taylor Naval Ship Research and Development Report 81/024 (April 1981).

16. Zaritsky, I., "Feasibility Study for Incorporating a Data Structure Definition and Manipulation Facility within the COMRADE Data Management System," David Taylor Naval Ship Research and Development Report 78/045 (May 1978).

17. Carlberg, J., "A Paged Hardware Associative Memory - Preliminary Report," David Taylor Naval Ship Research and Development Report 77-0083 (Aug 1977).

18. Berkowitz, S., "Graph Information Retrieval Language; Programming Manual for FORTRAN Complement; Revision One," David Taylor Naval Ship Research and Development Report 76-0085 (Feb 1976).

INITIAL DISTRIBUTION

Copies	
1	CNR
1	NRL
1	NUSC
12	DTIC

CENTER DISTRIBUTION

Copies	Code	Name
1	18	G. Gleissner
1	1805	E. Cuthill
2	1808	D. Wildy
1	182	A. Camara
5	1824	S. Berkowitz
1	1828	J. Garner
1	184	J. Schot
1	185	T. Corin
1	187	M. Zubkoff
1	189	G. Gray
1	522.1	TIC (C)
1	522.2	TIC (A)
1	93	L. Marsh

DTNSRDC ISSUES THREE TYPES OF REPORTS

1. DTNSRDC REPORTS, A FORMAL SERIES, CONTAIN INFORMATION OF PERMANENT TECHNICAL VALUE. THEY CARRY A CONSECUTIVE NUMERICAL IDENTIFICATION REGARDLESS OF THEIR CLASSIFICATION OR THE ORIGINATING DEPARTMENT.

2. DEPARTMENTAL REPORTS, A SEMIFORMAL SERIES, CONTAIN INFORMATION OF A PRELIMINARY, TEMPORARY, OR PROPRIETARY NATURE OR OF LIMITED INTEREST OR SIGNIFICANCE. THEY CARRY A DEPARTMENTAL ALPHANUMERICAL IDENTIFICATION.

3. TECHNICAL MEMORANDA, AN INFORMAL SERIES, CONTAIN TECHNICAL DOCUMENTATION OF LIMITED USE AND INTEREST. THEY ARE PRIMARILY WORKING PAPERS INTENDED FOR INTERNAL USE. THEY CARRY AN IDENTIFYING NUMBER WHICH INDICATES THEIR TYPE AND THE NUMERICAL CODE OF THE ORIGINATING DEPARTMENT. ANY DISTRIBUTION OUTSIDE DTNSRDC MUST BE APPROVED BY THE HEAD OF THE ORIGINATING DEPARTMENT ON A CASE-BY-CASE BASIS.