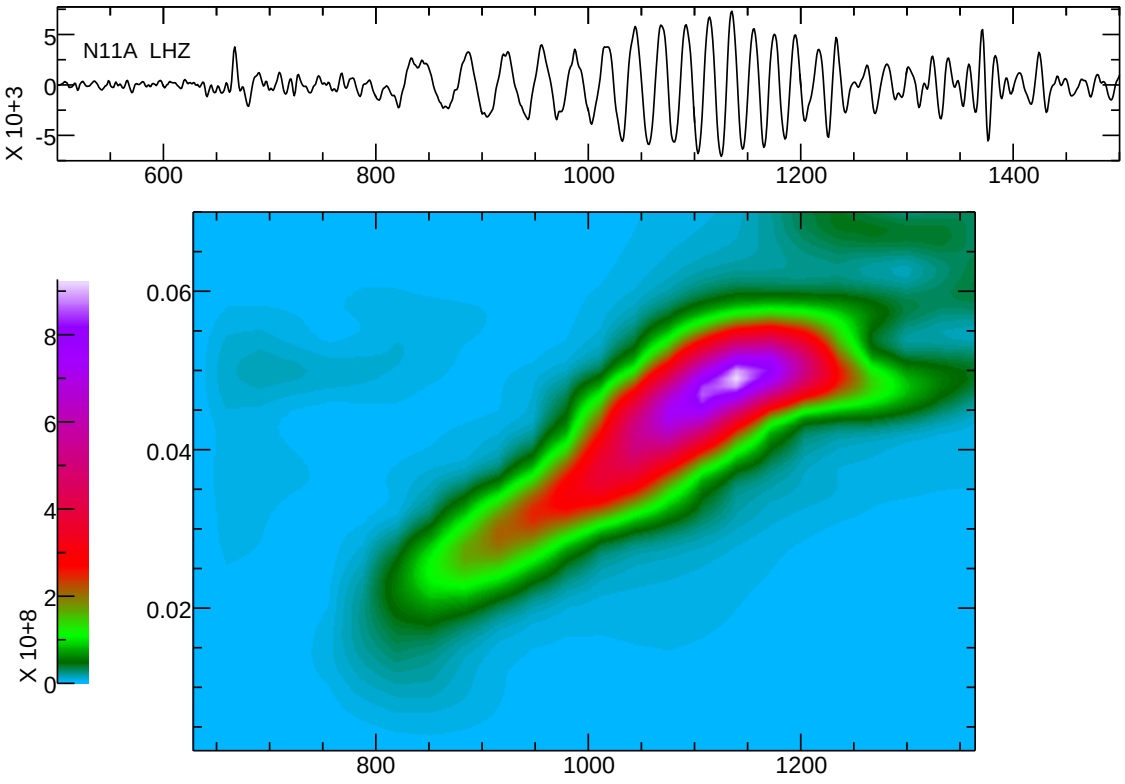
Seismic Analysis Code
Users Manual

Version 101.6a

November 17, 2014



Sections

- Using SAC
- SAC Commands
- Signal-Stacking Subprocess
- Spectral-Estimation Subprocess

Contents

1 Using SAC

Introduction

Overview

SAC (Seismic Analysis Code), previously SAC2000, is a general-purpose interactive program designed for the study of sequential signals, especially time-series data. Emphasis has been placed on analysis tools used by research seismologists in the detailed study of seismic events. Analysis capabilities include general arithmetic operations, Fourier transforms, three spectral estimation techniques, IIR and FIR filtering, signal stacking, decimation, interpolation, correlation, and seismic phase picking. SAC also contains an extensive graphics capability. Binary versions are available for Intel Mac and Linux, but SAC can be built from the source code for other computer operating systems. The source code is written in C. For further details regarding requirements to build and run SAC, see the README file that comes with the distribution.

SAC was developed at Lawrence Livermore National Laboratory and is copyrighted by the University of California. It is currently begin developed and maintained by a small group of developers working in cooperation with IRIS (http://www.iris.edu/hq/).

Thr Using SAC part of the SAC Users Manual contains general information for the new user about what SAC can do, how it works, and how to get started. It also contains detailed information for the more experienced user on topics such as how to use SAC macros, how to read and write SAC data files from C or FORTRAN programs, and how the SAC program is structured.

The Users Manual will be periodically updated to include new descriptions and to revise old ones. Please report any errors in this manual to the sac=help listserv: <sac-help@iris.washington.edu>. Although the version of the manual that comes with the binay or source distribution will be updated only when a new version comes out, the online version of the manual at <http://ds.iris.edu/files/sac-manual/> can be updated at any time.

Contents of Using SAC

Other Sections

Command Reference Manual contains detailed descriptions of each SAC command including purpose, syntax, default values, and examples. This manual also contains lists of SAC commands sorted alphabetically and functionally.

Spectral-Estimation Subprocess Manual describes a subprocess for the study of stationary random processes. A subprocess is like a small separate program within the main SAC program.

Signal-Stacking Subprocess Manual describes a subprocess for performing signal stacking with delays, traveltimes, and record section plots.

Notation

Repeating an important point made above, you may enter keywords and options in either uppercase or lowercase. SAC converts these to uppercase before interpreting them. The exceptions to this rule are text appearing within single or double quotes and the names of directories and files. The case of these items is not changed. They are interpreted literally.

SAC Tutorial Guide for New Users

Overview

SAC was designed as an aid to research seismologists in the study of seismic events. As such, it is used for quick preliminary analyses, for routine processing, for testing new techniques, for detailed research, and for creating publication quality graphics. It is used by both computer novices and experts. In order to make SAC quick to learn and easy to use, default values for all operational parameters were carefully chosen. At the same time, almost all of these parameters are under direct user control. This design combines ease of use with significant flexibility.

README

The first step is to study the README file that is in the top directory of the distribution: sac. It gives detailed instructions about setting up the environmental variables necessary to run SAC and other pieces of useful information.

User Interface

SAC is an interactive command-driven program. Commands may be typed at the terminal or placed in a macro file. SAC commands fall into three main categories: parameter-setting, action-producing and data-set manipulation. The parameter-setting commands change values of internal SAC parameters. Action-producing commands perform some operation on the signals currently in selected memory based upon the values of these parameters. Data-set commands determine which files are in active (selected) memory and therefore will be acted upon (data-set commands are not currently operational). The effect of a parameter-setting command remains in effect until it is reset. The effect of an action-producing command is immediate and transitory. Action-producing commands also have options which normally remain in effect until reset. These options, however, apply only to that particular command. The underlying assumption is that you are more likely than not to want to use the same values the next time you execute the same command. When you start up SAC, default values are defined for all of these parameters. SAC can be reinitialized to this default state at any time by executing the INICM command.

Mode of Operation

Each signal is stored in a separate data file. Each data file contains a header that describes the contents of that file. See the section on Data File Format for details. Signals are read from disk into memory using the READ command. CSS 3.0 formated flat files can be read using the READCSS command. SAC can process up to 200 signals of arbitrary size at a time. Once data is in memory other commands are typed at the terminal (or read from a macro file) to perform operations on these signals. All operations work concurrently on all signals in memory. You can look at the results at any time using the plot commands. There are several plot formats to choose from. You have control over titles and labels, plot limits, file identifications, axes and tick mark locations, etc. You can also save the results of these operations at any time using the WRITE command.

How SAC Handles Time

The SAC header contains a reference or zero time, stored as six integers (NZYEAR, NZJDAY, NZHOUR, NZMIN, NZSEC, NZMSEC), but normally printed in an equivalent alphanumeric format (KZDATE and KZTIME). This can be set to any reference time you wish. It is often the time of the first data point, but can also be the origin time of the event, midnight, your birthday, etc. It does not even have to be a time encompassed by the data itself. All other times are offsets in seconds from this reference time and are stored as floating point values in the header:

B Begin time of the file. E End time of the file. O Event origin time. A First arrival time. F Fini (end of signal.) Tn Time markers, where n is an integer from 0 to 9.

Getting Started

SAC will then print a short headline including the number and date of the version you have on your system. It may also print a bulletin giving some current information. SAC will then ask you for input by sending the prompt "SAC>".:

```
% sac
 SEISMIC ANALYSIS CODE [08/15/2006 (Version 100.1)]
 Copyright 1995 Regents of the University of California

SAC>
```

Interaction

SAC is an interactive command driven program. This means that you must type a command to get SAC to do something. It does not prompt you for input. Commands may be typed at the terminal or placed in a command file. Symbols within a command are separated by spaces and commands within a given line may be separated by a semicolon.

We'll start by creating a simple function:

```
SAC> FUNCGEN impluse
```

This generates an impulse function and stores it in SAC's memory. To see what this function looks like on your screen type:

```
SAC> PLOT
```

Abbreviations

There are abbreviations for the most used SAC commands. For example, fg and p are the abbreviations for FUNCGEN and PLOT respectively.

More functions

The FUNCGEN command can generate a number of different functions. To see them, use the HELP command:

```
SAC> help fg
```

Using fg can be very useful when first learning how to use SAC because you can see how the other SAC operations work on these functions. For example, type:

```
SAC> fg seismogram
```

This generates a sample seismic signal in SAC's memory. It also deletes the impulse generated earlier. Use the PLOT command to see this seismogram on your screen. Now for another function:

```
SAC> fg sine 2 npts 200 delta 0.01
```

The first thing to note is that SAC remembers seismogram as the function read most recently by fg. This is common for most commands in SAC: if a new argument for an option is not given, SAC uses the one most recently used in the current session. (Sometimes one forgot that an option had been used, so one may not correctly anticipate correctly the result of the operation.)

This more complicated example generates a 2 Hz sine wave in SAC's memory. The function will contain 200 data points and have a sampling interval of 0.01 seconds. You may want to use the PLOT command to plot this function also.

SAC Commands

There are several general points to be made at this point about SAC commands. All input is space delimited. The decimal point is optional wherever numeric input is needed. When you specify a value for a particular option, this value becomes the new current value. This means you don't have to keep entering values for options that you don't want to change. For example, you can now generate this same 2 Hz sine wave using the same sampling interval but with 400 data points by simply typing:

```
SAC> fg npts 400
```

SAC commands fall into two main categories: parameter-setting and action-producing. The parameter-setting commands basically change values of internal SAC parameters. Action-producing commands perform some operation on the data files currently in memory based upon the values of these same parameters. The effect of a parameter-setting command remains in effect until it is reset. The effect of an action-producing command, however, is immediate and transitory. For example, the parameter-setting command, YLOG, tells SAC to use logarithmic interpolation for the y axis in subsequent plots. The action-producing command, PLOT, does the actual plotting. Options to action-producing commands also remain in effect until reset just like parameter-setting commands. The underlying assumption is that you are more likely than not to want to use the same values the next time you execute the same command.

Default Values

All commands have "nice" default values for most options. The use of current and default values for command options can save you a lot of typing. For example, let's look at the BANDPASS command. This command applies a bandpass filter to the data currently in memory:

```
SAC> fg impulse npts 100 delta 0.01
SAC> bandpass bessel corner 0.1 0.3 npole 4
```

These two commands generate an impulse function and then apply a bandpass filter to that impulse. The filter is a four-pole Bessel filter with corner freqencies at 0.1 and 0.3 Hz. (To see the default values for BANDPASS, enter HELP BANDPASS.) You can see the result in the time domain by typing PLOT or you can see the amplitude reponse by taking the Fourier transform and using the PLOTSP command:

```
SAC> fft
SAC> plotsp am
```

You can now try a different set of corner frequencies very easily:

```
SAC> fg
SAC> bandpass corner 0.2 0.5
```

SAC generates the same impulse function and applies the same Bessel filter except for the new corner frequencies.

SAC Data Files

SAC is a program to examine, analyze, and plot data. This data is stored on disk as SAC data files. Each data file contains a single data set. For seismic data this means a single data component recorded at a single seismic station. SAC does not currently work on multiplexed data. The data will generally be evenly spaced time series data. SAC can also handle unevenly spaced data and spectral data. The spectral data can be in either real-imaginary or amplitude-phase format. Use help bandpass to see the defaults.

SAC Header

Each data file also contains a header record which describes the contents of that file. Certain header entries are always present (e.g., the number of data points, the file type.) Others are always present for certain file types (e.g., sampling interval, begin time, etc. for evenly spaced time series files.) Other header variables provide information needed by a particular operation (e.g., seismic component orientation used by the ROTATE command.) Still others are not used by SAC at all. They are simply informational. File sac data file format lists and discusses all header values. The LISTHDR command displays the contents of the headers for the data files currently in memory. You may wish to examine the header from the sample seismogram mentioned earlier:

```
SAC> FG seismogram
SAC> LH
```

If a particular header variable does not have a value for a particular file, then that variable is said to be "undefined" for that file. The LISTHDR command does not list undefined header variables, usless it is invoked with the INC or INCLUSIVE option (which includes undefined header variables). (Entr help lh to see the options.)

A few important header variables are listed below:

NPTS Number of points in data set. B Beginning value of the independent variable. E Ending value of the independent variable. IFTYPE Type of file. LEVEN TRUE if data set is evenly spaced. DELTA Increment between evenly spaced samples. IDEP Type of dependent variable. KZDATE Alphanumeric form of GMT reference date. KZTIME Alphanumeric form of GMT reference time. A First arrival time (seconds relative to reference time.) T n User defined time picks or markers, n=0,9.

Reading Data Files

SAC commands work on data already in SAC's working memory, not data on disk. The READ command is used to transfer data from disk to memory. Up to 100 data files can be in memory at the same time, and this limitation should be removed in upcoming versions. These can be of any size up the maximum size of SAC's working memory. You can use wildcard characters in the READ command to represent groups of files which have a similiar set of characters in their names. Each time you use the READ command to transfer data from disk to memory the data currently in memory is destroyed. If you want this data saved, you must write it to diskbefore reading more data into memory. There is an option called MORE in the READ command that lets you read data into memory without destroying the old data. See the Command Reference Manual for details.

Writing Data Files

At any time during your analysis, you may transfer this modified data back to disk using the WRITE command. You may overwrite the old data files on disk using the OVER option or create new ones by specifying their file names. Action commands (such as ADD, DECIMATE, and FFT) modify the data that is currently in memory. The data files on disk are not modified.

Reading and Writing Examples

A complete discussion of reading and writing SAC data files is given in sac reading and writing routines.

The examples below demonstrates several uses of the READ and WRITE commands.

Scaling Example

The first example reads two files into memory, multiplies each data point in each file by a constant, and then writes the results to disk in two new files:

```
SAC> R file1 file2
SAC> MUL 10 20
SAC> W file3 file4
```

Decimation Example

The next example reads a single file into memory, desamples the data by a factor of five (DECIMATE also applies an anti-aliasing filter), and then writes the results back to disk using the same file name:

```
SAC> R file1 file2 file3 file4
SAC> DECIMATE 5
SAC> WRITE OVER
```

Sample Data Files

You're going to need some data files for use in the next section on plotting. You'll also need them if you want to try any of the other commands discussed later in this guide. If you don't have any sample SAC data files around to play with, you can use FUNCGEN to generate some. This is shown in the example below:

```
SAC> fg triangle npts 200 delta 1.0
SAC> write file1
SAC> fg boxcar
SAC> write file2
SAC> fg step
SAC> write file3
```

This results in you having three files in your directory called file1, file2, file3 which contain the triangle, boxcar, and step functions respectively. Each will have 200 data points in them and be sampled at 1 sample per second. If you already had files in your directory by those names, they would be replaced by these new ones.

Displaying the Results

After reading data into SAC you can see it on your screen in several different formats using the various plot commands. Default values for each of the graphics display commands have been chosen to make it as easy as possible to display your data. By changing these default values before plotting, you also have complete control over the details of how each plot will look.

You've already used PLOT to display data files. With this command, each data file is plotted one at a time. SAC pauses between files to give you a chance to examine the data. This is shown in the following example.:

```
SAC> read file1 file2 file3
SAC> plot
Waiting [press return]
Waiting [press return]
Waiting [press return]
SAC>
```

Typing a "q" and then return will exit the plot command and not plot the remainder of the files in memory.

More Plot Commands

Several other canned plot formats are available. PLOT1 plots each file along a common x axis but with a separate y axes. By default all files are placed on the same plot. Try this with the three files from the example above. PLOT2 is an overlay plot. Again all files are plotted together, this time using both a common x and a common y axis. PLOTPK uses a format similiar to PLOT1. It lets you use the cursor to blow up parts of the plot, determine values of selected data points, pick phase arrival times, etc.

Display Options

By default, all SAC plots are self-scaling. SAC determines what limits to use for the x and y axes. If you want to set these limits yourself, you may do so using the XLIM and YLIM commands. If you wish, you may also change the location of annotated axes, change the linestyle, select a symbol to be plotted at each data point, create titles and labels, make logarithmic plots, change the size and type of text, and control a number of other even more exotic aspects of the plot. These commands are part of the Graphics Environment Module, and are defined in links frmon the Command Reference Manual.

An Overview of Graphics Capability in SAC

File Graphics in SAC has an overview of graphics in SAC. Of paaricular interest may be the command SAVEIMG that allws one to save displayed plots in several formats.

Analysis Capabilities

SAC does a lot more than just reading, writing, and plotting data files! SAC Analysis Capabilities provides an introduction to these features. Command Reference Manual lists all the commands and has links to help files for them.

SAC Analysis Capabilities

Overview

SAC is logically divided into functional modules. Each functional module performs a related set of tasks. This section briefly describes the commands in each of these modules. The full command names are used in these descriptions. Most of the commands have convenient abbreviations. See the SAC Command Reference Manual for details.

Function Module

Once you have succesfully started SAC, you need to know how to get rid of it! This is done with the QUIT command. END, EXIT, and DONE are also allowed so you shouldn't have any problems.

FUNCGEN lets you generate various functions in memory. It is useful for testing the other commands on known functions.

DATAGEN lets you read sample data from three events (one local, one regional, and one teleseismic) into memory. This lets you play with some real seismic data while you are getting your own data converted to the SAC data file format.

Files README, HISTORY, and CHANGES in the top directory of SAC give general information about the current version of SAC and previous versions. HELP gives you information about a specific command, and REPORT gives you the current values of important parameters. SAC has an extensive macro capability that is described in SAC Macros. A macro lets you execute a set of SAC commands from a file you write. You can define arguments complete with default values, perform simple arithmetic calculations, store and retrieve information, and control the flow of command execution with if-tests and do-loops.

- MACRO executes a macro file.
- SETMACRO defines the search path to be used to find a macro file.
- INSTALLMACRO lets you make a macro available for use by anyone else on your system.

An inline function is one that is enclosed in parenthesis and placed within a regular SAC command. SAC Inline Functions can be used both within macros or within regular SAC code.

You can store (SETBB) and retrieve (GETBB) information and do arithmetic calculations (EVALUATE) using the "blackboard." You can also save (WRITEBBF) and restore (READBBF) information in the blackboard into a disk file. See Blackboard Variables in SAC for more information about blackboard variables.

Other commands that are useful in a macro include the ability to send a message to the terminal (MESSAGE), echo commands to the terminal (ECHO), and temporarily suspend the execution of a macro (PAUSE).

Using SAC Reading and Writing Routines, one can write stand-alone codes in C or FORTRAN to read and write SAC-formatted data files.

You can write your own SAC command in FORTRAN or C routines that can be loaded into SAC (LOAD), and executed thereafter just like an indiginous SAC command (see Notes and external_interface in aux/external).

- TRACE can be used to have SAC trace header and blackboard variables, reporting to the screen when a variable changes values.
- TRANSCRIPT controls SAC's transcription capabilities, saving commands, and/or error messages, and/or warnings, and/or other output to a text file.
- COMCOR provides command correction. When SAC detects an error during the course of executing a command, if this option is set, SAC will allow the user to correct the command and continue execution.

- CD changes SAC's current working directory.

The SAC program can be run from a variety of scripting languages and shells. See Calling SAC from Scripts.

Finally, you can execute operating system commands while running SAC (SYSTEMCOMMAND) and reinitialize SAC to its default state (INICM).

Execution Module

This module provide commands which contol the flow of commands. These commands can only be called from within a macro, and are discussed in greater detail in the section on SAC macros. The commands in this module are IF, ELSEIF, ELSE, ENDIF,DO, WHILE, ENDDO, and BREAK. These are discussed in Sac Inline Functions, Blackboard Variables in SAC, and SAC Macros.

Data File Module

This module is used to read, write, and access SAC data files. These data files are described in detail in a later section.

- READ reads data files from disk into memory and WRITE writes the data currently in memory to disk.
- CUT defines how much of a data file is to be read.
- CUTIM appies CUT to multiple segments in each file in memory
- READERR controls errors that occur while files are being read and
- CUTERR controls errors due to bad cut parameters.

Each data file has a header which describes the contents of the file. You can read and write these headers without the data using READHDR and WRITEHDR.

You can also list the contents (LISTHDR), change values (CHNHDR), and copy header values from one file to the others in memory (COPYHDR).

The SYNCHRONIZE command changes the headers in memory so that they all have the same reference time. You must first use this command before using the CUT command on files with different reference times.

You can use READTABLE to read almost any alphanumeric data file directly into SAC.

The read commands let you use wildcard characters to easily read in groups of files that contain the same pattern of characters.

The WILD command controls certain aspects of this wildcard filename expansion.

The SAC data file is stored in binary format for fast reading and writing. There is also an alphanumeric equivalent of this binary format. This is useful when transfering SAC data files from one kind of computer to another kind.

- CONVERT can be used to convert between the binary and alphanumeric formats.
- READCSS reads CSS 3.0 formatted flat files. Preferences for the way picks are read in are set in a preferences file but can be modified using the PICKAUTHOR and PICKPHASE command.
- WRITECSS writes the data to flat files. WRITECSS is currently not working as comprehensively as READCSS.
- DELETECHANNEL allows you to delete one or more files from memory.
- READSDD and WRITESDD allow reading and writing of SDD data files.

Spectral Analysis Module

You can do a discrete Fourier transform (FFT) and an inverse transform (IFFT). You can also compute the amplitude and unwrapped phase of a signal (UNWRAP). This is an implementation of the algorithm due to Tribolet.

There is a set of Infinite Impulse Response filters (BANDPASS, BANDREJ, LOWPASS, and HIGHPASS), a Finite Impulse Response filter (FIR), an adaptive Wiener filter (WIENER), and two specialized filters (BENIOFF and KHRONHITE) used at LLNL.

- CORRELATE computes the auto- and cross-correlation functions.
- CONVOLVE computes the auto- and cross-convolution functions.
- FFT and UNWRAP commands produce spectral data in memory. You can plot this spectral data (PLOTSP), write it to disk as "normal" data (WRITESP), and read in back in again (READSP). You can also perform integration (DIVOMEGA) and differentiation (MULOMEGA) directly in the frequency domain.
- HANNING applies a "hanning" window to each data file.
- HILBERT applies a Hilbert transform.
- ENVELOPE computes the envelope function using a Hilbert transform.
- KEEPAM keeps amplitude component of spectral files (of either the AMPH or RLIM format) in SAC memory.

Unary-Operations Module

The commands in this module perform some arithmetic operation on each data point of the signals in memory. You can add a constant (ADD), subtract a constant (SUB), multiply by a constant (MUL), or divide by a constant (DIV). You can square each data point (SQR), take the square root (SQRT), or take the absolute value (ABS). You can take the natural (LOG) or base 10 (LOG10) logarithm of each data point. You can also compute the exponential (EXP) or base 10 exponential (EXP10) of each data point. Lastly you can perform integratation (INT) and differentation (DIF).

Binary-Operations Module

These commands perform operations on pairs of data files.

- MERGE merges (concantenates) a set of files to the data in memory.
- ADDF adds a set of data files to the data in memory.
- SUBF subtracts a set of data files from the ones in memory.
- MULF multiplies a set of data files by the data in memory.
- DIVF divides the data in memory by a set of files.
- BINOPERR controls errors that can occur during these binary operations.

Signal-Correction Module

These commands let you perform certain signal correction operations.

- RQ removes the seismic Q factor from spectral data.
- RTREND and RMEAN remove the linear trend and the mean from data respectively.
- RGLITCHES removes glitches and timing marks.

- TAPER applies a symmetric taper to each end of the data and SMOOTH applies an arithmetic smoothing algorithm.
- STRETCH upsamples data, including an optional interpolating FIR filter, while
- DECIMATE downsamples data, including an optional anti-aliasing FIR filter.
- INTERPOLATE You can interpolate evenly or unevenly spaced data to a new sampling interval using the INTERPOLATE command.
- LINEFIT computes the best straight line fit to the data in memory and writes the results to header blackboard variables.
- QUANTIZE converts continuous data into its quantized equivalent.
- REVERSE reverses the order of data points.
- ROTATE Finally, you can rotate pairs of data components through a specified angle with the ROTATE command.

Event-Analysis Module

This module is used to pick seismic phases.

An automatic phase picking algorithm can be applied using APK.

You can also use PPK to pick phases using the graphics cursor. (PPK is described in the section on Graphics Capabilities).

The TRAVELTIME command can be used to associate observed arrivals with predicted body-wave phases.

These picks can be saved in HYPO format using the OHPF (open HYPO pick file) and CHPF (close HYPO pick file) commands; WHPF writes auxiliary cards into the HYPO pick file. These picks can also be saved in a more general Alphanumeric format using the OAPF (open alphanumeric pick file) and CAPF (close alphanumeric pick file) commands. The picks are also saved in the headers.

Signal-Measurement Module

These commands measure and "mark" selected attributes about the data in memory. These marks are stored in the headers.

- MARKTIMES marks the data in memory with travel times from a velocity set.
- MARKPTP measures and marks the maximum peak to peak amplitude.
- MARKVALUE searches for and marks selected values in a signal.
- MTW sets the "measurement time window" option.

When this option is on, the measurements are made within this window only. Otherwise the measurements are made on the entire signal.

MTW applies to the MARKPTP and MARKVALUE commands only. RMS computes the root mean square of the data within the measurement time window.

Instrument-Correction Module

This module currently contains only one command, TRANSFER.

TRANSFER performs a deconvolution to remove one instrument response followed a convolution to apply another instrument response. Over 40 predefined instrument responses are available. A general instrument response can also be specified in terms of its poles and zeros, frequency-amplitude-phase files, or the EVALRESP package.

XYZ Data-Processing Module

The commands in this module produce output that is a function of two input domains.

- SPECTROGRAM calculates a spectrogram using all of the data in memory.
- GRAYSCALE produces grayscale images of data in memory.
- CONTOUR produces contour plots of data in memory.
- ZLEVELS controls the contour line spacing in subsequent contour plots.
- ZLINES controls the contour linestyles in subsequent contour plots.
- ZTICKS controls the labeling of contour lines with directional tick marks.
- ZLABELS controls the labeling of contour lines with contour level values.
- ZCOLORS controls the color display of contour lines.
- IMAGE produces color sampled image plots of data in memory.
- SONOGRAM calculates a spectrogram equal to the difference between two smoothed versions of the same spectrogram.

Frequency-waveform Spectral Analysis Module

Most of the command in this module are algorithms to extract wavefield parameters from a suite of seismograms.

- ARRAYMAP produces a map of the array or "coarray" using all files in SAC memory.
- BBFK computes the broadband frequency-wavenumber (FK) spectral estimate, using all files in SAC memory.
- BEAM computes the beam using all data files in SAC memory.
- MAP generates a GMT (Generic Mapping Tools) map with station/event symbols using all the files in SAC memory and an event file specified on the command line.

Matlab Module

This module provides an interface between SAC and MATLAB, allowing users who have MATLAB the ability to utilize its facilities and m-files on SAC files.

- 3C launches a Matlab GUI for manipulating 3-component data.
- MAT allows processing of SAC data from within SAC using the MATLAB engine.

See README_buildsac for information about some macros provided by a user to interface between SAC and MATLAB.

Nerual Network Module

This module has only one command, WRITENN, which writes data files to disk in neural net format.

Subprocesses

A subprocess is like a small program within the larger SAC program. It works like SAC in many ways but the differences are such that it could not be included in the main program. Once invoked, only the commands within that subprocess plus a selected group of commands from the main SAC program can be executed. The prompt changes to include the name of the subprocess. When done you can return to the main SAC program using the QUITSUB command or terminate SAC using the QUIT command.

Spectral-Estimation Subprocess Manual

This subprocess is for the study of stationary random processes (i.e. noise). Three spectral estimation techniques are available: the maximum entropy method, the maximum likelihood method, and the power density spectra method.

Signal-Stacking Subprocess Manual

This subprocess is for performing signal stacking with delays. The delays can be static or dynamic. Two velocity models are available. The signals can be individually weighted. Traveltimes can be computed, or read from a file. A record section plot is also part of this subprocess.

SAC Macros

Overview

A SAC macro is a file that contains a set of SAC commands to be executed together. As well as regular commands and inline functions, a SAC macro file can contain references to SAC header variables and blackboard variables that are evaluated and substituted into the command before it is executed. SAC macros can also have arguments that are evaluated as the macro is executed. Control flow features such as "if tests" and "do loops" are also available. These features let you control and alter the order of execution of commands within a macro. All of these features are discussed later in this section.

A Simple Example

Assume that you have a set of commands that you execute repeatedly. A macro file is the obvious solution. Simply fire up your favorite text editor, put the commands into a file, and then have SAC execute them using the MACRO command. Lets say you wanted to repeatedly read the same three files, multiply each file by a different value, take the Fourier transform, and plot the amplitude responses to a set of SAC Graphics Files. The macro file would look like this:

```
* * This certainly is a simple little macro.
READ ABC DEF XYZ
MUL 4 8 9
FFT
BG SGF
PSP AM
```

Assume the file is called MYSTUFF and that it and the data are in the directory to which you are currently attached. To execute the macro from SAC type:

```
SAC> MACRO MYSTUFF
```

Note that commands in a macro file are not normally echoed to the terminal as they are executed. You can use the ECHO command to turn command echoing on if you wish. Also note that an asterisk in the first column of a line denotes a comment line and is not processed by SAC.

Order Dependent Arguments

The above example while simple is also very inflexible. If you wanted to read a different set of files or use a different set of multiplicative values you have to edit the file. Allowing macros to have arguments that you enter at execution time greatly increases their flexibility. We will modify the previous macro to accept the names of the files as arguments:

```
READ $1 $2 $3
MUL 4 8 9
FFT
BG SGF
PSP AM
```

The dollar sign ("$") is used to delineate arguments in a macro file. $1 is the first argument, $2 the second, $3 the third, and so on. To execute this modified macro from SAC type:

```
SAC> MACRO MYSTUFF ABC DEF XYZ
```

The token "ABC" is substituted wherever the "$1" token is found. Also "DEF" and "XYZ" are substituted for "$2" and "$3" respectively. To execute the same macro with a different set of files only the execute line changes:

```
SAC> MACRO MYSTUFF AAA BBB CCC
```

Keyword-Driven Arguments

Keyword driven arguments let you enter arguments in any order and also makes the body of a macro easier to understand. This becomes increasingly important as the number of arguments and the size of the macro increase. Lets again modify our example to accept a list of files and also a list of multiplicative values:

```
$KEYS FILES VALUES
READ $FILES
MUL $VALUES
FFT
BG SGF
PSP AM
```

This simple change has increased both the flexibility and the readability of the macro. The first line says that there are two keywords, one called "FILES" and the other called "VALUES". To execute it you could type:

```
SAC> MACRO MYSTUFF FILES ABC DEF XYZ VALUES 4 8 9
```

Since the order of the arguments is no longer important you could also type:

```
SAC> MACRO MYSTUFF VALUES 4 8 9 FILES ABC DEF XYZ
```

This macro doesn't limit you to reading in only three files. It would work equally well for two or 10 files as long as the number of values match the number of files.

Default Argument Values

There are times when you have a macro where some arguments often (but not always) have the same value from one execution to the next. Providing default values for such arguments eliminates the need to enter the same values each time but allows you the flexibility to enter them when needed. This is demonstrated in the next example:

```
$KEYS FILES VALUES
$DEFAULT VALUES 4 8 9
READ $FILES
MUL $VALUES
FFT
BG SGF
PSP AM
```

The second line in the macro specifies a default value to be used for the variable "VALUES" if you don't enter one on the execute line:

```
SAC> MACRO MYSTUFF FILES ABC DEF XYZ
```

If you wanted to use a different set of values you could type:

```
SAC> MACRO MYSTUFF VALUES 10 12 3 FILES ABC DEF XYZ
```

Argument Querying

If you fail to enter a value for an argument on the execute line and it has no default value, SAC will ask you to enter a value from the terminal. Using the macro in the previous section, assume that you forgot to enter the filelist:

```
SAC> MACRO MYSTUFF
FILES?
SAC> ABC DEF XYZ
```

Note that SAC did not query for "VALUES" because it had a default value. The timing of this query can sometimes be used to your advantage. SAC does not query for a value until it first tries to evaluate the argument and finds that it has no default or input value. This allows part of the macro to execute showing you some partial results before asking you to enter values for an argument.

Blackboard Variables

SAC has a blackboard feature that can be used to temporarily store and retrieve information. A blackboard entry consists of a name and a value. Blackboard entries are created using the SETBB and EVALUATE commands. The value of a blackboard variable can be obtained using the GETBB command. You can also substitute the value of a blackboard variable directly in other commands by preceeding its name with a percent sign ("%") as shown below:

```
SAC> SETBB C1 2.45
SAC> SETBB C2 4.94
SAC> BANDPASS CORNERS %C1 %C2
```

Now lets see how blackboard variables can be used in macros. (You are probably getting tired of endless variations on our original macro, but we are almost done with it.) Assume that only the first value was a variable, i.e. the other values could be calculated from the first as shown below:

```
$KEYS FILES VALUE1
$DEFAULT VALUE1 4
READ $FILES
EVALUATE TO VALUE2 $VALUE1 * 2
EVALUATE TO VALUE3 %VALUE2 + 1
MUL $VALUE1 %VALUE2 %VALUE3
FFT
BG SGF
PSP AM
```

Now only the first value is input to the macro and only if it differs from the default value:

```
SAC> MACRO MYSTUFF VALUE1 6 FILES ABC DEF XYZ
```

See Blackboard Variables in SAC for further discussion and examples.

Header Variables

SAC header variables can also be evaluated and substituted directly in commands much like blackboard variables. You must specify which file (by name or number) and which variable to be evaluated. You must preceed this specification with an ampersand ("&") and you must separate the file and variable with a comma as shown below:

```
SAC> READ ABC
SAC> EVALUATE TO TEMP1 &ABC,A + 10
SAC> EVALUATE TO TEMP2 &1,DEPMAX * 2
SAC> CHNHDR T5 %TEMP1
SAC> CHNHDR USER0 %TEMP2
```

In the above example a file is read in and several temporary blackboard variables are calculated using header variables from the file itself. The first header reference is by file name and the second by file number. New header variables are then defined using these blackboard variables.

Concatenation

You can append or prepend any text string to a macro argument, blackboard variable, or header variable. To prepend simply concatenate the text string with the argument or variable. To append you must repeat the delimiter ($, %, or &) after the argument or variable and before the text string. Sounds confusing? See the examples below for some clarification:

Assume that the macro argument STATION has the value "ABC". Then value of "$STATION$.Z" would be "ABC.Z".

Assume that the blackboard variable TEMP has the value "ABC". Then value of "XYZ%TEMP" would be "XYZABC" and the value of "%TEMP%XYZ" would be "ABCXYZ".

Assume that the header variable KA for file Z has the value "IPU0". Then value of "(& Z,KA &)" would be "(IPU0)".

Nesting and Recursion

When a macro can call another macro which can call another macro, etc., this is often refered to as nesting. When one macro calls another, the second macro is said to be operating at a new (lower) level of execution. The top level of execution is always interactive input from the terminal. When a macro can call itself, then it is said to be recursive. The SAC macro capability supports nesting but not recursion. SAC does not check to ensure that macro calls are not recursive. It is the responsibility of the user to make sure a macro is not directly or indirectly calling itself.

Interrupting a MACRO

There are occasions when you need to temporarily interrupt the execution of a macro, enter a few commands from the terminal, and then continue executing the macro. This can be done in SAC using the pause and resume feature. When SAC sees a $TERMINAL in a macro it temporarily stops reading commands from the macro, changes its prompt to include the name of the macro, and starts prompting for commands from the terminal. Then when SAC sees a $RESUME entered from the terminal it stops reading commands from the terminal and begins reading from the macro starting at the next line (the one after the $TERMINAL.) If you don't want to continue executing the commands in the macro you can type a $KILL from the terminal. SAC will then close the macro file and return to the previous level of execution, normally interactive input from the terminal. You can have more than one $TERMINAL in a macro.

If Tests

This feature lets you alter the order of commands being executed from a macro file. The syntax is similiar but not identical to the if-then-else clause in F77:

IF expr commands ELSEIF expr commands ELSE commands ENDIF

In the above clause expr is a logical expression of the form:

```
token op token
```

where token is a constant, macro argument, blackboard variable, or a header variable and op is one of the following logical operators:

```
GT | GE | LE | LT | EQ | NE
```

The tokens are converted to floating point numbers before the logical expression is evaluated. The maximum number of nested if clauses is currently set at 10. The ELSEIF and ELSE elements are optional. There is no limit of the number of ELSEIF elements in an if clause. Note that there are no parentheses around a logical expression and no THEN keyword ending the IF and ELSEIF elements as in F77. (If a THEN is included, it is ignored.) Unlike Fortran, an ENDIF is always required -- even if there is only a single option. An example is given below:

```
READ $1
MARKPTP
IF &1,USER0 GE 2.45
FFT
PLOTSP AM
ELSE
MESSAGE "Peak to peak for \$1 below threshold."
ENDIF
```

In this example a file is read into memory and the maximum peak to peak amplitude is measured.

(MARKPTP stores this amplitude into the header variable USER0.) If this amplitude is above a certain value, a Fourier transform is calculated and the amplitude response is plotted. If not a message is sent to the terminal.

Do Loops

These features let you easily repeat a set of commands. You can execute a set of commands a fixed number of times, for each element in a list, or until a condition has been met. You can also break out (prematurely terminate the execution) of a do loop. The syntax for this group is summarized below:

```
DO variable = start, stop, {,increment}
commands
ENDDO

DO variable FROM start TO stop { BY increment}
commands
ENDDO

DO variable} LIST} entrylist}
commands
ENDDO

DO variable WILD {DIR name} entrylist
commands
ENDDO

WHILE expr
commands
ENDDO
BREAK
```

Where:

- variable is the name of the do loop variable. Its current value while the do loop is executing is stored as a macro argument and may be used in the body of the do loop (i.e., the commands) by preceeding its name with a dollar sign.
- start is the starting value for the do loop variable. It must be an integer.
- stop is the stopping value for the do loop variable and must also be an integer.
- increment is the optional increment in the do loop variable. If omitted, the default value is set to 1.
- entrylist is a space delimited list of values that the do loop variable is to have.

These may be integers, floating point numbers, or character strings. In the DO WILD case, the entrylist consists of character strings containing both regular and wildcard characters. This entrylist is expanded into a list of files that match the character strings before the do loop is executed.

expr is a logical expression as described in the section on if tests.

The maximum number of nested do loops is currently set at 10. Examples of each of these do loops are given below.

Do Loop Examples

The first macro applies the DIF command to a data file (to prewhiten the data), performs a Fourier transform on the data, and then applies the DIVOMEGA command to remove the effect of the prewhitening. Sometimes it is necessary to differentiate the data more than once before doing the transform. This can be handled with a do loop:

```
$KEYS FILE NPREW
$DEFAULT NPREW 1
READ \$FILE
DO J = 1 , $NPREW
DIF
ENDDO
FFT AMPH
DO J = 1 , $NPREW
DIVOMEGA
ENDDO
```

Notice the use of a default value for the order of the prewhitening. In the second example, particle motion plots are produced for five different two second time windows on the same data file:

```
READ ABC
SETBB TIME1 0
DO TIME2 FROM 2 TO 10 BY 2
XLIM %TIME1 $TIME2
TITLE 'Particle Motion from %TIME1 to $TIME2$'
PLOTPM
SETBB TIME1 $TIME2
ENDDO
```

(Why is a dollar sign needed after TIME2 in the TITLE command?) In the next example, a macro called PREVIEW exists that performs a set of commands on a single data file. A new macro is created with several nested do loops to run PREVIEW on a predefined group of data files:

```
DO STATION LIST ABC DEF XYZ
DO COMPONENT LIST Z N E
MACRO PREVIEW $STATION$.$COMPONENT$
ENDDO
ENDDO
```

In the next example we modify the previous one to now process all of the files in the directory called "MYDIR" that end in the letters ".Z":

```
DO FILE WILD DIR MYDIR *.Z
MACRO PREVIEW $FILE
ENDDO
```

The last (somewhat artificial) example has three arguments. The first is the name of a data file, the second a multiplicative constant, and the third a threshold value. The macro reads the data file into memory, and multiplies each data point by the constant until the maximum value is below the threshold:

```
READ $1
WHILE &1,DEPMAX GT $3
MUL $2
ENDDO
```

Another version of this macro illustrates the BREAK statement:

```
READ $1
WHILE 1 GT 0
DIV $2
IF &1,DEPMAX GT $3
BREAK
ENDIF
ENDDO
```

This WHILE loop in this macro is an example of a "do forever" loop which can only be terminated by a BREAK statement. (This version of the macro has a flaw. What happens if the maximum value is already below the threshold?) The BREAK statement terminates the execution of the do loop where the statement appears.

Executing Other Programs From SAC Macros

You can execute other programs from inside a SAC macro. You can pass an optional execution line message to the program. If the program is interactive, you can also send input lines to it. The syntax for this feature is given below:

```
$RUN program message
inputlines
$ENDRUN
```

Macro arguments, blackboard variables, header variables, and inline functions may be used in the above lines. They are all evaluated before the program is executed. When the program completes the SAC macro resumes at the line following the ENDRUN line.

If there are no inputlines, one can use command SC (SYSTEMCOMMAND):

```
SC program message
```

Macro Search Path

When you request a macro, SAC searches for it as follows:

```
- in the current directory.
- in the directories specified in the SETMACRO_ command.
- in the global macro directory that is maintained by SAC.
```

The global macro directory contains macros meant to be used by everyone on your system. Use the INSTALL-MACRO command to install a macro in this directory. You may also specify the absolute or relative pathname of a macro that is not in this search path.

Execution Line Macro

SAC treats command-line arguments as a sequence of macros to run before reading your typed-in commands from the SAC> command line. These are processed, in sequence, by SAC and may be used to customize the run-time environment to your preference. For example, you might open a graphical device window and place it in a preferred place on your screen, set up a path to search for SAC macro commands, or change plot colors or line widths.

The Escape Character

There may be times when you need to use a dollar sign or a percent sign in a command and not have SAC interpret it as a macro argument or blackboard variable entry. To do this you preceed the special character with another special character, called the escape character. The escape character is an "at" sign ("@"). The special characters that must be treated in this way are:

```
$  The macro argument expansion character.
%  The blackboard variable expansion character.
&  The header variable expansion character.
@  The escape character itself.
(  The inline function starting character.
)  The inline function terminating character.
```

More about the inline function delimiters in SAC Inline Functions.

Acknowledgements

The concept of blackboard variables are due to Dave Harris. The "if test" and "do loop" features were developed by Mandy Goldner.

SAC Inline Functions

Overview

An inline function is one that is enclosed in parenthesis and placed within a regular SAC command. The inline function is evaluated and its resulting value replaces the function in the SAC command before the command is executed.

There are three general classes of inline functions:

- embedded arithmetic functions that begin with a number and have the name of the function embedded in the argument list.
- regular arithmetic functions that begin with the function name and are followed by zero or more arguments.
- character string manipulation functions that begin with the function name and are followed by zero or more arguments.

Inline functions can be placed inside other inline functions. This is refered to as nesting. Beginning with v 101.6, there is no nesting limits of inline functions. Macro arguments, blackboard variables and header variables can be used as arguments to inline functions. They are inserted in inline functions using the same syntax as in regular SAC commands.

Embedded Arithmetic Functions

An embedded arithmetic function is a simple math operation similar to those in any programming language, e.g. FORTRAN, C, etc, and is of the general form:

```
( number operator number ... )
```

where number is a numeric value and operator is one of the following arithmetic operators:

```
+   -   *   /   **
```

All numbers are treated as real, and all arithmetic is done in double-precision floating point.

In the examples below, "echo on" is used and redundant output lines are left out.

Here is a simple example:

```
SAC> SETBB A (4 + 7 / 3)
===> SETBB A 6.33333
```

Prior to version 101.6, the answer would have been 3.666667 because operations were executed from left to right. Beginning with version 101.6, the FORTRAN heirarchy is used: ** then / then * then + and -. As in FORTRAN, the heirarchy can be changed by using parentheses:

```
SAC> SETBB A ((4 + 7) / 3)
==> SETBB A 3.66667
```

Because there are many scripts and macros written before v101.6, expressions with inline functions like the above will give the incorrect answer if there are no specific parentheses. If the new (with v101.6) function MATHOP is called with the option old before lines with inline functions, the precedence rules that held prior to v101.6 will be followed:

```
SAC> MATHOP OLD
SAC> SETBB A (4 + 7 / 3)
===> SETBB A 3.66667
```

Regular Arithmetic Functions

There are 20 regular arithmetic functions available. They correspond to the arithmetic functions found in the EVALUATE command. Each of these functions is described below. Some examples are given at the end of this subsection.

Command	Syntax	Purpose
ADD	(ADD v1 v2 ... vn)	Add (sum) a set of numbers.
SINE	(SINE v)	Take the sine of a number.
SUBTRACT	(SUBTRACT v1 v2 ... vn)	Subtract a set of numbers.
ARCSINE	(ARCSINE v)	Take the arcsine of a number.
MULTIPLY	(MULTIPLY v1 v2 ... vn)	Multiply data in memory by a set of numbers.
COSINE	(COSINE v)	Take the cosine of a number.
DIVIDE	(DIVIDE v1 v2 ... vn)	Divide data in memory by a set of numbers.
ARCCOSINE	(ARCCOSINE v)	Take the arccosine of a number.
SQRT	(SQRT v)	Take the square root of a number.
TANGENT	(TANGENT v)	Take the tangent of a number.
EXP	(EXP v)	Exponentiate a number.
ARCTANGENT	(ARCTANGENT v)	Take the arctangent of a number.
ALOG	(ALOG v)	Take the natural logarithm of a number.
INTEGER	(INTEGER v)	Convert a number to an integer.
POWER	(POWER v)	Raise a number to its power of 10.
PI	PI or (PI)	Return the value of pi.
ALOG10	(ALOG10 v)	Take the log to base 10 of a number.
MAXIMUM	(MAXIMUM v1 v2 ... vn)	Maximum value of a set of numbers.
MINIMUM	(MINIMUM v1 v2 ... vn)	Minimum value of a set of numbers.
ABSOLUTE	(ABSOLUTE v)	Take the absolute value of a number.

Exampples

To normalize a set of data files so that the maximum absolute value of any data point in the set is unity:

```
SAC> fg seismo
SAC> write one.sac
SAC> mul 2.0
SAC> write two.sac
SAC> mul 4.0
SAC> write four.sac
SAC> read one.sac two.sac four.sac
SAC> lh depmax depmin
FILE: one.sac - 1
depmax = 1.520640e+00  depmin = -1.569280e+00
FILE: two.sac - 2
depmax = 3.041280e+00  depmin = -3.138560e+00
FILE: four.sac - 3
depmax = 1.216512e+01  depmin = -1.255424e+01
SAC> setbb a (max &1,depmax &2,depmax &3,depmax)
==> SETBB A 1.87324
SAC> setbb b (min &1,depmin &2,depmin &3,depmin)
```

```
==> SETBB B -2.123371
SAC> div (max %a (abs %b))
==> DIV 2.123371
SAC> lh depmax depmin
FILE: one.sac - 1
depmax = 1.211256e-01   depmin = -1.250000e-01
ILE: two.sac - 2
depmax = 2.422512e-01   depmin = -2.500000e-01
FILE: four.sac - 3
depmax = 9.690049e-01   depmin = -1.000000e+00
SAC>
```

In the next example, we need to calculate the tangent of an angle that has already been stored in degrees:

```
SAC> setbb angle (45)
==>  setbb angle 45
SAC> SETBB VALUE (TAN (PI * %ANGLE / 180. ))
==>  SETBB VALUE 1
```

Prior to v101.6, one needed %ANGLE%. With the new parsing system, the trailing % is no longer needed.

Miscellaneous Arithmetic Functions

GETTIME, the only miscellaneous arithmetic function, returns the time offset (in seconds) relative to file begin time, for the first data point meeting the selection criteria.:

```
GETTIME
( GETTIME MAX|MIN [value])
```

If no value is specified, MAX returns the time of the file's first data-point having a value equal to DEPMAX and MIN returns the time of the file's first data-point having the value l equal to DEPMIN. Specifying a value controlls the value of the data-point being searched for.

For example, to return the time in seconds of the first data-point equal to the maximum amplitude for the file FILE1:

```
SAC> READ FILE1
SAC> SETBB MAXTIME ( GETTIME MAX )
==> SETBB MAXTIME 41.87
```

The file's maximum amplitude is located 41.87 seconds into the file.

To locate the time of the first data-point less than or equal to the value 123.45:

```
SAC> SETBB VALUETIME ( GETTIME MIN 123.45 )
==> SETBB VALUETIME 37.9
```

The first data-point in the file having a value less than or equal to 123.45 occurs at 37.9 seconds into the file.

Character strings

Prior to v101.6, Blackboard number variables were stored as strings, now they are stored as double-precision variables. In earlier versions, if a (..) appeared in a quoted string, escape character @ was needed to keep the inline parser from treating the expression as a math expression. Although that coding continues to work in 101.6, adding the escape characters is no longer necessary:

```
SAC> fg seismo
SAC> xlabel "Time @(sec@)"
==>  xlabel "Time (sec)"
SAC> xlabel "Time (sec)"
xlabel "Time (sec)"
SAC>
```

String Functions

There are currently seven string manipulation functions. Each of these functions is described below. Some examples are given at the end of this subsection.

Command	Syntax	Purpose
CHANGE	({CHA}NGE} s1 s2 s3)	Change one text string (s1) to another (s2) in a third text string (s3).
SUBSTRING	({SUBS}TRING n1 n2 s)	Return substring with characters n1 through n2 of text string (s).
DELETE	({DEL}ETE s1 s2)	Delete a text string (s1) within another text string (s2).
CONCATENATE	({CONC}ATENATE s1 s2 ... sn)	Place end to end text strings. with v101.6, not needed and may not give desired result.
BEFORE	({BEF}ORE s1 s2)	Return the portion of a text string (s2) that occurs before another text string (s1).
REPLY	({REP}LY s1)	Send a message to the terminal and get a reply.
AFTER	({AFT}ER s1 s2)	Return the portion of a text string (s2) that occurs after another text string (s1).

Because of the changes in handling strings in v101.6, code that previously worked will no longr work. For example, to use CONCATENATE to set the station and event names in the title of a plot prior ot v101.6, the following was used:

```
SAC> FUNCGEN SEISMOGRAM
SAC> ECNO ON
SAC> TITLE '(CONCATENATE 'Seismogram of ' &1,KEVNM ' ' &1,KSTNM )'
old output  ==> TITLE 'Seismogram of K8108838 CDV'
v101.6 output  ==>  TITLE "(CONCATENATE " Seismogram of " K8108838 " " CDV )"
```

The best way to do that in v101.6 is much simpler:

```
SAC> title "Seismogram of &1,KEVNM &1,KSTNM"
title "Seismogram of &1,KEVNM &1,KSTNM"
==>  title "Seismogram of K8108838 CDV"
```

CONCATENATE can still be used, but there is usually a better way.:

```
SAC> setbb a (CONCATENATE Seismogram of  &1,KEVNM  &1,KSTNM )
==>  setbb a SeismogramofK8108838CDV
SAC> setbb a (CONCATENATE Seismogram' ' of' '' '  &1,KEVNM  &1,KSTNM )
==>  setbb a Seismogram of  K8108838CDV
SAC> > setbb a 'Seismogram of  &1,KEVNM  &1,KSTNM'
==>  setbb a "Seismogram of  K8108838  CDV"
```

The next examples uses the SUBSTRING function.:

```
SAC> fg seismo
SAC> SETBB MONTH (SUBSTRING 1 3 &1,KZDATE)
==>  SETBB MONTH MAR
SAC> message (substring 1 5 &1,kevnm)
==>  message K8108
setbb VAL "1234567890"
SAC> message (substring 1 5 %VAL)
message (substring 1 5 %VAL)
==>  message 12345
```

The next example uses the REPLY function to control interactively the processing of a set of data files:

```
DO FILE LIST ABC DEF XYZ
   READ $FILE
   DO J FROM 1 TO 10
     MACRO PROCESSFILE
     PLOT
     SETBB RESPONSE (REPLY "Enter -1 to stop, 0 for next file, 1 for same file: ")
     IF %RESPONSE LE 0
       BREAK
     ENDIF
   ENDDO
   IF %RESPONSE LT 0
     BREAK
   ENDIF
ENDDO
```

The outer do loop reads in one file at a time from a list. The inner loop calls a macro to process this file. The inner loop executes up to 10 times. After each execution of the processing macro, the file is plotted, a message is sent to the terminal, and the reply is saved in a blackboard variable. The first IF tests this variable to see if the inner processing loop should be terminated (by executing the BREAK statement) or continued. The second IF tests this same variable to see if the loop on each data file should be terminated or continued. If only one IF test were needed, the REPLY function could be substituted directly into the IF test and a blackboard variable would not be needed.

The next example shows REPLY with a default value:

```
SAC> SETBB BBDAY (REPLY "Enter the day of the week: [Monday]")
```

When this function is executed, the quoted string will appear on the screen, prompting the user for input. If the user types a string, SAC will put the string that the user entered into the blackboard variable BBDAY. If the user simply hits return, SAC will put the default value (in this case, the string "Monday") into BBDAY.

If one copies a set of SAC commands back into SAC, the copied commands will start with SAC>, which is not a part of the command. The parser will remove a doubled SAC> SAC>, so lines like SAC> SAC> read a.sac will be translated into SAC> read a.sac

SAC Data File Format

Overview

This section discusses the contents of the SAC data file, describes the binary and alphanumeric formats of this file, and documents the SAC header in detail.

Since version 100.0, SAC can handle binary data files in either endian (byte order), so big-endian systems (Sun Solaris, Mac PPC) can read SAC data files written on little-endian systems (Linux, MAC i686, Cygwin) and vice versa.

Each signal is stored on disk in a separate SAC data file. These files contain a fixed length header section followed by one or two data sections. The header contains floating point, integer, logical, and character fields. Evenly spaced data files have only one data section which contains the dependent variable. Unevenly spaced data and spectral data files contain two data sections. For unevenly spaced data, the first data section contains the dependent variable and the second contains the independent variable. For spectral files the first component is either the amplitude or the real component and the second component is either the phase or imaginary component.

SAC Binary Format

This is the format that you will use most often. It is used in SAC itself(READ and WRITE commands) and in the subroutine library (RSAC1, RSAC2, WSAC1, WSAC2, WSAC0.) These are binary (unformatted) files so that they can be quickly read from disk into memory. The header is 158 32-bit words in length, followed by the data section(s). In order to rapidly read only a small section of a data file (see the CUT command), these files also have a physical record length of 512 bytes (128 32-bit words) and are opened for direct-access. There is no physical record structure. This format is shown schematically in the following figure.

Structure of SAC Binary Data File

Header Section	First Data Section	Second Data Section (if present)
start word: 0	start word: 158	start word: 158+NPTS
word length: 158	word length: NPTS	word length: NPTS
see table	• dependent variable • amplitude • real component	• independent variable unevenly spaced • phase • imaginary component

SAC Binary Header

The following table shows the contents and layout of the SAC binary data file header. The W and T columns give the beginning word and header data type for the header variables named on that line. These header variables and data types are described later in this section. If the name is INTERNAL then that variable is internal to SAC and not normally of interest to the user. If the name is UNUSED then that variable is not currently being used. For any given file, some of these variables will not have meaningful values. These are refered to as "undefined variables" for that file. For each data type, a special value signifies this undefined state. They are listed in a table at the end of this section.

Header Data Types

This table lists the header types and their definitions. The third column lists the special value used to signify that a particular header variable is undefined in a particular file.

Type	Definition	Undefined	Description
F	Floating	-12345.0	Single precision.
N	Integer	-12345	Name begins with an "N".
I	Enumerated	-12345	Name begins with an "I". Has a limited set of integer values. Each value is given a specific name. Each value represents a specific condition. Subroutines use the equivalent alphanumeric name.
L	Logical	FALSE	Name begins with an "L". Value is either TRUE or FALSE.
K	Alphanumeric	"-12345.."	Name begins with a "K". Either 8 or 16 characters long.
A	Auxilary		Not really in the header. Derived from other header fields.

Header Variables

Word	Type	NAMES	o	o	o	o
0	F	DELTA	DEPMIN	DEPMAX	SCALE	ODELTA
5	F	B	E	O	A	INTERNAL
10	F	T0	T1	T2	T3	T4
15	F	T5	T6	T7	T8	T9
20	F	F	RESP0	RESP1	RESP2	RESP3
25	F	RESP4	RESP5	RESP6	RESP7	RESP8
30	F	RESP9	STLA	STLO	STEL	STDP
35	F	EVLA	EVLO	EVEL	EVDP	MAG
40	F	USER0	USER1	USER2	USER3	USER4
45	F	USER5	USER6	USER7	USER8	USER9
50	F	DIST	AZ	BAZ	GCARC	INTERNAL
55	F	INTERNAL	DEPMEN	CMPAZ	CMPINC	XMINIMUM
60	F	XMAXIMUM	YMINIMUM	YMAXIMUM	UNUSED	UNUSED
65	F	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED
70	I	NZYEAR	NZJDAY	NZHOUR	NZMIN	NZSEC
75	I	NZMSEC	NVHDR	NORID	NEVID	NPTS
80	I	INTERNAL	NWFID	NXSIZE	NYSIZE	UNUSED
85	I	IFTYPE	IDEP	IZTYPE	UNUSED	IINST
90	I	ISTREG	IEVREG	IEVTYP	IQUAL	ISYNTH
95	I	IMAGTYP	IMAGSRC	UNUSED	UNUSED	UNUSED
100	I	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED
105	L	LEVEN	LPSPOL	LOVROK	LCALDA	UNUSED
110	K	KSTNM	KEVNM*			
116	K	KHOLE	KO	KA		
122	K	KT0	KT1	KT2		
128	K	KT3	KT4	KT5		

... continued on next page

Word	Type	NAMES	o	o	o	o
134	K	KT6	KT7	KT8		
140	K	KT9	KF	KUSER0		
146	K	KUSER1	KUSER2	KCMPNM		
152	K	KNETWK	KDATRD	KINST		

KEVNM is 16 characters (4 words) long.

All other K fields are 8 characters (2 words) long.

SAC Alphanumeric Format

This file is essentially the alphanumeric equivalent of the SAC binary data file. The header section is stored on the first 30 cards. This is followed by one or two data sections. The data is in 5G15.7 format. The following table shows the card number, formats and names of the variables on the header section cards.

CN	FORMAT	NAMES	o	o	o	o
01	(5G15.7)	DELTA	DEPMIN	DEPMAX	SCALE	ODELTA
02	(5G15.7)	B	E	O	A	INTERNAL
03	(5G15.7)	T0	T1	T2	T3	T4
04	(5G15.7)	T5	T6	T7	T8	T9
05	(5G15.7)	F	RESP0	RESP1	RESP2	RESP3
06	(5G15.7)	RESP4	RESP5	RESP6	RESP7	RESP8
07	(5G15.7)	RESP9	STLA	STLO	STEL	STDP
08	(5G15.7)	EVLA	EVLO	EVEL	EVDP	MAG
09	(5G15.7)	USER0	USER1	USER2	USER3	USER4
10	(5G15.7)	USER5	USER6	USER7	USER8	USER9
11	(5G15.7)	DIST	AZ	BAZ	GCARC	INTERNAL
12	(5G15.7)	INTERNAL	DEPMEN	CMPAZ	CMPINC	XMINIMUM
13	(5G15.7)	XMAXIMUM	YMINIMUM	YMAXIMUM	ADJTM	UNUSED
14	(5G15.7)	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED
15	(5I10)	NZYEAR	NZJDAY	NZHOUR	NZMIN	NZSEC
16	(5I10)	NZMSEC	NVHDR	NORID	NEVID	NPTS
17	(5I10)	NSPTS	NWFID	NXSIZE	NYSIZE	UNUSED
18	(5I10)	IFTYPE	IDEP	IZTYPE	UNUSED	IINST
19	(5I10)	ISTREG	IEVREG	IEVTYP	IQUAL	ISYNTH
20	(5I10)	IMAGTYP	IMAGSRC	UNUSED	UNUSED	UNUSED
21	(5I10)	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED
22	(5I10)	LEVEN	LPSPOL	LOVROK	LCALDA	UNUSED
23	(A8,A16)	KSTNM	KEVNM			
24	(3A8)	KHOLE	KO	KA		
25	(3A8)	KT0	KT1	KT2		
26	(3A8)	KT3	KT4	KT5		
27	(3A8)	KT6	KT7	KT8		
28	(3A8)	KT9	KF	KUSER0		

... continued on next page

CN	FORMAT	NAMES	o	o	o	o
29	(3A8)	KUSER1	KUSER2	KCMPNM		
30	(3A8)	KNETWK	KDATRD	KINST		

SAC Alphanumeric Data File Example

The header section and first five lines of the data section of a sample SAC alphanumeric data file is shown below. You can reproduce this file (with the entire data section) on your system by executing the following commands:

```
SAC> FUNCGEN SEISMOGRAM
SAC> LH
SAC> WRITE ALPHA TEMP1
```

You can then convert this alphanumeric file to a binary one and read it into SAC with the following commands:

```
SAC> CONVERT FROM ALPHA TEMP1 TO SAC TEMP2
SAC> READ TEMP2
SAC> LH
```

Alternatively, you can accomplish this without using CONVERT, but just using READ and WRITE:

```
SAC> READ ALPHA TEMP1
SAC> WRITE SAC TEMP2
SAC> READ TEMP2
SAC> LH
```

This little test shows the equivalence of the alphanumeric and binary file formats. Listed next are the first lines of TEMP1 -- header plus first 8 lines of data.:

```
0.01000000      -1.569280       1.520640      -12345.00      -12345.00
      9.459999       19.45000      -41.43000       10.46400      -12345.00
     -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
     -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
     -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
     -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
     -12345.00       48.00000      -120.0000      -12345.00      -12345.00
      48.00000      -125.0000      -12345.00       15.00000      -12345.00
     -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
     -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
      373.0627       88.14721       271.8528       3.357465      -12345.00
     -12345.00    -0.09854718       0.000000       0.000000      -12345.00
     -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
     -12345.00      -12345.00      -12345.00      -12345.00      -12345.00
     1981        88        10        38        14
        0         6         0         0      1000
   -12345    -12345    -12345    -12345    -12345
        1        50         9    -12345    -12345
   -12345    -12345        42    -12345    -12345
   -12345    -12345    -12345    -12345    -12345
   -12345    -12345    -12345    -12345    -12345
        1         1         1         1         0
```

```
CDV     K8108838
-12345  -12345  -12345
-12345  -12345  -12345
-12345  -12345  -12345
-12345  -12345  -12345
-12345  -12345  -12345
-12345  -12345  -12345
-12345  -12345  -12345
   -0.09728001    -0.09728001    -0.09856002    -0.09856002    -0.09728001
   -0.09600000    -0.09472002    -0.09344001    -0.09344001    -0.09344001
   -0.09344001    -0.09344001    -0.09472002    -0.09472002    -0.09344001
   -0.09344001    -0.09216000    -0.09216000    -0.09216000    -0.09216000
   -0.09088002    -0.09088002    -0.09216000    -0.09344001    -0.09472002
   -0.09472002    -0.09472002    -0.09472002    -0.09472002    -0.09472002
   -0.09344001    -0.09344001    -0.09216000    -0.09088002    -0.09088002
   -0.09216000    -0.09216000    -0.09216000    -0.09344001    -0.09472002
     ....
```

SAC Header Variables

This table lists the header variables, their types, and descriptions. They are grouped by category: required fields, time fields, phase picks, instrument parameters, station parameters, event parameters, misc. The header types are defined in the second table.

Name	Type	Description
NPTS	N	Number of points per data component. [required]
NVHDR	N	Header version number. Current value is the integer 6. Older version data (NVHDR < 6) are automatically updated when read into sac. [required]
B	F	Beginning value of the independent variable. [required]
E	F	Ending value of the independent variable. [required]
IFTYPE	I	**Type of file [required]:** • ITIME {Time series file} • IRLIM {Spectral file---real and imaginary} • IAMPH {Spectral file---amplitude and phase} • IXY {General x versus y data} • IXYZ {General XYZ (3-D) file}
LEVEN	L	TRUE if data is evenly spaced. [required]
DELTA	F	Increment between evenly spaced samples (nominal value). [required]
ODELTA	F	Observed increment if different from nominal value.

... continued on next page

Name	Type	Description
IDEP	I	**Type of dependent variable:** • IUNKN (Unknown) • IDISP (Displacement in nm) • IVEL (Velocity in nm/sec) • IVOLTS (Velocity in volts) • IACC (Acceleration in nm/sec/sec)
SCALE	F	Multiplying scale factor for dependent variable [not currently used]
DEPMIN	F	Minimum value of dependent variable.
DEPMAX	F	Maximum value of dependent variable.
DEPMEN	F	Mean value of dependent variable.
NZYEAR	N	GMT year corresponding to reference (zero) time in file.
NZJDAY	N	GMT julian day.
NZHOUR	N	GMT hour.
NZMIN	N	GMT minute.
NZSEC	N	GMT second.
NZMSEC	N	GMT millisecond.
NZDTTM	N	GMT date-time array. Six element array equivalenced to NZYEAR, NZJDAY, NZHOUR, NZMIN, NZSEC, and NZMSEC.
KZDATE	A	Alphanumeric form of GMT reference date. Derived from NZYEAR and NZJDAY.
KZTIME	A	Alphanumeric form of GMT reference time. Derived from NZHOUR, NZMIN, NZSEC, and NZMSEC.
IZTYPE	I	**Reference time equivalence:** • IUNKN (Unknown) • IB (Begin time) • IDAY (Midnight of refernece GMT day) • IO (Event origin time) • IA (First arrival time) • ITn (User defined time pick n, n=0,9)
O	F	Event origin time (seconds relative to reference time.)
KO	A	Event origin time identification.

Phase Picks

Name	Type	Description
A	F	First arrival time (seconds relative to reference time.)
KA	K	First arrival time identification.

... continued on next page

Name	Type	Description
F	F	Fini or end of event time (seconds relative to reference time.)
KF	A	Fini identification.
Tn	F	User defined time picks or markers, n = 0 - 9 (seconds relative to reference time).
KT{n}	K	A User defined time pick identifications, n = 0 - 9.

Instrument Fields

Name	Type	Description
KINST	K	Generic name of recording instrument.
IINST	I	Type of recording instrument. [not currently used]
RESPn	F	Instrument response parameters, n=0,9. [not currently used]

Station Fields

Name	Type	Description
KNETWK	K	Name of seismic network.
KSTNM	K	Station name.
ISTREG	I	Station geographic region. [not currently used]
STLA	F	Station latitude (degrees, north positive)
STLO	F	Station longitude (degrees, east positive).
STEL	F	Station elevation above sea level (meters). [not currently used]
STDP	F	Station depth below surface (meters). [not currently used]
CMPAZ	F	Component azimuth (degrees clockwise from north).
CMPINC	F	Component incident angle (degrees from vertical).
KCMPNM	K	Channel name. SEED volumes use three character names, and the third is the component/orientation. For horizontals, the current trend is to use 1 and 2 instead of N and E.
KSTCMP	A	Station component. Derived from KSTNM, CMPAZ, and CMPINC.
LPSPOL	L	TRUE if station components have a positive polarity (left-hand rule).

Event Fields

Name	Type	Description
KEVNM	K	Event name.
IEVREG	I	Event geographic region. [not currently used]
EVLA	F	Event latitude (degrees, north positive).
EVLO	F	Event longitude (degrees, east positive).
EVEL	F	Event elevation (meters). [not currently used]
EVDP	F	Event depth below surface (kilommeters -- previously meters)
MAG	F	Event magnitude.

... continued on next page

Name	Type	Description
IMAGTYP	I	**Magnitude type:** • IMB (Bodywave Magnitude) • IMS (Surfacewave Magnitude) • IML (Local Magnitude) • IMW (Moment Magnitude) • IMD (Duration Magnitude) • IMX (User Defined Magnitude)
IMAGSRC	I	**Source of magnitude information:** • INEIC (National Earthquake Information Center) • IPDE (Preliminary Determination of Epicenter) • IISC (Internation Seismological Centre) • IREB (Reviewed Event Bulletin) • IUSGS (US Geological Survey) • IBRK (UC Berkeley) • ICALTECH (California Institute of Technology) • ILLNL (Lawrence Livermore National Laboratory) • IEVLOC (Event Location (computer program)) • IJSOP (Joint Seismic Observation Program) • IUSER (The individual using SAC2000) • IUNKNOWN (unknown)

... continued on next page

Name	Type	Description
IEVTYP	I	**Type of event:** • IUNKN (Unknown) • INUCL (Nuclear event) • IPREN (Nuclear pre-shot event) • IPOSTN (Nuclear post-shot event) • IQUAKE (Earthquake) • IPREQ (Foreshock) • IPOSTQ (Aftershock) • ICHEM (Chemical explosion) • IQB (Quarry or mine blast confirmed by quarry) • IQB1 (Quarry/mine blast with designed shot info-ripple fired) • IQB2 (Quarry/mine blast with observed shot info-ripple fired) • IQBX (Quarry or mine blast - single shot) • IQMT (Quarry/mining-induced events: tremors and rockbursts) • IEQ (Earthquake) • IEQ1 (Earthquakes in a swarm or aftershock sequence) • IEQ2 (Felt earthquake) • IME (Marine explosion) • IEX (Other explosion) • INU (Nuclear explosion) • INC (Nuclear cavity collapse) • IO (Other source of known origin) • IL (Local event of unknown origin) • IR (Regional event of unknown origin) • IT (Teleseismic event of unknown origin) • IU (Undetermined or conflicting information) • IOTHER (Other)
NEVID	N	Event ID (CSS 3.0)
NORID	N	Origin ID (CSS 3.0)
NWFID	N	Waveform ID (CSS 3.0)
KHOLE	K	Nuclear: hole identifier; Other: location identifier.
DIST	F	Station to event distance (km).
AZ	F	Event to station azimuth (degrees).
BAZ	F	Station to event azimuth (degrees).
GCARC	F	Station to event great circle arc length (degrees).

Miscellaneous Fields

Name	Type	Description
LCALDA	L	TRUE if DIST, AZ, BAZ, and GCARC are to be calculated from station and event coordinates.
IQUAL	I	**Quality of data [not currently used]:** • IGOOD (Good data) • IGLCH (Glitches) • IDROP (Dropouts) • ILOWSN (Low signal to noise ratio) • IOTHER (Other)
ISYNTH	I	**Synthetic data flag [not currently used]:** • IRLDTA (Real data) • ????? (Flags for various synthetic seismogram codes)
KDATRD	K	Date data was read onto computer.
USER{n}	F	User defined variable storage area, n = 0,9.
KUSER{n}	K	User defined variable storage area, n = 0,2.
LOVROK	L	TRUE if it is okay to overwrite this file on disk.
NXSIZE	N	Spectral Length (Spectral files only)
NYSIZE	N	Spectral Width (Spectral files only)
XMINIMUM	F	Minimum value of X (Spectral files only)
XMAXIMUM	F	Maximum value of X (Spectral files only)
YMINIMUM	F	Minimum value of Y (Spectral files only)
YMAXIMUM	F	Maximum value of Y (Spectral files only)

Enumerated Header Field Values

The enumerated header field values are stored in the header as integers. Their names and values are given in the table below.

Name	ID
itime	01
irlim	02
iamph	03
ixy	04
iunkn	05
idisp	06
ivel	07
iacc	08
ib	09

... continued on next page

Name	ID
iday	10
io	11
ia	12
it0	13
it1	14
it2	15
it3	16
it4	17
it5	18
it6	19
it7	20
it8	21
it9	22
iradnv	23
itannv	24
iradev	25
itanev	26
inorth	27
ieast	28
ihorza	29
idown	30
iup	31
illlbb	32
iwwsn1	33
iwwsn2	34
ihglp	35
isro	36
inucl	37
ipren	38
ipostn	39
iquake	40
ipreq	41
ipostq	42
ichem	43
iother	44
igood	45
iglch	46
idrop	47
ilowsn	48
irldta	49
ivolts	50
imb	52
ims	53

... continued on next page

Name	ID
iml	54
imw	55
imd	56
imx	57
ineic	58
ipdeq	59
ipdew	60
ipde	61
iisc	62
ireb	63
iusgs	64
ibrk	65
icaltech	66
illnl	67
ievloc	68
ijsop	69
iuser	70
iunknown	71
iqb	72
iqb1	73
iqb2	74
iqbx	75
iqmt	76
ieq	77
ieq1	78
ieq2	79
ime	80
iex	81
inu	82
inc	83
io_	84
il	85
ir	86
it	87
iu	88
ieq3	89
ieq0	90
iex0	91
iqc	92
iqb0	93
igey	94
ilit	95
imet	96

... continued on next page

Name	ID
iodor	97
ios	103

LATEST REVISION

January 2012 (Version 101.6)

SAC Reading and Writing Routines

Overview

Using the SAC I/O library, ${SACHOME}/lib/libsacio.a, one can write stand-alone codes in C or FORTRAN to read and write SAC formatted data files. The SAC_I/O library is included in the sub-directory ${SACHOME}/lib. The complete listing of sample programs in both C and Fortran for reading and writing SAC data files and for getting and setting SAC header values, are give below in the online version of this file at <http://ds.iris.edu/files/sac-manual/manual/input_output.html>.

When compiling/linking your code, it is necessary to include ${SACHOME}/lib/libsacio.a in order to access the routines discussed below. To ease the requirements for compilation and linking, a helper script is provided, ${SACHOME}/bin/sac-config, which should output the necessary flags and libraries. Try the following:

```
gcc -o program source.c `sac-config --cflags --libs sacio`

f77 -o program source.f `sac-config --cflags --libs sacio`
```

There are two routines in the SAC_I/O library that can be used to read SAC data files into a C or FORTRAN program:

- RSAC1 reads evenly spaced files
- RSAC2 reads unevenly spaced or spectral files.

There is a set of routines that let one get the values of header variables after a file has been read:

- GETFHV gets float or REAL header variables
- GETIHV gets character strings enumerated as int or INTEGER header variables
- GETKHV gets character string header variables
- GETLHV gets LOGICAL header variables (declared as long in C)
- GETNHV gets int or INTEGER header variables.

There is a like set of routines that let one set the values of header variables currently in memory:

- SETFHV sets float or REAL header variables
- SETIHV sets character strings enumerated as int or INTEGER header variables
- SETKHV sets character string header variables
- SETLHV sets LOGICAL header variables (declared as long in C)
- SETNHV sets int or INTEGER header variables.

There are three routines used to write SAC data files to disk:

- WSAC1 writes evenly spaced files
- WSAC2 writes unevenly spaced and spectral files
- WSAC0 writes either format but has more comprehensive header files than the other two

WSAC1 and WSAC2 write SAC files with a minimum header contains only those variables needed to be able to read the file: B, E, DELTA, LEVEN, and NPTS. For calls to WSAC0, If it is a new file, it requires a call to subroutine NEWHDR supplemented by additional header variables to be set using the SETXXX routines (see examples below). If it is writing to a file that is based on one that had been read in previously in the program, one should not call NEWHDR. Before writing such a file using WSAC0, SAC updates the header variables such as DEPMAX and BAZ. As shown in the examples below, the type of SAC data file that gets written depends on header variables that must be set: IFTYPE and LEVEN. IFTYPE has the following values:

- ITIME {Time series file}
- IRLIM {Spectral file---real and imaginary}
- IAMPH {Spectral file---amplitude and phase}
- IXY {General x versus y data}
- IXYZ {General XYZ (3-D) file}

LEVEN should be set to TRUE unless the IFTYPE is IXY.

Reading a Evenly-Sampled SAC File

This routine will be used 95% of the time as most SAC files are of the evenly time sampled variety. Using rsac1(), the time sampling, beginning time, and amplitude data are returned and the remainder of the header values are held in memory for later access or until the next call to rsac1().

Fortran Example

```
      program rsac
      implicit none

!     Define the Maximum size of the data Array
      integer MAX
      parameter (MAX=1000)

!     Define the Data Array of size MAX
      real yarray
      dimension yarray(MAX)

!     Declare Variables used in the rsac1() subroutine
      real beg, del
      integer nlen
      character*10 KNAME
      integer nerr

!     Define the file to be read      g
      kname = 'FILE1'

!     Call rsac1 to read filename kname
!        - Data is loaded into yarray
!        - Length of data is stored in nlen
!        - Begining time and time sampling are in beg and del
!        - MAX is the maximum number of points to be read in
!        - nerr is the Error return flag
      call rsac1(kname, yarray, nlen, beg, del, MAX, nerr)

!     Check the error status, nerr
!        - 0 on Success
!        - Non-Zero on Failure
      if(nerr .NE. 0) then
          write(*,*)'Error reading in file: ',kname
          call exit(-1)
      endif
```

```
!     Do some processing ....

      call exit(0)
      end
```

Be sure to check the error value after the return from rsac1(). This will help solve a number of unforeseen problems in the future.

Reading a Evenly-Sampled SAC File: C Example

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include <sacio.h>

/* Define the maximum length of the data array */
#define MAX 1000

int
main(int argc, char **argv)
{
  /* Define variables to be used in the call to rsac1() */
  float yarray[MAX], beg, del;
  int nlen, nerr, max = MAX;
  char kname[ 11 ] ;

  /* Copy the name of the file to be read into kname */
  strcpy( kname , "FILE1" ) ;


  /* Call rsac1 to read filename kname
     - Data is loaded into yarray
     - Length of data is stored in nlen
     - Begining time and time sampling are in beg and del
     - max is the maximum number of points to be read in
     - nerr is the error return flag
     - strlen( kname ) is the length of character array kname
     All variables are passed as references either
         arrays like kname and yarray or
         using &varible to pass reference to variable
  */
  rsac1( kname, yarray, &nlen, &beg, &del, &max, &nerr, strlen( kname ) ) ;

  /* Check the error status, nerr
     - 0 on Success
     - Non-Zero on Failure
  */
  if ( nerr != 0 ) {
    fprintf(stderr, "Error reading in SAC file: %s\n", kname);
    exit ( nerr ) ;
  }
```

```
  /* Do some processing ... */

  exit(0);
}
```

Note that in the call to rsac1() in C there is an extra parameter after nerr. This is the string length specifier which specifies the length of the string kname. The length of the string does not include a null terminator. Note also that all of the parameters are passed by reference except the string length specifier.

Reading an Unevenly-Sampled or Spectral SAC File

In routine rsac2() for a spectral file, the time and amplitude data are both stored and returned on the call to rsac2(). For an unevenly-sampled file, the first array is the independent variable and the second one the dependent variable. Unlike rsac1(), the beginning time and time sampling are not returned as they can be determined from the returned time data.

Fortran Example

```
      program rsac_2
      implicit none

!     Define the Maximum size of the data Array
      integer MAX
      parameter (MAX=3000)

!     Define the Time and Amplitude arrays of zize MAX
      real xarray, yarray
      dimension xarray(MAX), yarray(MAX)

!     Declare Variables used in the rsac2() subroutine
      character*10 kname
      integer nlen
      integer nerr

!     Define the file to be read
      kname='file2'

!     Call rsac2 to read filename kname
!        - Amplitude Data is loaded into yarray
!        - Length of data is stored in nlen
!        - Time Data is loaded into xarray
!        - MAX is the maximum number of points to be read in
!        - nerr is the Error return flag
      call rsac2(kname,yarray,nlen,xarray,MAX,nerr)

!     Check the error status, nerr
!        - 0 on Success
!        - Non-Zero on Failure
      if(nerr .ne. 0) then
         write(*,*)'error reading in sac file: ',kname
         call exit(-1)
      endif
```

```
!     Do some processing ....

      call exit(0)
      end
```

Reading a Spectral SAC File: C Example

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include <sacio.h>

/* Define the maximum length of the data and time array */
#define MAX 3000

int
main(int argc, char *argv[])
{
  /* Define variables to be used in the call to rsac2() */
  float xarray[MAX], yarray[MAX];
  int nlen, nerr, max;
  char kname[ 11 ] ;

  max = MAX;

  /* Copy the name of the file to be read into kname */
  strcpy(kname, "FILE2") ;

  /* Call rsac1 to read filename kname
     - Amplitude Data is loaded into yarray
     - Length of data is stored in nlen
     - Time Data is loaded into xarray
     - max is the maximum number of points to be read in
     - nerr is the error return flag
     - strlen( kname ) is the length of character array kname
     All variables are passed as references either
         arrays like kname and yarray or
         using &varible to pass reference to variable
  */
  rsac2(kname, yarray, &nlen, xarray, &max, &nerr, strlen( kname )) ;

  /* Check the error status, nerr
     - 0 on Success
     - Non-Zero on Failure
  */
  if ( nerr > 0 ) {
    fprintf(stderr, "Error reading in SAC file: %s\n", kname);
    exit(nerr) ;
  }

  /* Do some processing ... */
```

```
  exit(0);
}
```

Accessing Header Variables

Accessing the header variables following either rsac1() or rsac2() is straight forward. Depending on the type of variable requested, the routine called will be different.

Fortran Example

```
      program rsac
      implicit none

!     Define the Maximum size of the data Array
      integer max
      parameter (MAX=1000)

!     Define the Data Array of size MAX
      real yarray
      dimension yarray(MAX)

!     Declare Variables used in the rsac1() and getfhv() subroutines
      character*10 kname
      integer nlen
      real beg, del
      integer nerr
      integer n1, n2
      real delta, b, t1, t2

!     Define the file to be read
      kname='file1'

!     Read in the SAC File
      call rsac1(kname,yarray,nlen,beg,del,MAX,nerr)

!     Check the Error status
      if(nerr .ne. 0) then
         write(*,*)'Error reading SAC file: ',kname
         call exit(-1)
      endif

!     Get floating point header value: Delta
!        'delta' - name of the header variable requested
!        delta   - value of the header variable delta, returned
!        nerr    - Error return flag
      call getfhv('delta',delta,nerr)
      if(nerr .ne. 0) then
         write(*,*)'Error reading variable: delta'
         call exit(-1)
      endif

!     Get floating point header value: B
```

```
      call getfhv('b',b,nerr)
      if(nerr .ne. 0) then
         write(*,*)'Error reading variable: b'
         call exit(-1)
      endif

!     Get floating point header value: t1
      call getfhv('t1',t1,nerr)
      if(nerr .ne. 0) then
         write(*,*)'Error reading variable: t1'
         call exit(-1)
      endif

!     Get floating point header value: t2
      call getfhv('t2',t2,nerr)
      if(nerr .ne. 0) then
         write(*,*)'Error reading variable: t2'
         call exit(-1)
      endif

!     Compute the time sample at which t1 and t2 occur
      n1 = int((t1 - b) / delta)
      n2 = int((t2 - b) / delta)

!     ......

      call exit(0)
      end
```

Accessing Header Variables: C Example

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <sacio.h>

/* Define the maximum length of the data array */
#define MAX 1000

int
main(int argc, char *argv[])
{
  /* Define variables to be used in the call to rsac1() and getfhv() */
  int max = MAX, nlen, nerr, n1, n2;
  float yarray[ MAX ] , beg , del , delta , B , T1 , T2 ;
  char kname[ 11 ] ;

  /* Copy the name of the file to be read into kname */
  strcpy ( kname , "FILE1" ) ;

  /* Read in the SAC File */
  rsac1( kname, yarray, & nlen, & beg, & del, & max, & nerr, strlen( kname ) ) ;
```

```
  /* Check the Error status */
  if ( nerr != 0 ) {
    fprintf(stderr, "Error reading SAC file: %s\n", kname);
    exit(-1);
  }

  /* Get floating point header value: Delta
     "DELTA" - name of the header variable requested
      delta  - value of the header variable delta, returned
      nerr   - Error return flag
      strlen("DELTA") - Length of the character array "DELTA"
  */
  getfhv ( "DELTA" , & delta , & nerr , strlen("DELTA") ) ;
  /* Check the Return Value */
  if ( nerr != 0 ) {
    fprintf(stderr, "Error getting header variable: delta\n");
    exit(-1);
  }

  /* Get floating point header value: B */
  getfhv ( "B" , &B , & nerr , strlen("B") ) ;
  if ( nerr != 0 ) {
    fprintf(stderr, "Error getting header variable: b\n");
    exit(-1);
  }

  /* Get floating point header value: T1 */
  getfhv ( "T1" , & T1 , & nerr , strlen("T1") ) ;
  if ( nerr != 0 ) {
    fprintf(stderr, "Error getting header variable: t1\n");
    exit(-1);
  }

  /* Get floating point header value: T2 */
  getfhv ( "T2" , & T2 , & nerr , strlen("T2") ) ;
  if ( nerr != 0 ) {
    fprintf(stderr, "Error getting header variable: t2\n");
    exit(-1);
  }

  /* Compute the time sample at which t1 and t2 occur  */
  n1 = (int) ( ( ( T1 - B ) / delta ) + 0.5 ) ;
  n2 = (int) ( ( ( T2 - B ) / delta ) + 0.5 ) ;

  /* ... */

  exit(0);

}
```

Writing an Evenly-Spaced SAC File

Fortran Example

```
      program wsac
      implicit none

!     Define the Maximum size of data array
      integer MAX
      parameter (MAX=200)

!     Define the data array
      real yfunc
      dimension yfunc(MAX)

!     Define variables to be passed to wsac1()
      character*10 kname
      integer j
      integer nerr
      real beg
      real del
      real x

!     Define the file to be written, the beginning time
!     time sampling, and the initial value
      kname = 'expdata'
      beg   = 0.00
      del   = 0.02
      x     = beg

!     Create the Amplitude data, an Exponential
      do j=1,MAX
         yfunc(j)=exp(-x)
         x=x+del
      enddo

!     Write the SAC file kname
!       - kname holds the name of the file to be written
!       - yfunc Input Amplitude data
!       - MAX number of points to be written
!       - beg Beginning Time of the data
!       - del Time Sampling of the series
!       - nerr Error return Flag
      call wsac1(kname,yfunc,MAX,beg,del,nerr)

!     Check the Error status
!       - 0 on Success
!       - Non-Zero on Error
      if(nerr .NE. 0) then
         write(*,*)'Error writing SAC File: ', kname
         call exit(-1)
      endif

      call exit(0)
```

```
      end
```

Writing an Evenly-Spaced SAC File: C Example

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#include <sacio.h>

/* Define the Maximum size of data array */
#define MAX 200

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

  /* Define variables to be passed to wsac1() */
  int max, j, nerr;
  float yfunc[ MAX ], x, beg, del;
  char kname[ 10 ];

  max = MAX;

  /* Define the file to be written, the beginning time
     time sampling, and the initial value
  */
  strcpy ( kname , "expdata" ) ;
  beg = 0.00;
  del = 0.02;
  x = beg;

  /* Create the Amplitude data, an Exponential */
  for ( j = 0; j < MAX ; j++ ) {
    yfunc[ j ] = exp ( -x ) ;
    x = x + del;
  }

  /* Write the SAC file kname
     - kname holds the name of the file to be written
     - yfunc Input Amplitude data
     - max number of points to be writtne
     - beg Beginning Time of the data
     - del Time Sampling of the series
     - nerr Error return Flag
     - strlen(kname) Length of the character array kname
  */
  wsac1 (kname, yfunc, &max, &beg, &del, &nerr, strlen( kname )) ;

  /* Check the Error status
     - 0 on Success
     - Non-Zero on Error
```

```
  */
  if(nerr != 0) {
    fprintf(stderr, "Error writing SAC File: %s\n", kname);
    exit(-1);
  }

  exit(0);
}
```

Writing an Unevenly-Spaced or Spectral SAC File

Fortran Example

```
      program wsac2f
      implicit none

!     Define the Maximum size of the data arrays      p
      integer MAX
      parameter (MAX=300)

!     Define both data arrays, time and amplitude
      real xdata, ydata
      dimension xdata(MAX), ydata(MAX)

!     Define the varaibles used in the call to wsac2()
      character*10 kname
      integer j
      integer nerr

!     Set the name the file to be written and initial x value
      kname='expdata    '
      xdata(1) = 0.1

!     Create the Amplitude and Time, an Exponential
!     Best viewed with axis as loglin
      ydata(1) = exp(-xdata(1))
      do j=2,MAX
         xdata(j) = xdata(j-1) + xdata(j-1) * 1.0/(4.0 * 3.1415);
         ydata(j) = exp(-xdata(j))
      enddo

!     Write the SAC file kname
!       - kname holds the name of the file to be written
!       - yfunc Input Amplitude Data
!       - MAX number of points to be written
!       - xdata Input Time Data
!       - nerr Error return Flag
      call wsac2(kname,ydata,MAX,xdata,nerr)

!     Check the Error status
!       - 0 on Success
!       - Non-Zero on Error
```

```
      if(nerr .NE. 0) then
         write(*,*)'Error writing SAC File: ', kname,nerr
         call exit(-1)
      endif

      call exit(0)

      end
```

Writing an Unevenly-Spaced or Spectral SAC File: C Example

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#include <sacio.h>

/* Define the Maximum size of the data arrays  */
#define MAX 300

int
main(int argc, char *argv[])
{
  /* Define the varaibles used in the call to wsac2() */
  float xdata[MAX], ydata[MAX] ;
  int max, nerr;
  char kname[ 11 ];
  int j;

  max = MAX;

  /* Set the name the file to be written and initial x value */
  strcpy ( kname , "expdata" ) ;
  xdata[0] = 0.1;

  /*  Create the Amplitude and Time, an Exponential
   *  Best viewed with axis as loglin
   */
  ydata[0] = exp(-xdata[0]);
  for(j = 1; j < max; j++) {
    xdata[j] = xdata[j-1] + xdata[j-1] * 1/(4 * M_PI);
    ydata[j] = exp(-xdata[j]);
  }

  /* Write the SAC file kname
     - kname holds the name of the file to be written
     - yfunc Input Amplitude Data
     - max number of points to be written
     - xdata Input Time Data
     - nerr Error return Flag
     - strlen(kname) Length of character string kname
```

```
  */
  wsac2(kname, ydata, &max, xdata, &nerr, strlen( kname )) ;


  /* Check the Error status
     - 0 on Success
     - Non-Zero on Error
  */
  if(nerr != 0) {
    fprintf(stderr, "Error writing SAC File: %s\n", kname);
    exit(-1);
  }

  exit(0);

}
```

Writing a File with a Comprehensive Header

To create a SAC data file with more information in the header than WSAC1 and WSAC2 allow, you need to use a set of subroutines that store header variables and then use WSAC0. Below are three examples, the first is similar to the example for WSAC2.

Writing Unevenly-Spaced Data: Fortran

```
      program wsac3f
      implicit none

!     Define the Maximum size of the data arrays      p
      integer MAX
      parameter (MAX=300)

!     Define both data arrays, time and amplitude
      real xdata, ydata
      dimension xdata(MAX), ydata(MAX)

!     Define the varaibles used in the call to wsac2()
      character*10 kname
      integer j
      integer nerr
      real cona, conb

!     Set the name the file to be written and initial x value
      kname='expdata    '
      xdata(1) = 0.1
      cona     = 12.3
      conb     = -45.6

!     Create the Amplitude and Time, an Exponential
!     Best viewed with axis as loglin
      ydata(1) = exp(-xdata(1))
      do j=2,MAX
         xdata(j) = xdata(j-1) + xdata(j-1) * 1.0/(4.0 * 3.1415);
```

```
         ydata(j) = exp(-xdata(j))
      enddo

!     Create a New Header to store more information
!     Newly created header value are set to a default state
      call newhdr()

!     Store values in the newly created header
!     You must define the following header variables
!        - delta  Time Sampling
!                 Only if the file is evenly spaced
!        - b      Beginning Time
!        - e      Ending Time
!        - npts   Number of Points in the File
!        - iftype File Type
!             - itime Time Series File
!             - irlim Spectral File Real/Imaginary
!             - iamph Spectral File Amplitue/Phase
!             - ixy   X-Y File
!             - iunkn Unknown
!
!     All other variables are up to the user
      call setnhv('npts',    max,        nerr)
      call setlhv('leven',   .false.,    nerr)
      call setfhv('b',       xdata(1),   nerr)
      call setfhv('e',       xdata(max), nerr)
      call setihv('iftype',  'ixy',      nerr)
      call setfhv('user0',   cona,       nerr)
      call setfhv('user1',   conb,       nerr)
      call setkhv('kuser0', 'gendat',    nerr)

!     Write the SAC file kname
!       - kname holds the name of the file to be written
!       - xdata Input Time Data
!       - yfunc Input Amplitude Data
!       - nerr Error return Flag
      call wsac0(kname,xdata,ydata,nerr)

!     Check the Error status
!       - 0 on Success
!       - Non-Zero on Error
      if(nerr .NE. 0) then
         write(*,*)'Error writing SAC File: ', kname,nerr
         call exit(-1)
      endif

      call exit(0)

      end
```

Writing Unevenly-Spaced Data: C

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#include <sacio.h>

/* Define the Maximum size of the data arrays  */
#define MAX 300

int
main(int argc, char *argv[])
{
  /* Define the varaibles used in the call to wsac2() */
  float xdata[MAX], ydata[MAX] ;
  int max, nerr;
  char kname[ 11 ];
  int j;
  int leven;
  float cona, conb;

  max = MAX;

  /* Set the name the file to be written and initial x value */
  strcpy ( kname , "expdata" ) ;
  xdata[0] = 0.1;
  leven    = 0;
  cona     = 12.3;
  conb     = -45.6;

  /*  Create the Amplitude and Time, an Exponential
   *  Best viewed with axis as loglin
   */
  ydata[0] = exp(-xdata[0]);
  for(j = 1; j < max; j++) {
    xdata[j] = xdata[j-1] + xdata[j-1] * 1/(4 * M_PI);
    ydata[j] = exp(-xdata[j]);
  }

  /* Create a New Header to store more information
     Newly created header value are set to a default state
  */
  newhdr();

  /* Store values in the newly created header
     You must define the following header variables
     - delta  Time Sampling
              Only if the file is evenly spaced
     - b      Beginning Time
     - e      Ending Time
     - npts   Number of Points in the File
     - iftype File Type
          - itime Time Series File
          - irlim Spectral File Real/Imaginary
```

```
            - iamph Spectral File Amplitue/Phase
            - ixy   X-Y File
            - iunkn Unknown

         All other variables are up to the user
      */
      setnhv ( "npts",   &max,            &nerr, strlen("npts"));
      setlhv ( "leven",  &leven,          &nerr, strlen("leven"));
      setfhv ( "b",      &(xdata[0]),     &nerr, strlen("b"));
      setfhv ( "e",      &(xdata[max-1]), &nerr, strlen("e"));
      setihv ( "iftype", "ixy",           &nerr, strlen("iftype"), strlen("ixy"));
      setfhv ( "user0",  &cona,           &nerr, strlen("user0"));
      setfhv ( "user1",  &conb,           &nerr, strlen("user1"));
      setkhv ( "kuser0", "gendat",        &nerr, strlen("kuser0"), strlen("gendat"));

      /* Write the SAC file kname
         - kname holds the name of the file to be written
         - xdata Input Time Data
         - yfunc Input Amplitude Data
         - nerr Error return Flag
         - strlen(kname) Length of character string kname
      */
      wsac0(kname, xdata, ydata, &nerr, strlen( kname )) ;

      /* Check the Error status
         - 0 on Success
         - Non-Zero on Error
      */
      if(nerr != 0) {
        fprintf(stderr, "Error writing SAC File: %s\n", kname);
        exit(-1);
      }

      exit(0);

    }
```

XYZ (3-D) Files: Fortran

```
          program wsac
          implicit none

    !     Maximum Size of Array, in 2-D
          integer MAX
          parameter (MAX=36)

    !     Size of arrays to store the data
          real dummy, zdata
          dimension dummy(MAX), zdata(MAX)

    !     Define variables to be passed into wsac0()
          character*10 kname
          integer i, j, k
          integer nerr
```

```
      integer nx, ny
      real minimum, maximum

!     Define the file to be written and the min and max of the 2-D Array
      kname   = 'xyzdata'
      minimum = 1.0
      maximum = 6.0
      nx      = 6
      ny      = 6

      ! Create the 2D Data
      k = 1
      do i = 1,nx
         do j = 1,ny
            zdata(k) = sqrt(j * 1.0 * j + i * 1.0 * i)
            k = k + 1
         enddo
      enddo

      ! Create a new Header and fill it
      !   We are defining the data type, iftype to be 'ixyz', a 2-D Array
      call newhdr
      call setnhv('npts',     MAX,     nerr)
      call setlhv('leven',    .true.,  nerr)
      call setihv('iftype',   'ixyz',  nerr)
      call setnhv('nxsize',   nx,      nerr)
      call setnhv('nysize',   ny,      nerr)
      call setfhv('xminimum', minimum, nerr)
      call setfhv('xmaximum', maximum, nerr)
      call setfhv('yminimum', minimum, nerr)
      call setfhv('ymaximum', maximum, nerr)

!     Write the SAC file kname
!       - kname holds the name of the file to be written
!       - dummy Input Amplitude Data
!       - zdata Input Time Data
!       - nerr Error return Flag
      call wsac0(kname,dummy,zdata,nerr)

!     Check the Error status
!       - 0 on Success
!       - Non-Zero on Error
      if(nerr .NE. 0) then
         write(*,*)'Error writing SAC File: ', kname,nerr
         call exit(-1)
      endif

      call exit(0)

      end
```

Although data in SAC memory is stored in a linear 1-D array, one should think of the Z data as being placed in a 2-D grid, in the order left-to-right, bottom-to-top. See the CONTOUR command for additional information.

XYZ (3-D) Files: C

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#include <sacio.h>

#define MAX 36

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

  /*  Maximum Size of Array, in 2-D */
  int max;

  /* Size of arrays to store the data */
  float dummy[MAX], zdata[MAX];

  /*  Define variables to be passed into wsac0 */
  char kname[10];
  int i, j, k;
  int nerr;
  int nx, ny;
  int leven;
  float minimum, maximum;

  /*  Define the file to be written and the min and max of the 2-D Array */
  strcpy(kname, "xyzdata");
  max     = MAX;
  minimum = 1.0;
  maximum = 6.0;
  nx      = 6;
  ny      = 6;
  leven   = 1;

  /* Create the 2D Data */
  k = 0;
  for(i = minimum-1; i < maximum; i++) {
    for(j = minimum-1; j < maximum; j++) {
      zdata[k] = sqrt(i * i + j * j);
      k = k + 1;
    }
  }

  /* Create a new Header and fill it
     We are defining the data type, iftype to be 'ixyz', a 2-D Array
  */
  newhdr();
  setnhv("npts",     &max,     &nerr, strlen("npts"));
  setlhv("leven",    &leven,   &nerr, strlen("leven"));
```

```
    setihv("iftype",   "ixyz",   &nerr, strlen("iftype"), strlen("ixyz"));
    setnhv("nxsize",   &nx,      &nerr, strlen("nxsize"));
    setnhv("nysize",   &ny,      &nerr, strlen("nysize"));
    setfhv("xminimum", &minimum, &nerr, strlen("xminimum"));
    setfhv("xmaximum", &maximum, &nerr, strlen("xmaximum"));
    setfhv("yminimum", &minimum, &nerr, strlen("yminimum"));
    setfhv("ymaximum", &maximum, &nerr, strlen("ymaximum"));
    /* Write the SAC file kname
       - kname holds the name of the file to be written
       - dummy Input Amplitude Data
       - zdata Input Time Data
       - nerr Error return Flag
    */

    wsac0(kname, dummy, zdata, &nerr,strlen(kname));

    /* Check the Error status
       - 0 on Success
       - Non-Zero on Error
    */
    if(nerr != 0) {
      fprintf(stderr, "Error writing SAC File: %s %d\n", kname,nerr);
      exit(-1);
    }

    exit(0);

}
```

Evenly-Spaced Data: Fortran

```
      program wsac5f
      implicit none

      integer NCOMP
      parameter(NCOMP=11)

      integer NDATA
      parameter(NDATA=4000)

      real sdata(NDATA,NCOMP+1), xdummy(NDATA)
      CHARACTER KNAME(NCOMP+1)*10
      real evla, evlo, stla, stlo
      character*11 kevnm, kstnm
      real b, delta
      real cmpaz, cmpinc
      integer npts
      integer nerr, j, i

      DATA KNAME/'STAZ','STBZ','STCZ','STDZ','STEZ',
     &           'STFZ','STGZ','STHZ','STHN','STHE','STHN','STNQ' /

      b      = 0.0
      delta  = 0.25
```

```
cmpaz  = 0.0
cmpinc = 0.0
npts  = NDATA
evla   = -23.56
evlo   = 123.56

call newhdr () ;
call setihv("IFTYPE", "ITIME", nerr)
call setihv("IZTYPE", "IB",    nerr)
call setfhv("B",      b,       nerr)
call setlhv("LEVEN",  .TRUE.,  nerr)
call setfhv("DELTA",  delta,   nerr)

kevnm = "Event Name"

call setnhv("NPTS",   npts,  nerr)
call setfhv("EVLA",   evla,   nerr)
call setfhv("EVLO",   evlo,   nerr)
call setkhv("KEVNM",  kevnm,  nerr)
call setfhv("CMPAZ",  cmpaz,  nerr)
call setfhv("CMPINC", cmpinc, nerr)

do j = 1,NCOMP-2
   kstnm = kname(j)
   call setkhv ( "KSTNM", kstnm, nerr)
   stla = j * 10
   stlo = j * 20
   do i = 1,NDATA
      sdata(i,j) = 1.0 * rand()
   enddo
   call setfhv ( "STLA" , stla , nerr )
   call setfhv ( "STLO" , stlo , nerr )
   call wsac0 ( kstnm, xdummy, sdata(1,j), nerr)
enddo

cmpinc = 90.0
call setfhv("CMPINC", cmpinc, nerr)
j = 9
do i = 1,NDATA
   sdata(i,j) = 1.0 * rand()
enddo
call wsac0(kname(9), xdummy, sdata(1,9), nerr)

cmpaz = 90.0
call setfhv("CMPAZ", cmpaz, nerr)
j = 10
do i = 1,NDATA
   sdata(i,j) = 1.0 * rand()
enddo
call wsac0(kname(10), xdummy, sdata(1,10), nerr)

end
```

Evenly-Spaced Data: C

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <sacio.h>

#define NCOMP   11
#define NDATA   4000
#define NSTA    11
#define FALSE   0
#define TRUE    1

int
main(int argc, char *argv[])
{
  float sdata[NCOMP][NDATA], xdummy[NDATA];
  float evla, evlo, stla, stlo;
  char kevnm[NSTA] , *kstnm ;
  int nerr, ndata, j, i;
  float b, delta;
  float cmpaz, cmpinc;

  char kname[NCOMP][NSTA] = { "STAZ" , "STBZ" , "STCZ" , "STDZ" , "STEZ" ,
                              "STFZ" , "STGZ" , "STHZ" , "STHN" , "STHE", "STHN" } ;

  int true  = TRUE;

  b      = 0.0;
  delta  = 0.25;
  cmpaz  = 0.0;
  cmpinc = 0.0;
  ndata  = NDATA;
  evla   = -23.56;
  evlo   = 123.56;

  newhdr () ;
  setihv("IFTYPE", "ITIME", &nerr , strlen("IFTYPE"), strlen("ITIME"));
  setihv("IZTYPE", "IB",    &nerr , strlen("IZTYPE"), strlen("IB"));
  setfhv("B",      &b,      &nerr , strlen("B"));
  setlhv("LEVEN",  &true,   &nerr , strlen("LEVEN"));
  setfhv("DELTA",  &delta,  &nerr , strlen("DELTA")) ;

  strcpy(kevnm, "Event Name");

  setnhv("NPTS",   &ndata,    &nerr, strlen("NPTS"));
  setfhv("EVLA",   &evla,     &nerr, strlen("EVLA"));
  setfhv("EVLO",   &evlo,     &nerr, strlen("EVLO"));
  setkhv("KEVNM",  &kevnm[0], &nerr, strlen("KEVNM"), SAC_STRING_LENGTH);
  setfhv("CMPAZ",  &cmpaz,    &nerr, strlen("CMPAZ"));
  setfhv("CMPINC", &cmpinc,   &nerr, strlen("CMPINC"));
```

```
  for ( j = 0 ; j < NCOMP - 2 ; j++ )
    {
      kstnm = kname[j]  ;
      setkhv ( "KSTNM", kstnm, &nerr, strlen("KSTNM"), strlen(kstnm));
      stla = j * 10;
      stlo = j * 20;
      for(i = 0; i < NDATA; i++) {
        sdata[j][i] = 1.0 * rand()/INT32_MAX;
      }
      setfhv ( "STLA" , &stla , &nerr , strlen("STLA"));
      setfhv ( "STLO" , &stlo , &nerr , strlen("STLO"));
      wsac0 ( kstnm, xdummy, sdata[j], &nerr, strlen(kstnm));
    }

  cmpinc = 90.0;
  setfhv("CMPINC", &cmpinc, &nerr, strlen("CMPINC")) ;
  j = 9;
  for(i = 0; i < NDATA; i++) {
    sdata[j][i] = 1.0 * rand()/INT32_MAX;
  }
  wsac0(kname[9], xdummy, sdata[9], &nerr, strlen(kname[9]));

  cmpaz = 90.0;
  setfhv("CMPAZ", &cmpaz, &nerr, strlen("CMPAZ")) ;
  j = 10;
  for(i = 0; i < NDATA; i++) {
    sdata[j][i] = 1.0 * rand()/INT32_MAX;
  }
  wsac0(kname[10], xdummy, sdata[10], &nerr, strlen(kname[10]));

  return 0;
}
```

Using the SAC Library

Overview

In addition to being able to read and write SAC data files in one's own C or FORTRAN programs (see SAC Reading and Writing Routines), one can use many of SAC's data-processing routines in stand-alone codes if one uses specific flags in the compiling stage and the SAC libraries in the linking stage. These libraries libsac.a and libsacio.a are in ${SACHOME}/lib.

If ${SACHOME} is /usr/local/sac, the following commands will compile the Fortran anc C kprograms for filter.f and filterc.c respectively:

```
f77 -o filterf filter.f -I/usr/local/sac/include  -L/usr/local/sac/lib  -lsacio -lsac

gcc -o filterc filterc.c -I/usr/local/sac/include  -L/usr/local/sac/lib  -lsacio -lsac
```

Filtering - Fortran

```
      program filter
      implicit none

      include "sacf.h"

!     Define the Maximum size of the data Array
      integer MAX
      parameter (MAX=1000)

!     Define the Data Array of size MAX
      real yarray, xarray
      dimension yarray(MAX)

!     Declare Variables used in the rsac1() subroutine
      real beg, delta
      integer nlen
      character*20 KNAME
      integer nerr

!     Define variables used in the filtering routine
      real *8 low, high
      real *8 transition_bandwidth, attenuation
      real *8 delta_d
      integer order, passes

      kname = 'filterf_in.sac'

      call rsac1(kname, yarray, nlen, beg, delta, MAX, nerr)
      delta_d = delta

      if(nerr .NE. 0) then
          write(*,*)'Error reading in file: ',kname
          call exit(-1)
      endif
```

```
      low    = 0.10
      high   = 1.00
      passes = 2
      order  = 4
      transition_bandwidth = 0.0
      attenuation = 0.0
!     Call xapiir ( Apply a IIR Filter )
!        - yarray - Original Data
!        - nlen   - Number of points in yarray
!        - proto  - Prototype of Filter
!                 - SAC_FILTER_BUTTERWORK        - Butterworth
!                 - SAC_FILTER_BESSEL            - Bessel
!                 - SAC_FILTER_CHEBYSHEV_TYPE_I  - Chebyshev Type I
!                 - SAC_FILTER_CHEBYSHEV_TYPE_II - Chebyshev Type II
!        - transition_bandwidth (Only for Chebyshev Filter)
!                 - Bandwidth as a fraction of the lowpass prototype
!                   cutoff frequency
!        - attenuation (Only for Chebyshev Filter)
!                 - Attenuation factor, equals amplitude reached at
!                   stopband egde
!        - order  - Number of poles or order of the analog prototype
!                   4 - 5 should be ample
!                   Cannot exceed 10
!        - type   - Type of Filter
!                 - SAC_FILTER_BANDPASS
!                 - SAC_FILTER_BANDREJECT
!                 - SAC_FILTER_LOWPASS
!                 - SAC_FILTER_HIGHPASS
!        - low    - Low Frequency Cutoff [ Hertz ]
!                   Ignored on SAC_FILTER_LOWPASS
!        - high   - High Frequency Cutoff [ Hertz ]
!                   Ignored on SAC_FILTER_HIGHPASS
!        - delta  - Sampling Interval [ seconds ]
!        - passes - Number of passes
!                 - 1 Forward filter only
!                 - 2 Forward and reverse (i.e. zero-phase) filtering
!
      call xapiir(yarray, nlen,
     +            SAC_BUTTERWORTH,
     +            transition_bandwidth, attenuation,
     +            order,
     +            SAC_BANDPASS,
     +            low, high, delta_d, passes)

!     Do more processing ....

      xarray = 0
      kname='filterf_out.sac'
!     Write the SAC file kname
!       - kname holds the name of the file to be written
!       - xdata Input Time Data
!       - yfunc Input Amplitude Data
!       - nerr Error return Flag
      call wsac0(kname,xarray,yarray,nerr)
```

```
      if(nerr .NE. 0) then
          write(*,*)'Error writing out file: ',kname
          call exit(-1)
      endif

      call exit(0)

      end program filter
```

Filtering - C

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include "sac.h"
#include "sacio.h"

#define MAX 1000

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

    /* Local variables */
    double low, high, attenuation, transition_bandwidth;;
    int nlen, nerr, max;

    float beg, delta;
    double delta_d;
    char *kname;
    int order;

    int passes;
    float yarray[1000];
    float xarray[1];

    max = MAX;

    /*     Define the Maximum size of the data Array
     * Filter Prototypes
     * Filter Types
     *     Define the Data Array of size MAX
     *     Declare Variables used in the rsac1() subroutine
     *     Define variables used in the filtering routine
     */

    kname = strdup("filterc_in.sac");
    rsac1(kname, yarray, &nlen, &beg, &delta, &max, &nerr, SAC_STRING_LENGTH);
    delta_d = delta;
    if (nerr != 0) {
      fprintf(stderr, "Error reading in file: %s\n", kname);
      exit(-1);
    }
```

```
low    = 0.10;
high   = 1.00;
passes = 2;
order  = 4;
transition_bandwidth = 0.0;
attenuation = 0.0;

/*     Call xapiir ( Apply a IIR Filter )
 *        - yarray - Original Data
 *        - nlen   - Number of points in yarray
 *        - proto  - Prototype of Filter
 *                 - SAC_FILTER_BUTTERWORK        - Butterworth
 *                 - SAC_FILTER_BESSEL            - Bessel
 *                 - SAC_FILTER_CHEBYSHEV_TYPE_I  - Chebyshev Type I
 *                 - SAC_FILTER_CHEBYSHEV_TYPE_II - Chebyshev Type II
 *        - transition_bandwidth (Only for Chebyshev Filter)
 *                 - Bandwidth as a fraction of the lowpass prototype
 *                   cutoff frequency
 *        - attenuation (Only for Chebyshev Filter)
 *                 - Attenuation factor, equals amplitude reached at
 *                   stopband egde
 *        - order  - Number of poles or order of the analog prototype
 *                   4 - 5 should be ample
 *                   Cannot exceed 10
 *        - type   - Type of Filter
 *                 - SAC_FILTER_BANDPASS
 *                 - SAC_FILTER_BANDREJECT
 *                 - SAC_FILTER_LOWPASS
 *                 - SAC_FILTER_HIGHPASS
 *        - low    - Low Frequency Cutoff [ Hertz ]
 *                   Ignored on SAC_FILTER_LOWPASS
 *        - high   - High Frequency Cutoff [ Hertz ]
 *                   Ignored on SAC_FILTER_HIGHPASS
 *        - delta  - Sampling Interval [ seconds ]
 *        - passes - Number of passes
 *                 - 1 Forward filter only
 *                 - 2 Forward and reverse (i.e. zero-phase) filtering
 */
xapiir(yarray, nlen, SAC_BUTTERWORTH,
       transition_bandwidth, attenuation,
       order,
       SAC_BANDPASS,
       low, high,
       delta_d, passes);

/*     Do more processing ....  */
xarray[0] = 0;
kname = strdup("filterc_out.sac");
wsac0(kname, xarray, yarray, &nerr, SAC_STRING_LENGTH);
if (nerr != 0) {
  fprintf(stderr, "Error writing out file: %s\n", kname);
  exit(-1);
}
```

```
    return 0;
}
```

Envelope Function - Fortran

```
      program envelopef
      implicit none

      include "sacf.h"

!     Define the Maximum size of the data Array
      integer MAX
      parameter (MAX=1000)

!     Define the Data Array of size MAX
      real yarray, xarray, yenv
      dimension yarray(MAX), yenv(MAX)

!     Declare Variables used in the rsac1() subroutine
      real beg, delta
      integer nlen
      character*20 KNAME
      integer nerr

      kname = 'envelopef_in.sac'

      call rsac1(kname, yarray, nlen, beg, delta, MAX, nerr)

      if(nerr .NE. 0) then
          write(*,*)'Error reading in file: ',kname
          call exit(-1)
      endif

!     Call envelope ( Envelope Routine )
!        - nlen   - Number of points in yarray
!        - yarray - Original Data
!        - yenv   - Envelope of the Original Data
!
      call envelope(nlen, yarray, yenv)

!     Do more processing ....

      xarray = 0
      kname='envelopef_out.sac'
!     Write the SAC file kname
!       - kname holds the name of the file to be written
!       - xdata Input Time Data
!       - yfunc Input Amplitude Data
!       - nerr Error return Flag
      call wsac0(kname,xarray,yenv,nerr)
      if(nerr .NE. 0) then
          write(*,*)'Error writing out file: ',kname
```

```
            call exit(-1)
        endif

        call exit(0)

        end program envelopef
```

Envelope Function - C

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include "sac.h"
#include "sacio.h"

#define MAX 1000

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

    /* Local variables */
    int nlen, nerr, max;

    float beg, delta;
    char *kname;

    float yarray[1000];
    float xarray[1];
    float *yenv;

    max = MAX;

    /*     Define the Maximum size of the data Array
     *     Define the Data Array of size MAX
     *     Declare Variables used in the rsac1() subroutine
     *     Define variables used in the envelope routine
     */

    kname = strdup("envelopec_in.sac");
    rsac1(kname, yarray, &nlen, &beg, &delta, &max, &nerr, SAC_STRING_LENGTH);

    if (nerr != 0) {
      fprintf(stderr, "Error reading in file: %s\n", kname);
      exit(-1);
    }

    /* Allocate space for the envelope of yarray */
    yenv = (float *) malloc(sizeof(float) * nlen);
    if(yenv == NULL) {
      fprintf(stderr, "Error allocating memory for envelope\n");
      exit(-1);
    }
```

```
    /*     Call envelope ( Envelope Function )
     *        - nlen   - Number of points in yarray
     *        - yarray - Input array to take the envelope of
     *        - yenv   - Envelope of yarray
     */
    envelope(nlen, yarray, yenv);

    /*     Do more processing ....  */
    xarray[0] = 0;
    kname = strdup("envelopec_out.sac");
    wsac0(kname, xarray, yenv, &nerr, SAC_STRING_LENGTH);
    if (nerr != 0) {
      fprintf(stderr, "Error writing out file: %s\n", kname);
      exit(-1);
    }

    free(yenv);

    return 0;
}
```

Correlation - Fortran

```
      program envelopef
      implicit none

      include "sacf.h"

      integer i
!     Define the Maximum size of the data Array
      integer MAX
      parameter (MAX=4000)

!     Define the Data Array of size MAX
      real yarray1, yarray2, xarray, out, ytmp
      dimension yarray1(MAX), yarray2(MAX), out(MAX*4), ytmp(MAX*4)

!     Declare Variables used in the rsac1() subroutine
      real beg1, beg2, delta, endv
      integer nlen, nlen1, nlen2
      character*30 KNAME
      integer nerr
      real fval

!     Cross Correlation Variables
      integer nwin, wlen, nfft
      character *256 error
      real max_value, max_time

!     Read in the first data file
      kname = 'correlatef_in1.sac'
      call rsac1(kname, yarray1, nlen1, beg1, delta, MAX, nerr)
```

```
      if(nerr .NE. 0) then
          write(*,*)'Error reading in file: ',kname
          call exit(-1)
      endif

!     Read in the second data file
      kname = 'correlatef_in2.sac'
      call rsac1(kname, yarray2, nlen2, beg2, delta, MAX, nerr)

      if(nerr .NE. 0) then
          write(*,*)'Error reading in file: ',kname
          call exit(-1)
      endif

      nlen = nlen1
!
!     If the signals are not the same length, then find the longest
!     signal, make both signals that length by filling the remainder
!     with zeros (pad at the end) and then run them through crscor
!     This should be fixed in upcoming releases and the introduction
!     of a "correlate" function so you do not need to handle the
!     signal length, padding, and window lengths, ...
!

      nwin = 1
      wlen = nlen
      nfft = 0
!     Call crscor ( Cross Correlation )
!        - yarray1 - First  Input array to correlate
!        - yarray2 - Second Input array to correlate
!        - nlen    - Number of points in yarray and yarray2
!        - nwin    - Windows to use in the correlation
!        - wlen    - Length of the windows
!        - type    - Type of Window (SAC_RECTANGLE)
!        - out     - output sequence
!        - nfft    - Length of the output sequence
!        - error   - Error Message
!

      call crscor(yarray1, yarray2, nlen,
     &            nwin, wlen, SAC_RECTANGLE,
     &            ytmp, nfft, error)

      do i = 1,MAX*4
         out(i) = 0.0
      enddo
!
!     out[1 : nlen1 - 1 ] <-- ytmp[ nfft - nlen1 + 2 : nfft  ]
!     out[nlen1 : nlen1 + nlen2 - 1 ] <-- ytmp[ 1 : nlen2 ]
!
      do i = 1,nlen1-1
         out(i) = ytmp(nfft - nlen1 + i + 1)
      enddo
      do i = 1,nlen2
```

```
         out(nlen1 + i - 1) = ytmp(i)
      enddo


      nfft = nlen1 + nlen2 - 1
      xarray = 0
      beg1 = -delta * (nlen1 - 1.0) + (beg2 - beg1)
      endv = beg1 + delta * (nfft - 1)

      call setnhv('npts',   nfft,    nerr)
      call setfhv('delta',  delta,   nerr)
      call setlhv('leven',  .true.,  nerr)
      call setfhv('b',      beg1,    nerr)
      call setfhv('e',      endv,    nerr)
      call setihv('iftype', "itime", nerr)
      call setnhv('nzyear', SAC_NUMBER_UNDEFINED, nerr)
      call setnhv('nzhour', SAC_NUMBER_UNDEFINED, nerr)
!     Find the maximum value and time of the correlation function
      max_value = out(1)
      max_time  = 0
      do i = 2,nfft
         if(out(i) > max_value) then
            max_value = out(i)
!           Negative shifts are at the end of the correlation sequence
            if(i > nfft/2) then
               max_time = beg1 + (i - nfft) * delta
            else
               max_time = beg1 + i * delta
            endif
         endif
      enddo

!      call setfhv( 'user0', max_time,  nerr)
!      call setfhv( 'user1', max_value, nerr)


!     Write the SAC file
      kname='correlatef_out1.sac'
      call wsac0(kname, xarray, out, nerr)
      if(nerr .NE. 0) then
          write(*,*)'Error writing out file: ',kname
          call exit(-1)
      endif

      call exit(0)

      end program envelopef
```

Correlation - C

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

```
#include <sac.h>
#include <sacio.h>

#define MAX        4000
#define ERROR_MAX  256

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

    /* Local variables */
    int i;
    int nlen, nlen1, nlen2, nerr, max;

    float beg1, beg2, delta, end;
    char *kname;

    float yarray1[MAX], yarray2[MAX], ytmp[MAX*4], xarray[1];
    float *out;
    float max_value, max_time;

    int nwin, wlen, nfft, leven, when;

    char error[ERROR_MAX];

    max = MAX;

    /* Read in the first file  */
    kname = strdup("correlatec_in1.sac");
    rsac1(kname, yarray1, &nlen1, &beg1, &delta, &max, &nerr, SAC_STRING_LENGTH);

    if (nerr != 0) {
      fprintf(stderr, "Error reading in file(%d): %s\n", nerr, kname);
      exit(-1);
    }

    /* Read in the second file  */
    kname = strdup("correlatec_in2.sac");
    rsac1(kname, yarray2, &nlen2, &beg2, &delta, &max, &nerr, SAC_STRING_LENGTH);

    if (nerr != 0) {
      fprintf(stderr, "Error reading in file: %s\n", kname);
      exit(-1);
    }

    nlen = nlen1;
    /**
     *  If the signals are not the same length, then find the longest
     *  signal, make both signals that length by filling the remainder
     *  with zeros (pad at the end) and then run them through crscor
     *  This should be fixed in upcoming releases and the introduction
     *  of a "correlate" function so you do not need to handle the
     *  signal length, padding, and window lengths, ...
     *
```

```
 */

/* Allocate space for the correlation of yarray1 and yarray2 */
max = next2((2 * nlen) - 1) * 2;
out = (float *) malloc(sizeof(float) * max);
if(out == NULL) {
  fprintf(stderr, "Error allocating memory for correlation\n");
  exit(-1);
}

/* Set up values for the cross correlation */
nwin = 1;
wlen = nlen;
nfft = 0;

/*     Call crscor ( Cross Correlation )
 *        - yarray1 - First  Input array to correlate
 *        - yarray2 - Second Input array to correlate
 *        - nlen    - Number of points in yarray and yarray2
 *        - nwin    - Windows to use in the correlation
 *        - wlen    - Length of the windows
 *        - type    - Type of Window (SAC_RECTANGLE)
 *        - out     - output sequence
 *        - nfft    - Length of the output sequence
 *        - error   - Error Message
 *        - err_len - Length of Error Message (on input)
 */
crscor(yarray1, yarray2, nlen,
       nwin, wlen, SAC_RECTANGLE,
       ytmp, &nfft, error, ERROR_MAX);

for(i = 0; i < max; i++) {
  out[i] = 0;
}

/*
 *     out[0 : nlen1 - 2 ] <-- ytmp[ nfft - nlen1 + 1 : nfft -1 ]
 *     out[nlen1 - 1 : nlen1 + nlen2 - 2 ] <-- ytmp[ 0 : nlen2-1 ]
 */
for(i = 0; i <= nlen1 - 2; i++) {
  out[i] = ytmp[nfft - nlen1 + i + 1];
}
for(i = 0; i <= nlen2 - 1; i++) {
  out[nlen1 + i - 1] = ytmp[i];
}


nfft = nlen1 + nlen2 - 1;
xarray[0] = 0;
leven = TRUE;
beg1 = -delta * (nlen1 - 1) + (beg2 - beg1);
end = beg1 + delta * (nfft - 1);

setnhv ( "npts",   &nfft,   &nerr, SAC_STRING_LENGTH);
```

```
    setfhv ( "delta",  &delta,  &nerr, SAC_STRING_LENGTH);
    setlhv ( "leven",  &leven,  &nerr, SAC_STRING_LENGTH);
    setfhv ( "b",      &beg1,   &nerr, SAC_STRING_LENGTH);
    setfhv ( "e",      &end,    &nerr, SAC_STRING_LENGTH);
    setihv ( "iftype", "itime", &nerr, SAC_STRING_LENGTH, SAC_STRING_LENGTH);
    when = SAC_NUMBER_UNDEFINED;
    setnhv ( "nzyear", &when,   &nerr, SAC_STRING_LENGTH);
    setnhv ( "nzhour", &when,   &nerr, SAC_STRING_LENGTH);

    /* Find the maximum value and time of the correlation function */
    max_value = out[0];
    max_time  = 0;
    for(i = 1; i < nfft; i++) {
      if(out[i] > max_value) {
        max_value = out[i];
        /* Negative shifts are at the end of the correlation sequence */
        if(i > nfft/2) {
          max_time  = (i - nfft) * delta;
        } else {
          max_time  = i * delta;
        }
      }
    }

    /*
      setfhv( "user0", &max_time,  &nerr, SAC_STRING_LENGTH);
      setfhv( "user1", &max_value, &nerr, SAC_STRING_LENGTH);
    */

    /*   Write out the correlation function   */
    kname = strdup("correlatec_out1.sac");
    wsac0(kname, xarray, out, &nerr, SAC_STRING_LENGTH);
    if (nerr != 0) {
      fprintf(stderr, "Error writing out file: %s\n", kname);
      exit(-1);
    }

    free(out);

    return 0;
}
```

Convolution - Fortran

```
      program envelopef
      implicit none

      include "sacf.h"

      integer i,j
!     Define the Maximum size of the data Array
      integer MAX
      parameter (MAX=4000)
```

```
!     Define the Data Array of size MAX
      real yarray1, yarray2, ytmp, xarray, out
      dimension yarray1(MAX), yarray2(MAX), ytmp(MAX), out(MAX*4)

!     Declare Variables used in the rsac1() subroutine
      real beg, delta, endv
      integer nlen1, nlen2, nlen
      character*30 KNAME
      integer nerr

!     Cross Correlation Variables
      integer nwin, wlen, nfft
      character *256 error
      character *24  kevnm

!     Read in the first data file
      kname = 'convolvef_in1.sac'
      call rsac1(kname, ytmp, nlen1, beg, delta, MAX, nerr)

      if(nerr .NE. 0) then
          write(*,*)'Error reading in file: ',kname
          call exit(-1)
      endif

!     Read in the second data file
      kname = 'convolvef_in2.sac'
      call rsac1(kname, yarray2, nlen2, beg, delta, MAX, nerr)

      if(nerr .NE. 0) then
          write(*,*)'Error reading in file: ',kname
          call exit(-1)
      endif

!     Reverse the First Signal */
      j = 1
      do i = nlen1,1,-1
         yarray1(j) = ytmp(i)
         j = j + 1
      enddo

      nlen = nlen1
      if(nlen2 > nlen) then
         nlen = nlen2
      endif


      nwin = 1
      wlen = nlen
      nfft = 0
!     Call crscor ( Cross Correlation )
!        - yarray1 - First  Input array to correlate
!        - yarray2 - Second Input array to correlate
!        - nlen    - Number of points in yarray and yarray2
!        - nwin    - Windows to use in the correlation
```

```
!        - wlen    - Length of the windows
!        - type    - Type of Window (SAC_RECTANGLE)
!        - out     - output sequence
!        - nfft    - Length of the output sequence
!        - error   - Error Message
!

      call crscor(yarray1, yarray2, nlen,
     &            nwin, wlen, SAC_RECTANGLE,
     &            out, nfft, error)


!     Zero out the tmp signal
      do i = 1, MAX
         ytmp(i) = 0.0
      enddo

!     Reconstruct the signal from the "cross correlation" back to front
!
!     ytmp[1        : nlen1 - 1         ] <- out[nfft-nlen1+2 : nfft  ]
!     ytmp[nlen1    : nlen1 + nlen2 - 1 ] <- out[1            : nlen2 ]
!
!     nfft is the last point of the output sequence
!
      do i = 1,nlen1-1
         ytmp(i) = out(nfft - nlen1 + i + 1)
      enddo
      do i = 1, nlen2
         ytmp(nlen1 + i - 1) = out(i)
      enddo


      nfft = nlen1 + nlen2 - 1
      xarray = 0
      beg = 0
      endv = beg + delta * (nfft - 1)
      j = 1

      call newhdr()
      call setnhv('npts',   nfft,    nerr)
      call setfhv('delta',  delta,   nerr)
      call setlhv('leven',  .true.,  nerr)
      call setfhv('b',      beg,     nerr)
      call setfhv('e',      endv,    nerr)
      call setihv('iftype', 'itime', nerr)
      call setkhv('kstnm',  'sta',   nerr)
      call setkhv('kcmpnm', 'Q',     nerr)
      call setnhv('nwfid',  j, nerr)
      kevnm = 'FUNCGEN: TRIANGLE'
      call setkhv ('kevnm', kevnm, nerr)
!     Write the SAC file
      kname='convolvef_out1.sac'
      call wsac0(kname, xarray, ytmp, nerr)
      if(nerr .NE. 0) then
```

```
            write(*,*)'Error writing out file: ',kname,nerr
            call exit(-1)
        endif

        call exit(0)

        end program envelopef
```

Convolution - C

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include <sac.h>
#include <sacio.h>

#define MAX        4000
#define ERROR_MAX  256

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

    /* Local variables */
    int i, j;
    int nlen, nlen1, nlen2, nerr, max;

    float beg, delta, end;
    char *kname;

    float yarray1[MAX], yarray2[MAX], ytmp[MAX], xarray[1];
    float *out;

    int nwin, wlen, nfft, leven;

    char error[ERROR_MAX];

    max = MAX;

    for(i = 0; i < MAX; i++) {
      yarray1[i] = 0.0;
      yarray2[i] = 0.0;
      ytmp[i] = 0.0;
    }
    /* Read in the first file  */
    kname = strdup("convolvec_in1.sac");
    rsac1(kname, ytmp, &nlen1, &beg, &delta, &max, &nerr, SAC_STRING_LENGTH);

    if (nerr != 0) {
      fprintf(stderr, "Error reading in file(%d): %s\n", nerr, kname);
      exit(-1);
    }
```

```
/* Read in the second file  */
kname = strdup("convolvec_in2.sac");
rsac1(kname, yarray2, &nlen2, &beg, &delta, &max, &nerr, SAC_STRING_LENGTH);

if (nerr != 0) {
  fprintf(stderr, "Error reading in file: %s\n", kname);
  exit(-1);
}

/* Reverse the First Signal */
j = 0;
for(i = nlen1 - 1; i >= 0; i--) {
  yarray1[j] = ytmp[i];
  j++;
}

nlen = nlen1;
if(nlen2 > nlen) {
  nlen = nlen2;
}
/* Allocate space for the correlation of yarray1 and yarray2 */
max = next2((2 * nlen) - 1) * 2;
out = (float *) malloc(sizeof(float) * max);
if(out == NULL) {
  fprintf(stderr, "Error allocating memory for correlation\n");
  exit(-1);
}

/* Set up values for the cross correlation */
nwin = 1;
wlen = nlen;
nfft = 0;

/*     Call crscor ( Cross Correlation, no, wait, uh Convolution )
 *        - yarray1 - First  Input array to correlate
 *        - yarray2 - Second Input array to correlate
 *        - nlen    - Number of points in yarray and yarray2
 *        - nwin    - Windows to use in the correlation
 *        - wlen    - Length of the windows
 *        - type    - Type of Window (SAC_RECTANGLE)
 *        - out     - output sequence
 *        - nfft    - Length of the output sequence
 *        - error   - Error Message
 *        - err_len - Length of Error Message (on input)
 */
crscor(yarray1, yarray2, nlen,
       nwin, wlen, SAC_RECTANGLE,
       out, &nfft, error, ERROR_MAX);

/* Zero out the tmp signal */
for(i = 0; i < MAX; i++) {
  ytmp[i] = 0.0;
}
```

```
    /* Reconstruct the signal from the "cross correlation" back to front
     *
     *  ytmp[0         : nlen1 - 2         ] <- out[nfft-nlen1+1 : nfft  - 1 ]
     *  ytmp[nlen1 - 1 : nlen1 + nlen2  -2 ] <- out[0            : nlen2 - 1 ]
     *
     *  nfft-1 is the last point of the output sequence
     */
    for(i = 0; i <= nlen1 - 2; i++) {
      ytmp[i] = out[nfft - nlen1 + i + 1];
    }
    for(i = 0; i <= nlen2 - 1; i++) {
      ytmp[nlen1 + i - 1] = out[i];
    }

    nfft = nlen1 + nlen2 - 1;
    xarray[0] = 0;
    leven = TRUE;
    beg = 0;
    end = beg + delta * (nfft - 1);
    newhdr();
    j = 1;
    setnhv ( "npts",   &nfft,    &nerr, SAC_STRING_LENGTH);
    setfhv ( "delta",  &delta,   &nerr, SAC_STRING_LENGTH);
    setlhv ( "leven",  &leven,   &nerr, SAC_STRING_LENGTH);
    setfhv ( "b",      &beg,     &nerr, SAC_STRING_LENGTH);
    setfhv ( "e",      &end,     &nerr, SAC_STRING_LENGTH);
    setihv ( "iftype", "itime",  &nerr, SAC_STRING_LENGTH, SAC_STRING_LENGTH);
    setkhv ( "kcmpnm", "Q",      &nerr, SAC_STRING_LENGTH, SAC_STRING_LENGTH);
    setkhv ( "kstnm",  "sta",    &nerr, SAC_STRING_LENGTH, SAC_STRING_LENGTH);
    setnhv ( "nwfid",  &j,       &nerr, SAC_STRING_LENGTH);
    setkhv ( "kevnm",  "FUNCGEN: TRIANGLE", &nerr, SAC_STRING_LENGTH, SAC_STRING_LENGTI

    /*   Write out the correlation function   */
    kname = strdup("convolvec_out1.sac");
    wsac0(kname, xarray, ytmp, &nerr, SAC_STRING_LENGTH);
    if (nerr != 0) {
      fprintf(stderr, "Error writing out file: %s\n", kname);
      exit(-1);
    }

    free(out);

    return 0;
}
```

Blackboard Variables in SAC

The blackboard is a feature that can be used to temporarily store and retrieve information while inside SAC. Blackboard variables can also be saved in a disk file using the WRITEBBF command and later restored into SAC using the READBBF command. There are four functions in the sacio library which allow the user to read and write blackboard variables in home grown software. This library is available in the lib directory of the SAC distribution for all platforms.

A blackboard entry consists of a name and a value. Blackboard entries are created using the SETBB and EVALUATE commands. The value of a blackboard variable can be obtained using the GETBB command. You can also substitute the value of a blackboard variable directly in other commands by preceeding its name with a percent sign ("%") as shown below:

```
SAC> SETBB C1 2.45
SAC> SETBB C2 4.94
SAC> BANDPASS CORNERS %C1 %C2
```

Prior to v101.6, Blackboard number variables were stored as strings, now they are stored as double-precision variables.

Now lets see how blackboard variables can be used in macros. In the following example, the first value is a variable, and the other values are calculated from the first:

```
$KEYS FILES VALUE1
$DEFAULT VALUE1 4
READ $FILES
EVALUATE TO VALUE2 $VALUE1 * 2
EVALUATE TO VALUE3 %VALUE2 + 1
MUL $VALUE1 %VALUE2 %VALUE3
FFT
BG SGF
PSP AM
```

You can append or prepend any text string to a blackboard variable. To prepend simply concatenate the text string with the variable. To append you must repeat the delimiter % after the variable and before the text string.

Examples

Assume that the blackboard variable TEMP has the value "ABC". Then value of "XYZ%TEMP" would be "XYZABC" and the value of "%TEMP%XYZ" would be "ABCXYZ":

```
SAC> fg
SAC> echo on
SAC> setbb TEMP "ABC"
 setbb TEMP "ABC"
SAC> ch kname ABC%TEMP
 ch kname XYZ%TEMP
 ==> ch kname XYZABC
 ch kevnm %TEMP%XYZ
 ch kevnm %TEMP%XYZ
  ==> ch kevnm ABCXYZ
```

More information on the use of blackboard variables in SAC macros is given in the section on SAC macros.

Blackboard I/O in SAC

There are four SAC commands which are used to read and write blackboard variables and to set and get blackboard variable values. These are READBBF, WRITEBBF, GETBB, and SETBB. These are SAC commands which can be called at the SAC prompt or within a SAC macro.

Blackboard I/O in Your Own C or FORTRAN Programs

The sacio library , which is included in the SAC distribution, contains four blackboard I/O routines which you can call from C or FORTRAN programs. These routines: read the blackboard variable files (READBBF), write blackboard variable files (WRITEBBF), get the current values of blackboard variables (GETBBV), and set new values of blackboard variables (SETBBV).

readbbf -- Read a Blackboard File

```
void readbbf(char *name, int *nerr, int kname_s)
```

Arguments

kname: File to be read

nerr: Error return Flag,

- 0 on Success
- Non-Zero on Error

kname_s: Length of character array p name

writebbf Write a Blackboard File:

```
void writebbf(char *name, int *nerr, int kname_s)
```

Arguments

kname: File to be written

nerr: Error return Flag

- 0 on Success
- Non-Zero on Error

kname_s: Length of character array p name

getbbv Get a Variable:

```
void getbbv(char *kname, char *kvalue, int *nerr,
            int kname_s, int kvalue_s)
```

Arguments:

kname: Input Variable Name

kvalue: Output Variable Value

nerr: Error return Flag,

- 0 on Success
- Non-Zero on Error

kname_s: Length of character array p name

kvalue_s: Length of character array p value

setbbf Set a Variable:

```
void setbbv(char *kname, char *kvalue, int *nerr,
            int kname_s, int kvalue_s)
```

Arguments:

kname: Input Variable Name

kvalue: Input Variable Value
nerr: Error return Flag,

- 0 on Success
- Non-Zero on Error

kname_s: Length of character array p name
kvalue_s: Length of character array p value

Fortran Example

The following is a short FORTRAN program that reads in a blackboard variable file gets the values of a few variables, sets the value of a new one, and then writes the file back to disk:

```
program bbv
implicit none
character(len=10) kname, kvalue
integer nerr

! Read in the Blackboard Variable File
kname = 'bbf       '
call readbbf(kname, nerr)
if(nerr .NE. 0) then
   write(*,*)'Error reading blackboard variable file'
   call exit(-1);
endif
call test("error reading blackboard file: bbf", (nerr .eq. 0))

! Set a New Variable on the Blackboard
kname  = 'newvar    '
kvalue = '1         '
call setbbv(kname, kvalue, nerr)
if(nerr .NE. 0) then
   write(*,*)'Error setting blackboard variable'
   call exit(-1);
endif

! Get a Variable already on the Blackboard
kname = 'newvar    '
call getbbv(kname, kvalue, nerr)
if(nerr .NE. 0) then
   write(*,*)'Error getting blackboard variable'
   call exit(-1);
endif

! Get a Variable already on the Blackboard
kname = 'somevar    '
call getbbv(kname, kvalue, nerr)
if(nerr .NE. 0) then
   write(*,*)'Error getting blackboard variable'
   call exit(-1);
endif

! Write out the new set of Blackboard Variables
```

```
kname = 'bbfout    '
call writebbf(kname, nerr)
if(nerr .NE. 0) then
   write(*,*)'Error writing blackboard variable file'
   call exit(-1);
endif

return
end
```

Case Insensitive Variable Names

The names of blackboard variables are converted to uppercase before being stored or retrieved. This means that you can use either uppercase or lowercase in your program. However, the name of the blackboard variable file must be given exactly as it appears on disk. No case conversion is done on file names.

To compile your code with the above blackboard variable routine the sacio library must be linked in at compile time. This can be accomplished with a command similar to the one below. This exact command will depend on your Fortran compilier, here we are using f77 and assuming SAC is installed in the default location of /usr/local/sac and the sacio.a library is at /usr/local/sac/lib/sacio.a

```
f77 -o my_blackboard_program my_blackboard_program.f /usr/local/sac/lib/sacio.a
```

C Example

Below is a C program which performs the same functions as the FORTRAN program above. It can be compiled in a similar manner as the Fortran examples:

```
#include <stdio.h>
#include <stdlib.h>

int
main(int argc, char* argv[]) {
  int kname_s, kvalue_s, nerr;
  char *kname, *kvalue;

  char kvalue2[128];

  readbbf("bbf", &nerr, -1);
  if(nerr != 0) {
    fprintf(stderr, "Error reading in blackboard variable file\n");
    exit(-1);
  }

  kname = strdup("newvar");
  kname_s = strlen(kname);

  kvalue = strdup("1  ");
  kvalue_s = strlen(kvalue);

  sprintf(kvalue2, "%s", "1");
  setbbv(kname, kvalue2, &nerr, -1, -1);
  if(nerr != 0) {
```

```
    fprintf(stderr, "Error setting blackboard variable\n");
    exit(-1);
  }

  kvalue = (char *) malloc(sizeof(char) * 128);
  memset(kvalue, 0, 128);

  kname = strdup("newvar");
  kname_s = strlen(kname);

  getbbv(kname, kvalue2, &nerr, -1, 128);
  if(nerr != 0) {
    fprintf(stderr, "Error getting blackboard variable\n");
    exit(-1);
  }

  kvalue = (char *) malloc(sizeof(char) * 128);
  memset(kvalue, 0, 128);

  kname = strdup("somevar");
  kname_s = strlen(kname);

  getbbv(kname, kvalue2, &nerr, -1, sizeof(kvalue2));
  if(nerr != 0) {
    fprintf(stderr, "Error getting blackboard variable\n");
    exit(-1);
  }

  kname = strdup("bbfout");
  kname_s = strlen(kname);

  writebbf(kname, &nerr, kname_s);
  if(nerr != 0) {
    fprintf(stderr, "Error writing blackboard variable file\n");
    exit(-1);
  }

  return 0;
}
```

Notice that in C, more parameters are required in the function calls than in FORTRAN. This is because unlike C, FORTRAN implicitly passes string length specifiers for each string in the parameter list. These specifiers are at the end of the parameter list, and are declared as INTEGER*4 or long int. Notice also that the values passed as string length specifiers do not include the null termintor '0'.:

```
gcc -o my_blackboard_program my_blackboard_program.c /usr/local/sac/lib/sacio.a
```

Graphics in SAC

Overview

This section describes the graphics devices that are currently supported and then briefly describes the commands in each of the graphics functional modules.

Graphics Devices

There are two graphics "devices" currently supported.

- **XWINDOWS is a general windowing system running on most** high-resolution, bit-mapped graphics workstations.
- **SGF is a general purpose device driver representing a** large class of actual physical devices.

Each device is described in more detail below.

XWINDOWS (or X for short) is a windowing scheme developed under the industry-financed Athena project at MIT. X employs what is called a network model, where a single process or server controls the screen display. Other programs send requests to this server when they want to modify part of the screen. X is widely used on the graphics workstation and offers one of the best frameworks for developing portable window-based applications. (A problem with backward compatibility on many platforms is that the location of the X11 libraries may change.)

Beginning with v101.5, after an image has been displayed on the terminal using X11, command SAVEIMG can be used to create a high-definition Postscript or PDF file of the displayed image. See SAVEIMG for details.

SGF stands for SAC Graphics File. A SAC Graphics File contains all the information needed to generate a single plot on any graphics device. (Using the current computer jargon, these are called graphics "metafiles.") Each plot is stored in a separate file. The file names are of the form "Fnnn.SGF" where "nnn" is the plot number, beginning with "001". You can control some features of this file name using the SGF command. The program SGFTOPS can convert a SGF file to postscript, and scrpts are provided to print the files and/or convert them to PDF format. See sac/utils/README_utils for details.

Graphics Control Module

These commands control device selection and certain aspects of the display.

BEGINDEVICES: selects one or more graphics devices for plotting and

ENDDEVICES: deselects plotting to those devices.

ERASE: erases the graphics display area,

VSPACE: controls the maximum size and shape of plots, and

SGF: controls certain options for the SAC Graphics File device.

Graphics Action Module

The commands in this module are mostly action-producing ones that create plots in various formats.

PLOT: plots each signal in memory on a separate plot.

PLOT1: plots a set of signals on a single plot with a common x axis and separate y axes.

PLOT2: plots a set of signals on a single plot with common x and y axes (i.e. an overlay plot).

PLOTPK: produces a plot for the picking of arrival times, seismic phases, coda, etc. The format is similiar to that of PLOT1. A cursor is used to do the picking. The picks go into the header and can also be written into a HYPO pick file (OHPF) or an alphanumeric pick file (OAPF).

PLOTPM: generates a "particle-motion" plot on pairs of signals.

FILEID: controls the display of a file identification and

FILENUMBER: controls the display of file numbers on the sides of plots.

PICKS: controls the display of time picks on these plots.

SETDEVICE: lets you select a default graphics device to be used when plotting.

PLOTC: notates SAC plots and creates figures using cursor.

PLOTALPHA: reads alphanumeric data files on disk into memory and plots the data to the current output device.

PLOTDY: creates a plot with error bars.

PLOTXY: plots one or more data files versus another data file.

PRINT: prints most recent .sgf file in memory.

SAVEIMG: saves displlayed image in one of several formats.

Graphics Environment Module

The commands in this module are mostly parameter-setting ones that control various parts of the plots produced by the Graphics Action Module.

XLIM: control the plot limits for the y axes.

YLIM: control the plot limits for the x axes

XVPORT: control the location of the plot within the plotting area

YVPORT: control the location of the plot within the plotting area

TITLE: specify a title (TITLE)

XLABEL: x axes labels

YLABEL: y axes labels

PLABEL: set of general plot labels

There are several commands that control the displaying of the data itself:

LINE: controls linestyle selection and fill options

SYMBOL: controls symbol plotting, and

COLOR: controls color selection.

GTEXT: controls the quality and font of text used in plots and

TSIZE: controls the text size attributes. If you are using a multi-windowing workstation, you can use the WINDOW command to set the location and shape of the graphics windows and the

BEGINWINDOW: command to select a specific graphics window for plotting.

BEGINFRAME: turns off automatic new frame actions between plots and

ENDFRAME: resumes automatic new frame actions. Combined with other graphics commands (especially XVPORT and YVPORT), these two commands can be used to create fairly complicated plots.

XLIN: set the x axis to linear scaling

XLOG: set the x axis to logarithmic scaling

YLIN: set the y axis to linear scaling

YLOG: set the y axis to logarithmic scaling

LINLIN: to set the scaling for both axes, x-linear, y-linear

LINLOG: to set the scaling for both axes, x-linear, y-log

LOGLIN: to set the scaling for both axes, x-log, y-linear

LOGLOG: to set the scaling for both axes, x-log, y-log

XDIV: control the spacing between labeled divisions

YDIV: control the spacing between labeled divisions

XFUDG_E: change the "fudge factors" on the x axis

YFUDGE: change the "fudge factors" on the y axis

AXES: control the location of labeled axes

TICKS: control the location of tick marks.

GRID: control the plotting of grid lines

BORDER: control the plotting of a surrounding border.

XGRID: that let you independently control gridding on the x axis

YGRID: that let you independently control gridding on the y axis

QDP: allows one to speed up plotting by NOT plotting each data point.

There are several commands which control the display of logarithmic axes:

XFULL: control the plotting of full logarithmic decades,

YFULL: control the plotting of full logarithmic decades,

LOGLAB: controls the plotting of secondary labels, and

FLOOR: puts a minimum value on logarthimically scaled data.

LOADCTABLE: allows the user to select a new color table for use in image plots.

WAIT: tells SAC whether or not to pause between plots.

WIDTH: controls line-width selection for graphics devices.

NULL: controls the plotting of null values.

SAC Graphics File

Overview

Each SAC Graphics File (SGF) contains all the information needed to describe a single picture, called a frame. Prior to the mid90s when the C-based SAC2000 replaced the Fortran-based SAC, several utilities programs, external to SAC, were written to perform various function on an SGF file: a program to merge up to several SGF files, a program that directly displayed the SGF plots on the console, a program that listed the commands encoded in an SGF file, ond sgftops, which converted an SGF file to a postscript file. Most of these programs were never converted to C, and currently, sgftops is the only program that is maintained. A description of program sgftops is given at the end of this section.

SGF Format

Overview

Each SGF contains all the information needed to describe a single picture (called a frame.) The filenames are normally of the form "f nnn.sgf" where nnn is the three digit frame number. A translation program must be written to convert these files to the format needed for any specific graphics device.

Physical Format

A SGF contains variable length records with a maximum record size of 2500 32-bit words. The first 32-bit word of each record contains the length of that record, including this word count. They are written in binary format for faster i/o. To keep them small and portable between different computer systems, all commands and data are stored in 16-bit integer format, 2 bytes.

Command

The draw command (draw a line from the previous location to the new location) is the most common command. This command is simply a pair of integers giving the new x and y locations. These integers are in the range 0 to 32000 in the x direction and 0 to 24000 in the y direction. (This produces an aspect ratio of 3:4 which maps well to most output devices.)

Other Commands

The rest of the commands (with one exception) consist of a command identification number, a data count, and zero or more data words. The identification number is a negative integer and tells the translation program what operation is to be performed. The use of negative integers makes it easy to distinguish these commands from the draw commands. The data count is the number of 16-bit data words, 2 bytes, contained in this command. This format allows for the future addition of new commands. Also it allows each translation program to quickly skip over commands that it cannot process. The one exception to this format is the null or no-op command. This has an identification number of -1 and contains no data count and no data words. It is used to fill out a record to an even number of 32-bit words. The table on the next page summarizes the current commands. A plot produced from a simple SGF is also included, along with a table describing the contents of that simple SGF.

SGF Commands Table

ID	Count	Command	Description
-1	0	No-op	No Operation
-2	0	End	End of picture.
-3	2	Move	Move to the location contained in the two (x,y) data words.
-4	1	Color	Change color to value contained in data word.
-5	•	Text	Write hardware text at current location. Data count contains number of 16-bit words of text plus one. First data word is the number of characters in the text. Rest of data words contain the text, two characters per word. Last byte of last word is not significant if character count is odd.
-6	2	Text Size	Change hardware text size. Data words contain the text width and height as integer fractions of the maximum coordinate system size (32000). For example a value of 320 would set text size to 0.01 or one percent of the full plot size.
-7	1	Line Style	Change linestyle to value contained in data word.
-8	1	Plot Size	Change the physical size of the plot. Data word is the desired length in the x direction in 0.001 inch increments. Default value is 10000 which is equivalent to 10.0 inches. None of the SGF conversion programs currently make use of this option.
-9	1	Line Width	Change the linewidth to value contained in data word.
-10	3	Polygon Fill	Fill a polygon with a gray value in the first data word by moving to data words 2 and 3.
-11	5	Plot Filled Rectangle	Fill a rectangle defined at data word 1 and 2 (x, y) with a width and height data words 3 and 4 and a color at data word 5.
-12	1	Text Angle	Set the Angle of the Text
-13	4	Color Image	Plot a color image of width data word 1 and height data word 2 at data word 3 and 4 (x, y).
-14	1	color Fill flag	Fill polygon with current color. Can be used to fill positive and negative parts of waveform with different colors

PROGRAM SGFTOPS

Plots from SAC can be saved to a file as a Sac Graphics Format (SGF) file. Program sgfops converts a binary .sgf file to a postscript file. As of version 101.4, sgftops can handle .sgf files with eithr endian.

The source code can be found in sac/utils/sgftops.c, and it is built and put in sac/bin at the time SAC is built and installed.

Entering sgftops with no arguments produces:

Usage: sgftops sgf_file ps_file [line_width scale_id]

where:

line_width = 1, 1.5, 2, 3, etc.
scale_id = i (landscape mode plus id); scale_id = s (shift,rotate & scale); scale_id = si (s plus id).

time/date in id is file creation date for the .sgf file.

Example: sgftops foo.sgf foo.ps 2 si

Produces a plot with line thick=2 and ID at the bottom. Prompts further for translation, rotation and scale.

The origin of plot is lower left corner of portrait mode and angle is Counter Clockwise (CCW)

The reason that sgftops default is landscape is that when it was first written (20+ years ago), the major use of a postscript file was to be ported to a postscript printr.

PLOT-CONVERSION SCRIPTS

There are two scripts in sac/bin that call sgftops and produce a screen display (sgftox.csh) or an EPS file (sgftoeps.csh). Enter the script names with no arguments to get further information. Both, as written, require that the program "gs" is in your path. Script sgftox.csh uses gs to display the image on the screen, but it is easily modified to use other available postscript file viewers, such as gv, ggv, or evince. Script sgftoeps.csh can be modified to produce a PDF file if script epstopdf is on the system and in the path. Note that the output file for both sgftoeps.csh and sgftox.csh is in portrait format.

Calling SAC from Scripts

Overview

The SAC program can be run from a variety of scripting languages and shells. Provided below are a few examples of using SAC within the bounds of either a shell or high-level scripting language. Included are examples for sh, csh, perl and python. Other languages very likely have a similar format. Terminator string EOF is required to start at the first character of a line or the scripts will not work.

Setting the environment variable SAC_DISPLAY_COPYRIGHT to 0 will force SAC not to display the copyright header information. In the sh shell the option is:

```
export SAC_DISPLAY_COPYRIGHT=0
```

and in csh the syntax is:

```
setenv SAC_DISPLAY_COPYRIGHT 0
```

Simple Examples

sh:

```
#!/bin/sh

sac <<EOF
fg seismo
lh columns 2
quit
EOF
```

csh:

```
#!/bin/csh

sac <<EOF
fg seismo
lh columns 2
quit
EOF
```

perl:

```
#!/usr/bin/env perl

open(SAC, "| sac ") or die "Error opening sac";
print SAC "fg seismo\n";
print SAC "lh columns 2\n";
print SAC "quit\n";
close(SAC);
```

python:

```
#!/usr/bin/env python

import subprocess

p = subprocess.Popen(['sac'],
                     stdout = subprocess.PIPE,
                     stdin  = subprocess.PIPE,
                     stderr = subprocess.STDOUT )

out = p.communicate('''
fg seismo
lh columns 2
quit
''')

print out[0]
```

Extended Examples

The following examples take a set of SAC files in the current directory and low pass filter then at 1.0 Hz with a 2 pass, 4th order filter. The shell examples, sh and csh, require an invocation of sac for each file, but the scripting languages, perl and python, do not. The python and perl scripts use more complex and more powerful string handling than do the shell scripts.

sh:

```
#!/bin/sh

for file in *.SAC; do
   sac <<EOF
   echo on
   read $file
   rmean
   rtrend
   lp co 0.1 p 2 n 4
   write ${file}.filtered
   quit
EOF
done
```

csh:

```
#!/bin/csh

foreach file ( *SAC )
   sac <<EOF
   echo on
   read $file
   rmean
   rtrend
   lp co 0.1 p 2 n 4
   write ${file}.filtered
   quit
EOF
```

```
end
```

perl:

```
#!/usr/bin/env perl

open(SAC, "| sac ") or die "Error opening sac";
foreach $file ( glob("*.SAC") ) {
   print SAC qq[
     read $file
     rmean
     rtrend\
     lp co 0.1 p 2 n 4
     write ${file}.filtered
   ];
}
print SAC "quit\n";
close(SAC);
```

python:

```
#!/usr/bin/env python

import subprocess
import glob

p = subprocess.Popen(['sac'],
                     stdout = subprocess.PIPE,
                     stdin  = subprocess.PIPE,
                     stderr = subprocess.STDOUT )

s = "echo on\n"
for filename in glob.glob("*.SAC"):
    s += '''
       read %(file)s
       rmean
       rtrend
       lp co 0.1 p 2 n 4
       write %(file)s.filtered
    ''' % ( {'file': filename } )
s += "quit\n"
out = p.communicate( s )
print out[0]
```

SAC Output Messages

TABLE

Number	Translation
0002	Converting ascii to float - possible bad format.
0000	FILE I/O SERVICE LEVEL
0100	Operating system error
0101	opening file
0102	creating file
0103	for new file
0104	closing file
0105	destroying file
0106	Formatting error encountered while reading file
0107	File unit is in use:
0108	File does not exist:
0109	File already exists:
0110	Illegal file unit number:
0111	File unit is not in use:
0112	truncating file
0113	Illegal file type for file
0114	reading file
0115	writing file
0116	No available file units.
0117	Illegal hardcopy device:
0118	Can't send to
0119	checking existence of file
0120	No wfdisc file specified
0121	Error encoding XDR output file
0122	Partial updates not allowed for XDR file.
0123	Error decoding XDR input file
0124	Can't change to that directory. Check your permissions.
0125	csspickprefs not formatted properly.
0126	.wfdisc filenames require an explicit '.' on the command line:
0127	No data file specified.
0128	CSS file not version 3.0:
0129	Cannot form a path to that file
0130	XDR and ALPHA options are incompatible
0131	sac/datagen data directory not found.
0200	GRAPHICS SERVICE LEVEL
0201	Illegal graphics device
0202	Current graphics device does not have cursor capability.
0203	Can't create X window. Check DISPLAY environmental.
0300	ARRAY MANAGER FUNCTION
0301	Out of memory.

... continued on next page

Number	Translation
0302	Memory manager links clobbered for block starting at:
0400	ENLARGE DECOMPRESSION
0401	Enlarge: input record too small
0402	Enlarge: input record too large
0403	Enlarge: record has too many samples
0404	Enlarge: nmap
0405	Enlarge: ndiffs
0406	Enlarge: unexpected end-of-file
0407	Enlarge: nx, nrecl disagree
0408	Enlarge: too few samples found
0409	Enlarge: Number given in record header:
0410	Enlarge: Number of samples decompressed:
0411	Enlarge: inconsistent last values
0412	Enlarge: Last value of original data:
0413	Enlarge: Last value from decompression:
0800	USER SERVICE LEVEL
0801	File is not evenly spaced:
0802	File is not unevenly spaced:
0803	Data truncated to fit in user space for file
0000	GENERAL SERVICE LEVEL
0901	SAC programming logic error
0902	Can't take logarithm of a non-positive number.
0903	Please answer with a YES or NO.
0904	DISTAZ calcuation failed internal check for entry
0905	Time field must be at least 12 characters long.
0906	Date field must be at least 18 characters long.
0907	Bad time field entry detected:
0908	Bad date field entry detected:
0909	Bad julian date field entry detected:
0910	Maximum array that can be sorted is
0912	(A,I6,I4,I3,I3,I3,I4)
0913	Interrupt received.
0914	Illegal base name:
0915	Illegal base numbers:
0916	File name too long:
0917	Size of passed array(s) too small.
0918	Can't read or write into the global variable file.
0919	SAC data array is too small to execute this command.
0920	Character list delimiter found in character entry:
0921	Not enough room in character list for character entry:
0922	Text would exceed the maximum available space:
0923	Expected to find option in range t0 - t9; none found.
1000	COMMAND MODULE

... continued on next page

Number	Translation
1001	Bad command syntax at symbol
1002	Bad value for
1003	Value out of allowed range at symbol
1004	Illegal command.
1005	Illegal subprocess command.
1006	Length of string variable exceeded at symbol
1007	Not enough room for command file
1008	No command file name given.
1009	Too many command file arguments at
1010	Wrong number of command file arguments
1011	Bad command file syntax.
1012	Following option is not currently available:
1013	Obsolete command. Please use
1014	Undefined variable in command:
1015	Too many levels of nesting to execute macro
1016	Terminating execution of macro
1017	Illegal macro command:
1018	Exceeded maximum number of nested inline functions:
1019	Incorrect nesting of inline functions:
1020	Invalid inline function name:
1021	Correct number of arguments for this inline function call is
1022	Illegal arithmetic operation in inline function:
1023	All arguments to this inline function must be numeric.
1024	This argument in inline function should be an operator:
1025	This argument in inline function should be numeric:
1026	Maximum number of arguments for this inline function call is
1027	Exceed maximum number of external commands =
1028	External command does not exist:
1029	Command line too long.
1030	There is no year 0,
1100	EXECUTIVE MODULE
1101	Will terminate production run.
1102	Remainder of command file not executed.
1103	No help package is available.
1104	No help information is available for
1105	Error reading help information for
1106	Not a valid SAC command.
1107	Invalid entry in sitechan file
1108	PLOTTING FUNCTION
1109	UNUSED
1110	No news is good news.
1111	Error executing system command, insufficient memory.
1112	Error finding program

... continued on next page

Number	Translation
1113	Error starting program
1114	Error ending program
1115	This option is not currently implemented:
1116	This function is not available on the
1117	Can't evaluate expression because of bad operand value:
1118	Maximum number of open transcript files is
1119	Maximum number of active traceable variables is
1200	VARS FUNCTION
1201	Could not find VARS variable
1202	Maximum number of vars sections exceeded:
1203	Could not find VARS section
1204	Incorrect data type for VARS variable
1205	Could not delete VARS variable
1206	VARS option not currently implemented:
1207	Bad data block flag for VARS variable
1208	Disk file is not in VARS format:
1209	No current vars section has been defined.
1210	Bad input to subroutine
1211	VARS list already exists:
1234	RMS Noise greater than RMS Signal, setting to 0.0
1300	DATA FILE MODULE
1301	No data files read in.
1302	Maximum memory size exceeded.
1303	Overwrite flag is not on for file
1304	Illegal operation on data file
1305	Illegal operation on time series file
1306	Illegal operation on unevenly spaced file
1307	Illegal operation on spectral file
1308	Maximum smoothing half width is
1309	Maximum special header list length is
1310	Illegal data file list number
1311	No list of filenames to write.
1312	Bad number of files in write file list:
1313	Illegal relative time pick
1314	Data file list can't begin with a number.
1315	Maximum number of files in data file list is
1316	Can't smooth an unevenly spaced data file.
1317	The following file is not a SAC data file:
1318	Header in disk file is out of date:
1319	Bad data found in card image data file header.
1320	Available memory too small to read file
1321	Can't cut spectral file
1322	Undefined starting cut for file

... continued on next page

Number	Translation
1323	Undefined stop cut for file
1324	Start cut less than file begin for file
1325	Stop cut greater than file end for file
1326	Start cut greater than file end for file
1327	Stop cut less than file begin for file
1328	Start cut greater than stop cut for file
1329	Corrected by filling with zeros.
1330	Corrected by using file begin.
1331	Corrected by using file end.
1332	Fatal error condition.
1333	Unable to read some files
1334	Can't read or write DS2 formatted data files
1335	Illegal operation---only data file headers in memory.
1336	Undefined header field value.
1337	Illegal header field name.
1338	Too many data points to perform operation for file
1339	Too few data points to perform operation for file
1340	data points outside allowed range contained in file
1341	Can't write headers because CUT is ON.
1342	Illegal number of files in data file list:
1343	Formatting error while reading file
1344	Problem writing GSE file
1350	Could not find requested header entry
1351	Not enough room in header for new header entry
1352	Can not delete header entry
1353	Output variable too short for header entry
1354	No end-of-header found.
1355	Incorrect data type for header entry
1356	Can't cut unevenly spaced data file
1357	Decoding formatted alphanumeric data card.
1358	Maximum number of free format entries exceeded:
1359	Maximum number of alphanumeric data channels exceeded:
1360	Illegal character in alphanumeric content descriptor:
1361	Can only have one X channel per file.
1362	Must have at least one Y channel per file.
1363	Illegal data file list name:
1364	No data file list specifier (name or number) given.
1365	Illegal enumerated header field value:
1366	This command requires that data in memory be of type XYZ:
1377	Unable to adjust the time in the SDD header
1378	Illegal operation on XYZ data
1379	No SORT parameters given
1380	Too many SORT parameters:

... continued on next page

Number	Translation
1381	Not a valid SORT parameter:
1382	ALL and COMMIT options both set, ignoring COMMIT option.
1383	SORT failed
1384	ASCEND and DESCEND options go after the related header in command line
1385	No worksets in memory.
1386	Could not get Workset name.
1387	No file name specified.
1388	Reference time not equal to zero: Reference time is
1389	NVHDR, NPTS, NWFID, NORID, and NEVID cannot be changed with CHNHDR
1390	KSTNM and KCMPNM cannot be undefined
1393	Cannot Write Table of file:
1394	Unexpected option on PICKPREFS; expecting ON, OFF, or blank.
1400	SEISMGR SAC INTERFACE
1401	Data may be corrupt. Proceed with caution.
1402	Data may have been removed. Proceed with caution.
1403	Cannot CUTIM: would result in too many files in memory.
1404	Cannot CUTIM: would exceed length of filename list (use shorter filenames)
1405	Cannot CUTIM: illegal cut point information
1406	Data has been corrupted, re-read or regenerate new data.
1500	GRAPHICS ACTION MODULE
1501	Floor used
1502	Bad cursor position. Please retry.
1503	Invalid character. Please retry.
1504	Probable discrepancy in reference date fields in headers.
1505	Must specify at least two data file list names or numbers.
1600	SPECTRAL ANALYSIS MODULE
1601	File and filter sampling intervals not equal for
1602	Inadequate memory to perform FIR filter using DFT.
1603	Inadequate memory to perform FIR filter.
1604	Following file now in amplitude-phase format:
1605	Following file now in real-imaginary format:
1606	Maximum allowable DFT is
1607	DC level after DFT is
1608	Bad Wiener filter noise window for file
1609	Numerical instability in Wiener filter for file
1610	Unwrap failed at data point for file
1611	Corner frequency greater than Nyquist for file
1612	Window length exceeds maximum:
1613	Minimum size of data file for Hilbert transform is
1614	Numerical instability in Wiener; will retry with epsilon =
1615	Noise window outside of data window
1616	Noise window larger than data window
1617	Noise window partially outside of data window

... continued on next page

Number	Translation
1618	Order = 0 in HQR
1619	HQR failed, too many iterations
1620	Gain out of range, Filterdesign failed for Whitening coefficients for file:
1700	UNARY OPERATIONS MODUDE
1701	Can't divide by zero.
1702	Non-positive values found in file
1800	BINARY OPERATIONS MODULE
1801	Header field mismatch:
1802	Time overlap:
1803	No binary data files read in.
1804	Illegal binary data file list number:
1805	Time gap (zeros added):
1900	EVENT ANALYSIS MODULE
1901	Can't open HYPO pick file
1902	Can't open card image pick file
1903	Can't close previous card image pick file.
1904	All global card image pick files are in use.
1905	Need an integer. Retry.
1906	Need an integer in the range 0 to 4. Retry.
1907	HYPO line already written.
1908	HYPO pick file not open.
1909	Can't compute waveform.
1910	No valid pick found for the following file(s):
1911	Can't estimate back azimuth because of
2000	SIGNAL CORRECTION MODULE
2001	Command requires an even number of data files.
2002	Following files are not an orthogonal pair:
2003	Following files are not both horizontals:
2004	Insufficient header information for rotation:
2005	Points outside file's time window set to zero =
2006	Gains must be monotonically decreasing.
2007	Data clipped for file
2008	Requested begin time is less than files begin time. Output truncated.
2009	Requested end time is greater than files end time. Output truncated.
2010	Number of points in pair of files are not equal:
2011	Cannot read filter coefficient file:
2012	Interpolate data dx not positive:
2100	INSTRUMENT CORRECTION MODULE
2101	Need free period and magnification for ELMAG.
2102	Need number of zeros for EYEOMG.
2103	Need number of zeros, free period, scale and damping factors for GENL.
2104	Need an instrument sub-type for
2105	Unknown instrument sub-type for

... continued on next page

Number	Translation
2106	Need free period and damping factor for LLL sub-type BB.
2107	Need free period, damping factor, and corner frequency for PORT.
2108	Maximum number of poles exceeded in POLEZERO file:
2109	Maximum number of zeros exceeded in POLEZERO file:
2110	Illegal option in POLEZERO file:
2111	Taper frequency limits are invalid. No taper applied.
2112	Incorrect value for free period or magnification for ELMAG.
2113	Need free period, damping, corner, gain, and highpass for REFTEK.
2114	No response information for this channel in response file.
2115	No response file found in database
2116	Not a recognized response file type.
2117	SUBTYPE and FNAME options not compatible with DBASE, filenames ignored.
2118	No transfer function applied.
2119	The SCALE option is not required if the TRANSFER command is to be used on data. Out of date: scale no longer used in transfer
2120	Interpolation Failed: adjacent frequencies indistinguishable. Freq:
2121	NDC transfer had an OS error.
2122	NDC transfer had an application error.
2123	NDC transfer had a SQL error.
2124	NDC transfer had an unknown error.
2125	Interpolate begin value too large:
2126	Bad pole value in file
2127	Bad zero value in file
2200	GRAPHICS DEVICE MODULE
2201	First three elements in color table entry must be numeric:
2202	Size of passed color table arrays are too large.
2203	Bad values in color table arrays found and corrected:
2204	opening font file:
2205	reading font file:
2300	GRAPHICS DEVICE 1
2301	No TERM environmental variable set.
2400	GRAPHICS DEVICE 2
2401	Can't find an unused SAC Graphics File.
2402	Can't PRINT on ENDFRAME if SGF device is not on.
2403	Ignoring PRINT option in the middle of a frame.
2404	SPECTROGRAM, SONOGRAM, and IMAGE only PRINT if SGF is the only graphics device running
2405	Cannot PRINT: no SGF files produced.
2500	GRAPHICS DEVICE 3
2600	GRAPHICS DEVICE 4
2700	CONDITIONAL EXECUTION MODULE
2701	Syntax error in DO statement
2702	Do loop list exceeds maximum number of characters =

... continued on next page

Number	Translation
2703	Can't evaluate logical expression:
2704	Reading macro file
2705	Searching macro file for
2800	NEURAL NETWORK MODULE
2801	All data files must have the same number of data points.
2900	XYZ (3-D) DATA PROCESSING MODULE
2901	No xyz data in memory.
2902	Zoomed input too large to display. Maximum dimension is
3000	CONTOURING MODULE
3001	Exceeded maximum number of contouring levels:
3501	Plot Label number exceeds total number of current labels:
4000	NUMBER CONVERSION MODULE
4002	Number out of range
4003	Number too small, near -inf
4004	Number too large, near inf
4005	Number string contains non-numeric characters
4006	Number string contains extra non-numeric characters
4007	Number string not converted
4008	Number below resolution, number too small, near 0.0
5000	SPECTRAL ESTIMATION SUBPROCESS.
5001	Spectral Estimation Subprocess.
5002	Only one file can be processed by SPE at a time.
5003	No correlation function calculated.
5004	No spectral estimate calculated.
5005	Error within Dave Harris's subroutine package.
5006	A single evenly spaced data file is not in memory.
5007	Confidence limits option not currently implemented.
5000	SIGNAL STACKING SUBPROCESS.
5101	Signal Stacking Subprocess.
5102	No files in stack file list.
5103	No time window defined.
5104	No distance defined for file(s)
5105	Time window mismatch:
5106	File name not in file list:
5107	File number not in file list:
5108	Maximum length of stack file list exceeded:
5109	Sampling intervals are not equal.
5110	Illegal velocity model number:
5111	Error in calculating velocity model values.
5112	Insufficient input for velocity model calculation.
5113	No valid stack sum exists.
5120	Cannot use both model and file. Continuing with model.
5121	Data file expected, none found.

... continued on next page

Number	Translation
5122	Distance out of range of data; blackboard variable not set:
5123	No blackboard variable name given, no variable set.
5124	TAUP and MODEL options are incompatable in TRAVELTIME. Input file required (no longer relevant)
5125	No phases found
5200	FIR FILTER DESIGN SUBPROCESS
5300	FK module
5301	Station and event latitudes and longitudes must be set for this command.
5302	Unable to determine offsets, type HELP BEAM
5303	OFFSET set to REF, but no reference data, use REFERENCE option
5304	OFFSET set to USER, but some files missing USER7 or USER8
5305	OFFSET set to STATION, but some files missing STLA or STLO
5306	OFFSET set to EVENT, but some files missing EVLA or EVLO
5307	Illeagle setting for OFFSET option
5308	Number of files must be between 3 and MXLENS
6000	DATA SET MODULE
6001	Can't find file in data-set file index.
6002	No more data-sets available.
6003	Max number of files in data-set memory. No more room.
6004	Invalid data set name. Must be a character string.
6005	Another data-set already exists by this name
6006	Maximum allowed number of current data-set exceeded.
6008	Bad syntax in command.
7006	Illegal window size
7007	Location could not be transformed
7008	Exceeded maximum size of a data array:
7009	Illegal option found on card:
7010	Unable to open ZONESDATA file.
7011	Unable to open gctp messages file tmp.????.
8001	SETMAT takes only one parameter.
8002	Cannot link to MATLAB shared object:
8003	Cannot link to a MATLAB function:
8004	Cannot start MATLAB
8100	ORACLE DATABASE CONNECTION Oracle no longer supported
8101	command or option not operational; requires Oracle version
8201	No data points found for gettime
9000	MERGE
9005	Amplitude mismatch

LATEST REVISION

July 2011 (Version 101.5)

2 SAC Commands

Alphabetical Comamnd Listing

3c: Launch a Matlab GUI for manipulating 3-component data.

about: Displays version and copywrite information.

absolutevalue: Takes the absolute value of each data point.

add: Adds a constant to each data point.

addf: Adds a set of data files to data in memory.

apk: Applies an automatic event picking algorithm.

arraymap: Produces a map of the array or "coarray" using all files in SAC memory.

axes: Controls the location of annotated axes.

bandpass: Applies an IIR bandpass filter.

bandrej: Applies an IIR bandreject filter.

bbfk: Computes the broadband frequency-wavenumber (FK) spectral estimate, using allfiles in SAC memory.

beam: Computes the beam using all data files in SAC memory.

begindevices: Begins plotting to one or two of the two possible graphics devices.

beginframe: Turns off automatic new frame actions between plots.

beginwindow: Begins plotting to a new graphics window.

benioff: Applies a Benioff filter to the data.

binoperr: Controls errors that can occur during binary file operations.

border: Controls the plotting of a border around plots.

capf: Closes the currently open alphanumeric pick file.

cd: Change the working directory within SAC.

chnhdr: Changes the values of selected header fields.

commit: Commits (dopies) SAC data to the I/O buffers

chpf: Closes the currently open HYPO pick file.

color: Controls color selection for color graphics devices.

comcor: Controls SAC's command correction option.

contour: Produces contour plots of data in memory.

convert: Converts data files from one format to another.

convolve: Compute the convolution of a master signal with itself and one or more other signals.

copyhdr: Copies header variables from one file in memory to all others.

correlate: Computes the auto- and cross- correlation functions.

crr: Commit, Rollback, Recalltrace

cut: Defines how much of a data file is to be read.

cuterr: Controls errors due to bad cut parameters.

cutim: Cuts files in memory. Can cut multiple segments from each file.

datagen: Generates sample data files and stores them in memory.

decimate: Decimates (downsamples) data, including an optional anti-aliasing FIR filter.

deletechannel: Deletes one or more files from the file list.

depmec: Launch a Matlab GUI for estimating source depth and mechanism.

dif: Differentiates data in memory.

div: Divides each data point by a constant.

divf: Divides data in memory by a set of data files.

divomega: Performs integration in the frequency domain.

echo: Controls echoing of input and output to the terminal.

enddevices: Terminates one or more graphics devices.

endframe: Resumes automatic new frame actions between plots.

envelope: Computes the envelope function using a Hilbert transform.

erase: Erases the graphics display area.

evaluate: Evaluates simple arithmetic expressions.

exp: Computes the exponential of each data point.

exp10: Computes the base 10 exponential (10.**y) of each data point.

fft: Performs a discrete Fourier transform.

fileid: Controls the file id display found on most SAC plots.

filenumber: Controls the file number display found on most SAC plots.

filterdesign: Produces a graphic display of a filter's digital vs. analog characteristics

fir: Applies a finite-impulse-response filter.

floor: Puts a minimum value on logarithmically scaled data.

funcgen: Generates a function and stores it in memory.

getbb: Gets (prints) values of blackboard variables.

grayscale: Produces grayscale images of data in memory.

grid: Controls the plotting of grid lines in plots.

gtext: Controls the quality and font of text used in plots.

hanning: Applies a "hanning" window to each data file.

help: Displays information about SAC commands and features on the screen.

highpass: Applies an IIR highpass filter.

hilbert: Applies a Hilbert transform.

history: prints a list of the recently issued SAC commands

ifft: Performs an inverse discrete Fourier transform.

image: Produces color sampled image plots of data in memory.

inicm: Reinitializes all of SAC's common blocks.

installmacro: Installs macro files in the global SAC macro directory.

int: Performs integration using the trapezoidal or rectangular rule.

interpolate: Interpolates evenly or unevenly spaced data to a new sampling rate.

keepam: Keep amplitude component of spectral files (of either the AMPH or RLIMformat) in SAC memory.

khronhite: Applies a Khronhite filter to the data.

line: Controls the linestyle selection in plots.

linefit: Computes the best straight line fit to the data in memory and writesthe results to header blackboard variables.

linlin: Turns on linear scaling for the x and y axes.

linlog: Turns on linear scaling for x axis and logarithmic for y axis.

listhdr: Lists the values of selected header fields.

load: Load an external command.

loadctable: Allows the user to select a new color table for use in image plots.

log: Takes the natural logarithm of each data point.

log10: Takes the base 10 logarithm of each data point.

loglab: Controls labels on logarithmically scaled axes.

loglin: Turns on logarithmic scaling for x axis and linear for y axis.

loglog: Turns on logarithmic scaling for the x and y axes.

lowpass: Applies an IIR lowpass filter.

macro: Executes a SAC macro file.

map: Generate a GMT (Generic Mapping Tools) map which can include station/eventsymbols topography and station names using all the files in SAC memory

markptp: Measures and marks the maximum peak to peak amplitude of each signal withinthe measurement time window.

marktimes: Marks files with travel times from a velocity set.

markvalue: Searches for and marks values in a data file.

mat: Copy SAC workspace into Matlab and either execute a user-specifiedm-file or else get a Matlab prompt for interactive manipulation.

mathop: provides options for sequncing in inline expressions

merge: Merges (concantenates) a set of files to data in memory.

message: Sends a message to the user's terminal.

mtw: Determines the measurement time window for use in subsequent measurementcommands.

mul: Multiplies each data point by a constant.

mulf: Multiplies a set of files by the data in memory.

mulomega: Performs differentiation in the frequency domain.

nplotc: Annotates SAC plots and creates figures using cursor.

null: Controls the plotting of null values.

oapf: Opens a alphanumeric pick file.

ohpf: Opens a HYPO formatted pick file.

pause: Sends a message to the terminal and pauses.

pickauthor: Tell sac to read author list (and possibly phase pick information) froma user-defined preferences file, or interactively enter author list on

pickphase: Tell sac to read phase pick information (and possibly the author list) froma user-defined preferences file, or interactively enter phase pick information

pickprefs: Control the way that SAC manages and or loadspicks from a variety of input data formats (e.g., CSS, GSE, SUDS etc...)

picks: Controls the display of time picks on most SAC plots.

plabel: Defines general plot labels and their attributes.

plot: Generates a single-trace single-window plot.

plot1: Generates a multi-trace multi-window plot.

plot2: Generates a multi-trace single-window (overlay) plot.

plotalpha: Reads alphanumeric data files on disk into memory and plots the data to thecurrent output device.

plotc: Annotates SAC plots and creates figures using cursor.

plotctable:

plotdy: Creates a plot with error bars.

plotpk: Produces a plot for the picking of arrival times.

plotpktable:

plotpm: Generates a "particle-motion" plot of pairs of data files.

plotsp: Plots spectral data in several different formats.

plotxy: Plots one or more data files versus another data file.

print: Prints the most recent SGF file.

printhelp: Prints hardcopies of information about SAC commands and features.

production: Controls the production mode option.

qdp: Controls the "quick and dirty plot" option.

quantize: Converts continuous data into its quantized equivalent.

quit: Terminates SAC.

quitsub: Terminates the currently active subprocess.

read: Reads data from SAC data files on disk into memory.

readbbf: Reads a blackboard variable file into memory.

readcss: Read data files in CSS external format from disk into memory.

readdb: Reads data from Oracle database into memory. NOT SUPPORTED

readerr: Controls errors that occur during the READ command.

readgse: Read data files in GSE 2.0 format from disk into memory.

readhdr: Reads headers from SAC data files into memory.

readsdd: Reads data from SDD data files on disk into memory.

readsp: Reads spectral files written by WRITESP and WRITESPE.

readsuds: Read data files in PC-SUDS format from disk into memory.

readtable: Reads alphanumeric data files in collumn format on disk into memory.

recalltrace: rolls back the laste committed waveform and most header fields

report: Informs the user about the current state of SAC.

reverse: Reverse the order of data points.

rglitches: Removes glitches and timing marks.

rmean: Removes the mean.

rms: Computes the root mean square of the data within the measurement time window.

rollback: reverts SAC to last committed version in I/O buffers

rotate: Rotates a pair of data components through an angle.

rq: Removes the seismic Q factor from spectral data.

rtrend: Removes the linear trend.

saveimg: Sves displayed graphics windows in several formats

scallop: Calculate a spectrogram equal to the difference between two smoothed versionsof the same spectrogram.

setbb: Sets (defines) values of blackboard variables.

setdevice: Defines a default graphics device to use in subsequent plots.

setmacro: Defines a set of directories to search when executing a SAC macro file.

sgf: Controls the SAC Graphics File (SGF) device options.

smooth: Applies an arithmetic smoothing algorithm to the data.

sonogram: Calculate a spectrogram equal to the difference between two smoothed versionsof the same spectrogram.

sort: Sorts files in memory by header fields.

spectrogram: Calculate a spectrogram using all of the data in memory.

sqr: Squares each data point.

sqrt: Takes the square root of each data point.

stretch: Stretches (upsamples) data, including an optional interpolating FIR filter.

sub: Subtracts a constant from each data point.

subf: Subtracts a set of data files from data in memory.

symbol: Controls the symbol plotting attributes.

synchronize: Synchronizes the reference times of all files in memory.

systemcommand: Executes system commands from SAC.

taper: Applies a symmetric taper to each end of data.

ticks: Controls the location of tick marks on plots.

title: Defines the plot title and attributes.

trace: Controls the tracing of blackboard and header variables.

transcript: Controls output to the transcript files.

transfer: Performs deconvolution to remove an instrument response and convolution to apply another instrument response.

transfertable: Details on older instrument tpes called in transfer

traveltime: Computes traveltime curves for pre-defined models

tsize: Controls the text size attributes.

unsetbb: Unsets (deletes) blackboard variables.

unwrap: Computes amplitude and unwrapped phase.

vspace: Changes the maximum size and shape of plots.

wait: Tells SAC whether or not to pause between plots.

whiten: Flattens the spectrum of the input time series.

whpf: Writes auxiliary cards into the HYPO pick file.

width: Controls line-width selection for graphics devices.

wiener: Designs and applies an adaptive Wiener filter.

wild: Sets wildcard characters used in read commands to expand filelists.

window: Sets the location and shape of graphics windows.

write: Writes data in memory to disk.

writebbf: Writes a blackboard variable file to disk.

writecss: Writes data in memory to disk in CSS 3.0 format.

writegse: Write data files in GSE 2.0 format from memory to disk.

writehdr: Overwrites the headers on disk with those in memory.

writesdd: Writes data in memory to disk in SDD format.

writesp: Writes spectral files to disk as "normal" data files.
xdiv: Controls the x axis division spacing.
xfudge: Changes the x axis "fudge factor."
xfull: Controls plotting of x axis full logarithmic decades.
xgrid: Controls plotting of grid lines in the x direction.
xlabel: Defines the x axis label and attributes.
xlim: Determines the plot limits for the x axis.
xlin: Turns on linear scaling for the x axis.
xlog: Turns on logarithimic scaling for the x axis.
xvport: Defines the viewport for the x axis.
ydiv: Controls the y axis division spacing.
yfudge: Changes the y axis "fudge factor."
yfull: Controls plotting of y axis full logarithmic decades.
ygrid: Controls plotting of grid lines in the y direction.
ylabel: Defines the y axis label and attributes.
ylim: Determines the plot limits for the y axis.
ylin: Turns on linear scaling for the y axis.
ylog: Turns on logarithimic scaling for the y axis.
yvport: Defines the viewport for the y axis.
zcolors: Controls the color display of contour lines.
zlabels: Controls the labeling of contour lines with contour level values.
zlevels: Controls the contour line spacing in subsequent contour plots.
zlines: Controls the contour linestyles in subsequent contour plots.
zticks: Controls the labeling of contour lines with directional tick marks.

3C

SUMMARY

Launch a Matlab GUI for manipulating 3-component data.

SYNTAX

`3C options` where options are one or more of

`{AUTO}`

`{A,T0,T1,T2,T3,T4,T5,T6,T7,T8,T9}`

`{WINLEN value}`

DESCRIPTION

3c identifies all 3-component channel sets in the files currently in memory. These channel sets are copied into a Matlab workspace and a GUI is launched. Within this GUI the user may obtain estimates of back azimuth, incidence angle, and polarization, rotate traces, make 3-D particle motion plots, pick phase arrivals, and do polarization analysis using maximum likelihood estimators. Back azimuth, incidence angle, and polarization estimates are returned in the SAC header variables USER0 (KUSER0), USER1 (KUSER1), and USER2 (KUSER2). Picks are returned in T0 - T9 (KT0 - KT9). Instructions for using the GUI are available through a built-in help system in the GUI.

When the AUTO option is specified, 3c does the polarization analysis without intervention using a window starting at the specified time marker {A,T0,T1,T2,T3,T4,T5,T6,T7,T8,T9}, and a window length of WINLEN.

HEADER CHANGES

USER0, KUSER0, USER1, KUSER1, USER2, KUSER2, T0 - T9, KT0 - KT9

ERROR MESSAGES

No 3-component sets found.

LATEST REVISION

June 5, 1997 (Version 00.53a)

ABOUT

SUMMARY

Displays version and copywrite information.

SYNTAX

```
ABOUT
```

LATEST REVISION

January 20, 1999 (Version 0.58)

ABS

SUMMARY

Takes the absolute value of each data point.

SYNTAX

ABS

ERROR MESSAGES

- 1301: No data files read in.
- 1307: Illegal operation on spectral file

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

LATEST REVISION

January 8, 1983 (Version 8.0)

ADD

SUMMARY

Adds a constant to each data point.

SYNTAX

```
ADD {v1 {v2 ...  vn} }
```

INPUT

v1: Constant to add to first file.
v2: Constant to add to second file.
vn: Constant to add to nth file.

DEFAULT VALUES

```
ADD 0.0
```

DESCRIPTION

This command will add a constant to each element of each data file in memory. The constant may be the same or different for each data file. If there are more data files in memory than constants, then the last constant entered is used for the remainder of the data files.

EXAMPLES

To add 5.1 to each element of F1 and 6.2 to each element of F2 and F3:

```
u:  READ F1 F2 F3
u:  ADD 5.1 6.2
```

ERROR MESSAGES

- 1301: No data files read in.
- 1307: Illegal operation on spectral file

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

LATEST REVISION

January 8, 1983 (Version 8.0)

ADDF

SUMMARY

Adds a set of data files to data in memory.

SYNTAX

```
ADDF {NEWHDR ON|OFF} filelist
```

INPUT

NEWHDR ON|OFF: By default, the resultant file will take its header field from the original file in memory. Turning NEWHDR ON, causes the header fields to be taken from the new file in the filelist.

filelist: A list of SAC binary data files. This list may contain simple filenames, full or relative pathnames, and wildcard characters. See the READ command for a complete description.

DESCRIPTION

This command can be used to add a single file to a set of files or to add one set of files to another set. An example of each case is presented below. The files must be evenly spaced and should have the same sampling interval and number of data points. These last two restrictions can be eliminated using the BINOPERR command. If there are more data files in memory than in the filelist, then the last file in the filelist is used for the remainder of the data files in memory.

EXAMPLES

To add one file to three other files:

```
u:  READ FILE1 FILE2 FILE3
u:  ADDF FILE4
```

To add two files to two other files:

```
u:  READ FILE1 FILE2
u:  ADDF FILE3 FILE4
```

HEADER CHANGES

If NEWHDR is OFF (the default) the headers in memory are unchanged). If NEWHDR is ON, the headers are replaced with the headers from the files in the filelist.

DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES

- 1301: No data files read in.
- 1803: No binary data files read in.
- 1307: Illegal operation on spectral file

- 1306: Illegal operation on unevenly spaced file
- 1801: Header field mismatch:
  - sampling interval or number of points are not equal.
  - can be controlled using the BINOPERR command.

WARNING MESSAGES

- 1802: Time overlap: - the file addition is still performed.

SEE COMMANDS

READ, BINOPERR

LATEST REVISION

May 26, 1999 (Version 0.58)

APK

SUMMARY

Applies an automatic event picking algorithm.

SYNTAX

```
APK {param v {param v} ...  },{VALIDATION ON|OFF}
```

INPUT

param v: Define a new value for one of the pick parameters.

param: C1|C2|C3|C4|C5|C6|C7|C8|D5|D8|D9|I3|I4|I6. ,BREAK These parameters are defined below.

VALIDATION ON: Turn validation phase on.

VALIDATION OFF: Turn validation phase off.

DEFAULT VALUES

```
APK C1 0.985 C2 3.0 C3 0.6 C4 0.03 C5 5.0 C6 0.0039 C7 100.  C8 -0.1 D5
2.  D8
3.  D9 1.  I3 3 I4 40 I6 3 VALIDATION ON
```

DESCRIPTION

The algorithm used in this automatic picker was originally obtained from the USGS in Menlo Park and is based upon work by Rex Allen (see reference below.) The detection of a pick is based upon abrupt changes in the ratio of a short term and long term running average of the signal. Once detected, the pick is subjected to an optional validation phase which attempts to distinguish a true event from cultural noise. Once validated, the pick is further evaluated to determine other characteristics of the event. Currently this is limited to its duration. Other features such as maximum amplitude, period, and decay rate may be added as required. Most of the parameters in this command need never be changed. They are available if the user wishes to fine tune the algorithm. Most of these parameters have the same meaning here as they do in the referenced article.

1. C1 is the constant used in the recursive high pass filter that is applied to remove any D.C. bias.

2. C2 is the constant used to vary the weight assigned to the amplitude and first difference in the characteristic function.

3. C3 is the timing constant, used to compute the short term average of the characteristic function.

4. C4 is the timing costant used to compute the long term average of the characteristic function.

5. C5 is the constant used to compute the threshold reference level. A potential event is declared when the short term average becomes larger than C5 times the long term average.

6. C6 is the timing constant used to compute the running mean absolute value of the filtered data.

7. A station is assumed to be dead when the absolute value of the characteristic function is greater than C7.

8. C8 is used to determine the signal termination level. The signal is terminated when its absolute value falls below this level for D8 seconds. There are currently two different algorithms in use so C8 has two different interpretations. If C8 is positive, then the termination level is C8 times the running

mean absolute value of the signal just before the event was declared. This method is useful if the background level at a station is large. If C8 is negative, then the termination level is the absolute value of C8. This will give more consistent terminations from station to station if the noise level is well below this termination level.

9. D5 is the minimum duration in seconds for an event to be declared valid.

10. D9 is the duration in seconds used to initialize the long term average of the characteristic function.

11. I3, I4, and I6 are integer constants used during the validation phase and should not be changed.

HEADER CHANGES

The time of the pick is stored into A; the quality and sense of motion is stored into KA; the end of the event is stored into F.

ERROR MESSAGES

- 1301: No data files read in.
- 1306: Illegal operation on unevenly spaced file
- 1307: Illegal operation on spectral file

WARNING MESSAGES

- 1910: No valid pick found for the following file(s):

SEE COMMANDS

OHPF, OAPF Rex V. Allen, Automatic Earthquake Recognition and Timing from Single Traces, BSSA, Vol. 68, No. 5, Oct. 1978.

LATEST REVISION

May 15, 1987 (Version 10.2)

ARRAYMAP

SUMMARY

Produces a map of the array or "coarray" using all files in SAC memory.

SYNTAX

```
ARRAYMAP ARRAY | COARRAY
```

INPUT

ARRAY: This option maps the offsets X and Y, assumed to have been set up in the SAC header (see the HEADER DATA section below).

COARRAY: This option plots delta X and delta Y for all pairs of stations.

DEFAULT VALUES

```
ARRAYMAP ARRAY
```

INPUT

HEADER DATA: The following header variables must be set up in advance, using the SAC macro WRXYZ, or its functional equivalent. All offsets are measured in kilometers from a reference location.

USER7: Contains easterly offset (x).

USER8: Contains northerly offset (y). The upward offset (z) is not used by this command.

LIMITATIONS

Maximum number of stations allowed in BBFK command.

SEE COMMANDS

WRXYZ. This is a SAC macro; It can be found in the global SAC macro directory, SACAUX/macros . Documentation provided in the macro. BBFK July 22, 1991 (Version 10.5c)

AXES

SUMMARY

Controls the location of annotated axes.

SYNTAX

```
AXES   ON|OFF|ONLY   sides
```

where sides is the keyword:

```
ALL
```

or one or more of the following:

```
TOP,BOTTOM,RIGHT,LEFT
```

ALTERNATE FORMS

AXIS may be used for AXES. (Useful for grammarians only.)

INPUT

ON: Turn axes on for listed sides; others unchanged.
OFF: Turn axes off for listed sides; others unchanged.
ONLY: Turn axes on only for listed sides; others off.
ALL: All four axes.
TOP: X axis above plot.
BOTTOM: X axis below plot.
RIGHT: Y axis to right of plot.
LEFT: Y axis to left of plot.

DEFAULT VALUES

```
AXES ONLY BOTTOM LEFT
```

DESCRIPTION

Axes can be drawn on one or more of the four sides of a plot. Axes annotation is drawn using the division spacing set by the XDIV command. Tick mark labeling is controlled independently using the TICKS command.

EXAMPLES

To turn on the top axes and leave the others unchanged:

```
u:  AXES ON TOP
```

To turn off all axes annotation:

```
u:  AXES OFF ALL
```

To turn axes annoation on for the bottom side and off for the rest:

```
u:  AXES ONLY BOTTOM
```

SEE COMMANDS

XDIV, TICKS

LATEST REVISION

January 8, 1983 (Version 8.0)

BANDPASS

SUMMARY

Applies an IIR bandpass filter.

SYNTAX

```
[B]AND[P]ASS {[BU]TTER|[BE]SSEL|C1|C2},{[C]ORNERS v1 v2}, {[N]POLES n},{[P]ASSES
n},{[T]RANBW v},{[A]TTEN v}
```

INPUT

BUTTER: Apply a Butterworth filter.
BESSEL: Apply a Bessel filter.
C1: Apply a Chebyshev Type I filter.
C2: Apply a Chebyshev Type II filter.
CORNERS v1 v2: Set corner frequencies to V1 and V2.
NPOLES n: Set number of poles to N {range: 1-10}.
PASSES n: Set number of passes to N {range: 1-2}.
TRANBW v: Set the Chebyshev transition bandwidth to v.
ATTEN v: Set the Chebyshev attenuation factor to v.

DEFAULT VALUES

```
BANDPASS BUTTER CORNER 0.1 0.4 NPOLES 2 PASSES 1 TRANBW 0.3 ATTEN 30.
```

DESCRIPTION

A set of Infinite Impulse Response (IIR) filters is available in SAC. These recursive digital filters are all based upon classical analog designs: Butterworth, Bessel, Chebyshev type I, and Chebyshev type II. These analog prototype filters are mapped to digital filters via the bilinear transformation, a transformation which preserves the stability of the analog prototypes. A complete description of this method of design can be found in the reference given below. However, it is not necessary to read that description, unless you want complete control over the more complicated Chebyshev filters.

Generally speaking, the Butterworth filter is a good choice for most applications, since it has a fairly sharp transition from pass band to stop band, and its group delay response is moderate. The Butterworth filter is the default filter type. It's 3 db point is at the designated cutoff frequency. The Bessel filter is best for those applications which require linear phase without twopass filtering. It's amplitude response is not very good. The SAC Bessel filters have been normalized so that their 3 db points are also at the designated cutoff freqency. The two Chebyshev filters are included for situations which require very rapid transitions from pass band to stop band. Although they have good magnitude discrimination, their group delay responses are the worst among the filters contained in SAC.

Some caution must be exercised in applying these filters. First, all recursive filters have non-linear phase, which can result in some dispersion of filtered waveforms. For applications where the phase of the resulting filtered waveform is important, a zero-phase implementation of the recursive filters is provided. Zero-phase filtering is possible by running the filter forward and backward over the data, instead of just forward over the data. This two-pass operation results in a effective filter magnitude response which is the square of the original magnitude response. It also results in a non-causal filter impulse response, which can leave a signal containing a sharp time onset with a ringing precursor. For

this reason, you should not measure arrival times of data that has been filtered using this two-pass option. For cases where signal precursors cannot be tolerated, such as onset picking operations, it may not be a good idea to do two-pass filtering. Second, the filters can become numerically unstable if the width of the filter pass band is very small compared to the folding frequency of the data. The problem is only aggravated by increasing the number of poles in the filter. Situations that seem to require an exceptionally narrow band filter can be handled more reliably by decimation, filtering with a filter of more moderate band width, and interpolation to the original sampling rate. Recourse to this resampling strategy should be made when the filter band width drops below a few percent of the folding frequency.

Generally, the filter will have a sharper transition from pass band to stop band as the number of poles is increased. However, there are penalties for using a large numbers of poles. Filter group delays generally get wider as the number of poles increases, resulting in worse dispersion of the filtered waveform. Applications that appear to require more than three or four poles should probably be reconsidered.

The design of Butterworth and Bessel filters is particularly simple. You simply specify the cutoffs of the filter and the number of poles. Chebyshev filters are more complicated to design. In addition to cutoffs and number of poles, you must supply a transition band width, and a stop band attenuation factor for the analog prototype filter. The transition band width is the width of the region between the filter pass band and stop band. It is specified as a fraction of the analog prototype pass band width.

Due to the non-linear warping of the frequency axis of the bilinear transformation, the transition band width of the recursive digital filter may be smaller than that specified in the design. In SAC, the analog prototype filter cutoffs are compensated to ensure that they map to the requested cutoffs after the bilinear transformation is performed. The same is not true of the stop band edges. Consequently, if precisely located stop band edges are necessary, you must compensate for this shrinkage when choosing your cutoffs.

The stop band attenuation is specified as the ratio of the pass band gain to the stop band gain.

EXAMPLES

To apply a four-pole Butterworth with corners at 2 and 5 Hz.:

```
SAC>  BANDPASS NPOLES 4 CORNER 2 5
```

To later apply a two-pole two-pass Bessel with the same corners.:

```
SAC>  BP N 2 BE P 2
```

ERROR MESSAGES

- 1301: No data files read in.
- 1306: Illegal operation on unevenly spaced file
- 1307: Illegal operation on spectral file
- 1002: Bad value for
  - corner frequency larger than Nyquist frequency.

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

LATEST REVISION

January 8, 1983 (Version 8.0) Magnitude Frequency Response of Chebyshev Type II Filter

BANDREJ

SUMMARY

Applies an IIR bandreject filter.

SYNTAX

```
BANDREJ {BUTTER|BESSEL|C1|C2},{CORNERS v1 v2},
{NPOLES n},{PASSES n},{TRANBW v},{ATTEN v}
```

INPUT

BUTTER: Apply a Butterworth filter.
BESSEL: Apply a Bessel Filter.
C1: Apply a Chebyshev Type I filter.
C2: Apply a Chebyshev Type II filter.
CORNERS v1 v2: Set corner frequencies to v1 and v2.
NPOLES n: Set number of poles to n {range: 1-10}.
PASSES n: Set number of passes to N {range: 1-2}.
TRANBW v: Set the Chebyshev transition bandwidth to V.
ATTEN v: Set the Chebyshev attenuation factor to V.

DEFAULT VALUES

```
BANDREJ BUTTER CORNER 0.1 0.4 NPOLES 2 PASSES 1 TRANBW 0.3 ATTEN 30.
```

DESCRIPTION

See the BANDPASS command for definitions of the filter parameters and descriptions on how to use them.

EXAMPLES

To apply a four-pole Butterworth with corners at 2 and 5 Hz.:

```
u:  BANDREJ NPOLES 4 CORNER 2 5
```

To apply a two-pole two-pass Bessel with the same corners.:

```
u:  BR N 2 BE P 2
```

ERROR MESSAGES

- 1301: No data files read in.
- 1306: Illegal operation on unevenly spaced file
- 1307: Illegal operation on spectral file
- 1002: Bad value for
  - corner frequency larger than Nyquist frequency.

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

SEE COMMANDS

BANDPASS

LATEST REVISION

January 8, 1983 (Version 8.0)

BBFK

SUMMARY

Computes the broadband frequency-wavenumber (FK) spectral estimate, using all files in SAC memory.

SYNTAX

```
BBFK {FILTER} {NORMALIZE} {EPS v} {MLM | PDS}
{EXP n} {WAVENUMBER v} {SIZE m n} {LEVELS n}
{DB} {TITLE text} {WRITE {ON | OFF} fname} {SSQ n}
{PRINT {pname} }
```

INPUT

FILTER: Apply the bandpass filter designed in the most recent FILTERDESIGN command.

NORMALIZE: Normalizes the covariance matrix with the Capo method. A good idea if the signals vary much in amplitude among channels.

EPS v: Regularization quantity for covariance matrix. Diagonal matrix entries are multiplied by (1.0 + EPS).

MLM: Use maximum likelihood method for high-resolution estimate.

PDS: Take power density spectra without maximum likelihood method.

EXP n: Power to which the wavenumber spectrum will be raised.

WAVENUMBER v: Number of waves from which to sample spectral estimates.

SIZE m n: Size of contour plot in polar mode: m is an even num of plot samples in the azimuth direction; n is an even num of plot samples in the wavenumber direction (m*n is limited to 40,000).

LEVELS n: Number of contour intervals.

DB: Log scaling of plot in decibels.

TITLE text: Title used in plot.

WRITE {ON | OFF} fname: Whether to compute & write contour data in square mode to disk (as a type xyz SAC file). fname is file or path name (may be an absolute or relative). If no filename has been specified, the default is "BBFK". ON will reactivate fname most recently used. OFF turns writing off.

SSQ n: Size of the square (number of samples taken along each margin of the square). Maximum allowed is 200.

PRINT {pname}: Prints the resulting plot to the printer named in pname, or to the default printer if pname is not used. (This makes use of the SGF capability.)

DEFAULT VALUES

```
BBFK EPS .01 PDS EXP 1 WVENUMBER 1.0 SIZE 90 32 LEVELS 11 WRITE OFF SSQ
100
(SSQ matters only if WRITE has been positively specified).
```

DESCRIPTION

The BBFK command allows the user to compute broadband frequency wavenumber spectra. It is based on the work of NAWAB et al., 1985 and many other references in the seismic and engineering literature.

Nawab, SH, FU Dowla, and RT Lacoss, Direction determination of wideband signals, IEEE Trans. Acous. Speech Sig. Proc., 33: (5), 1114-1122, 1985

FUNCTIONAL MODULE: FK Spectrum (fks)

HEADER DATA: The following logic is used to determine how to choose or calculate station/event offsets:

Case 1: If a reference station is set in KUSER1 and is the same for all files, and USER7 and USER8 are set for all files, USER7 and USER8 are used as offsets.

Case 2: If station latitude (STLA) and station longitude (STLO) are set for all files, offsets are calculated using these, using the first file as the reference station.

Case 3: If USER7 and USER8 are set for all files, they are used as offsets.

Case 4: If event latitude (EVLA) and event longitude (EVLO) are set for all files then these are used to calculate offsets, using the first station as the reference station.

OUTPUT: The polar output is plotted immediately (not retained), the square output if requested is written out to disk. The FK peak, back azimuth and wavenumber are written to blackboard variables BBFK_AMP, BBFK_BAZIM and BBFK_WVNBR respectively.

ERROR MESSAGES: Size m or n not an even number. Offsets X,Y,Z not set in USER7,8,9 of headers. Coefficients produced by FILTERDESIGN not found, or filter type used was not "BP".

LIMITATIONS: The maximum number of stations allowed is 100. The maximum size of polar contour plot is m x n = 40,000. The maximum size of square contour output is i = 200.

SEE COMMANDS

MAP: for plotting stations in an array, according to X,Y offsets stored in SAC header variables USER7 & USER8. July 22, 1991 (Version 10.6c)

BEAM

SUMMARY

Computes the beam using all data files in SAC memory.

SYNTAX

```
BEAM {BEARING v} {VELOCITY v} {REFERENCE ON|OFF| lat lon {el} }
{OFFSET  REF|USER|STATION|EVENT|CASCADE}
{EC anginc survel} {CENTER x y z} {WRITE fname}
```

INPUT

BEARING v: Bearing, in degrees from the north.

VELOCITY v: Velocity, in kilometers per second.

[REF]ERENCE lat lon {el}: Reference point. Turns REFERENCE option on and defines a reference point relative to which the offsets can be determined. (See OFFSET REF below)

lat: latitude.

lon: longitude.

el: elevation (positive is down).

[REF]ERENCE ON|OFF: Turns REFERENCE option on or off. (See OFFSET REF below) Be careful not to use REFERENCE ON the first time this option is used with BEAM unless you really want the point where the prime maridian meets the equator.

OFFSET REF: Offsets are determined relative to the reference point entered with the REFERENCE option. This requires the REFERENCE option to be on.

OFFSET USER: Offsets are taken directly from USER7, USER8, and USER9, (in the directions of latitude, longitude, and elevation, respectively). This requires all the files to have defined values of USER7 and USER8. If the EC option is set, then OFFSET USER also requires that USER9 be set.

OFFSET STATION: Offsets are determined relative to the location of the first station. This requires all the files to have defined values of STLA and STLO.

OFFSET EVENT: Offsets are determined relative to the location of the first event. This requires all the files to have defined values of EVLA and EVLO.

OFFSET CASCADE: SAC will consider each of the previous methods of determining the offsets in the order listed above, and look to see if the necessary data is present; it will use the first method for which the requesite information is available.

EC: Elevation correction:

anginc: Angle of incidence in degrees from the z axis (the more distant the signal source, the smaller anginc).

survel: Surface medium velocity in kilometers per second.

CENTER: Center station for which the beam is to be computed:

x: Easterly offset from the reference station, in meters.

y: Northerly offset from the reference station, in meters.

z: Upward offset from the reference station, in meters.

Note: CENTER positions the beam with respect to the offsets already determined according to the OFFSET option. CENTER is not intended to offset a beam great distances; it is offered as a way to offset a beam to the center of an emplacement.

WRITE: Write beam to disk.

fname: File or path name. May be an absolute or relative pathname, or simple name of a file to appear in the dir in which SAC was started.

DEFAULT VALUES

```
BEAM  B 90  V 9.0 EC 33  6.0 C  0.  0.  0.  W beam
```

DESCRIPTION

BEAM does not overwrite existing input data in SAC memory, so it can be repeatedly issued while varying bearing and velocity. The beam result is written to disk and may be targetted to a different file each time. These design features anticipates users' need to compare multiple runs of this command to find the bearing and velocity that produced the maximal beam.

HEADER DATA

See HEADER DATA section of BBFK command.

ERROR MESSAGES

CENTER parameter missing offset z, when the presence of the EC parameter requires it.

LIMITATIONS

The maximum number of stations allowed (see BBFK).

SEE COMMANDS

MAP: for plotting stations in an array, according to x and y offsets stored in SAC header variables USER7 & USER8. July 22, 1991 (Version 10.5c)

BEGINDEVICES

SUMMARY

Begins plotting to one or two of the two possible graphics devices.

SYNTAX

```
BEGINDEVICES devices
```

where devices is one or more of the following:

```
SGF, XWINDOWS
```

ALTERNATE FORMS

BEGG and BG are obsolete but acceptable names for this command.

INPUT

SGF: The SAC Graphics File device driver.
XWINDOWS: The X-windows window display system.

DESCRIPTION

The arguments to this command consists of the list of one or two graphics devices. Subsequent plots are sent to open devices. This remains in effect until the next execution of a BEGINDEVICES or ENDDEVICES command or until SAC is terminated. Details about each graphics device are given below. There are two graphics "devices" currently being supported. The first one, SAC Graphics File (SGF), opens and sends plot commands to a binary file. The second, XWINDOWS, displays plots on an X-windows-capable sreen. workstations. SGF and XWINDOWS are described in detail in the graphics help command.

SEE COMMANDS

ENDDEVICES, SGF

LATEST REVISION

Mar. 24, 2009 (Version 101.3)

BEGINFRAME

SUMMARY

Turns off automatic new frame actions between plots.

SYNTAX

```
BEGINFRAME {PRINT {pname} }
```

INPUT

PRINT {pname}: When PRINT is used with BEGINFRAME, it signals the assocated call to ENDFRAME to print the resulting plot to the printer named in pname, or to the default printer if pname is not used. (This makes use of the SGF capability.)

ALTERNATE FORMS

BEGFR is an obsolete but allowable form of this command.

DESCRIPTION

A "new frame action" is defined as the clearing of the current graphics display surface. Specifically it is:

- the erasing of the screen for a graphics terminal.
- the closing of the current file for the SGF device driver.
- the erasing of the current graphics window on a multi-window workstation.
- the advancing of the film one frame for a film device.
- the movement of the paper to a new area on a pen plotter.

Normally SAC does a new frame action before each new plot (PLOT, PLOT1, etc.) SAC stops doing this new frame action when the BEGINFRAME command is executed. It resumes automatic framing when the ENDFRAME command is executed. Therefore, all plot commands executed between these two commands will have their output placed on the same frame. By changing the viewport (XVPORT, YVPORT) between plot commands, by changing some of the various plot options, and by reading in different sets of data files, fairly complicated plots with multiple images can be easily generated. See the example and figure below. You MUST execute the ENDFRAME command to discontinue this mode and to resume automatic framing between plots.

EXAMPLES

The plot that follows was generated using the set of commands shown below. Comments about the process are given in parenthesis:

```
u:  CUT A -0.2 N 512          (set up cut and read file)
u:  READ FILE1
u:  BEGINFRAME           (turn off automatic framing)
u:  XVPORT .1 .9         (define viewport and options)
u:  YVPORT .7 .9
```

```
u:  TITLE 'SEISMIC TRACE'
u:  FILEID OFF          (turn off fileid and qdp option)
u:  QDP OFF
u:  PLOT                (plot the trace)
u:  FFT WMEAN           (take transform of data)
u:  XVPORT .1 .45       (second viewport and options)
u:  YVPORT .15 .55
u:  TITLE 'Amplitude Response (linlog)'
u:  YLIM 1E-5 1
u:  PLOTSP AM LINLOG    (plot the amplitude)
u:  XVPORT .55 .9       (third viewport and options)
u:  TITLE 'Amplitude Response (loglog)'
u:  XLIM 1 60
u:  PLOTSP AM LOGLOG    (plot amplitude again)
u:  ENDFRAME            (resume automatic framing)
u:  CUT OFF             (reset parameters used to default values)
u:  FILEID ON
u:  XLIM OFF
u:  YLIM OFF
```

The last four commands reset some of the parameters used in this operation to their default values. This is a good habit to get into, especially when writing macros, as a way of avoiding the problem of one macro effecting the operation of others that follow.

SEE COMMANDS

ENDFRAME, XVPORT, YVPORT

LATEST REVISION

May 15, 1987 (Version 10.2) Use of BEGINFRAME and ENDFRAME to Create a Special Plot

BEGINWINDOW

SUMMARY

Begins plotting to a new graphics window.

SYNTAX

```
BEGINWINDOW n
```

INPUT

n: The graphics window number to begin plotting in. There are a total of five graphics windows.

DEFAULT VALUES

```
BEGINWINDOW 1
```

DESCRIPTION

Many of the newer graphics terminals and workstations support the concept of multiple "windows". Different jobs or activities can run in each window and display their results on the screen at the same time. "X-windows" and "Sun windows" are two of the more popular systems currently available. If you are using a device that supports one of these systems, then you can use multiple graphics windows in SAC to display your results. If you are not using such a device, SAC will accept but ignore all commands that refer to multiple graphics windows. There are two commands that control the use of this multi-windowing option. The WINDOW command lets you control the location and shape of the graphics windows. The BEGINWINDOW command lets you select the window in which to display subsequent plots. BEGINWINDOW will create the requested window if it does not currently exist on your display. The WINDOW command only works BEFORE the window is created. On most systems you can also move and resize these windows dynamically using the mouse and pop-up menus. Generally but not always (you should check for yourself), the moving of a window will result in the current plot being automatically redrawn whereas the resizing of a window results in the current plot being redrawn but not rescaled. The next plot in a resized window will be scaled correctly. All text (the commands you type and SAC's responses) are displayed in the window in which you started SAC.

SEE COMMANDS

WINDOW

LATEST REVISION

May 15, 1987 (Version 10.2)

BENIOFF

SUMMARY

Applies a Benioff filter to the data.

SYNTAX

BENIOFF

DESCRIPTION

This command is a digital approximation used to emulate the response of a short-period seismograph which was used by a VELA Program started by the U. S. Air Force about 1960. This Long Range Seismic Measurements (LRSM) program used truck vans and trailers to deploy moveable seismic systems, principally in North America, to record controlled source seismic experiments. Most of the seismic profiles were radial lines or circular arcs about the Nevada Test Site (NTS). Two semi-permanent sites or installations were Kanab, UT, and Mina, NV.

LLNL continued operation of KN-UT and MI-NV after the LRSM program. These two stations used a variable-reluctance short-period seismometer (with a natural frequency of 1 Hz, critically damped) which was designed and named after Professor Hugo Benioff of Cal Tech. This short-period seismometer was coupled to a galvanometer (with a natural frequency of 5 Hz and damped to 0.9 critical). The coupling factor was nominally defined at 0.01 (or loosely coupled at low magnification settings which were used for recording the larger explosions) and the response was nearly flat-to-velocity between 1 and 5 Hz. When LLNL converted this system to a broadband, flat-to-velocity telemetered system, an analog filter was designed to shape a passband into the LRSM short-period passband. This command executes a digital equivalent of that analog shaping filter which produces an output (measured in nanometers) analogous to the LRSM short-period system.

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

LATEST REVISION

May 15, 1987 (Version 10.2)

BINOPERR

SUMMARY

Controls errors that can occur during binary file operations.

SYNTAX

```
BINOPERR {NPTS FATAL|WARNING|IGNORE},
{DELTA FATAL|WARNING|IGNORE}
```

INPUT

NPTS: Change error condition for unequal number of data points.

DELTA: Change error condition for unequal sampling intervals.

FATAL: Make error condition fatal. Control is immediately returned to the user's terminal. Additional commands typed on the same line or in the same command file are ignored.

WARNING: Send a warning message to the user. Correct the error condition and continue.

IGNORE: Correct the error condition and continue.

DEFAULT VALUES

```
BINOPERR NPTS FATAL DELTA FATAL
```

DESCRIPTION

SAC checks for certain common errors whenever you execute a binary operations module command (ADDF, DIVF, etc.) Using this command, you can control what SAC does when it finds one of these errors. If you make an error condition fatal, then SAC will stop executing the current command, will ignore all commands in its queue, will print an error message to the terminal, and will return control to you. If you make an error condition a warning, then SAC will send you a warning message, correct the condition as best it can, and continue. If you tell SAC to ignore a condition, then SAC will correct the condition and continue without telling you the condition even occurred.

One of these error conditions occurs when the number of data points in the two files to be operated on are not equal. Corrective action in this case is to perform the operation using the number of data points in the smaller file.

Another error condition occurs when the sampling intervals of the two files are not the same. The corrective action in this case is to use the sampling interval of the first data file.

EXAMPLES

Assume that FILE1 has 1000 data points and FILE2 has 950 data points.:

```
u:  BINOPERR NPTS FATAL
u:  READ FILE1
u:  ADDF FILE2
u:  ERROR:  Header field mismatch: NPTS FILE1 FILE2
```

The file addition was not performed. Assume you now type:

```
u:  BINOPERR NPTS WARNING
u:  ADDF FILE2
u:  WARNING:  Header field mismatch: NPTS FILE1 FILE2
```

The file addition was performed on the first 950 data points of each file.

SEE COMMANDS

ADDF, SUBF, MULF, DIVF

LATEST REVISION

January 8, 1983 (Version 8.0)

BORDER

SUMMARY

Controls the plotting of a border around plots.

SYNTAX

```
BORDER {ON|OFF}
```

INPUT

{ON}: Turn border plotting on.
OFF: Turn border plotting off.

DEFAULT VALUES

```
BORDER OFF
```

DESCRIPTION

When this option is on, a solid border is drawn around the sides of the plot at the edge of the viewport (see XVPORT.) Note that an axis line is always drawn on each side of the plot that contains an annotated axis (see AXES) or a set of tick marks (see TICKS). Thus the border option only applies to those sides without axes or tick marks.

SEE COMMANDS

XVPORT, YVPORT, AXES, TICKS

LATEST REVISION

January 8, 1983 (Version 8.0)

CAPF

SUMMARY

Closes the currently open alphanumeric pick file.

SYNTAX

```
CAPF
```

SEE COMMANDS

OAPF

LATEST REVISION

January 8, 1983 (Version 8.0)

CHNHDR

SUMMARY

Changes the values of selected header fields.

SYNTAX

```
CHNHDR  { file n1 n2 ...  } field v  {field v ...  }
```

INPUT

file: This is an optional keyword that can be followed by a list of numbers indicating which file headers are to be changed.

n1 n2 ...: Integers indicating the file headers to be changed.

field: The name of a SAC header variable. These variables are listed in SAC Data File Format. Also, field may be the keyword ALLT as discussed below. Note, in order to maintain internal consistency, the following header variables cannot be changed with CHNHDR: NVHDR, NPTS, NWFID, NORID, and NEVID.

v: Set the value of that field to v. The type of the field and its new value must match. Use single quotes for alphanumeric fields with embedded blanks. Use TRUE or FALSE for logical fields. YES or NO are also acceptable for logical fields. Use variable names (see SAC Data File Format) for value fields. For offset time fields (B, E, O, A, F, and Tn), v may also be of the form -- GMT v1 v2 v3 v4 v5 v6 where v1, v2, v3, v4, v5, and v6 are the GMT year, day-of-year, hour, minute, second, and millisecond of the time. If v1 is a two-digit number, SAC will assume it is in the current century, unless that would mean that the year is in the future, in which case, SAC assumes the previous century. To be certain you get what you want, use four digits.

UNDEF: Use this keyword instead of v to "undefine" a header field.

ALLT v: Add v seconds to all defined header times. Subtract v seconds from the reference time.

DESCRIPTION

This command lets you change any of SAC's header fields. A specific file or list of files can be changed by specifying the integer value(s) corresponding to the order in which the file(s) were read in. If no integer filelist is specified, all files in memory will have their header fields changed. To change the headers of the files on disk, follow this command with the WRITE or WRITEHDR command. SAC does some validity checking on the new values but you may want to verify the results using the LISTHDR command.

There are dight enumerated (I type) variaables, such as IZTYPE, IDEP, and IZTYPE. These are explained and options listed in SAC Data File Format.

There is a set of six variables in the header (NZYEAR, NZJDAY, NZHOUR, NZMIN, NZSEC, and NZMSEC) that contain the reference or "zero" time of the file. This is the only GMT in the SAC header. All other times in the header (B, E, O, A, F, and Tn) are offsets in seconds relative to this reference time. You may change the reference time and all of the defined offset times by using the "ALLT v" option. That number of seconds are added to each defined offset time. That same number of seconds is also subtracted from the reference time. This preserves the actual GMT time of the data. As a convenience, you may enter a GMT time instead of a relative time when changing the offset times. When the GMT time is entered it is converted to a relative time before storing it in the offset time field.

EXAMPLES

To define the event latitude, longitude and name in all the files in memory:

```
SAC>  CHNHDR EVLA 34.3 EVLO -118.5
SAC>  CHNHDR KEVNM 'LA goes under'
```

To define the event latitude, longitude and name in files 2 and 4:

```
SAC>  CHNHDR file 2 4 EVLA 34.3 EVLO -118.5
SAC>  CHNHDR file 2 4 KEVNM 'LA goes under'
```

To change the event type to earthquake:

```
SAC>  CHNHDR IEVTYP IQUAKE
```

To set the first arrival time to its undefined state:

```
SAC>  CHNHDR A UNDEF
```

Assume you know the GMT origin time of an event and that you want to quickly change all the times in the header so that this origin time is the zero or reference time and all other offset times are correct relative to this time. First set the origin time using the GMT option:

```
SAC>  CHNHDR O GMT 1982 123 13 37 10 103
```

Now use the [L]IST[H]DR command to find out what O is relative to the current reference time:

```
SAC>  LISTHDR O
O 123.103
```

Now use the ALLT option to subtract this value from all of the offset times and add it to the reference time. You also want to change the field that describes the type of reference time stored in these files:

```
SAC>  CHNHDR ALLT -123.103 IZTYPE IO
```

Notice the minus sign because you are subtracting this value from the offset times.

Alternatively, if you have several waveforms in memory for the same event bur with different reference times, after setting O as above, the following command will subtract off the origin time from all defined times for all files and change the reference time to origin time:

```
SAC> chnhdr allt (0 - &1,o&) IZTYPE IO
```

HEADER CHANGES

Potentially almost all header fields (exceptions given above).

ERROR MESSAGES

- 1006: Length of string variable exceeded at symbol
  - Alphanumeric header field too long.
- 1301: No data files read in.

SEE COMMANDS

LISTHDR, WRITE, WRITEHDR, SAC Data File Format

LATEST REVISION

January 8, 1983 (Version 8.0) Wording updated in Ocober 2011

CHPF

SUMMARY

Closes the currently open HYPO pick file.

SYNTAX

```
CHPF
```

DESCRIPTION

Automatically appends the instruction card "10" to the end of the file being closed.

SEE COMMANDS

OHPF, WHPF

LATEST REVISION

March 20, 1992 (Version 10.6e)

COLOR

SUMMARY

Controls color selection for color graphics devices.

SYNTAX

```
COLOR  {ON|OFF|color} options
```

where options are one or more of the following:

```
{INCREMENT {ON|OFF}}
{SKELETON color}
{BACKGROUND color}
{LIST STANDARD|colorlist}
```

and where color is one of the following:

```
WHITE|RED|GREEN|YELLOW|
BLUE|MAGENTA|CYAN|BLACK
```

SPECIAL NOTE The LIST option must appear last in this command.

INPUT

color: The name of a standard color or the number of a color from the color table.

COLOR ON: Turn color option on but don't change data color.

COLOR OFF: Turn color option off.

COLOR color: Change data color and turn color option on.

INCREMENT {ON}: Increment data color from color list after each data file is plotted.

INCREMENT OFF: Do not increment data color.

SKELETON color: Change color of skeleton to standard color name or color table number.

BACKGROUND color: Change background color to standard color name or color table number.

LIST colorlist: Change the content of the color list. Enter list of standard color names or color table numbers. Sets data color to first color in list and turns color option on.

LIST STANDARD: Change to the standard color list. Sets data color to first color in list and turns color option on.

DEFAULT VALUES

```
COLOR BLACK INCREMENT OFF SKELETON BLACK BACKGROUND WHITE LIST STANDARD
```

DESCRIPTION

This command controls color attributes for those devices which can display a large number of colors. The data color is the color that is used when plotting the data files. The data color may be automatically incremented from a color list after each data file is plotted. The skeleton color is the color used to plot and label the axes, titles, grids, and frame ids. The backgound color is the color of an empty frame, before any lines or text are plotted.

Most of the time you will select the name of a standard color, such as red or blue. This will be the color, independent of the selected graphics device. At times, however, you may want to choose a non-standard color, such as aquamarine. This can be done by "downloading" a color table to the graphics device. This color table associates a specific hue, saturation, and lightness with a specific integer number. You can then select aquamarine for a particular part of the plot by setting that attribute to the correct number from the color table. This may sound like a lot of work, but if aquamarine is your favorite color, it may be worth it.

If you are plotting several data files on the same plot, you may want each to be in a different color. This is done using the INCREMENT option. When this option is on, the data color is incremented from a list of colors each time a data file is plotted. The order of colors in the standard or default color list is given below:

```
RED, GREEN, BLUE, YELLOW, CYAN, MAGENTA, BLACK
```

You may change the order or content of this color list using the LIST option. This is useful if you are doing a series of overlay plots (see PLOT2) and want the same colors used in the same order on each plot.

EXAMPLES

To select an incrementing data color starting with red:

```
u:  COLOR RED INCREMENT
```

To select red data colors on a white backgound with a blue skeleton:

```
u:  COLOR RED BACKGROUND WHITE SKELETON BLUE
```

To set up an incrementing data color list of red, white, and blue with an aquamarine (!!!) background:

```
u:  COLOR RED INCREMENT BACKGROUND 47 LIST RED WHITE BLUE
```

The above example assumes that aquamarine is color 47 in the color table for the selected graphics device. Background color is currently being ignored.

LATEST REVISION

April 13, 1987 (Version 10.1)

COMCOR

SUMMARY

Controls SAC's command correction option.

SYNTAX

```
COMCOR {ON|OFF}
```

INPUT

ON: Turn command correction option on.
OFF: Turn command correction option off.

DEFAULT VALUES

```
COMCOR OFF
```

DESCRIPTION

SAC checks the form and content of each command you type. When it detects an error, it sends an error message to you, telling you what the error was and where it occurred. If the command correction option is on, SAC then lets you correct the command and have SAC automatically reexecute it. If this option is off, SAC merely prints the error message and returns control to you. More details and several examples are given in the User's Guide. "Command Correction Capability" in the SAC Users Manual.

LATEST REVISION

October 11, 1984 (Version 9.1)

CONTOUR

SUMMARY

Produces contour plots of data in memory.

SYNTAX

```
CONTOUR {ASPECT ON|OFF}
```

INPUT

ASPECT {ON}: Turn aspect ratio option on. When this option is on, the viewport of the contour plot will be adjusted to maintain the y to x aspect ratio of the data.

ASPECT OFF: Turn aspect ratio option off. When off, the full viewport is used.

DEFAULT VALUES

```
CONTOUR  ASPECT  OFF
```

DESCRIPTION

This command can be used to produce a contour plot of the of any other two-dimensional array data, including the output of the SPECTROGRAM command. The SAC data plotted by this command must of of file type "XYZ" (SAC header variable IFTYPE set to "IXYZ"). Several commands control how the data is displayed: ZLEVELS for the spacing and number of contour levels, ZLINES for linestyles, ZLABELS for contour labeling, ZTICKS for directional tick marks, and ZCOLORS for line colors. Depending upon the contouring options selected, two different contouring algorithms are used. A fast scan method is used if no only solid linestyles are selected and no tick marks or labels are requested. Otherwise, a slower method, where entire line segments are first assembled before they are drawn, is used. You may want to use the fast scan method for a quick look at your data and then select other options for a final version.

EXAMPLES

In the first example (shown below) a file is read and contoured using default values.

EXAMPLES (cont.): In this example, the same file is read and the header is listed to determine the range of the z data (DEPMIN and DEPMAX.) Only selected portions of the output from LISTHDR are shown. A range of contour levels between 700 km and 1150 km and an increment of 25 km is selected. A list of four linestyles is selected, starting with a solid line. The list will be repeated for every four contour levels. A title is defined and the contour plot was generated:

```
u: READ MYDATA
u: LISTHDR
s: FILE: MYDATA
s:      NPTS = 10000
s:    IFTYPE = GENERAL XYZ (3-D) FILE
s:    DEPMIN = 697.71
s:    DEPMAX = 1154.4
s:    NXSIZE = 100
s:  XMINIMUM = 82574.
```

```
s:  XMAXIMUM = 86992.
s:    NYSIZE = 100
s:  YMINIMUM = 0.47439E+06
s:  YMAXIMUM = 0.47720E+06
u: ZLEVELS RANGE 700 1150 INCREMENT 25
u: ZLINES LIST 1 2 3 4
u: TITLE 'Katmai topography from survey data [inc = 25 km]'
u: CONTOUR
```

The result of this example is shown in the figure below.

EXAMPLES (cont.): In the final example, the same data is used but different display options are selected. Integer labels are selected for every fourth contour level and "down" tick marks are selected for the contour levels in between. Solid linestyles are used for all contour levels:

```
u: READ MYDATA
u: ZLEVELS RANGE 700 1150 INCREMENT 25
u: ZLABELS ON LIST INT OFF OFF OFF
u: ZTICKS ON LIST 0 -1 -1 -1
u: ZLINES LIST 1
u: TITLE 'Katmai topography from survey data [labels and ticks]'
u: CONTOUR
```

The result of this example is shown in the figure below.

HEADER VARIABLES

REQUIRED: IFTYPE (set to "IXYZ"), NXSIZE, NYSIZE
USED: XMINIMUM, XMAXIMUM, YMINIMUM, YMAXIMUM

SEE COMMANDS

ZCOLORS, ZLABELS, ZLEVELS, ZLINES, ZTICKS, SPECTROGRAM and the SAC User's Manual section on Writing SAC Data Files.

ACKNOWLEDGEMENTS

The fast scan contouring subroutine was developed by Dave Harris (DBH).

LATEST REVISION

JULY 22, 1991 (Version 10.6d)

CONVERT

SUMMARY

Converts data files from one format to another.

SYNTAX

`CONVERT {FROM} {format} infile`

`{TO {format} outfile}|{OVER {format}}` where format is one of the following:

`SAC|ALPHA`

INPUT

infile: The name of the input data file.
outfile: The name of the output data file.
OVER: Overwrite the input data file.
SAC: SAC formatted binary data file.
ALPHA: Alphanumeric equivalent of SAC binary data file.

DEFAULT VALUES

`CONVERT FROM SAC infile OVER SAC`

DESCRIPTION

This command converts a single data file from one format to another. In the previous version of this help file written in 1983, it was stated that convert would be replaced in the future by improved capability in READ and WRITE. In 2011, CONVERT is no longer needed, but for back-compatibility it is being kept.

LATEST REVISION

August 2011

CONVOLVE

SUMMARY

Compute the convolution of a master signal with itself and with all other signals in memory.

SYNTAX

```
CONVOLVE  {MASTER name|n},   {NUMBER n},{LENGTH ON|OFF|v},
{TYPE  RECTANGLE|HAMMING|HANNING|COSINE|TRIANGLE}
```

INPUT

MASTER name|n: Select master file in data file list by name or number. All files will be cornvolved with the MASTER.

NUMBER n: Set number of correlation windows to be used.

LENGTH {ON}: Turn fixed window length option on.

LENGTH OFF: Turn fixed window length option off.

LENGTH v: Turn fixed window length option on and change window length in seconds to v.

TYPE RECTANGLE: Apply a rectangle function to each window. This is equivalent to applying no function to each window.

TYPE HAMMING: Apply a hamming function to each window.

TYPE HANNING: Apply a hanning function to each window.

TYPE COSINE: Apply a cosine function to each window.

TYPE TRIANGLE: Apply a triangle function to each window.

DEFAULT VALUES

```
CONVOLVE MASTER 1 NUMBER 1 LENGTH OFF TYPE RECTANGLE
```

DESCRIPTION

If there are N SAC files in memory, there are N output files that are the results of the convolution with MASTER. There is no normalization. All signals in memory must have the same DELTA.

The algorithm assumes all time series are causal, so if one wants to convolve signals with a boxcar (as a method of non-causal,low-pass filtering perhaps to smooth out spikes in a synthetic waveform), the output signals will be time-shifted by half the width of the boxcar.

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

ACKNOWLEDGEMENTS

This command is based on an algorithm developed by Dave Harris (DBH).

LATEST REVISION

Dec. 4, 1996 (Version 52a)

COPYHDR

SUMMARY

Copies header variables from one file in memory to all others.

SYNTAX

```
COPYHDR {FROM name|n} hdrlist
```

INPUT

FROM name: Copy header list from named file in memory.
FROM n: Copy header list from numbered file in memory.
hdrlist: Space delimited list of header variables to copy.

DEFAULT VALUES

```
COPYHDR FROM 1
```

DESCRIPTION

This command lets you copy the values of any SAC header variable from one file in memory to all of the remaining files in memory. You can select which file you want to copy from.

EXAMPLES

Assume you are using PPK to mark several times in the header of a file called FILE1. You are using the header variables T3 and T4. To copy those same markers into files FILE2 and FILE3:

```
u:  READ FILE1
u:  PPK
u:  ... use cursor to mark times T3 and T4.
u:  READ MORE FILE2 FILE3
u:  COPYHDR FROM 1 T3 T4
```

In this next example, assume you have read in a large number of files and you want to copy the event location, EVLA and EVLO, from the file called ABC into all of the other headers. This can be easily done by referencing the file by name not number:

```
u:  COPYHDR FROM ABC STLA STLO
```

HEADER CHANGES

Potentially all.

LATEST REVISION

May 15, 1987 (Version 10.2)

CORRELATE

SUMMARY

Computes the auto- and cross- correlation functions.

SYNTAX

```
CORRELATE  {MASTER name|n},
{NUMBER n},{LENGTH ON|OFF|v},
{TYPE  RECTANGLE|HAMMING|HANNING|COSINE|TRIANGLE}
```

INPUT

MASTER name|n: Select master file in data file list by name or number. All files will be correlated against this one.

NUMBER n: Set number of correlation windows to be used.

NORMALIZED: Results are normalized between -1.0 and 1.0

LENGTH {ON}: Turn fixed window length option on.

LENGTH OFF: Turn fixed window length option off.

LENGTH v: Turn fixed window length option on and change window length in seconds to v.

TYPE RECTANGLE: Apply a rectangle function to each window. This is equivalent to applying no function to each window.

TYPE HAMMING: Apply a hamming function to each window.

TYPE HANNING: Apply a hanning function to each window.

TYPE COSINE: Apply a cosine function to each window.

TYPE TRIANGLE: Apply a triangle function to each window.

DEFAULT VALUES

```
CORRELATE MASTER 1 NUMBER 1 LENGTH OFF TYPE RECTANGLE
```

DESCRIPTION

An auto-correlation function is computed on the signal which you declare to be the master one, and a cross-correlation function is calculated between it and each of the other signals in memory.

The windowing features of this command allow you to compute an average correlation function over a set of data windows. The number of windows is selectable and there are five standard windowing functions to choose from. When this windowing feature is on, a cross-correlation function is computed for each window. This collection of cross-correlation functions is then averaged, cut to the same length as the original data file, and replaces the data file in memory. You may also select the length of each window.

Window overlap is automatically calculated and used whenever the product of the requested window length (LENGTH option) and the number of windows (NUMBER option) exceeds the number of points in the data file (NPTS). By default, this windowing feature is off.

EXAMPLES

To calculate the correlation functions using the third file in memory as the master file:

```
u:  CORRELATE MASTER 3
```

You could also specify the master file by name if this is easier. Assume you have two data files that each contain 1000 points of noise. To compute the average correlation functions using 10 windows of 100 points each (i.e. no overlap) with a hanning function applied to each window:

```
u:  CORRELATE TYPE HANNING NUMBER 10
```

To achieve a twenty percent overlap of each window, set the window length to the equivalent of 120 data points. Assuming a sampling interval of 0.025 (40 samples per second) this would be three seconds as shown below:

```
u:  CORRELATE TYPE HANNING NUMBER 10 LENGTH 3.0
```

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

ACKNOWLEDGEMENTS

This command is based on an algorithm developed by Dave Harris (DBH)

LATEST REVISION

Dec. 4, 1996 (Version 52a)

COMMIT , RECALLTRACE , ROLLBACK

SUMMARY

ROLLBACK: reverts SAC data to last committed version in I/O buffers. COMMIT: commits (copies) SAC data to the I/O buffers. RECALLTRACE: rolls back the last committed waveform and a few header fields, commits most of the header fields.

SYNTAX

```
ROLLBACK
COMMIT
RECALLTRACE (or simply RECALL)
```

DESCRIPTION

Context: In order to support multiple data formats with as little information loss as possible, SAC's internal data storage has been augmented with I/O buffers based on the CSS 3.0 schema. Computations continue to be performed on the data stored in the original SAC-format headers, but most I/O takes place using the data stored in the I/O buffers. Because there are two copies of the data in memory, and because most SAC commands do not affect the copy stored in the I/O buffers, SAC can revert to the copy in the I/O buffer to effectively erase unwanted changes without having to re-read the data.

ROLLBACK

After a series of operations on the data you can issue the ROLLBACK command, and the operations will be undone; the datafiles in SAC's internal data storage are replaced with the corresponding files in the I/O buffer, which represent the last committed version of the data files.

COMMIT

After a series of operations on the data you can issue the COMMIT command, any changes to the header values and the waveforms will be copied from the SAC headers to the I/O buffers. Future ROLLBACK commands will revert to this committed data.

RECALLTRACE

The RECALLTRACE command:

- rolls back the waveforms
- rolls back those header variables which are tightly linked to the waveforms
- commits those variables which are loosely linked to the waveforms.

This allows the user the flexibility to read a file, filter it, process it, and change some of the header variables not tightly linked to the waveform (make picks, establish new event location, etc.) and then recall the original waveform while saving the new header values. The user need not worry about ending up with header variables which are inconsistent with the waveform, because RECALLTRACE will roll those variables back with the waveform.

The following header variables are considered tightly linked to the waveform: DELTA ODELTA DEPMIN DEPMAX DEPMEN STLA STLO STEL STDP CMPAZ CMPINC XMINIMUM XMAXIMUM YMINIMUM YMAXIMUM NPTS NXSIZE NYSIZE NVHDR NORID NEVID NWFID IFTYPE IDEP IINST ISTREG IZTYPE ISYNTH LEVEN LPSPOL KSTNM KINST KCMPNM KHOLE KNETWK

The following header variables are considered loosely linked to the waveform: B E O A T0 T1 T2 T3 T4 T5 T6 T7 T8 T9 F EVLA EVLO EVEL EVDP MAG USER0 USER1 USER2 USER3 USER4 USER5 USER6 USER7 USER8 USER9 DIST AZ BAZ GCARC SCALE RESP0 RESP1 RESP2 RESP3 RESP4 RESP5 RESP6 RESP7 RESP8 RESP9 NZYEAR NZJDAY NZHOUR NZMIN NZSEC NZMSEC IEVTYP IMAGTYP IMAGSRC IEVREG IQUAL LOVROK LCALDA KDATRD KEVNM KO KA KT0 KT1 KT2 KT3 KT4 KT5 KT6 KT7 KT8 KT9 KF KUSER0 KUSER1 KUSER2

EXAMPLES

The COMMIT command is used on the 14th line of the following example. Run the example three times: the second time use ROLLBACK in place of COMMIT. The third time, use RECALLTRACE:

```
u: FG SEIS
u: RTR
u: P1
u: LH KSTNM KEVNM
u: CH KSTNM KAH KEVNM SOMEEVENT
u: ENVELOPE
u: PPK
u: user picks T1 at the change in frequency content
u: P1
u: LH KSTNM KEVNM
u: COMMIT # replace with ROLLBACK or RECALLTRACE as appropriate
u: P1
u: LH KSTNM KEVNM
```

RESULTS

When running this example with the COMMIT command, the final P1 will display the enveloped data and the T1 pick; the LH will display the new values for KSTNM and KEVNM which were set with the CH command.

When running this example with the ROLLBACK command, the final P1 will display the unprocessed file produced by the FG command and the T1 pick will be lost; the LH will display the original values of KSTNM and KEVNM.

When running this example with the RECALLTRACE command, the final P1 will display the unprocessed file produced by the FG command but the T1 pick will be preserved; the LH will display the original value of KSTNM and the new value of KEVNM.

SEE COMMANDS

DATAGEN, DELETECHANNEL, DELETESTACK, MERGE, READ, READALPHA, READCSS, READGSE, READHDR, READSDD, READSUDS, SORT, WRITE, WRITECSS, WRITEHDR, WRITESP, WRITESTACK

WARNING

Certain SAC commands will automatically commit your data for you. Because there are now two data storage locations, certain SAC commands will require that the two sets of files be made consitent with each other prior to executing the command. The following commands require consistency every time they are called,

MERGE SORT WRITE WRITECSS WRITEHDR WRITESP WRITESTACK

The following commands require consistency when the MORE option is specified,

DATAGEN READ READALPHA READCSS READGSE READHDR READSDD READSUDS

These commands will by default commit the data before executing. Each of these commands takes options to allow it to rollback or recall the data prior to execution. The options are COMMIT, ROLLBACK, and RECALLTRACE. COMMIT is the default.

Changing the option in any one of these commands changes it in all of them for future calls.

Note: Because there are now two data storage locations, we optimized flexibility of the DELETECHANNEL and DELETESTACK commands by allowing the COMMIT option to control whether the datafiles are deleted from the I/O buffers. When either of these delete commands are used with COMMIT ON, the specified data files are deleted from the I/O buffers as well as the SAC internal data storage; subsequent ROLLBACK commands will find no trace of the deleted files. When one of these delete commands is issued with COMMIT OFF, the specified datafiles are deleted from the SAC internal data storage, but not from the I/O buffers; subsequent calls to ROLLBACK will return these files to the SAC internal data storage. There is an exception: when DELETECHANNEL is called with the ALL option, all datafiles will be deleted from SAC internal storage and the I/O buffers regardless of the COMMIT option. The default is COMMIT OFF. See DELETECHANNEL and DELETESTACK for details.

Changing the COMMIT option in one delete command changes it in both for future calls. The COMMIT option for DELETECHANNEL and DELETESTACK is unrelated to the COMMIT option in other commands listed in the Warning above.

LATEST REVISION

October 29, 1998 (Version 00.58beta)

CUT

SUMMARY

Defines how much of a data file is to be read. CUT does not act on data currently in memory.

SYNTAX

```
CUT {ON|OFF|pdw|SIGNAL}
```

INPUT

ON: Turn cut option on but don't change pdw.

OFF: Turn cut option off.

pdw: Turn cut option on and enter/change pdw. A pdw is a partial data window. It consists of a starting and a stopping value of the independent variable, usually time, which defines which segment of a file one wishes to read. The most general form of a pdw is :ref offset ref offset:, where

ref: A number or a reference value that is one of the following: B|E|O|A|F|Tn, where n=0,1...9, and N, the number of points. The reference values are defined in'SAC data file format'_ and reviewed below.

offset: A positive or negative number which is added to the reference value.

SIGNAL: Equivalent to typing: A -1 F +1.

DEFAULT VALUES

`CUT OFF` (equivalent to CUT b e)

DESCRIPTION

The CUT command simply sets cut points and does not change the file in memory. For the command to take effect, CUT must be followed by a READ. This is in contrast with command CUTIM, which carries out cut (or cuts) on the data currently in memory.

If the start or stop offset is omitted it is assumed to be zero. If the start reference value is omitted it is assumed to be zero (in contrast with CUTIM, for which the reference is B). If the stop reference value is omitted it is assumed to be the same as the start reference value.

With CUT off, the entire file is read. With CUT is on, only that portion of the file between the starting and stopping cut values is read. These are values in terms of the independent variable in the data file, normally time. (See the SAC data file format for a discussion of dependent and independent variables.) The following header variables are used to represent certain values of the independent variable:

B: Disk file beginning value;

E: Disk file ending value;

O: Event origin time;

A: First arrival time;

F: Signal end time;

Tn: User defined time picks, n = 0,1...9

B and E are required for each data file in memory. O, A, F, and Tn can be defined for a data file in memory using the CHNHDR command. If one want to select the same time window from a group of data files that have different reference times, one must use the SYNCHRONIZE command before executing the CUT command. SYNCHRONIZE modifies the headers so that each file has the same reference time. It also adjusts all of the relative times, including B and E. Then when the files are cut, they will have the same time reference values. Since CUT is applied to the headers on disk, you must use the WRITEHDR command after the SYNCHRONIZE command and before the READ command to get the correct set of files before applying CUT.

For CUT (but not for CUTIM) an option for the stop value is to enter N, which is the offset in the number of points from the start reverence value.

EXAMPLES

The macro below demonstrates several possible uses of CUT. It is suggested that on run this macro and compare the results with those fro the macro in the help file for CUTIM:

```
fg seismo
wrie seismo.sac
echo on
 no cutting
lh b e a kztime
read seismo.sac
 begin to end---same as not cutting.
cut B E
read
lh b e a kztime
read seismo.sac
 First 3 secs of the file
cut B 0 3
read
lh b e a kztime
read seismo.sac
 First 100 points of the file.
cut B N 100
read
lh b e a delta kztime
read seismo.sac
 From 0.5 secs before to 3 secs after first arrival
cut A -0.5 3
read
lh b e a kztime
read seismo.sac
 From 19 to 15 secs relative to zero (DIFFERENT FROM CUTIM).
cut 10 15
read
lh b e a kztime
read seismo.sac
 First 3 secs of the file and next 3 sec
cut b 0 3
read
write tmp.1
read seismo.sac
cut b 3 6
read
```

```
write tmp.2
cut off
read tmp.?
lh b e a kztime
p1
```

one can also pad the beginning or end of a file with zeros by turning on the FILLZ option in the CUTERR command, defining a cut that extends beyond the current limits of the file, and then reading the file into memory using the READ command:

```
SAC> r N11A.lhz
SAC> lh npts
FILE: N11A.lhz - 1
npts = 3101
SAC> cuterr fillz; cut b n 4096
SAC> r
SAC> lh npts
FILE: N11A.lhz - 1
npts = 4096
SAC>
```

ERROR MESSAGES

- 1322: Undefined starting cut for file
  - undefined reference value in the header record.
  - this error can be controlled by use of CUTERR command.
  - when this error is off, the disk begin value is used.
- 1323: Undefined stop cut for file
  - undefined reference value in the header record.
  - this error can be controlled by use of CUTERR command.
  - when this error is off, the disk end value is used.
- 1324: Start cut less than file begin for file
  - bad CUT parameters.
  - this error can be controlled by use of CUTERR command.
  - when this error is off, the disk begin value is used or zeros are insertedat the beginning of the data.
- 1325: Stop cut greater than file end for file
  - bad CUT parameters.
  - this error can be controlled by use of CUTERR command.
  - when this error is off, the disk end value is used or zeros are inserted at the end of the data.
- 1326: Start cut greater than file end for file
  - bad CUT parameters.
  - this error cannot be turned off.

  SPECIAL NOTE: Since this is a parameter-setting command, the above errors will not appear until the READ command is executed. Also, some of the above errors can be converted to warnings by the use of the CUTERR command.

LIMITATIONS

There is currently no provision for cutting unevenly-spaced files or spectral files.

SEE COMMANDS

READ, APK, PLOTPK, SYNCHRONIZE, CUTERR

LATEST REVISION

April 21, 2010 (Version 101.4)

CUTERR

SUMMARY

Controls errors due to bad cut parameters.

SYNTAX

```
CUTERR FATAL|USEBE|FILLZ
```

INPUT

FATAL: Treat cut errors as fatal.

USEBE: Replace bad start cut with file begin and bad stop cut with file end.

FILLZ: Fill with zeros before file begin or after file end to account for difference between bad cut and file begin and end.

DEFAULT VALUES

```
FILLZ for signal stacking subprocess, USEBE for others.
```

DESCRIPTION

This command controls error conditions arising from bad cut parameters. These error conditions can be defined as fatal errors (FATAL). If the starting or ending cut parameter is undefined in the header record the disk file beginning or ending value can be chosen (USEBE). If the cut parameter is defined but the resulting cut value is less than the disk beginning value or greater than the disk ending value, the beginning or ending values can be used, or enough zeros can be added before or after the actual data to make up the difference (FILLZ).

EXAMPLES

Assume that FILE1 has a begin time, B, of 25 seconds, a first arrival, A, of 40 seconds, and a sampling rate of 100 samples per second. Assume the following commands were typed:

```
u:  CUT A -20 E
u:  READ FILE1
```

The starting cut evaluates to 20 seconds and generates a bad cut error condition. In the USEBE mode, the starting value would become 25 seconds (B). In the FILLZ mode, 500 zeros (5 seconds at 100 samples per second) would be inserted before the data and the starting value would remain as 20 seconds.

SEE COMMANDS

CUT, READ

LATEST REVISION

January 8, 1983 (Version 8.0)

CUTIM

SUMMARY

Cuts files in memory. Can cut multiple segments from each file currently in memory

SYNTAX

```
CUTIM pdw [pwd ...  ]
```

INPUT

pdw: Partial Data Window. It consists of a starting and a stopping value of the independent variable (usually time), which defines which segment of a file (or files) one wishes to read. The most general form of a pde is :ref offset ref offset:, where

ref: A reference value that is one of the following: B|E|O|A|F|Tn, where n=0,1...9. These reference values are defined in SAC data file format and reviewed below.

offset: A positive or negative number which is added to the reference value (optional).

DEFAULT VALUES

Start and stop reference values are required. See examples below for an exception. If the start or stop offset is omitted, it is assumed to be zero.

DESCRIPTION

While the CUT command simply sets cut points and does not change the file in memory, CUTIM carries out the cut(s) when the command is given. The user can READ a file aand type CUTIM with the desired cutpoints, and SAC will cut the file to those specified cutpoints. CUTIM allows multiple pairs of cutpoints, with an output file for each pair. If there are more than one file in memory, CUTIM produces the cuts on all the files For example, the user can READ three files into SAC, and use CUTIM with four sets of cutpoints; the result will be 12 files in memory.

The start and stop values are given in terms of the independent variable in the data file, normally time. (See the SAC data file format for a discussion of dependent and independent variables.) Unlike CUT, the N option (point number in file) is not available for CUTIM. The following header variables are used to represent certain values of the independent variable:

B: Disk file beginning value;

E: Disk file ending value;

O: Event origin time;

A: First arrival time;

F: Signal end time;

Tn: User defined time picks, n = 0,1...9

B and E are required for each data file in memory. O, A, F, and Tn can be defined for a data file in memory using the CHNHDR command. If one want to select the same time window from a group of data files that have different reference times, one must use the SYNCHRONIZE command before executing the CUTIM command. SYNCHRONIZE modifies the headers so that each file has the same reference time. It also adjusts all of the relative times, including B and E. Then when the files are cut, they will have the same time reference values.

EXAMPLES

The macro below demonstrates several possible uses of CUTIM on a single file. In these examples, time is the independent variable and seconds are the units.:

```
fg seismo
echo on
 no cutting
lh b e a kztime
 begin to end---same as not cutting.
cutim B E
lh b e a kztime
fg seismo
 First 3 secs of the file.
cutim B 0 3
lh b e a kztime
fg seismo
 From 0.5 secs before to 3 secs after first arrival
cutim A -0.5 3
lh b e a kztime
fg seismo
 From 0.5 to 5 secs relative to disk file start.
cutim 0.5 5
lh b e a kztime
fg seismo
 First 3 secs of the file and next 3 sec
cutim b 0 3 b 3 6
lh b e a kztime
p1
```

Because CUTIM changes the file(s) in memory, the fg seismo is rquired betweem cals to CUTIM. The lh (LISTHDR) command for seismo with no calls to CUTIM is:

```
b = 9.459999e+00                        e = 1.945000e+01
a = 1.046400e+01                  kztime = 10:38:14.000
```

Note that B is nonzero. The output for CUTI 0.5 5 does not have a reference value. The numbers are relative to b on input, but relative to zero on output. (If there is no stop reference value, it is assumed to be the same as the start reverence value, in this case B.):

```
b = 5.000000e-01                        e = 5.000000e+00
a = 1.004001e+00                  kztime = 10:38:23.460
```

The final lh produces values for the two files:

```
FILE: SEISMOGR - 1
b = 0.000000e+00                        e = 3.000000e+00
a = 1.004001e+00                  kztime = 10:38:23.460

FILE: CDV.CHAN.1981088103823 - 2
b = 3.000000e+00                        e = 6.000000e+00
a = 1.004001e+00                  kztime = 10:38:23.460
```

Note that B for cutim b 0 3 is now zero, all other times have ben altered so that arrivals (such as A) will have the same clock time. The call to p1 (PLOT1) will plot both segmnets in emory. Alternatively, a call p (PLOT) will plt the two segments individually, separated by a waiting prompt.

ERROR MESSAGES

- 1322: Undefined starting cut for file
  - undefined reference value in the header record.
  - this error can be controlled by use of CUTERR command.
  - when this error is off, the disk begin value is used.
- 1323: Undefined stop cut for file
  - undefined reference value in the header record.
  - this error can be controlled by use of CUTERR command.
  - when this error is off, the disk end value is used.
- 1324: Start cut less than file begin for file
  - bad CUT parameters.
  - this error can be controlled by use of CUTERR command.
  - when this error is off, the disk begin value is used or zeros are inserted at the beginning of the data.
- 1325: Stop cut greater than file end for file
  - bad CUT parameters.
  - this error can be controlled by use of CUTERR command.
  - when this error is off, the disk end value is used or zeros are inserted at the end of the data.
- 1326: Start cut greater than file end for file
  - bad CUT parameters.
  - this error cannot be turned off.

SPECIAL NOTE Also, some of the above errors can be converted to warnings by the use of the CUTERR command.

LIMITATIONS

There is currently no provision for cutting unevenly-spaced files or spectral files.

SEE COMMANDS

CUT, READ, APK, PLOTPK, SYNCHRONIZE, CUTERR

LATEST REVISION

October 2013 (Version 101.6a)

DATAGEN

SUMMARY

Generates sample data files and stores them in memory.

SYNTAX

```
DATAGEN {MORE} {COMMIT|ROLLBACK|RECALLTRACE} {SUB name} {filelist}
```

where name is one of the following:

```
LOCAL
REGIONAL
TELESEISEM
```

INPUT

MORE: Place the new sample data files in memory AFTER the old data. If this option is omitted, the new sample data files REPLACE the old ones.

Note: if the MORE option is not specified, the COMMIT, ROLLBACK, and RECALLTRACE options have no effect.

COMMIT: If the MORE option is specified, the COMMIT option commits headers and waveforms in SAC memory -- removing any previous versions of headers or waveforms from RAM -- prior to generating more files. COMMIT is the default.

ROLLBACK: If the MORE option is specified, the ROLLBACK option reverts to the last committed version of the header and waveform before generating more files.

RECALLTRACE: If the MORE option is specified, the RECALLTRACE option:

- reverts to the last committed version of the waveform,
- reverts to the last committed version of those header variables closely linked to the waveform,
- commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

SUB name: Select the sub-directory name from which to read the data. Where the sub-directory name is local, regional, or teleseismic.
A filelist is required. Possible filenames are listed below.

name: LOCAL|REGIONAL|TELESEIS ,BREAK Specifics about the contents of these sub-directories is given below.

filelist: A list of SAC sample data files. The names of the files in each subdirectory are given below. This list may contain simple filenames and wildcard characters. See the READ and WILD commands for a complete description.

DEFAULT VALUES

```
DATAGEN COMMIT SUB LOCAL cdv.z
```

DESCRIPTION

This command reads one or more files from a set of sample seismograms provided with the program. In fact, it operates much like the READ command except that it gets the data from a special data directory. You specify the name of the sub-directory containing the event of interest. Each sub-directory contains a different type of event. You can use wildcards just like the READ command to specify groups of files in the sub-directory.

ERROR MESSAGES

- 1301: No data files read in.
  - haven't given a list of files to read.
  - none of the files in the list could be read.
- 1314: Data file list can't begin with a number.
- 1315: Maximum number of files in data file list is

WARNING MESSAGES

- 0101: opening file
- 0108: File does not exist:
- 0114: reading file
  - Normally when SAC encounters one of these errors it skips that file and reads the remainder. These errors can be made to be fatal using the READERR command.

LOCAL EVENT

The local event occurred in the Livermore Valley of California. It was a small unfelt event (ML 1.6). It was recorded by the Livermore Local Seismic Network (LLSN). LLSN is a set of vertical and three-component stations operated by LLNL and the USGS. Data from nine three-component stations are included in this set. There is 40 seconds of data sampled at 100 samples per second. Station information, event information, p-wave time picks, and coda picks are included in the headers. The filenames are:

```
cal.z, cal.n, cal.e
cao.z, cao.n, cao.e
cda.z, cda.n, cda.e
cdv.z, cdv.n, cdv.e
cmn.z, cmn.n, cmn.e
cps.z, cps.n, cps.e
cva.z, cva.n, cva.e
cvl.z, cvl.n, cvl.e
cvy.z, cvy.n, cvy.e
```

REGIONAL EVENT

The regional event occurred in Nevada and was recorded by the Digital Seismic Network (DSS). DSS is a set of four broadband three-component stations in the Western U.S. The stations are:

elk: Elko, NV
lac: Landers, CA
knb: Kanab, UT
mnv: Mina, NV

The sampling rate is 40 samples per second. The files contain 300 seconds of data, starting 5 seconds before the origin time of the event. The filenames are:

```
elk.z, elk.n, elk.e
lac.z, lac.n, lac.e
knb.z, knb.n, knb.e
mnv.z, mnv.n, mnv.e
```

TELESEISMIC EVENT

The teleseismic event occurred off the coast of Northern California near Eureka on September 10, 1984. It was a moderate to large event (ML 6.6, MB 6.1, MS 6.7) and was felt from the San Francisco Bay area to Roseburg, Oregon. It was recorded at the Regional Seismic Test Network (RSTN), a set of five stations in the U.S. and Canada. The stations are:

cpk: Tennessee
ntk: Northwest Territories, Canada
nyk: New York
onk: Ontario, Canada
sdk: South Dakota

Both mid-period and long period data is included. Data from cpk was not available and the long-period data from sdk is clipped. There is 1600 seconds of data in this set. The long-period data was recorded at 1 sample per second and the mid-period data at 4 samples per second. The filenames are:

```
ntkl.z, ntkl.n, ntkl.e, ntkm.z, ntkm.n, ntkm.e
nykl.z, nykl.n, nykl.e, nykm.z, nykm.n, nykm.e
onkl.z, onkl.n, onkl.e, onkm.z, onkm.n, onkm.e
sdkl.z, sdkl.n, sdkl.e, sdkm.z, sdkm.n, sdkm.e
```

LATEST REVISION

August 2011 (version 101.5)

DECIMATE

SUMMARY

Decimates (downsamples) data, including an optional anti-aliasing FIR filter.

SYNTAX

```
DECIMATE {n},{FILTER {ON|OFF}}
```

INPUT

n: Set decimation factor to n. Range is 2 to 7. This command may be applied several times if a larger decimation factor is required.

FILTER {ON}: Turn anti-aliasing FIR filter on.

FILTER OFF: Turn anti-aliasing FIR filter off.

DEFAULT VALUES

```
DECIMATE 2 filter on
```

DESCRIPTION

This command is used to downsample data after it has been read into memory. An optional finite impulse response (FIR) filter is applied to the data as it is being decimated to prevent aliasing effects normally associated with downsampling digitized analog signals. These filters also preserve the phase information. The application of these FIR filters often produces undesirable transients at each end of the data so the results should be checked graphically. Turning the anti-aliasing filter option off should only be done when the accuracy of the high frequency response is unimportant, such as when plotting.

EXAMPLES

To reduce the sampling rate by a factor of 42:

```
u:  READ FILE1
u:  DECIMATE 7
u:  DECIMATE 6
```

HEADER CHANGES

NPTS, DELTA, E, DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES

- 1003: Value out of allowed range at symbol
  - Range on decimation factor is 2 to 7.
- 1301: No data files read in.

- 1306: Illegal operation on unevenly spaced file
- 1307: Illegal operation on spectral file

Note The decimation by 7 filter has occasionally been unstable.

LATEST REVISION

May 15, 1987 (Version 10.2)

DELETECHANNEL

SUMMARY

Deletes one or more files from the file list.

SYNTAX

```
[D]ELETE[C]HANNEL ALL
[D]ELETE[C]HANNEL filename|filenumber|range {filename|filenumber|range
...  }
```

INPUT

ALL: Deletes all files from memory. The user need not specify filenames or filenumbers

filename: Name of a file in the file list.

filenumber: Number of a specific file in the file list. The first file in the list is 1, the second is 2, etc. (The command FILENUMBER ON tells SAC to display the file numbers in most of the plots.)

range: Two file numbers separated by a dash: eg. 11-20.

TYPE: Action-taking

EXAMPLES

```
dc 3 5                        * deletes 3rd and 5th file.
dc SO01.sz SO02.sz            * deletes named files.
dc 11-20                      * deletes all the files from
                              * the 11th through the 20th,
                              * inclusive.
dc 3 5 11-20 SO01.sz SO02.sz  * deletes all of the above.
```

ERROR MESSAGES

- 5106: File name not in file list
- 5107: File number not in file list

SEE COMMANDS

DELETESTACK, FILENUMBER

DIF

SUMMARY

Differentiates data in memory.

SYNTAX

```
DIF {TWO|THREE|FIVE}
```

INPUT

TWO: Apply a two point difference operator.
THREE: Apply a three point difference operator.
FIVE: Apply a five point difference operator.

DEFAULT VALUES

```
DIF TWO
```

DESCRIPTION

The two-point algorithm is:

```
OUT(J)= (DATA(J+1) - DATA(J))  / DELTA
```

The last output point is not defined by this algorithm. It is also not a centered algorithm. SAC takes care of these problems by decreasing the number of points in the file (NPTS) by one and by increasing the begin time (B) by half the sampling interval (DELTA).

The three-point (centered two-point) algorithm is:

```
OUT(J)= 1/2 * (DATA(J+1) - DATA(J-1)) / DELTA
```

The first and last output point is not defined by this algorithm. SAC decreases NPTS by 2 and increases B by DELTA.

The five-point (centered four-point) algorithm is:

```
OUT(J)=2/3 (DATA(J+1) - DATA(J-1)) / DELTA -
       1/12 * (DATA(J+2) - DATA(J-2)) / DELTA
```

The first two and last two output points are not defined by this algorithm. SAC applies the three-point operator to the second points from each end, decreases NPTS by 2, and increases B by DELTA.

ERROR MESSAGES

- 1301: No data files read in.
- 1306: Illegal operation on unevenly spaced file

HEADER CHANGES

NPTS, B, E, DEPMIN, DEPMAX, DEPMEN

LATEST REVISION

January 15, 1985 (Version 9.10)

DIV

SUMMARY

Divides each data point by a constant.

SYNTAX

```
DIV {v1 {v2 ...  vn} }
```

INPUT

v1: Constant to divide first file by.
v2: Constant to divide second file by.
vn: Constant to devide nth file by.

DEFAULT VALUES

```
DIV 1.
```

DESCRIPTION

This command will divide each element of each data file in memory by a constant. The constant may be the same or different for each data file. If there are more data files in memory than constants, then the last constant entered is used for the remainder of the data files in memory.

EXAMPLES

To divide each element of F1 by 5.1 and each element of F2 and F3 by 6.2:

```
u:  READ F1 F2 F3
u:  DIV 5.1 6.2
```

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES

- 1301: No data files read in.
- 1307: Illegal operation on spectral file
- 1701: Can't divide by zero.

LATEST REVISION

January 8, 1983 (Version 8.0)

DIVF

SUMMARY

Divides data in memory by a set of data files.

SYNTAX

```
DIVF {NEWHDR ON|OFF} filelist
```

INPUT

NEWHDR ON|OFF: By default, the resultant file will take its header field from the original file in memory. Turning NEWHDR ON, causes the header fields to be taken from the new file in the filelist.

filelist: A list of SAC binary data files. This list may contain simple filenames, full or relative pathnames, and wildcard characters. See the READ command for a complete description.

DESCRIPTION

This command can be used to divide a set of files by a single file or by another set of files. An example of each case is presented below. The files must be evenly spaced and should have the same sampling interval and number of points. This last two restrictions can be eliminated using the BINOPERR command. If there are more data files in memory than in the filelist, then the last file in the filelist is used for the remainder of the data files in memory.

EXAMPLES

To divide three files by a single file:

```
u:  READ FILE1 FILE2 FILE3
u:  DIVF FILE4
```

To divide two files by two other files:

```
u:  READ FILE1 FILE2
u:  DIVF FILE3 FILE4
```

HEADER CHANGES

If NEWHDR is OFF (the default) the headers in memory are unchanged). If NEWHDR is ON, the headers are replaced with the headers from the files in the filelist.

DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES

- 1301: No data files read in.
- 1803: No binary data files read in.
- 1307: Illegal operation on spectral file

- 1306: Illegal operation on unevenly spaced file
- 1801: Header field mismatch: - either the sampling interval or the number of points are not equal. - can be controlled using the BINOPERR command.

WARNING MESSAGES

- 1802: Time overlap:
  - the file division is still performed.

SEE COMMANDS

READ, BINOPERR

LATEST REVISION

May 26, 1999 (Version 0.58)

DIVOMEGA

SUMMARY

Performs integration in the frequency domain.

DESCRIPTION

This command divides each point of a spectral file by its frequency given by:

```
OMEGA = 2.0 * PI * FREQ
```

This is analogous to integrating the equivalent time series file. The spectral file can in either amplitude-phase or real-imaginary format. This is often convenient with normal data but is critical when obtaining the FFT of data whose spectra ranges over many orders of magnitude. For example, suppose you have prewhitened a data file by using the DIF command, and then taken its transform using the FFT command. The effect of the differentiation in time domain can be removed by an integration in the frequency domain using this command.

EXAMPLES

The steps discussed above are shown in this example:

```
u:  READ FILE1
u:  DIF
u:  FFT AMPH
u:  DIVOMEGA
```

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

SEE COMMANDS

DIF, FFT, MULOMEGA, WRITESP, READSP

LATEST REVISION

May 15, 1987 (Version 10.2)

ECHO

SUMMARY

Controls echoing of input and output to the terminal.

SYNTAX

`ECHO ON|OFF list` where list is one or more of the following:

ERRORS

WARNINGS

OUTPUT

COMMANDS

MACROS

PROCESSED

INPUT

ON: Turn on echoing of the items in the list that follows.

OFF: Turn off echoing of the items in the list that follows.

ERRORS: Error messages generated during the execution of a command.

WARNINGS: Warning messages generated during the execution of a command.

OUTPUT: Output messages generated during the execution of a command.

COMMANDS: Raw commands as they were typed at the terminal.

MACROS: Raw commands as they appears in a macro file.

PROCESSED: Processed commands originating from the terminal or a macro file. A processed command is one where all macro arguments, blackboard variables, header variables, and inline functions have been processed (evaluated) and substituted into the raw command.

DEFAULT VALUES

ECHO ON ERRORS WARNINGS OUTPUT OFF COMMANDS MACROS PROCESSED

DESCRIPTION

This commands lets you control which categories of the SAC input and output stream is to be echoed to the terminal or screen. There are three categories of output: error messages, warning messages, and output messages. There are three categories of input: commands typed at the terminal, commands executed from a macro, and "processed" commands. A processed command is one in which all macro arguments, blackboard variables, header variables, and inline functions have been evaluated. You can control the echoing of these categories individually. When you type a command at your terminal, the operating system normally echos each character. Thus the commands echoing option is of limited use for interactive sessions. The macro and processed options are useful when debugging a macro.

LATEST REVISION

April 21, 1989 (Version 10.4c)

ENDDEVICES

SUMMARY

Terminates one or more graphics devices.

SYNTAX

```
ENDDEVICES devices
```

where devices is one or more of the following:

```
SGF, XWINDOWS
```

ALTERNATE FORMS

ENDG or EG are obsolete but acceptable names for this command.

INPUT

SGF: The SAC Graphics File device driver.
XWINDOWS: The X-windows window display system.

DESCRIPTION

This command terminates one or more graphics devices. Devices are activated using the BEGINDEVICES command. The command help graphics has a description of each of these graphics devices.

SEE COMMANDS

BEGINDEVICES

LATEST REVISION

March 24, 2009 (101.3)

ENDFRAME

SUMMARY

Resumes automatic new frame actions between plots.

SYNTAX

```
ENDFRAME
```

ALTERNATE FORMS

ENDFR is an obsolete but allowable form of this command.

DESCRIPTION

See the BEGINFRAME documentation.

SEE COMMANDS

BEGINFRAME

LATEST REVISION

May 15, 1987 (Version 10.2)

ENVELOPE

SUMMARY

Computes the envelope function using a Hilbert transform.

SYNTAX

ENVELOPE

DESCRIPTION

This command computes the envelope function of the data in memory. The envelope is defined by the square root of x(n)^2 + y(n)^2, where x(n) is the original signal and y(n) its Hilbert transform (see HILBERT). As with HILBERT, very long period datashould be decimated (see DECIMATE) prior to processing.

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

SEE COMMANDS

HILBERT, DECIMATE

ACKNOWLEDGEMENT

The subroutines used to perform the Hilbert transform were designed and developed by Dave Harris (DBH).

LATEST REVISION

April 21, 1989 (Version 10.4c)

ERASE

SUMMARY

Erases the graphics display area.

SYNTAX

```
ERASE
```

DESCRIPTION

This command works only if SAC knows what graphics device you are using. This is true only if you have already done some plotting. This command is necessary for the ADM terminal which does not have an erase screen key and is useful in command files when you want the screen erased prior to sending out a large amount of text.

LATEST REVISION

October 11, 1984 (Version 9.1)

EVALUATE

SUMMARY

Evaluates simple arithmetic expressions.

SYNTAX

`EVALUATE {TO TERM|name} {v} op v {op v ...}` where op is one of the following:

`ADD|SUBTRACT|MULTIPLY|DIVIDE|POWER|`

`SQRT|EXP|ALOG|ALOG10|SIN|ASIN|`

`COS|ACOS|TAN|ATAN|`

`EQ|NE|LE|GE|LT|GT`

INPUT

TO TERM: Result is written to the user's terminal.

TO name: Result is written to the blackboard variable name.

v: An floating point or integer number. (Since all arithmetic is done in floating point, integers are converted to floating point numbers.)

op: One of the arithmetic or logical operators listed above.

ALTERNATE FORMS FOR LOGICAL OPERAATORS ADD SUBTRACT MULTIPLY DIVIDE POWER

- `-  *  /  **`

DEFAULT VALUES

```
EVALUATE TO TERM 1.  * 1.
```

DESCRIPTION

This command lets you evaluate arithmetic and logical expressions. The arithmetic expression can be a compound containing more than one operator. In this case the expression is evaluated left to right. There is no nesting capability. A logical expression can contain only one operand. The result of evaluating this expression can be written to the user's terminal or to a specified blackboard variable. This blackboard variable can later be used directly in other commands. This is especially useful when writing macros. You can also get the value of a blackboard variable using the GETBB command. Previously, there was a maximum number of operators (10) in a single command. As of v101.6, there is no maximum number.

EXAMPLES

Two simple examples:

```
SAC>  EVALUATE 2 * 3
==>  6
SAC> evaluate tan 45
==>  1.61978
```

Here is a sightly more complicated example:

```
SAC>  EVALUATE 4 * atan 1 / PI
==>  1
```

Finally let's repeat the previous example but this time use a blackboard variable:

```
SAC> evaluate to temp1 4 * atan 1
SAC> evaluate %temp1 / PI
==> 1
SAC>
```

SEE COMMANDS

GETBB SAC_MACROS

LATEST REVISION

2013 (v101.6a)

EXP

SUMMARY

Computes the exponential of each data point.

SYNTAX

`EXP`

ERROR MESSAGES

- 1301: No data files read in.
- 1307: Illegal operation on spectral file

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

LATEST REVISION

January 15, 1985 (Version 9.10)

EXP10

SUMMARY

Computes the base 10 exponential (10.**y) of each data point.

SYNTAX

```
EXP10
```

ERROR MESSAGES

- 1301: No data files read in.
- 1307: Illegal operation on spectral file

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

LATEST REVISION

January 15, 1985 (Version 9.10)

EXTERNAL COMMAND INTERFACE

SUMMARY

Description of interface for external commands callable by SAC.

DESCRIPTION

C language interface

The following definitions and structures will be used to pass data into and out of external functions specified by the user. These external commands will be loaded by SAC at run time by executing the LOAD command (See LOAD help page for further details).

The application programming interface for external functions (commands) is:

```
long ext_func(argc, argv, call_data, update)
   int argc;
   char **argv;
   sac_files *call_data;
   long *update;
```

This function should return a long to be used as an error status flag. By convention, if this function returns a non-zero value, SAC will indicate that an error occurred within this function. By default, SAC will print out the error number returned. If the user wants to add a customized error message, this can be done by editing the messages file in the SAC aux directory. Care must be taken not to use an error number that has already been used in another context.

Where argc and argv contain the command line arguments, defined the same as the command line arguments for a C program. argc is set to the number of arguments, and argv contains the tokenized command line. argc is always greater than or equal to one, since argv[0] contains the command name.

sac_files is a pointer to a call_data struct which is used to package the sac headers and data for efficient communication with the external function. This data structure is defined in the file extfunc.h, which must be included in the external function.

update is a flag which tells SAC how to handle the data returned from the external function. It should be set to one of the enumerated values, APPEND, REPLACE, or IGNORE. If this flag is set to APPEND, the data returned from the external function will be appended to the existing data file list (files in memory). If set to REPLACE, the returned data will replace the data in memory (which is the way that most SAC commands work). If set to IGNORE, it will be disregarded.

Several support routines are provided to facilitate header access. They include:

```
sac_header *makehdr( sac_header *header_in )
```

Allocate a new header struct. If header_in is not NULL, copy its values. If header_in is NULL, initialize the new header to default values:

```
long getehdr(sac_header *header, char *fieldname, long *error)
```

Return the value of the enumerated header field pointed to by fieldname from the header struct pointed to by header:

```
void setehdr(sac_header *header, char *fieldname, long value, long *error)
```

Set the enumerated field specified in fieldname to value in the header specified in header:

```
float getfhdr(sac_header *header, char *fieldname, long *error)
```

Return the value of the floating point header field fieldname from the header specified in header:

```
void setfhdr(sac_header *header, char *fieldname, float value, long *error)
```

Set the floating point field fieldname to value in header pointed to by header:

```
long getnhdr(sac_header *header, char *fieldname, long *error)
```

Return the value of the long field specified in fieldname from the header specified by header:

```
void setnhdr(sac_header *header, char *fieldname, long value, long *error)
```

Set the long header field fieldname to value in the header specified by header:

```
long getlhdr(sac_header *header, char *fieldname, long *error)
```

Return the value of the logical header field fieldname from the header specified by header:

```
void setlhdr(sac_header *header, char *fieldname, long value, long *error)
```

Set the logical header field fieldname to value in the header header:

```
char *getahdr(sac_header *header, char *fieldname, long *error)
```

Return a pointer to the value of the character header field fieldname from the header specified by header. This function returns a pointer to the actual header field. You should not modify this and also should not free this returned address:

```
void setahdr(sac_header *header, char *fieldname, char *value, long *error)
```

Set the character header field fieldname to the value pointed to by value in the header specified by header.

All header access routines return zero in the error variable if no error occurred, otherwise they return non-zero.

The file extfunc.h contains tables of the names of the various header fields which can be returned or set by the above functions. It also contains definitions of all the enumerated values known to SAC.

FORTRAN Language Interface

The FORTRAN language interface to external commands consists of a C language function which maps data into and out of a FORTRAN routine having the following interface:

```
subroutine fmycommand(fargs, fyinput, fxinput, numfiles, nptsmax, ferror)

include 'fext_params'

character*(*) fargs
real*4 fyinput(nptsmax,numfiles)
real*4 fxinput(nptsmax,numfiles)
integer*4 numfiles, nptsmax, ferror
```

Where fargs is the command line, blank delimited. fyinput contains the input y data. fyinput is zero filled for data consisting of less than nptsmax points. fxinput contains the x data for unevenly spaced files. In the case of evenly spaced data, fxinput is all zeroes. numfiles is the number of input files, nptsmax is the maximum number of points of all the input files and ferror is an error return flag.

The include file "fext_params" contains parameters defining the valid enumerated header values.

The C language function referred to above is fextern.c.template.

Several support routines are provided to facilitate header access. They include:

```
fgetahdr(integer*4 hdr_index, character fieldname, character value, integer*4 error)
```

Returns value of character header field fieldname in value. Value returned is from header(hdr_index), where hdr_index ranges from 1 to numfiles:

```
fsetahdr(integer*4 hdr_index, character fieldname, character value, integer*4 error)
```

Set the value of character header field fieldname to value:

```
fgetehdr(integer*4 hdr_index, character fieldname, integer*4 value, integer*4 error)
```

Returns the value of enumerated header field fieldname in value:

```
fsetehdr(integer*4 hdr_index, character fieldname, integer*4 value, integer*4 error)
```

Set the value of enumerated header field fieldname to value:

```
fgetfhdr(integer*4 hdr_index, character fieldname, real*4 value, integer*4 error)
```

Returns the value of real header field fieldname in value:

```
fsetfhdr(integer*4 hdr_index, character fieldname, real*4 value, integer*4 error)
```

Set the value of real header field fieldname to value:

```
fgetlhdr(integer*4 hdr_index, character fieldname, integer*4 value, integer*4 error)
```

Returns the value of logical header field fieldname in value:

```
fsetlhdr(integer*4 hdr_index, character fieldname, integer*4 value, integer*4 error)
```

Set the value of logical header field fieldname to value:

```
fgetnhdr(integer*4 hdr_index, character fieldname, integer*4 value, integer*4 error)
```

Returns the value of integer header field fieldname in value:

```
fsetnhdr(integer*4 hdr_index, character fieldname, integer*4 value, integer*4 error)
```

Set the value of integer header field fieldname to value.

FFT

SUMMARY

Performs a discrete Fourier transform.

SYNTAX

```
FFT {WOMEAN|WMEAN},{RLIM|AMPH}
```

INPUT

WOMEAN: Remove mean before transform.
WMEAN: Leave mean in transform.
RLIM: Output should be in real-imaginary format.
AMPH: Output should be in amplitude-phase format.

ALTERNATE FORMS

Purists may use DFT in place of FFT.

DEFAULT VALUES

```
FFT WMEAN AMPH
```

DESCRIPTION

Before the transform is performed, each data file is padded with zeros to the next power of two. SAC data files, on disk and in memory, can contain either time-series data or spectral data. The spectral data may be in either amplitude-phase format or real-imaginary format. The IFTYPE field in the header tells you which kind of data is stored in a particular file and what its format is. Most commands work on only one type, either time-series or spectal. Certain commands such as FFT, IFFT, UNWRAP, etc. change data in memory from one data type or format to another. The spectral files that result from this command can be plotted using the PLOTSP command or saved on disk with the WRITE or WRITESP command. If one has more than one data file in memory, PLOT2 can be used to plot the amplitudes or real part. The SAC sign convention is such that the phase for a causal function decreases with increasing frequency. This is the same convention as in program EVALRESP and RDSEED.

HEADER CHANGES

B, E, and DELTA are changed to the beginning, ending and sampling frequencies of the transform respectively. The original values of B, E, NPTS and DELTA are saved as SB, SE, NSNPTS and SDELTA and are restored if an inverse transform is done.

ERROR MESSAGES

- 1301: No data files read in.
- 1306: Illegal operation on unevenly spaced file
- 1307: Illegal operation on spectral file
- 1606: Maximum allowable DFT is

LIMITATIONS

The maximum transform is 2**24=16777216 data points.

SEE COMMANDS

PLOTSP, IFFT, WRITESP

LATEST REVISION

May 6, 2010 (Version 101.4)

FILEID

SUMMARY

Controls the file id display found on most SAC plots.

SYNTAX

```
FILEID {ON|OFF} {TYPE DEFAULT|NAME|LIST hdrlist},
LOCATION UR|UL|LR|LL},
{FORMAT EQUALS|COLONS|NONAMES}
```

INPUT

FILEID {ON}: Turn on file id option. Does not change file id type or location.
FILEID OFF: Turn off file id option.
TYPE DEFAULT: Change to the default file id.
TYPE NAME: Use the name of the file as the file id.
TYPE LIST hdrlist: Define a list of header fields to display in the fileid.
LOCATION UR: Place file id in upper right hand corner.
LOCATION UL: Place file id in upper left hand corner.
LOCATION LR: Place file id in lower right hand corner.
LOCATION LL: Place file id in lower left hand corner.
FORMAT EQUALS: Format consists of header field name, an equals sign, and the header field value.
FORMAT COLON: Format consists of header field name, a colon, and the value.
FORMAT NONAMES: Format consists of header field value only.

DEFAULT VALUES

```
FILEID ON TYPE DEFAULT LOCATION UR FORMAT NONAMES
```

DESCRIPTION

This command controls the file id that is displayed on most SAC plot formats. The file id identifies the content of the plot. The default file id consists of the event name, the station name and component, and the zero date and time. The name of the file can be substituted for the default id if desired. A special file id can be defined and displayed. This special file id can consist of up to 10 SAC header fields. The location and format of the fileid can also be changed.

EXAMPLES

To put the filename in the upper left corner:

```
u:  FILEID LOCATION UL TYPE NAME
```

To define a special file id consisting of the station component, latitude, and longitude:

```
u:  FILEID TYPE LIST KSTCMP STLA STLO
```

To include the name of the header field followed by a colon:

```
u:  FILEID FORMAT COLON
```

LATEST REVISION

October 11, 1984 (Version 9.1)

FILENUMBER

SUMMARY

Controls the file number display found on most SAC plots.

SYNTAX

```
FILENUMBER {ON|OFF}
```

INPUT

FILENUMBER ON: Turn on file number option.
FILENUMBER {OFF}: Turn off file number option.

DEFAULT VALUES

```
FILENUMBER OFF
```

DESCRIPTION

This command controls the file number that is displayed on most SAC plots. When filenumber is on, the file number appears on the plot. This can be used to identify a specific waveform by number when a command requires the information.

LATEST REVISION

February 5, 1997 (Version 53)

FILTERDESIGN

SUMMARY

Produces a graphic display of a filter's digital vs. analog characteristics for: amplitude, phase, and impulse response curves, and the group delay.

SYNTAX

```
FILTERDESIGN [PRINT [pname] ] [FILE [prefix] ][filteroptions] [delta]
```

where filteroptions are the same as those used in the various filter commands in SAC, including the filter type. delta is the sampling interval of the data
Note Order of options is important. If the PRINT option is used, it must be the first option. If the FILE option is used, it must precede the filter options."

INPUT

:PRINT {pname} (**Prints the resulting plot to the printer named in**)
pname, or to the default printer if pname is not used." **Note** this must be the first option given on the command line. (This makes use of the SGF capability.)

:FILE {prefix} (**Writes three SAC files to disk. Theses files**)
contain the digital responses determined in the FILTERDESIGN:

[prefix].spec: is of type IAMPH, and contains both the amplitude and phase information from the FILTERDESIGN.

[prefix].gd: is of type ITIME, and contains the group delay information from the FILTERDESIGN. **Note** that in spite of the fact that the file is of type ITIME, group delay is a function of frequency. It is incumbent upon the user to remember that even though the plots will have seconds for units, the actual units are hertz.

[prefix].imp: is of type ITIME, and contains the impulse response.

In each of these SAC files, the user header fields are set as follows:

user0: pass code
- **1:** low pass
- **2:** high pass
- **3:** band pass
- **4:** band reject

user1: type code
- **1:** Butterworth
- **2:** Bessel
- **3:** C1
- **4:** C2

user2: number of poles
user3: number of passes
user4: tranbw
user5: attenuation
user6: delta
user7: first corner
user8: second corner if present, or -12345 if not
kuser0: pass (lowpass, highpass, bandpass, or bandrej)
kuser1: type (Butter, Bessel, C1, or C2)

DEFAULT VALUES

Only the delta parameter has a default (0.025 seconds). Options for filter type and related parameters must be supplied.

DESCRIPTION

The FILTERDESIGN command is implemented through XAPiir, a basic recursive digital filtering package (see REFERENCES). XAPiir implements the standard recursive digital filter design through bilinear transformation of prototype analog filters. These prototype filters, specified in terms of poles and zeros, are then transformed to highpass, bandpass and band reject filters using analog spectral transformations. FILTERDESIGN displays digital filter responses as solid lines and analog responses as dashed lines. On color monitors, digital curves are blue while analog curves are amber.

EXAMPLES

The following example shows how the FILTERDESIGN command is used to produce the digital and analog response curves for a highpass, 2 Hz., six pole, two pass filter on data with a sampling rate of .025 seconds.:

```
u1: fd hp c 2 n 6 p 2 delta .025
```

SEE COMMANDS

HIGHPASS, LOWPASS, BANDPASS, BANDREJECT UCRL-ID-106005. XAPiir: A Recursive Digital Filtering Package. David Harris. September 21, 1990 In Xwindows, a linestyle problem may cause both analog and digital traces to plot as solid lines.

LATEST REVISION

July 22, 1991 (Version 0.58)

FIR

SUMMARY

Applies a finite-impulse-response filter.

SYNTAX

```
FIR {REC|FFT},file
```

INPUT

FFT: Apply the FIR filter using the transform method.
REC: Apply the FIR filter recursively.
file: The name of the file containing the FIR filter.

ALTERNATE FORMS

DFT may be used in place of FFT.

DEFAULT VALUES

```
FIR FFT FIR
```

DESCRIPTION

The filter applied by this command must have been designed by using the DFIR interactive filter design program (see BUGS below). The filter is applied using the transform method unless you request the recursive method or the number of data points is too large for the transform method. These filters all have zero phase distortion but can produce precursors with impulsive signals.

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES

- 1301: No data files read in.
- 1306: Illegal operation on unevenly spaced file
- 1307: Illegal operation on spectral file
- 1601: File and filter sampling intervals not equal for
  - The filter must be designed using the same sampling rate as the data to be filtered.
- 1603: Inadequate memory to perform FIR filter.

WARNING MESSAGES

- 1602: Inadequate memory to perform FIR filter using DFT.
  - the recursive method will be used automatically.

LIMITATIONS

Maximum number of data points for transform method is 4096. The DFIR routine has since vanished, as this has not been used by the Seismologists at LLNL for some years. 1. See the author for information on the use of DFIR. 2. See Chapter 3 of Rabiner and Gold, Theory and Application of Digital Signal Processing, Prentice-Hall, 1975 for a discussion of FIR filters.

LATEST REVISION

July 22, 1991 (Version 8.0)

FLOOR

SUMMARY

Puts a minimum value on logarithmically scaled data.

SYNTAX

```
FLOOR {ON|OFF|v}
```

INPUT

{ON}: Turn floor option on but don't change value of floor.
OFF: Turn floor option off.
v: Turn floor option on and change value of floor.

DEFAULT VALUES

```
FLOOR 1.0E-10
```

DESCRIPTION

The floor option applies only when logarithmic scaling is being used. It applies to both the x and y axes. When this option is on, any data values less than the floor are set to the floor before plotting. By using a small positive value for the floor, errors in taking logarithms of non-positive numbers are avoided.

LATEST REVISION

January 8, 1983 (Version 8.0)

FUNCGEN

SUMMARY

Generates a function and stores it in memory.

SYNTAX

```
FUNCGEN {type},{DELTA v},{NPTS n},{BEGIN v}
```

where type is one of the following:

```
IMPULSE
STEP
BOXCAR
TRIANGLE
SINE {v1 v2}
LINE {v1 v2}
QUADRATIC {v1 v2 v3}
CUBIC {v1 v2 v3 v4}
SEISMOGRAM
RANDOM {v1 v2}
IMPSTRIN  {n1 n2 ...  nN}
```

INPUT

IMPULSE: Impulse at central data point.

IMPSTRIN: A series of impulses at the specified sample points.

STEP: Step function. Zero in first half. One in second half.

BOXCAR: Boxcar function. Zero in first and last thirds. One in middle third.

TRIANGLE: Triangle function. Zero in first and last quarters. Linearly increasing from zero to one in second quarter and decreasing from one to zero in third quarter.

SINE {v1 v2}: Sine wave with frequency in Hz given by v1 and phase angle in degrees given by v2. Amplitude is one.

Note that there is a factor of 2*pi in the phase argument: function = 1.0 * sin (2 * Pi * f * t)

LINE {v1 v2}: Linear function with slope given by v1 and intercept by v2.

QUADRATIC {v1 v2 v3}: Quadratic function of the form: ,BREAK ,TEX v1*t^{2} + v2*t + v3

CUBIC {v1 v2 v3 v4}: Cubic function of the form: ,BREAK ,TEX v1*t^{3} + v2*t^{2} + v3*t + v4

SEISMOGRAM: Sample seismogram. This is an obsolete option that has been replaced by the DATAGEN command. The DELTA, NPTS, and BEGIN options are ignored for this function. There are 1000 points in the sample seismogram.

RANDOM {v1 v2}: Random sequence (Gausian white noise) generator. v1 is the number of random sequence files to generate and v2 is the "seed" used to generate the first random number. This seed value is stored in USER0 so that you can regenerate the same random sequence at a later time if desired.

DELTA v: Set increment between samples to v. Stored in header as DELTA.

NPTS n: Set number of data points in function to n. Stored in header as NPTS.

BEGIN v: Set begin time to v. Stored in header as BEGIN.

DEFAULT VALUES

```
FUNCGEN IMPULSE NPTS 100 DELTA 1.0 BEGIN 0.
```

Frequency and phase angle for SINE function are 0.05 and 0. Coefficients for LINE, QUADRATIC, and CUBIC are all 1. Number of random sequences is 1 and seed is 12357.

DESCRIPTION

Executing this command is equivalent to reading a single file (except for the RANDOM option in which more than one file can be generated) into memory whose name is the name of the function generated. Any data previously in memory is destroyed. Other functions will be added as needed.

Any command which loads data into memory is monitored to maintain a level of confidence in the event infomation when transfered from the SAC data buffer to the CSS data buffer. When FUNCGEN is used, the confidence is set to LOW, indicating that SAC should consider any matching event IDs as artifacts and reassign the event ID of the incoming file. For more details, use HELP READ.

HEADER CHANGES

A header is set up in memory which accurately describes the function generated.

SEE COMMANDS

DATAGEN

LATEST REVISION

October 11, 1984 (Version 9.1)

GETBB

SUMMARY

Gets (prints) values of blackboard variables.

SYNTAX

GETBB {options} ALL|variable {variable ...} where options is one or more of the following:

```
TO TERMINAL|filename
NAMES ON|OFF
NEWLINE ON|OFF
```

INPUT

TO TERMINAL: Print the values to the terminal.

TO filename: Append the values to a file called filename.

NAMES [ON]: Include the name of the blackboard variable followed by an equals sign and then its value.

NAMES OFF: Only print the value of the blackboard variable.

NEWLINE [ON]: Put a newline (carriage-return) after each blackboard value printed.

NEWLINE OFF: Do not a newline after each value.

ALL: Print the values of all currently defined blackboard variables.

variable: Print the values of the specific blackboard variables listed.

DEFAULT VALUES

```
GETBB TO TERMINAL NAMES ON NEWLINE ON ALL
FUNCTIONAL MODULE: Executive
```

DESCRIPTION

The blackboard is a place to temporarily store information. This command lets you print the values of selected blackboard variables. Variables can be defined using the SETBB command. You can also use the EVALUATE command to perform basic arithmetic operations on blackboard variables and store the results in new blackboard variables. Blackboard variables can also be substituted directly into SAC commands. See the section on Macros in the Users Manual. The options to this command let you control where the values are printed and in what format to print them. You can print them to the terminal or append them to the end of a text file. You can include the variable name and an equals sign before the value or you can just have the value printed. You can have a newline placed after each value printed in a list or you can have them placed on a single line. You can use these options to make measurements on a set of data files, extract these measurements into a text file, and then read this file back into SAC using the READALPHA command to plot the results or to perform more analysis. This is illustrated in the examples section.

EXAMPLES

Assume you have already set several blackboard variables:

```
u:  SETBB C1 2.45 C2 4.94
```

To later print their values you would use this command:

```
u:  GETBB C1 C2
s:  C1 = 2.45
s:  C2 = 4.94
```

To print just their values on a single line:

```
u:  GETBB NAMES OFF NEWLINE OFF C1 C2
s:  2.45 4.94
```

Assume you have a macro called GETXY that performs some type of analysis on a single data file and stores the results into two blackboard variables called X and Y. You want to perform this analysis on all of the vertical components in the current directory, save each set of X and Y values, and plot them. In the following macro the first (and only) argument is the name of the text file to be used to store the results:

```
DO FILE WILD *Z
READ FILE
MACRO GETXY
GETBB TO 1 NAMES OFF NEWLINE OFF X Y
ENDDO

GETBB_ TO TERMINAL
READALPHA CONTENT P 1
PLOT
```

The text file would contain pairs of x-y data points, one per line, for each of the vertical data files. The final GETBB command redirecting the output back to the terminal is needed in order to close the text file and dump the buffer.

SEE COMMANDS

SETBB, EVALUATE

LATEST REVISION

Sept. 1, 1988 (Version 10.3E)

GRAYSCALE

SUMMARY

Produces grayscale images of data in memory.

SYNTAX

```
GRAYSCALE {options}
```

where options are one or more of the following:

```
VIDEOTYPE NORMAL|REVERSED
SCALE v
ZOOM n
XCROP n1 n2|ON|OFF
YCROP n1 n2|ON|OFF
```

SPECIAL NOTE This command uses executables that are not distributed with SAC. To use this command you must first install the Utah Raster Toolkit. The Utah Raster Toolkit can be obtained via anonymous FTP as follows:

```
ftp cs.utah.edu
cd pub
get urt-3.0.tar.Z
```

If ARPAnet is not available, or if you have questions about the Utah Raster Toolkit, send mail to: toolkit-requestCS.UTAH.EDU (ARPA), OR {ihnp4,decvax}!utah-cs!toolkit-request (UUCP).

INPUT

VIDEO NORMAL: Set video type to normal. In normal mode, data with near minimum values are black and data near maximum are white.

VIDEO REVERSED: Set video type to reversed. In reversed mode, data with near minimum values are white and data near maximum are black.

SCALE v: Change data scaling factor to v. The data is scaled by raising it to the vth power. Values less than one will smooth the image, reducing peaks and valleys. Values greater than one will spread the data.

ZOOM n: Image is increased to n times its normal size by pixel replication.

XCROP n1 n2: Turn x cropping option on and change cropping limits to n1 and n2. The limits are in terms of the image size.

XCROP {ON}: Turn x cropping option on and use previously specified cropping limits.

XCROP OFF: Turn x cropping option off. All of the data in the x direction is displayed.

YCROP n1 n2: Turn y cropping option on and change cropping limits to n1 and n2. The limits are in terms of the image size.

YCROP {ON}: Turn y cropping option on and use previous specified cropping limits.

YCROP OFF: Turn y cropping option off. All of the data in the y direction is displayed.

DEFAULT VALUES

```
GRAYSCALE VIDEOTYPE NORMAL SCALE 1.0 ZOOM 1 XCROP OFF YCROP OFF
```

DESCRIPTION

This command can be used to produce a grayscale image of the output of the SPECTROGRAM command or of any other two-dimensional array data. The SAC data displayed by this command must of of file type "xyz".

ANOTHER SPECIAL NOTE: SAC starts a shell script which runs the image manipulation and display programs and then displays the SAC prompt again. There is a delay, significant for large images and/or slower machines, before the image is actually displayed.

LIMITATIONS

Images of 512 by 1000 are the maximum displayed.

ACKNOWLEDGEMENTS

This command was developed by Terri Quinn. The grayscale images are maninpulated and display using the University of Utah's Raster Toolkit. The Utah Raster Toolkit and accompanying documentation; John W. Peterson, Rod G. Bogart, and Spencer W. Thomas.

HEADER VARIABLES

REQUIRED: : IFTYPE, NXSIZE, NYSIZE

ERROR MESSAGES

- SAC> getsun: Command not found.
  - Several utility programs distributed with the Utah Raster Toolkit are required.

SEE COMMANDS

SPECTROGRAM

LATEST REVISION

March 22, 1990 (Version 10.5a)

GRID

SUMMARY

Controls the plotting of grid lines in plots.

SYNTAX

```
GRID {ON|OFF|SOLID|DOTTED}
```

INPUT

ON: Turn grid plotting on but don't change grid type.
OFF: Turn grid plotting off.
SOLID: Turn grid plotting on using solid grid lines.
DOTTED: Turn grid plotting on using dotted grid lines.

DEFAULT VALUES

```
GRID OFF
FUNCTIONAL MODULE: Graphic Environment
```

DESCRIPTION

This command controls grid lines in both directions. The XGRID and YGRID commands can be used to generate grid lines in only one direction.

SEE COMMANDS

XGRID, YGRID

LATEST REVISION

January 8, 1983 (Version 8.0)

GTEXT

SUMMARY

Controls the quality and font of text used in plots.

SYNTAX

```
GTEXT {SOFTWARE|HARDWARE},{FONT n},{SIZE size} {SYSTEM system} {NAME name}
```

INPUT

SOFTWARE: Use software text in plots.

HARDWARE: Use hardware text in plots.

FONT n: Set software text font to n. The range for n is currently 1 to 8.

FORCE n: Use harware text in all cases for plots. Overrides HARDWARE option. HARDWARE still uses software for rotated fonts.

SIZE size: Change default text size. See TSIZE command for definitions of text sizes. Option size is one fo the following:

`TINY|SMALL|MEDIUM|LARGE`

SYSTEM system: Change the Font subsystem, current values are - SOFTWARE Traditional SAC Font system - CORE X11 Core Fonts, this creates a real font - XFT X Freetype library, this creates a real font

NAME name: Change the default font used in the CORE or XFT subsystem Available fonts are: Helvetica, Times-Roman, Courier, ZapfDingbats

DEFAULT VALUES

```
GTEXT SOFTWARE FONT 1 SIZE SMALL
```

DESCRIPTION

Software text uses the text display capabilities of the graphics library. Characters are stored as small line segments and thus can be scaled to any desired size and can be rotated to any desired angle. Use of software text will produce the same result on different graphics devices. Use of software text is slower than hardware text, especially to the terminal. There are currently 8 software fonts available: simplex block (font 1), simplex italics (2), duplex block (3), duplex italics (4), complex block (5), complex italics (6), triplex block (7), and triplex italics (8). Examples of each font and each default text size is shown in the figure on the next page. Hardware text uses the text display capabilities of the graphics device itself. Hardware text sizes vary considerably between devices, so its use can produce different looking plots on different devices. If a device has more than one hardware text size, the one closest to the desired size is used. Its primary asset is that it is much faster than software text and should therefore be used only when speed is more important than quality.

EXAMPLES

To select the triplex software font:

```
u:  GTEXT SOFTWARE FONT 6
```

SEE COMMANDS

TSIZE

LATEST REVISION

July 22, 1991 (Version 9.1) Text Fonts and Default Text Sizes

HANNING

SUMMARY

Applies a "hanning" window to each data file.

SYNTAX

```
HANNING
```

DESCRIPTION

The "hanning" window is a recursive smoothing algorithm defined at each interior data point, j, as:

```
Y(j) = 0.25*Y(j-1) + 0.50*Y(j) + 0.25*Y(j+1)
```

Each end point is set equal to its closest interior point.

ERROR MESSAGES

- 1301: No data files read in.
- 1306: Illegal operation on unevenly spaced file

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN Blackman and Tukey, "The Measurement of Power Spectra", Dover Publications, New York, 1958.

LATEST REVISION

January 8, 1983 (Version 8.0)

HELP

SUMMARY

Displays information about SAC commands and features on the screen.

SYNTAX

```
HELP {item ...}
```

INPUT

item: The (full or abbreviated) name of a command, module, subprocess, feature, etc.

DEFAULT VALUES

```
If no item is requested, an introductory help package is displayed.
```

DESCRIPTION

Each requested item in the help package is displayed in the order they are requested. A short message is displayed if no information is available for an item. After a full page of output, the user is prompted to see if he or she wishes to see more information on that item. A response of "NO" or "N" will terminate the display of that item and will begin the display of the next item if any. A response of "QUIT" or "Q" will terminate the display of all items. The help package for each command consists of the entry in the SAC Command Reference Manual. The help package for non-commands may be paragraphs from the SAC Users Manual or other information.

EXAMPLES

To get the introductory help package type:

```
u:  HELP
```

Now lets say you want information on several commands:

```
u:  HELP READ CUT BEGINDEVICE PLOT
```

SAC begins displaying the READ help package. After a full page, it asks if you've seen enough:

```
s:  MORE?
u:  YES
```

SAC displays the rest of the help package on READ, and then begins displaying the help package on the CUT command:

```
s:  MORE?
u:  NO
```

SAC stops displaying the CUT help package and begins displaying the BEGINDEVICE help package:

```
s:  MORE?
u:  QUIT
```

You're getting impatient so you type QUIT. SAC terminates the HELP command so you can try some of the features discussed.

ERROR MESSAGES

- 1103: No help package is available.
  - SAC can't find the help package. Check your SACAUX environment.

SEE COMMANDS

PRINTHELP

LATEST REVISION

November 13, 1998 (Version 0.58)

HIGHPASS

SUMMARY

Applies an IIR highpass filter.

SYNTAX

```
HIGHPASS {BUTTER|BESSEL|C1|C2},{CORNERS v1 v2},
{NPOLES n},{PASSES n},{TRANBW v},{ATTEN v}
```

INPUT

BUTTER: Apply a Butterworth filter.
BESSEL: Apply a Bessel filter.
C1: Apply a Chebyshev Type I filter.
C2: Apply a Chebyshev Type II filter.
CORNER v: Set corner frequency to v.
NPOLES n: Set number of poles to n {range: 1-10}.
PASSES n: Set number of passes to n {range: 1-2}.
TRANBW v: Set the Chebyshev attenuation factor to v.
ATTEN v: Set the Chebyshev attenuation factor to v.

DEFAULT VALUES

```
HIGHPASS BUTTER CORNER 0.2 NPOLES 2 PASSES 1 TRANBW 0.3 ATTEN 30.
```

DESCRIPTION

See the BANDPASS command for definitions of the filter parameters and descriptions on how to use them.

EXAMPLES

To apply a four-pole Butterworth with a corner at 2 Hz.:

```
u:  HIGHPASS NPOLES 4 CORNER 2
```

To apply a two-pole two-pass Bessel with the same corner.:

```
u:  HP N 2 BE P 2
```

ERROR MESSAGES

- 1301: No data files read in.
- 1306: Illegal operation on unevenly spaced file
- 1307: Illegal operation on spectral file
- 1002: Bad value for
  - corner frequency larger than Nyquist frequency.

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

SEE COMMANDS

BANDPASS

LATEST REVISION

January 8, 1983 (Version 8.0)

HILBERT

SUMMARY

Applies a Hilbert transform.

SYNTAX

HILBERT

DESCRIPTION

Each data file, y(n), in the data file list is replaced by its Hilbert transform, x(n). The transform is found by convolving y(n) (in the time domain) with a 201 point FIR filter: The filter impulse response is obtained by windowing an ideal Hilbert transformer impulse response with a Hamming window: In the frequency domain, this filter approximates the transfer function: The phase criterion is met exactly (90 degree phase shift at each frequency), and the magnitude response is (ideally) unity.

Note that the operation is inexact in small regions about DC and the folding frequency. If transforms are to be taken of very low frequency data, such as long period surface waves, the signals should first be decimated. Since the transformation is performed in the time domain, computations are done in-place using the overlap-save algorithm. There are no restrictions on the length of data file.

(Added in 2013) Hilbert transforms can be used to calculate the minimum-delay phase from (the log of) the spectral amplitude. Such amplitudes are effectively low-pass filters, which are not band-limited, and the procedure used here does not work very well for such functions.

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

ACKNOWLEDGEMENT

The subroutines used to perform the Hilbert transform were designed and developed by Dave Harris.

LATEST REVISION OF CODE

April 21, 1989 (Version 10.4c) Amplitude Response of Hilbert Transform.

HISTORY

SUMMARY

prints a list of the recently issued SAC commands

SYNTAX

```
HISTORY
```

INPUT

none

DEFAULT VALUES

```
none
```

DESCRIPTION

The history module provides a subset of the history capabilities available in the unix C-shell. Issuing the comand "history" will print a numbered list of the most recent commands (up to 100). Several of the event designators from the C-shell are also available. These are references to command lines in the history list. Available designators are:

! Start a history substitution, except when followed by a space character, tab, newline, = or (.

!! Repeat the previous command. !n Repeat command line n. !-n Repeat current command line minus n. !str Repeat the most recent command starting with str

EXAMPLES

To print the history list:

```
u:  history
```

To repeat command 1:

```
u:  !1
```

To repeat the last command:

```
u:  !!
```

To repeat the 2nd-to-last command:

```
u:  !-2
```

To repeat the command starting with ps:

```
u:  !ps
```

ERROR MESSAGES

none

LATEST REVISION

March 03, 1997

IFFT

SUMMARY

Performs an inverse discrete Fourier transform.

SYNTAX

```
IFFT
```

ALTERNATE FORMS

Purists may use IDFT instead of IFFT.

DESCRIPTION

Data files must have been previously transformed using the FFT command. They may be in either real-imaginary or amplitude-phase format.

HEADER CHANGES

B, DELTA, and NPTS are changed to the beginning frequency, sampling frequency, and number of data points in the transform. The original values of B, DELTA, and NPTS are saved in the header as SB, SDELTA, and NSNPTS and are restored when this command performed.

ERROR MESSAGES

- 1301: No data files read in.
- 1305: Illegal operation on time series file
- 1606: Maximum allowable DFT is

LIMITATIONS

The maximum inverse transform that can currently be performed is 65536 points.

SEE COMMANDS

FFT

LATEST REVISION

October 11, 1984 (Version 9.1)

IMAGE

SUMMARY

Produces color sampled image plots of data in memory.

SYNTAX

```
IMAGE {COLOR|GREY} {BINARY|FULL} {PRINT {pname} }
```

INPUT

COLOR|GREY: Produce a color or greyscale image.

BINARY|FULL: Produce an image where all positive values plot in one color and all negative values plot in a second color, or plot the full range of the data.

PRINT {pname}: Prints the resulting plot to the printer named in pname, or to the default printer if pname is not used. (This makes use of the SGF capability.)

DEFAULT VALUES

```
IMAGE COLOR FULL
```

DESCRIPTION

The image command allows the user to make color or grayscale images from a SAC 3-D data file such as those generated by the spectrogram, scallop, or bbfk commands. It can also be used to plot imported data provided they are in the SAC 3-D data format. Different sections of the image can be viewed using the xlim and ylim commands and amplitudes can be scaled using the usual unary operations provided in SAC.

HEADER VARIABLES

REQUIRED: IFTYPE (set to "IXYZ"), NXSIZE, NYSIZE

USED: XMINIMUM, XMAXIMUM, YMINIMUM, YMAXIMUM

LATEST REVISION

May 26, 1995 (Version 00.31)

INICM

SUMMARY

Reinitializes all of SAC's common blocks.

SYNTAX

```
INICM
```

DESCRIPTION

This command can be used at any time to put SAC back into its initial state. SAC-related environmental variables are honored, but an initializing macro is not. All active graphics devices are terminated and the graphics library ended. All common blocks are reinitialized to their original values. All data in memory is lost.

LATEST REVISION

January 8, 1983 (Version 8.0)

INSTALLMACRO

SUMMARY

Installs macro files in the global SAC macro directory.

SYNTAX

```
INSTALLMACRO name {name ...}
```

INPUT

name: The name of a SAC macro file.

DESCRIPTION

This command lets you install your macro files into the global SAC macro directory so they can be used by anyone on your system. The location of this directory is defined by the SACAUX environmental variable, as SACAUX/macros. See the section on Macros in the SAC Users Manual.

SEE COMMANDS

MACRO

LATEST REVISION

March 20, 1992, (version 10.6e)

INT

SUMMARY

Performs integration using the trapezoidal or rectangular rule.

SYNTAX

```
INT TRAPEZOIDAL | RECTANGULAR
```

DEFAULT VALUES

```
INT TRAPEZOIDAL
```

DESCRIPTION

This command uses the trapezoidal or rectangular integration method. The first output data point is set to zero. If the trapezoidal option is used, the number of points is reduced by one. The data does not have to be evenly spaced.

ERROR MESSAGES

- 1301: No data files read in.
- 1307: Illegal operation on spectral file

HEADER CHANGES

DEPMIN, DEPMAX, DEPMIN

LATEST REVISION

March 20, 1992 (Version 10.6e)

INTERPOLATE

SUMMARY

Interpolates evenly-spaced data to a new sampling rate. Interpolate can also be used with unevenly-spaced data.

SYNTAX

```
INTERPOLATE {DELTA v} {NPTS n} {BEGIN v}
```

INPUT

DELTA v: Set new sampling rate to v. The time range (E-B) is not changed, so NPTS is changed. However, E will be changed so that it is a multiple of DELTA from b. Both DELTA and NPTS cannot be used in the same call.

NPTS n: Force the number of points in interpolated file to be n. The time range (E-B) is not changed, so DELTA is changed. Both DELTA and NPTS cannot be used in the same call.

BEGIN v: Start interpolation at v. This value becomes the begin time of the interpolated file. BEGIN can be used with either DELTA or NPTS.

DEFAULT VALUES

The time series is unchanged.

DESCRIPTION

This command uses the Wiggins' weighted average-slopes interpolation method (1976, BSSA, 66, p. 2077) to convert unevenly-spaced data to evenly-spaced data but which works quite well at resampling evenly-spaced data to a different sampling rate. Unlike cubic-spline interpolation, there is no extrema between input sample points. If the sample rate is decreased, there is no antialiasing, so for downsampling, DECIMATE may be a better option. An alternative to using BEGIN is to CUT the time series to the desired B and E before calling INTERPOLATE. If DELTA and NPTS are in the same call to INTERPOLATE, the last one in the command sequence will be used.

Wiggins' subroutine included EPSILON, which gives a lower limit for local slopes. In earlier versions of INTERPOLATE, one could modify EPSILON. As of the version accompanying SAC v101.5, the code has been rewritten so that there is no reason to consider modifying EPSILON. Hence, that option has been removed.

EXAMPLES

Assume that FILEA is an evenly-spaced data file with a sampling interval of 0.025. To convert it to a sampling rate of 0.02 seconds:

```
SAC>  READ FILEA
SAC>  INTERPOLATE DELTA 0.02
```

Because the new delta is less than the data delta, a warning message will be printed because of the potential for aliasing.

Assume that FILEB has NPTS=3101 and one wants to have it sample the same time range but with NPS=4096 points (a power of 2):

```
SAC> READ FILEB
SAC> INTERPOLATE NPTS 4096
```

If one tries to change DELTA and NPTS in the same call, only the second call will be used. Hence if the previous call were replaced by:

```
SAC> READ FILEB
SAC> INTERPOLATE NPTS 4096 DELTA 0.02
```

DELTA would be changed to 0.02 and NPTS would be calculated from the new DELTA and the input B and E. If the order were reversed:

```
SAC> READ FILEB
SAC> INTERPOLATE DELTA 0.02 NPTS 4096
```

the output file would have NPTS=4096 and DELTA would be calculated.

Assume that FILEC is an unevenly spaced data file. To convert it to an evenly spaced file with a sampling interval of 0.01 seconds:

```
SAC>  READ FILEC
SAC>  INTERPOLATE DELTA 0.01
```

WARNING MESSAGES

- 2008: Requested begin time is less than data begin time. Output truncated.
- 2125: Requested begin time is greater than data end time. No action taken.

HEADER CHANGES

DELTA, NPTS, E, B (if FIRST invoked), LEVEN (if initially unevenly spaced.)

LATEST REVISION

August 2011 (Version 101.5)

KEEPAM

SUMMARY

Keep amplitude component of spectral files (of either the AMPH or RLIM format) in SAC memory.

SYNTAX

```
KEEPAM
```

DESCRIPTION

This command is an easy way for users to drop the phase component, so that the amplitude data may be subjected to algebraic operations which require single-dimensional data. If the files exist in the RLIM format, the data is first converted to the AMPH format, before phase is dropped. The resulting data files containing the amplitude component will exist as as generic xy files, so that they can be distinguished for time-domain files. May 28, 1991 (Version 10.5c)

KHRONHITE

SUMMARY

Applies a Khronhite filter to the data.

SYNTAX

```
KHRONHITE {v}
```

INPUT

v: Cutoff frequency in hertz.

DEFAULT VALUES

```
KHRONHITE 2.0
```

DESCRIPTION

This lowpass filter is a digital approximation of an analog filter which was a cascade of two fourth-order Butterworth lowpass filters. This lowpass filter has been used with a corner frequency of 0.1 Hz to enhance measurements of the amplitudes of the fundamental mode Rayleigh wave (Rg) at regional distances.

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

LATEST REVISION

February 15, 1987

LINE

SUMMARY

Controls the linestyle selection in plots.

SYNTAX

```
LINE {ON|OFF|SOLID|DOTTED|n} {FILL ON|OFF|{POS_COLOR|NEG_COLOR}}
{INCREMENT {ON|OFF}}, {LIST STANDARD|nlist}
```

INPUT

{ON}: Turn line drawing on. Don't change linestyle.

OFF: Turn line drawing off.

SOLID: Change to solid linestyle and turn line drawing on.

DOTTED: Change to dotted linestyle and turn line drawing on.

n: Change to linestyle n and turn line drawing on. A linestyle of 0 is the same as turning turning line drawing off.

FILL ON/OF: Turn filling on/off.

FILL POS_COLOR/NEG_COLOR: Fill color for positive/negative section of the seismogram trace.

FILL LIST STANDARD: Use Standard color list for Color Filling.

fILL LIST POS_COLOR/NEG_COLOR: Turn on color filling incrementing m ultiple colors in a list are available to set colors. Colors are specified in either Color Name or Number

INCREMENT {ON}: Increment linestyle from linestyle list after each data file is plotted.

INCREMENT OFF: Do not increment data linestyle.

LIST nlist: Change the content of the linestyle list. Enter list of linestyle numbers.

LIST STANDARD: Change to the standard linestyle list.

DEFAULT VALUES

```
LINE SOLID INCREMENT OFF LIST STANDARD
```

DESCRIPTION

This command controls the linestyle used when plotting data. The skeleton (axes, titles, etc.) are always plotted using solid lines. Grid linestyle is controlled by the GRID command.

There are other commands that control other aspects of the data display. The SYMBOL command can be used to display a set of scalable, centered symbols at each data point. The COLOR command controls color selection for color graphics devices. All of these attributes are independent of each other. You may select a blue dotted line with a symbol at each data point if you desire A linestyle of 0 is the same as turning turning line drawing off. This is useful in the LIST option and the SYMBOL command to display some data with lines and some with symbols on the same plot. See the example below.

EXAMPLES

To select an incrementing linestyle starting with linestyle 1:

```
SAC>  LINE 1 INCREMENT
```

To change the linestyle list to contain linestyles 3, 5, and 1:

```
SAC>  LINE LIST 3 5 1
```

To plot three files on the same plot using PLOT2 with the first file plotted using a solid line and no symbol, the second with no line and a triangle symbol, and the third with no line and a cross symbol:

```
SAC>  READ FILE1 FILE2 FILE3
SAC>  LINE LIST 1 0 0 INCREMENT
SAC>  SYMBOL LIST 0 3 7 INCREMENT
SAC>  PLOT2
```

To fill in the positive ecursions on a seismogram with red and the negtive excursions with blue. If one left out the 0, the colored regions would be outlined with a black line:

```
SAC> fg seismo
SAC> line 0 fill red/blue
SAC> p1
```

SEE COMMANDS

SYMBOL, COLOR

LATEST REVISION

Version 101.6

LINEFIT

SUMMARY

Computes the best straight line fit to the data in memory and writes the results to header blackboard variables.

SYNTAX

```
LINEFIT
```

DESCRIPTION

A least squares curve fit to a straight line is calculated. The slope, y intercept, standard deviation of the slope, standard deviation of the y intercept, standard deviation of the data and correlation coefficient between the data and the linear fit are written to blackboard variables SLOPE, YINT, SDSLOPE, SDYINT, SDDATA and CORRCOEF respectively. The data does not have to be evenly spaced.

ERROR MESSAGES

- 1301: No data files read in.
- 1307: Illegal operation on spectral file

HEADER CHANGES

none

LATEST REVISION

September 12, 1995 (Version 00.38)

LINLIN

SUMMARY

Turns on linear scaling for the x and y axes.

SYNTAX

```
LINLIN
```

DEFAULT VALUES

```
Linear scaling for both axes.
```

SEE COMMANDS

LINLOG, LOGLOG, LOGLIN

LATEST REVISION

January 8, 1983 (Version 8.0)

LINLOG

SUMMARY

Turns on linear scaling for x axis and logarithmic for y axis.

SYNTAX

```
LINLOG
```

DEFAULT VALUES

```
Linear scaling for both axes.
```

SEE COMMANDS

LINLIN, LOGLOG, LOGLIN

LATEST REVISION

January 8, 1983 (Version 8.0)

LISTHDR

SUMMARY

Lists the values of selected header fields.

SYNTAX

LISTHDR {listops} {hdrlist} where listops are one or more of the following:

```
DEFAULT|PICKS|SPECIAL
FILES ALL|NONE|list
COLUMNS 1|2
INCLUSIVE ON|OFF
```

INPUT

DEFAULT: Use the default list, which includes all defined header fields.
PICKS: Use the picks list, which includes those header fields used to define time picks.
SPECIAL: Use the special user defined list.
FILES ALL: List headers from all files in data file list.
FILES NONE: Don't list headers, set defaults for future commands.
FILES list: List headers from a subset of the files in the data file list. The subset is defined as a list of file numbers.
COLUMNS 1: Format output into a single column of entries.
COLUMNS 2: Format output into two columns.
INCLUSIVE: ON includes header variables which are undefined. OFF excludes them.
hdrlist: List of header fields to be included in the special list.

DEFAULT VALUES

```
LISTHDR DEFAULT FILES ALL COLUMNS 1 INCLUSIVE OFF
```

DESCRIPTION

The user can define which items to list or can use either of two standard lists. The first list (DEFAULT) contains all of the header fields. The second list (PICKS) contains those header fields which are directly or indirectly used to define time picks. This list contains the following fields: B, E, O, A, Tn, KZTIME, KZDATE. More standard lists can be added if needed. A special list can be defined by the user at any time and can then be requested again by using the SPECIAL option. The full listing for a header field consists of its name, an equals sign, and its current value. Some of the fields for some files will be undefined. SAC stores a special value in those fields to flag them as undefined. The listing excludes these undefined fields unless the INCLUSIVE option is ON. For integers and floats the undefined value is -12345; for character strings and those integers which are used to indicate character strings, the undefined value it "UNDEFINED".

If one reads in a waveform file and transforms to the frequency domain, the data extremes (MAXDEP, MINDEP) will return values in LH for the time domain, not the frequency domain.

A description of each of the SAC header fields is contained in SAC Data File Format.

ERROR MESSAGES

1301: No data files read in.

EXAMPLES

To get a two column listing of the time picks only:

u: LISTHDR PICKS COLUMNS 2

To get a default listing of the third and fourth files in the data file list:

u: LISTHDR FILES 3 4

To list the values of the begin and end time only:

u: LISTHDR B E

To define a special list of the station parameters:

u: LISTHDR KSTNM STLA STLO STEL STDP

To reuse this special list later during the same execution:

u: LISTHDR SPECIAL

To set default two column output:

u: LISTHDR COLUMNS 2 FILES NONE

LATEST REVISION

August 30, 2008 (Version 101.2)

LOAD

SUMMARY

Load an external command.

external commands and load require extra work in the linux version of SAC.

SYNTAX

```
LOAD comname {ABBREV abbrevname}
```

INPUT

comname: The name of an external function to load from a shared object.
ABBREV abbrevname: An abbreviation or alias for comname.

DESCRIPTION

This command allows the user to load commands written to the SAC external command interface specification (See EXTERNAL_INTERFACE help page). This command must be a function stored in a shared object library (a .so file- see UNIX LD manpage for details). SAC will look in all shared object libraries listed in environmental variable SACSOLIST. This environmental variable should contain one or more names of shared objects in a blank delimited list. The path to these shared objects must be specified in the LD_LIBRARY_PATH environmental variable. If SACSOLIST is not set, then SAC will look for a shared object library called libsac.so, using the paths specified in LD_LIBRARY_PATH. A library called libcom.so is distributed with SAC (see EXTERNAL COMMAND . below).

EXAMPLE

Set up your environment to have SAC look in the current directory for a command named foo from a shared object called libbar.so. Set up an alias for foo called myfft.:

```
% setenv SACSOLIST "libcom.so libbar.so"
#  Add the current directory to the search path.
% setenv LD_LIBRARY_PATH {$LD_LIBRARY_PATH}:.

% sac
SAC> load foo abbrev myfft          * load the command
SAC> read file1.z file2.z file3.z   * input files to pass to the command
SAC> myfft real-imag                * invoke command with its arguments,
                                    * commands must parse their own args.
SAC> psp
```

How to create a shared object library containing your command(s): Solaris:

```
cc -o libxxx.so -G extern.c foo.c bar.c
```

SGI:

```
cc -g -o libxxx.so -shared foo.c bar.c
```

LINUX: (gcc):

```
gcc -o libxxx.so -shared extern.c foo.c bar.c sac.a
```

where sac.a is the sac library available where you got sac.

EXTERNAL COMMAND INCLUDED IN THE DISTRIBUTION OF SAC

There is one external command which is distributed with SAC. It is called FLIPXY. FLIPXY takes as input one or more X-Y datafiles, and transposes the data. This command is in libcom.so in ${SACAUX}/external along with the source code of FLIPXY for reference. To load FLIPXY, libcom.so must be included in SACSOLIST.

ERRORS

- 1028: External command does not exist: This means that SAC did not find your external command.

  This error can arise for a couple of reasons. One possibility is that your LD_LIBRARY_PATH environmental variable does not contain the path to your shared library. Another possibility is that you have not set your SACSOLIST environmental variable to contain the names of your shared libraries.

LATEST REVISION

March 21, 1996 (Version 00.50)

LOADCTABLE

SUMMARY

Allows the user to select a new color table for use in image plots.

SYNTAX

LOADCTABLE n | [options] [filelist] where n is a number (currently between 1 and 17) of a standard SAC

color table stored in directory SACAUX, or where options is the following:

`DIR CURRENT|name`

`options MUST preceed any element in the filelist.`

INPUT

n: The number of a standard SAC color table.

DIR CURRENT: Load color table from the current directory. This is the directory from which you started SAC.

DIR name: Load color table from the directory called name. This may be a relative or absolute directory name.

filelist: file

file: A legal color table filename. This may be a simple filename or a pathname. The pathname can be a relative or absolute one.

DESCRIPTION

This command allows the user to select a new color table or provide their own custom color table by specifying the color table file with a pathname, relative to the current directory. If the DIR option is not used, SAC looks first in SACAUX for the color table, then in the user's working directory.

LATEST REVISION

May 26, 1995 (Version 00.31)

LOG

SUMMARY

Takes the natural logarithm of each data point.

SYNTAX

LOG

ERROR MESSAGES

- 1301: No data files read in.
- 1307: Illegal operation on spectral file
- 1340: data points outside allowed range contained in file
  - All data points must be positive.

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

LATEST REVISION

January 15, 1985 (Version 9.10)

LOG10

SUMMARY

Takes the base 10 logarithm of each data point.

SYNTAX

```
LOG10
```

ERROR MESSAGES

- 1301: No data files read in.
- 1307: Illegal operation on spectral file
- 1340: data points outside allowed range contained in file
  - All data points must be positive.

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

LATEST REVISION

January 15, 1985 (Version 9.10)

LOGLAB

SUMMARY

Controls labels on logarithmically scaled axes.

SYNTAX

```
LOGLAB {ON|OFF}
```

INPUT

{ON}: Turn log labeling option on.
OFF: Turn log labeling option off.

DEFAULT VALUES

```
LOGLAB ON
```

DESCRIPTION

Labels are normally put on each decade of logarithmically interpolated axes. Secondary labels (ones between full decades) are placed on these axes if this option is on and if there is enough room.

LATEST REVISION

January 8, 1983 (Version 8.0)

LOGLIN

SUMMARY

Turns on logarithmic scaling for x axis and linear for y axis.

SYNTAX

```
LOGLIN
```

DEFAULT VALUES

```
Linear scaling for both axes.
```

SEE COMMANDS

LINLIN, LINLOG, LOGLOG

LATEST REVISION

January 8, 1983 (Version 8.0)

LOGLOG

SUMMARY

Turns on logarithmic scaling for the x and y axes.

SYNTAX

```
LOGLOG
```

DEFAULT VALUES

```
Linear scaling for both axes.
```

SEE COMMANDS

LINLIN, LINLOG, LOGLIN

LATEST REVISION

January 8, 1983 (Version 8.0)

LOWPASS

SUMMARY

Applies an IIR lowpass filter.

SYNTAX

```
LOWPASS {BUTTER|BESSEL|C1|C2},{CORNER v},
{NPOLES n},{PASSES n},{TRANBW v},{ATTEN v}
```

INPUT

BUTTER: Apply a Butterworth filter.
BESSEL: Apply a Bessel filter.
C1: Apply a Chebyshev Type I filter.
C2: Apply a Chebyshev Type II filter.
CORNER v: Set corner frequency to v.
NPOLES n: Set number of poles to n {range: 1-10}.
PASSES n: Set number of passes to n {range: 1-2}.
TRANBW v: Set the Chebyshev transition band width to v.
ATTEN v: Set the Chebyshev attenuation factor to v.

DEFAULT VALUES

```
LOWPASS BUTTER CORNER 0.4 NPOLES 2 PASSES 1 TRANBW 0.3 ATTEN 30.
```

DESCRIPTION

See the BANDPASS command for definitions of the filter parameters and descriptions on how to use them.

EXAMPLES

To apply a four-pole Butterworth with a corner at 2 Hz.:

```
u:  LOWPASS NPOLES 4 CORNER 2
```

To apply a two-pole two-pass Bessel with the same corner.:

```
u:  LP N 2 BE P 2
```

A Butterworth causal low-pass filter will time shift (forward) the waveform by an amount that depends on the corner frequency. The following macro will time-shift the data.:

```
setbb wf $1
setbb ts $2
r %wf
ch b (%ts + &1,b&)
write %wf%-TS
```

If the macro is named time-shift.m, the waveform file named XXX and the time shift -0.25, the following sequence will time-shift the data and output a file named XXX-TS. This macro will fail if IZTYPE is IB.:

```
SAC> m time-shift.m XXX -0.25
```

ERROR MESSAGES

- 1301: No data files read in.
- 1306: Illegal operation on unevenly spaced file
- 1307: Illegal operation on spectral file
- **1002: Bad value forcorner frequency larger than Nyquist frequency.** See Chapter 4 of Rabinerand Gold, Theory and Application of Digital Signal Processing, Prentice-Hall, 1975 for a discussion of IIR filters.

SEE COMMANDS

BANDPASS

LATEST REVISION

Version 101.6

MACRO

SUMMARY

Executes a SAC macro file and the startup/init commands when invoking SAC.

SYNTAX

```
MACRO name {arguments}
```

INPUT

name: The name of the SAC macro to execute.
arguments: The arguments (if any) of the macro.

COMMAND ABBREVIATION

M:

DESCRIPTION

A SAC macro is a file that contains a set of SAC commands that you want to execute as a group. You can pass arguments to the macro, define default values for these arguments, evaluate blackboard and header variables within the body of a macro, etc. The macro file can be in the current directory, in a predefined directory you specify using the SETMACRO command, or in the global SAC macro directory. See the section on Macros in the SAC Users Manual.

SEE COMMANDS

SETMACRO, INSTALLMACRO

LATEST REVISION

May 15, 1987 (Version 10.2)

MAP

SUMMARY

Generate a GMT (Generic Mapping Tools) map which can include station/event symbols topography and station names using all the files in SAC memory and an optional event file specified on the command line. Event symbols can be scaled according to magnitude, residual, etc. A variety of projections are available. The result of this command is a postscript file and a screen display of that file plus a shell script with the GMT commands that produced the plot.

SYNTAX

```
MAP   {MERcator | EQuidistant | AZimuthal_equidistant | ROBinson }
{WEST minlon} {EAST maxlon} {NORTH maxlat} {SOUTH minlat}
{MAGnitude | REsidual | RMean_residual}
{EVentfile filename} {TOPOgraphy} {STANames}
{MAPSCALE on|off } {PLOTSTATIONS on|off} {PLOTEVENTS on|off}
{PLOTLEGEND on|off} {LEGENDXY x y} {FILE output-file}
```

Note Shorthand notations for keywords are in capital letters.

INPUT OPTIONS

MERCATOR: Generate a Mercator projection. [Default]

EQUIDISTANT: Generate an Equidistant cylindrical projection. Linear in latitude and longitude.

ROBINSON: Robinson projection, nice for world map.

LAMBERT: Good for large east-west areas.

UTM: Universal transverse mercator. (unimplemented)

Distances from center are true.

The following options allow the user to specify the map region. The default is to use the min and max defined by the plotted stations and events.

WEST: Define minimum longitude for map window.

EAST: Define maximum longitude for map window.

NORTH: Define maximum latitude for map window.

SOUTH: Define minimum latitude for map window.

AUTOLIMITS: Automatically Determine the Limits [Default]

The following options allow the user to add locations and annotations to the map.

STANames: on | off [Off]

MAPSCALE: on | off [Off] Plot a Distance Scale on the Map

PLOTSTATIONS: on | off [On] Plot all the Stations from seismograms

PLOTEVENTS: on | off [On] Plot all the Events from eventfile and/or seismograms

The following options allow the user to scale the event symbols. The default is a constant symbol size.

MAGnitude: Scale event symbols linearly with user0. [Off]

REsidual: Scale event symbols linearly with abs(user0). [Off] Positive values are (+) negatives are (-).

RMean_residual: Same as residual except mean is removed [Off] from all the residuals.

PLOTLEGEND: on | off [On] Plot a legend for Earthquake Magnitudes and Residuals

LEGENDXY x y: Absolute Location to Plot the Legend [1,1] The location is relative to the lower left corner of the Page Values for x and y are to be in inches. This is a legend for Earthquake Magnitudes and Residuals

EVENTFILE: Specify a free-format ASCII text file containing additional event data. Each line in the file contains data for a single event. The first two columns of each line must contain latitude and longitude (in degrees), respectively. The third column is optional and contains symbol size information (e.g., magnitudes, depth, travel-time residual, ...). The following is an example of a few lines in an eventfile:

```
38.5    42.5   6.5
25.5    37.3   5.5
44.2    40.9   5.7
```

A TITLE can be specified using SAC's TITLE command.

TOPOgraphy: on | off [Off] Setting TOPO on allows the user to add topography and bathymetry to the maps. The command reads option (1) in grdraster.info, and the data file referenced for that option must be installed. The topography color map used is in $SACAUX/ctables/gmt.cpt. The grid files are written in the current directory.

The default output file is gmt.ps. An alternative file name can be specified using the FILE option.

DEFAULT VALUES

```
MAP  MERCATOR TOPO off STAN off FILE gmt.ps PLOTSTATIONS on PLOTEVENTS
on
FUNCTIONAL MODULE: FK Spectrum (fks)
```

A title can be added using ch TITLE "..." before running map.

HEADER DATA

Station latitudes (stla) and longitudes (stlo) must be set. If event latitudes (evla) and longitudes (evlo) are set they will be included in the map. If this command is executed after running BBFK, MAP will plot a great circle arc path along the back azimuth.

This version of MAP was based on version 4.0 of the Generic Mapping Tools software and it assumes that the GMT4.0 software is loaded on the user's machine and the executables are in the path.

The results of each MAP command are written to a shell file called gmt.csh, in the current directory. The user can modify this file to take advantage of GMT options not readily available through SAC. Default units are in inches, but can be changed in the shell script.

The results of each MAP command will automatically be displayed. The default program used to create the display is gs (ghostscript). The user can choose an alternative display tool by setting the SACPSVIEWER environmental variable. The default is:

```
setenv SACPSVIEWER "gs -sDEVICE=x11 -q -dNOPROMPT -dTTYPAUSE"
```

Possible values for SACPSVIEWER on different platforms may include:

```
Linux:    gs, gv, ggv, evince
Mac/OSX:  open, Preview, gs, gv
Sun/Solaris:  gs, gv
```

MARKPTP

SUMMARY

Measures and marks the maximum peak to peak amplitude of each signal within the measurement time window.

SYNTAX

```
MARKPTP {LENGTH v},{TO marker}
```

INPUT

LENGTH v: Change the length of the sliding window to v seconds.

TO marker: Define the first time marker in the header to store results. The time of the minimum is stored in this marker. The time of the maximum is stored in the next marker.

marker: T0|T1|T2|T3|T4|T5|T6|T7|T8|T9

DEFAULT VALUES

```
MARKPTP LENGTH 5.0 TO T0
```

DESCRIPTION

This command measures the times and the amplitude of the maximum peak-to-peak excursion of the data within the current measurement time window (see MTW.) The results are written into the header. The time of the minimum value (valley) is written into the requested marker. The time of the maximum value (peak) is written into the next marker. The peak-to-peak amplitude is written into USER0. The results are also written into the alphanumeric pick file if it is open (see OAFP.)

EXAMPLES

To set the measurement time window to be between the two header fields, T4 and T5, and the default sliding window length and marker:

```
u:  MTW T4 T5
u:  MARKPTP
```

To set the measurement time window to be the 30 seconds immediately after the first arrival, and the sliding window length to to 3 seconds, and the starting marker to T7:

```
u:  MTW A 0 30
u:  MARKP L 3. TO T7
```

HEADER CHANGES

Tn, USER0, KTn, KUSER0

SEE COMMANDS

MTW, OAPF

LATEST REVISION

May 15, 1987 (Version 10.2)

MARKTIMES

SUMMARY

Marks files with travel times from a velocity set.

SYNTAX

```
MARKTIMES {TO marker},{DISTANCE HEADER|v},
{ORIGIN HEADER|v|GMT time},{VELOCITIES v ...}
```

INPUT

TO marker: Define the first time marker in the header to store results. The time markers are incremented for each requested velocity.

marker: T0|T1|T2|T3|T4|T5|T6|T7|T8|T9

DISTANCE HEADER: Use the distance (DIST) from the header in the travel time calculations.

DISTANCE v: Use v as the distance in the travel time calculations.

ORIGIN HEADER: Use the origin time (O) in the header in the travel time calculations.

ORIGIN v: Use v as the offset origin time.

ORIGIN GMT time: Use the Greenwich mean time time as the origin time.

time: Greenwich mean time in the form of six integers: year, julian day, hour, minute, second, and millisecond.

VELOCITIES v ...: Set the velocity set to use in the travel time calculations. Up to 10 velocities may be entered.

ALTERNATE FORMS

UTC for Universal Time Coordinate may be used instead of GMT.

DEFAULT VALUES

```
MARKTIMES VELOCITIES 2.  3.  4.  5.  6.   DISTANCE HEADER ORIGIN HEADER
TO T0
```

DESCRIPTION

This command marks travel travel times in the header, given the origin time of the event, the epicentral distance, and an input velocity set. The following simple equation is used to estimate travel times.:

```
time(j) = origin + distance/velocity(j)
```

The results are written into the header in the requested time marker.

EXAMPLES

To use the default velocity set but force the distance to be 340 kilometers and the first marker to be T4:

```
u:  MARKTIMES DISTANCE 340. TO T4
```

To select a different velocity set:

```
u:  MARKT V 3.5 4.0 4.5 5.0 5.5
```

To set the origin time in GMT and store the results in T2:

```
u: MARKT ORIGIN GMT 1984 231 12 43 17 237 TO T2
```

HEADER CHANGES

Tn, KTn

LATEST REVISION

May 15, 1987 (Version 10.2)

MARKVALUE

SUMMARY

Searches for and marks values in a data file.

SYNTAX

```
MARKVALUE {GE v|LE v},{TO marker}
```

INPUT

GE v: Search for and mark the first data point that is greater than or equal to v.
LE v: Search for and mark the first data point that is less than or equal to v.
TO marker: Define the time marker in the header in which to store the result.
marker: T0|T1|T2|T3|T4|T5|T6|T7|T8|T9

DEFAULT VALUES

```
MARKVALUE GE 1 TO T0
```

DESCRIPTION

This command searches for the requested value in each data file and marks the time of the first occurance of that value. If a measurement time window has been defined (see MTW), only that portion of each data file is searched. Otherwise the entire file is searched. The results are written into the header in the requested time marker.

EXAMPLES

To search for the first data point whose value is at least 3.4 and to store the result in the header as T7:

```
u:  MARKVALUE GE 3.4 TO T7
```

To later perform that same search in the measurement time window 10 seconds long beginning at T4:

```
u:  MTW T4 0 10
u:  MARKVALUE
```

HEADER CHANGES

Tn, KTn

SEE COMMANDS

MTW

LATEST REVISION

May 15, 1987 (Version 10.2)

MAT

SUMMARY

Copy SAC workspace into Matlab and either execute a user-specified m-file or else get a Matlab prompt for interactive manipulation. The SAC workspace is updated with changes made to the data after the return from Matlab.

SYNTAX

```
MAT [mfile]
```

DESCRIPTION

The mat command allows processing of SAC data from within SAC using the Matlab (Version 5) engine and any user-written m-files. When this command is executed, the SAC workspace is copied into the following Matlab variables: SeisData --- an M-points by N-traces array of waveforms. SACdata --- an M-element structure array containing the header information from the SAC workspace. BlackBoard --- a structure array containing any blackboard variables.

SEISDATA

If the SAC data are time-domain, the SeisData array is real. Other wise it is complex. However, be aware that the default behavior of SAC's fft command is to produce transformed data in amplitude-phase format while in Matlab, the data will be treated as real-imaginary. The easiest way around that is to use the rlim option with SAC's fft.

You must return trace data from Matlab to SAC in the same domain as it was in before the mat command was executed. Otherwise, changes to the trace data made in Matlab will not be preserved. Also, you must not change the length of the traces in Matlab.

SACDATA

The SACdata structure array contains the following elements: times station event user descrip evsta llnl response trcLen

Most of these elements are themselves structures and their members are as follows:

times	**station**	**event**	**user**	**descrip**	**evsta**	**llnl**
delta	stla	evla	data(10)	iftype	dist	xminimum
b	stlo	evlo	label(3)	idep	az	xmaximum
e	stel	evel	iztype		baz	yminimum
o	stdp	evdp	iinst		gcarc	ymaximum
a	cmpaz	nzyear		istreg		norid
t0	cmpinc	nzjday		ievreg		nevid
t1	kstnm	nzhour		ievtyp		nwfid
t2	kcmpnm	nzmin		iqual		nxsize
t3	knetwk	nzsec		isynth		nysize
t4		nzmsec				

... continued on next page

times	station	event	user	descrip	evsta	llnl
t5		kevnm				
t6		mag				
t7		imagtyp				
t8		imagsrc				
t9						
f						
k0						
ka						
kt1						
kt2						
kt3						
kt4						
kt5						
kt6						
kt7						
kt8						
kt9						
kf						

response is a 10-element array, and trcLen is a scalar. Thus, to reference the begin time for the 10th trace in memory you would write: SACdata(10).times.b

To reference the first 4 characters of the station name for the first trace, you would write: SACdata(1).station.kstnm(1:4)

BLACKBOARD The BlackBoard variable is also a structure array which will be missing if you have no numeric or string black board variables in the SAC workspace. Otherwise there is an element for each black board variable. Each element is a structure containing a name and a value. You cannot create new black board variables in Matlab (If you do, the changes will not be saved). However, you can modify the ones passed from SAC to Matlab. So, if your Matlab script will create a number of output quantities that you want to store in SAC's blackboard, create the variables in SAC before executing the mat command.

MATLAB PATH

By default, the Matlab path available to you from within SAC will consist of the current directory, ~/matlab and the $MATLAB hierarchy. You can add an additional directory to the Matlab path from within SAC using the SETMAT command (SETMAT directoryName). Also, from within Matlab, you can modify the search path command using the path command. (Type help path for more information).

EXITING THE MATLAB INTERPRETER

There are two ways to exit the Matlab interpreter and return to SAC. You can type "exitmat" at the SACMAT» prompt. This will return you to SAC and leave the engine running. This can be useful if you intend to move back and forth between the two environments frequently, since there is no delay associated with starting the Matlab engine. However, a Matlab license will be tied up while you are running SAC and this may inconvenience other users who cannot start a session. To exit the Matlab interpreter and/or close the engine, type "closemat" at either the SACMAT» or the SAC> prompt.

HEADER CHANGES

Potentially all. User is responsible for consistency of changes.

EXAMPLE

Execute an m-file that converts the data to their absolute values. Assume the m-file is named absv.m and contains the one line SeisData=abs(SeisData):

```
u:  mat absv
```

NOTES

You may find it easier to develop a complex m-file directly from Matlab rather than from within the SAC-Matlab environment. The primary reasons are that there is no command line recall at the SACMAT» prompt and because SACMAT does not trap ^C (used to stop errant m-files in Matlab). The easiest way to do this is to load your data into SAC, start the Matlab engine with mat, and then type save. This will save the workspace in a file called matlab.mat. You may then start a normal matlab session, and type load. This will load matlab.mat and you may then develop your application within Matlab.

The entire range of plotting commands are available. However, if you execute your m-file from SAC (i.e. mat mfilename) Matlab will return to SAC immediately after executing the last command in the m-file. Therefore, if you want to look at your plots created in Matlab either execute the m-file from the Matlab command line, or execute a pause in your Matlab script:

```
plot(SeisData)
pause(10)
```

LATEST REVISION

Aug 9, 1997 (Version 00.56a)

MATHOP

SUMMARY

Controls Math Operator Precedence

SYNTAX

```
MATHOP NORMAL|MATH|FORTRAN|NONE|OLD
```

INPUT

NORMAL: Use Normal Math Operator Precedence
MATH: Same as Normal
FORTRAN: Same as Normal
NONE: Use No Operator Precedence
OLD: Same as None

DEFAULT VALUES

```
MATHOP NORMAL
```

DESCRIPTION

This command controls math operator prededence. Normally, multiplication and division have a higher predecence than addition and subtraction. Exponentiation has the highest precedence.

Older version of SAC (pre-101.6) used a math evaluation without operator precedence. Terms were evaluated in order where the earlier in the expression operators had higher precedence.

If matop is not called, the effect is the same as mathop normal. The function mathop has been introduced to deal with scripts and macros that were written using the mathop old precedence. Rather than change the coding, one simply enters mathop old before the lines with inline expressions.

EXAMPLES

With Operator Precedence:

```
SAC> mathop normal
SAC> evaluate 1+2*3
===> 7
SAC> evaluate 1+(2*3)
===> 7
```

Without Operator Precedence (as was true without mathop prior to v101.6):

```
SAC> mathop old
SAC> evaluate 1+2*3
===> 9
SAC> evaluate 1+(2*3)
===> 7
```

LATEST REVISION

Version 101.6 (new)

MERGE

SUMMARY

Merges (concantenates) a set of files to data in memory.

SYNTAX

```
MERGE {VERBOSE} {GAP ZERO|INTERP} {OVERLAP COMPARE|AVERAGE} {filelist}
```

INPUT

GAP ZERO | INTERP: How to handle data gaps ZERO - Fill with 0.0 amplitude INTERP - Interpolate, linear, within the data gap

OVERLAP COMPARE | AVERAGE: How to handle data overlaps COMPARE - Compare overlapping data points, exit on mismatch AVERAGE - Average overlapping data points

VERBOSE: Describe the merge details

filelist: A list of SAC binary data files. This list may contain simple filenames, full or relative pathnames, and wildcard characters. See the READ command for a complete description.

DESCRIPTION

The data in the files in this merge list is appended or concantenated to the data in memory. Each pair of files to be merged is checked to make sure they have the same sampling interval and station name. Any number of file, in any order are able to be merged. Data currently in memory and data identified in the merge command are merged together. If no data is specified with the merge command, data currently in memory will be merged.

EXAMPLES

To merge FILE3, FILE4, FILE1 and FILE2 into one file:

```
u:  READ FILE1 FILE2
u:  MERGE FILE3 FILE4
```

or

u: READ FILE1 FILE2 FILE3 FILE4 u: MERGE

or

u: DELETECHANNEL ALL u: MERGE FILE1 FILE2 FILE3 FILE4

To merge files for the same station, say ELKO.Z, from four different events each stored in a separate UNIX directory:

```
u:  READ data/event1/elko.z
u:  MERGE data/event2/elko.z data/event3/elko.z data/event4/elko.z
```

HEADER CHANGES

NPTS, DEPMIN, DEPMAX, DEPMEN, E

ERROR MESSAGES

- 1301: No data files read in.
- 1803: No binary data files read in.
- 1307: Illegal operation on spectral file
- 1306: Illegal operation on unevenly spaced file
- 1801: Header field mismatch:
- 9005: Amplitude mismatch

WARNING MESSAGES

- 1805: Time gap (zeros added):

SEE COMMANDS

READ

LATEST REVISION

Feb. 27, 2013 (Version 101.6)

MESSAGE

SUMMARY

Sends a message to the user's terminal.

SYNTAX

```
MESSAGE text
```

INPUT

text: Text of message to be sent. If there are blanks in the message it must be enclosed within single quotes.

DESCRIPTION

This command is useful within macro files to send status or informational messages to the user while the macro file is executing. It is not particularly useful in the interactive mode (unless you like to talk to yourself.)

EXAMPLES

To send a message without any blanks:

```
u: MESSAGE FINISHED
s: FINISHED
```

To send a message with blanks, you must use single or double quotes:

```
u: MESSAGE 'Job has finished.'
s: Job has finished.
```

LATEST REVISION

January 8, 1983 (Version 8.0)

MTW

SUMMARY

Determines the measurement time window for use in subsequent measurement commands.

SYNTAX

```
MTW {ON|OFF|pdw}
```

INPUT

{ON}: Turn measurement time window option on but don't change window values.

OFF: Turn measurement time window off. Measurements are done on the entire file.

pdw: Turn measurement time window on and set window values to a new "partial data window." A pdw consists of a starting and a stopping value of the independent variable, usually time, which defines the desired window of data that you wish to make measurements on. See the CUT command for a complete explanation of how to define and use a pdw. Some examples are given below.

DEFAULT VALUES

```
MTW OFF
```

DESCRIPTION

When this option is on, measurements are made on the data within the window only. When this option is off, measurements are made on the entire file. This option currently applies to the MARKPTP and MARKVALUE commands only. Others measurement commands will be added as needed.

EXAMPLES

Some examples of pdw are given below:

```
B 0 30: First 30 secs of the file.
A -10 30: From 10 secs before to 30 secs after first arrival.
T3 -1 T7: From 1 sec before T3 time pick to T7 time pick.
B N 2048: First 2048 points of file.
30.2 48: 30.2 to 48 secs relative to file zero.
```

SEE COMMANDS

CUT, MARKPTP, MARKVALUE

LATEST REVISION

May 15, 1987 (Version 10.2)

MUL

SUMMARY

Multiplies each data point by a constant.

SYNTAX

```
MUL {v1 {v2 ...  vn} }
```

INPUT

v1: Constant to multiply first file by.
v2: Constant to multiply second file by.
vn: Constant to multiply nth file by.

DEFAULT VALUES

```
MUL 1.
```

DESCRIPTION

This command will multiply each element of each data file in memory by a constant. The constant may be the same or different for each data file. If there are more data files in memory than constants, then the last constant entered is used for the remainder of the data files in memory.

EXAMPLES

To multiply each element of F1 by 5.1 and each element of F2 and F3 by 6.2:

```
u:  READ F1 F2 F3
u:  MUL 5.1 6.2
```

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES

- 1301: No data files read in.
- 1307: Illegal operation on spectral file

LATEST REVISION

January 8, 1983 (Version 8.0)

MULF

SUMMARY

Multiplies a set of files by the data in memory.

SYNTAX

```
MULF {NEWHDR ON|OFF} filelist
```

INPUT

NEWHDR ON|OFF: By default, the resultant file will take its header field from the original file in memory. Turning NEWHDR ON, causes the header fields to be taken from the new file in the filelist.

filelist: A list of SAC binary data files. This list may contain simple filenames, full or relative pathnames, and wildcard characters. See the READ command for a complete description.

DESCRIPTION

This command can be used to multiply a single file by a set of files or to multiply one set of files by another set. An example of each case is presented below. The files must be evenly spaced and should have the same sampling interval and number of points. This last two restrictions can be eliminated using the BINOPERR command. If there are more data files in memory than in the filelist, then the last file in the filelist is used for the remainder of the data files in memory.

EXAMPLES

To multiply three files by a single file:

```
u:  READ FILE1 FILE2 FILE3
u:  MULF FILE4
```

To multiply two files by two other files:

```
u:  READ FILE1 FILE2
u:  MULF FILE3 FILE4
```

HEADER CHANGES

If NEWHDR is OFF (the default) the headers in memory are unchanged). If NEWHDR is ON, the headers are replaced with the headers from the files in the filelist.

DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES

- 1301: No data files read in.
- 1803: No binary data files read in.
- 1307: Illegal operation on spectral file

- 1306: Illegal operation on unevenly spaced file
- 1801: Header field mismatch:
  - either the sampling interval or the number of points are not equal. - can be controlled using the BINOPERR command.

WARNING MESSAGES

- 1802: Time overlap:
  - the file multiplication is still performed.

SEE COMMANDS

READ, BINOPERR

LATEST REVISION

May 26, 1999 (Version 0.58)

MULOMEGA

SUMMARY

Performs differentiation in the frequency domain.

DESCRIPTION

This command multiplies each point of a spectral file by its frequency given by:

```
OMEGA = 2.0 * PI * FREQ
```

This is analogous to differentiating the equivalent time series file. The spectral file can in either amplitude-phase or real-imaginary format.

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

LATEST REVISION

May 15, 1987 (Version 10.2)

PLOTC

SUMMARY

Annotates SAC plots and creates figures using cursor.

SYNTAX

```
PLOTC {REPLAY|CREATE} {FILE|MACRO filename},
{BORDER {ON|OFF}}
```

INPUT

REPLAY: Replay or replot an existing file or macro. The difference between a file and a macro is described below.
CREATE: Create a new file or macro.
FILE {filename}: Replay or create a file. The previous file is used if filename is omitted.
MACRO {filename}: Replay or create a macro.
BORDER {ON}: Turn border around plot on.
BORDER OFF: Turn border around plot off.

DEFAULT VALUES

```
PLOTC CREATE FILE OUT BORDER ON
```

DESCRIPTION

This command lets you annotate SAC plots and create figures for meetings and reports. A device with cursor capability is required. You "build" a figure by placing objects and text on the terminal screen. The cursor position determines where an object will be drawn and the character typed determines what object is to be drawn. Objects include circles, rectangles, n-sided polygons, lines, arrows, and arcs. Several ways of placing text are included.

This command creates two different type of output files, simple files and macro files. Both are alphanumeric files that can be changed using an editor. They contain the history of the cursor responses and locations from a single execution of the PLOTC command. A macro file, once created, can be used in more than one plot or figure. It can be scaled in size and can also be rotated. A simple PLOTC filename is the name you request with a ".PCF" appended to it. A macro file has a ".PCM" appended to its' name. This provides a check for SAC when you request a particular file and also lets you distinguish these files in your directories.

When you create a new file or macro, SAC draws a rectangle on the screen showing you the allowable area for the figure. It then turns the cursor on in the middle of this area. You move the cursor to the desired location and type a character representing the object you want drawn or the action you want to occur.

There are two types of cursor options, action and parameter-setting. The action options do something (draw a polygon, place text, etc.) How they do that action is based in part upon the current values of the parameter-setting options (how many sides on the polygon, what size text to draw, etc.) This distinction is similiar to the idea of action and parameter-setting commands in SAC itself. The tables on the following pages list the action and parameter-setting options.

When you replay a file or macro, the figure is redrawn on the terminal screen and then the cursor is turned on. You may then add to the file or macro as if you were creating it for the first time. When you

have created a figure that you want to send to a different graphics device, use the BEGINDEVICES command to temporarily turn off the terminal and turn on the other device. Then simply replay the file.

To annotate a SAC plot, execute the VSPACE command to set up the correct aspect ratio (see below), execute the BEGINFRAME command to turn off automatic framing, execute the desired SAC plot command, execute the PLOTC command (in create or replay mode), and then execute the ENDFRAME command to resume automatic framing.

EXAMPLES

An example of the use of PLOTC to add annotation to a standard SAC plot is the figure in the BANDPASS command description of this manual. The commands used to create that figure are given below with comments given in parentheses:

```
u:  FG IMPULSE NPTS 1024       \ \ (generate filter response)
u:  LOWPASS C2 NPOLES 7 CORNER 0.2 TRANBW 0.25 A 10
u:  FFT
u:  AXES ONLY LEFT BOTTOM      \ \ (set up desired plot options)
u:  TICKS ONLY LEFT BOTTOM
u:  BORDER OFF
u:  FILEID OFF
u:  QDP OFF
u:  VSPACE 0.75              \ \ (see discussion below)
u:  BEGINFRAME               \ \ (turn off automatic framing)
u:  PLOTSP AM LINLIN         \ \ (plot filter response)
u:  PLOTC CREATE FILE BANDPASS       \ \ (create annotation)
u:  ENDFRAME                 \ \ (turn automatic framing back on)
```

PLOTSP was used to produce the curve of the filter response and the two axes. PLOTC was used interactively to produce the annotation (i.e., the lines, arrows, and labels.) The viewspace command constrains the plot be the largest enclosed area of the graphics screen that has an (y:x) aspect ratio of 3:4. This is necessary so that when the output is later sent to the SGF device which also has a 3:4 aspect ratio, everything will be plotted correctly. At this point you would have a file called "BANDPASS.PCF" containing the annotations for this plot. To write this annotated plot to the SAC graphics file:

```
u:  BEGINDEVICES SGF    \ \ (select the SGF device)
u:  BEGINFRAME          \ \ (turn off automatic framing)
u:  PLOTSP              \ \ (plot filter response again)
u:  PLOTC REPLAY        \ \ (replay the annotation)
u:  ENDFRAME            \ \ (turn automatic framing back on)
```

A SAC graphics file will be created containing the annotated plot. Two examples (one somewhat frivolous) of the use of PLOTC to create figures and viewgraphs are given on the following pages. The replay files are also shown. (It is an exercise left to the reader to determine which of the examples is frivolous.)

1. The circle and sector opcodes only produce correct output when you have set the viewspace to a square one (VSPACE 1.0).
2. All all of the opcodes except text are scaled to fit in the graphics window.

The text sizes aren't currently scaled. This creates a problem when you create a figure and want to enclose some text in a rectangle or a circle. In this case the graphics window must be the same size as the output page in order to avoid misalignment.

This can be achieved by using the WINDOW command to set the horizontal (x) size of the window to be 0.75 and the vertical (y) size to be 0.69. For example:

```
WINDOW_ 1 X 0.05 0.80 Y 0.05 0.74
```

SEE COMMANDS

VSPACE, BEGINDEVICES, BEGINFRAME, ENDFRAME

LATEST REVISION

October 11, 1984 (Version 9.1)

TABLE: Action Options

```
char  meaning
A     Draw an arrow from ORIGIN to CURSOR.
B     Draw border tick marks around plot region.
C     Draw a circle centered at ORIGIN through CURSOR.
D     Delete last action option from replay file.
G     Set ORIGIN and make it global.
L     Draw a line from ORIGIN to CURSOR.
M     Invoke a macro at CURSOR.
Enter name of macro, scale factor, and rotation angle.
Previous values are used if omitted.
Defaults are OUT, 1., and 0.
O     Set ORIGIN at CURSOR.
N     Draw an n-sided polygon centered at ORIGIN
with one vertex at CURSOR.
Q     Quit PLOTC_.
R     Draw a rectangle with opposing corners
at ORIGIN and CURSOR.
S     Draw a sector of a circle centered at ORIGIN through
CURSOR  Move CURSOR to define the sector angle.
Type an S to get the sector whose angle is less than
 180 degrees or C to get its' complement.
T     Place a single line of text at cursor.
Text is ended by a carriage-return.
U     Place multiple lines of text at cursor.
Text is ended by a blank line.
```

Notes: -- CURSOR is the current cursor location -- ORIGIN is normally the last cursor location -- The G option forces ORIGIN to remain fixed -- The O option allows ORIGIN to move again

NULL

SUMMARY

Controls the plotting of null values.

SYNTAX

```
NULL {ON|OFF|value}
```

INPUT

ON: Turns the NULL option on for plotting.
OFF: Turns the NULL option off for plotting.
value: Sets the value of a NULL to be value.

DEFAULT VALUES

```
NULL OFF
```

DESCRIPTION

Many times in a data set, when there are gaps in the data, no data is available. In many cases the data has been set to a predefined value. Typical values are 0.0, -1.0, -99. Usually the user will not want these values displayed on plots. The NULL command allows the user to define the NULL value and not connect a line through these data points. To set the NULL value to -1.0 and enable the NULL option during plotting:

```
u: NULL ON -1.0
```

LATEST REVISION

March 20, 1992 (Version 10.6e)

OAPF

SUMMARY

Opens a alphanumeric pick file.

SYNTAX

```
OAPF {STANDARD|NAME},{file}
```

INPUT

STANDARD: Use the standard file id when writing picks. The standard id consists of the event name, the station name, and the component azimuth and incident angles from the SAC header.

NAME: Use the name of the SAC data file instead of the standard id.

file: The name of the alphanumeric pick file to open. If a file by that name already exists, it is opened and new picks are added at the bottom.

DEFAULT VALUES

```
OAPF STANDARD APF
```

DESCRIPTION

The alphanumeric pick file can be used like a simple data base for picks produced by the automatic picker (APK) and manual pick plot (PLOTPK) command. One line is written for each pick. Each normal line in one of these files consists of the file id, the pick id, the time of the pick, the amplitude of the pick, and some formatting information. These lines are 80 characters long. The file id is the standard one consisting of selected fields from the header as mentioned above or the name of file. The time of the pick is either the GMT time or the offset time. This depends upon the options specified in the commands generating the pick, such as APK or PLOTPK. This leads to four distinct formats, designated by a different character in column 79. Some lines, such as those from waveform and peak-to-peak picks, contain additional fields after column 80. The maximum length of a line is 200 characters. The specific formats of these various lines are shown in the tables on the following pages.

ERROR MESSAGES

- 1903: Can't close previous card image pick file.
- 1902: Can't open card image pick file
  - Probably an illegal character in filename.

SEE COMMANDS

PLOTPK, APK, CAPF

LATEST REVISION

January 8, 1983 (Version 8.0)

FILE FORMAT

Standard file id and GMT time options:

```
column  format  contents
01      a16     event name
17      a8      station name
25      f7.2    component azimuth
32      f7.2    component incident angle
39      a4      pick id
43      i5      year of pick
48      i3      julian day
51      i3      hours
54      i3      minutes
57      f6.2    seconds
63      g10.4   amplitude of pick
74      a1      source of pick
= "A" if an automatic pick (APK)
= "M" if a manual pick (PLOTPK)
79      a1      "B"
```

File name and GMT time options:

```
column  format  contents
01      a32     file name
39      a4      pick id
43      i5      year of pick
48      i3      julian day
51      i3      hours
54      i3      minutes
57      f6.2    seconds
63      g10.4   amplitude of pick
74      a1      source of pick
79      a1      "C"
```

Standard file id and offset time options:

```
column  format  contents
01      a16     event name
17      a8      station name
25      f7.2    component azimuth
32      f7.2    component incident angle
39      a4      pick id
 53      g10.4   offset time of pick
63      g10.4   amplitude of pick
74      a1      source of pick
79      a1      "D"
```

File name and offset time options:

```
column  format  contents
01      a32     file name
```

```
39      a4      pick id
53      g10.4   offset time of pick
63      g10.4   amplitude of pick
74      a1      source of pick
79      a1      "E"
```

For waveform picks, the pick time and amplitude is that of the first zero crossing. Additional waveform pick fields are:

```
column  format  contents
080     f6.3    incremental time of first peak
086     f6.3    incremental time of second crossing
092     f6.3    incremental time of second peak
098     f6.3    incremental time of third crossing
105     g10.4   amplitude of first peak
116     g10.4   amplitude of second peak
```

For peak-to-peak picks, the pick time and amplitude is that of the first peak. Additional peak-to-peak pick fields are:

```
column  format  contents
80      f6.3    incremental time of second peak
87      g10.5   amplitude of second peak
```

OHPF

SUMMARY

Opens a HYPO formatted pick file.

SYNTAX

```
OHPF {file}
```

INPUT

file: Name of file to open. If a file by that name already exists, it is opened and new picks are added at the bottom.

DEFAULT VALUES

```
OHPF HPF
```

DESCRIPTION

The HYPO pick file generated by SAC can be used as input to HYPO71 and similiar event location programs. Picks from the automatic picker (APK) and manual pick plot (PLOTPK) commands are written into this file once open. This file can be closed using the CHPF command. Opening of a new HYPO pick file automatically closes the previously open one. Opening an existing HYPO pick file automatically deletes the last line of the file, which should be the instruction card "10" that indicates the end of the HYPO input file. Terminating SAC also automatically closes any open pick files. Event delimiters can be written into a HYPO pick file with the WHPF command. See the reference for details on the format of each card.

ERROR MESSAGES

- 1901: Can't open HYPO pick file
  - Probably an illegal character in filename.
  - Occasionally a system error. If error persists contact the programmer.

SEE COMMANDS

APK, PLOTPK, WHPF, CHPF W.H.K. Lee and J.C. Lahr; HYPO71 (Revised): A Computer Program for Determining Hypocenter, Magnitude, and First Motion Pattern of Local Earthquakes; U. S. Geological Survey report 75-311.

LATEST REVISION

March 20, 1992 (Version 10.6e)

PAUSE

SUMMARY

Sends a message to the terminal and pauses.

SYNTAX

```
PAUSE {MESSAGE text},{PERIOD {ON|OFF|v}}
```

INPUT

MESSAGE text: Text of message to send to terminal before pausing. Enclose text in single quotes if it contains any blanks.

PERIOD {ON}: Turn period option on but don't change length of pause. When this option is on, SAC pauses for a certain period of time and then resumes execution automatically.

PERIOD OFF: Turn period option off. When this option is off, SAC pauses until you type a carriage-return.

PERIOD v: Turn period option on and change length of pause to v seconds.

DEFAULT VALUES

```
PAUSE MESSAGE 'Pausing' PERIOD OFF
```

DESCRIPTION

This command lets you temporarily suspend the execution of a SAC macro file. When this command is executed, SAC sends a message to your terminal, pauses, and then either waits until you type a carriage return or waits for a specified period of time. This might be of interest if you wanted to study the output of a particular command before allowing the next command in the macro to be executed. It is of particular interest in the preparation of macro files to be used in demonstrations and tutorials. The ECHO command is also useful for preparing such demonstrations.

SEE COMMANDS

ECHO

LATEST REVISION

May 15, 1987 (Version 10.2)

PICKAUTHOR

SUMMARY

Tell sac to read author list (and possibly phase pick information) from a user-defined preferences file, or interactively enter author list on the PICKAUTHOR command line.

SYNTAX

```
PICKAUTHOR author1 {author2 author3 ...  }
PICKAUTHOR FILE {filename}
PICKAUTHOR PHASE {filename}
```

INPUT

authorlist: sac uses the input to create the author list.

FILE: if the FILE option is used, sac will read the author list from the preferences file. If a filename is given on the command line, sac will read the specified file, else it will read the most recently entered file name from a previous call to PICKAUTHOR. If no filename was ever entered, sac will look for SACAUX/csspickprefs.

PHASE: this option behaves essencially like the FILE option with the added benefit of having sac read specific header variable information as well.

DEFAULT VALUES

```
PICKAUTHOR FILE
```

DESCRIPTION

PICKAUTHOR is provided as a means to override the preferences file on the command line. It can be used to provide a prioritized list of authors at the command line, or to redirect SAC from one preferences file to another. For more on the preferences files, see PICKPREFS and READCSS.

Note: If the user alters the preference settings while data is in the data buffers, the picks in the SAC data buffer (the buffer visible to the user through LISTHDR and CHNHDR etc.) may be modified. Eg. if the author list is "john rachel michael" and some files are read with the READCSS command some arrivals may be read with author = michael. (The user will proably not be aware of who the author is for a given pick, because the author field in CSS does not appear in the SAC format.) If the user later uses the PICKAUTHOR command to change the author list to "peter doug rachel", then on a READCSS MORE command, no arrivals with author = michael will be read from the data files, and the file already in memory will lose the picks which have michael as the author. The user may not know why seemingly random picks have disappeard. For an explanation, type HELP PICKPREFS.

SEE COMMANDS

PICKPREFS, READCSS, PICKPHASE

PICKPHASE

SUMMARY

Tell sac to read phase pick information (and possibly the author list) from a user-defined preferences file, or interactively enter phase pick information on the PICKPHASE command line.

SYNTAX

```
PICKPHASE header phase author {header phase author ...  }
PICKPHASE FILE {filename}
PICKPHASE AUTHOR {filename}
```

INPUT

header: name of a header variable: t0 - t9.

phase: name of phase of pick desired for the given header variable.

author: name of the author desired for the given header or hyphen, "-", to tell sac to use the author list.

FILE: if the FILE option is used, sac will read the phase pick info. from the preferences file. If a filename is given on the command line, sac will read the specified file, else it will read the most recently entered file name from a previous call to PICKPHASE. If no filename was ever entered, sac will look for SACAUX/csspickprefs.

PHASE: this option behaves essencially like the FILE option with the added benefit of having sac read the author list as well.

DEFAULT VALUES

```
PICKPHASE FILE
```

DESCRIPTION

PICKPHASE is provided as a means to override the preferences file on the command line. It can be used to provide specific header/phase/author information at the command line, or to redirect SAC from one preferences file to another. For more on the preferences files, see PICKPREFS and READCSS.

Note: If the user alters the preference settings while data is in the data buffers, the picks in the SAC data buffer (the buffer visible to the user through LISTHDR and CHNHDR etc.) may be modified. Eg. if the allowed phases include pP and PKiKP when some SAC files are read with the READ command which have some pP picks or some PKiKP picks these picks would be present in the Tn markers. If PICKPHASE is later used to remove pP and PKiKP from the allowed phases before the next READCSS MORE call, then pP and PKiKP arrivals will not be read from the CSS files, and the pP and PKiKP picks in the existing data will be removed from the Tn markers. For an explanation, type HELP PICKPREFS.

SEE COMMANDS

PICKPREFS, READCSS, PICKAUTHOR

PICKPREFS

SUMMARY

The PICKPREFS command is used to control the way that SAC manages and or loads picks from a variety of input data formats (e.g., CSS, GSE, SUDS etc...) into the time marker variables T0 to T9 (aka. Tn). When this option is OFF (the default), the picks loaded into the time markers correspond to the first picks that SAC finds in the input data. If this options is ON, SAC will use the preferences file described in the READCSS command.

Note: Because of the structured nature of the preferences file (which aligns specific phases with specific marker variables), and the free flowing nature of the interactions without the preferences, a change in the PICKPREFS in the middle of processing can change the picks in the datafiles. See the descrition below for details.

SYNTAX

```
PICKPREFS ON
PICKPREFS OFF
PICKPREFS
```

INPUT

ON: instructs SAC to pass arrivals from the CSS buffer through the preferences file on its way to the SAC buffer. This is useful in macros that require specific arrivals to be in specific Tn header variables.

OFF: instructs SAC to bypass the preferences file and load the first 10 picks it encounters for a given file. This is the default. It allows the user to be aware of picks s/he may not be aware of with the PICKPREFS ON.
If now option is provided on the commandline, PICKPREFS will toggle the use of preferences file ON or OFF.

DEFAULT VALUES

```
PICKPREFS OFF
```

DESCRIPTION

Since version 0.58, sac2000 has had two different header buffers: one formatted according to the SAC file format, and one formatted according to the relational CSS 3.0 file format. Adding the CSS data buffer has made it easier to read relational formats such as CSS, GSE, and SUDS. Having two buffers has allowed the process management commands: COMMIT, ROLLBACK, and RECALLTRACE.

One drawback of having these two buffers is the complexity of moving arrivals from the dynamic CSS arrival table to the rather ridged T0 - T9 picks in the SAC format. This problem was solved in version 0.58 by setting in place a preferences file called csspickprefs. This file is in the aux directory and can be overridden by writing one of your own. For more information about how to use the csspickprefs file, use HELP READCSS. For details on how to override the default preferences file, use HELP PICKAUTHOR or HELP PICKPHASE.

The drawback of using the preferences file was that it would only accept phase names and/or author names listed in the preferences file or those entered at the command line with PICKPHASE or PICKAUTHOR. In other words, if a CSS data file from either a flat-file, or the Oracle database has a pP arrival, and pP is not specified in the preferences file, the user would never know that the pP is there. The pP pick will be read into the CSS data buffer in SAC, but it will not be transfered to the

SAC data buffer, and will not participate in any of the SAC commands. It may be written out by the WRITECSS command, or it may get flushed out during a COMMIT command, and be lost entirely.

The solution we have worked out is to allow the user to bypass the preferences file. In version 0.59, the default is to read the first 10 available picks from the CSS buffer directly into the SAC buffer whenever data is transferred from the one to the other. By use of this new command, PICKPREFS, the user can tell SAC to use the preferences file. This is useful if the user has a macro which expects to find a specific phase in a specific Tn header variable.

Data is transfered from the CSS buffer to the SAC buffer on any READCSS, READGSE, or READSUDS command, as well as COMMIT, ROLLBACK, and RECALLTRACE. COMMIT, ROLLBACK, or RECALLTRACE get called by default by any of the following commands:

- any command that writes data (WRITE, WRITECSS, WRITEGSE, etc.)
- any command that reads data with the MORE option specified
- the SORT command.

If the user alters PICKPREFS and or the preference settings while data is in the data buffers, the picks in the SAC buffer may be modified. Eg. if PICKPREFS is OFF (the default) when some SAC files are read with the READ command they may have some pP picks or some PKiKP picks which would be present in the Tn markers. If PICKPREFS is later turned OFF, for a READCSS, if pP and/or PKiKP aren't listed in the preferences file, then pP and PKiKP arrivals will not be read from the CSS files, and the pP and PKiKP picks in the existing data will be removed from the Tn markers.

SEE COMMANDS

READCSS, READDB, PICKAUTHOR, PICKPHASE, COMMIT, ROLLBACK, RECALLTRACE

PICKS

SUMMARY

Controls the display of time picks on most SAC plots.

SYNTAX

```
PICKS {ON|OFF} {pick type},{WIDTH v},{HEIGHT v}
```

where pick is one of the following

```
O|A|F|Tn, n=0...9
```

where type is one of the following

```
VERTICAL|HORIZONTAL|CROSS
```

INPUT

PICKS ON: Turn on pick display.

PICKS OFF: Turn off pick display.

pick: The name of a SAC time pick header variable: ,BREAK O|A|F|Tn, n=0...9

VERTICAL: A vertical line at time pick. Pick id at bottom right of line.

HORIZONTAL: A horizontal line at the data point nearest the time pick. Pick id is placed above or below the line.

CROSS: A vertical line at the time pick and a horizontal line at the nearest data point.

WIDTH v: Change the width of the pick display to v.

HEIGHT v: Change the height of the pick display to v. Height and width refer to HORIZONTAL and CROSS only and are in fractions of full plot size.

DEFAULT VALUES

```
PICK ON WIDTH 0.1 HEIGHT 0.1; all pick types are VERTICAL.
```

DESCRIPTION

This command controls the display of time pick information on most SAC plots. These time picks identify previously defined times of interest such as phase arrivals, event origin, etc. When this display is on, each defined time pick is displayed on the plot at the time of the pick with a time pick id near the line. The time pick id is a header variable 8 characters in length. KA, KF, KO, and KTn are the time pick ids for A, F, O, and Tn respectively. If the time pick id is not defined, the name of the pick itself is used. Each pick may be displayed as a vertical line, a horizontal line, or a cross.

EXAMPLES

To display time picks T4, T5, and T6 as crosses and to change the height and width of the crosses:

```
u:  PICKS T4 C T5 C T6 C W 0.3 H 0.1
```

Other time pick display types will remain unchanged.

LATEST REVISION

January 8, 1983 (Version 8.0)

PLABEL

SUMMARY

Defines general plot labels and their attributes.

SYNTAX

```
PLABEL {n} {ON|OFF|text},{SIZE size},
{BELOW|POSITION x y {a}}
```

where size is one of the following:

```
TINY|SMALL|MEDIUM|LARGE
```

INPUT

n: Plot label number. If omitted, the previous label number is incremented by one.
ON: Turn this plot label on.
OFF: Turn this plot label off.
text: Change text of plot label. Also turns plot label on.
SIZE size: Change the plot label size.
TINY: Tiny text size has 132 characters per line.
SMALL: Small text size has 100 characters per line.
MEDIUM: Medium text size has 80 characters per line.
LARGE: Large text size has 50 characters per line.
BELOW: Position this label "below" the previous label.
POSITION x y a: Define a specific position for this label. The range
of the positions are: 0. to 1. for x and 0. to the maximum viewspace (normally 0.75) for y. a is the angle of the label in degrees clockwise from horizontal.

DEFAULT VALUES

Default size is small.

Default position for label 1 is 0.15 0.2 0.

Default position for other labels is below previous label.

DESCRIPTION

This command lets you define general purpose plot labels for subsequent plot commands. You can define the location and size of each label. The text quality and font used can be set using the GTEXT command. You can also generate a title and axes labels using the TITLE, XLABEL, and YLABEL commands.

EXAMPLES

The following commands would generate a four line label in the upper left hand corner of subsequent plots:

```
u:  PLABEL 'Sample seismogram' POSITION .12 .5
u:  PLABEL 'from earthquake'
u:  PLABEL 'on January 24, 1980'
u:  PLABEL 'in Livermore Valley, CA'
```

An additional tiny label could be placed in the lower left hand corner:

```
u:  PLABEL 5 'LLNL station: CDV' S T P .12 .12
```

SEE COMMANDS

GTEXT, TITLE, XLABEL, YLABEL

LATEST REVISION

July 22, 1991 (Version 9.1)

PLOT

SUMMARY

Generates a single-trace single-window plot.

SYNTAX

```
PLOT {PRINT {pname} }
```

INPUT

PRINT {pname}: Prints the resulting plot to the printer named in pname, or to the default printer if pname is not used. (This makes use of the SGF capability.)

DESCRIPTION

Each data file is displayed in a separate plot. The total size of the plot is determined by the current viewport (see XVPORT and VPORT.) The y axis limits for each plot can be scaled to the data file's extrema or they can have fixed limits. See the YLIM command for details. The x axis limits are controlled by the XLIM command. A user controllable file identification (see FILEID) is generated for each file in the plot. Time picks can be displayed (see PICKS).

If you are plotting to a graphics terminal or workstation, SAC will pause between each plot to give you an opportunity to examine the plot. It will type "Waiting" in the text area and wait for your response. You can type a carriage-return to see the next plot, the keyword "go" (or "g") to plot the remainder of the files without pausing, or the keyword "kill" (or "k") to not plot the remainder of the files at all.

ERROR MESSAGES

- 1301: No data files read in.

SEE COMMANDS

XVPORT, YVPORT, XLIM, YLIM, FILEID, PICKS, FILENUMBER

LATEST REVISION

January 8, 1983 (Version 8.0)

PLOT1

SUMMARY

Generates a multi-trace multi-window plot.

SYNTAX

```
[P]LOT[1] {ABSOLUTE|RELATIVE},{PERPLOT {ON|OFF|n}} {PRINT {pname} }
```

INPUT

ABSOLUTE: Plot files treating time as an absolute. Files with different begin times will be shifted relative to each other.

RELATIVE: Plot all files relative to their begin time.

PERPLOT {ON}: Plot n files per frame. Use old value for n.

PERPLOT OFF: Plot all files on one frame.

PERPLOT n: Plot n files per frame.

PRINT {pname}: Prints the resulting plot to the printer named in pname, or to the default printer if pname is not used. (This makes use of the SGF capability.)

ALTERNATE FORMS

PERPLOT ALL has the same meaning as PERPLOT OFF.

DEFAULT VALUES

```
PLOT1 ABSOLUTE PERPLOT OFF
```

DESCRIPTION

Each data file shares a common axis in the x direction, but each has a separate subplot region in the y direction. The total size of the plot is determined by the current viewport (see XVPORT and YVPORT.) The size of each subplot is determined by this viewport and the number of files plotted on each frame. The y axis limits for each subplot can be scaled to that data file's extrema or they can have fixed limits. See the YLIM command for details. The x axis limits can also be fixed (see XLIM) or scaled to the data. There are two types of x axis scaling for this type of plot: relative and absolute. In absolute scaling the x axis limits become the smallest minimum and the largest maximum for the active memory files. Time differences measured between points on different subplots will be correct. In relative scaling mode, the x axis will run from zero to the maximum time differential (i.e., the maximum difference between end time and begin time) for the active memory files. Each file will be plotted from the left edge of the plot, corresponding to zero on the x axis. The actual value corresponding to this zero for each file will be given below the name of the file. This type of scaling is useful if you are cutting the files relative to some time pick, say the first arrival time. It is then easy to see the similarities or differences between the wave forms of each file. A user controllable file identification (see FILEID) is generated for each file in the plot. Time picks can be displayed (see PICKS).

EXAMPLES

The zero time (KZDATE and KZTIME) has been set to the event origin time:

```
SAC> READ *V
ELK.V KNB.V LAC.V MNV.V
SAC> CUT -5 200
SAC> READ *V
ELK.V KNB.V LAC.V MNV.V
SAC> FILEID LOCATION UL TYPE LIST KSTCMP
SAC> TITLE 'Regional earthquake:  &1,KZTIME&  &1,KZDATE&'
SAC> QDP 2000
SAC> P1
```

Note the use of a UNIX wildcard character in the READ command, the echoing of the filelist by SAC, the specification of a special file id, and the evaluation of several header variables to create the title.

ERROR MESSAGES

- 1301: No data files read in.

SEE COMMANDS

XLIM, YLIM, FILEID, PICKS, FILENUMBER

LATEST REVISION

January 8, 1983 (Version 8.0) Text altered in August 2011.

PLOT2

SUMMARY

Generates a multi-trace single-window (overlay) plot.

SYNTAX

```
[P]LOT[2] {ABSOLUTE|RELATIVE} {PRINT {pname} }
```

INPUT

ABSOLUTE: Plot files treating time as an absolute. Files with different begin times will be shifted relative to the first file.

RELATIVE: Plot each file relative to it's own begin time.

PRINT {pname}: Prints the resulting plot to the printer named in pname, or to the default printer if pname is not used. (This requires that program sgftops is in the path.)

DEFAULT VALUES

```
P2 ABSOLUTE
```

DESCRIPTION

All files in the data file list are plotted in the same plot window. An optional legend containing the plot symbol and file name can be generated. Fixed x and y axis limits may be defined before using this command. See the XLIM and YLIM commands. The plot is sized to the extrema of the entire file list if fixed limits are not requested. The location of the legend is controlled by the FILEID command.

Unlike PLOT and PLOT1, PLOT2 will plot spectral data. Real/Imaginary data is plotted as Real vs. Frequency. Amplitude/Phase data is plotted as Amplitude vs. Frequency. Imaginary and Phase information are ignored. Spectral data is always plotted in relative mode. Note that in the fequency domain,b, e, and delta are reset to 0, the Nqquist frequency, and df resjpectively. The header values depmin and dapmax are not changed. As with PLOTSP, if XLIM is off, the plot starts at DF=DELTA rather than 0. If XLIM or YLIM were changed before transferring to the frequency domain, it is best to enter XLIM off and YLIM off before calling PLOT2.

Note: If for some reason, the user has both time-series data and spectral data in memory at the same time and does not elect to use the RELATIVE option, the time-series files will be plotted ABSOLUTE and the spectral files will be plotted RELATIVE. Relative mode means relative to the first file. So the order of the files in memory will effect the relation of the plots with respect to each other.

EXAMPLES

The commands used to generate the example plot are given below:

```
u:  READ MNV.Z.AM KNB.Z.AM ELK.Z.AM
u:  XLIM 0.04 0.16
u:  YLIM 0.0001 0.006
u:  LINLOG
u:  SYMBOL 2 INCREMENT
```

```
u:  TITLE 'Rayleigh Wave Amplitude Spectra for NESSEL'
u:  XLABEL 'Frequency (Hz)'
u:  PLOT2
u:  FFT
u:  XLIM off YLIM off
u:  line increment list 1 3
u:  PLOT2 print
```

ERROR MESSAGES

- 1301: No data files read in.

SEE COMMANDS

XLIM, YLIM, FILEID, FILENUMBER

LATEST REVISION

April 11, 2010 (Version 101.4)

PLOTALPHA

SUMMARY

Reads alphanumeric data files on disk into memory and plots the data to the current output device. This differs from readalpha followed by plot because it allows you to plot a label with each data point.

SYNTAX

```
READALPHA {options}  {filelist}
```

where options is one or more of the following:

```
MORE
DIR CURRENT|name
FREE|FORMAT text
CONTENT text
PRINT {printer}
ALL options MUST preceed any element in the filelist.
```

The last two options may also be placed on the first line of file itself.

INPUT

MORE: Append the new data files after the old ones in memory. If this option is missing, the new data replaces the old data in memory. See the READ command for more details about this option.

DIR CURRENT: Read and plot all simple filenames (with or without wildcards) from the current directory. This is the directory from which you started SAC.

DIR name: Read and plot all simple filenames (with or without wildcards) from the directory called name. This may be a relative or absolute directory name.

FREE: Read and plot the data in the filelist in free format (space delimited) mode.

FORMAT text: Read and plot the data in the filelist in fixed format mode. The format statement to use is given in text.

CONTENT text: Define the content of the data in the filelist. The meaning of the content text is described below.

PRINT {pname}: Print the resultant plot. If pname is specified, print to named printer, else use default printer.

filelist: A list of alphanumeric files. This list may contain simple filenames, full or relative pathnames, and wildcard characters. See the READ command for a complete description.

DEFAULT VALUES

```
PLOTALPHA FREE CONTENT Y. DIR CURRENT
```

DESCRIPTION

All commands in SAC work on the data that is currently in memory. This data in memory is analogous to the temporary or working files used by a text editor. The READ command reads binary SAC data files into memory. This command can be used to read a wide variety of alphanumeric data files into memory. These files can be in a fixed format or in free format. They may contain evenly or unevenly spaced data. They may contain more than one set of data. There may be only one label and the label is not retained in memory with the data.

The simplest use of this command is free field input of a Y data set. This is also the default. Free field input of X-Y pairs can be done by simply changing the content option. By combining the fixed format and content options, this command can also be used to read very complicated formatted output from other programs directly into SAC. Multiple Y data sets can be read from the same file using this method. Only a single X data set is allowed.

The basic header variables needed for processing are computed. These are NPTS, B, E, DELTA, LEVEN, DEPMIN, DEPMAX, and DEPMIN. If there is only a single Y data set, the name of the data file in memory will be the same as that of the alphanumeric disk file. If there are multiple Y data sets in the file, a two digit sequence number is appended to the file name. Each line of the alphanumeric data file is read in either free format or using the format statement provided. Each line can be up to 160 characters long. In the case of a free format file, the number of data entries in each line is also determined. The content field is then used to determine what to do with each of these data entries. Each specific character in the context field represents a different kind of data element and the order of these characters mimics the order of the data in each line of the file. The meanings of the allowed characters in the content field are given below:

L: Next entry is the label to plot with each data point (only one per data set).
Y: Next entry belongs to Y (dependent variable) data set.
X: Next entry belongs to X (independent variable) data set.
N: Next entry belongs to next Y data set.
P: Next pair of entries belong to X and Y data sets.
R: Next pair of entries belong to Y and X data sets.
I: Ignore (skip) this data element.

An optional repetition count may follow any of the above characters. This repetition count is a one or two digit integer and has the same meaning as repeating the content character that number of times. A period (".") is an infinite repetition count and means use the last characters meaning to decode the remaining data elements in the line. The period can only appear at the end of a content field.

EXAMPLES

To read and plot X-Y pairs in free format where the first field is the label:

```
u:  PLOTALPHA CONTENT LP FILEA
```

You can't break an X-Y pair between lines in the file.

ERROR MESSAGES

- 1301: No data files read in.
  - haven't given a list of files to read.
  - none of the files in the list could be read.
- 1020: Invalid inline function name:

  - Expected inline function. Preceed parenthesis with an atsign.

- 1320: Available memory too small to read file
- 1314: Data file list can't begin with a number.
- 1315: Maximum number of files in data file list is

WARNING MESSAGES

- 0101: opening file
- 0108: File does not exist:

HEADER CHANGES

B, E, DELTA, LEVEN, DEPMIN, DEPMAX, DEPMEN.

SEE COMMANDS

READ, WRITE, READALPHA

LATEST REVISION

July 22, 1992 (Version 10.6f)

PLOTC

SUMMARY

Annotates SAC plots and creates figures using cursor.

SYNTAX

```
PLOTC {REPLAY|CREATE} {FILE|MACRO filename},
{BORDER {ON|OFF}}
```

INPUT

REPLAY: Replay or replot an existing file or macro. The difference between a file and a macro is described below.
CREATE: Create a new file or macro.
FILE {filename}: Replay or create a file. The previous file is used if filename is omitted.
MACRO {filename}: Replay or create a macro.
BORDER {ON}: Turn border around plot on.
BORDER OFF: Turn border around plot off.

DEFAULT VALUES

```
PLOTC CREATE FILE OUT BORDER ON
```

DESCRIPTION

This command lets you annotate SAC plots and create figures for meetings and reports. A device with cursor capability is required. You "build" a figure by placing objects and text on the terminal screen. The cursor position determines where an object will be drawn and the character typed determines what object is to be drawn. Objects include circles, rectangles, n-sided polygons, lines, arrows, and arcs. Several ways of placing text are included.

This command creates two different type of output files, simple files and macro files. Both are alphanumeric files that can be changed using an editor. They contain the history of the cursor responses and locations from a single execution of the PLOTC command. A macro file, once created, can be used in more than one plot or figure. It can be scaled in size and can also be rotated. A simple PLOTC filename is the name you request with a ".PCF" appended to it. A macro file has a ".PCM" appended to its' name. This provides a check for SAC when you request a particular file and also lets you distinguish these files in your directories.

When you create a new file or macro, SAC draws a rectangle on the screen showing you the allowable area for the figure. It then turns the cursor on in the middle of this area. You move the cursor to the desired location and type a character representing the object you want drawn or the action you want to occur.

There are two types of cursor options, action and parameter-setting. The action options do something (draw a polygon, place text, etc.) How they do that action is based in part upon the current values of the parameter-setting options (how many sides on the polygon, what size text to draw, etc.) This distinction is similiar to the idea of action and parameter-setting commands in SAC itself. The tables on the following pages list the action and parameter-setting options.

When you replay a file or macro, the figure is redrawn on the terminal screen and then the cursor is turned on. You may then add to the file or macro as if you were creating it for the first time. When you

have created a figure that you want to send to a different graphics device, use the BEGINDEVICES command to temporarily turn off the terminal and turn on the other device. Then simply replay the file.

To annotate a SAC plot, execute the VSPACE command to set up the correct aspect ratio (see below), execute the BEGINFRAME command to turn off automatic framing, execute the desired SAC plot command, execute the PLOTC command (in create or replay mode), and then execute the ENDFRAME command to resume automatic framing.

EXAMPLES

An example of the use of PLOTC to add annotation to a standard SAC plot is the figure in the BANDPASS command description of this manual. The commands used to create that figure are given below with comments given in parentheses:

```
u:  FG IMPULSE NPTS 1024       \ \ (generate filter response)
u:  LOWPASS C2 NPOLES 7 CORNER 0.2 TRANBW 0.25 A 10
u:  FFT
u:  AXES ONLY LEFT BOTTOM      \ \ (set up desired plot options)
u:  TICKS ONLY LEFT BOTTOM
u:  BORDER OFF
u:  FILEID OFF
u:  QDP OFF
u:  VSPACE 0.75               \ \ (see discussion below)
u:  BEGINFRAME                \ \ (turn off automatic framing)
u:  PLOTSP AM LINLIN          \ \ (plot filter response)
u:  PLOTC CREATE FILE BANDPASS      \ \ (create annotation)
u:  ENDFRAME                  \ \ (turn automatic framing back on)
```

PLOTSP was used to produce the curve of the filter response and the two axes. PLOTC was used interactively to produce the annotation (i.e., the lines, arrows, and labels.) The viewspace command constrains the plot be the largest enclosed area of the graphics screen that has an (y:x) aspect ratio of 3:4. This is necessary so that when the output is later sent to the SGF device which also has a 3:4 aspect ratio, everything will be plotted correctly. At this point you would have a file called "BANDPASS.PCF" containing the annotations for this plot. To write this annotated plot to the SAC graphics file:

```
u:  BEGINDEVICES SGF    \ \ (select the SGF device)
u:  BEGINFRAME          \ \ (turn off automatic framing)
u:  PLOTSP              \ \ (plot filter response again)
u:  PLOTC REPLAY        \ \ (replay the annotation)
u:  ENDFRAME            \ \ (turn automatic framing back on)
```

A SAC graphics file will be created containing the annotated plot. Two examples (one somewhat frivolous) of the use of PLOTC to create figures and viewgraphs are given on the following pages. The replay files are also shown. (It is an exercise left to the reader to determine which of the examples is frivolous.)

1. The circle and sector opcodes only produce correct output when you have set the viewspace to a square one (VSPACE 1.0). Otherwise, they produce an ellipse with the ratio of the minor to major axis equal to the aspect ratio of the viewspace.
2. All all of the opcodes except text are scaled to fit in the graphics window.

The text sizes aren't currently scaled. This creates a problem when you create a figure and want to enclose some text in a rectangle or a circle. In this case the graphics window must be the same size as the output page in order to avoid misalignment.

This can be achieved by using the WINDOW command to set the horizontal (x) size of the window to be 0.75 and the vertical (y) size to be 0.69. For example: WINDOW 1 X 0.05 0.80 Y 0.05 0.74 This command must be executed before the window is created (i.e. before the BEGINWINDOW or BEGINDEVICES command.)

3. The text feature of this command works only in SunView graphics windows.

SEE COMMANDS

VSPACE, BEGINDEVICES, BEGINFRAME, ENDFRAME

LATEST REVISION

March 20, 1992 (Version 10.6e)

TABLE: Action Options:

```
char  meaning
A     Draw an arrow from ORIGIN to CURSOR.
B     Draw border tick marks around plot region.
C     Draw a circle centered at ORIGIN through CURSOR.
D     Delete last action option from replay file.
G     Set ORIGIN and make it global.
L     Draw a line from ORIGIN to CURSOR.
M     Invoke a macro at CURSOR.
Enter name of macro, scale factor, and rotation angle.
Previous values are used if omitted.
Defaults are OUT, 1., and 0.
O     Set ORIGIN at CURSOR.
N     Draw an n-sided polygon centered at ORIGIN
with one vertex at CURSOR.
Q     Quit PLOTC_.
R     Draw a rectangle with opposing corners
at ORIGIN and CURSOR.
S     Draw a sector of a circle centered at ORIGIN through
CURSOR  Move CURSOR to define the sector angle.
Type an S to get the sector whose angle is less than
180 degrees or C to get its' complement.
T     Place a single line of text at cursor.
Text is ended by a carriage-return.
U     Place multiple lines of text at cursor.
Text is ended by a blank line.
```

Notes:

- CURSOR is the current cursor location
- ORIGIN is normally the last cursor location
- The G option forces ORIGIN to remain fixed
- The O option allows ORIGIN to move again
- The Q option is not automatically copied to the file but may be added to it with a text editor.

If SAC does not see a Q in the file during replay mode, it goes back into cursor mode after displaying the contents of the file. This lets you append more options to the end of a file. If SAC does see a Q in the file, it displays the contents and ends PLOTC.

- A line beginning with an asterisk is treated as a comment line.

PLOTCTABLE

TABLE

Parameter-Setting Options These options consist of sets of two to four characters enclosed by square brackets. More than one set can be enclosed within the same set of brackets. For example, "[ALAFF7]" changes to large filled arrow heads and to text font 7. You MUST type only one character at a time and wait for the cursor to reappear before typing the next character.

chars	**meaning**
AT	Set arrow head size to TINY.
AS	Set arrow head size to SMALL. [Default]
AM	Set arrow head size to MEDIUM.
AL	Set arrow head size to LARGE.
AF	Set arrow head type to FILLED.
AU	Set arrow head type to UNFILLED. [Default]
AV	Set arrow shaft type to VISIBLE. [Default]
AI	Set arrow head shaft type to INVISIBLE.
BHn	Set number of horizontal border tick marks to n, n=0,99. [Default=9]
BVn	Set number of vertical border tick marks to n, n=0,99. [Default=9]
CN	Set color to NORMAL. [Default]
CR	Set color to RED.
CG	Set color to GREEN.
CY	Set color to YELLOW.
CB	Set color to BLUE.
CM	Set color to MAGENTA.
CC	Set color to CYAN.
CW	Set color to WHITE.
Fn	Set text font number to n, n=1,8. [Default=1]
HL	Set horizontal text justification to LEFT. [Default]
HC	Set horizontal text justification to CENTER.
HR	Set horizontal text justification to RIGHT.
Ln	Set linestyle to n, n=1,4. [Default=1]
Nn	Set number of polygon sides to n, n=2,9. [Default=3]
QN	Set text quality to HARDWARE.
QS	Set text quality to SOFTWARE. [Default]
ST	Set text size to TINY.
SS	Set text size to SMALL. [Default]
SM	Set text size to MEDIUM.
SL	Set text size to LARGE.
VB	Set vertical text justification to BOTTOM.
VC	Set vertical text justification to CENTER. [Default]
VT	Set vertical text justification to TOP.

PLOTDY

SUMMARY

Creates a plot with error bars.

SYNTAX

```
PLOTDY {ASPECT ON|OFF} {PRINT pname} name|number  name|number  { name|number
}
```

INPUT

ASPECT: ON maintains a 3/4 aspect ratio. OFF allows the aspect ratio to vary with the dimensions of the window.

PRINT {pname}: Print the resultant plot. If a printer name is given, print to that printer, else use default printer.

name: The name of a data file in the data file list.

number: The number of a data file in the data file list.

DESCRIPTION

This command allows you to plot a data set with error bars. The first data file you select (either by name or number) is plotted along the y axis. The second data file is the dy value. If a third data file is selected it is the positive dy value.

Assume you have an evenly spaced ascii file that contains two columns of numbers. The first is the y-value. The second column is the dy-value. You wish to read these into SAC and plot the data with error bars.:

```
u: READALPHA CONTENT YY MYFILE
u: PLOTDY 1 2
```

ERROR MESSAGES

- 1301: No data files read in.

LATEST REVISION

July 22, 1992 (Version 10.6f)

PLOTPK

SUMMARY

Produces a plot for the picking of arrival times.

SYNTAX

```
PLOTPK options
```

where options are one or more of the following:

```
{PERPLOT {ON|OFF|n}},
{BELL {ON|OFF}},
{ABSOLUTE|RELATIVE},
{REFERENCE {ON|OFF|v}},
{MARKALL {ON|OFF}},
{SAVELOCS {ON|OFF}}
```

INPUT

PERPLOT {ON}: Plot n files per frame. Use old value for n.
PERPLOT OFF: Plot all files on one frame.
PERPLOT n: Plot n files per frame.
BELL {ON}: Ring bell on each keystroke in active window.
BELL OFF: keystrokes are silent.
ABSOLUTE: Display times in absolute (GMT) format.
RELATIVE: Display times relative to each files's zero time.
REFERENCE {ON}: Turn reference line display on.
REFERENCE OFF: Turn reference line display off.
REFERENCE v: Turn reference line display on and change reference value to v.
MARKALL {ON}: Store header picks in all of the files displayed on a particular plot.
MARKALL OFF: Store header pick only in the file marked by the horizontal cursor.
SAVELOCS {ON}: Save pick locations (from L cursor command) in the blackboard.
SAVELOCS OFF: Do not save pick locations in the blackboard.

ALTERNATE FORMS

GMT may be used instead of ABSOLUTE and ZERO may be used instead of RELATIVE.

DEFAULT VALUES

```
PLOTPK PERPLOT OFF ABSOLUTE REFERENCE OFF MARKALL OFF SAVELOCS OFF
```

DESCRIPTION

The format of the PLOTPK plot is similiar to the PLOT1 plot. This plot requires a terminal with a cursor. When the cursor comes on and the bell rings, you enter single character responses to perform the various functions. Some but not all responses produce graphic output on the screen. Error and output messages are printed at the top of the screen. Picks that are currently in the header are automatically displayed on the screen. Output from the various cursor responses can be directed to the SAC header, to a alphanumeric pick file if open (see OAPF), to a HYPO pick file if open, and to the terminal. If you use the SAVELOCS option to save the cursor locations from the L cursor option in the blackboard, the following blackboard variables will be defined:

NLOCS: The number of locations picked during the execution of this command. This is initialized to 0 each time PLOTPK is invoked and incremented by 1 each time a cursor location pick is made.

XLOCn: The x value for the nth cursor location pick. This will be the GMT time of the pick if the reference time fields in the header are defined. Otherwise, this will be an offset time.

YLOCn: The y value for the nth cursor location pick.

The command 'help plotpktable' lists valid cursor options for plotpk

HEADER CHANGES

Depending upon user responses any of A, KA, F, KF, Tn, KTn.

ERROR MESSAGES

- 1301: No data files read in.
- 1202: Maximum number of vars sections exceeded:

WARNING MESSAGES

- 1502: Bad cursor position. Please retry.
  - cursor is positioned outside of the plot window.
  - occasionally the Tektronix sends the wrong position to SAC. Move the cursor a little and retry.
- 1503: Invalid character. Please retry.
  - A character was input that SAC didn't recognize as a legal response.
- 1905: Need an integer. Retry.
  - Didn't input an integer following the T response.
- 1906: Need an integer in the range 0 to 4. Retry.
  - Didn't input a 0, 1, 2 or 3 after a Q response.
- 1907: HYPO line already written.
- 1908: HYPO pick file not open.
- 1909: Can't compute waveform.
  - Adjust cursor position and retry. Plot is always in ABSOLUTE mode.

SEE COMMANDS

PLOT1, OHPF, OAPF, APK, PLOTPKTABLE

LATEST REVISION

May 16, 2006 (version 100.1)

PLOTPKTABLE

PLOTPK Cursor Options

char	meaning
A	Define first arrival in header {1,7}.
B	Display previous plot if any.
C	Evaluate first arrival and end of event {1,4,7}.
D	Set phase direction to DOWN.
E	Set phase onset to EMERGENT.
F	Define end of event {1,2,3,7}.
G	Display picks to terminal in HYPO format {4}.
H	Write picks to HYPO pick file {3,4}.
I	Set phase onset to IMPULSIVE.
J	Set noise level {2,6,8}.
L	List cursor location {2,4}.
M	Compute maximum amplitude waveform {2,3,5}.
N	Display next plot if any.
O	Display previous plot window. A maximum of five windows are saved.
P	Define P wave arrival time {1,2,3,7}.
Q	Terminate PLOTPK immediately.
S	Define S wave arrival time {1,2,3,7}.
T	Define user time Tn in header {1,2,7}. Next response an integer between 0 and 9.
U	Set phase direction to UP.
V	Define a Wood-Anderson waveform {2,5}.
W	Define a general waveform {2,5}.
X	Display current plot with new x plot window. Plot begins at current cursor position. This response must be followed by moving the cursor and hitting any key. The new plot will end at this second cursor position. If the second key is an S then these new x limits are saved and used in subsequent plots.
Z	Set zero (reference) level {2,6,8}.
	DELETE ALL CURRENT PICK FIELD DEFINITIONS. Useful when a file contains more than one event.
'+'	Set phase direction to PLUS.
'-'	Set phase direction to MINUS.
(space)	Set the phase direction to NEUTRAL.
n	Set phase quality to n, n=0, 1, 2, 3, or 4.

Notes

- {1} Written to SAC header.
- {2} Written to alphanumeric pick file if open.
- {3} Written to HYPO pick file if open.
- {4} Written to terminal.

- **{5} Place horizontal cursor at zero (reference) level and vertical** cursor beyond first extrema. Terminal echo is a box encompassing waveform.
- {6} Place horizontal cursor at desired level.
- {7} Terminal echo is a labeled vertical line.
- {8} Terminal echo is a labeled horizontal line.

PLOTPM

SUMMARY

Generates a "particle-motion" plot of pairs of data files.

SYNTAX

```
PLOTPM {PRINT {pname} }
```

INPUT

PRINT {pname}: Print the resultant plot. If a printer name is specified, print to that printer, else use default printer.

DESCRIPTION

In a particle-motion plot one evenly spaced file is plotted against another. For each value of the independent variable, normally time, the value of the dependent variable of the first file is plotted along the y axis and the value of the dependent variable of the second file is plotted along the x axis. For a pair of seismograms this type of plot shows the motion of a "particle" in the plane of the two seismograms as a function of time. A square plot is generated, with the limits along each axis being the minimum and maximum values of the dependent variable. Annotated axes are generated along the bottom and left. Axes labels and title can be set by the XAXIS, YAXIS, and TITLE commands. If no x and y axis labels are set, then the name and azimuth of the station are used as axes labels. The XLIM command can be used to control how much of each file to plot.

EXAMPLES

To create a particle-motion plot of two seismograms, XYZ.T and XYZ.R and set up your own axes labels and title:

```
u:  READ XYZ.T XYZ.R
u:  XLABEL 'Radial component'
u:  YLABEL 'Transverse component'
u:  TITLE 'Particle-motion plot for station XYZ'
u:  PLOTPM
```

If you wanted to plot only a small part of each file around the first arrival time, you could use the XLIM command:

```
u:  XLIM A -0.2 2.0
u:  PLOTPM
```

You could also use PLOTPK, possibly in a different graphics window as in this example, to mark which portion of the files you wanted to see in the particle motion plot:

```
u:  BEGINWINDOW 2
u:  PLOTPK
u:  ... mark the portion you want using X and S
u:  ... terminate PLOTPK with a Q
u:  BEGINWINDOW 1
u:  PLOTPM
```

LATEST REVISION

May 15, 1987 (Version 10.2)

PLOTSP

SUMMARY

Plots spectral data in several different formats.

SYNTAX

PLOTSP {type},{mode} where type is one of the following:

ASIS|RLIM|AMPH|RL|IM|AM|PH

where mode is one of the following:

LINLIN|LINLOG|LOGLIN|LOGLOG

INPUT

ASIS: Plot components in their present format.
RLIM: Plot real and imaginary components.
AMPH: Plot amplitude and phase components.
RL: Plot real component only.
IM: Plot imaginary component only.
AM: Plot amplitude component only.
PH: Plot phase component only.
LINLIN: Set x-y scaling mode to linear-linear.
LINLOG: Set x-y scaling mode to linear-logarithmic.
LOGLIN: Set x-y scaling mode to logarithmic-linear.
LOGLOG: Set x-y scaling mode to logarithmic-logarithmic.

DEFAULT VALUES

```
PLOTSP ASIS LOGLOG
```

DESCRIPTION

SAC data files may contain either time-series data or spectral data. Certain fields in the header distinguish between the two formats. Most plot commands (PLOT, PLOT1, etc.) only plot time-series data. This command lets you plot spectral data.

You may plot one or both spectral components using this command. One frame is generated for each spectral component plotted. Other plot formats will be added as needed. You can also select the scaling mode to be used. This scaling mode applies only to PLOTSP.

EXAMPLES

To get a logarithmic-linear plot of the spectral amplitude of a data file:

```
u:  READ FILE1
u:  FFT
u:  PLOTSP AM LOGLIN
```

ERROR MESSAGES

- 1301: No data files read in.
- 1305: Illegal operation on time series file

LATEST REVISION

May 15, 1987 (Version 10.2)

PLOTXY

SUMMARY

Plots one or more data files versus another data file.

SYNTAX

```
PLOTXY name|number  name|number  { name|number ...  }
```

INPUT

name: The name of a data file in the data file list.
number: The number of a data file in the data file list.

DESCRIPTION

This command lets you plot one or more data files versus another data file. The first data file you select (either by name or number) becomes the independent variable and is plotted along the x axis. The remainder of the data files you select become the dependent variables and are plotted along the y axis. All of the graphics environment commands such as TITLE, LINE, and SYMBOL can be used to control attributes about the plot. This command can be used to easily plot multi-columned data that has been read in with the READALPHA command. In this case it can be viewed as a spreadsheet like plotting command. An example is given below.

EXAMPLES

Assume you have an ascii file that contains four columns of numbers. You wish to read these into SAC and plot various columns versus each other. The following commands would read this file in and store it as four separate data files inside SAC, turn linestyle incrementing on and then plot the first, third, and fourth columns versus the second column:

```
u: READALPHA CONTENT YNNN MYFILE
u: LINE INCREMENT ON
u: PLOTXY 2 1 3 4
```

LATEST REVISION

April 21, 1989 (Version 10.4c)

PRINT

SUMMARY

Prints the most recent SGF file. This command requires that at least one SGF file has been produced.

SYNTAX

```
PRINT {printer}
```

INPUT

printer: sends output to the named printer, if no printer name is supplied, it will print to the system's default printer.

Note: PRINT will not work if the SGF device has remained off since boot. Use BEGINDEVICES to turn on the SGF device. Use the SGF command to set prefered behavior for the SGF device. The SGF command has an overwrite option which prevents disk file buildup by clobbering previous SGF files with the new ones.

DEFAULT VALUES

```
PRINT
```

ERROR MESSAGES

- 2405: Cannot PRINT: no SGF files produced.

SEE COMMANDS

SGF, BEGINDEVICES

LATEST REVISION

April 22, 1999 (Version 0.58)

PRINTHELP

SUMMARY

Prints hardcopies of information about SAC commands and features.

SYNTAX

```
PRINTHELP {item ...}
```

INPUT

item: The (full or abbreviated) name of a command, module, subprocess, feature, etc.

DEFAULT VALUES

```
If no item is requested, an introductory help package is printed.
```

DESCRIPTION

Each requested item in the help package is printed in the order they are requested. A short message is printed if no information is available for an item. The help package for each command consists of the entry in the SAC Command Reference Manual. The help package for non-commands may be paragraphs from the SAC Users Manual or other information.

EXAMPLES

To get the introductory help package type:

```
u:  PRINTHELP
```

Now lets say you want information on several commands:

```
u:  PRINTHELP READ CUT BEGINDEVICE PLOT
```

ERROR MESSAGES

- 1103: No help package is available.
  - SAC can't find the help package. Check your SACAUX environment.

SEE COMMANDS

HELP

LATEST REVISION

November 13, 1998 (Version 0.58)

PRODUCTION

SUMMARY

Controls the production mode option.

SYNTAX

```
PRODUCTION ON|OFF
```

INPUT

ON: Turn production mode option on.
OFF: Turn production mode option off.

DEFAULT VALUES

```
PROD OFF
```

DESCRIPTION

When this option is on, fatal errors terminate SAC immediately. When this option if off, control is returned to the terminal after fatal errors.

LATEST REVISION

January 8, 1983 (Version 8.0)

QDP

SUMMARY

Controls the "quick and dirty plot" option.

SYNTAX

```
QDP {ON|OFF|n},{TERM ON|OFF|n},{SGF ON|OFF|n}
```

INPUT

ON: Turn QDP option on for both the terminal and SAC Graphics File (SGF) devices.

OFF: Turn QDP option off for both devices.

n: Turn QDP option on for both devices and change the approximate number of data points to plot to n.

TERM ON: Turn quick and dirty plotting on for the terminal.

TERM OFF: Turn quick and dirty plotting off for the terminal.

TERM n: Turn QDP option on for the terminal and change the approximate number of data points to plot to n.

SGF ON: Turn quick and dirty plotting on for the SGF.

SGF OFF: Turn quick and dirty plotting off for the SGF.

SGF n: Turn QDP option on for the SGF and change the approximate number of data points to plot to n.

DEFAULT VALUES

```
QDP TERM 5000 SGF 5000
```

DESCRIPTION

Plotting large files (greater than say 1000 points) can take a long time. The "quick and dirty plot" option speeds up plotting by NOT plotting each data point. When this option is on, SAC will compute a section size by dividing the number of data points in the file by the number of data points you want displayed. The larger the file, the more data points in each section.

SAC then computes and displays only the minimum and the maximum data point in each section. SAC displays a "desampling factor" (half the section size) in a small box in the corner of the plot when this option is on. Displayed data points may be somewhat closer or further apart than this number indicates since the extremum in each region are being plotted.

There is a separate QDP option for the terminal and the SAC Graphics File device. The terminal QDP factor also applies to the XWINDOWS and SUNWINDOWS graphics devices. By default the QDP factor is considerably smaller for the terminal than for the SGF. This allows for very fast plots to the terminal and a more representative plot to the SGF. If both devices are on at the same time, the terminal QDP option applies. You may turn either of these options off or change the number of displayed points.

EXAMPLES

Assume FILE1 has 2100 data points and FILE2 has 4700 data points. If you typed:

```
u:  READ FILE1 FILE2
u:  BEGINDEVICES TERMINAL
u:  PLOT
```

both plots would contain approximately 200 data points. The plot of FILE1 would contain approximately every tenth data point and the plot of FILE2 every twenty-third data point. The section size is rounded down to ensure that you will see at least the number of requested data points. If you now plotted those same files to the SGF:

```
u:  BEGINDEVICES SGF
u:  PLOT
```

both plots would contain approximately 1000 data points. If both devices were on, the plots would contain approximately 200 data points, the factor for the terminal.

LATEST REVISION

February 20, 1985 (Version 9.13)

QUANTIZE

SUMMARY

Converts continuous data into its quantized equivalent.

SYNTAX

```
QUANTIZE [GAINS n ...],[LEVEL v],[MANTISSA n]
```

INPUT

GAINS n ...: Set list of allowed gains. They must be monotonically decreasing. The maximum number of allowed gains is 8.

LEVEL v: Set the quantization level of the lowest gain. This is the value of the least significant bit in volts.

MANTISSA n: Set the number of bits in the mantissa.

DEFAULT VALUES

```
QUANTIZE GAINS 128 32 8 1 LEVEL 0.00001 MANTISSA 14
```

DESCRIPTION

This command exercises a quantization algorithm equivalent to the "rounding" quantization described in Oppenheim and Schafer (1975, Fig. 9.1). The number of bits used in this algorithm are partitioned into the bits used to represent the characteristic (exponent), the sign bit, and the mantissa bits. The user can specify the number of bits used for the mantissa. The quantization level (value of least significant bit or LSB) can also be specified by the user. The default quantization level is 10 microvolts. The error of the signal represented by this quantized function is numerically equal to one-half of this quantization level. In the spectral domain, this error or quantization noise is:

```
ERROR = 1/12 * (DELTA * LEVEL^2)
```

where DELTA is the sampling interval. This quantization noise is measured in units of counts*counts/Hz, as a power spectral density. The rms-squared quantization noise is:

```
(1/6)*LEVEL^2.
```

However, this is an accurate approximation to the noise due to quantization only if the rms level of the signal is much larger than the rms quantization noise. In other words, if the signal is not resolved by several hundred counts, then there is a correlation between the quantization noise and the signal being quantized. The fraction of correlation is approximately equal to the ratio of the LEVEL to the rms of the signal being quantized (see Fig. 11.13, Oppenheim and Schaffer, 1975). The gains can be specified by the user to simulate the gain steps in an automatic gain-ranging system. The default gains are those of the Regional Seismic Test Network (RSTN.) Oppenheim, Alan V., and Ronald W. Schafer; Digital Signal Processing; Prentice-Hall; 1975; 585pp.

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

LATEST REVISION

May 15, 1987 (Version 10.2)

QUIT

SUMMARY

Terminates SAC.

SYNTAX

```
QUIT
```

ALTERNATE FORMS

END, EXIT, and DONE are also allowed.

DESCRIPTION

This command terminates SAC gracefully. (There are a number of ways to terminate it ungracefully!) Before terminating, SAC ends all active graphics devices, closes all output files, and destroys any temporary files it has created.

LATEST REVISION

October 11, 1984 (Version 9.1)

QUITSUB

SUMMARY

Terminates the currently active subprocess.

SYNTAX

```
QUITSUB
```

DESCRIPTION

This command terminates the currently active subprocess, returning to the main SAC command environment. Files in memory are retained. There are currently two subprocesses available in SAC:

SES Spectral Estimation Subproess

SSS Signal Stacking Subprocess

SEE COMMANDS

SES, SSS

LATEST REVISION

October 11, 1984 (Version 9.1)

READ

SUMMARY

Reads data from SAC data files on disk into memory.

SYNTAX

```
READ [options] [filelist]
```

where options is one or more of the following:

```
MORE
TRUST ON|OFF
COMMIT|ROLLBACK|RECALLTRACE
DIR CURRENT|name
XDR
ALPHA
SEGY
SCALE [ON|OFF]
```

ALL options MUST preceed any element in the filelist.

INPUT

MORE: Place the new data files in memory AFTER the old ones. If this option is omitted, the new data files REPLACE the old ones.

Note: if the MORE option is not specified, the COMMIT, ROLLBACK, and RECALLTRACE options have no effect.

TRUST ON|OFF: This option is used to resolve an ambiguity in converting files from SAC to CSS format. When converting data, matching event IDs could mean the files have identical event information, or they could be an artifact of the merging of these two very different formats. When TRUST is ON, SAC is more likely to accept matching event IDs as identical event information than when TRUST is OFF, depending on the history of READ commands associated with the current data files in memory.

COMMIT: If the MORE option is specified, the COMMIT option commits headers and waveforms in SAC memory -- removing any previous versions of headers or waveforms from RAM -- prior to reading more files. COMMIT is the default.

ROLLBACK: If the MORE option is specified, the ROLLBACK option reverts to the last committed version of the header and waveform before reading more files.

RECALLTRACE: If the MORE option is specified, the RECALLTRACE option:

- reverts to the last committed version of the waveform,
- reverts to the last committed version of those header variables closely linked to the waveform,
- commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of header variables which are committed, and which are rolled back.)

DIR CURRENT: Read all simple filenames (with or without wildcards) from the current directory. This is the directory from which you started SAC.

DIR name: Read all simple filenames (with or without wildcards) from the directory called name. This may be a relative or absolute directory name.

XDR: The input files are in XDR format. This format is used for moving binary data files to/from a different architecture, such as a pc running LINUX.

ALPHA: The input files are SAC formatted alphanumeric (ascii) files. the ALPHA option is incompatible with the XDR option.

SEGY: Read file formatted according to the IRIS/PASSCAL form of the SEGY format. This format allows one waveform per file.

SCALE: Used only in conjunction with the SEGY option, the SCALE option is OFF by default. When SCALE is OFF, SAC reads the counts from the SEGY files. When SCALE is ON, SAC multiplies counts by a SCALE factor given in the file. This scale option changes with the SCALE options of READCSS, READGSE, and READSUDS. If SCALE is OFF, the SCALE value from the file will be stored in SAC's SCALE header variable. If SCALE is on, SAC's SCALE header field is set to 1.0. SCALE is a crude method of accounting for the instrument response. The prefered method is with the TRANSFER command. It is recommended to leave SCALE off for the READ and use the TRANSFER command. SCALE should really only be used if the response information necessary for the TRANSFER command is not available.

filelist: file | wild .

file: A legal filename. This may be a simple filename or a pathname. The pathname can be a relative or absolute one. See the DESCRIPTION and EXAMPLES sections below for more details.

wild: A wildcard laden token that expands to a list of filenames. See the DESCRIPTION and EXAMPLES sections below and the WILD command for more details.

DEFAULT VALUES

```
READ COMMIT DIR CURRENT
```

DESCRIPTION

All commands in SAC work on the data that is currently in memory. This data in memory is analogous to the temporary or working files used by a text editor. The READ command transfers data from one or more disk files into memory. The default is to read all of the data from each disk file.

The CUT command can be used to specify that only a portion of each disk file be read. SAC files produced in or after the year 2000 are presumed to have a four digit value for the year. Files with two digit year values will be assumed to be in the twentieth century, and will be incremented by 1900. Normally all data in memory prior to the execution of another READ command is lost. The new data replaces the old data.

If the keyword MORE is the second symbol in the command, the new data is placed in memory after the old data. The data file list becomes the concatenation of the old file list and the new file list. There are three cases where the MORE option may be useful:

1. The filelist is too long to be typed on one line.
2. A name was misspelled in a long filelist.
3. **A file is read, some analysis performed, and a comparison with** the original is desired.

Examples of each of these cases are given below. The filenames may be simple filenames in the current directory or they may be absolute or relative pathnames pointing to other directories on your system. Examples of absolute pathnames are:

```
UNIX: /dir/subdir/file
VMS: disk:[dir.subdir]file
PRIMOS: <disk>dir>subdir>file
```

Examples of relative pathnames are:

```
UNIX: subdir/file
PRIMOS: *>subdir>file
```

In the above examples "disk" is the name of a physical disk partition, "dir" is the name of a top level directory, "subdir" is a subdirectory of that partition, and "file" is a file in that subdirectory. In general there is no limit on the nesting of subdirectories. Filenames may also contain wild-card characters. You can use them match a single character, to match zero or more characters, and to form groupings of characters. Some examples are given below. See the WILD command for more examples and a complete explanation of all the wildcarding options.

*** Important *** SAC has two data buffers; this is what allows SAC to provide the COMMIT, ROLLBACK and RECALLTRACE commands. One data buffer stores the header information in SAC format, and the second stores headers in CSS 3.0 format. This CSS 3.0 data buffer allows seamless consistency with CSS 3.0 in READCSS and WRITECSS; it also allows direct access to the CSS 3.0 formatted Oracle database.

In CSS (a relational format), it is important to maintain consistency with the event IDs (evid, or nevid in SAC). In SAC format (a very flat format), such consistency is not as important, and in some cases, it is lost. Anytime data is loaded into SAC, it is stored in both buffers. When transfering data from SAC to CSS data buffers, there is a potential ambiguity in handling event information. If matching evids are found, it could be that the two files have identical event information, or it could be that the match is an artifact of the merge of these two different data formats within SAC.

Two peices of information are involved in resolving this ambiguity, one is the history of data loaded into SAC memory, and the other is the confidence the user sets with the TRUST ON|OFF option on the command line of most Read commands and ADDSTACK. It is expected that the user will have some idea if the data files are consistent, if they share event information, etc. The history of data loaded into SAC memory begins when data is loaded into memory without the MORE option, and ends the next time data is loaded into memory without the MORE option. Any time in between that data is loaded into memory with the MORE option, it becomes part of the existing history.

All commands which load data into memory are now monitored to maintain a level of confidence in the event information when moved from the SAC data buffer to the CSS data buffer. The READDB command was the most reliable way to load data into SAC but ORACLE isis no longer supported. Still, for this reason, the levels of confidence (in ascending order) are LOW, HIGH, and RDB. TRUST is an option set on the commandline of most commands that load SAC data. TRUST can be ON or OFF. Each time a command loads data into SAC, it responds to both the confidence level and the TRUST to determine how to treat matching event IDs.

When requisite levels of TRUST and confidence are present, matching event IDs are treated as an indicator that the two files share identical event information. This being the case, the event IDs are left unchanged. When requisite levels of TRUST and confidence are not met, matching event IDs are are treated as artifacts of the merging of two different data formats: SAC and CSS.

READDB, being the only truely reliable way to load data, will always treat the data as reliable as long as READDB is the only method used to load data into SAC. In this case, the TRUST will not be used, and the confidence lever will always be set to RDB. If any other data loading method is used, then the confidence will be reduced to HIGH, or LOW, and RDB will respond similarly to the next set of commands. The following commands are considered HIGH confidence: READ, READCSS, READHDR, and ADDSTACK. These commands will consider both the confidence level and the TRUST in determining how to handle event ID matches. If the confidence is HIGH (or RDB) and the TRUST is ON, then confidence is set to HIGH, and the event IDs are treated as reliable. However if the TRUST is OFF or if the confidence level is LOW, then the confidence is set to LOW and the matching event IDs are treated as artifacts, and new IDs are generated for the incoming data file.

The following commands are considered LOW confidence because event ID information is not available: READTABLE, READGSE, READSUDS, FUNCGEN, DATAGEN, READSP, READSDD, and READ

with the SEGY option and READCSS when the input files are in the CSS 2.8 data format. These commands will always generate event IDs and set the confidence level to LOW.

Commands written by the user and added to SAC via the external command interface (the LOAD command) are a special case, since the user writes the code. In these cases, the confidence is based entirely on the TRUST that the user sets. Hence, it is incumbent upon the user to set the TRUST either on the commandline or within the code. For more information use HELP EXTERNAL. Within a given history, if data is loaded into SAC by any means other than READDB, the data is probably not consistent with the database, and should probably not be loaded back into the database, unless the user takes responsibility to insure consistency among events.

EXAMPLES

In the following examples is it assummed that the following SAC data files are in your current disk directory: F01, F02, F03, and G03. In these examples, the UNIX wildcard characters (e.g., "?" matches any single character and "*" matches zero or more characters) are used. See the WILD command for more information on how to use wildcards. To read the first three files:

```
u:  READ F01 F02 F03
```

The following command produces the same result using the wildcard operator:

```
u:  READ F*
```

This command also produces the same result by using the concatenation operator:

```
u:  READ F0[1,2,3]
```

To read the second, third, and fourth files:

```
u:  R F02 ?03
```

The following examples show the use of the MORE option:

```
u: R F03 G03
```

files F03 and G03 are in memory:

```
u: R F01 F02
```

files F01 and F02 are in memory:

```
u: R MORE F03 G03
```

files F01, F02, F03, and G03 are in memory

This example uses the MORE option when a filename was misspelled:

```
u: R F01 G02 F03
s: WARNING: File does not exist: G02
s: Will read the remainder of the data files.
```

files F01 and F03 are in memory:

```
u: R MORE F02
```

files F01, F03, and F02 are now in memory:

note the order of the files in this case.

If you wanted to apply a highpass filter to a data file and then graphically compare the results to the original:

```
u: READ F01
u: HIGHPASS CORNER 1.3 NPOLES 6
u: READ MORE F01
u: PLOT1
```

plot shows filtered and original data

Now assume you were in the directory "/me/data" when you started up SAC and that you wanted to work with the data files in the subdirectories "event1" and "event2":

```
u: READ DIR EVENT1 F01 F02
```

files in directory /me/data/event1 are read:

u: READ F03 G03

files in same directory are read:

```
u: READ DIR EVENT2 *
```

all files in /me/data/event2 are read:

u: READ DIR CURRENT F03 G03

files in directory /me/data are read.

Note: For examples of the differing behavior between the COMMIT, ROLLBACK, RECALLTRACE options, see the commands of the same names.

ERROR MESSAGES

- 1301: No data files read in.
  - haven't given a list of files to read.
  - none of the files in the list could be read.
- 1320: Available memory too small to read file
- 1314: Data file list can't begin with a number.
- 1315: Maximum number of files in data file list is
- 6002: No more data-sets available.

WARNING MESSAGES

- 0101: opening file
- 0108: File does not exist:
- 0114: reading file
  - Normally when SAC encounters one of these errors it skips that file and reads the remainder. These errors can be made to be fatal using the READERR command.

HEADER CHANGES

E, DEPMIN, DEPMAX, DEPMEN, B if cut option is on.

SEE COMMANDS

CUT, READERR, WILD, COMMIT, ROLLBACK, RECALLTRACE

LATEST REVISION

June. 18, 1999 (Version 0.58)

READBBF

SUMMARY

Reads a blackboard variable file into memory.

SYNTAX

```
READBBF {file}
```

INPUT

file: The name of a blackboard variable file. It may be a simple filename or a relative or absolute pathname.

DEFAULT VALUES

```
READBBF BBF
```

DESCRIPTION

This command lets you read in a blackboard variable file. This file must have been previously written to disk using the WRITEBBF command. This feature lets you save information from one execution of SAC to another. You can also add coding to your own programs to access the information in these blackboard variable files. This lets you transfer information between your own programs and SAC. See the SAC Subroutines Reference Manual for details.

SEE COMMANDS

WRITEBBF, SETBB, GETBB

LATEST REVISION

May 15, 1987 (Version 10.2)

READCSS

SUMMARY

Read data files in CSS external format from disk into memory.

NOTE: The READCSS command reads flat files which adhere to CSS 3.0 or 2.8 data formats. The following tables are supported for version 3.0:

wfdisc, wftag, origin, arrival, assoc, sitechan, site, affiliation, origerr, origin, event, sensor, instrument, gregion, stassoc, remark sacdata.

For version 2.8 only wfdisc, arrival, and origin are supported. Previous versions of READCSS required that the origin file have only one line which would be associated with the waveforms pointed to by the wfdisc file. The current version can extract the correct origin (or origins) using information from a wftag file or using an evid column in the wfdisc file (position 284 - 291). If no such information is available, READCSS will default to its previous behavior, and use the first row in the origin file. There is now no information lost when data is read using READCSS. Although existing SAC commands can only access a subset of the CSS data, everything read from CSS flatfiles is retained in memory and will be written to disk with the WRITECSS command.

READCSS now reads a non-standard table named sacdata (written by the WRITECSS command) which holds data from the SAC header that does not have a place in the standard schema. With the sacdata table, there is now no information loss when SAC data is written in CSS format and then re-read. For instance, you can now write frequency domain data to disk with WRITECSS and re-read it later with READCSS.

READCSS now has a binary option that allows it to read binary CSS files written by WRITECSS. In binary mode the css options have no effect. That is, the entire contents of the specified file(s) will be read.

READCSS supports the following binary datatypes: On bigendin machines (eg. Sun) t8, t4, f8, f4, s4, s3, s2, i4, i2, g2, e1, and ri (real-imag).

On littleendin machines (eg. DEC or PC) f8, f4, t8, t4, i4, i2, s4, s2, and g2

SYNTAX

```
READCSS {BINARY|ASCII} {MAXMEM v} {MORE} {TRUST ON|OFF} {VERBOSE ON|OFF}
{SHIFT ON|OFF} {SCALE ON|OFF} {COMMIT|ROLLBACK|RECALLTRACE} {DIR name}
wfdisclist {filelist} {css options}
```

The css options are one or more of the following:

```
STATION station
CHANNEL channel
BANDWIDTH band code
ORIENTATION orientation code
```

which causes this command to further select from files that are qualifying members of filelist based on the content of their corresponding records in the wfdisc file.

INPUT

ASCII: (Default) Reads normal ASCII flatfiles.

BINARY: Reads binary CSS files. See the WRITECSS command for more information on this format.

TRUST ON|OFF: This option is used to resolve an ambiguity in converting files from SAC to CSS format. When converting the data, matching event IDs could mean the files have identical event information, or they could be an artifact of the merging of these two very different formats. When TRUST is ON, SAC is more likely to accept matching event IDs as identical event information than when TRUST is OFF, depending on the history of READ commands associated with the current data files in memory.

MAXMEM: Specify the maximum fraction of physical memory to use when reading large data sets. When this limit is reached, no more waveforms will be read, although other tables may still be read. The default value for MAXMEM is 0.3.

MORE: See the READ command.

VERBOSE ON|OFF: If VERBOSE is ON, SAC displays extended information about the waveforms being read, and prints a summary of the CSS tables that were filled. SAC also displays a progress indicator for the conversion to SAC internal format.

SHIFT ON|OFF: If SHIFT is on, the origin time is set to zero, and other time related header variables are shifted back to be consistent with the origin time. Some of the distance related header variables are also affected. SHIFT ON is the default.

SCALE ON|OFF: The SCALE option is OFF by default. When SCALE is OFF, SAC reads the counts from the .w file. When SCALE is ON, SAC multiplies counts by a SCALE factor given as CALIB in the .wfdisc file. This scale option changes with the SCALE options of READ SEGY, READGSE, and READSUDS. If SCALE is OFF, the CALIB value from the file will be stored in SAC's SCALE header variable. If SCALE is on, SAC's SCALE header field is set to 1.0. SCALE is a crude method of accounting for the instrument response. The prefered method is with the TRANSFER command. It is recommended to leave SCALE off for the READ and use the TRANSFER command. SCALE should really only be used if the response information necessary for the TRANSFER command is not available.

COMMIT: If the MORE option is specified, the COMMIT option commits headers and waveforms in SAC memory -- removing any previous versions of headers or waveforms from RAM -- prior to reading more files. COMMIT is the default.

ROLLBACK: If the MORE option is specified, the ROLLBACK option reverts to the last committed version of the header and waveform before reading more files.

RECALLTRACE: If the MORE option is specified, the RECALLTRACE option:

- reverts to the last committed version of the waveform,
- reverts to the last committed version of those header variables closely linked to the waveform,
- commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

Note: if the MORE option is not specified, the COMMIT, ROLLBACK, and RECALLTRACE options have no effect.

DIR name: The directory to be searched for wfdisc(s).

wfdisclist: The name(s) of one or more wfdisc files.

filelist: A list of data file names contained in the previously specified wfdisc(s). These files will be searched for first in the directory specified with the DIR option, then using the path specified in the wfdisc. If no filelist is supplied, all the data files defined in the specified wfdisc(s) will be read into memory.

STATION station: station is a string of six or fewer characters. Selects those lines from the .wfdisc file whose KSTNM matches station. station may contain * and ? wildcards.

CHANNEL channel: channel is a string of eight or fewer characters. Selects those lines from the .wfdisc file whose channel specifier matches channel. channel may contain * and ? wildcards.

BANDWIDTH type: A 1-letter code. Usual values are E Extremely Short Period S Short Period H High Broad Band B Broad Band M Mid Period L Long Period V Very Long Period U Ultra Long Period R Extremely Long Period
Selects those files whose 'chan' field has a leading character which is s, m or l. The character * selects all.

ORIENTATION type: Usual values are: Z N E (Vertical North East) A B C (Triaxial along edges of cube standing on corner) 1 2 3 Orthogonal but non-standard orientation
Selects those files whose 'chan' field has a final character which matches code. The character * selects all.

MAGNITUDE

- MB
- MS
- ML
- DEF

Determines which value of magnitude to put into SAC's magnitude hearder variable. MB gets the bodywave magnitude, MS gets the surfacewave magnitude, ML gets the local magnitude, and def (the default) follows this algorithm: if Ms exists and is greater than or equal to 6.6, take it, else, if Mb exists take it, else, if Ms exists take it, else take Ml.

DEFAULT VALUES

```
READCSS * ASCII MAXMEM 0.3 VERBOSE OFF COMMIT STATION * BAND * CHAN * ORIENT
*
```

DESCRIPTION

See the READ command. Oct. 27, 1998 (Version 00.58)

All commands which load data into memory have are now monitored to maintain a level of confidence in the event information when moved from the SAC data buffer to the CSS data buffer. For READCSS, when the confidence is HIGH that all the data files are cosistent in the numbering of event IDs, matching event IDs are treated as having identical event information. When the confidence is LOW in READCSS, matching event IDs are understood as an artifact, and new event IDs are generated for the incoming file. For more details use HELP READ.

How READCSS reads picks from the .arrival file:

SAC has two data buffers. One holds the data in SAC format, and one holds it in CSS3.0 format. READCSS reads all the available arrivals into the CSS buffer. Only 10 picks will fit into the SAC formatted buffer. The command PICKPREFS controls the way the picks are transfered from the CSS buffer to the SAC buffer.

There is a preferences file which SAC uses to determine which phases and authors' picks to transfer between buffers. The default preferences file is $SACAUX/csspickprefs. This default can be overridden by either the PICKAUTHOR or PICKPHASE commands. These commands can select a user-defined preferences file, or they can interactively override the preferences file.

If PICKPREFS is OFF (the default), SAC will transfer the first 10 picks from the CSS data buffer to the SAC data buffer. If PICKPREFS is ON, SAC will transfer the picks according to the preferences file, or the PICKAUTHOR and PICKPHASE commands.

The following is an example of a preferences file:

john rachel michael

t0 P - t1 Pn rachel t2 Pg - t3 S - t4 Sn - t5 Sg - t6 Lg - t7 LR - t8 Rg - t9 pP -

Note: phase names are case sensitive; author names are not.

The first few lines are a prioritized list of author names (analysts who have made picks) that sac can use to select picks from the data. The remaining lines tell sac which css phase picks should be mapped into which sac header variables (T0 through T9). A hyphen (-) in the third column tells sac to use the prioritized author list. An optional author name can be specified in the third column which will overide the default author list for this pick.

For a given waveform, sac will choose from the available picks those which match the given phase and author name. If an author name is specified in the third column, sac will try to match that; if it fails to match that author name, it will leave the header variable undefined. If the third column has a hyphen, sac will try to match the first name in the list; if it fails it will try to match the second name and so on until it gets a match, or the author list is exhaused (in which case the header variable is left undefined). In the example file shown above, T0 will have a P phase with john, rachel, or micheal as the author, or it will be left blank; T1 will have a Pn phase and rachel as the author, or it will be left blank. For each pick header variable there is a corresponding string header variable (KT0 through KT9). These will be populated with the phase names of the corresponding picks.

The basic format of the preferences file is: Author names are delimited by newlines. There are no blank lines before the first author name, and no blank spaces at the begining of a line. There are no blank spaces in the middle of an author name. Author names must be unique. Author names may be up to 15 characters long. There may be any number of author names.

The names are listed in order of priority, the most important author name first. The last name in the author list is followed by an empty line to designate the end of the author list.

The header variable information occupies 10 lines in three columns. The first column simply lists the names of the header variables in numerical order. The second column lists specific phase names; phase names can be up to eight characters long. The third column can have a specific author name, or a hyphen. The columns are separated by tabs. There are no spaces anywhere in these 10 lines.

SEE COMMANDS

READ, PICKPREFS, PICKAUTHOR, PICKPHASE, COMMIT, ROLLBACK, RECALLTRACE

READDB

SUMMARY

Reads data from Oracle database into memory.

As of v101.5, the Oracle database has been removed from the distribution.

READERR

SUMMARY

Controls errors that occur during the READ command.

SYNTAX

```
READERR {BADFILE FATAL|WARNING|IGNORE},
{NOFILES FATAL|WARNING|IGNORE}
{MEMORY SAVE|DELETE}
```

INPUT

BADFILE: Errors that occur when the file could not be read or didn't exist.
NOFILES: None of the files in the read filelist could be read.
FATAL: Make error condition fatal. Send error message and stop processing the command.
WARNING: Send warning message but continue processing the command.
IGNORE: Ignore condition and continue processing the command.
MEMORY: Action on files in memory if no files could be read.
DELETE: This MEMORY option indicates that files previously in memory are to be deleted.
SAVE: This MEMORY option indicates that files previously in memory are to remain in memory.

DEFAULT VALUES

```
READERR BADFILE WARNING NOFILES FATAL MEMORY DELETE
```

DESCRIPTION

Several errors can occur when you try to read a data file into memory using the READ command. The file may not exist or it may exist but can't be read. When SAC encounters one of these bad files, it normally sends a warning message and then tries to read the rest of the files in the filelist. If you want SAC to stop reading in files whenever a bad file is encountered set the BADFILE condition to FATAL. If you don't even want to see the warning message, set the BADFILE condition to IGNORE. If none of the files in the filelist could be read, SAC normally sends an error message and stops processing. If you want SAC to send a warning message or ignore this problem completely, set the NOFILES condition accordingly. Also, any files previously in memory can be deleted (removed from) or remain in memory by using the MEMORY DELETE or MEMORY SAVE options. The CUTERR command can be used to control certain errors that occur due to bad cut parameters.

SEE COMMANDS

READ, CUTERR

LATEST REVISION

March 20, 1992 (Version 10.6e)

READGSE

SUMMARY

Read data files in GSE 2.0 format from disk into memory.

Note: GSE data enters SAC via SAC's CSS data buffers. To understand how arrivals are handled, use HELP READCSS and HELP PICKPREFS.

SYNTAX

```
READGSE {MAXMEM v} {MORE} {VERBOSE ON|OFF} {SHIFT ON|OFF}
{SCALE ON|OFF} {COMMIT|ROLLBACK|RECALLTRACE} {DIR name} filelist
```

INPUT

MAXMEM: Specify the maximum fraction of physical memory to use when reading large data sets. When this limit is reached, no more waveforms will be read, although other tables may still be read. The default value for MAXMEM is 0.3.

MORE: See the READ command.

VERBOSE ON|OFF: If VERBOSE is ON, SAC displays extended information about the waveforms being read, and prints a summary of the CSS tables that were filled. SAC also displays a progress indicator for the conversion to SAC internal format.

Note: the SHIFT option is moot at this point. For the time being, origin information is not read because it cannot be associated with a waveform. The release of GSE 2.1 format should allow us to make the association, then we will be able to read origin, and the SHIFT option will have meaning.

SHIFT ON|OFF: If SHIFT is on, the origin time is set to zero, and other time related header variables are shifted back to be consistent with the origin time. Some of the distance related header variables are also affected. SHIFT ON is the default.

SCALE ON|OFF: The SCALE option is OFF by default. When SCALE is OFF, SAC reads the counts from the GSE file. When SCALE is ON, SAC multiplies counts by a SCALE factor given as CALIB in the GSE file. This scale option changes with the SCALE options of READ SEGY, READCSS, and READSUDS. If SCALE is OFF, the CALIB value from the file will be stored in SAC's SCALE header variable. If SCALE is on, SAC's SCALE header field is set to 1.0. SCALE is a crude method of accounting for the instrument response. The prefered method is with the TRANSFER command. It is recommended to leave SCALE off for the READ and use the TRANSFER command. SCALE should really only be used if the response information necessary for the TRANSFER command is not available.

COMMIT: If the MORE option is specified, the COMMIT option commits headers and waveforms in SAC memory -- removing any previous versions of headers or waveforms from RAM -- prior to reading more files. COMMIT is the default.

ROLLBACK: If the MORE option is specified, the ROLLBACK option reverts to the last committed version of the header and waveform before reading more files.

RECALLTRACE: If the MORE option is specified, the RECALLTRACE option:

- reverts to the last committed version of the waveform,
- reverts to the last committed version of those header variables closely linked to the waveform,
- commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

Note: if the MORE option is not specified, the COMMIT, ROLLBACK, and RECALL-TRACE options have no effect.

DIR name: The directory to be searched for gsefile(s).

filelist: The name(s) of one or more gse files.

DEFAULT VALUES

```
READGSE * MAXMEM 0.3 VERBOSE OFF COMMIT
```

DESCRIPTION

See the READ command. October 27, 1998 (Version 00.58)

Any command which loads data into memory is monitored to maintain a level of confidence in the event infomation when transfered from the SAC data buffer to the CSS data buffer. When READGSE is used, the confidence is set to LOW, indicating that SAC should consider any matching event IDs as artifacts and reassign the event ID of the incoming file. For more details, use HELP READ.

NOTES

The following GSE Data messages can be read:

- WAVEFORM
- STATION
- CHANNEL
- ARRIVAL

Waveform formats of INT, CM6, and CM8 can be read.

Arrivals, although read, will not appear in SAC since the DETECTIONS message is not yet read, and without a DETECTION ID, arrivals cannot be associated with channels.

LATEST REVISION

April 22, 1999 (Version 00.58)

READHDR

SUMMARY

Reads headers from SAC data files into memory.

SYNTAX

```
READHDR [options] [filelist]
```

where options is one or more of the following:

```
MORE
TRUST ON|OFF
COMMIT|ROLLBACK|RECALLTRACE
DIR CURRENT|name
```

ALL options MUST preceed any element in the filelist.

INPUT

MORE: Place the new data file headers in memory AFTER the old ones. If this option is omitted, the new data file headers REPLACE the old ones.

Note: if the MORE option is not specified, the COMMIT, ROLLBACK, and RECALLTRACE options have no effect.

TRUST ON|OFF: This option is used to resolve an ambiguity in converting files from SAC to CSS format. When converting the data, matching event IDs could mean the files have identical event information, or they could be an artifact of the merging of these two very different formats. When TRUST is ON, SAC is more likely to accept matching event IDs as identical event information than when TRUST is OFF, depending on the history of READ commands associated with the current data files in memory.

COMMIT: If the MORE option is specified, the COMMIT option commits headers and waveforms in SAC memory -- removing any previous versions of headers or waveforms from RAM -- prior to reading more files. COMMIT is the default.

ROLLBACK: If the MORE option is specified, the ROLLBACK option reverts to the last committed version of the header and waveform before reading more files.

RECALLTRACE: If the MORE option is specified, the RECALLTRACE option:

- reverts to the last committed version of the waveform,
- reverts to the last committed version of those header variables closely linked to the waveform,
- commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

DIR CURRENT: Read all simple filenames (with or without wildcards) from the current directory. This is the directory from which you started SAC.

DIR name: Read all simple filenames (with or without wildcards) from the directory called name. This may be a relative or absolute directory name.

filelist: file | wild .

file: A legal filename. This may be a simple filename or a pathname. The pathname can be a relative or absolute one. See the DESCRIPTION and EXAMPLES sections of the READ command for more details.

wild: A wildcard laden token that expands to a list of filenames. See the DESCRIPTION and EXAMPLES sections of the READ command and the WILD command for more details.

DESCRIPTION

This command reads the headers from a set of SAC files into memory. You can then list the header contents (LISTHDR), change header values (CHNHDR), and then write the headers back to disk (WRITEHDR). This is much faster than reading entire files into memory, when only the headers are needed.

All commands which load data into memory have are now monitored to maintain a level of confidence in the event information when moved from the SAC data buffer to the CSS data buffer. For READHDR, when the confidence is HIGH that all the data files are cosistent in the numbering of event IDs, matching event IDs are treated as having identical event information. When the confidence is LOW in READHDR, matching event IDs are understood as an artifact, and new event IDs are generated for the incoming file. For more details use HELP READ.

ERROR MESSAGES

- 1301: No data files read in.
  - haven't given a list of files to read.
  - none of the files in the list could be read.
- 1314: Data file list can't begin with a number.
- 1315: Maximum number of files in data file list is
- 1335: Illegal operation---only data file headers in memory.
  - only LISTHDR, CHNHDR, and WRITEHDR operations. can be performed after a READHDR.

WARNING MESSAGES

- 0101: opening file
- 0108: File does not exist:
- 0114: reading file
  - Normally when SAC encounters one of these errors it skips that file and reads the remainder. These errors can be made to be fatal using the READERR command.

SEE COMMANDS

READ, LISTHDR, CHNHDR, WRITEHDR, READERR, COMMIT, ROLLBACK, RECALLTRACE

LATEST REVISION

Oct. 27, 1998 (Version 0.58)

READSDD

SUMMARY

Reads data from SDD data files on disk into memory.

SYNTAX

```
READSDD [options] [filelist]
```

where options is one or more of the following:

```
MORE
COMMIT|ROLLBACK|RECALLTRACE
DIR CURRENT|name
```

ALL options MUST preceed any element in the filelist.

INPUT

MORE: Place the new data files in memory AFTER the old ones. If this option is omitted, the new data files REPLACE the old ones.

Note: if the MORE option is not specified, the COMMIT, ROLLBACK, and RECALLTRACE options have no effect.

COMMIT: If the MORE option is specified, the COMMIT option commits headers and waveforms in SAC memory -- removing any previous versions of headers or waveforms from RAM -- prior to reading more files. COMMIT is the default.

ROLLBACK: If the MORE option is specified, the ROLLBACK option reverts to the last committed version of the header and waveform before reading more files.

RECALLTRACE: If the MORE option is specified, the RECALLTRACE option:

- reverts to the last committed version of the waveform,
- reverts to the last committed version of those header variables closely linked to the waveform,
- commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

DIR CURRENT: Read all simple filenames (with or without wildcards) from the current directory. This is the directory from which you started SAC.

DIR name: Read all simple filenames (with or without wildcards) from the directory called name. This may be a relative or absolute directory name.

filelist: file|wild .

file: A legal filename. This may be a simple filename or a pathname. The pathname can be a relative or absolute one.

wild: A wildcard laden token that expands to a list of filenames. more details.

DEFAULT VALUES

```
READ COMMIT DIR CURRENT
```

DESCRIPTION

All the same restrictions apply to READSDD as to the READ command. See the READ command DESCRIPTION and EXAMPLES sections for more detail.

Any command which loads data into memory is monitored to maintain a level of confidence in the event infomation when transfered from the SAC data buffer to the CSS data buffer. When READSDD is used, the confidence is set to LOW, indicating that SAC should consider any matching event IDs as artifacts and reassign the event ID of the incoming file. For more details, use HELP READ.

LATEST REVISION

Oct. 27, 1998 (Version 0.58)

READSP

SUMMARY

Reads spectral files written by WRITESP and WRITESPE.

SYNTAX

```
READSP {AMPH|RLIM|SPE} {filelist}
```

INPUT

RLIM: Read real and imaginary components.

AMPH: Read amplitude and phase components.

SPE: Read spectral estimation subprocess files. The data is converted from power to amplitude. The phase component is set to zeros.

filelist: A list of SAC binary data files. This list may contain simple filenames, full or relative pathnames, and wildcard characters. See the READ command for a complete description.

DEFAULT VALUES

```
READSP AMPH
```

DESCRIPTION

The WRITESP command writes each spectral data component to disk as a separate file. You may then process each component separately. This command lets you reconstruct the spectral data from the two components. See the WRITESP documentation for more details. The SPE option allows you to read in and convert to spectral format, files that were written using the WRITESPE command in the Spectral Estimation Subprocess. This allows you to use commands such as MULOMEGA and DIVOMEGA on these spectral estimates.

Any command which loads data into memory is monitored to maintain a level of confidence in the event infomation when transfered from the SAC data buffer to the CSS data buffer. When READSP is used, the confidence is set to LOW, indicating that SAC should consider any matching event IDs as artifacts and reassign the event ID of the incoming file. For more details, use HELP READ.

EXAMPLES

See the example in the WRITESP documentation.

SEE COMMANDS

WRITESP

REFERENCES

Spectral Estimation Subprocess Manual

LATEST REVISION

April 21, 1989 (Version 10.4c)

READSUDS

SUMMARY

Read data files in PC-SUDS format from disk into memory.

Note SUDS data enters SAC via SAC's CSS data buffers. To understand how arrivals are handled, use HELP READCSS and HELP PICKPREFS.

SYNTAX

```
READSUDS {MAXMEM v} {MORE} {VERBOSE ON|OFF} {SHIFT ON|OFF}
{COMMIT|ROLLBACK|RECALLTRACE} {DIR name}  filelist
```

INPUT

MAXMEM: Specify the maximum fraction of physical memory to use when reading large data sets. When this limit is reached, no more waveforms will be read, although other tables may still be read. The default value for MAXMEM is 0.3.

MORE: See the READ command.

VERBOSE ON|OFF: If VERBOSE is ON, SAC displays extended information about the waveforms being read, and prints a summary of the CSS tables that were filled. SAC also displays a progress indicator for the conversion to SAC internal format.

SHIFT ON|OFF: If SHIFT is on, the origin time is set to zero, and other time related header variables are shifted back to be consistent with the origin time. Some of the distance related header variables are also affected. SHIFT ON is the default.

COMMIT: If the MORE option is specified, the COMMIT option commits headers and waveforms in SAC memory -- removing any previous versions of headers or waveforms from RAM -- prior to reading more files. COMMIT is the default.

ROLLBACK: If the MORE option is specified, the ROLLBACK option reverts to the last committed version of the header and waveform before reading more files.

RECALLTRACE: If the MORE option is specified, the RECALLTRACE option:

- reverts to the last committed version of the waveform,
- reverts to the last committed version of those header variables closely linked to the waveform,
- commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

Note if the MORE option is not specified, the COMMIT, ROLLBACK, and RECALLTRACE options have no effect.

DIR name: The directory to be searched for sudsfile(s).

filelist: The name(s) of one or more suds files.

DEFAULT VALUES

```
READSUDS * MAXMEM 0.3 VERBOSE OFF COMMIT
```

DESCRIPTION

See the READ command. Oct. 27, 1998 (Version 00.58)

Any command which loads data into memory is monitored to maintain a level of confidence in the event infomation when transfered from the SAC data buffer to the CSS data buffer. When READSUDS is used, the confidence is set to LOW, indicating that SAC should consider any matching event IDs as artifacts and reassign the event ID of the incoming file. For more details, use HELP READ.

NOTES

READSUDS assumes that the data are still in PC byte-order, and swaps bytes as necessary while reading the files.

The following SUDS headers should be populated:

- DESCRIPTRACE
- STATIONCOMP
- FEATURE
- EVENT
- ORIGIN

Statident structs for a given channel must have all fields set identically to allow joining: i.e. dt->dt_name = fe->fe_name = sc->sc_name.

There should be only 1 origin and 1 event in the SUDS file since PC-SUDS has no way to associate origins with features or descriptraces.

ev->number must equal or->number to associate the event with the origin.

SUDS Magnitude, Authority, Program, Instrument, and Phase codes must be from the following code lists in order to translate to CSS.

Suds Magnitude Codes

or->mag_type in origin (type char):

S: "ms"
b: "mb"
c: "md"
l: "ml"
m: "mw"
s: "ms"
w: "mw"

Suds Authority codes

or->authority in origin (type short): ev->authority in event

1000: "USGS-Menlo-Park"
1002: "CALNET"
1050: "RTP-USGS-Menlo-Park"
2000: "Geophysical-Institute-U-of-Alaska"
3000: "University-of-Washington"

4000: "Lamont-Doherty-Geological-Observatory"
5000: "IRIS"
5100: "GSN"
5200: "ASRO"
5300: "PASSCAL"
6000: "LLNL"
7000: "LBL"
8000: "LANL"

Suds program codes

or->program in origin (type char):

'7': "Hypo-71"
'h': "HypoInverse"
'l': "HypoLayer"
'c': "Centroid"
'v': "Velest"

Suds event codes

ev->ev_type in event (type char):

'e': "ke" known earthquake
'E': "qb" quarry blast
'n': "kn" known nuclear explosion
'i': "iq" icequake
'r': "rq" regional earthquake
't': "tq" teleseismic earthquake
'K': "kr" known rockburst
'k': "sr" suspected rockburst
'm': "sm" suspected mine explosion
'M': "km" known mine explosion
's': "se" suspected earthquake
'S': "sn" suspected nuclear explosion
'l': "ls" landslide
'd': "si" suspected induced event
'D': "ki" known induced event
'x': "sx" suspected experimental explosion
'X': "kx" known experimental explosion

Suds instrument codes

suds_statident->inst_type (type short):

0: "Unk"
1: "sp-usgs"

2: "sp-wwssn"
3: "lp-wwssn"
4: "sp-dwwssn"
5: "lp-dwwssn"
6: "hglp-lamont"
7: "lp-hglp-lamont"
8: "sp-sro"
9: "lp-sro"
10: "sp-asro"
11: "lp-asro"
12: "sp-rstn"
13: "lp-rstn"
14: "sp-uofa-U-of-Alaska"
15: "STS-1/UVBB"
16: "STS-1/VBB"
17: "STS-2"
18: "FBA-23"
19: "Wilcoxin"
50: "USGS-cassette"
51: "GEOS"
52: "EDA"
53: "Sprengnether-refraction"
54: "Teledyne-refraction"
55: "Kinemetrics-refraction"
300: "amplifier"
301: "amp/vco"
302: "filter"
303: "summing-amp"
304: "transmitter"
305: "receiver"
306: "antenna"
307: "battery"
308: "solar-cell"
309: "discriminator"
310: "discr-rack"
311: "paper-recorder"
312: "film recorder"
313: "smoked glass recorder"
314: "atod convertor"
315: "computer"
316: "clock"
317: "time receiver"
318: "magnetic tape"
319: "magntic disk"
320: "optical disk"

SUDS Phases

suds phases in fe->feature (type short)

0: "none"
1: "window"
2: "f finis"
3: "MaxAmp"
50: "P-first"
51: "P"
52: "P*"
53: "PP"
54: "PPP"
55: "PPPP"
56: "PPS"
57: "Pg"
58: "Pn"
59: "Pdiff"
60: "PcP"
61: "PcPPKP"
62: "PcS"
63: "pP"
64: "pPP"
65: "PKP"
66: "PKPPKP"
67: "PKPPKS"
68: "PKPSKS"
69: "PKS"
70: "pPKS"
71: "PKKP"
72: "PKKS"
73: "PcPPKP"
74: "PcSPKP"
100: "S-first"
101: "S"
102: "S*"
103: "SS"
104: "SSS"
105: "SSSS"
106: "Sg"
107: "Sn"
108: "ScS"
109: "SPcS"
110: "sS"
111: "sSS"
112: "sSSS"
113: "SScS"

114: "ScSPKP"
115: "ScP"
116: "SKS"
117: "SKKS"
118: "SKKKS"
119: "SKSSKS"
120: "SKP"
121: "SKKP"
122: "SKKKP"
201: "Lg"
202: "Lr"
203: "Lr2"
204: "Lr3"
205: "Lr4"
206: "Lq"
207: "Lq2"
208: "Lq3"
209: "Lq4"
301: "t"

SEE COMMANDS

READ, PICKAUTHOR, PICKPHASE, COMMIT, ROLLBACK, RECALLTRACE

LATEST REVISION

October 27, 1998 (Version 00.58)

READTABLE

SUMMARY

Reads alphanumeric data files in collumn format on disk into memory.

SYNTAX

```
READTABLE {options}  {filelist}
```

where options is one or more of the following:

```
MORE
TRUST ON|OFF
COMMIT|ROLLBACK|RECALLTRACE
DIR CURRENT|name
FREE|FORMAT text  **** NOTE:  the FORMAT option is not working.  ****
CONTENT text
HEADER number
```

ALL options MUST preceed any element in the filelist. The last two options may also be placed on the first line of file itself.

INPUT

MORE: Append the new data files after the old ones in memory. If this option is missing, the new data replaces the old data in memory. See the READ command for more details about this option.

Note: if the MORE option is not specified, the COMMIT, ROLLBACK, and RECALLTRACE options have no effect.

TRUST: This option is used to resolve an ambiguity in converting files from SAC to CSS format. When converting the data, matching event IDs could mean the files have identical event information, or they could be an artifact of the merging of these two very different formats. When TRUST is ON, SAC is more likely to accept matching event IDs as identical event information than when TRUST is OFF, depending on the history of READ commands associated with the current data files in memory.

COMMIT: If the MORE option is specified, the COMMIT option commits headers and waveforms in SAC memory -- removing any previous versions of headers or waveforms from RAM -- prior to reading more files. COMMIT is the default.

ROLLBACK: If the MORE option is specified, the ROLLBACK option reverts to the last committed version of the header and waveform before reading more files.

RECALLTRACE: If the MORE option is specified, the RECALLTRACE option:

- reverts to the last committed version of the waveform
- reverts to the last committed version of those header variables closely linked to the waveform,
- commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

DIR CURRENT: Read all simple filenames (with or without wildcards) from the current directory. This is the directory from which you started SAC.

DIR name: Read all simple filenames (with or without wildcards) from the directory called name. This may be a relative or absolute directory name.

FREE: Read the data in the filelist in free format (space delimited) mode.

FORMAT text: Read the data in the filelist in fixed format mode. The format statement to use is given in text.

CONTENT text: Define the content of the data in the filelist. The meaning of the content text is described below.

HEADER: The number of header lines in the file to skip.

filelist: A list of alphanumeric files. This list may contain simple filenames, full or relative pathnames, and wildcard characters. See the READ command for a complete description.

DEFAULT VALUES

```
READTABLE COMMIT FREE CONTENT Y. DIR CURRENT
```

DESCRIPTION

All commands in SAC work on the data that is currently in memory. This data in memory is analogous to the temporary or working files used by a text editor. The READ command reads binary SAC data files into memory. This command can be used to read a wide variety of alphanumeric data files into memory. These files can be in a fixed format or in free format. They may contain evenly or unevenly spaced data. They may contain more than one set of data. Once in memory the WRITE command can be used to create SAC binary data files for later use.

The simplest use of this command is free field input of a Y data set. This is also the default. Free field input of X-Y pairs can be done by simply changing the content option. By combining the fixed format and content options, this command can also be used to read very complicated formatted output from other programs directly into SAC. Multiple Y data sets can be read from the same file using this method. Only a single X data set is allowed.

The basic header variables needed for processing are computed. These are NPTS, B, E, DELTA, LEVEN, DEPMIN, DEPMAX, and DEPMIN. If there is only a single Y data set, the name of the data file in memory will be the same as that of the alphanumeric disk file. If there are multiple Y data sets in the file, a two digit sequence number is appended to the file name.

Each line of the alphanumeric data file is read in either free format or using the format statement provided. Each line can be up to 160 characters long. In the case of a free format file, the number of data entries in each line is also determined. The content field is then used to determine what to do with each of these data entries. Each specific character in the context field represents a different kind of data element and the order of these characters mimics the order of the data in each line of the file. The meanings of the allowed characters in the content field are given below:

Y: Next entry belongs to Y (dependent variable) data set.

X: Next entry belongs to X (independent variable) data set.

N: Next entry belongs to next Y data set.

P: Next pair of entries belong to X and Y data sets.

R: Next pair of entries belong to Y and X data sets.

I: Ignore (skip) this data element.

An optional repetition count may follow any of the above characters. This repetition count is a one or two digit integer and has the same meaning as repeating the content character that number of times. A period (".") is an infinite repetition count and means use the last characters meaning to

decode the remaining data elements in the line. The period can only appear at the end of a content field.

Any command which loads data into memory is monitored to maintain a level of confidence in the event infomation when transfered from the SAC data buffer to the CSS data buffer. When READTABLE is used, the confidence is set to LOW, indicating that SAC should consider any matching event IDs as artifacts and reassign the event ID of the incoming file. For more details, use HELP READ.

EXAMPLES

To read in X-Y pairs in free format where there may be any number of pairs on the same line:

```
u:  READTABLE CONTENT P. FILEA
```

You can't break an X-Y pair between lines in the file. Assume you have a file which contains formatted data including a set of X and Y data buried somewhere in the middle of each line. Other data that is of no interest is also on each line. Also assume that the Y data preceeds the X data in each line. Once the format statement needed to read in the proper data is determined, the following command could be used:

```
u: READTABLE CONTENT R FORMAT \(24X,F12.3,14X,F10.2\) FILEB
```

Special Note: The atsign "" BEFORE EACH LEFT AND RIGHT PARENTHESIS IS SAC's escape character, and is necessary because SAC uses parenthesis in inline functions. Since there is no repeat count, only a single Y-X pair will be read from each line of the file.

Assume you have a file, FILEC, which contains a table consisting of an X value followed by Y values belonging to seven different data sets on each line. This data is in (8F10.2) format. To create seven different sets of data in memory, the following command could be used:

```
u:  READTABLE CONTENT XN . FORMAT \(8F10.2\) FILEC
```

This would produce seven different "data files" in memory with the names FILEC01, FILEC02, etc. Now assume that you did not want the fifth Y data set to be read. This could be done by executing the following command:

```
u:  READTABLE CONTENT XN6 FORMAT \(5F10.20X,2F10.2\) FILEC
```

Another way that means less typing but is slightly less efficient is given below:

```
u:  READTABLE CONTENT XN4IN2 FORMAT \(8F10.2\) FILEC
```

```
Note:  for examples of the behavior of the COMMIT, ROLLBACK, and RECALLTRACE
options, see the commands of the same names.
```

ERROR MESSAGES

- 1301: No data files read in.
  - haven't given a list of files to read.
  - none of the files in the list could be read.
- 1020: Invalid inline function name:
  - Expected inline function. Preceed parenthesis with an atsign.
- 1320: Available memory too small to read file
- 1314: Data file list can't begin with a number.
- 1315: Maximum number of files in data file list is

WARNING MESSAGES

- 0101: opening file
- 0108: File does not exist:

HEADER CHANGES

B, E, DELTA, LEVEN, DEPMIN, DEPMAX, DEPMEN.

SEE COMMANDS

READ, WRITE, COMMIT, ROLLBACK, RECALLTRACE

LATEST REVISION

Oct. 27, 1998 (Version 0.58)

REPORT

SUMMARY

Informs the user about the current state of SAC.

SYNTAX

```
REPORT {list}
```

where list is one or more of the following:

```
APF, COLOR, CUT,
DEVICES, FILEID, GTEXT,
HPF, LINE, MEMORY,
MTW, PICKS, SYMBOL,
TITLE, XLABEL,
XLIM, YLABEL, YLIM
```

INPUT

APF: The name of the alphanumeric pick file.

COLOR: The current color attributes. No color table is read in until a graphics device is activated. Unless a graphics device has been activated, this report will not be correct.

CUT: The current CUT status.

DEVICES: A list of the graphics devices available on your system.

FILEID: The current file id display attributes.

GTEXT: The current graphics text attributes.

HPF: The name of the HYPO pick file.

LINE: The current linestyle attributes.

MEMORY: A dump of the available memory blocks from the memory manager. This is probably of little interest unless the memory manager is not working properly.

MTW: The current measurement time window status.

PICKS: The current time pick display attributes.

SYMBOL: The current symbol drawing attributes.

TITLE: The current plot title attributes.

XLABEL: The current x axis label attributes.

XLIM: The current x axis plot limits.

YLABEL: The current y axis label attributes.

YLIM: The current y axis plot limits.

DESCRIPTION

This command can be used to find out about the current values of certain SAC options. The values are printed to the terminal.

EXAMPLES

To get a list of the current color attributes:

```
u: REPORT COLOR
s:   COLOR option is ON
s:   DATA color is YELLOW
s:   INCREMENT data color is OFF
s:   SKELETON color is BLUE
s:   BACKGROUND color is NORMAL
```

To get the names of the HYPO and card image pick files:

```
u: REPORT APF HPF
s:   Alphanumeric pick file is MYPICKFILE
s:   HYPO pick file is HYPOPICKFILE
```

LATEST REVISION

March 20, 1991 (Version 10.6e)

REVERSE

SUMMARY

Reverse the order of data points.

SYNTAX

```
REVERSE
```

DESCRIPTION

This command reverses the order of data points in each file in memory.

LATEST REVISION

May 15, 1987 (Version 10.2)

RGLITCHES

SUMMARY

Removes glitches and timing marks.

SYNTAX

```
RGLITCHES options
```

where options are one or more of the following:

```
THRESHOLD v
TYPE LINEAR|ZERO
WINDOW ON|OFF|pdw
METHOD ABSOLUTE|POWER|RUNAVG
```

INPUT

THRESHOLD v: Set onset threshold level to v. Data points whose absolute values are greater than or equal to v are corrected.

TYPE LINEAR: Correct data points above the threshold by linearly interpolating between the data points on each side of the bad data.

TYPE ZERO: Correct data points above the threshold by setting them to zero.

METHOD ABSOLUTE: Corrects data points having absolute values >= the threshold v.

METHOD POWER: Corrects data points where the power of the signal computed using a backward difference method exceeds the threshold v.

METHOD RUNAVG: Corrects data points by calculating a running average and standard deviation in a window SWINLEN seconds long that moves from the end of the trace to the beginning of the trace in 1-point increments. Each new point is compared to the average, and if it differs by more than THRESH2 times the current standard deviation, and if the difference is greater than MINAMP counts, it is replaced by the current mean. This method is always applied to the entire seismogram.

There are three options associated with the RUNAVG method. These are:

SWINLEN v: Set length in seconds of running average window.

THRESH2 v: Set the threshold value for glitches.

MINAMP v: Set the minimum amplitude for glitches.

WINDOW ON: Only correct data points within the previously defined pdw.

WINDOW OFF: Correct data points within the entire data file.

WINDOW pdw: Only correct data points within the defined pdw. A pdw consists of a starting and a stopping value of the independent variable, usually time, which defines the desired window of data that you wish to make measurements on. See the CUT command for a complete explanation of how to define and use a pdw. Some examples are given below.

DEFAULT VALUES

```
RGLITCHES THRESHOLD 1.0E+10 TYPE LINEAR WINDOW OFF METHOD ABSOLUTE SWINLEN
0.5
THRESH2 5.0 MINAMP 50
```

DESCRIPTION

This command can be used to smooth out irregularities caused by "glitches" in the data acquistion system and by timing marks produced by some data acquistion systems. It checks each data point to see if it's value is greater than or equal to the requested "onset threshold level". It then zeros out these bad data points or linearly interpolates between the data point just before and the data point just after the bad ones. You can have it remove glitches in the entire file or select a smaller portion of the file by setting the window. Using this option lets you remove glitches that are smaller than the maximum in the entire data file.

EXAMPLES

Some examples of pdw are given below:

```
B 0 30: First 30 secs of the file.
T3 -1 T7: From 1 sec before T3 time pick to T7 time pick.
30.2 48: 30.2 to 48 secs relative to file zero.
```

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES

- 1301: No data files read in.
- 1306: Illegal operation on unevenly spaced file
- 1307: Illegal operation on spectral file

LATEST REVISION

March, 1997 (Version 00.53a)

RMEAN

SUMMARY

Removes the mean.

SYNTAX

RMEAN

ERROR MESSAGES

- 1301: No data files read in.
- 1307: Illegal operation on spectral file

HEADER CHANGES

DEPMEN

LATEST REVISION

October 11, 1984 (Version 9.1)

RMS

SUMMARY

Computes the root mean square of the data within the measurement time window.

SYNTAX

```
RMS {NOISE ON|OFF|pdw},{TO USERn}
```

INPUT

NOISE ON: Turn noise normalization option on.

NOISE OFF: Turn noise normalization option off.

NOISE pdw: Turn noise normalization option on and change noise "partial data window." A pdw consists of a starting and a stopping value of the independent variable, usually time, which defines the desired window of data that you wish to make measurements on. See the CUT command for a complete explanation of how to define and use a pdw. Some examples are given below.

TO USERn: Define the user header variable in which to store the result. n is an integer in the range 0 to 9.

DEFAULT VALUES

```
RMS NOISE OFF TO USER0
```

DESCRIPTION

This command computes the root mean square of the data within the current measurement time window (see MTW.) The result is written into one of the floating point user header variables. The result may be corrected for noise if desired by defining a noise window. The general form of the calculation is: where the first summation is over the signal window and the second is over the optional noise window.

EXAMPLES

To compute the uncorrected root mean square of data between the two header fields, T1 and T2, and to store the result into the USER4 header field:

u: MTW T1 T2 u: RMS TO USER4

To compute the corrected root mean square using a noise window 5 seconds long ending at the header field T3:

u: MTW T1 T2 u: RMS NOISE T3 -5.0 0.0

HEADER CHANGES

USERn

SEE COMMANDS

MTW, CUT

LATEST REVISION

March 22, 1991 (Version 10.6d)

ROTATE

SUMMARY

Rotates a pair of data components through an angle.

SYNTAX

```
ROTATE {to GCP|TO v|THROUGH v|,{NORMAL|REVERSED}
```

INPUT

TO GCP: Rotate to the "great circle path". Both components must be horizontals. The station and event coordinates header fields must be defined.

TO v: Rotate to the angle v in degrees. Both components must be horizontals.

THROUGH v: Rotate through the angle v in degrees. One component may be vertical.

NORMAL: Output (horizontal) components with the second leading the first by 90 degrees

REVERSED: Output (horizontal) components with the second lagging the first by 90 degrees.

DEFAULT VALUES

```
ROTATE TO GCP NORMAL
```

DESCRIPTION

Pairs of data components are rotated in this command. Each pair must have the same station name, event name, and sampling rate. In the THROUGH option both components are simply rotated through the requested angle. One of those components may be a vertical. Rotations in the horizontal plane are clockwise from north. Rotations with a vertical component are clockwise from up.

Both components must be horizontals when the TO option is used. This means that CMPAZ must be definined and that CMPINC must be 90 degrees. After the rotation is completed the first component of each pair will be directed along the angle given after to TO keyword. If the TO GCP option is used this component will be directed along the angle given by the station-event back azimuth plus or minus 180 degrees. This component therefore points from the event toward the station. The station and event coordinates header fields (STLA, STLO, EVLA, and EVLO) must be defined so that the back azimuth can be calculated.

The NORMAL and REVERSED options also apply only to horizontal rotations. If the NORMAL option is used the second component leads the first by 90 degrees. If the REVERSED option is used it lags the first by 90 degrees.

EXAMPLES

To rotate a pair of horizontals through 123.43 degrees:

```
SAC> READ XYZ.N XYZ.E
SAC> ROTATE TO 123.43
```

To rotate two sets of horizontals to the great circle path:

```
SAC> READ ABC.N ABC.E DEF.N DEF.E
SAC> ROTATE TO GCP
SAC> W ABC.R ABC.T DEF.R DEF.T
```

In the above example if the BAZ header variable had been 33 degrees, the radial components would be at 213 degrees and the tangential components at 303 degrees. If reversed polarity had been requested the tangential components would be at 123 degrees.

HEADER CHANGES

CMPAZ, CMPINC

ERROR MESSAGES

- 1301: No data files read in.
- 2001: Command requires an even number of data files.
- 2004: Insufficient header information for rotation:
  - STLA, STLO, EVLA, EVLO must be defined for GCP option.
- 2002: Following files are not an orthogonal pair:
- 2003: Following files are not both horizontals:
  - TO option only works on horizontals.

LATEST REVISION

January 8, 1983 (Version 8.0)

RQ

SUMMARY

Removes the seismic Q factor from spectral data.

SYNTAX

```
RQ [Q v],[R v],[C v]
```

INPUT

Q v: Set quality factor to v.
R v: Set distance in km. to v.
C v: Set group velocity in km/sec to v

DEFAULT VALUES

```
RQ Q 1.  R 0.  C 1.
```

DESCRIPTION

The equation used to correct the amplitude is given below:

```
AMP_corrected(F) = AMP_uncorrected(F) * Exp( (pi*R*F) / (Q*C) )
```

where: F is the frequency in Hz. R is the distance in km. C is the group velocity in km/sec. Q is the the nondimensional quality factor.

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES

- 1301: No data files read in.
- 1305: Illegal operation on time series file

WARNING MESSAGES

- 1604: Following file now in amplitude-phase format:
  - file was in real-imaginary format.

LIMITATIONS

Can only handle constants for the various parameters. Modifications to allow these parameters to vary with frequency may be added at a later date.

LATEST REVISION

January 8, 1983 (Version 8.0)

RTREND

SUMMARY

Removes the linear trend.

SYNTAX

```
[RTR]END  [QUIET [ON | OFF]] [VERBOSE [ON | OFF]]
```

INPUT

QUIET: If ON, supresses screen output
VERBOSE: If on, there is output to the screen about slope removed, et.
DEFAULT VALUES: RTR QUIET

DESCRIPTION

A least-squares curve-fit to a straight line is calculated. This straight line (trend and intercept) is then "subtracted" from the data. The data does not have to be evenly spaced.

OUTPUT: The best-fitting straight line parameters for the last file in the data file list are written to blackboard variables begining with RTR.

- RTR_SLP is the slope of the line.
- RTR_SDSLP is the standard deviation in the slope.
- RTR_YINT is the y intercept of the line.
- RTR_SDYINT is the standard deviation in the y intercept.
- RTR_SDDTA is the standard deviation in the data.
- RTR_CORRCF is the data correlation coeficient.

ERROR MESSAGES

- 1301: No data files read in.
- 1307: Illegal operation on spectral file

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

LATEST REVISION

July, 2011 (Version 101.5)

SAVEIMG

SUMMARY

Saves a display graphics window to an image file in a variety of formats

SYNTAX

```
[SAVE]IMG filename.format
```

INPUT

filename: Filename for the saved image.

format: Format for saved file from among one of the four choices: ps (Postscript); pdf (Portable Document Format); png (Image file) xpm (Pixmap format)

DESCRIPTION

This command will save a current plot to an image file format including Postscript (ps), Portable Document Format (pdf), Image file (png), and Pixmap file (xpm). The format is derived from the filename's extension.

An advantage of SAVEIMG to producing SGF files is that letters and numbers in .sgf files are made up of drawn line segments, while those in the .ps or .pdf images produced by SAVEIMG use the Postscript feature of producing fonts directly. Also, for many applications, a lower-resolution .png or ,xpn file is sufficient. Because of potential problems with portability, the PNG format is not enabled in the default builds of SAC.

The .xpm and .png files will have the aspect ratio of the current window. (See WINDOW for instructions about choosing the size and aspect ratio of display windows.) The .pdf and .ps files produced by SAVEIMG will have a fixed aspect ratio of X/Y = 11/8.5 = 1,2941. For these plots, the plots will look best if the display window aspect ratio is 1.2941.

As with .sgf files, the plots will not have a tight boundingbox. For .sgf files the script SACDIR/bin/sgftoeps.csh will produce an EPS file with a tight boundingbox if the program Ghsostscript (gs) is in the path. Similar scripts could be written for the output files from SAVEIMG.

To save a plot using SAVEIMG, the plot must already be visible. SAVEIMG will not work in the SSS subprocess, but if one enters qs after creating the plot, SAVEIMG can then be used for that image. Also, if a frame has been opened to produce multiple panels in a single file, saveimg cannot be used until after the ENDFRAME command.

EXAMPLES

To save a file as a PDF document:

```
SAC> read PAS.CI.BHZ.sac
SAC> p1
SAC> saveimg pas.ci.pdf
```

To save a spectrogram in three different formats:

```
SAC> fg seismo
SAC> spectrogram
SAC> save spectrogram.ps
SAC> save spectrogram.xpm
SAC> save spectrogram.pdf
```

LATEST REVISION

version 101.6

SCALLOP

Note: This command has been renamed SONOGRAM (SONO). Either command, SONOGRAM or SCALLOP, will do the same thing.

SUMMARY

Calculate a spectrogram equal to the difference between two smoothed versions of the same spectrogram.

SYNTAX

```
SCALLOP options
```

where options are one or more of the following:

```
WINDOW v
SLICE v
ORDER n
CBAR    {ON|OFF}
YMIN v
YMAX v
FMIN v
FMAX v
BINARY|FULL
METHOD {PDS|MEM|MLM}
{COLOR|GRAY}
PRINT {pname}
```

INPUT

WINDOW v: Set the sliding data window length in seconds to v. This window length determines the size of the fft.

SLICE v: Set the data slice interval in seconds to v. A single spectrogram line is produced for each slice interval.

ORDER n: Specifies the number of points in the autocorrelation function used to compute the spectral estimate.

CBAR {ON|OFF}: Turn reference color bar on or off.

BINARY|FULL: Produce a binary image, or a full color image.

YMIN v: Specifies the minimum frequency to plot.

YMAX v: Specifies the maximum frequency to plot.

FMIN v: Specifies the smallest bandwidth over which each slice in the spectrogram will be smoothed.

FMAX v: Specifies the maximum bandwidth over which each slice in the spectrogram will be smoothed.

METHOD {PDS|MEM|MLM}: Specifies the type of spectral estimator used. MLM stands for maximum likelihood and MEM stands for maximum entropy spectral estimators, respectively. See description and references below.

{COLOR|GRAY}: Specifies a color or grayscale image.

PRINT {pname}: Prints the resulting plot to the printer named in pname, or to the default printer if pname is not used. (This makes use of the SGF capability.)

DEFAULT VALUES

```
SCALLOP WINDOW 2 SLICE 1 METHOD MEM ORDER 100 YMIN 0 YMAX FNYQUIST FMIN
2.0
fmax 6.0 full color
```

DESCRIPTION

The scalloping command computes a spectrogram equal to the difference between two smoothed version of the same spectrogram. Depending on the choice of smoothing parameters, fmin and fmax, the resulting spectrogram can enhance small amplitude spectral features that are more difficult to observe in a conventional spectrogram. This is particularly useful when looking for features like high frequency spectral modulations in seismic signals from mine blasts (c.f., Hedlin, 1990, Wuster, 1993).

LIMITATIONS

The size of the image in the frequency direction is 512.

PROBLEMS

There is currently very little error checking of the headers to make sure that they are from the same component and are contiguous in time. This will be corrected in the future.

HEADER VARIABLES

REQUIRED: DELTA
CHANGED: NPTS, DELTA, B, E, IFTYPE, DEPMIN, DEPMAX, DEPMEN
CREATED: NXSIZE, XMINIMUM, XMAXIMUM, ,BREAK NYSIZE, YMINIMUM, YMAXIMUM

LATEST REVISION

May 26, 1995 (Version 00.31)

SETBB

SUMMARY

Sets (defines) values of blackboard variables.

SYNTAX

```
SETBB variable {APPEND} value {variable {APPEND} value ...}
```

INPUT

variable: The name of a blackboard variable. It may be a new variable or one that already has a value. The variable name can be up to 32 characters in length.

value: The new value of that blackboard variable. It must be enclosed in single or double quotes if it contains any spaces.

APPEND: Append value to the old value of variable. If this option is omitted then the new value replaces the old value.

DESCRIPTION

The blackboard is a place to temporarily store information. This information can later be accessed by the GETBB command or used directly in a command by preceeding the name of the variable with a percent sign ("%".) If you want to concantenate some other text string on the end of a blackboard variable you need to put a second percent sign at the end of the name. You can also use the EVALUATE command to perform basic arithmetic operations on blackboard variables and store the results in new blackboard variables. You can unset (delete) blackboard variables using the UNSETBB command.

EXAMPLES

To set several blackboard variables at once:

```
u:  SETBB C1 2.45 C2 4.94
```

To later use these variables in a command:

```
u:  BANDPASS CORNERS %C1 %C2
```

To set a blackboard variable that contains spaces:

```
u:  SETBB MYTITLE 'Sample filter response'
```

To check and make sure the value is correct:

```
u:  GETBB MYTITLE
s:  MYTITLE = Sample filter response
```

To later use this variable in the title command it must be enclosed in quotes and have a percent sign on both ends of the name:

```
u:  TITLE '%MYTITLE%'
```

See the section on Macros in the SAC Users Manual for more examples of the use of blackboard variables in macros.

SEE COMMANDS

GETBB, EVALUATE, UNSETBB

LATEST REVISION

May 15, 1987 (Version 10.2)

SETDEVICE

SUMMARY

Defines a default graphics device to use in subsequent plots.

SYNTAX

```
SETDEVICE name
```

INPUT

name: The name of a graphics device.

DESCRIPTION

This command lets you define the name of a default graphics device to use in subsequent plots. This command is only useful before you do any plotting. It should be placed in your default macro file. You can override the name specified in this command by using the BEGINDEVICES command. See the section on Graphics Devices in the SAC Users Manual. Also see the section on Macros for information on specifying and using a default macro file.

SEE COMMANDS

BEGINDEVICES

LATEST REVISION

May 15, 1987 (Version 10.2)

SETMACRO

SUMMARY

Defines a set of directories to search when executing a SAC macro file.

SYNTAX

```
SETMACRO {MORE} directory {directory ...}
```

INPUT

directory: The name of a directory in which SAC macro files are stored. This may be either a relative or absolute directory name. On some operating systems, the directory path may be case sensitive.

DESCRIPTION

This command lets you define a set of directories to search when executing SAC macro files using the MACRO command. You can define up to 100 such directories.

MORE: When the MORE option is used with setmacro, the specified directories are added to the existing list. When MORE is not used with setmacros, the existing list is replaced with the new list.

When the MACRO command is run, SAC searches for the macro in the current directory; if no file is found with the given name, SAC searches the directories listed in SETMACRO in the order that they are listed. If there are still no files found with the given name, SAC searches the global macro directory.

See the section on Macros in the SAC Users Manual.

SEE COMMANDS

MACRO

LATEST REVISION

December 5, 1996 (Version 52a)

SGF

SUMMARY

Controls the SAC Graphics File (SGF) device options.

SYNTAX

```
SGF {options}
```

where options are one or more of the following:

```
PREFIX text
NUMBER n
DIRECTORY CURRENT|pathname
SIZE NORMAL|FIXED v|SCALED v
OVERWRITE ON|OFF
```

INPUT

PREFIX text: Set the frame prefix to text (up to 24 characters long.)

NUMBER n: Set next frame number to n. If n is zero, then SAC searches the directory for SGFs and sets the frame number to the next value in the sequence.

DIRECTORY CURRENT: Put the SGFs in the current directory.

DIRECTORY pathname: Put the SGFs in the directory specified by pathname.

SIZE NORMAL: Produce a "normal" sized plot. A normal plot has a viewspace (the maximum plotting area) of 10 by 7.5 inches. Using default values, the viewport (the portion of the viewspace where the plot is drawn excluding axes and labels) itself is approximately 8 by 5 inches.

SIZE FIXED v: Produce a plot where the x viewport is v inches in length.

SIZE SCALED v: Produce a plot where the x viewport in inches is determined by multiplying v by the x world coordinate limits.

OVERWRITE ON|OFF: When it is turned on, the file numbers are not incremented. Each new file erases the previous file. This is especially useful with the PRINT option on most plot commands.

DEFAULT VALUES

```
SGF PREFIX F NUMBER 1 DIRECTORY CURRENT SIZE NORMAL
ALTERNATE NAMES: ID for PREFIX and FRAME for NUMBER.
```

DESCRIPTION

This command controls the frame naming conventions and final plot size for subsequent SAC Graphics Files. Each frame is stored in a separate file on disk. Each frame name is made up of four parts. In order they are:

pathname The optional directory path name.

prefix The frame prefix.

number The three digit frame number.

.sgf The suffix used to denote a SAC Graphics File.

By default the frame prefix is simply the letter "f", the frame number 1 and the files are put in the current directory (i.e. the first name is "f001.sgf".) You might want to changed the prefix to identify a set of files you wish to save. You can also specify a directory in which to store the files. This is very useful when you are changing directories while running SAC and want all the frame files in one place. The frame number is incremented each time a new frame is created. You can force the frame number to start at any given value. Starting at a number other than 1 might be useful if you are generating figures for a report over several days and wish to keep them in sequential order.

The folowing paragraph was writen more thaan 20 years ago, and so far as we can see the size options in the current SGFTOPS program provides a much cleaner way to change the overall size of the plot in an SGF file. Based on sample runs using the examples given below, the output SGF files all have the same physical size, and the aspect ratio of the plots are all the same. Hence, SGF SIZE simply introduces a scaling factor for the plots. We are leaving in the paragraph and examples because they seem to work, and there may be a feature we are missing.

There are several options that can be used to control the size of the plot. A normal plot has viewspace limits of 10 by 7.5 inches. Using the default viewport limits, this results in an approximately 8 by 5 inch plot. You can force the x viewport to a fixed length or you can have the x viewport be scaled to the world coordinate limits of your data. This size information is written to the SGF. It is the responsibility of program that converts a SGF to a specific output device to generate the coding to produce a correctly sized plot. SGFTOPS performs this conversion correctly although plots larger than a single page have to post-processed correctly.

EXAMPLES

To define a directory other than where you are attached and to reset to frame number to the next value in a sequence:

```
u:   SGF DIRECTORY /MYDIR/SGFSTORE FRAME 0
```

To set the x viewport plot size to 3 inches (i.e., wallet size):

```
u:   SGF SIZE FIXED 3.0
```

For create a poster size plot to put on your wall:

```
u:   SGF SIZE FIXED 30.0
```

To set the x viewport plot size to be 1 inch long for every 10 seconds of seismic data:

```
u:   SGF SIZE SCALED 0.1
```

In this last example, a plot where the data was 60 seconds in duration would be 6 inches long whereas a plot where the data was 600 seconds in duration would be 60 inches long and would require special post processing to produce.

SEE COMMANDS

BEGINDEVICES

LATEST REVISION

May 6, 2010 (Version 101.4)

SMOOTH

SUMMARY

Applies an arithmetic smoothing algorithm to the data.

SYNTAX

```
SMOOTH {MEAN|MEDIAN},{HALFWIDTH n}
```

INPUT

MEAN: Apply a mean (average) smoothing algorithm.

MEDIAN: Apply a median point smoothing algorithm.

HALFWIDTH n: Set halfwidth of smoothing window to n. The moving window will contain n points on each side of the point being smoothed.

DEFAULT VALUES

```
SMOOTH MEAN HALFWIDTH 1
```

DESCRIPTION

This command applies an arithmetic smoothing algorithm to each data point. The type of algorithm and the size of the sliding window around each data point can be varied. The size of the window is defined by specifying its halfwidth. This forces the moving window to be centered around each data point and forces the window size to be an odd number of points, which makes the algorithms easier and less ambiguous.

HEADER CHANGES

DEPMIN, DEPMAX,DEPMEN

LATEST REVISION

April 13, 1987 (Version 10.1)

SONOGRAM

SUMMARY

Calculate a spectrogram equal to the difference between two smoothed versions of the same spectrogram.

SYNTAX

```
SONOGRAM options
```

where options are one or more of the following:

```
WINDOW v
SLICE v
ORDER n
CBAR    {ON|OFF}
YMIN v
YMAX v
FMIN v
FMAX v
BINARY|FULL
METHOD {PDS|MEM|MLM}
{COLOR|GRAY}
PRINT {pname}
```

INPUT

WINDOW v: Set the sliding data window length in seconds to v. This window length determines the size of the fft.

SLICE v: Set the data slice interval in seconds to v. A single spectrogram line is produced for each slice interval.

ORDER n: Specifies the number of points in the autocorrelation function used to compute the spectral estimate.

CBAR {ON|OFF}: Turn reference color bar on or off.

BINARY|FULL: Produce a binary image, or a full color image.

YMIN v: Specifies the minimum frequency to plot.

YMAX v: Specifies the maximum frequency to plot.

FMIN v: Specifies the smallest bandwidth over which each slice in the spectrogram will be smoothed.

FMAX v: Specifies the maximum bandwidth over which each slice in the spectrogram will be smoothed.

METHOD {PDS|MEM|MLM}: Specifies the type of spectral estimator used. MLM stands for maximum likelihood and MEM stands for maximum entropy spectral estimators, respectively. See description and references below.

{COLOR|GRAY}: Specifies a color or grayscale image.

PRINT {pname}: Prints the resulting plot to the printer named in pname, or to the default printer if pname is not used. (This makes use of the SGF capability.)

DEFAULT VALUES

```
SONOGRAM WINDOW 2 SLICE 1 METHOD MEM ORDER 100 YMIN 0 YMAX FNYQUIST FMIN
2.0
fmax 6.0 full color
```

DESCRIPTION

The sonogram command computes a spectrogram equal to the difference between two smoothed version of the same spectrogram. Depending on the choice of smoothing parameters, fmin and fmax, the resulting spectrogram can enhance small amplitude spectral features that are more difficult to observe in a conventional spectrogram. This is particularly useful when looking for features like high frequency spectral modulations in seismic signals from mine blasts (c.f., Hedlin, 1990, Wuster, 1993).

LIMITATIONS

The size of the image in the frequency direction is 512.

PROBLEMS

There is currently very little error checking of the headers to make sure that they are from the same component and are contiguous in time. This will be corrected in the future.

HEADER VARIABLES

REQUIRED: DELTA
CHANGED: NPTS, DELTA, B, E, IFTYPE, DEPMIN, DEPMAX, DEPMEN
CREATED: NXSIZE, XMINIMUM, XMAXIMUM, ,BREAK NYSIZE, YMINIMUM, YMAXIMUM

LATEST REVISION

May 26, 1995 (Version 00.31)

SORT

SUMMARY

Sorts files in memory by header fields.

SYNTAX

```
SORT COMMIT|ROLLBACK|RECALLTRACE
header {ASCEND|DESCEND} {header {ASCEND|DESCEND} ...  }
```

INPUT

COMMIT: Commits headers and waveforms in SAC memory -- removing any previous versions of headers or waveforms from RAM -- prior to writing files. COMMIT is the default.

ROLLBACK: reverts to the last committed version of the header and waveform before writing files.

RECALLTRACE:
- reverts to the last committed version of the waveform,
- reverts to the last committed version of those header variables closely linked to the waveform,
- commits those header variables which are loosely linked to the waveform. (RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

HEADER: header field upon which to sort the files.

ASCEND: Sort files on header in ascending order. This is the default.

DESCEND: Sort files on header in descending order

DESCRIPTION

Sort the files in memory in order according to the header field given. The earlier a header field appears on the command line, the higher priority that field will receive in the sort, the first field receiving the highest priority, and subsequent fields used to break ties. No more than five header fields may be entered. Each may be followed by either ASCEND or DESCEND to indicate the direction of the sort on that particular field. If neither ASCEND nor DESCEND is specified, ASCEND will be used by default. If Sort is called without specifying any header fields, it will sort on the fields specified in the previous call to SORT. If the first call to SORT is without any header fields, it will produce error 1379.

DEFAULTS

It is presumed that all sorts will be in ascending order unless DESCEND is specified on the command line.

ERROR MESSAGES

- 301: Out of memory.
- 1379: No SORT parameters given
- 1380: Too many SORT parameters:

- 1381: Not a valid SORT parameter:
- 1383: SORT failed

WARNING MESSAGES

- 1384

SEE COMMANDS

COMMIT, ROLLBACK, RECALLTRACE

LATEST REVISION

October 27, 1998 (Version 0.58)

SPECTROGRAM

SUMMARY

Calculate a spectrogram using all of the data in memory.

SYNTAX

SPECTROGRAM options

where options are one or more of the following

WINDOW: v
SLICE: v
ORDER: n
CBAR: {ON|OFF}
Choose from: {SQRT|NLOG|LOG10|NOSCALING{
YMIN: v
YMAX: v
METHOD: {PDS|MEM|MLM}
Choose from: {COLOR|GRAY}
PRINT: {pname}

INPUT

WINDOW v: Set the sliding data window length in seconds to v. This window length determines the size of the fft.

SLICE v: Set the data slice interval in seconds to v. A single spectrogram line is produced for each slice interval.

ORDER n: Specifies the number of points in the autocorrelation function used to compute the spectral estimate.

CBAR {ON|OFF}: Turn reference color bar on or off.

{SQRT|NLOG|LOG10|NOSCALING}: Specify natural log, log base 10, or square root scaling of amplitudes.

YMIN v: Specifies the minimum frequency to plot.

YMAX v: Specifies the maximum frequency to plot.

METHOD {PDS|MEM|MLM}: Specifies the type of spectral estimator used. MLM stands for maximum likelihood and MEM stands for maximum entropy spectral estimators, respectively. See description and references below.

{COLOR|GRAY}: Specifies a color or grayscale image.

PRINT {pname}: Prints the resulting plot to the printer named in pname, or to the default printer if pname is not used. (This makes use of the SGF capability.)

DEFAULT VALUES

```
SPECTROGRAM WINDOW 2 SLICE 1 METHOD MEM ORDER 100 NOSCALING YMIN 0 YMAX
FNYQUIST COLOR
```

DESCRIPTION

A spectrogram is computed by calculating power spectra of consecutive, possibly overlapping time windows of data and plotting the spectra side by side along a time axis. The spectra are calculated from a truncated autocorrelation function using either the maximum likelihood method (MLM), maximum entropy method (MEM), or Power Density Spectral method (PDS). In general, the high resolution, maximum likelihood and maximum entropy methods are preferred because they improve resolution and because they do not produce artifacts (sidelobes) in the spectra due leakage of energy between different frequencies. Descriptions of these techniques can be found in Kanasewich (1981) and Lacoss (1971) and the references therein. The length of the truncated autocorrelation function is determined by the order parameter. To maintain consistency with the spe subroutines we have set the defaults order to 200 for the power density spectra (pds) and 100 for the maximum entropy and maximum likelihood spectral estimates. In sac the length of each data window is determined by the window parameter. The spacing between spectra along the spectrograms time axis is determined by the slice parameter. The difference between these two parameters determines the amount of overlap between adjacent time window as indicated in the diagram below.:

```
Time --->
0     1     2     3     4     5     6     7     8     9    10    11
|.....|.....|.....|.....|.....|.....|.....|.....|.....|.....|.....|
|__^__| window 1, First time will be at the center of this window.
      |__^__| window 2
            |__^__| window 3

|.....| Slice: Difference between the start times of adjacent windows.
```

The start and end points on the spectrograms time axis depend on the length of the time series being analysed and the window and slice parameters. The spectrogram's start time is one-half a window later than the time series start time because it is defined as the center of time of the first window. SAC doesn't pad the start of the data with zeros.

Kanasewich, E. R., "Time Sequence Analysis in Geophysics", The University of Alberta Press, Edmonton, 1981.

Lacoss, R. T., Data Adaptive Spectral Analysis Methods", Geophysics, Vol. 36, 661-675, 1971.

LIMITATIONS

The size of the image in the frequency direction is 512.

PROBLEMS

There is currently very little error checking of the headers to make sure that they are from the same component and are contiguous in time. This will be corrected in the future.

HEADER VARIABLES

REQUIRED: DELTA

CHANGED: NPTS, DELTA, B, E, IFTYPE, DEPMIN, DEPMAX, DEPMEN

CREATED: NXSIZE, XMINIMUM, XMAXIMUM, ,BREAK NYSIZE, YMINIMUM, YMAXIMUM

LATEST REVISION

May 26, 1995 (Version 00.31)

SQR

SUMMARY

Squares each data point.

SYNTAX

SQR

ERROR MESSAGES

- 1301: No data files read in.
- 1307: Illegal operation on spectral file

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

LATEST REVISION

January 8, 1983 (Version 8.0)

SQRT

SUMMARY

Takes the square root of each data point.

SYNTAX

SQRT

ERROR MESSAGES

- 1301: No data files read in.
- 1307: Illegal operation on spectral file
- 1702: Non-positive values found in file

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

LATEST REVISION

January 8, 1983 (Version 8.0)

STRETCH

SUMMARY

Stretches (upsamples) data, including an optional interpolating FIR filter.

SYNTAX

```
STRETCH {n},{FILTER {ON|OFF}}
```

INPUT

n: Set upsampling factor. Must be in the range 2 to 7.
FILTER {ON}: Turn interpolating filter option on.
FILTER OFF: Turn interpolating filter option off.

DEFAULT VALUES

```
STRETCH 2 FILTER ON
```

DESCRIPTION

By using the interpolating filter option, this command can be used to create a file with a smaller sampling interval (more data points) but which looks similar to the original. Care should be taken when using this command, because the filter does effect the frequency content. When this filter option is off, the appropriate number of zeros are simply inserted between each of the original data points.

HEADER CHANGES

NPTS, DELTA, E, DEPMIN, DEPMAX, DEPMEN

LATEST REVISION

May 15, 1987 (Version 10.2)

SUB

SUMMARY

Subtracts a constant from each data point.

SYNTAX

```
SUB {v1 {v2 ...  vn} }
```

INPUT

v1: Constant to subtract from first file.
v2: Constant to subtract from second file.
nv: Constant to subtract from nth file.

DEFAULT VALUES

```
SUB 0
```

DESCRIPTION

This command will subtract a constant from each element of each data file currently in memory. The constant may be the same or different for each data file. If there are more data files in memory than constants, then the last constant entered is used for the remainder of the data files in memory.

EXAMPLES

To subtract 5.1 from each element of F1 and 6.2 from each element of F2 and F3:

```
u:  READ F1 F2 F3
u:  SUB 5.1 6.2
```

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES

- 1301: No data files read in.
- 1307: Illegal operation on spectral file

LATEST REVISION

January 8, 1983 (Version 8.0)

SUBF

SUMMARY

Subtracts a set of data files from data in memory.

SYNTAX

```
SUBF {NEWHDR ON|OFF} filelist
```

INPUT

NEWHDR ON|OFF: By default, the resultant file will take its header field from the original file in memory. Turning NEWHDR ON, causes the header fields to be taken from the new file in the filelist.

filelist: A list of SAC binary data files. This list may contain simple filenames, full or relative pathnames, and wildcard characters. See the READ command for a complete description.

DESCRIPTION

This command can be used to subtract a single file from a set of files or to subtract one set of files from another set. An example of each case is presented below. The files must be evenly spaced and should have the same sampling interval and number of points. This last two restrictions can be eliminated using the BINOPERR command. If there are more data files in memory than in the filelist, then the last file in the filelist is used for the remainder of the data files in memory.

EXAMPLES

To subtract one file from three other files:

```
u:  READ FILE1 FILE2 FILE3
u:  SUBF FILE4
```

To subtract two files from two other files:

```
u:  READ FILE1 FILE2
u:  SUBF FILE3 FILE4
```

HEADER CHANGES

If NEWHDR is OFF (the default) the headers in memory are unchanged). If NEWHDR is ON, the headers are replaced with the headers from the files in the filelist.

DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES

- 1301: No data files read in.
- 1803: No binary data files read in.
- 1307: Illegal operation on spectral file

- 1306: Illegal operation on unevenly spaced file
- 1801: Header field mismatch:
  - either the sampling interval or the number of points are not equal. - can be controlled using the BINOPERR command.

WARNING MESSAGES

- 1802: Time overlap:
  - the file subtraction is still performed.

SEE COMMANDS

READ, BINOPERR

LATEST REVISION

May 26, 1999 (Version 0.58)

SYMBOL

SUMMARY

Controls the symbol plotting attributes.

SYNTAX

```
SYMBOL {ON|OFF|n} {SIZE v},{SPACING v},
{INCREMENT {ON|OFF}},{LIST STANDARD|nlist}
```

INPUT

ON: Turn symbol plotting on. Don't change symbol number.

OFF: Turn symbol plotting off.

n: Turn symbol plotting on. Change symbol number to n. There are 16 different symbols. A symbol number of 0 is the same as turning symbol plotting off.

SIZE v: Set symbol size to v. A value of 0.01 sets the size to 1 percent of the full plot size.

SPACING v: Set symbol spacing to v. This is the minimum spacing between drawn symbols. Use 0 if you want a symbol at every data point. Use 0.2 to 0.4 for annotating lines.

INCREMENT {ON}: Increment symbol number after each data file. The symbol number is the next one in the symbol list.

INCREMENT OFF: Do not increment symbol number.

LIST nlist: Change the content of the symbol list. Enter list of symbol numbers. Sets symbol number to first entry in list and turns symbol plotting on.

LIST STANDARD: Change to the standard symbol list. Sets symbol number to first entry in list and turns symbol plotting on.

DEFAULT VALUES

```
SYMBOL OFF SIZE 0.01 SPACING 0.  INCREMENT OFF LIST STANDARD
```

DESCRIPTION

The figure that follows shows each of the sixteen symbols. Symbol 1 cannot be scaled in size. It is a replacement for the point symbol which does not show up well on many devices (e.g. Versatec, pen plotter). This figure also shows examples of different symbol size and spacing values. These symbol attributes are independent of the line drawing attributes defined by the LINE command. With line drawing on, they can be used to annotate different lines on the same plot. By turning the line drawing off, they can be used to create scatter plots.

If you are plotting several data files on the same plot, you may want each to be plotted with a different symbol. This is done using the INCREMENT option. When this option is on, the symbol is incremented from a list of symbols each time a data file is plotted. The default symbol list contains symbols 2 through 16. You may change the order or content of this list using the LIST option. This is useful if you are doing a series of overlay plots (see PLOT2) and want the same symbols used in the same order on each plot. A symbol number of 0 is the same as turning symbol plotting off. This is useful in the LIST option and the LINE command to display some data with lines and some with symbols on the same plot. See the example below.

EXAMPLES

To create a scatter plot, turn the line drawing off, choose an appropriate symbol, and plot:

```
u:  LINE OFF
u:  SYMBOL 5
u:  PLOT
```

To annotate four solid lines on a PLOT2 plot using symbols 7, 4, 6, and 8, and a spacing of 0.3:

```
u:  LINE SOLID
u:  SYM SPACING .3 INCREMENT LIST 7 4 6 8
u:  R FILE1 FILE2 FILE3 FILE4
u:  PLOT2
```

To plot three files on the same plot using PLOT2 with the first file plotted using a solid line and no symbol, the second with no line and a triangle symbol, and the third with no line and a cross symbol:

```
u:  READ FILE1 FILE2 FILE3
u:  LINE LIST 1 0 0 INCREMENT
u:  SYMBOL LIST 0 3 7 INCREMENT
u:  PLOT2
```

SEE COMMANDS

LINE

LATEST REVISION

October 11, 1984 (Version 9.1) Summary of SYMBOL Command Attributes

SYNCHRONIZE

SUMMARY

Synchronizes the reference times of all files in memory.

SYNTAX

```
SYNCHRONIZE {ROUND {ON|OFF}} {BEGIN {ON|OFF}}
```

INPUT

ROUND {ON}: Turn begin time rounding on. When this opotions is on, the begin times for each file are rounded to the nearest multiple of the sampling interval.

ROUND OFF: Turn begin time rounding off.

BEGIN {ON}: Sets begin time of each file to zero.

BEGIN OFF: Maintains the GMT values of the reference times.

DEFAULT VALUES

```
SYNCHRONIZE ROUND OFF BEGIN OFF
```

DESCRIPTION

This command synchronizes the references times for all files in memory. It determines the latest starting time of all files by examining their reference times and beginning offset times. This latest starting time then becomes the reference time for ALL of the files in memory. New values for all of the offset times (B, E, A, Tn, etc.) for each of the files are then calculated. This command is useful when a set of files have different reference times and you want to use the CUT or XLIM command to analyze or plot portions of them. Once they have been synchronized to the same reference time, the cuts will then refer to the exact same GMT time window. If the BEGIN option is used, GMT values of reference times are not preserved. The BEGIN option sets the kztime of all files the same, it sets the kzdate of all files the same, and it sets the begin time of all files to zero. Other reference points retain their relation to the begin time of the file.

EXAMPLES

Assume you read two files into memory with different reference times:

```
u:  READ FILE1 FILE2
u:  LISTHDR B KZTIME KZDATE
s:
s:  FILE: FILE1
s:  -----------
s:         B = 0.
s:    KZTIME = 10:38:14.000
s:    KZDATE = MAR 29 (088), 1981
s:
s:  FILE: FILE2
s:  -----------
s:         B = 10.00
```

```
s:    KZTIME = 10:40:10.000
s:    KZDATE = MAR 29 (088), 1981
```

The files have the same reference date but different reference times and different beginning offsets. Now if you execute the SYNCHRONIZE command followed by another LISTHDR you would find:

```
u:  SYNCHRONIZE
u:  LISTHDR
s:
s:  FILE: FILE1
s:  -----------
s:         B = -126.00
s:    KZTIME = 10:40:20.000
s:    KZDATE = MAR 29 (088), 1981
s:
s:  FILE: FILE2
s:  -----------
s:         B = 0.
s:    KZTIME = 10:40:20.000
s:    KZDATE = MAR 29 (088), 1981
```

Now the files in memory have the same reference time which is the beginning of the later file. If there had been any defined time markers in these headers, their values would be adjusted so that they point to the same time as before.

HEADER CHANGES

NZYEAR, NZJDAY, NZHOUR, NZMIN, NZSEC, NZMSEC, B, E, A, O, Tn.

LATEST REVISION

May 15, 1987 (Version 10.2)

SYSTEMCOMMAND

SUMMARY

Executes system commands from SAC.

SYNTAX

```
[S]YSTEM[C]OMMAND command {options}
```

INPUT

command: The name of the system command.
options: Any options needed by that command.

DESCRIPTION

This command allows you to execute system command while running SAC.

EXAMPLES

To get a list of files in the current UNIX directory:

```
u:  SYSTEMCOMMAND LS
s:  ... produces a list of files
```

LATEST REVISION

May 15, 1987 (Version 10.2)

TAPER

SUMMARY

Applies a symmetric taper to each end of data.

SYNTAX

```
TAPER {TYPE HANNING|HAMMING|COSINE},{WIDTH v}
```

INPUT

TYPE HANNING: Apply a Hanning taper.
TYPE HAMMING: Apply a Hamming taper.
TYPE COSINE: Apply a cosine taper.
WIDTH v: Set the taper width on each end to v. This is a value between 0.0 and 0.5.

DEFAULT VALUES

```
TAPER TYPE HANNING WIDTH 0.05
```

DESCRIPTION

A taper is a monotonically varying function between zero and one. It is applied in a symmetric manner to the data such that the signal is zero for the first and last data points and increases smoothly to its original value at an interior point relative to each end. The general form for the taper is:

```
DATA(J)=DATA(J)*(F0-F1*COS(OMEGA*(J-1))
```

This equation would be applied to the left hand side of each signal. A symmetric one is applied to the right hand side. The following table defines the various parameters used in the different tapers. In this table N is the length of the taper on each end.

TYPE	OMEGA	F0	F1
HANNING	PI/N	0.50	0.50
HAMMING	PI/N	0.54	0.46
COSINE	PI/(2*N)	1.00	1.00

ERROR MESSAGES

- 1301: No data files read in.
- 1306: Illegal operation on unevenly spaced file

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

LATEST REVISION

January 15, 1985 (Version 9.10) A Comparison of the Various Types of Tapers

TICKS

SUMMARY

Controls the location of tick marks on plots.

SYNTAX

```
TICKS   ON|OFF|ONLY   sides
```

where sides is the keyword:

```
ALL
```

or one or more of the following:

```
TOP,BOTTOM,RIGHT,LEFT
```

INPUT

ON: Turn ticks on for listed sides; others unchanged.
OFF: Turn ticks off for listed sides; others unchanged.
ONLY: Turn ticks on only for listed sides; others off.
ALL: All four ticks.
TOP: X axis above viewport.
BOTTOM: X axis below viewport.
RIGHT: Y axis to right of viewport.
LEFT: Y axis to left of viewport.

DEFAULT VALUES

```
TICKS ON ALL
```

DESCRIPTION

Tick marks can be drawn on one or more of the four sides of a plot. They are drawn at the current division spacing controlled by the XDIV command. Tick marks are automatically drawn on sides where annotated axes have been requested using the AXES command.

EXAMPLES

To turn on the top tick marks and leave the others unchanged:

```
u:  TICKS ON TOP
```

To turn off all tick marks (at least where there are no annotated axes):

```
u:  TICKS OFF ALL
```

To turn tick marks on for the bottom side and off for the rest:

```
u:  TICKS ONLY BOTTOM
```

SEE COMMANDS

XDIV, AXES

LATEST REVISION

January 8, 1983 (Version 8.0)

TITLE

SUMMARY

Defines the plot title and attributes.

SYNTAX

`TITLE {ON|OFF|text},{LOCATION location},{SIZE size}` where location is one of the following:

`TOP|BOTTOM|RIGHT|LEFT`

and where size is one of the following:

`TINY|SMALL|MEDIUM|LARGE`

INPUT

ON: Turn title option on. Don't change title text.
OFF: Turn title option off.
text: Turn title option on. Change text of title. If text contains embedded blanks, it must be enclosed in single quotes.
LOCATION location: Change location of title.
TOP: Top of the plot window.
BOTTOM: Bottom of the plot window.
RIGHT: To the right of the plot window.
LEFT: To the left of the plot window.
SIZE size: Change title text size.
TINY: Tiny text size has 132 characters per line.
SMALL: Small text size has 100 characters per line.
MEDIUM: Medium text size has 80 characters per line.
LARGE: Large text size has 50 characters per line.

DEFAULT VALUES

`TITLE OFF LOCATION TOP SIZE SMALL`

DESCRIPTION

If this option is on, a title is placed on each plot. The size and location of the title can be changed as well as the text of the title itself. The text quality and font used can be set using the GTEXT command.

SEE COMMANDS

GTEXT

LATEST REVISION

January 8, 1983 (Version 8.0)

TRACE

SUMMARY

Controls the tracing of blackboard and header variables.

SYNTAX

```
TRACE [ON|OFF] name [name ...]
```

INPUT

ON: Turn tracing on for variables that follow.

OFF: Turn tracing off for variables that follow.

name: The name of the blackboard or header variable to trace. If

this is a header variable it is of the form: filename,hdrname ,BREAK where filename is the name (or number) of the SAC data file and hdrname is the name of a SAC header variable.

DEFAULT VALUES

```
TRACE ON
```

DESCRIPTION

This command can be used to trace or track the values of SAC blackboard or header variables while SAC is executing. It is useful primarily for debugging long or complicated macros. When the tracing for a variable is turned on, its current value is printed. While the tracing is on, its value is checked after the execution of each command. Each time its value changes a new output line is printed. When tracing is turned off, its current value is also printed.

EXAMPLES

To turn tracing on for the blackboard variable called TEMP1 and for the header variable called T0 belonging to the file called MYFILE:

```
u:  TRACE ON TEMP1 MYFILE,T0
s:  TRACE  (on) TEMP1 = 1.45623
s:  TRACE  (on) MYFILE,T0 = UNDEFINED
```

As you execute commands, either typed at the terminal or executed from a macro, SAC will check the values of the variables versus the saved value and print a message whenever either one of them changes. Assume that some calculations are performed that caused TEMP1 to change and T0 to become defined. SAC would print the messages:

```
s:  TRACE (mod) TEMP1 = 2.34293
s:  TRACE (mod) MYFILE,T0 = 10.3451
```

Later in the processing TEMP1 may change again:

```
s:  TRACE (mod) TEMP1 = 1.93242
```

When the tracing is turned off, SAC will print the current value one last time:

```
u:  TRACE OFF TEMP1 MYFILE,T0
s:  TRACE (off) TEMP1 = 1.93242
s:  TRACE (off) MYFILE,T0 = 10.3451
```

LATEST REVISION

January 27, 1989 (Version 10.4B)

TRANSCRIPT

SUMMARY

Controls output to the transcript files.

SYNTAX

```
TRANSCRIPT options
```

where options are one or more of the following:

```
OPEN|CREATE|CLOSE|CHANGE|WRITE|HISTORY
FILE filename
CONTENTS ALL|list
MESSAGE text
```

where list is one or more of the following:

```
ERRORS
WARNINGS
OUTPUT
COMMANDS
MACROS
PROCESSED
```

INPUT

OPEN: Open and append transcript to the bottom of an existing file.

CREATE: Create a new transcript file.

CLOSE: Close an open transcript file. (NEW version 101.2)

CHANGE: Change the contents of an open transcript file.

WRITE: Write message to transcript file without changing its status or contents.

HISTORY FILE filename: Save/restore command-line history to a file.

FILE filename: Define the name of a transcript.

MESSAGE text: Write message contained in text to transcript file. This message can be used to identify the processing being done or to identify different events as they are being processed. This message is NOT retained between executions of this command.

CONTENTS ALL: Define the contents of the transcript file to be all input/output types.

CONTENTS list: Define the contents of the transcript file. This is a list of the types of input and output to include in the file.

ERRORS: Error messages generated during the execution of a command.

WARNINGS: Warning messages generated during the execution of a command.

OUTPUT: Output messages generated during the execution of a command.

COMMANDS: Raw commands as they were typed at the terminal.

MACROS: Raw commands as they appears in a macro file.

PROCESSED: Processed commands originating from the terminal or a macro file. A processed command is one where all macro arguments, blackboard variables, header variables, and inline functions have been processed (evaluated) and substituted into the raw command.

DEFAULT VALUES

```
TRANSCRIPT OPEN FILE TRANSCRIPT CONTENTS ALL
```

DESCRIPTION

A transcript file can be used to record the results of executing SAC. It can be a complete or partial transcript. It can contain the results from one or more executions. You can have up to five transcripts active at any given time, each keeping track of different aspects of the execution. One use as illustrated below is to record the commands typed at the terminal and to later use this as a macro file.

EXAMPLES

To create a new transcript file called MYTRAN containing everything except the processed commands:

```
u:  TRANSCRIPT CREATE FILE MYTRAN CONTENTS ERRORS WARNINGS OUTPUT
```

COMMANDS MACROS

If later during this session you did not want the macro commands to be sent to this file you would use the CHANGE option:

```
u:  TRANSCRIPT CHANGE FILE MYTRAN CONTENTS ERRORS WARNINGS OUTPUT
```

COMMANDS

To define a transcript file called MYRECORD which records the commands as they are typed at the terminal:

```
u:  TRANSCRIPT CREATE FILE MYRECORD CONTENTS COMMANDS
```

Later this file, perhaps after some editing, could be used as a macro to automatically execute the same set of commands. In the final example assume you needed to process a number of events overnight. You could set up transcript files for each of these events (with different names) that recorded the results of the processing. In addition you could store any error messages from the processing of all of these events in a single transcript file:

```
u:  TRANSCRIPT OPEN FILE ERRORTRAN CONTENTS ERRORS
u:  TRANSCRIPT WRITE MESSAGE 'Processing event 1'
```

These commands would be placed in the macro that processes each event. It is assumed that the name of the event is passed into the macro as the first argument. By using the open option, the message and any errors would be appended to the end of the file. By examining this error transcript the next day, you could quickly see whether any errors occurred during processing and for which events these errors occurred.

To save a command-line transcript that records SAC commands from curent and future runs, use:

```
u:  TRANSCRIPT HISTORY FILE .sachist
```

This creates and writes to a transcript file, "./.sachist", in the current directory. Any commands stored there are loaded into your command history, and you can scroll back through them. If this command is in your startup initialization macro, there will be a separate command-line history for each directory in which you run SAC. In a new run of SAC, the up/down or previous/next keys scroll through the complete history. You can edit a previously-typed command and enter it again. If you do not enter this command within SAC or in an initialization macro, the command-line history will be automatically logged to ~/.sac_history. See README_utils in subdirectory sac/utils for further discussion.

LATEST REVISION

September 2008 (version 101.2)

TRANSFER

SUMMARY

Performs deconvolution to remove an instrument response and convolution to apply another instrument response.

SYNTAX

```
[TRANS]FER {FROM type {options}} , {TO type {options}} ,
      {FREQlimits f1 f2 f3 f4} , {PREWhitening ON|OFF|n}
```

INPUT

FROM type: Remove the instrument type by deconvolution using spectral division.

TO type: Insert the instrument type by convolution using spectral multiplication.
The allowed instrument types and their options for both TO and FROM are listed in a table below.

DEFAULT VALUES: TRANS FROM NONE TO NONE

DESCRIPTION

The default input and output "instrument" in TRANSFER is displacement, which in SAC is designated as NONE. Hence, if a FROM type or a TO type is not specified, SAC assumes it to be NONE. If the output instrument is NONE, IDEP in the SAC header is set to DISPLACEMENT (NM). For the seismometers listed in the table below, the output units for TO NONE will be in nm, SAC's choice for displacement. If TRANSFER has TO VEL or TO ACC, the header variable IDEP is changed accordingly for all waveforms in memory.

If the TO type is specified as anything other than NONE, VEL, or ACC, the waveforms in memory are transformed to that instrument type. If the FROM instrument type is NONE, then no instrument is removed, and the original trace is presumed to be a displacement. This is useful for adding instrument responses to synthetic seismograms.

Care must be taken when calling TRANSFER a second time within a single SAC session, because in the second call TRANSFER will use the same arguments for FROM, TO, FREQ, etc. as in the first call unless an alternative argument is explicitly provided.

Many of the instruments have options that further specify the response. The most common of these options is the instrument subtype. A few instruments require that certain numerical parameters be specified and do not use the subtype option. For a list of instruments and a list of the instruments that use subtypes or other parameters, see the table below.

When TRANSFER was introduced more than 20 years ago, the data acquisition systems were much simpler. The seismometers in the list at the end of this message include the most popular ones used previously. Now the majority of data analysis is done on data sets commonly exchanged as SEED volumes, which can be downloaded from the IRIS Data Management Center over the Internet. These data can be read from SEED volumes using the program RDSEED, which is available for all computer operating systems that can run SAC. For the SEED manual, go to <http://www.fdsn.org/seed_manual/SEEDManual_V2.4.pdf>.

The EVALRESP program calculates the complete system response from response (RESP) files produced by RDSEED. EVALRESP routines have been embedded within TRANSFER as an "instrument" type, allowing a user to perform (de)convolution using the complete system response represented in RESP files. Recently (October 2013) v3.3.4 has been introduced, which has options to deal with problems with some asymmetric FIR coefficients. The code used in SAC is based on v3.33, and does

not recognize all the options available in the full program. If one wants an option only availalble in the full program, one can run EVALRESP with a FAP output option and use the FAP option (see below) to correct a SAC data file. For the source code for programs RDSEED and EVALRESP, go to URL <http://ds.iris.edu/ds/nodes/dmc/software/downloads/>.

Most of the implementation is done using double-precision (64-bit) arithmetic even though the SAC data is single-precision.

The SAC sign convention for Laplace/Fourier transforms is the same as that used in RDSEED and EVALRESP: phase for a causal response decreases with increasing frequency. For displacements, the SAC convention is nm, while RDSEED RESP files use meters. The EVALRESP option in TRANSFER corrects the output to the SAC convention. For other options (FAP, PZ) it may be ncessary to change the units. (See examples below.)

FREQLIMITS f1 f2 f3 f4 : All seismometers have zero response at zero frequency. When deconvolving and not convolving with another response (e.g. "TO NONE"), it is therefore necessary to modify the response at very low frequencies. At high frequencies, the signal-to-noise ratio is often low, so it may be desirable to dampen the response. FREQLIMITS serves this purpose within SAC. FREQLIMITS has both a low-pass and a high-pass taper. It is necessary that f1 < f2 < f3 < f4. The taper is unity between f2 and f3 and zero below f1 and above f4. Frequencies f1 and f2 specify the high-pass filter at low frequencies, while frequencies f3 and f4 specify the low-pass filter at high frequencies. Both f3 and f4 should be less than the Nyquist frequency: 0.5/DELTA. The filters applied between f1 and f2 and between f3 and f4 are quarter cycles of a cosine wave. To avoid ringing in the output time series, a suggestr rule-of-thumb is f1 ,= f2/2 and f4 >= 2*f3.

If you want to do a low-pass filter but have no filtering at low frequencies, one way is to set f1=-2 and f2=-1. If you want to do a high-pass filter but have no filtering at the high frequencies, for a Nyquist frequency of 0.5, set f3=10. and f4=20.

Note that because this filter has zero phase, it is not causal. As a result, if npts is not a power of 2, the output amplitude will not be zero outside the interval (f1,f4). If it is important to have the number of points an exact power of 2, the help file for CUT explains how to modify your file within SAC.

Note that the default has no FREQLIMITS. It is strongly advised that one includes FREQLIMITS if one is doing a deconvolution.

PREWHITENING: Prewhitening can be used to flatten the spectrum of the input time series before transforming in the frequency domain. This should reduce the dynamic range of the spectral values, and improve the accuracy of the overall operation at high frequencies for seismic data. The default for prewhitening is off. See command WHITEN for further information.

PREWHITENING ON : Turns on prewhitening in the time domain before spectral operations, and compensating dewhitening in the time domain after spectral operations.

PREWHITENING OFF : Turns off prewhitening.

PREWHITENING n : Turns on prewhitening and change the prewhitening order to n. If the user turns it on without specifying the order, it will default to n=6, unless the order has been changed in the WHITEN command.

EVALRESP OPTION

This option enables the application of transfer functions extracted from SEED data volumes using the EVALRESP code (Version 3.3.3). To use this option, one needs RESP files, commonly these are produced from SEED volumes by the RDSEED program. The RESP files must be in the current directory or must be specified by full path and name.

There is no formal documentation for the RESP files themselves, but since they refer directly to the SEED format you can read the SEED manual to learn more about the values.

To identify the correct RESP file and to extract the proper transfer function from that file, EVALRESP uses information from the SAC headers. The fields are station (KSTNM), channel (KCMPNM), date and time (KZDATE & KZTIME), network (KNETWK), and location ID. Location ID is referred to as LOCID; it commonly distinguishes between multiple instruments with the same network, station and channel names, operating at the same time. Data received from IRIS in SAC format (or converted to SAC with RDSEED) will have KHOLE set to a valid LOCID if one is necessary. If the user is informed of real LOCIDs in the EVALRESP file, the user can set KHOLE with CH (HELP CH for details). SAC will use KHOLE as LOCID if it is a two character alpha-numeric string (padded with spaces or not).

It is possible to override the header values by specifying additional options to EVALRESP. The possible options are:

```
STATION, CHANNEL, NETWORK, DATE, TIME, LOCID, FNAME
```

and each option must be followed by an appropriate value. If DATE is not set in the header and is not specified as an option, then the current date is used in the search. If TIME is not set in the seismogram header and is not specified as an option, then the current system time is used in the search. If network is not specified, then the search for a transfer function defaults to use any network. If LOCID is not set at the command line or in KHOLE, then the search for the transfer function defaults to use any LOCID. To force TRANSFER to use a specific SEED response file use the FNAME option followed by the filename.

If the FNAME option is not specified EVALRESP will attempt to identify the correct file in the current working directory using the general form:

```
RESP.<NET>.<STA>.<LOCID>.<CHAN>
```

for example: "RESP.IU.ANMO..BHZ"

The embedded version of EVALRESP is configured to always produce a displacement response in SI units (i.e. displacement in meters), which SAC scales by a factor of 1.0e9 to conform to the SAC convention of units of displacement (nanometers).

EVALRESP EXAMPLES

To remove the instrument response from the seismogram in memory (assuming a response file exists):

```
SAC> r 2006.253.14.30.24.0000.TA.N11A..LHZ.Q.SAC
SAC> RTR
SAC> TAPER
SAC> TRANS FROM EVALRESP TO NONE freq 0.004 0.007 0.2 0.4
```

To remove the instrument response from the same waveform but using a response contained in file /tmp/Responses/RESP.TA.N11A..LHZ:

```
SAC> SETBB RESP "/tmp/Responses/RESP.TA.N11A..LHZ"
SAC> r 2006.253.14.30.24.0000.TA.N11A..LHZ.Q.SAC
SAC> RTR
SAC> TAPER
SAC> TRANSFER FROM EVALRESP FNAME %resp TO NONE FREQLIM 0.004 0.007 0.2 0.4
```

To remove the instrument response from 16.42.05.5120.TS.PAS.BHZ.SAC and apply the response from station COL, channel BHZ for the same time period:

```
SAC> R 16.42.05.5120.TS.PAS.BHZ.SAC
SAC> RTR
SAC> TAPER
SAC> TRANS FROM EVALRESP TO EVALRESP STATION COL
```

To display the instrument response in units of displacement for station COL, channel BHZ, network IU, for the date 1992/02 and time 16:42:05:

```
SAC> FG IMPULSE NPTS 16384 DELTA .05 BEGIN 0.
SAC> TRANS TO EVALRESP STATION COL CHANNEL BHZ NETWORK IU DATE 1992/2 TIME 16:42:05
SAC> FFT
SAC> PSP AM
```

COMMENTS: rtr removes any trend and offset. Because the FFT called by TRANSFER pads with zeroes to a power of 2 number of points, TAPER eliminates any large jumps at the ends of the time series. FREQLIMITS is necessary for deconvolutions TO NONE because the instrument has zero response at zero frequency.

POLEZERO OPTION

POLEZERO is an instrument type that can be used to put in or take out the (analog) seismometer response. A good reference is Appendix C in the SEED manual.

A polezero file as written may be for displacement, velocity, or acceleration, and the units of the output should be known in advance. If the polezero file was written by program RDSEED 5.0 or later, this information is included in the file (see example below).

The transfer function, H(s), is the Laplace transform of the linear system impulse response of the seismometer. The Laplace variable s = 2*pi*i*f, where f is the frequency in Hz.

The response H(w) is the ratio of the product of the difference between s and each of the np poles and nz zeros:

$$H(s) = \frac{(s-z_1)*(s-z_2)*\ldots*(s-z_{nz})}{(s-p_1)*(s-p_2)*\ldots*(s-p_{np})}$$

The options in the file (poles, zeros, constant, and comment lines) are keyword driven and numbers are in free format. CONSTANT is a scaling factor. (See the SEED manual for how it is defined for polezero files produced by RDSEED.) The default for CONSTANT is 1.0 if one omits this line. one specifies the number of poles by putting a line in the file with the keyword "POLES" followed by an integer number (np in the above example). The next np lines in the file, each containing two floating-point numbers, are the poles for this instrument. One specifies the zeros with a line starting with "ZEROS" followed by an integer specifying the number of zeros (nz). Because a typical polezero file has one or more zeros that are (0.0,0.0), SAC does not require one to write out a line for a zero equal to (0.0,0.0). One may specify up to 30 poles and 30 zeros.

The original SAC polezero files only contained poles, zeroes, and a constant. Recently, it has been determined that supplying formatted comments as a header in the polezero file helps greatly for users to organize and understand the origins of the coefficients presented. For this reason, SAC now supports the 'annotated' polezero file, which can be readily produced by IRIS DMC's RDSEED program (v5.2, October 2011) as well as IRIS DMC's SAC PZ web service <http://service.iris.edu/irisws/sacpz/>. (RDSEED v5.0 had annotation lines with slightly different formats for some entries. SAC v 101.5 can handle files written by either v5.0 or 5.2 of RDSEED. Polezero files produced by a request to sacpz are in the RDSEED 5.2 format.) If a waveform in memory has header values KSTNM, KCMPNM, KZDATE, and KZTIME that match entries for a polezero file in the call to TRANSFER, the command will work for that waveform. Depending on the request format, a fpolezero file returned by sacpz may include multile polezero files covering more than one time epoch as well as more than one station and/or channel. A call to transfer using such a file will work satisfactorily for all waveforms in memory with anotation values that match the header values.

Here is a polezero file written by program RDSEED (v5.2). Earlier versions of RDSEED have a slightly different format for the comments (v5.0) or may have no comments (earlier versions):

```
* ********************************
 NETWORK   (KNETWK): II
 STATION    (KSTNM): PFO
 LOCATION   (KHOLE): 00
 CHANNEL   (KCMPNM): BHZ
 CREATED           : 2011-08-11T00:24:07
 START             : 2010-07-30T18:50:00
 END               : 2599-12-31T23:59:59
 DESCRIPTION       : Pinon Flat, California, USA
 LATITUDE          : 33.610700
 LONGITUDE         : -116.455500
 ELEVATION         : 1280.0
 DEPTH             : 5.3
 DIP               : 0.0
 AZIMUTH           : 0.0
 SAMPLE RATE       : 20.0
 INPUT UNIT        : M
 OUTPUT UNIT       : COUNTS
 INSTTYPE          : Streckeisen STS-1 Seismometer with Metrozet E300
 INSTGAIN          : 3.314400e+03 (M/S)
 COMMENT           : S/N #119005
 SENSITIVITY       : 5.247780e+09 (M/S)
 A0                : 7.273290e+01
 ********************************
ZEROS       6
-7.853982e+01       +0.000000e+00
-1.525042e-01       +0.000000e+00
-1.525042e-01       +0.000000e+00
POLES       6
-1.207063e-02       +1.224561e-02
-1.207063e-02       -1.224561e-02
-1.522510e-01       +9.643684e-03
-1.522510e-01       -9.643684e-03
-4.832398e+01       +5.817080e+01
-4.832398e+01       -5.817080e+01
CONSTANT    3.816863e+11
```

For this transfer function, there are six poles, for which the complex values are listed on the five lines following the line POLES 6. There are six zeros, but because only three are listed, the three not listed have the value of (0.0,0.0).

Note: On some platforms, RDSEED v5.1 produces incorrect END times -- often earlier than the START time, so TRANSFER will not work. Manually changing END to a time later than CREATED will allow TRANSFER to process that waveform.

To use this option, one specifies the type to be POLEZERO and the [S]ubtype is the name of the file. This may be a file in the current directory or in some other directory if one specifies the absolute or relative pathname.

PZ EXAMPLES

The PZ file SAC_PZs_XC_OR075_LHZ is the correct one to remove the instrument response from waveform OR075_LHZ.SAC. (Names are shortened from those rturned by RDSEED.):

```
SAC> SETBB pzfile "SAC_PZs_XC_OR075_LHZ"
SAC> READ OR075_LHZ.SAC
SAC> RTR
SAC> TAPER
SAC> TRANS FROM POLEZERO S %pzfile TO NONE FREQ 0.008 0.016 0.2 0.4
SAC> MUL 1.0e9
SAC> w OR075.z
```

COMMENTS: If the polezero file was produced by the RDSEED program from a SEED volume, the poles and zeros will be in units of displacement with length units the same as contained in the SEED volume, which is almost always meters. The polezero file for a given sensor represents only one piece of the response information available in a SEED volume. The polezero file in this example is such a file. It includes only the first stage of the instrument response: the seismometer. Not included are later stages in the response that include FIR and IIR filters. EVALRESP calculates responses using all stages, but in most applications the POLEZERO stage suffices. Unlike EVALRESP, SAC does not know where the polezero file comes from, so in this case, it is necessary to multiply the waveform by 1.0e9 to convert from meters to nanometers. The command w OR075.z produces a waveform that is instrument corrected to displacement in nm.

Fot the above example, suppose one had not used SAC_PZs_XC_OR075_LHZ but instead has used an inappropriate PZ file: SAC_PZs_wrong. The following procedure shows how one can use one call to TRANSFER to take out the incorrect response and put in the correct response:

```
SAC> READ OR075.z
SAC> write OR075.zbad
SAC> SETBB pzo "SAC_PZs_wrong"
SAC> SETBB pzn "SAC_PZs_XC_OR075_LHZ"
SAC> TRANS FROM POLEZERO S %pzn TO POLEZERO S %pzo FREQ 0.008 0.015 0.2 0.4
SAC> write OR075.z
```

The first write statement makes a copy of the original file.

As a final example we consider the case for which one has several stations and BH* channels for waveforms from an event in the calling directory written by RDSEED v5.2. Assume one has either made a call to sacpz or concatinated all the BH* PZ files for this event into a single file named event.pz. The following sequence will read all the BH* waveforms into memory and overwrite those files in memory with instrument-corrected waveforms:

```
SAC> r *BH*SAC
SAC> rtr;taper
SAC> TRANS FROM POLEZERO S event.pz freq 0.05 0.1 10.0 15.0
```

FAPfile OPTION

Reintroduced into Sac in version 101.4, is the FAPfile option. A FAPFILE is an ascii file in which each line has three entries: a frequency (in HZ), an amplitude, and a phase (in degrees that will decrease with increasing frequency). This FROM option will deconvolve the waveform over the frequency range from the frequency in the first line to the frequency in the last line. The frequencies need not be equally spaced. When applying the correction, for frequencies less that the frequency in the first line, the amplitude and phase of that first line are used. Similarly, for frequencies greater than that in the last line, the amplitude and phase for the frequency in the last line are used.

As of version 3.3.2 in EVALRESP, a FAPfile output can be generated. An advantage of using a FAPFILE generated by EVALRESP rather than a POLEZERO file generated from the same RESP file is that one can include additional stages of the instrument response and/or control more explicitly the frequency range over which the correction is applied. Historically, a FAPFILE was used because one did not have a polezero file for the instrument or the full response included analog stages.

The format of a FAP file is consistent with that produced by the standalone program EVALRESP (v3.3.3), but is different from the format used by SAC2000.

FAP EXAMPLES

Suppose one has a fapfile fap.n11a.lhz_0.006-0.2, where the name is a short-hand for the fact that the frequency range is from0.006 HZto 0.2 HZ, and one wants to remove the instrument response from waveform 2006.253.14.30.24.0000.TA.N11A..LHZ.Q.SA.

```
SAC> READ 2006.253.14.30.24.0000.TA.N11A..LHZ.Q.SAC
SAC> RTR
SAC> TAPER
SAC> TRANSFER FROM FAP S fap.n11a.lhz_0.006-0.2 FREQ 0.004 0.006 0.1 0.2
SAC> MUL 1.0e9
```

COMMENTS: As with the EVALRESP and POLEZERO options, one should accompany the instrument correction with a FREQLIMITS option to handle the highest and lowest frequencies. (The Nyquist for this LHZ file is 0.5 Hz.)

TABLE OF AVAILABLE INSTRUMENT TYPES

ACC	acceleration [+]
BBDISP	Blacknest specification of Broadband Displacement
BBVEL	Blacknest specification of Broadband Velocity
BENBOG	Blacknest specification of Benioff by Bogert
DSS	LLNL Digital Seismic System
DWWSSN	Digital World Wide Standard Seismograph Station
EKALP6	Blacknest specification of EKA LP6
EKASP2	Blacknest specification of EKA SP2
ELMAG	Electromagnetic
EVALRESP	Response specified in SEED RESP files [++]
GBALP	Blacknest specification of GBA LP
GBASP	Blacknest specification of GBA SP
GENERAL	General seismometer
GSREF	USGS Refraction
HFSLPWB	Blacknest specification of HFS LPWB
IW	EYEOMG-spectral differentiation
LLL	LLL broadband analog seismometer
LLSN	LLSN L-4 seismometer
LNN	Livermore NTS Network instrument
LRSMLP	Blacknest specification of LRSM LP
LRSMSP	Blacknest specification of LRSM SP
NONE	displacement, this is the default [+]
NORESS	NORESS (NRSA)
NORESSHF	NORESS high frequency element
OLDBB	Old Blacknest specification of BB
OLDKIR	Old Blacknest specification of Kirnos
POLEZERO	reads Pole Zero file [++]

PORTABLE	Portable seismometer with PDR2
PTBLLP	Blacknest specification of PTBL LP
REDKIR	Blacknest specification of RED Kirnos
REFTEK	Reftek 97-01 portable instrument
RSTN	Regional Seismic Test Network
S750	S750 Seismometer
SANDIA	Sandia system 23 instrument
SANDIA3	Sandia new system with SL-210
SRO	Seismic Research Observatory
VEL	velocity [+]
WA	Wood-Anderson
WABN	Blacknest specification of Wood-Anderson
WIECH	Wiechert seismometer
WWLPBN	Blacknest specification of WWSSN long period
WWSP	WWSSN short period
WWSPBN	Blacknest specification of WWSSN short period
YKALP	Blacknest specification of YKA long period
YKASP	Blacknest specification of YKA short period

Note [+] ACC, VEL, and NONE do not refer to actual seismometer specifications but to acceleration, velocity, and displacement respectively. When these are specified as the TO type, IDEP is set accordingly.

Note [++] EVALRESP and POLEZERO do not refer to actual seismometer specifications. They are described in greater detail above.

INSTRUMENT TYPE OPTIONS

SUBTYPE: the following instrument types use the following subtypes:

LLL: LV, LR, LT, MV, MR, MT, EV, ER, ET, KV, KR, KT

LNN: BB, HF

NORESS: LP, IP, SP

POLEZERO: name of file to be read

RSTN: [CP, ON, NTR, NY, SD][KL, KM, KS, 7S][Z, N, E]

SANDIA: [N, O][T, L, B, D, N, E][V, R, T]

SRO: BB, SP, LPDE

FREEPERIOD v: ELMAG, GENERAL, IW, LLL SUBTYPE BB, REFTEK (v must be 15.0 or 30.0 for ELMAG)

MAGNIFICATION n: ELMAG, GENERAL (n must be 375, 750, 1500, 3000, or 6000 for ELMAG)

NZEROS n: GENERAL, IW

DAMPING v: GENERAL, LLL SUBTYPE BB, REFTEK

CORNER v: LLL SUBTYPE BB, REFTEK

GAIN v:

HIGHPASS v: REFTEK

EXAMPLES

To remove the instrument response from the RSTN station NYKM.Z and apply the instrument response for DSS without prewhitening (which is the default):

```
SAC> READ NYKM.Z
SAC> TRANS FROM RSTN SUBTYPE NYKM.Z TO DSS PREW OFF
```

To remove the LLL broadband instrument response and apply the SRO instrument response with frequency tapering and prewhitening:

```
SAC> READ ABC.Z
SAC> TRANS FROM LLL TO SRO FREQ .02 .05 1. 2. PREW 2
```

The passband of the resulting trace will be flat from .05 Hz to 1 Hz and will be zero below .02 Hz and above 2 Hz. Prewhitening of order 2 is applied in the time domain before deconvolution and the effect is removed in the time domain after convolution.

To transfer from the electromagnetic instrument response to displacement:

```
SAC>  READ XYZ.Z
SAC>  TRANSFER FROM ELMAG FREEP 15. MAG 750. TO NONE
```

ACKNOWLEDGEMENTS

Roger Hanscom did the original conversion of Keith Nakanishi's TRANSFER program. George Randall added the prewhitening option and was a major contributor to the testing and documentation of this command. Doug Dodge included the EVALRESP option.

LATEST REVISION

August 2011 (Version 101.5) In previous versions, if the header vaiable SCALE was defined, it might be used in scaling the output and would be changed. In this version, SCALE is ignored.

INSTRUMENT DETAILS IN TRANSFER

INSTRUMENT TYPES AND THEIR DEFINITIONS

Type Definition ACC Acceleration DSS LLNL Digital Seismic System DWWSSN Digital World Wide Standard Seismograph Station ELMAG Electromagnetic GENERAL General seismometer GSREF USGS Refraction IW EYEOMG-spectral differentiation S750 S750 Seismometer LLL LLL broadband analog seismometer LLSN LLSN L-4 seismometer LNN Livermore NTS Network instrument NORESS NORESS (NRSA) NORESSHF NORESS high frequency element POLEZERO Instrument specification in terms of poles and zeros PORTABLE Portable seismometer with PDR2 RSTN Regional Seismic Test Network REFTEK Reftek 97-01 portable instrument SANDIA Sandia system 23 instrument SANDIA3 Sandia new system with SL-210 SRO Sesimic Research Observatory VEL Velocity Spectal Operator WA Wood-Anderson WIECH Wiechert seismometer WWSP WWSSN short period NONE No instrument

References to the Blacknest specifications for various instrument types were removed from the documentation in version 10.6e and are no longer supported, although these instruments still remain in the SAC program.

INSTRUMENTS WHICH REQUIRE SUBTYPES

LLL LV, LR, LT, MV, MR, MT, EV, ER, ET, KV, KR, KT, BB (station and component abbreviation, or broadband) LNN BB, HF (broadband or high frequency) NORESS LP, IP, SP (long-, intermediate- or short-period) POLEZERO file (file describing poles and zeroes) RSTN [CP, ON, NT, NY, SD][KL, KM, KS, 7S][Z, N, E] (station; KS36000 long-, medium- or short-period, or S750 short-period; component) SANDIA [N, O][T, L, B, D, N, E][V, R, T] (new or old acquisition system; station abbreviation; component) SRO BB, SP, LP (broadband, short or long period)

INSTRUMENTS WHICH REQUIRE OTHER OPTIONS

ELMAG FREEPERIOD v where v = one of 15.0,30.0 MAGNIFICATION n where n = one of 375,750,1500,3000,6000 GENERAL FREEPERIOD v DAMPING v LLL SUBTYPE BB DAMPING v FREEPERIOD v REFTEK FREEPERIOD v DAMPING v CORNER v HIGHPASS v

TRAVELTIME

SUMMARY

Computes traveltime of selected phases for pre-defined velocity models. To get the Signal-Stacking Subprocess versio nof traveltime, go to SSSTRAVELTIME.

SYNTAX

```
[T]RAVEL[TIME] {MODEL string} {PICKS number} {PHASE phase list}  {VERBOSE | QUIET} {M
```

INPUT

MODEL: iasp91 [default], ak135

Picks: There are a total of 10 time picks in the SAC header: t0 to t9. If the number is n (0<=n<=9), the first phase will be at Tn. If PICKS is not included among the command-line options, VERBOSE will be turned on and the phase arrival times will be displayed but will not be put in the header.

PHASE: List of phases for which times are picked and displayed. If PICKS n is among the options, the phase arrivala times and their labels will be added to the header starting at Tn. If n is 0, one can have up to 10 phases, if n is 8, one can have no more than 2.

VERBOSE | QUIET: If VERBOSE is among the options in the TRAVELTIME command line, phase arrival times are displayed relative to both the origin time (O) and to the first-point time (B). If QUIET is among the options in the TRAVELTIME command line, VERBOSE is turned off (if it was on) and nothing is displayed on the screen. If neither has been displayed, the depth used by TRAVELTIME is displayed.

M | KM: If M is among the options in the command line, SAC interprets EVDP as being in meters. If KM is displayed, EVDP is interpreted as being in kilometers.

DEFAULT VALUES: MODEL iasp91 KM PHASE P S Pn Pg Sn Sg

DESCRIPTION

All waeforms in memory must have event and station locations defined as well as the origin time.

This command calculates traveltimes using the iaspei-tau procedures developed for models iasp91 or ak135. For more information on this package, go to the iaspei-tau link at URL <https://seiscode.iris.washington.edu/projects/iaspei-tau/>. The phase picks are stored in the SAC header for all files in memory in header variables Tn, where n is in the range 0 to 9. The times are calculated relative to the origin time (O) but the Tn times are relative to the first-point time (B).

The traveltime tables used to calculate the stored degree-distance measure (GCARC). (GCARC is calculated from the event and station latitude and longitudes using spherical-triangle geometry after converting geographic latitudes to geocentric.)

For historical reason, the units for EVDP were meters, and SAC waveforms proudced by RDSEED have EVDP in meters. In v101.5, the default for EVDP is kilometers, but as many waveforms have EVDP in meters, we introduced the command option M which if set to ON means the input EVDP for all files in memory have EVDP inmeters. In the final example, the SAC files had been extracted from a SEED vlume using RDSEED for a version prior to v5.2, so EVDEP is in meters, (Starting with v5.2, the depth in RDSEED is in kilometers.)

In SAC2000 (the most recent releae in which TRAVELTIME was functional) , TRAVELTIME was in the Signal Stacking Subprocess (SSS). The current version (V101.5) does not have all the capabilities of the SAC2000 version and can run without going into SSS. See SSSTRAVELTIME for the syntax in SSS.

EXAMPLES

A regional event using the default phases:

```
SAC> fg seismo
SAC>  traveltime
traveltime: depth: 15.000000
traveltime: error finding phase P
traveltime: error finding phase S
traveltime: setting phase Pn       at 10.464321 s [ t = 51.894321 s ]
traveltime: setting phase Pg       at 22.904724 s [ t = 64.334724 s ]
traveltime: setting phase Sn       at 50.047722 s [ t = 91.477722 s ]
traveltime: setting phase Sg       at 66.414337 s [ t = 107.844337 s ]
SAC>
```

For regional events the first arrivals are Pn or Pg, so by the convention used here there is no "P" arival. For the above example, no picks are added to the header.:

```
SAC> fg seismo
SAC>  traveltime picks 0 phase Pn Pg Sn Sg
traveltime: depth: 15.000000
SAC> lh AMARKER T0MARKER T1MARKER T2MARKER T3MARKER
AMARKER = 10.464                        T0MARKER = 10.464          (Pn)
T1MARKER = 22.905          (Pg)         T2MARKER = 50.048          (Sn)
T3MARKER = 66.414          (Sg)
SAC> write seismo-picks.z
SAC>
```

We see that the already defined A is at Pn. The file written to seismo-picks.z will have T0 through T3 in the header, and a call to PLOT1 will show labeled vertical lines at the calculated times.

Note that even though VERBOSE was not turned on, the depth (in km) used is printed out. This is a safeguard to assure that one has made the correct assumption about the EVDP units. The printing of traveltime depth can be suppressed by entering QUIET on the command line.

The header for the waveform in the following example was produced by RDSEED (V5.0) and EVDP is in meters:

```
SAC> r 2008.052.14.16.03.0000.XC.OR075.00.LHZ.M.SAC
SAC> lh evdp
evdp = 6.700000e+03
SAC> traveltime M picks 0
traveltime: depth: 6.700000 km
SAC> lh t0marker t1marker t2marker t3marker
t0marker = 61.48           (Pn)         t1marker = 76.413          (Pg)
t2marker = 109.66          (Sn)         t3marker = 132.11          (Sg)
SAC> ch evdp (0.001 * &1,evdp&)
SAC> setbb station &1,KSTNM&
SAC> write %station%.z
SAC>
```

The saved file, OR075.z, will have evdp in kilometers and annotated picks at the times for Pn, Pg, Sn, and Sg. (One could have done the chnhdr command before calling TRAVELTME and then not entered M on the command line.)

Phase names are case sensitive. See the iaspei-tau documentation.

LATEST REVISION

August, 2011 (Version 101.5)

TSIZE

SUMMARY

Controls the text size attributes.

SYNTAX

`TSIZE {size v},{RATIO v},{OLD|NEW}` where size is one of the following:

`TINY|SMALL|MEDIUM|LARGE`

INPUT

size v: Change the value of one of the text sizes to v.

RATIO v: Change the text width to height ratio to v.

OLD: Change the values of all the text sizes to their "old" values. These are the values used by SAC prior to Version 9.

NEW: Change the values of all the text sizes to their "new" values. These are the default values when SAC is initialized.

DEFAULT VALUES

`TSIZE RATIO 1.0 NEW`

DESCRIPTION

Most of the text annotation commands (TITLE, XLABEL, FILEID, etc.) allow you to change the size of the text being displayed. You may choose from a set of four named sizes (TINY, SMALL, MEDIUM, and LARGE.) Each named size has an initial value given in the table below. These sizes are the height of a character as a fraction of the full (0.0 to 1.0) viewspace. There are times when you may want some of this annotation to be of a size different from these default values. TSIZE allows you to redefine any or all of these four named sizes. You may also use this command to change the width to height ratio of the characters.

The default text sizes were changed, starting with Version 9 of SAC. The new set covers a wider range and generally looks better on most devices. You can easily change back to the original set of sizes by using the OLD option. This might be useful if you want to create a plot that looks very similar to one that was generated using an older version of SAC. Also old PLOTC files and macros will not look the same when replotted unless you first set the text sizes to their old values. The NEW option resets the sizes to their default values.

DEFAULT TEXT SIZES

NAME	A	B	C	D	E
TINY	0.015	66	50	68	110
SMALL	0.020	50	37	66	82
MEDIUM	0.030	33	25	44	55
LARGE	0.040	25	18	33	41

The column definitions in the table above are as follows:

- A Height of character as a fraction of full viewspace.
- B Number of lines of text in full viewspace.
- **C Number of lines of text in a normal viewspace.** Normal means 0. to 1. in x and 0. to 0.75 in y.
- D Minimum number of characters per normal viewspace line.
- **E Average number of characters per normal viewspace line.** This is larger because the text is proportionally spaced.

EXAMPLES

To change the definition of MEDIUM and then use it to create a specially sized title:

```
u:  TSIZE MEDIUM 0.35
u:  TITLE 'Rayleigh Wave Spectra' SIZE MEDIUM
u:  PLOT2
```

To reset this (and any other) size definitions to their default values:

```
u:  TSIZE NEW
```

SEE COMMANDS

TITLE, XLABEL, FILEID, PLOTC

LATEST REVISION

July 22, 1991 (Version 9.1)

UNSETBB

SUMMARY

Unsets (deletes) blackboard variables.

SYNTAX

```
UNSETBB ALL | variable ...
```

INPUT

ALL: Unset all of the currently defined blackboard variables.
variable: Unset the blackboard variable variable.

DESCRIPTION

The blackboard is a place to temporarily store information. Blackboard variables are defined using the SETBB and EVALUATE commands. They can be accessed by the GETBB command or used directly in a command by preceeding the name of the variable with a percent sign ("%".) This command allows you to unset previously defined blackboard variables. You may unset all variables or only a named subset.

EXAMPLES

To unset several blackboard variables at once:

```
u:  UNSETBB C1 C2 X
```

To unset all blackboard variables:

```
u:  UNSETBB ALL
```

SEE COMMANDS

SETBB, EVALUATE, GETBB

LATEST REVISION

January 26, 1989 (Version 10.4B)

UNWRAP

SUMMARY

Computes amplitude and unwrapped phase.

SYNTAX

```
UNWRAP {FILL {ON|OFF|n}},{INTTHR v},{PVTHR v}
```

INPUT

FILL {ON}: Turn zero fill option on.
FILL OFF: Turn zero fill option off.
FILL n: Turn zero fill option on and change fill value to n.
INTTHR v: Change the integration threshold constant to v.
PVTHR v: Change the principal value threshold constant to v.

DEFAULT VALUES

```
UNWRAP FILL OFF INTTHR 1.5 PVTHR 0.5
```

DESCRIPTION

This command transforms time-series data in memory to spectral data containing amplitude and "unwrapped" phase components. This procedure works for data with a "smoothly varying phase." The data is filled with zeros to the next power of two before being transformed. You may specify a larger number of zeros by using the FILL option.

This is an implementation of the algorithm due to Tribolet. Two methods are used to estimate the unwrapped phase at each frequency. One is to numerically integrate the phase derivative through the use of the fast Fourier transform. The step size used in this trapezoidal integration is halved at each frequency if necessary to obtain a consistent estimate. You can control the threshold value on this check using the INTTHR option. This value is in radians. Decreasing INTTHR will improve the phase estimate. Too small a value, however, will not allow the solution to converge.

The second method used in this algorithm is to first compute the principle value of the phase using the inverse tangent function. The unwrapped phase is estimated by adding multiples of 2*PI to the principal value until the discontinuities are reduced to values less than a threshold value. You control the threshold value on this check using the PVTHR option. Again, decreasing this threshold value will improved the phase estimate, but will also increase the chance that no solution may be found. Initial trial values for these two thresholds are usually such that:

```
PI/4 < PVTHR < INTTHR < 2*PI
```

ERROR MESSAGES

- 1301: No data files read in.
- 1306: Illegal operation on unevenly spaced file
- 1606: Maximum allowable DFT is
  - Too many data points in file.

WARNING MESSAGES

- 1610: Unwrap failed at data point for file
  - Adjust threshold constants and retry.

HEADER CHANGES

B, E, and DELTA are changed to the beginning, ending and sampling frequencies of the transform respectively. The original values of B, E, and DELTA are saved as SB, SE, and SDELTA and are restored when an inverse transform is performed.

LIMITATIONS

The maximum length of data that can currently be transformed is 4096. Tribolet, Jose M.; "A New Phase Unwrapping Algorithm"; IEEE Transactions on Acoustics, Speech, and Signal Processing; Vol. ASSP-25, No 2, April 1977; page 170.

LATEST REVISION

January 8, 1983 (Version 8.0)

VSPACE

SUMMARY

Changes the maximum size and shape of plots.

SYNTAX

```
VSPACE FULL|v
```

INPUT

FULL: Use full viewspace. This is the largest possible screen or window size.

v: Force the viewspace to have a y:x aspect ratio of v. The largest possible area with this aspect ratio becomes the viewspace.

DEFAULT VALUES

```
VSPACE FULL
```

DESCRIPTION

The viewspace represents that portion of the viewing surface on which plots can be drawn. There is a large variation in viewspace shapes and sizes between different graphics devices:

1. Although differing greatly in size, many graphics terminals have an aspect ratio of 0.75. Some terminals, however, have different aspect ratios. The HP 26xx family of terminals have an aspect ratio of 0.5. The Tektronix 4025 terminals can have a wide range of aspect ratios, depending upon how many lines on the screen are assigned to the alphanumeric display and how many to the graphic display.
2. The SAC Graphics File (SGF) has an aspect ratio of 0.75 This is the approximate ratio of a standard sheet of 8.5 by 11 paper.
3. The graphics windows created by the XWINDOWS or SUNWINDOWS device can have any aspect ratio you wish.

This variation among graphics devices can be a problem if you are need complete control over the size and shape of a plot. This command gives you control over the shape of a plot by letting you select a fixed aspect ratio. (SAC does not currently give you much control over the size.) The default is to plot to the full viewspace. If you do select a fixed aspect ratio, then the viewspace becomes the largest enclosed area on the device with that aspect ratio.

This command is useful when you are creating a figure on an interactive device using PLOTC that you eventually want to send to the SGF device. You should set the aspect ratio to 0.75 before doing any plotting. This will ensure that the figure will have the same in the SGF file as it is does on the interactive device. Another use is when you want a square viewspace independent of the graphics device. This is easily done by requesting an aspect ratio of 1.0.

LATEST REVISION

May 15, 1987 (Version 10.2)

WAIT

SUMMARY

Tells SAC whether or not to pause between plots.

SYNTAX

```
WAIT {ON|OFF|EVERY}
```

INPUT

{ON}: Turn wait option on in normal mode.
OFF: Turn wait option off.
EVERY: Turn wait option on in every plot mode.

DEFAULT VALUES

```
WAIT ON
```

DESCRIPTION

When you read in more than one data file and then plot them using the PLOT command, one frame is generated for each file. If you are plotting to the terminal, SAC normally pauses after each plot and sends the message "WAITING" to the terminal. You can then hit the return key to see the next plot, type "GO" to have SAC plot the rest of the current set of plots without pausing, or type "KILL" to terminate the plotting of this set of files. SAC does not pause after the last plot, because the normal input prompt serves the same function. When this wait option is off, SAC does not pause between plots. With the wait option in the "every plot" mode SAC will pause between every plot, not just the ones generated by the PLOT command. This is useful when you are running SAC under the control of a command file or job control program.

EXAMPLES

The following example shows how SAC functions in the normal wait mode:

```
u:  READ FILE1 FILE2 FILE3 FILE4
u:  PLOT
s:  waiting         {plot of FILE1 to terminal}
u:  (return)
u:  waiting         {plot of FILE2}
u:  kill            {user has seen enough}
s:  (prompt)        {SAC now waiting for next command}
```

LATEST REVISION

October 11, 1984 (Version 9.1)

WHITEN

SUMMARY

Flattens the spectrum of the input time series.

SYNTAX

```
WHITEN {N} {FILTERDESIGN|FD}
```

INPUT

N: The order (number of poles). The higher this number, the flatter the resultent data. High orders can clean the data up better, but they can clean the data up too much, and important data can be lost if the order is set too high. The default is 6.

FD: Performs something akin to the filterdesign command. Using the whitening coefficients, it designs the whitening filter. Output is written to disc as a set of three output files per input file. Output files have the following suffixes:

- .imp = impulse response,
- .spec = spectral

responses (amplitude and phase), and .gd = group delay. Note that while the group delay claims to be a time series file, it is really a frequency series.

DEFAULT VALUES

```
WHITEN 6
```

DESCRIPTION

Add white noise to the data. Flattens the spectrum of the input time series. When this is performed prior to the spectral commands (like those in SPE, or transfer or spectrogram), it reduces the dynamic range of the spectral values, and improves the accuracy of the overall operation at high frequencies for seismic data.

Note WHITEN can be called from within the SPE subprocess, or from SAC's main shell. The WHITEN in SPE maintains the order separately from the WHITEN in the main shell. From the main shell, you can call WHITEN 4. Future calls to WHITEN from the main shell will have an order of 4, but calls to WHITEN in SPE will still have order of 6, unless it is changed at the commandline in SPE. Furthermore, the order in SPE is the same as the order related to the PREWHITEN option in SPE's COR command (setting one sets the other). Also the order in the main shell's WHITEN command is the same as the order in the TRANSFER command's PREWHITEN option.

SEE COMMANDS

SPE, SPE_COR, TRANSFER

WHPF

SUMMARY

Writes auxiliary cards into the HYPO pick file.

SYNTAX

```
WHPF IC n m
```

INPUT

IC n m: Insert an "instruction card" with the two integers n and m in columns 18 and 19. Allowed values for n are 0, 1, 5, and 6. Allowed values for m are 0, 1, and 9.

DESCRIPTION

The "instruction card" can be used to separate events in a HYPO pick file. See the HYPO71 manual for details on the use of this card. Closing an open HYPO pick file (CHPF command) or quitting SAC automatically appends the "10" instruction card to the HYPO pick file.

ERROR MESSAGES

- 1908: HYPO pick file not open.

SEE COMMANDS

CHPF, OHPF W.H.K. Lee and J.C. Lahr; HYPO71 (Revised): A Computer Program for Determining Hypocenter, Magnitude, and First Motion Pattern of Local Earthquakes; U. S. Geological Survey report 75-311.

LATEST REVISION

March 20,1992 (Version 10.6e)

WIDTH

SUMMARY

Controls line-width selection for graphics devices.

SYNTAX

```
WIDTH  {ON|OFF|linewidth} options
```

where options are one or more of the following:

```
{SKELETON width}
{INCREMENT {ON|OFF}}
{LIST STANDARD|widthlist}
```

and where linewidth, width and widthlist are integer values.

SPECIAL NOTE The LIST option must appear last in this command.

INPUT

WIDTH ON: Turn WIDTH option on but don't change current width values.
WIDTH OFF: Turn width option off.
WIDTH linewidth: Change data width to linewidth and turn WIDTH option on.
SKELETON width: Change width of skeleton to width and turn WIDTH option on.
INCREMENT {ON}: Increment width from widthlist list after each data file is plotted.
INCREMENT OFF: Do not increment data line width.
LIST widthlist: Change the content of the width list. Enter list of widths. Sets data width to first width in list and turns width option on.
LIST STANDARD: Change to the standard width list. Sets data width to first width in list and turns width option on.

DEFAULT VALUES

```
WIDTH OFF SKELETON 1 INCREMENT OFF LIST STANDARD
```

DESCRIPTION

This command controls width attributes for those devices which can display a large number of line-widths. The data width is the width that is used when plotting the data files. The data width may be automatically incremented from a width list after each data file is plotted. The skeleton width is the width used to plot the axes. Only plot axes change with SKELETON option. Grids, text, labels and frame ids are always displayed with the thin line-width of value 1.

If plotting several data files on the same plot, you may want each to be in a different width. This is done using the INCREMENT option. When this option is on, the data width is incremented from a list of widths each time a data file is plotted. The order and value of widths in the standard (default) list is:

```
1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

You may change the order or content of this list using the LIST option. This is useful for doing overlay plots (see PLOT2) when you want the same widths used in the same order on each plot.

EXAMPLES

To select an incrementing data width starting with 1:

```
u:  WIDTH 1 INCREMENT
```

To set up an incrementing data width list of 1, 3, and 5 with an skeleton of 2:

```
u:  WIDTH SKELETON 2 INCREMENT list 1 3 5
```

LATEST REVISION

June 20, 1992 (Version 10.6e)

WIENER

SUMMARY

Designs and applies an adaptive Wiener filter.

SYNTAX

```
WIENER {[W]INDOW pdw} {[N]COEFF n} {MU OFF | ON | v} {[EPS]ILON OFF | ON
| e}
```

INPUT

WINDOW pdw: Set filter design window to pdw. A partial data window which consists of a start time and stop time. These times can be absolute ones or ones relative to certain header fields. See the CUT command for details on pdw.

NCOEFF n: Set the number of filter coefficients to n.

MU off | on | v: Set the adaptation step size parameter. Off sets mu to zero. On sets mu = 1.95 / Rho(0). Where Rho(0) is the autocorrelation in pdw at zero lag. v sets mu = v.

EPSILON e: Set ridge regression parameter to epsilon. Can help stabilize the wiener filter by increasing the diagonal elements of the autocorrelation matrix by epsilon. When epsilon is ON, SAC will use the value entered by the user (or zero if no value was entered). When epsilon is OFF, SAC will loop through the following increasing values of epsilon (0.0, 1e-5, 1e-4, 1e-3, 1e-2), until the wiener filter is stable, or until the list has been exhausted. If epsilon == 0 does not work, SAC will produce one or more warning messages. If none of the values work, SAC will produce an error message.

DEFAULT VALUES

```
WIENER WINDOW B 0 10 NCOEFF 30 MU OFF EPSILON OFF
```

DESCRIPTION

A prediction error filter is designed using the Yule-Walker Method from an autocorrelation function estimated from the designated partial data window. This window can be any portion of the file. The filter is then applied to the entire signal, i.e. the signal is replaced by the residual error sequence. This filter may be used as a prewhitener or as a detection preprocessor for transient signals. The filter can be made adaptive in time by specifying a non-zero value for MU. Large values of MU may cause instability.

EXAMPLES

The following command would apply a non-adaptive filter, with the first ten seconds being the design window:

```
u: WIENER WINDOW B 0 10 MU 0.
```

The following command would apply a filter with 40 coefficients, with a design window from the beginning of the file to 1 second before the first arrival:

```
u: WIENER NCOEFF 40 WINDOW B A -1
```

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN

ERROR MESSAGES

- 1301: No data files read in.
- 1306: Illegal operation on unevenly spaced file
- 1307: Illegal operation on spectral file
- 1608: Bad Wiener filter noise window for file
  - Filter design window does not lie within file window. - Make sure header fields used in window are defined.

WARNING MESSAGES

- 1609: Numerical instability in Wiener filter for file
- 1614: Numerical instability in Wiener; will retry with epsilon = e where e denotes the next value of epsilon to be tried.
  - The filtered data may or may not be incorrect.

SEE COMMANDS

CUT

LATEST REVISION

March 12, 1997 (Version 00.53)

WILD

SUMMARY

Sets wildcard characters used in read commands to expand filelists.

SYNTAX

```
WILD {ECHO {ON|OFF}},{SINGLE char},{MULTIPLE char},
{CONCATENATION chars}
```

INPUT

ECHO {ON}: Turn echoing of expanded filelist on. Echoing is only when this option is on and there are wildcard characters in the filelist.

ECHO OFF: Turn echoing of expanded filelist off.

SINGLE char: Change the character used to match single characters.

MULTIPLE char: Change the character used to match multiple characters.

CONCATENATION chars: Change the two characters used to enclose concatenation lists.

DEFAULT VALUES

OPTION UNIX VAX PRIME ECHO ON ON ON SINGLE ? ? + MULTIPLE * * ' CONCATENATION [,] (,) [,]

DESCRIPTION

This feature is available at the command level of many modern operating systems and is called "wildcarding" or "filename expansion." It is a notation that allows you to abbreviate filenames and to specify entire groups of files using a simple shorthand notation. SAC has implemented wildcarding, along with several extensions, in its READ, READALPHA, and READHDR commands. Using this notation, you can easily access lists such as:

- All files beginning with the letters "abc".
- All files ending with the letter "z".
- All files with exactly three letters in their names.

There are three elements in this wildcard notation. We will use the default wildcard characters for the UNIX version in this description and in the examples below. The defaults may be different on the computer you are using. You may also use this command to change the characters to be anything you want. The multiple match character ("*") is used to match an arbitrary character string, including an empty string. The single match character ("?") is used to match any single character. The concatenation characters ("[" and "]") are used to enclose a comma delimited list of character strings to match. The character strings in a concatenation list may not contain the single or multiple match wildcard characters. These are the steps that SAC uses to perform this wildcard filename expansion:

1) Strip away the directory part of the token if it exists. Otherwise use the current directory.
2) Make a system call to get a list of all files in the directory.

3) If a concatenation list is in the token, form new tokens from each character string in the concatenation list with the other characters in the token and then match them to the list of files. If there is no concatenation list in the token, simply match the token to the list of files.
4) Remove any duplicates matches to form the expanded filelist.
5) Echo the expanded filelist if requested.
6) Attempt to read the expanded filelist into memory.

Each operating system uses a somewhat different scheme to store and access files in a directory. The system call in (1) above reflects these differences. For example, the filenames are returned in alphabetical order in UNIX but are not on the PRIME or VAX. The order of the files in a PRIME directory is arbitrary. These differences are reflected in the order of the files in the expanded filelist. You may have to experiment with different variations of wildcard characters and concatenation lists if the order of the files in the expanded list is important.

The examples below will help clarify how to use these wildcard elements. One useful feature is that SAC saves the character strings contained in the concatenation list. When you enter an empty list, then the previous list is reused. This can save a lot of typing.

EXAMPLES

Assume that the contents of the current directory contain the following files in the order shown:

```
ABC   DEF   STA01E   STA01N   STA01Z   STA02E   STA02N   STA02Z   STA03Z
```

Also, assume that expanded filelist echoing is on. The following shows how the various wildcarding elements can be used to read parts of the above filelist into memory.:

```
u:  READ S\*
s:  STA01E STA01N STA01Z STA02E STA02N STA02Z STA03Z
u:  READ \*Z
s:  STA01Z STA02Z STA03Z
u:  READ ???
s:  ABC DEF
u:  READ STA01[Z,N,E]
s:  STA01Z STA01N STA01E
u:  READ *[Z,N,E]
s:  STA01Z STA02Z STA03Z STA01N STA02N STA01E STA02E
u:  READ *1[Z,N,E] *2[ ]
s:  STA01Z STA01N STA01E STA02Z STA02N STA02E
```

LIMITATIONS

You may have only one concatenation string in a token. This limitation will be eliminated in a future version. Several other wildcard and filename expansion options will also be added at that time.

SEE COMMANDS

READ, READALPHA, READHDR

LATEST REVISION

May 15, 1987 (Version 10.2)

WINDOW

SUMMARY

Sets the location and aspect ratio of graphics windows.

SYNTAX

```
WINDOW n {XSIZE xwmin xwmax} {YSIZE ywmin ywmax} {ASPECT [ value | ON |
OFF ]}
```

INPUT

n: The graphics window number of interest. There are a total of five possible graphics windows.

X xwmin xwmax: Set the x (horizontal) location of graphics window n on the screen. xwmin is the location of the left edge of the window and xwmax is the location of the right edge. The range of these screen coordinates is 0.0 to 1.0.

Y ywmin ywmax: Set the y (vertical) location of graphics window n on the screen. ywmin is the location of the bottom edge of the window and ywmax is the location of the top edge. The range of these screen coordinates is 0.0 to 1.0.

ASPECT value: If ASPECT is ON (the default), xwmax is not used and the xsize is set so that the aspect ratio (xsize/ysize) on the screen for the window is value if given or the default if no value is explicitly included. If a xsize is not followed by ASPECT on the command line, ASPECT is effectively OFF and xwmax is used. xwmin is always used.

Default settings:

```
n  xwmin  xwmax  ywmin  ywmax
1  0.05   0.65   0.45   0.95
2  0.07   0.67   0.43   0.93
3  0.09   0.69   0.41   0.91
4  0.11   0.71   0.39   0.89
5  0.13   0.73   0.37   0.87

ASPECT is ON and set to 11.0/8.5 = 1.294, so xwmax is not used.
```

DESCRIPTION

Most graphics terminals and workstations support multiple "windows." Different jobs or activities can run in each window and display their results on the screen at the same time. "X-windows" is the (X11) window system supported by SAC.

The BEGINWINDOW command lets you select the window in which to display subsequent plots. BEGINWINDOW will create the requested window -- erasing its contents, if it already existed. If one wants to change a setting from the default setting, the WINDOW command must be called BEFORE a call to BEGINWINDOW. On most systems the user can also move and resize these windows dynamically using the mouse and pop-up menus. Generally but not always (you should check for yourself), the moving of a window will result in the current plot being redrawn automatically, whereas the resizing of a window results in the current plot being redrawn but not rescaled. The next plot in a resized window will be scaled correctly.

Window choice n=1 is the default window. Its left side is near the left of the display and is towards the top of the display. If your window from which you are running SAC is towards the bottom of the display, on many systems you can be typing commands while looking at the plot. Default windows 2 through 5 are moved down and to the right by 0.02 screen units for as n is incremented. Depending on your platform and display, having these options for windows being displayed at the same time makes it efficient to compare the displays for windows with different values of n. All five of the above windows have the same aspect ratio.

If ASPECT is OFF, the aspect ratio of the displayed window depends on the aspect ratio of the screen. Older terminals typically had an x:y aspect ratio of 4:3; newer ones are more varied, but many have an aspect ratio of 16:10. On a 4:3 screen, the default window has an aspect ratio of 1.6:1, and on a 16:10 screen the aspect ratio is 1,9:1. SGF files have a fixed aspect ratio of 4:3.

EXAMPLES

To change from the default (n=1) to n=2 giving more space at the left of the screen:

```
SAC> window 1 x 0.25 0.85
SAC> beginwindow 1
```

The y limits will be unchanged. If the xsize are explicitly stated, ASPECT is set to OFF. Note that the order matters:

```
SAC> window 1 ASPECT 1.33 x 0.25 0.85
SAC> beginwindow 1
```

will give the same answer as the previous command sequence because later items on the command line take precedence. However,

```
SAC> window 1 x 0.25 0.85 ASPECT (4.0 / 3.0)
SAC> beginwindow 1
```

will ignore xwmax and fix the aspect ratio to 4/3 = 1.33.

To get the n=3 window from previous versions of SAC, enter the following:

```
SAC> window 3 x 0.05 0.85 y 0.05 0.60
SAC> beginwindow 3
```

LATEST REVISION

August 2011 (v 101.5)

WRITE

SUMMARY

Writes data in memory to disk.

SYNTAX

```
WRITE {options} {namingoptions}
```

where options are one or more of the following:

```
SAC|ALPHA|XDR
COMMIT|ROLLBACK|RECALLTRACE
DIR OFF|CURRENT|name
KSTCMP
```

These options MUST preceed any element in the namingoptions:

```
OVER
APPEND text
PREPEND text
DELETE text
CHANGE text1 text2
filelist
```

Only one of these namingoptions is allowed at a time.

INPUT

no arguments: Use previous format and previous write filelist.

SAC: Write in SAC binary data file format.

ALPHA: Write in SAC alphanumeric data file format.

SEGY: Write file formatted according to the IRIS/PASSCAL form of the SEGY format. This format allows one waveform per file. **Note** only evenly-spaced, time-series files will be written in SEGY.
The SCALE field in SAC is ignored. Since SAC stores the waveform as a series of floating point (real) numbers, and SEGY stores the waveform as a series of long integers, the datapoints from SAC are normalized to the maxmimum allowable integer. The scale field in SEGY is determined to be the factor which will restore the waveform as close as possible to that of the original SAC file, when read with the READ SEGY command.
The following SAC header fields are saved as the following SEGY header fields

SAC	SEGY
KZDATE	year, day, hour, minute, second, and m_secs are .
KZTIME	set to BEGIN time corrected by KZDATE and KZTIME.
BEGIN	trigyear, trigday, trighour, trigminute, trigsecond,
ORIGIN	and trigmills are ORIGIN corrected by KZDATE & TIME.
NPTS	sampleLength and/or num_samps

... continued on next page

SAC	SEGY
DELTA	deltaSample and/or samp_rate
DEPMAZ	max, corrected by SEGY's scale.
DEPMIN	min, corrected by SEGY's scale.
DIST	sourceToRecDist
STLA	recLatOrY (written as latitude in degrees)
STLO	recLongOrX (written as longitude in degrees)
EVLA	sourceLatOrY (written as latitude in degrees)
EVLO	sourceLongOrX (written as longitude in degrees) lats and lons are multiplied by 3600 to correct units
STEL	recElevation
EVEL	sourceSurfaceElevation
EVDP	sourceDepth
KSTNM	station_name
KCMPNM	channel_name
KEVNM	event_number (only if KEVNM is numeric and < 1e09)

The following SEGY fields are hardwired

SEGY	Value
elevationScale	1
coordScale	1
coordUnits	2
gainType	1
gainConst	1
data_form	1

XDR: Write in SAC binary xdr format. This format is used for the moving binary data files to/from a different architecture, such as a pc running LINUX.

COMMIT: Commits headers and waveforms in SAC memory -- removing any previous versions of headers or waveforms from RAM -- prior to writing files. COMMIT is the default.

ROLLBACK: reverts to the last committed version of the header and waveform before writing files.

RECALLTRACE:
- reverts to the last committed version of the waveform,
- reverts to the last committed version of those header variables closely linked to the waveform,
- commits those header variables which are loosely linked to the waveform. (See RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

DIR OFF: Turn directory option off. When off, writes to current directory.

DIR CURRENT: Turn directory option on and set name of write directory to the "current directory" (e.g. the directory from which you started SAC.)

DIR name: Turn directory option on and set name of write directory to name. Write all filenames to the directory called name. This may be a relative or absolute directory name.

KSTCMP: Use the KSTNM and KCMPNM header variables to define a file name for each data file in memory. The names generated will be checked for uniqueness, and will

have sequencing digits added as necessary to avoid name clashes.

OVER: Use current read filelist as write filelist. Overwrite files on disk with data in memory.

APPEND text: Write filelist is created by appending text to each name in the current read filelist.

PREPEND text: Write filelist is created by prepending text to each name in the current read filelist.

DELETE text: Write filelist is created by deleting the first occurrence of text in each name in the current read filelist.

CHANGE text1 text2: Write filelist is created by changing the first occurrence of text1 in each name in the current read filelist to text2.

filelist: Write filelist is set to filelist. This list may contain simple filenames, relative pathnames, or full pathnames. IT MAY NOT CONTAIN WILDCARDS.

DEFAULT VALUES

```
WRITE SAC COMMIT
```

DESCRIPTION

This command allows you, at any point in the processing of data, to save the results on disk. Several disk file formats are available. More will be added as needed. Each file in memory is written without being cut or desampled. Most of the time, you will want to use to the SAC data file format. This is a compact binary file format which is fast to read and write. It contains a large header record and one or two data records. See the Users Manual for details on the physical format. The alphanumeric data file format is an ASCII equivalent of the SAC data file format. It takes up much more room on disk and is much slower to read and write. It is useful if you wish to look at the content of the file using a text editor or wish to transfer data to a different kind of computer.

You can directly specify the names of the files to write or you can indirectly specify them by having SAC modify the names of files that are currently in memory. The OVER options sets the write file list to the read file list. It is used to overwrite the last set of disk files read with the data that is currently in memory. The APPEND, PREPEND, DELETE, or CHANGE options create a write file list by modifying each of the names in the read file list in the requested way. This is very useful in macros where you are automatically processing large numbers of data files and need to keep trace of the output files in a consistent manner. The write file list is output when any of these four options is selected. This lets you see the names that were actually used.

EXAMPLES

To filter a set of data files and then save the results in a new set of data files:

```
u:  READ D1 D2 D3
u:  LOWPASS BUTTER NPOLES 4
u:  WRITE F1 F2 F3
```

This could have also been done using the CHANGE option:

```
u:  READ D1 D2 D3
u:  LOWPASS BUTTER NPOLES 4
u:  WRITE CHANGE D F
s:  F1 F2 F3
```

Notice that SAC output the write file list in this case. To replace the original data on disk with the filtered data the third line in the above example would be:

```
u:  WRITE OVER
```

Note: for examples of the behavior of COMMIT, ROLLBACK, and RECALLTRACE, see the commands of the same name.

ERROR MESSAGES

- 1301: No data files read in.
- 1311: No list of filenames to write.
- 1312: Bad number of files in write file list:
  - the number of files in the write file list must be the same as the number in the data file list (the number read into memory).
- 1303: Overwrite flag is not on for file
  - header variable LOVROK is .FALSE.
  - this provides some protection for valuable data.

SEE COMMANDS

READ, COMMIT, ROLLBACK, RECALLTRACE

ACKNOWLEDGEMENTS

Our thanks to Steve Roecker or RPI for providing SAC2SEGY which served as our starting point.

LATEST REVISION

Oct. 27, 1998 (Version 0.58)

WRITEBBF

SUMMARY

Writes a blackboard variable file to disk.

SYNTAX

```
WRITEBBF {file}
```

INPUT

file: The name of a blackboard variable file. It may be a simple filename or a relative or absolute pathname.

DEFAULT VALUES

```
WRITEBBF BBF
```

DESCRIPTION

This command lets you write a blackboard variable file to disk. It can later be read into SAC using the READBBF command. This feature lets you save information from one execution of SAC to another. You can also add coding to your own programs to access the information in these blackboard variable files. This lets you transfer information between your own programs and SAC. See the SAC Subroutines Reference Manual for details.

SEE COMMANDS

READBBF, SETBB, GETBB

LATEST REVISION

May 15, 1987 (Version 10.2)

WRITECSS

SUMMARY

Writes data in memory to disk in CSS 3.0 format.

SYNTAX

```
WRITE {BINARY|ASCII} {COMMIT|ROLLBACK|RECALLTRACE}
{DIR ON|OFF|CURRENT|name} name
```

INPUT

ASCII: (Default) Write standard ASCII flatfiles.

BINARY: Write output as a single CSS 3.0 binary file.

COMMIT: The COMMIT option commits headers and waveforms in SAC memory prior to writing the traces. COMMIT is the default.

ROLLBACK: The ROLLBACK option reverts to the last committed version of the header and waveform before writing the traces.

RECALLTRACE: The RECALLTRACE option:

- reverts to the last committed version of the waveform,
- reverts to the last committed version of those header variables closely linked to the waveform,
- commits those header variables which are loosely linked to the waveform before writing the traces. (use HELP RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

DIR ON: Turn directory option on but don't change name of write directory.

DIR OFF: Turn directory option off. When off, writes to current directory.

DIR CURRENT: Turn directory option on and set name of write directory to the "current directory" (e.g. the directory from which you started SAC.)

DIR name: Turn directory option on and set name of write directory to name. Write all files to the directory called name. This may be a relative or absolute directory name.

name: Write filelist in set to name. There should be only one name specified. It may not contain wildcards. For ASCII output, name will be prepended to the table name for each flatfile. (i.e. name.wfdisc, name.origin, ...). In BINARY mode, name is the output file name.

DEFAULT VALUES

```
WRITECSS ASCII COMMIT DIR OFF
```

DESCRIPTION

This command allows you, at any point in the processing of data, to save the results to disk in CSS 3.0 format. In ASCII mode (default) one or more ASCII flatfiles are written. The exact files written will depend upon the source of the data but can be any of:

wfdisc, wftag, origin, arrival, assoc, sitechan, site, affiliation, origerr, origin, event, sensor, instrument, gregion, stassoc, remark sacdata.

In Binary mode a single file will be written containing the same set of tables as would be written in ASCII mode, but with all tables in binary format and with the waveform data embedded in the file.

For more information on the CSS format see the "Center for Seismic Studies Version 3 Database: Schema Reference Manual".

ERROR MESSAGES

- 1301: No data files read in.
- 1311: No list of filenames to write.
- 1312: Bad number of files in write file list

SEE COMMANDS

READ, READCSS, WRITE, COMMIT, ROLLBACK, RECALLTRACE

LATEST REVISION

October 27, 1998 (Version 00.58)

WRITEGSE

SUMMARY

Write data files in GSE 2.0 format from memory to disk.

SYNTAX

```
WRITEGSE {TYPE} {SOURCE ON|OFF|str} {COMMIT|ROLLBACK|RECALLTRACE} {DIR
name} filename
```

INPUT

TYPE: Determines wether the data are written in ascii integer format (INT) or as compressed gse format (CM6). Default is INT.

SOURCE str: str is a string 20 characters or less specifying the institution at which the GSE file was written. str is written in the MSG_ID line of the resultant GSE file.

COMMIT: Commits headers and waveforms in SAC memory -- removing any previous versions of headers or waveforms from RAM -- prior to reading more files. COMMIT is the default.

ROLLBACK: Reverts to the last committed version of the header and waveform before reading more files.

RECALLTRACE:
- reverts to the last committed version of the waveform,
- reverts to the last committed version of those header variables closely linked to the waveform,
- commits those header variables which are loosely linked to the waveform. (see RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

DIR name: The directory in which to write the gsefile. This directory name is the same one that is used in WRITE command.

filename: The name of the gse file to be written.

DEFAULT VALUES

```
WRITEGSE INT SOURCE OFF COMMIT
```

DESCRIPTION

Writes all data in memory to a single file according to the GSE 2.0 data format

The following GSE Data messages are written:
- WAVEFORM
- STATION
- CHANNEL
- ARRIVAL
- ORIGIN

Waveforms are written in INT format: floating point data is truncated to the nearest integer.

Note There is no way in GSE 2.0 to associate ORIGIN data with a waveform, so SAC's READGSE command does not read ORIGIN data, but WRITEGSE writes it.

Note SAC does not currently read nor write DETECTIONS information. Therefore, ARRIVAL information is not associated with specific channels.

LATEST REVISION

April 22, 1999 (Version 00.58)

WRITEHDR

SUMMARY

Overwrites the headers on disk with those in memory.

SYNTAX

```
WRITEHDR {COMMIT|ROLLBACK|RECALLTRACE}
```

INPUT

COMMIT: Commits headers and waveforms in SAC memory -- removing any previous versions of headers or waveforms from RAM -- prior to writing files. COMMIT is the default.

ROLLBACK: reverts to the last committed version of the header and waveform before writing files.

RECALLTRACE:
- reverts to the last committed version of the waveform,
- reverts to the last committed version of those header variables closely linked to the waveform,
- commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

DEFAULT VALUES

```
WRITEHDR COMMIT
```

DESCRIPTION

The data on disk is NOT overwritten by this command. Use the WRITE OVER command to overwrite headers and data. The WRITEHDR command should NEVER be used if the CUT option is on. The header in memory is modified to reflect the effects of the CUT, but the data on disk is not modified. Use of the WRITEHDR command on cut data files will have the effect of apparently shifting and truncating the data on disk in time.

ERROR MESSAGES

- 1301: No data files read in.

HEADER CHANGES

Updates headers on disk.

LIMITATIONS

See description above about use of CUT and WRITEHDR.

SEE COMMANDS

CUT, WRITE, COMMIT, ROLLBACK, RECALLTRACE

LATEST REVISION

Oct. 27, 1998 (Version 0.58)

WRITESDD

SUMMARY

Writes data in memory to disk in SDD format.

SYNTAX

```
WRITESDD {options} {namingoptions}
```

where options are one or more of the following:

```
DIR ON|OFF|CURRENT|name
```

These options MUST preceed any element in the namingoptions:

```
OVER
APPEND text
PREPEND text
DELETE text
CHANGE text1 text2
filelist
```

Only one of these namingoptions is allowed at a time.

INPUT

DIR ON: Turn directory option on but don't change name of write directory.

DIR OFF: Turn directory option off. When off, write files to current directory.

DIR CURRENT: Turn directory option on and set name of write directory to the "current directory" (e.g. the directory from which you started SAC.)

DIR name: Turn directory option on and set name of write directory to name. Write all filenames to the directory called name. This may be a relative or absolute directory name.

OVER: Use current read filelist as write filelist. Overwrite files on disk with data in memory.

APPEND text: Write filelist is created by appending text to each name in the current read filelist.

PREPEND text: Write filelist is created by prepending text to each name in the current read filelist.

DELETE text: Write filelist is created by deleting the first occurrence of text in each name in the current read filelist.

CHANGE text1 text2: Write filelist is created by changing the first occurrence of text1 in each name in the current read filelist to text2.

filelist: Write filelist is set to filelist. This list may contain simple filenames, relative pathnames, or full pathnames. IT MAY NOT CONTAIN WILDCARDS.

LATEST REVISION

September 08, 1990 (Version 10.6)

WRITESP

SUMMARY

Writes spectral files to disk as "normal" data files.

SYNTAX

WRITESP {type} {COMMIT|ROLLBACK|RECALLTRACE} {OVER|filelist} where type is one of the following:

ASIS|RLIM|AMPH|RL|IM|AM|PH

INPUT

ASIS: Write components in their present format.

RLIM: Write real and imaginary components.

AMPH: Write amplitude and phase components.

RL: Write real component only.

IM: Write imaginary component only.

AM: Write amplitude component only.

PH: Write phase component only.

COMMIT: Commits headers and waveforms in SAC memory -- removing any previous versions of headers or waveforms from RAM -- prior to writing files. COMMIT is the default.

ROLLBACK: reverts to the last committed version of the header and waveform before writing files.

RECALLTRACE:
- reverts to the last committed version of the waveform,
- reverts to the last committed version of those header variables closely linked to the waveform,
- commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

filelis: A list of SAC binary data files. This list may contain simple filenames and full or relative pathnames.

DEFAULT VALUES

```
WRITESP ASIS COMMIT
```

DESCRIPTION

SAC data files may contain either time-series data or spectral data. Certain fields in the header distinguish between the two formats. When you read (see READ) a time-series file into memory, take the fast fourier transform (see FFT), and write the data to disk (see WRITE), then the data on disk will be in the spectral format.

Certain operations can only be performed on time-series data and certain operations only on spectral data. For example, you can't apply a taper to spectral data files or multiply two spectral files together. This is a protection mechanism built into SAC.

Sometimes, however, you may need to perform some of these operations on spectral data. To override SAC's protection mechanism, you can use this command to write spectral data to disk as time-series data. Each component is written as a separate data file. You may then read these files back into SAC and perform any operation that you wish, since SAC thinks they are time series data files. Once these calculations are completed, you may write the modified data back to disk using the WRITE command. If you wish to reconstruct the spectral data file, use the READSP command. To help you keep track of the data on disk, SAC appends a suffix to the filename you request that identifies the spectral component stored in that file. The suffixes are ".RL", ".IM", ".AM", and ".PH"" for the real component, imaginary component, amplitude, and phase respectively.

EXAMPLES

Assume that you want to perform some operations on the spectral amplitude of FILE1:

```
u:  READ FILE1
u:  FFT AMPH
u:  WRITESP OVER
```

SAC will then write out two files, FILE1.AM and FILE1.PH. Now you perform the operations on the amplitude file:

```
u:  READ FILE1.AM
u:  ...perform operations.
u:  WRITE OVER
```

Now the files on disk represent the modified spectral data. If you wanted to reconstruct the SAC spectral data file and take the inverse transform:

```
u:  READSP FILE1
u:  IFFT
u:  WRITE FILE2
```

Note for examples of the behavior of COMMIT, ROLLBACK, and RECALLTRACE, see the commands of the same name.

ERROR MESSAGES

- 1301: No data files read in.
- 1305: Illegal operation on time series file

HEADER CHANGES

B, E, and DELTA for the files on disk will contain the beginning, ending, and incremental frequency in Hz.

SEE COMMANDS

READSP, COMMIT, ROLLBACK, RECALLTRACE

LATEST REVISION

Oct. 27, 1998 (Version 0.58)

XDIV

SUMMARY

Controls the x axis division spacing.

SYNTAX

```
XDIV {NICE|INCREMENT v|NUMBER n},{POWER {ON/OFF}}
```

INPUT

NICE: Use "nice-numbered" division spacings.

INCREMENT v: Set division spacing increment to v.

NUMBER n: Set number of division spacings to n.

POWER {ON}: Turn power option on. When this option is on, SAC may print the division spacings as a number raised to a power of 10.

POWER OFF: Turn power option off.

DEFAULT VALUES

```
XDIV NICE POWER ON
```

DESCRIPTION

This command controls the selection of x axis division spacings. Most of the time the default "nice-numbered" spacings are satisfactory. SAC determines these based on the minimum and maximum axis limits, the length of the axis, and the current axis character size. You may also force the division spacing to be a certain value by use of the INCREMENT option or you may set the number of division spacings by use of the NUMBER option.

LATEST REVISION

October 11, 1984 (Version 9.1)

XFUDGE

SUMMARY

Changes the x axis "fudge factor."

SYNTAX

```
XFUDGE {ON|OFF|v}
```

INPUT

{ON}: Turn fudge option on but don't change fudge factor.
OFF: Turn fudge option off.
v: Turn fudge option on and change fudge factor to v.

DEFAULT VALUES

```
XFUDGE 0.03
```

DESCRIPTION

When this option is on, the actual axis limits are changed by a "fudge factor". The algorithm for a linearly interpolated axis is:

```
XDIFF=XFUDGE*(XMAX-XMIN)
XMIN=XMIN-XDIFF
XMAX=XMAX+XDIFF
```

where XMIN and XMAX are the data extrema and XFUDGE is the fudge factor. The algorithm is similiar for logarithmically interpolated axes. The fudge option only applies when the axis limits are scaled to the data extrema (see XLIM.)

SEE COMMANDS

XLIM

LATEST REVISION

January 8, 1983 (Version 8.0)

XFULL

SUMMARY

Controls plotting of x axis full logarithmic decades.

SYNTAX

```
XFULL {ON|OFF}
```

INPUT

{ON}: Turn full decade plotting on.
OFF: Turn full decade plotting off.

DEFAULT VALUES

```
XFULL ON
```

DESCRIPTION

Full decade plotting applies only when logarithmic interpolation is being used and when the fixed limits option is off (see XLIM.) When on, the actual axis limits will be set to the first full decade before and after the data limits. When off, the actual data limits will be used.

SEE COMMANDS

XLIM

LATEST REVISION

January 8, 1983 (Version 8.0)

XGRID

SUMMARY

Controls plotting of grid lines in the x direction.

SYNTAX

```
XGRID {ON|OFF|SOLID|DOTTED}
```

INPUT

{ON}: Turn x axis grid plotting on but don't change grid type.
OFF: Turn x axis grid plotting off.
SOLID: Turn x axis grid plotting on using solid grid lines.
DOTTED: Turn x axis grid plotting on using dotted grid lines.

DEFAULT VALUES

```
XGRID OFF
```

DESCRIPTION

This command controls only x grid lines. The GRID command can be used to control grid lines in both directions.

SEE COMMANDS

GRID

LATEST REVISION

January 8, 1983 (Version 8.0)

XLABEL

SUMMARY

Defines the x axis label and attributes.

SYNTAX

XLABEL {ON|OFF|text},{LOCATION location},{SIZE size} where location is one of the following:

TOP|BOTTOM|RIGHT|LEFT

and where size is one of the following:

TINY|SMALL|MEDIUM|LARGE

INPUT

{ON}: Turn x axis labeling option on. Don't change text.
OFF: Turn x axis labeling option off.
text: Turn x axis labeling option on. Change text of label. If text contains embedded blanks, it must be enclosed in single quotes.
LOCATION location: Change location of x axis label.
TOP: Top of the plot window.
BOTTOM: Bottom of the plot window.
RIGHT: To the right of the plot window.
LEFT: To the left of the plot window.
SIZE size: Change x axis label text size.
TINY: Tiny text size has 132 characters per line.
SMALL: Small text size has 100 characters per line.
MEDIUM: Medium text size has 80 characters per line.
LARGE: Large text size has 50 characters per line.

DEFAULT VALUES

```
XLABEL OFF LOCATION BOTTOM SIZE SMALL;
```

DESCRIPTION

If this option is on, an x axis label is placed on each plot. The size and location of the x axis label can be changed as well as the text of the x axis label itself. The text quality and font used can be set using the GTEXT command.

SEE COMMANDS

GTEXT

LATEST REVISION

January 8, 1983 (Version 8.0)

XLIM

SUMMARY

Determines the plot limits for the x axis.

SYNTAX

```
XLIM {ON|OFF|pdw|SIGNAL}
```

INPUT

{ON}: Turn x limits on but don't change limits.

OFF: Turn x limits off.

pdw: Turn x limits on and set limits to a new "partial data window." A pdw consists of a starting and a stopping value of the independent variable, usually time, which defines the desired window of data that you wish to plot. See the CUT command for a complete explanation of how to define and use a pdw. Some examples are given below.

SIGNAL: Equivalent to typing: A -1 F +1.

DEFAULT VALUES

```
XLIM OFF
```

DESCRIPTION

When this option is on, fixed plot limits are used for the x axis. When this option is off, the limits are scaled to the data. Fixed x limits can be used to "blowup" part of the data currently in memory.

EXAMPLES

In these examples we assume time is the independent variable and seconds are the units.:

```
B 0 30: First 30 secs of the file.
A -10 30: From 10 secs before to 30 secs after first arrival.
T3 -1 T7: From 1 sec before T3 time pick to T7 time pick.
B N 2048: First 2048 points of file.
30.2 48: 30.2 to 48 secs relative to file zero.
```

SEE COMMANDS

CUT

LATEST REVISION

January 8, 1983 (Version 8.0)

XLIN

SUMMARY

Turns on linear scaling for the x axis.

SYNTAX

```
XLIN
```

DEFAULT VALUES

```
Linear scaling.
```

LATEST REVISION

January 8, 1983 (Version 8.0)

XLOG

SUMMARY

Turns on logarithimic scaling for the x axis.

SYNTAX

```
XLOG
```

DEFAULT VALUES

```
Linear scaling.
```

LATEST REVISION

January 8, 1983 (Version 8.0)

XVPORT

SUMMARY

Defines the viewport for the x axis.

SYNTAX

```
XVPORT xvmin xvmax
```

INPUT

xvmin: X axis viewport minimum value. Must be in the range 0.0 to xvmax.
xvmax: X axis viewport maximum. Must be in the range xvmin to 1.0.

DEFAULT VALUES

```
XVPORT 0.1 0.9
```

DESCRIPTION

The viewport is the portion of the viewspace (see VSPACE command) in which the actual plot is drawn. The coordinate system used to define the viewspace and viewport is called a virtual coordinate system. A virtual coordinate system does not depend upon the size, shape, or resolution of a particular physical device's display surface. SAC's coordinate system runs from 0.0 to 1.0 in both the x and y directions. The lower left hand corner of the viewspace is the point (0.0, 0.0) and the upper right hand corner of the viewspace is the point (1.0, 1.0). (See the figure on the next page.) The use of this coordinate system lets you position a plot without worrying about a specific output device. The XVPORT and YVPORT commands control where in the viewspace a specific plot is to be drawn. The default values use most of the viewspace for the plot while leaving some room on each side for axes, labels, and a title. You can place a particular plot anywhere you want using these commands. When used in conjunction with the BEGINFRAME and ENDFRAME commands, these commands let you create your own special layout by putting several different plots on the same frame.

EXAMPLES

See the example in the BEGINFRAME documentation.

SEE COMMANDS

VSPACE, BEGINFRAME Principles of Interactive Computer Graphics, Second Edition; William M. Newman and Robert F. Sproull; 1979; McGraw-Hill.

LATEST REVISION

January 8, 1983 (Version 8.0) Viewspace and Viewport Coordinates

YDIV

SUMMARY

Controls the y axis division spacing.

SYNTAX

```
YDIV {NICE|INCREMENT v|NUMBER n},{POWER {ON/OFF}}
```

INPUT

NICE: Use "nice-numbered" division spacings.

INCREMENT v: Set division spacing increment to v.

NUMBER n: Set number of division spacings to n.

POWER {ON}: Turn power option on. When this option is on, SAC may print the division spacings as a number raised to a power of 10.

POWER OFF: Turn power option off.

DEFAULT VALUES

```
YDIV NICE POWER ON
```

DESCRIPTION

This command controls the selection of y axis division spacings. Most of the time the default "nice-numbered" spacings are satisfactory. SAC determines these based on the minimum and maximum axis limits, the length of the axis, and the current axis character size. You may also force the division spacing to be a certain value by use of the INCREMENT option or you may set the number of division spacings by use of the NUMBER option.

LATEST REVISION

October 11, 1984 (Version 9.1)

YFUDGE

SUMMARY

Changes the y axis "fudge factor."

SYNTAX

```
YFUDGE ON|OFF|v
```

INPUT

ON: Turn fudge option on but don't change fudge factor.
OFF: Turn fudge option off.
v: Turn fudge option on and change fudge factor to v.

DEFAULT VALUES

```
YFUDGE 0.03
```

DESCRIPTION

When this option is on, the actual axis limits are changed by a "fudge factor". The algorithm for a linearly scaled axis is:

```
YDIFF=YFUDGE*(YMAX-YMIN)
YMIN=YMIN-YDIFF
YMAX=YMAX+YDIFF
```

where YMIN and YMAX are the data extrema and YFUDGE is the fudge factor. The algorithm is similiar for logarithmically scaled axes. The fudge option only applies when the axis limits are scaled to the data extrema (see YLIM.)

SEE COMMANDS

YLIM

LATEST REVISION

January 8, 1983 (Version 8.0)

YFULL

SUMMARY

Controls plotting of y axis full logarithmic decades.

SYNTAX

```
YFULL {ON|OFF}
```

INPUT

{ON}: Turn full decade plotting on.
OFF: Turn full decade plotting off.

DEFAULT VALUES

```
YFULL ON
```

DESCRIPTION

Full decade plotting applies only when logarithmic scaling is being used and when the fixed limits option is off (see YLIM.) When on, the actual axis limits will be set to the first full decade before and after the data limits. When off, the actual data limits will be used.

SEE COMMANDS

YLIM

LATEST REVISION

January 8, 1983 (Version 8.0)

YGRID

SUMMARY

Controls plotting of grid lines in the y direction.

SYNTAX

```
YGRID {ON|OFF|SOLID|DOTTED}
```

INPUT

{ON}: Turn y axis grid plotting on but don't change grid type.
OFF: Turn y axis grid plotting off.
SOLID: Turn y axis grid plotting on using solid grid lines.
DOTTED: Turn y axis grid plotting on using dotted grid lines.

DEFAULT VALUES

```
YGRID OFF
```

DESCRIPTION

This command controls only y grid lines. The GRID command can be used to control grid lines in both directions.

SEE COMMANDS

GRID

LATEST REVISION

January 8, 1983 (Version 8.0)

YLABEL

SUMMARY

Defines the y axis label and attributes.

SYNTAX

YLABEL {ON|OFF|text},{LOCATION location},{SIZE size} where location is one of the following:

TOP|BOTTOM|RIGHT|LEFT

and where size is one of the following:

TINY|SMALL|MEDIUM|LARGE

INPUT

{ON}: Turn y axis labeling option on. Don't change text.
OFF: Turn y axis labeling option off.
text: Turn y axis labeling option on. Change text of label. If text contains embedded blanks, it must be enclosed in single quotes.
LOCATION location: Change location of y axis label.
TOP: Top of the plot window.
BOTTOM: Bottom of the plot window.
RIGHT: To the right of the plot window.
LEFT: To the left of the plot window.
SIZE size: Change y axis label text size.
TINY: Tiny text size has 132 characters per line.
SMALL: Small text size has 100 characters per line.
MEDIUM: Medium text size has 80 characters per line.
LARGE: Large text size has 50 characters per line.

DEFAULT VALUES

```
YLABEL OFF LOCATION LEFT SIZE SMALL;
```

DESCRIPTION

If this option is on, a y axis label is placed on each plot. The size and location of the y axis label can be changed as well as the text of the y axis label itself. The text quality and font used can be set using the GTEXT command.

SEE COMMANDS

GTEXT

LATEST REVISION

January 8, 1983 (Version 8.0)

YLIM

SUMMARY

Determines the plot limits for the y axis.

SYNTAX

```
YLIM {ON|OFF|ALL|min max|PM v ....}
```

INPUT

{ON}: Turn y limits option on, but don't change limits.

OFF: Turn y limits option off.

ALL: Scale y limits to the minimum and maximum of all files in memory.

min max: Turn fixed y option on and change limits to min and max.

PM v: Turn fixed y option on and change limits to minus and plus the absolute value of v. ,SKIP You may define different y limit options for each file in memory if you wish. The first entry in the command applies to the first file in memory, the second entry to the second file, etc. The last entry applies to the remainder of the files in memory.

DEFAULT VALUES

```
YLIM OFF
```

DESCRIPTION

When this option is on, fixed limits are used in plotting. When off, the limits are scaled to the data. The limits can also be scaled to the entire data set if desired. Different values may be set for each file in memory.

EXAMPLES

Consider the following set of commands:

```
u:  YLIM 0.0 30.0 ALL OFF
u:  READ FILE1 FILE2 FILE3
u:  PLOT
```

FILE1 would be plotted with y limits of 0.0 and 30. FILE2 would be scaled to the minimum and maximum values of all files in memory. FILE3 would be scaled to its own minimum and maximum values. If more than three files were read in, they would also be scaled to their own minimum and maximum values.

LATEST REVISION

January 8, 1983 (Version 8.0)

YLIN

SUMMARY

Turns on linear scaling for the y axis.

SYNTAX

```
YLIN
```

DEFAULT VALUES

```
Linear scaling.
```

LATEST REVISION

January 8, 1983 (Version 8.0)

YLOG

SUMMARY

Turns on logarithimic scaling for the y axis.

SYNTAX

```
YLOG
```

DEFAULT VALUES

```
Linear scaling.
```

LATEST REVISION

January 8, 1983 (Version 8.0)

YVPORT

SUMMARY

Defines the viewport for the y axis.

SYNTAX

```
YVPORT yvmin yvmax
```

INPUT

yvmin: Y axis viewport minimum value. Must be in the range 0.0 to yvmax.
yvmax: Y axis viewport maximum. Must be in the range yvmin to 1.0.

DEFAULT VALUES

```
YVPORT 0.15 0.9
```

DESCRIPTION

The viewport is the portion of the viewspace (see VSPACE command) in which the actual plot is drawn. The coordinate system used to define the viewspace and viewport is called a virtual coordinate system. A virtual coordinate system does not depend upon the size, shape, or resolution of a particular physical device's display surface. SAC's coordinate system runs from 0.0 to 1.0 in both the x and y directions. The lower left hand corner of the viewspace is the point (0.0, 0.0) and the upper right hand corner of the viewspace is the point (1.0, 1.0). (See the figure in the XVPORT documentation.) The use of this coordinate system lets you position a plot without worrying about a specific output device. The XVPORT and YVPORT commands control where in the viewspace a specific plot is to be drawn. The default values use most of the viewspace for the plot while leaving some room on each side for axes, labels, and a title. You can place a particular plot anywhere you want using these commands. When used in conjunction with the BEGINFRAME and ENDFRAME commands, these commands let you create your own special layout by putting several different plots on the same frame.

EXAMPLES

See the example in the BEGINFRAME documentation.

SEE COMMANDS

VSPACE, XVPORT, BEGINFRAME Principles of Interactive Computer Graphics, Second Edition; William M. Newman and Robert F. Sproull; 1979; McGraw-Hill.

LATEST REVISION

January 8, 1983 (Version 8.0)

ZCOLORS

SUMMARY

Controls the color display of contour lines.

SYNTAX

ZCOLORS {ON|OFF} {options} where options is currently limited to:

```
LIST c1 c2 ...  cn
More options will be added in the future.
```

INPUT

ON: Turn color display of contour lines on.

OFF: Turn color display of contour lines off.

LIST c1 c2 . cn: Set the list of contour color names to use. Each entry in this list is used for the corresponding contour level. If the number of contour levels is larger than the length of this list, the entire list is repeated.

cn: The name of a color from SAC's current color table.

DEFAULT VALUES

```
ZCOLORS OFF LIST RED GREEN BLUE
```

SEE COMMANDS

CONTOUR, COLOR

LATEST REVISION

April 30, 1990 (Version 10.5b)

ZLABELS

SUMMARY

Controls the labeling of contour lines with contour level values.

SYNTAX

```
ZLABELS  {ON|OFF}  {options}
```

where options are one or more of the following:

```
SPACING v1 {v2 {v3} }
SIZE v
ANGLE v
LIST c1 c2 ...  cn
```

SPECIAL NOTE The LIST option must appear last in this command.

INPUT

ON: Turn labeling of contour lines on.

OFF: Turn labeling of contour lines off.

SPACING v1 {v2 {v3} }: Set the minimum, optimum, and maximum spacing between adjacent labels (in viewport coordinates) to v1, v2, and v3 respectively. If the second or third values are omitted, the previous values are used.

SIZE v: Set the size (height) of the labels (in viewport coordinates) to v.

ANGLE v: Set the desired maximum text angle the labels (in degrees from horizontal) to v.

LIST c1 c2 . cn: Set the list of contour labels to use. Each entry in this list is used for the corresponding contour level. If the number of contour levels is larger than the length of this list, the entire list is repeated.

cn: ON|OFF|INT|FLOATn|EXPn|text

ON: Place a label on corresponding contour line. Use Fortran's free format capabilities to format the label from the contour level value.

OFF: Do not place a label on corresponding contour line.

INT: Place an integer label on corresponding contour line.

FLOATn: Place a floating point label on corresponding contour line with n values to the right of the decimal point. If n is omitted, the previous value for n is used.

EXPn: Place an exponentially formatted label on corresponding contour line with n values to the right of the decimal point. If n is omitted, the previous value for n is used.

text: Use text to label the corresponding contour line.

DEFAULT VALUES

```
ZLABELS  OFF  SPACING 0.1 0.2 0.3  SIZE  0.0075 ANGLE 45.0  LIST ON
```

EXAMPLES

See CONTOUR for examples of the use of ZLABELS.

SEE COMMANDS

CONTOUR

LATEST REVISION

April 30, 1990 (Version 10.5b)

ZLEVELS

SUMMARY

Controls the contour line spacing in subsequent contour plots.

SYNTAX

```
ZLEVELS {options}
```

where options are one or more of the following:

```
SCALE
RANGE v1 v2
INCREMENT v
NUMBER n
LIST v1 v2 ...  vn
```

INPUT

SCALE: Scale the range of the contour levels to the data.

RANGE v1 v2: Set the range (minimum and maximum) of the contour levels to v1 and v2. You should use either the SCALE or the RANGE option but not both.

INCREMENT v: Set the increment between contour levels to v.

NUMBER n: Set the number of contour levels to n. You should use either the INCREMENT or the NUMBER option but not both.

LIST v1 v2 . vn: Set the list of contour levels to v1, v2, etc. All other options are ignored if you use this one.

DEFAULT VALUES

```
ZLEVELS SCALE NUMBER 20
```

EXAMPLES

See CONTOUR for examples of the use of ZLEVELS.

LIMITATIONS

The maximum number of contour levels is 40.

SEE COMMANDS

CONTOUR

LATEST REVISION

April 30, 1990 (Version 10.5b)

ZLINES

SUMMARY

Controls the contour linestyles in subsequent contour plots.

SYNTAX

```
ZLINES  {ON|OFF}  {options}
```

where options are one or more of the following:

```
LIST n1 n2 ...  nn
REGIONS v1 v2 ...  vn
```

INPUT

ON: Turn display of contour lines on.

OFF: Turn display of contour lines off.

LIST n1 n2 . nn: Set list of linestyles to use. Each entry in this list is used for the corresponding contour level. If the number of contour levels is larger than the number of entries in the list, the entire list is repeated.

REGIONS v1 v2 . vn: Set list of contour regions. The length of this list should be one less than the linestyle list. Contour levels less than a contour region value are assigned the linestyle of the corresponding entry in the linestyle list. Contour levels above the last contour region value are assigned the value of the last entry in the linestyle list.

DEFAULT VALUES

```
ZLINES ON LIST 1
```

EXAMPLES

To set up contours which cycle between four different linestyles:

```
u: ZLINES LIST 1 2 3 4
```

To set contours with dotted lines representing levels below 0.0 and solid lines representing contours above 0.0:

```
u: ZLINES LIST 2 1 REGIONS 0.0
```

See CONTOUR for more examples of the use of ZLINES.

SEE COMMANDS

CONTOUR

LATEST REVISION

April 30, 1990 (Version 10.5b)

ZTICKS

SUMMARY

Controls the labeling of contour lines with directional tick marks.

SYNTAX

ZTICKS {ON|OFF} {options} where options are one or more of the following:

```
SPACING v
LENGTH v
DIRECTION DOWN|UP
LIST c1 c2 ... cn
```

INPUT

ON: Turn tick mark labeling of contour lines on.

OFF: Turn tick mark labeling of contour lines off.

SPACING v: Set the spacing between adjacent tick marks (in viewport coordinates) on each line segment to v.

LENGTH v: Set the length of each tick mark (in viewport coordinates) to v.

DIRECTION DOWN: Tick marks point in the direction of decreasing z value.

DIRECTION UP: Tick marks point in the direction of increasing z value.

LIST c1 c2 . cn: Set the list of contour ticks marks to use. Each entry in this list is used for the corresponding contour level. If the number of contour levels is larger than the length of this list, the entire list is repeated. A value of ON means that tick marks are placed on that contour line. A value of OFF means that no tick marks are placed on that contour line. line.

DEFAULT VALUES

```
ZTICKS  OFF  SPACING 0.1  LENGTH 0.005  DIRECTION DOWN  LIST ON
```

EXAMPLES

See CONTOUR for examples of the use of ZTICKS.

SEE COMMANDS

CONTOUR

LATEST REVISION

April 30, 1990 (Version 10.5b)

3 Signal-Stacking Subprocess

Signal Stacking Subprocess

Introduction

A subprocess is like a small program within the main SAC program. You start a subprocess by typing its name (SSS in this case.) You can terminate it and return to the main program using the quitsub command. You can also terminate SAC from within a subprocess using the QUIT command.

While within a subprocess, you can execute any command belonging to that subprocess plus a limited number of main SAC commands.

SSS is a package for doing signal stacking (i.e. summation or beamforming).

Each signal (i.e. SAC file) has properties such as a static delay, epicentral distance, weighting factor, and polarity associated with it. The dynamic delays can be calculated using a normal moveout or refracted wave velocity model.

Certain delay properties can be automatically incremented between summations. Files are easily added to or removed from the stack file list. The time window for the stack is easily adjusted. Files which do not contain data throughout the stack time window are filled with zeros.

The stack file list can be plotted with or without the summation. Each summation can be saved on disk for later use. A record section plot is also included in this subprocess.

The SS commands are listed below in alphabetical order. A list of the allowed main SAC commands is also shown. You can also use all of the SAC macro features in this subprocess.

SSS Commands

ADDSTACK: Add a new file to the stack file list.

CHANGESTACK: Change properties of files currently in the stack file list.

DELETESTACK: Deletes one or more files from the stack file list.

DELTACHECK: Change the sampling rate checking option.

DISTANCEAXIS: Define the record section plot distance axis parameters.

DISTANCEWINDOW: Controls distance window properties on subsequent record section plots.

GLOBALSTACK: Sets global stack properties.

INCREMENTSTACK: Increments properties for files in the stack file list.

LISTSTACK: Lists the properties of the files in the stack file list.

PLOTRECORDSECTION: Plots a record section of the files in the stack file list.

PLOTSTACK: Plots the files in the stack file list.

QUITSUB: Terminates the Signal Stacking Subprocess.

SUMSTACK: Sums the files in the stack file list.

TIMEAXIS: Controls the time axis properties on subsequent record section plots.

TIMEWINDOW: Sets the time window limits for subsequent stack summation.

SSSTRAVELTIME: Function traveltime called from within SSS.Computes traveltime curves for predefined models

VELOCITYMODEL: Sets stack velocity model parameters for computing dynamic delays.

VELOCITYROSET: Controls placement of a velocity roset on subsequent record section plots.

WRITESTACK: Writes a stack summation to disk.

ZEROSTACK: Zeros or reinitializes the signal stack.

Main SAC Commands

This is a list of the allowed main SAC commands. Their abbreviated names are also allowed.

AXES	BEGINDEVICES	BEGINFRAME
BEGINWINDOW	BORDER	COLOR
COMCOR	COPYHDR	DATAGEN
ECHO	ENDDEVICES	ENDFRAME
ERASE	EVALUATE	FLOOR
GETBB	GRID	GTEXT
HELP	INSTALLMACRO	LISTHDR
LINE	LINLIN	LINLOG
LOGLAB	LOGLIN	LOGLOG
MACRO	MESSAGE	PAUSE
PLABEL	PLOTC	QDP
QUIT	READBBF	REPORT
SETBB	SETDATADIR	SETDEVICE
SETMACRO	SGF	SYMBOL
SYNTAX	SYSTEMCOMMAND	TICKS
TITLE	TSIZE	VSPACE
WAIT	WINDOW	WRITEBBF
XDIV	XFUDGE	XFULL
XGRID	XLABEL	XLIM
XLIN	XLOG	XVPORT
YDIV	YFUDGE	YFULL
YGRID	YLABEL	YLIM
YLIN	YLOG	YVPORT

ADDSTACK

SUMMARY

Add a new file to the stack file list.

SYNTAX

```
[A]DD[S]TACK filename [property ...]
```

where property is one or more of the following:

TRUST ON|OFF:
[W]EIGHT v:
[DI]STANCE v:
[BE]GINTIME v:
[END]TIME v:
[DE]LAY v [[S]ECONDS|[P]OINTS]:
[I]NCREMENT v [[S]ECONDS|[P]OINTS]:
[N]ORMAL:
[R]EVERSE:

INPUT

filename: Name of the file to be added to the stack file list.

TRUST ON|OFF: This option is used to resolve an ambiguity in converting files from SAC to CSS format. When converting the data, matching event IDs could mean the files have identical event information, or they could be an artifact of the merging of these two very different formats.
When TRUST is ON, SAC is more likely to accept matching event IDs as identical event information than when TRUST is OFF, depending on the history of READ commands associated with the current data files in memory.

[W]EIGHT v: Weighting factor for this file in the range zero to one. Each data point is multiplied by this value when the stack is summed.

[DI]STANCE v: Station to epicenter distance in kilometers for this file. This is used to calculate dynamic time delays.

[BE]GINTIME v: Time of the begining of the event.

[END]TIME v: Time of the end of the event.

[DE]LAY v [[S]ECONDS|[P]OINTS]: Static time delay to apply to file. This delay is in either seconds or number of data points.

[I]NCREMENT v [[S]ECONDS|[P]OINTS]: Static time delay increment for this file. This increment is in either seconds or number of data points. The static time delay is incremented by this ammount each time the incrementstack command is executed.

[N]ORMAL: File has normal polarity.

[R]EVERSED: File has reversed polarity. (Each data point in the signal is multiplied by -1.0 when the stack is summed.)

DEFAULTS VALUES

Each file is given the global property value if no local one is entered. The default units for the DELAY and INCREMENT options is SECONDS.

DESCRIPTION

There are seven properties associated with each stack list file. They are

- the weighting factor.
- the station to epicenter distance.
- the begin time of the event.
- the end time of the event.
- the static time delay in either seconds or number of data points.
- the static time delay increment in either seconds or number of data points
- the polarity of the file, either normal or reversed.

There is a global value associated with each of these properties. They are defined by the globalstack command. When a file is added to the stack file list, that file's properties will be set to the global value if no local value is given. The changestack command can be used to change a file's properties after it has been added to the stack file list.

All commands which load data into memory have are now monitored to maintain a level of confidence in the event information when moved from the SAC data buffer to the CSS data buffer. For ADDSTACK, when the confidence is HIGH that all the data files are cosistent in the numbering of event IDs, matching event IDs are treated as having identical event information. When the confidence is LOW in ADDSTACK, matching event IDs are understood as an artifact, and new event IDs are generated for the incoming file. For more details use HELP READ.

EXAMPLES

The following examples illustrate several of the features of the Signal Stacking Module. Suppose you entered the following set of commands:

```
u:  GLOBALSTACK DELAY 1.0 INCREMENT 0.03
:u:  ADDSTACK FILEA DELAY 2.0
:u:  ADDSTACK FILEB DELAY 3.0 INCREMENT 0.01 REVERSED
:u:  ADDSTACK FILEC
:u:  ADDSTACK FILED WEIGHT 0.5
```

```
The first command changes the global property values for time delay and
time delay increment.
The other global properties have their default values.
FILEA's properties would be the global ones except for the time delay.
FILEB's properties would be the global ones except for the time delay,
the time delay increment, and the signal polarity.
FILEC's properties would be the same as the global ones.
FILED's properties would be the global ones except for the weighting factor.
Now you enter::
```

```
    u:  SUMSTACK
```

```
The summation is done on the four files in the stack file list::

   FILEA, FILEB, FILEC, and FILED.

The time delays are  2.0, 3.0, 1.0, and 1.0 respectively.
The polarity of FILEC is reversed.
FILED's weighting in the summation is half that of the other files.
```

```
Now you enter::

    u:  INCREMENTSTACK
    u:  CHANGESTACK FILEC NORMAL
    u:  SUMSTACK

This stack is performed with the following delays::

     2.03, 3.01, 1.03, and 1.03.  The polarity of FILEC is now normal.

Now you enter::

    u:  DELETESTACK FILED
    u:  INCREMENTSTACK
    u:  SUMSTACK

This third stack is performed on the three files::

   FILEA, FILEB, and FILEC.  The delays are 2.06, 3.02, and 1.06 respectively.
```

ERRORS MESSAGES

- 5108: Maximum length of stack file list exceeded.
- 1306: Illegal operation on unevenly spaced file
  - stacking module requires evenly spaced data files.
- 1307: Illegal operation on spectral file
- 5109: Sampling intervals are not equal.
  - **the sampling rates for all files in the stack file list must agree** to within a given tolerance.
- the SRCHECK command can turn this check off or change the tolerance.
- the default check is for agreement within machine roundoff error.

LIMITATIONS

The maximum number of files in the stack file list is limited to the maximum number of data files allowed by SAC.

SEE COMMANDS

GLOBALSTACK, SUMSTACK, CHANGESTACK, INCREMENTSTACK, DELETESTACK

CHANGESTACK

SUMMARY

Change properties of files currently in the stack file list.

SYNTAX

`[C]HANGE[S]TACK filename|filenumber property {property}`

where property is one or more of the following:

`[W]EIGHT v [DI]STANCE v [BE]GINTIME v [END]TIME v [DE]LAY v {S]ECONDS]|[P]OINTS}`
`[I]NCREMENT v {[S]ECONDS|[P]OINTS} [N]ORMAL [R]EVERSED`

INPUT

filename: The name of the file in the stack file list.

filenumber: The number of the file in the stack hfile list.

[W]EIGHT v: Weighting factor for this file in the range zero to one. Each data point is multiplied by this value when the stack is summed.

[DI]STANCE v: Station to epicenter distance in kilometers for this file. This is used to calculate dynamic time delays.

[BE]GINTIME v: Time of the begining of the event.

[END]TIME v: Time of the end of the event.

[DE]LAY v {[S]ECONDS|[P]OINTS}: Static time delay to apply to file. This delay is in either seconds or number of data points.

[I]NCREMENT v {[S]ECONDS|[P]OINTS}: Static time delay increment for this file. This increment is in either seconds or number of data points. The static time delay is incremented by this ammount each time the INCREMENTSTACK command is executed.

[N]ORMAL: File has normal polarity.

[R]EVERSED: File has reversed polarity. (Each data point in the signal is multiplied by -1.0 when the stack is summed.)

DESCRIPTION

This command allows you to change any of the properties associated with files in the stack file list. These properties are discussed in more detail in the ADDSTACK command and an example of the use of this command is given there. This command leaves all other properties for all other files unchanged.

ERROR MESSAGES

- 5106: File name not in stack file lists:

SEE COMMNADS

ADDSTACK

DELETESTACK

SUMMARY

Deletes one or more files from the stack file list.

SYNTAX

```
[D]ELETE[S]TACK filename|filenumber {filename|filenumber...}
```

INPUT

filename: The name of the file in the stack file list.
filenumber: The number of the file in the stack file list.

EXAMPLES

See the example for the ADDSTACK command.

ERROR MESSAGES

- 5106: File name not in file list
- 5107: File number not in file list

SEE COMMANDS

ADDSTACK

DELTACHECK

SUMMARY

Change the sampling rate checking option.

SYNTAX

```
DELTACHECK ON|OFF|[R]OUNDOFF|v
```

INPUT

:ON; Turn sampling rate checking option on.

OFF: Turn sampling rate checking option off.

ROUNDOFF: Turn sampling rate checking option on and force sampling rates to agree within machine roundoff factor.

v: Turn sampling rate checking option on and force sampling rates to agree within a tolerance of v.

DEFAULTS VALUES

“ DELTACHECK ROUNDOFF“

DESCRIPTION

This command changes the sampling rate checking option. When this option is off, no check is made to see if the sampling rates for each of the files in the stack file list agree. When this option is on, then the sampling rates must agree within a given tolerance or it is considered an error. The tolerance can be set to a factor near the roundoff error for a particular machine or it can be set to a specific value. The absolute value of the difference between all sampling rates must be less than or equal to this tolerance in order to be allowed.

ERROR MESSAGES

The check is done and the error is reported as files are added to the stack file list (see ADDSTACK).

DISTANCEAXIS

SUMMARY

Define the record section plot distance axis parameters.

SYNTAX

" [D]ISTANCE[A]XIS FIXED v|SCALED v"

INPUT

FIXED v: Force axis length to be v cm long.

SCALED v: Allow the axis length to be scaled to the data. The axis length (in cm) will be the range of the axis (in km.) divided by v.

DEFAULTS VALUES

```
DISTANCEAXIS FIXED 35
```

DESCRIPTION

This command defines the properties of the distance axis for subsequent record section plots (see PLOTRECORDSECTION.) The length of the axis can be a fixed size or it can be scaled to the range of the axis variable (distance in this case.) The TIMEAXIS command controls the time axis properties.

EXAMPLES

If you entered the following command:

```
u:  DISTANCEAXIS SCALED 2.0
```

and the minimum and maximum distances in the data set being plotted are 150 and 300 km respectively, the distance axis would be 75 cm long.

KNOWN BUGS

The y to x aspect ratio defined by this command and by the TIMEAXIS command is correct on plots to any device. The physical size requested is ignored when plotting to an interactive device (e.g. TERMINAL, XWINDOWS)

If the SGF device is requested, the physical size is stored in these files. The current SGF plot programs do not have the capability to make arbitrary sized plots (i.e. panelling). This panelling option is planned for a future release.

SEE COMMANDS

PLOTRECORDSECTION, TIMEAXIS

DISTANCEWINDOW

SUMMARY

Controls the distance window properties on subsequent record section plots.

SYNTAX

" [D]ISTANCE[W]INDOW [options]"

where options is one or more of the following:

`[U]SEDATA|[W]IDTH v|[F]IXED v1 v2`

`[UN]ITS [K]ILOMETERS|[D]EGREES`

INPUT

USEDATA: Use the minimum and maximum values of the distance properties of the files in the stack file list.

WIDTH v: Use the minimum value of the distance property of the files in the stack file list but force the width to be a fixed value. The maximum distance is then set to the minimum distance plus v.

FIXED v1 v2: Fix the minimum and maximum distances to be v1 and v2 respectively.

UNITS KILOMETERS: Set the distance window units to be in kilometers.

UNITS DEGREES: Set the distance window units to be in degrees of arc.

DEFAULTS VALUES

```
DISTANCEWINDOW USEDATA UNITS KILMETERS
```

KNOWN BUGS

The KILOMETER option is not currently implemented.

SEE COMMANDS

PLOTRECORDSECTION

GLOBALSTACK

SUMMARY

Sets global stack properties.

SYNTAX

```
[G]LOBAL[S]TACK property [property ...]
```

where property is one or more of the following:

```
[W]EIGHT v [DI]STANCE v [DE]LAY v [[S]ECONDS|[P]OINTS] [I]NCREMENT
v [[s]ECONDS|[P]OINTS] [N]ORMAL [R]EVERSED
```

INPUT

[W]EIGHT v: Global weighting factor in the range zero to one.

[DI]STANCE v: Global station to epicenter distance in kilometers.

[DE]LAY v [[S]ECONDS|[P]OINTA]: Global static time delay. This delay is in either seconds or number of data points.

[I]NCREMENT v [[S]ECONDS|[P]OINTS]: Global static time delay increment. This increment is in either seconds or number of data points.

[N]ORMAL: Normal global polarity property.

[R]EVERSED: Reversed global polarity property.

DESCRIPTION

This command allows you to define global stack properties. These global properties are associated with each file as it is added to the stack file list (see ADDSTACK) unless local values are given for that file. These properties are discussed in more detail in the ADDSTACK command and an example of the use of this command is given there.

SEE COMMANDS

ADDSTACK

INCREMENTSTACK

SUMMARY

Increments properties for files in the stack file list.

SYNTAX

```
[I]NCREMENT[S]TACK
```

DEFAULTS VALUES

All property INCREMENT values are initially 0.

DESCRIPTION

The properties that can be incremented are the static time delay, the apparent velocity, and the velocity model intercept time. Property increment values of 0. obviously leave those properties unchanged. The apparent velocity or the velocity model intercept time can be incremented, and the other one calculated in order to maintain a zero delay at a specified point.

EXAMPLES

An example of the use of this command to INCREMENT static time delays is shown in the ADDSTACK command. An example of the use of this command to increment dynamic delays is given below::

```
u: ADDSTACK FILEA
u: ADDSTACK FILEB
u: ADDSTACK FILEC
u: ADDSTACK FILED
u: VELOCITYMODEL 1 REFR VAPP 7.9 VAPPI 0.1 T0VM CALC DIST 320. TVM 45.
u: SUMSTACK
u: WRITESTACK STACK1
u: INCREMENTSTACK
u: SUMSTACK
u: WRITESTACK STACK2
u: INCREMENTSTACK
u: SUMSTACK
u: WRITESTACK STACK3
```

The above commands will produce three summations, the results being stored in SAC files STACK1, STACK2, and STACK3. The refracted wave velocity model is used and the apparent velocities, VAPP are 7.9, 8.0 and 8.1 respectively. The velocity model intercept time, T0VM, is calculated in each case to maintain a zero delay at the point 320. km and 45. seconds.

SEE COMMANDS

VELOCITYMODEL

LISTSTACK

SUMMARY

Lists the properties of the files in the stack file list.

SYNTAX

```
[L]IST[S]TACK {[N]ARROW |[W]IDE}
```

INPUT

NARROW: Use the narrow report format. Two lines of about 70 characters in width are output for each file.

WIDE: Use the wide report format. A single line of about 120 characters in width is output for each file.

DEFAULT VALUES

```
LISTSTACK NARROW
```

PLOTRECORDSECTION

SUMMARY

Plots a record section of the files in the stack file list.

SYNTAX

```
[P]LOT[R]ECORD[S]ECTION [ options ]
```

where options are one or more of the following:

```
[L]ABLES ON | OFF | headerfield [O]RIGIN [D]EFAULT | [R]EVERSED [R]EFERENCELINE
ON | OFF [S]IZE v [W]EIGHT ON | OFF [P]OLARITY ON | OFF [C]URSOR ON | OFF
[RED]UCED ON | OFF | [P]HASE phase_name | [V]ELOCITY velocity_value [A]SPECT
ON | OFF `` ``[ORIE]NT [P]ORTRAIT | [L]ANDSCAPE [T]TIME ON | OFF | [D]EFAULT
| TEXT [X]LABEL ON | OFF | [D]EFAULT | TEXT [Y]LABEL ON | OFF | [D]EFAULT
| TEXT PRINT {pname}
```

INPUT

LABELS ON | OFF: Turn the file labeling option on or off. When this option is on, each file is labeled with header field.

LABELS headerfield: Turn the file labeling option on and set the name of the header field.

ORIGIN DEFAULT | REVERSED: In portrait mode, distance is along the y axis and default puts the distance origin at the top left corner. In landscape mode, distance is along the x axis and default puts the distance origin at the bottom left corner.

REFERENCELINE ON | OFF: Turn reference line option on or off. When this option is on, a vertical dotted line is drawn at the distance property value for each file.

SIZE v:

WEIGHT ON | OFF: Turn the file weighting option on or off.

POLARITY ON | OFF: Turn the file polarity option on or off.

CURSOR ON | OFF: See below.

REDUCED ON | OFF | VELOCITY number | PHASE name: Reduced travel time curves can be computed relative to a specific velocity or a phase from the traveltime curves. See SSTRAVELTIME.

ORIENT portrait | landscape: In portrait mode, horizontal axis indicates time and the vertical axis indicates event to station distance. In landscape mode, the horizontal axis indicates the event to station distance and time is along the vertical axis.

TTIME ON | OFF | DEFAULT | TEXT: Turn traveltime curves on. Traveltime curves must have been computed with the traveltime command.

XLABEL ON | OFF | DEFAULT | TEXT: Turn xlabel on and/or set xlabel text.

YLABEL ON | OFF | DEFAULT | TEXT: Turn ylabel on and/or set ylabel text.

PRINT {pname}: Print the resultant plot. If pname is specified, print to named printer, else use default printer.

DEFAULT VALUES

```
PLOTRECORDSECTION LABELS filename ORIGIN default REFERENCELINE on `` ``SIZE
0.1 WEIGHT on POLARITY on ORIENT portrait REDUCED off `` ``CURSOR off TTIME
off
```

DESCRIPTION

This command plots the files in the stack file list in a record section format. The effect of a particular velocity model can easily be seen with this plot. In portrait mode, the x axis is time, the y axis is epicentral distance. Landscape mode reverses these axes. The zero amplitude of each file is plotted at its epicentral distance along the distance axis. A distance must be defined for all files in the stack file list for this plot to be generated. The distance can come from the header or it can be defined in the DISTANCE option of the GLOBALSTACK, ADDSTACK, or CHANGESTACK command. The DISTANCEWINDOW and TIMEWINDOW commands control how much data will be displayed. The DISTANCEAXIS and TIMEAXIS commands control the size of each axis and thus the aspect ratio of the complete plot. Dynamic delays are controlled by the use of the VELOCITYMODEL command. A line showing the effects of a second velocity model is also controlled by the VELOCITYMODEL command. A velocity rosette showing the effects of other velocities can be placed on this plot. It is controlled by the VELOCITYROSET command. Static delays if they have been defined are also applied to each of the signals.

CURSOR ON MODE

In cursor on mode two additional functionalities are available: zooming and apparent velocity determination. The zoom capability requires the user to specify where to crop the picture. The user does this by moving the cursor to one corner of the desired display area and typing c1 (no mouse click is needed) and moving the cursor to the opposite corner and typing c2. When the user types c2, plotrecordsection replots the data including only those data files which lie within the distance window, and cutting off all the data points which fall outside the time window.

The user can type o (the letter 'o') to replot the original. Zoomed plots can be nested to five levels (ie, you can zoom a previously zoomed plot, entering o causes prs to unzoom one level and replot). The cursor on option also allows the user to measure apparent velocity by moving the cursor and typing v1 and v2 to mark the points. Once the v2 is selected, the apparent velocity is printed on the output device and stored in a blackboard variable, vapp. Multiple v2 values can be set but only the latest will be stored in the blackboard variable. While the cursor is on, the available commands in the plot window are c1, c2, v1, v2, and q which quits cursor on (q returns control to the sss subprocess).

SEE COMMANDS

GLOBALSTACK. ADDSTACK, CHANGESTACK DISTANCEWINDOW, TIMEWINDOW, DISTANCEAXIS, TIMEAXIS, VELOCITYMODEL, VELOCITYROSET, FILENUMBER

PLOTSTACK

SUMMARY

Plots the files in the stack file list.

SYNTAX

`[P]LOT[S]TACK [ options ]`

where options are one or more of the following:

`[S]UM ON | OFF [P]ERPLOT ON | OFF | n [W]EIGHT ON | OFF [P]OLARITY ON | OFF`

INPUT

SUM ON | OFF: When this option is on the summed file is plotted first followed by the files in the stack file list. When this option is off, the summed file is not plotted.

PERPLOT ON | OFF: Turn the per plot option on or off. When this option is on, a fixed number of files are plotted in each frame. When this option is off, all of the files in the stack file list are plotted in a single frame.

PERPLOT n: Turn the per plot option on and set the number of files per frame to n.

WEIGHT ON | OFF: Turn the file weighting option on or off.

POLARITY ON | OFF: Turn the file polarity option on or off.

DEFAULTS VALUES

`PLOTSTACK SUM ON PERPLOT OFF WEIGHT ON POLARITY ON`

DESCRIPTION

This command plots the files in the stack file list. The files are always plotted with their delays. They may be plotted with or without their weighting factors and polarities. They may also be plotted with or without the summed signal. A selectable number of the files can be plotted on each frame. The format of the plot is identical to that of the PLOT1 command. Each file is plotted in its own subplot region. These subplot regions have a common x axis and separte y axes. A legend consisting of the file name and any non-default properties is placed in the upper left-hand corner of each subplot region.

SEE COMMANDS

TIMEWINDOW

SUMSTACK

SUMMARY

Sums the files in the stack file list.

SYNTAX

" [S]UM[S]STACK [[N]ORMALIZATION ON|OFF]"

INPUT

NORMALIZATION ON | OFF: Turn normalization option on or off. When this option is on the resulting summation is normalized by dividing each point by a factor that is the sum of each file's weight.

DEFAULTS VALUES

```
SUMSTACK NORMALIZATION ON
```

DESCRIPTION

This command sums the files in the stack file list. A stack time window (see TIMEWINDOW) must have been defined before this command is executed. Each signal is shifted in accordance with its static and dynamic delays. Zeros are added to the sum for that part of each file that is not in the time window. Each file is given the requested weighting and files with reversed polarity are inverted.

A plot of the summation is automatically produced. The summation can be saved on disk using the WRITESTACK command.

ERRORS MESSAGES

- 5103: No time window defined.

SEE COMMANDS

TIMEWINDOW, WRITESTACK

TIMEAXIS

SUMMARY

Controls the time axis properties on subsequent record section plots.

SYNTAX

```
[T]IME[A]XIX [F]IXED v|[S]CALED v
```

INPUT

FIXED v: Fix the length of the time axis in cm to v.
SCALED v: Scale the length of the time axis in cm to be v times the total time window.

DEFAULTS VALUES

```
TIMEAXIS FIXED 23.0
```

EXAMPLES

If you are making several record setion plots with different time windows and you want each 2 seconds on these plots to be 1 cm long:

u: TIMEAXIS SCALED 0.5

KNOWN BUGS

The y and x aspect ration defined by this command and by the DISTANCEAXIS command is correct onplots to any device. The physical size requested is ignored when plotting to an interactive device (e.g. TERMINAL, XWINDOWS) If the SGF device is requested, the physical size is stored in these files. The current SGF plot programs do not have the capability to make arbitrary sized plots (i.e. panelling). This panelling option is planned for a future release.

SEE COMMANDS

PLOTRECORDSECTION, DISTANCEAXIS

TIMEWINDOW

SUMMARY

Sets the time window limits for subsequent stack summations.

SYNTAX

```
[T]IME[W]INDOW v1 v2
```

INPUT

v1 v2: The time window limits to use when reading in files before doing a stack summation.

DEFAULT VALUES

None. You MUST specify a time window before doing a summation

DESCRIPTION

This command sets the stack time window. This is the time window that will be used in subsequent SUMSTACK PLOTSTACK, and PLOTRECORDSECTION commands. The stack time window must be defined before any of these commands are executed. If a particular file does not fall entirely within this stack time window enough zeros are added before or after the actual data to make up the difference.

SEE COMMANDS

SUMSTACK, PLOTSTACK, PLOTRECORDSECTION

SSSTRAVELTIME

SUMMARY

Computes traveltime of selected phases for pre-defined velocity models Description and examples assume the calls are made from within SSS. To see the help file for the non-SSS version, go to TRAVELTIME.

SYNTAX

```
TRAVELTIME {MODEL string} {PICKS number} {PHASE phase list}  {VERBOSE | QUIET} {M |KM}
```

INPUT

MODEL: iasp91 [default], ak135

Picks: There are a total of 10 time picks in the SAC header: t0 to t9. If the number is n (0<=n<=9), the first phase will be at Tn. If PICKS is not included among the command-line options, VERBOSE will be turned on and the phase arrival times will be displayed but will not be put in the header.

PHASE: List of phases for which times are picked and displayed. If PICKS n is among the options, the phase arrivala times and their labels will be added to the header starting at Tn.

VERBOSE | QUIET: If VERBOSE is among the options in the TRAVELTIME command line, phase arrival times are displayed relative to both the origin time (O) and to the first-point time (B). If QUIET is among the options in the TRAVELTIME command line, VERBOSE is turned off (if it was on) and nothing is displayed on the screen. If neither has been displayed, the depth used by TRAVELTIME is displayed.

M | KM: If M is among the options in the command line, SAC interprets EVDP as being in meters. If KM is displayed, EVDP is interpreted as being in kilometers.

DEFAULT VALUES: MODEL iasp91 KM PHASE P S Pn Pg Sn Sg

DESCRIPTION

All waeforms in memory must have event and station locations defined as well as the origin time.

This command calculates traveltimes using the iaspei-tau procedures developed for models iasp91 or ak135. For more information on this package, go to the iaspei-tau link at URL <https://seiscode.iris.washington.edu/projects/iaspei-tau/>. The phase picks are stored in the SAC header for all files in memory in header variables Tn, where n is in the range 0 to 9. The times are calculated relative to the origin time (O) but the Tn times are relative to the first-point time (B). The resulting traveltime curves can be plotted on top of a record section plot using the PLOTRECORDSECTION command. If n and the number of phases are such that the phase list runs about T9, those extra phases are not plotted or recorded in the header, but curves will be displayed with the PLOTRECORDSECTION command if the TTIME option is turned on.

The traveltime tables used to calculate the stored degree-distance measure (GCARC). (GCARC is calculated from the event and station latitude and longitudes using spherical-triangle geometry after converting geographic latitudes to geocentric.)

For historical reason, the units for EVDP were meters, and SAC waveforms proudced by RDSEED have EVDP in meters. In v101.5, the default for EVDP is kilometers, but as many waveforms have EVDP in meters, we introduced the command option M which if set to ON means the input EVDP for all files in memory have EVDP inmeters. In the final example, the SAC files had been extracted from a SEED vlume using RDSEED for a version prior to v5.2, so EVDEP is in meters, (Starting with v5.2, the depth in RDSEED is in kilometers.)

EXAMPLES

A regional event using the default phases:

```
SAC> fg seismo
SAC> SSS
Signal Stacking Subprocess.
SAC/SSS>  traveltime
traveltime: depth: 15.000000
traveltime: error finding phase P
traveltime: error finding phase S
traveltime: setting phase Pn       at 10.464321 s [ t = 51.894321 s ]
traveltime: setting phase Pg       at 22.904724 s [ t = 64.334724 s ]
traveltime: setting phase Sn       at 50.047722 s [ t = 91.477722 s ]
traveltime: setting phase Sg       at 66.414337 s [ t = 107.844337 s ]
SAC/SSS>
```

For regional events the first arrivals are Pn or Pg, so by the convention used here there is no "P" arival. For the above example, no picks are added to the header.:

```
SAC> fg seismo
SAC> SSS
Signal Stacking Subprocess.
SAC/SSS> traveltime picks 0 phase Pn Pg Sn Sg
traveltime: depth: 15.000000
SAC/SSS> lh AMARKER T0MARKER T1MARKER T2MARKER T3MARKER
AMARKER = 10.464                        T0MARKER = 10.464          (Pn)
T1MARKER = 22.905          (Pg)         T2MARKER = 50.048          (Sn)
T3MARKER = 66.414          (Sg)
SAC/SSS> qs
SAC> write seismo-picks.z
SAC>
```

We see that the already defined A is at Pn. The file written to seismo-picks.z will have T0 through T3 in the header, and a call to PLOT1 will show labeled vertical lines at the calculated times.

Note that even though VERBOSE was not turned on, the depth (in km) used is printed out. This is a safeguard to assure that one has made the correct assumption about the EVDP units. The printing of traveltime depth can be suppressed by entering QUIET on the command line.

The header for the waveform in the following example was produced by RDSEED (V5.0) and EVDP is in meters:

```
SAC> r 2008.052.14.16.03.0000.XC.OR075.00.LHZ.M.SAC
SAC> lh evdp
evdp = 6.700000e+03
SAC> SSS
Signal Stacking Subprocess.
SAC/SSS> traveltime M picks 0
traveltime: depth: 6.700000 km
SAC/SSS> qs
SAC> lh t0marker t1marker t2marker t3marker
t0marker = 61.48           (Pn)         t1marker = 76.413          (Pg)
t2marker = 109.66          (Sn)         t3marker = 132.11          (Sg)
SAC> ch evdp (0.001 * &1,evdp&)
SAC> setbb station &1,KSTNM&
SAC> write %station%.z
SAC>
```

The saved file, OR075.z, will have evdp in kilometers and annotated picks at the times for Pn, Pg, Sn, and Sg. (One could have done the chnhdr command before calling TRAVELTME and then not entered M on the command line.)

Phase names are case sensitive. See the iaspei-tau documentation.

See Commands

PLOTRECORDSECTION

LATEST REVISION

August, 2011 (Version 101.5)

VELOCITYMODEL

SUMMARY

Sets stack velocity model parameters for computing dynamic delays.

SYNTAX

`[V]ELOCITY[M]ODEL n options`

where options are one or more of the following:

`ON|OFF REFRACTEDWAVE|NORMALMOVEOUT FLIP VAPP v|CALCULATE T0VM v|CALCULATE DVM v1 [v2] TVM v1 [v2] VAPPI v T0VMI v`

INPUT

n: Set velocity model number. This is either "1" or "2". The use of each velocity model is described below.

ON|OFF: Turn velocity model option on or off. When this option is on the model is applied. When off it is ignored.

REFRACTEDWAVE: Turn velocity model option on and change to the "refracted wave" model.

NORMALMOVEOUT: Turn velocity model option on and change to the "normal moveout" model.

FLIP: Interchange the properties of the two velocity models.

VAPP v: Set the apparent velocity to v.

VAPP CALCULATE: Have SAC calculate the apparent velocity.

T0VM v: Set the time axis intercept to v.

T0VM CALCULATE: Have SAC calculate the time axis intercept.

DVM v1 [v2]: Define one or two reference distances.

TVM v1 [v2]: Define one or two reference times.

VAPPI v: Set the apparent velocity increment to v. The apparent velocity is incremented by this ammount each time the INCREMENTSTACK command is executed.

T0VMI v: Set the time axis intercept increment to v. The time axis intercept is incremented by this ammount each time the INCREMENTSTACK command is executed.

DEFAULT VALUES

" VELOCITYMODEL 1 OFF" " VELOCITYMODEL 2 OFF"

DESCRIPTION

The first velocity model is used in calculating dynamic station delays for a particular phase.

It is applied when doing a stack summation (SUMSTACK), a stack plot (PLOTSTACK), or a record section plot (PLOTRECORDSECTION.) The second velocity model is used in the record section plot to show delays associated with a second phase. The parameters associated with the two velocity models can be easily flipped.

There are two different types of velocity models ("refracted wave" and "normal moveout") that can be applied. They are defined by the following equations:

```
   TDELAY = TVM(1) - ( T0VM + DIST / VAPP )
   TDELAY = TVM(1) - SQRT ( T0VM**2 + ( DIST / VAPP )**2 )

There are several ways in which these velocity model delays can be
calculated:

Enter values for VAPP, T0VM, and TVM(1) directly.

Enter values for DVM(1), TVM(1), and either VAPP or T0VM.
SAC will calculate the missing variable such that TDELAY will be zero
at the distance given by DVM(1).

Enter values for DVM(1), TVM(1), DVM(2), and TVM(2).
SAC will calculate both VAPP and T0VM such that TDELAY will be zero
at the distance given by DVM(1).
```

EXAMPLES

To set the first stack velocity model the refracted wave model with an apparent velocity of 6.5 km/sec and to have SAC calculate T0VM such that the delay at 200 km will be zero:

```
u: VELOCITYMODEL 1 REFRACTEDWAVE VAPP 6.5 T0VM CALCULATE DVM 200 TVM 35
```

SEE COMMANDS

SUMSTACK, PLOTSTACK, PLOTRECORDSECTION

VELOCITYROSET

SUMMARY

Controls the placement of a velocity roset on subsequent record section plots.

SYNTAX

```
[V]ELOCITY[R]OSET [ON|OFF]  [[L]OCATION UL|UR|LL|LR]
```

INPUT

ON | OFF: Turn velocity roset plotting option on or off.

LOCATION UL|UR|LL|LR: Change location on plot of velocity roset. Locations are respectively upper left, upper right, lower left, and lower right of the record section plot.

DEFAULTS

```
VELOCITYROSET OFF LOCATION LL
```

WRITESTACK

SUMMARY

Writes the stack summation to disk.

SYNTAX

```
[W]RITE[S]TACK {COMMIT|ROLLBACK|RECALLTRACE} [filename]
```

INPUT

COMMIT: Commits headers and waveforms in SAC memory -- removing any previous versions of headers or waveforms from RAM -- prior to writing files. COMMIT is the default.

ROLLBACK: reverts to the last committed version of the header and waveform before writing files.

RECALLTRACE:
- reverts to the last committed version of the waveform
- reverts to the last committed version of those header variables closely linked to the waveform,
- commits those header variables which are loosely linked to the waveform. (use HELP RECALLTRACE for a list of which header variables are committed, and which are rolled back.)

filename: The name of the disk file into which the summation is to be written.

DEFAULT VALUES

WRITESTACK SUM COMMIT

ZEROSTACK

SUMMARY

Zeros or reinitializes the signal stack.

SYNTAX

```
[Z]ERO[S]TACK
```

DESCRIPTION

This command zeros the signal stack. It deletes all entries in the stack file list and sets the global stack properties back to their original values.

4 Spectral-Estimation Subprocess

Spectral Estimation (SPE)

Introduction

A subprocess is effectively a small program within the main SAC program. You start a subprocess by typing its name (SPE in this case.) You can terminate it and return to the main program using the QUITSUB command. You can also terminate SAC from within a subprocess using the QUIT command.

While within a subprocess, you can execute any command belonging to that subprocess plus a limited number of main SAC commands.

SPE is a Spectrum Estimation package intended primarily for use with stationary random processes. It contains three different spectral estimation techniques:

- Power Density Spectra (PDS),
- Maximum Likelihood Method (MLM), and
- Maximum Entropy Method (MEM).

These are all indirect methods, because they use a sample correlation function, rather than the data itself, to estimate the spectral content.

SPE Commands

COR: Computes the correlation function.

MEM: Calculates the spectral estimate using Maximum Entropy Method.

MLM: Calculates the spectral estimate using Maximum Likelihood Method.

PDS: Calculates the spectral estimate using Power Density Spectra Method.

PLOTCOR: Plots the correlation function.

PLOTPE: Plots the RMS prediction error function.

PLOTSPE: Plots the spectral estimate.

QUITSUB: Terminates a SAC subprocess.

READCOR: Almost the same as the normal READ. See below.

WRITECOR: Writes a SAC file containing the correlation function.

WRITESPE: Writes a SAC file containing the spectral estimate.

Their abbreviated names are also allowed.

Main SAC Commands executable from within the SPE subprocess:

```
AXES            BEGINDEVICES    BEGINFRAME
BEGINWINDOW     BORDER          COLOR
COMCOR          COPYHDR         DATAGEN
ECHO            ENDDEVICES      ENDFRAME
ERASE           EVALUATE        FLOOR
GETBB           GRID            GTEXT
HELP            INSTALLMACRO    LINE
LINLIN          LINLOG          LOGLAB
LOGLIN          LOGLOG          MACRO
MESSAGE         PAUSE           PLABEL
```

```
PLOTC           QDP             QUIT
READALPHA       READBBF         REPORT
SETBB           SETDATADIR      SETDEVICE
SETMACRO        SGF             SYMBOL
SYNTAX          SYSTEMCOMMAND   TICKS
TITLE           TSIZE           VSPACE
WAIT            WINDOW          WRITEBBF
XDIV            XFUDGE          XFULL
XGRID           XLABEL          XLIM
XLIN            XLOG            XVPORT
YDIV            YFUDGE          YFULL
YGRID           YLABEL          YLIM
YLIN            YLOG            YVPORT
```

The Theory

SPE is intended primarily for use with stationary random processes. It implements three different indirect spectral estimators. They are called indirect, because they do not estimate the spectrum directly from the data, but from a sample correlation function that is computed from the data. The choice of indirect methods is a matter of taste, since direct spectral estimation techniques are also available. The correlation function itself is a useful quantity. You may wish to examine it in the course of performing spectral estimation tasks.

The choice of indirect techniques is supported by "Spectral Analysis and Its Application," by Jenkins and Watts, a respected reference on the subject of spectrum estimation.

The type of spectrum estimated by SPE is properly described as the power density spectrum, with the spectrum defined in the frequency domain. Thus, the estimated power delivered by the random process in some band of frequencies is the integral of the spectral power density estimate over that band of frequencies.

User Control

SPE affords the user some control over the details of estimation process. For some, with experience in estimating spectra, this is highly desirable. Defaults are provided for those who do not wish to become involved in the details of the theory.

The user has a choice of data window type, size, and the number of windows used when estimating the correlation function. Generally these parameters control the resolution of the estimate, and the amount of reduction of variance desired in the final estimate. In addition, prewhitening of the data may be specified as part of the process of estimating the correlation function. Prewhitening often has the effect of mitigating a severe "window bias" that can occur in spectral estimates having a high dynamic range. The warping of the spectrum that occurs with prewhitening is compensated for in the final result. In this implementation, low-order prediction error filters are used for prewhitening.

The Estimators

The user has a choice of three spectral estimators: Power Density Spectra (PDS), Maximum Likelihood Method (MLM), and Maximum Entropy Method (MEM). Command COR must be run before running any of these.

PDS: The PDS estimator is quite simple: the sample correlation function is multiplied by a correlation window, then the result is transformed with an FFT to obtain the spectral estimate. The user again has a choice of the window type and the size of the window. The above mentioned book by Jenkins and Watts could be considered as the detailed documentation for the PDS technique.

MLM: The MLM estimator generates a spectral estimate which is the power output of a bank of narrow band-pass filters which have been optimized to reject out-of-band power. The result is a smoothed, parametric estimate of the power density spectrum. The user can choose the number of parameters. Documentation for this method can be found in the paper by Richard Lacoss in the IEEE book "Modern Spectrum Analysis" by Donald Childers.

MEM: The MEM estimator is another parametric method, which uses a prediction error filter to whiten the data. The resulting spectral estimate is proportional to the inverse of the filter's power frequency response. The user is free to choose the order of the prediction error filter. Documentation for this method can be found in the review paper on linear prediction by John Makhoul in "Modern Spectrum Analysis." The formal name of the actual method implemented is the Yule-Walker method.

Diagnostics

In addition to the spectrum, several diagnostic functions can be calculated and plotted. The prediction error can be plotted as a function of order. This plot can be used to select a good size for the prediction error filter used in the MEM method. Since much is known about the performance of the PDS estimator, more diagnostic information is available for this method in SPE. The 90" confidence limits can be estimated theoretically, as can the frequency resolution of the estimate. Both of these quantities can be indicated on a PDS spectral plot.

Differences between SPE> and SAC>

There are two primary differences between SPE and the main SAC program. Only one data file can be processed by SPE at a time. This is because SPE produces and stores a number of auxiliary functions (the correlation function, the prediction error function, and the spectral estimate itself) as it proceeds. This restriction to a single data file may be removed in the future. The second difference is that, unlike SAC itself, there is a specific order or progression in which the commands are generally executed.

Initialization

This progression begins when the SPE command is executed. A data file must be in memory when SPE is initiated. While in SPE, command READ can be used to read in an additional file at any time. Space for the above mentioned auxiliary functions is created for each new file.

READCOR run from within SPE works just like the READ command in the main SAC program with two exceptions.

First, only ONE file may be read in while in SPE. Second, executing this command deletes any correlation function or spectral estimate that may already have been computed. Parameters within SPE, such as the number of prewhitening coefficients or the window type and length, are not changed when this command is executed.

To reinitialize all SPE parameters, terminate the subprocess using the QUITSUB command and then start it over again.

Correlation

The correlation function is then computed, using the COR command. COR must be run prior to running a spectral estimator. The correlation function may be saved as a SAC data file using the WRITECOR command and later read back in using the READCOR command. This is more efficient than recomputing the correlation each time, especially if the data file is very long. At this point, you may wish to examine the correlation function using the PLOTCOR command. You may also wish to examine the prediction error function using the PLOTPE command if you are going to use the MEM method.

Estimation

Now you are ready to select one of the three spectral estimation techniques using the PDS, MLM, or MEM commands. If the data file has a non-zero mean, MLM and MEM may not work correctly. Running command RMEAN before entering SPE should solve this problem. Each technique has its own options. You may now examine the resulting spectrum using the PLOTSPE command. There are several different scaling options available. You can also save the spectral estimate as a SAC data file using the WRITESPE command.

Termination

At this point you have several options: you can select a different spectral estimate technique, read in a different correlation function, read in a different data file, terminate the subprocess using the QUITSUB command, or terminate SAC using the QUIT command.

COR

SUMMARY

Computes the correlation function.

SYNTAX

```
COR  {[N]UMBER n|ON|OFF}, {[L}ENGTH v}, {[T]YPE type}{[P]REWHITEN ON|OFF|n},
{[S]TOCASTIC|[TR]ANSIENT}
```

where type is one of the following:
`[HAM]MING [HAN]NING [C]OSINE [R]ECTANGLE [T]RIANGLE`

INPUTS

NUMBER n: Fix number of windows to n.
NUMBER {ON}: Fix number of windows to previous value.
NUMBER OFF: Compute number of windows based upon data length and window length. There will be no data overlap when using this option.
LENGTH v: Set window length to v seconds.
TYPE type: Set window type. The advantages of each is discussed below.
PREWHITEN {ON}: Turn prewhitening of data on.
PREWHITEN OFF: Turn prewhitening of data off.
PREWHITEN n: Turn prewhitening of data on and change number of coefficients to n.
STOCHASTIC: Set correlation scaling assuming that the data is stochastic (random.)
TRANSIENT: Set correlation scaling assuming that the data is transient (signal.)

DEFAULT VALUES

```
COR NUMBER OFF TYPE HAMMING PREWHITEN OFF
```

note that if PREWHITEN is turned on without specifying the order, it will default to 6 unless it has been previously set by the WHITEN command in SPE.

DESCRIPTION

This correlation command assumes that the data is stationary. Under that assumption the data is segmented into many windows, and a sample correlation function is calculated for each window. These sample correlation functions are averaged to produce a more stable estimate of the underlying correlation function of the random process. The number of windows, the window length, and the window type (called a data window, to distinguish it from a window used in the PDS command) are under user control. If the window length times the number of windows exceeds the total data length, the program overlaps the windows. The amount of overlap is not under user control.

For a fixed data size, there is obviously a tradeoff between the number of windows to be used and the window size. This tradeoff ultimately results in a tradeoff between the bias and variance of the spectral estimates made using the correlation function.

The frequency-domain resolution of a spectral estimator depends on the length of the available correlation and, therefore, indirectly on the size of the data window. The larger the correlation window, the smaller the bias in the spectral estimate resulting from frequency-domain smoothing.

However, as the data window size is increased, fewer windows can be used in the averaging process. Consequently, the variance of the correlation function estimate increases, and with it, the variance of the spectral estimate.

The choice of data window type can be used to fine-tune the tradeoff between bias and variance. The smoother windows tend to taper the data off near the window edges, effectively reducing window length. Thus, the windows can be overlapped more, and more can be used. This choice decreases variance at the expense of increasing bias.

There is another important source of bias when the dynamic range of the spectrum is quite large. This is the effect of window leakage, that shows up most clearly when the PDS estimator is used. Power leakage through the sidelobes of the Fourier Transform of the correlation window puts a floor on the estimated spectrum. In typical seismic data, this floor is quite regular and appears at high frequencies, where the spectrum is typically quite small. The correlation function estimator has an optional prewhitening capability that mitigates the sidelobe-leakage problem. A low-order prediction error filter is used to flatten the spectrum of the data prior to the calculation of the correlation function. The effect of the filter is compensated for in the calculation of the spectrum.

Prewhitening of the data is done in place and thus corrupts the original signal. If you use prewhitening, quit the subprocess, and wish to use the original signal in some other operation, you MUST reread it into SAC.

This correlation function is used in the calculation of the spectral estimate.

COR must therefore be executed before PDS, MLM, or MEM.

You may plot the correlation function using the PLOTCOR command and save it as a SAC data file using the WRITECOR command. Such a file can then be read in using READCOR.

HEADER CHANGES

DEPMIN, DEPMAX, DEPMIN

LIMINTATIONS

Prior to 1998, maximum window length was 2048 samples. No restriction now.

SEE COMMANDS

PLOTCOR, WRITECOR, WHITEN, READCOR

MEM

SUMMARY

Calculates the spectral estimate using the Maximum Entropy Method.

SYNTAX:

```
MEM {[O]RDER n}, {[N]UMBER n}
```

INPUT

ORDER n: Set the order of the prediction error filter in lags to n.
NUMBER n: Set the number of points to be used in the spectral estimate.

DEFAULT VALUES

" MEM ORDER 25"

DESCRIPTION

This command implements the Maximum Entropy Method estimator. This estimator is a parametric method, which uses a prediction error filter to whiten the data. The resulting spectral estimate is proportional to the inverse of the filter's power frequency response. The user is free to choose the order of the prediction error filter. See the documentation of the PPE command for further details.

The principal advantage of this method is the very high resolution that it can achieve with a relatively small amount of data. Its disadvantage is that less can be said about it theoretically than about the conventional method.

Documentation for this method can be found in the review paper on linear prediction by John Makhoul in "Modern Spectrum Analysis." The formal name of the actual method implemented is the Yule-Walker method.

ERROR MESSAGES

- 5003 No correlation function calculated.

SEE COMMANDS

COR, WRITESPE, PLOTSPE

MLM

SUMMARY

Calculates the spectral estimate using the Maximum Likelihood Method.

SYNTAX

```
MLM {[O]RDER n}, {[N]UMBER n}
```

INPUT

ORDER n: Set the number of parameters in the estimate in lags to v.
NUMBER n: Set the number of points to be used in the spectral estimate.

DEFAULT VALUES

" MEM ORDER 25"

DESCRIPTION

This command implements the Maximum Likelihood Method estimator for the power density spectrum. This estimator generates a spectral estimate which represents the power outputs of a bank of narrow band-pass filters which have been optimized to reject out-of-band power. The result is a smoothed, parametric estimate of the power density spectrum. The parameters are the coefficients of the (finite impulse response) narrowband filters. The user can choose the number of parameters. The filters are not actually computed by the algorithm, which accounts for the speed of the method.

The method is desirable because it generally has better resolution than the conventional method, and much better sidelobe reduction. The order of the algorithm is limited to 100, since it requires the inversion of a matrix with dimension equal to the order. A fast method exists for the inversion, but numerical noise can be a problem for large order estimates.

Documentation for this method can be found in the paper by Richard Lacoss in the IEEE book "Modern Spectrum Analysis" by Donald Childers.

ERROR MESSAGES

- 5003 No correlation functioncalculated.

SEE COMMANDS

COR, PLOTSPE, WRITESPE

PDS

SUMMARY

Calculates the spectral estimate using the Power Density Spectra Method.

SYNTAX

" PDS {[S]ECONDS v|[L]AGS n}, {[N]UMBER n}, {[T]YPE type}"

where type is one of the following:
`[HAM]MING | [HAN]NING | [C]OSINE | [R]ECTANGLE | [T]RIANGLE`

INPUT

SECONDS v: Set the window length in seconds to v.
LAGS n: Set the window length in lags to n.
NUMBER n: Set the number of points to be used in the spectral estimate.
TYPE type: Set type of window to be used. The advantages of each type is discussed in the writeup of the COR command.

DEFAULT VALUES

" PDS TYPE HAMMING "

DESCRIPTION

This command implements the "conventional" spectral estimator. It is the simplest the sample correlation function is first windowed with a correlation window, and the resulting function is transformed with an FFT to obtain the spectral estimate. As mentioned in the documentation on the COR command, there is a tradeoff between the bias of the estimate, primarily expressed in loss of resolution, and the variance of the estimate. As the window is made longer, the bias is reduced, since frequency-domain resolution is increased. However, the variance of the spectral estimate is increased, since the variance of the sample correlation function values is larger at larger lags. This occurs because fewer data points are used to estimate the values at larger lags.

The choice of correlation window type has a different effect than that of the choice of data window described in the COR documentation. It is a choice between two types of bias.

The spectral estimate approaches the convolution of the true spectrum with the Fourier transform of the correlation window. The window transform is characterized by a central lobe, which controls resolution, and sidelobes, which cause out-of-band power leakage. Typically one wants a narrow main lobe and small sidelobes. Large sidelobes tend to put an artificial, high regular "floor" on the spectral estimate, that can mask the rolloff of a spectrum with high dynamic range. The choice of window type trades off main lobe resolution against power-leakage through the sidelobes.

The rectangular window has the narrowest main lobe, and, therefore, the best resolution. However, it has the largest sidelobes. The cosine taper window reduces the sidelobes slightly without affecting the main lobe width much. These two windows were primarily included for estimating the spectra of transients, which requires little time-domain distortion. The Hamming and Hanning windows are popular windows which have small sidelobes and rather wide main lobes. They are useful when the user has a lot of data, and can control resolution by increasing the window size. Both are raised cosine windows, but the Hamming window is optimized to minimize the size of the largest sidelobe.

It is generally to be prefered, and is the default window in this command. The triangular window also has rather good sidelobe structure, but has the especially desirable property that it guarantees that the spectral estimate will always be positive or zero.

Generally, PDS is to be prefered over the two parametric methods, MLM and MEM, when the user has a large data set available. This is because resolution is not constrained in that situation, and much more is known about this estimator than is known about the others. For example, the theory is available which allows us to estimate confidence limits, and the resolution of the method. Both of these diagnostics are included in SPE. The parametric methods generally exhibit better resolution than PDS, especially when estimating line spectra, and are more useful when a limited amount of data is available.

ERROR MESSAGES

- 5003: No correlatin function calculated.

SEE COMMANDS

COR, WRITESPE, PLOTSPE

PLOTCOR

SUMMARY

Plots the correlation function.

SYNTAX

" [P]LOT[COR] {[X]LIM v|ON|OFF} {PRINT {pname} } "

INPUT

XLIM v: Turn partial x limits option on and set upper limit to v seconds. The lower limit is always 0.

XLIM {ON}: Turn partial x limits option on and use previous upper limit.

XLIM off: Turn partial x limits option off. All of the correlation function is plotted.

PRINT {pname}: Print the resultant plot. If a printer name is specified, print to that printer, else use default printer.

DEFAULT VALUES

" PLOTCOR XLIM OFF"

ERROR MESSAGES

- 5003: No correlation function calculated.

SEE COMMANDS

COR

PLOTPE

SUMMARY

Plots the RMS prediction error function.

SYNTAX

" [P]LOT[PE] "

DESCRIPTION

This command produces a diagnostic plot which may be used to select the order of the MEM spectral estimate. The plot is of the normalized prediction error function, displayed as a function of the order of the estimator. Typically, the prediction error is large for small orders, but decreases rapidly as the order is increased. The prediction error is the "residual power" left after the prediction filter is applied to the data. When this quantity is small, so the theory goes, most of the structure in the spectrum has been captured in the power frequency response of the filter. The residual data is white noise. Consequently, one may examine the prediction error function for "knees" in the curve, where the function drops dramatically to some value that is not reduced much further by further increases in order. The order of the predictor at the "knee" is often used as the order of the MEM spectral estimator.

ERROR MESSAGES

- 5003: No correlation function calculated.

PLOTSPE

SUMMARY

Plots the spectral estimate.

SYNTAX `[P]LOT[SPE] {[P]OWER|[L]OG|[A]MPLITUDE} {[C]ONFIDENCE {ON|OFF}}`

INPUT

POWER: Plot the power response using linear interpolation.
LOG: Plot the power response using logarithmic interpolation.
AMPLITUDE: Plot the amplitude response.
CONFIDENCE {ON}: Include confidence limits on the plot.
CONFIDENCE OFF: Do not include confidence limits.

DEFAULT VALUES

```
PLOTSP POWER CONFIDENCE OFF
```

DESCRIPTION

The plot includes a legend describing the parameters used to calculate the correlation function and the spectral estimate.

ERROR MESSAGES

- 5004 No spectral estimate claculated.

QUITSUB

SUMMARY

Terminates a SAC subprocess.

SYNTAX

```
[Q]UIT[S]UB
```

DESCRIPTION

SAC has several subprocesses which act like separate programs. Use this command to exit one of these subprocesses and return to the main SAC program.

READCOR

SUMMARY

Reads data from a SAC data file into memory.

SYNTAX

```
READCOR file
```

INPUT

file: A legal filename.

DESCRIPTION

This command works just like the READ command in the main SAC program with two exceptions.

First, only ONE file may be read in while in SPE. Second, executing this command deletes any correlation function or spectral estimate that may already have been computed. Parameters within SPE, such as the number of prewhitening coefficients or the window type and length, are not changed when this command is executed.

To reinitialize all SPE parameters, terminate the subprocess using the QUITSUB command and then start it over again.

SEE COMMANDS

QUITSUB

WRITECOR

SUMMARY

Writes a SAC file containing the correlation function.

SYNTAX

" [W]RITE[COR] {file} "

INPUT

file: The name of the SAC file to write.

DEFAULT VALUES

" WRITECOR COR "

DESCRIPTION

The structure of the correlation function written out by this command is determined by the algorithm used to compute it. Since the data is partitioned into windows, and sample correlation functions are calculated from each window, then averaged, the length of the correlation function is determined by the data window size. It contains exactly one less sample than twice the number of samples in the data window. However, since FFT's are used to calculate the sample correlation functions, the number of points in the file is a power of two. It is, in fact, the first power of two larger than the data window size (in samples). The additional samples are zero. The correlation function is also circularly rotated within the file, due to the pecularities of computing correlations with the FFT algorithm. This means that the zero-lag sample is the first sample in the file, and the negative-lag samples follow the positive-lag samples.

ERROR MESSAGES

- 5003: No correlation function calculated.

WRITESPE

SUMMARY

Writes a SAC file containing the spectral estimate.

SYNTAX

" [W]RITE[SPE] {file}"

INPUT

file: The name of the SAC file to write.

DEFAULT VALUES

" WRITESPE SPE "

DESCRIPTION

The spectral estimate file contains the spectral estimate from zero up to the folding frequency. The spectral estimate is calculated with an FFT. The number of points in this file is half the length of the FFT used plus one.

This format was chosen so that multiple spectra computed with SPE could be compared using the P2 plotting function, without any need to cut the files prior to plotting.

ERROR MESSAGES

- 5004: No spectral estimate calculated.