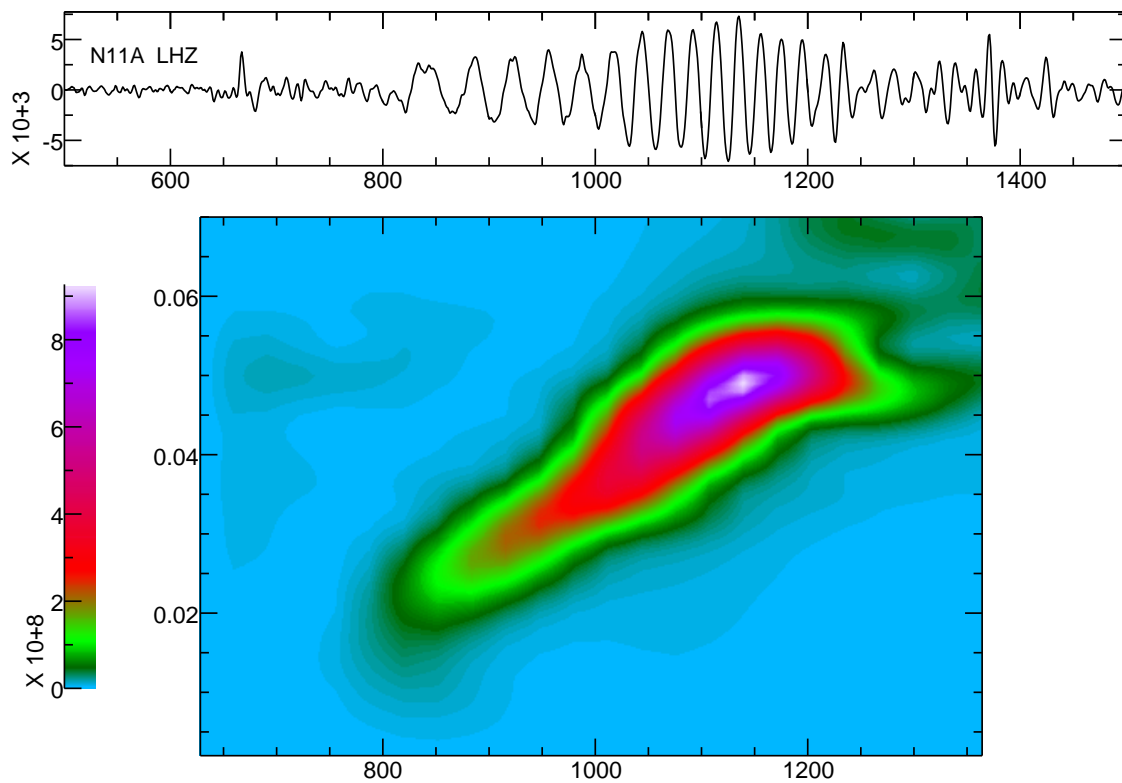


Seismic Analysis Code Users Manual

Version 102.0

December 4, 2020



Sections

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

Contents

1 Using SAC	10
Introduction	10
SAC Tutorial Guide for New Users	12
OVERVIEW	12
GETTING STARTED	13
SAC DATA FILES	14
READING AND WRITING DATA FILES	15
PLOTTING IN SAC	16
FLOATING-POINT PRECISION IN SAC	17
SAC Analysis Capabilities	21
Data Access	27
OVERVIEW	27
TIME FORMATS	28
EVENT SEARCH	28
STATION SEARCH	29
DATA SEARCH	30
INSTRUMENT RESPONSES	33
COMPUTING TRAVELTIMES	33
REDIRECTING SACLST IUTPUT	34
RDSEED - End of Life	35
SAC Macros	36
SAC Inline Functions	44
SAC Data File Format	50
SAC Reading and Writing Routines	65
Using the SAC Libraries	91
Overview	91
Compiling	91
Fourier Transform (FFT)	91
Remove Mean	93
Remove Trend	93
Filtering	94
Cross Correlation	96
Cross Correlation Extras	96
Envelope Calculation	97
Differentiate	98

Integerate	99
Taper Data	100
Cut Data	101
Interpolation using cubic splines	102
Time-Shift	103
Convolution	104
Convolution Primer	105
Blackboard Variables in SAC	108
Graphics in SAC	113
SAC Graphics File	116
Calling SAC from Scripts	119
SAC Output Messages	122
2 SAC Commands	132
Functional Command Listing	132
Alphabetical Comamnd Listing	138
3C	145
ABBREV	146
ABOUT	150
ABS	151
ADD	152
ADDF	153
APK	155
ARRAYMAP	157
AXES	158
BANDPASS	160
BANDREJ	163
BBFK	165
BEAM	167
BEGINDEVICES	169
BEGINFRAME	170
BEGINWINDOW	172
BENIOFF	173
BINOPERR	174
BORDER	176
CAPF	177
CHNHDR	178

CHPF	181
COLOR	182
COMCOR	184
CONTOUR	185
CONVERT	187
CONVOLVE	188
COPYHDR	191
CORRELATE	192
COMMIT , RECALLTRACE , ROLLBACK	195
CUT	198
CUTERR	202
CUTIM	204
DATA	207
DATAGEN	211
DECIMATE	215
DELETECHANNEL	217
DIF	218
DIV	220
DIVF	221
DIVOMEGA	223
ECHO	224
ENDDEVICES	225
ENDFRAME	226
ENVELOPE	227
ERASE	228
EVALUATE	229
EVENT	231
EXP	234
EXP10	235
EXTERNAL COMMAND INTERFACE	237
FFT	241
FILEID	243
FILENUMBER	245
FILTERDESIGN	246
FIR	248
FLOOR	250

FUNCGEN	251
GETBB	253
GRAYSCALE	255
GRID	257
GTEXT	258
HANNING	260
HELP	261
HIGHPASS	263
HILBERT	265
HISTORY	266
Help Introduction	268
IFFT	271
IMAGE	272
INICM	273
INSTALLMACRO	274
INT	275
INTERPOLATE	276
KEEPAM	278
KHRONHITE	279
LINE	280
LINEFIT	282
LINLIN	283
LINLOG	284
LISTHDR	285
LOAD	287
LOADCTABLE	289
LOG	290
LOG10	291
LOGLAB	292
LOGLIN	293
LOGLOG	294
LOWPASS	295
MACRO	297
MAP	298
MARKPTP	300
MARKTIMES	302

MARKVALUE	304
MAT	305
MATHOP	308
MERGE	310
MESSAGE	313
METADATA	314
MTW	319
MUL	320
MULF	321
MULOMEGA	323
PLOTC	324
NULL	328
OAPF	329
OHPF	332
PAUSE	333
PICKAUTHOR	334
PICKPHASE	335
PICKPREFS	336
PICKS	338
PLABEL	340
PLOT	342
PLOT1	343
PLOT2	345
PLOTALPHA	347
PLOTC	350
PLOTCTABLE	354
PLOTDY	355
PLOTPK	356
PLOTPKTABLE	359
PLOTPM	361
PLOTSP	363
PLOTXY	365
PRINT	366
PRINTHELP	367
PRODUCTION	368
QDP	369

QUANTIZE	371
QUIT	373
QUITSUB	374
READ	375
READBBF	380
READCSS	381
READDB	385
READERR	386
READGSE	387
READHDR	389
READSDD	391
READSP	393
READSUDS	395
READTABLE	401
REPORT	405
RESPONSE	407
REVERSE	409
RGLITCHES	410
RMEAN	412
RMS	413
ROTATE	415
ROTINC	417
RQ	420
RTREND	422
SAVEIMG	423
SCALLOP	425
SETBB	427
SETDEVICE	429
SETMACRO	430
SGF	431
SMOOTH	433
SONOGRAM	434
SORT	436
SPECTROGRAM	437
SQR	440
SQRT	441

STATION	442
STRETCH	446
SUB	447
SUBF	448
SYMBOL	450
SYNCHRONIZE	453
SYSTEMCOMMAND	455
TAPER	456
TICKS	458
TITLE	460
TRACE	462
TRANSCRIPT	464
TRANSFER	467
INTRODUCTION	467
FROM EVALRESP	468
FROM POLEZERO	470
FROM FAPfile	473
AVAILABLE INSTRUMENT TYPES	473
INSTRUMENT DETAILS IN TRANSFER	477
TRAVELTIME	479
TSIZE	483
UNSETBB	485
UNWRAP	486
VSPACE	488
WAIT	489
WHITEN	490
WHPF	491
WIDTH	492
WIENER	494
WILD	496
WINDOW	499
WRITE	501
WRITEBBF	505
WRITECSS	506
WRITEGSE	508
WRITEHDR	509

WRITESDD	510
WRITESP	511
XDIV	513
XFUDGE	514
XFULL	515
XGRID	516
XLABEL	517
XLIM	519
XLIN	520
XLOG	521
XVPORT	522
YDIV	523
YFUDGE	524
YFULL	525
YGRID	526
YLABEL	527
YLIM	529
YLIN	530
YLOG	531
YVPORT	532
ZCOLORS	534
ZLABELS	535
ZLEVELS	537
ZLINES	538
ZTICKS	539
3 Signal-Stacking Subprocess	540
Signal Stacking Subprocess	540
ADDSTACK	542
CHANGESTACK	545
DELETSTACK	547
DELTACHECK	548
DISTANCEAXIS	549
DISTANCEWINDOW	550
GLOBALSTACK	551
INCREMENTSTACK	552
LISTSTACK	553

PLOTRECORDSECTION	554
PLOTSTACK	556
SUMSTACK	557
TIMEAXIS	558
TIMEWINDOW	559
SSSTRAVELTIME	560
VELOCITYMODEL	561
VELOCITYROSET	563
WRITESTACK	564
ZEROSTACK	565
4 Spectral-Estimation Subprocess	566
Spectral Estimation (SPE)	566
COR	570
MEM	572
MLM	573
PDS	574
PLOTCOR	576
PLOTPE	577
PLOTSPE	578
QUITSUB	579
READCOR	580
WRITECOR	581
WRITESPE	582

1 Using SAC

Introduction

Overview

SAC (Seismic Analysis Code) 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 being developed and maintained by a small group of developers working in cooperation with IRIS.

In all versions of SAC through and including v101.6a, all reals in the SAC header/data are single-precision (32 bits; 4 bytes). SAC v102.0 includes double-precision for both time and distance calculations using header variables, and does so in a way that maximizes compatibility so that existing SAC data files can still be used. There is a detailed discussion in the section on *Floating-Point Precision in SAC* in [TUTORIAL](#).

Starting in version 102.0, SAC is capable of viewing and downloading event, station, and response metadata and downloading waveforms from the IRIS online Web services. See [Data Access](#) for an overview.

Through version 101.6a, epicentral distances were calculated using an Earth spheroid defined by a: 6378160.0 m and f: 0.00335293. In version 102.0, a header variable IBODY has been added to allow for distance calculations on terrestrial planets, the Sun, and the Moon. Options for IBODY are given in the *Event Field* section of [SAC Data File Format](#), and an example (macro and output) has been added in `${SACHOME}/macros`.

When testing and demonstrating SAC commands, it is useful to have an easily-accessible set of seismograms. The one seismogram most used is from command `FUNCGEN: fg seismo`. However, all one gets is the first several seconds of a not-very-interesting *P* arrival. Command `DATAGEN` has given access to seismograms from three events: a local event, a regional-distance event, and an epicentral event. In SAC v102.0, a new event has been added: `${SACHOME}/aux/datagen/SUB/DEEP/`: three-component seismograms from eight broadband stations. The arrivals are impulsive, and one can see many converted phases. The help files [SAC_MACROS](#) and [TRAVELTIME](#) have examples using `DATAGEN` seismograms. (The `TRAVELTIME` command has been improved and expanded in version 102.0.)

The 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@lists.ds.iris.edu. Although the version of the manual that comes with the binary 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

- [Introduction \(this file\)](#)
- [Tutorial for New Users](#)
- [SAC Analysis Capabilities](#)
- [Data Access](#) (Get Data, Events, Stations, Responses, Traveltimes, ...)
- [SAC Macros](#)
- [SAC Inline Functions](#)
- [SAC Data File Format](#)
- [SAC Reading and Writing Routines](#)
- [Using the SAC Libraries](#)
- [Blackboard Variables in SAC](#)
- [Graphics in SAC](#)
- [SAC Graphics File \(SGF\)](#)
- [Calling SAC from Scripts](#)
- [SAC Output Messages](#)

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.

LATEST REVISION

Version 102.0

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).

Within a single SAC session, the effect of a parameter-setting command remains in effect until that parameter 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 (for that particular 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 [sac data file format](#) for details. Signals are read from disk into memory using the `READ` command. CSS 3.0 formatted 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 single-precision 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.)

T_n: Time markers, where n is an integer from 0 to 9.

ANALYSIS CAPABILITIES

[SAC Analysis Capabilities](#) provides an introduction to these features. [Command Reference Manual](#) lists all the commands and has links to help files for them.

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 impulse
```

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.

For a complete listing of the abbreviations, please consult the [ABBREV](#) help file or run **help abbrev** while in SAC

CHANGING OPTIONS IN REPEATED CALLS TO SAC COMMANDS

As an example, 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:

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

This 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 can use the [PLOT](#) command to plot this function also.

When you specify a value for a particular option, this becomes the new current value, which 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 remembers sine as the function called 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 most recently used value in the current session. (Sometimes one forget that an option had been used, so one may not correctly anticipate correctly the result of the operation.)

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 frequencies 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 response 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. Section [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, unless 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:

NVHDR: If 7, Double-precision used for internal calculations with header variables
B: Beginning 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 AND WRITING DATA FILES

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 to 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 similar 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 disk before 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.

PLOTTING IN SAC

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.

Additional 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 similar 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.

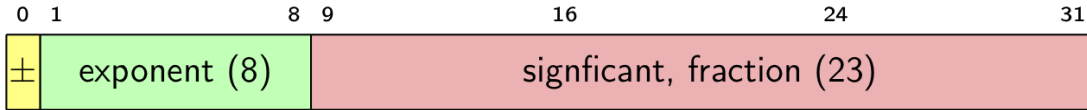


Figure 1: Figure (1): the bit structure of a single-precision floating-point number.

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 from the [Command Reference Manual](#).

An Overview of Graphics Capability in SAC

File [Graphics in SAC](#) has an overview of graphics in SAC. Of particular interest may be the command [SAVEIMG](#) that allows one to save displayed plots in several formats.

FLOATING-POINT PRECISION IN SAC

Background

In all version of SAC through and including v101.6a, all reals in the SAC header/data are single-precision (32 bits; 4 bytes). The smallest time increment within SAC I/O is 0.001 s (header variable NZMSEC is milliseconds). When SAC was written 30+ years ago, a sampling rate of 0.01 s was about as high as the technology allowed, and single SAC files used in analysis were at most several minutes in duration. Now thanks to GPS and advances in both hardware and software, many experiments routinely have sampling rates of 0.001 s, and single SAC data files can be up to hours/days in duration. If SAC is to remain useful for such data sets, single-precision is not sufficient. SAC v102.0 includes double-precision for both time and distance calculations using header variables, and does so in a way that maximizes compatibility so that existing SAC data files can still be used.

We discuss floating-point precision in SAC and give examples showing errors at large times due to the limitations of single-precision for header variables and how these have been resolved using the modified SAC data file structure introduced in v102.0.

Single-Precision in SAC

In a 32-bit floating-point number, 23 bits define the maximum precision. For all floating-point numbers with a size between 2^{n-1} and 2^n floating-point values are separated by $\delta(n)$, given by

$$\delta(n) = \frac{2^{n-1}}{2^{23}} = \frac{t(n)_{ref}}{2^{23}},$$

where, in our applications of the above equation, $\delta(n)$ and the reference time $t(n)_{ref}$ are times in seconds. For $n=17$, $t(17)_{ref} = 65,536$ s and $\delta(17) = 0.0078$ s. Incrementing n by 1 doubles the floating-point increment.

In the examples below, we use the following macro that sets DELTA = 0.01 s and B = 48 hours = 172,800.00 s, (for which $t(18)_{ref} = 131,072$ s and the floating-point increment $\delta(18) = 0.0156$ s = 156% DELTA):

```
fg sine 2.0 delta 0.01 npts 13
symbol 2
fileid location ul
ch b 172800
ch t0 (&1,b + &1,delta * 1)
```

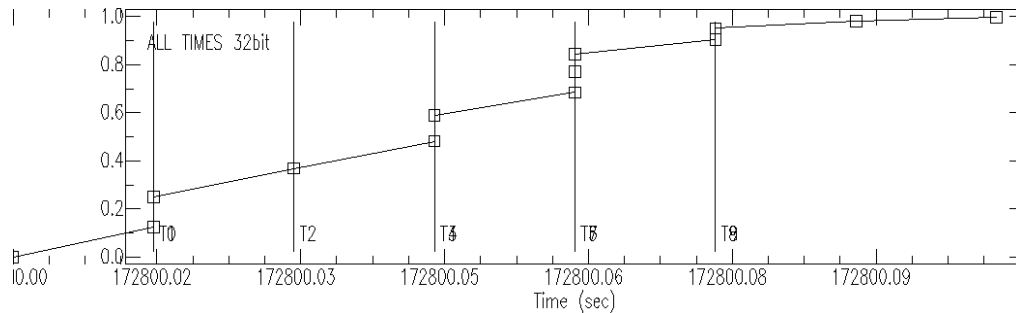


Figure 2: Figure (2): SAC v101.6a using the above macro.

```

ch t1 (&1,b + &1,delta * 2)
ch t2 (&1,b + &1,delta * 3)
ch t3 (&1,b + &1,delta * 4)
ch t4 (&1,b + &1,delta * 5)
ch t5 (&1,b + &1,delta * 6)
ch t6 (&1,b + &1,delta * 7)
ch t7 (&1,b + &1,delta * 8)
ch t8 (&1,b + &1,delta * 9)
ch t9 (&1,b + &1,delta * 10)
lh picks

```

From the macro, we see that waveform times (squares) should be at 176,800.00, 176,800.01, ... ; and the ten time picks should be at 176,800.01, 176,800.02,

In v101.6a, SAC used double-precision for some internal calculations, but not in calculations used to increment times when plotting waveforms or in producing the time labels in plots. In Fig. (2), the time picks and the symbols are at the same times, which is because they were both calculated using single precision. (Even though the time-plot labels in Figure 2 are distracting, we can ignore them as they were not calculated the same way as the plotted times.)

The separation between all the squares is supposed to be $\text{DELTA} = 0.01$ s, but $\delta(18) = 0.0156$ s is the minimum separation between neighboring values. The first time point is at 176,800.00 s. The plotted time for the symbol of the next point will be at the nearest floating-point neighbor to 176,800.01 s, which is 176,800.0156 s rather than 176,800.00 s so that is where the next symbol is plotted. 176,800.02 s is closer to 176,800.0146 s than to 176,800.0312 s, so that symbol is at the same time as the previous symbol – and time picks T0 and T1 overlap at that same time as the symbols).

In 2017 the problems with the time plot labels and waveform times were fixed, which was possible with the existing SAC data-file format because the calculations of those times are done *internally* so could be done in double precision. Because B, DELTA, T0 ... T9 are header variables, they could not be changed to double precision.

Figure (3) uses double precision for the internal calculations for times but uses the same procedure as used in Fig. (2) for time picks. The waveform time symbols (and the time axis) are now "correct", but the time picks are unchanged from Figure 2.

Upgrade To Double Precision

Header variable NVHDR is the SAC version number. For the past several decades $\text{NVHDR} = 6$. In SAC v102.0, the file structure has been changed to address the precision problem. The change is done in such a way that existing SAC files can be read and processed as before.

The file structure when NVHDR = 7 is discussed in [sac data file format](#); when NVHDR = 7 there is a "footer" in the file structure that contains double-precision versions of 22 time/distance (single-precision) header variables.

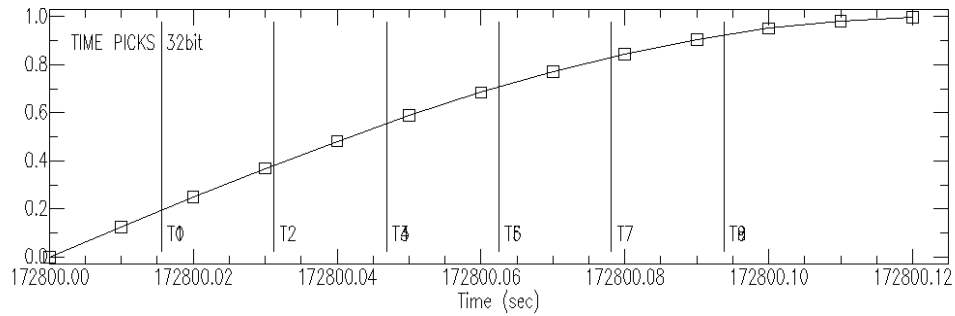


Figure 3: Figure (3): SAC v101.6a but double-precision internal calculations for times.

These 22 variables are DELTA, B, E, O, A, T0 ... T9, F, EVLO, EVLA, STLO, STLA, and internal variables SB and SDELTA. The names are the same as in the header; SAC distinguishes between the single- and double-precision variables in internal calculations by their location in the file.

If a SAC file with NVHDR = 6 is read into memory using SAC v102.0, double-precision copies of those 22 header variables are stored in memory, but the single-precision version of those header variables is still used so long as NVHDR = 6. However, if one enters *CH NVHDR 7*, one has the potential of changing the precision of variable like DELTA for internal calculations.

Important

Switching a file from **NVHDR = 6** to **7** does not magically produce a proper double-precision value. Values stored in 32 and 64 bits will only be modified when a value is changed through [CHNHDR](#) or equivalent methods.

SAC v102.0 introduces *OUTPUT_FORMAT*, which can change the format for floating-point variables in SAC commands [LISTHDR](#), [MESSAGE](#), and [EVALUATE](#).

Setting **NVHDR = 7** leads to time picks being calculated and stored in double precision. Using the default for *OUTPUT_FORMAT* does not show enough significant figures to distinguish between the values for NVHDR 6 or 7. Using *longE* for the output format shows the difference:

```
SAC> fg impulse
SAC> setbb OUTPUT_FORMAT longE
SAC> ch t0 (1.0/3.0)
SAC> lh nvhdr t0
nvhdr = 6
t0 = 3.333333432674408e-01
SAC> ch nvhdr 7
SAC> lh nvhdr t0
nvhdr = 7
t0 = 3.333333333333333e-01
```

If one ran the above macro in SAC v102.0, one gets the same result as shown in Fig. (3). If one then enters *CH NVHDR 7 ; P1* one gets the plot shown in Fig. (4). If one then enters *SETBB OUTPUT_FORMAT longE ; LH*, one can see the differences in T0, T1, T2, etc. See [GETBB](#) for more information about *OUTPUT_FORMAT*.

In Figure 4, the times/picks, and tic-marks are all lined up.

For an unevenly-spaced, time-series file (IFTYPE=ITIME[1] and LEVEN=False[0]), the time data are single-precision, so for times $\geq 1e7$ seconds, the times will be incorrect.

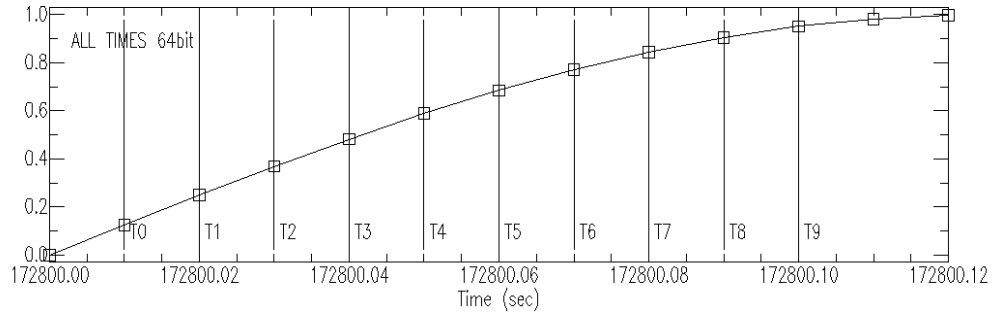


Figure 4: Fig. (4): SAC v102.0 with *ch NVHDR 7* preceding *P1*.

Distance And Geographic Variables Precision

There are several distance-related variables in the header, but DIST, AZ, BAZ, and GCARC are not used as input for internal computation; they are derived from the header variables EVLA, EVLO STLA, and STLO,, which is why only those four are in the footer.

Distance precision using real^*4 (32-bits) is less restrictive than for time precision. Half the circumference of the Earth is about 20,000 km, so if the numerator in the right-hand side of the above equation is a distance reference in meters, $\Delta(n)_{ref}$, the half circumference has a distance reference $\Delta(24)_{ref}$ with a $\delta(24) = 2$ m. The largest relative latitude or longitude for that distance would be about 180 degrees in longitude at the equator, which has a similar reference. A more significant limitation for such distances is probably the accuracy of the reference ellipsoid. Promoting the station and hypocenter latitudes and longitudes removes any potential distance-related precision problems.

Dynamic Range

20 years ago, 16 bits was the standard for data. Today 24 bits is standard. $2^{24} = 16,777,216$. Using single precision for the SAC amplitude data is sufficient.

Maximum Number Of Points

Integer header variable **NPPTS** has 4 bytes (32 bits). The SAC header uses signed integers, so the maximum value is $2^{31} - 1 = 2,147,483,647$. If the sampling rate is 0.001 s, the maximum record length is 24.85 days.

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 successfully 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 four events (one local, one regional, one teleseismic and shallow, and one teleseismic and deep) 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. Examples in help files [SAC_MACROS](#) and [TRAVELTIME](#) use the deep teleseismic event.

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 indigenous 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 control 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](#) applies [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 transferring 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 it 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 integration (**INT**) and differentiation (**DIF**).

Binary-Operations Module

These commands perform operations on pairs of data files.

- **MERGE** merges (concatenates) 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.

Neural 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.

Data Access

OVERVIEW

Starting in version 102.0, SAC is capable of viewing and downloading event, station, and response metadata and downloading waveforms from online Web services. These abilities include

- event searches ([EVENT](#))
- station searches ([STATION](#))
- data: searches and downloading ([DATA](#))
- applying station and/or event meta-data to data ([METADATA](#))
- reading miniseed directly into SAC ([READ](#))
- download instrument responses ([RESPONSE](#))
- travelttime data using the the IRIS Traveitime Web Service ([TRAVELTIME](#))

Definitions

Currently (2020), as described on the IRIS DMC Web Service Web page <<https://service.iris.edu>>:

- **miniSEED** is the subset of the SEED standard that is used for time series data. Very limited metadata for the time series is included in miniSEED beyond time series identification and simple state-of-health flags. In particular, geographic coordinates, response/scaling information and other information needed to interpret the data values are not included.
- **FDSN StationXML** is the XML-based standard for seismological metadata. These metadata contain the most important and commonly used structures of SEED 2.4 metadata with many enhancements.
- **PH5** is the name of the repository at the IRIS DMC commonly used for active-source, controlled-source and mixed-source data sets and is an alternative to the DMC's SEED repository. In [DATA](#), [METADATA](#), and [STATION](#) ph5 is an option that when invoked will search this alternate repository.

Web requests are made to the collection of Web Services at <http://service.iris.edu> along with event searches through the [USGS](#) and [ISC](#).

An **EventID** is defined by *source:uniqueid*. See [EVENT](#) for more details.

Event search results, [EVENT](#), can be used as input for timing and location in [DATA](#) and [STATION](#) searches using associated *EventID*. Examples are given below.

The format for latitudes and longitudes is degrees.decimal-degrees. Longitudes run from -180.00 degrees to 180.00 degrees. The formats used for time are given at the end of this file.

For repeated calls to most SAC command in a single session, the program "remembers" previous values for options if not reset explicitly. That is **not** true for these commands; For any option not explicitly stated in each call, the default is assumed. If there is no default value for an option, it must be explicitly given.

The SAC convention for units differs from others such as EVALRESP and SEED that use SI units (meters, meters/sec, meters/sec/sec). SAC uses nm for displacement. As discussed in [TRANSFER](#), unit conversions may be needed to convert output files from instrument-corrections to get consistency between the data and the SAC header parameter IDEP.

SACLST & SACSET: These two stand-alone programs allow one to parse (*saclst*) or modify the header of one or more SAC files. They are analogous to the internal SAC commands [LISTHDR](#) and [READHDR](#) respectively. One can display the results on the screen, print them, or (for *saclst*) redirect the output into an application. See `${SACHOME}/utils/README_utils` for more details. Examples are given below showing how *saclst* output can be redirected to applications.

TIME FORMATS

Times are of the form *h:m:s* or *h:m:s.ms* where *h* is the hour, *m* the minute, *s* the seconds, and *ms* the milliseconds. If the time is left out, the beginning of the day is used, i.e. *00:00:00.000*.

Dates are in the form *Y-M-D* or *Y-O* where *Y* is the year, *M* the month, *D* the day of the month, and *O* the day of the year (or ordinal day). Separators for the date components include */*, *,*, and *-*. Separators between the date and time is either a *T* or the date separator.

Date Only	Date and Time	Date and Time
Y/O	Y/OTh:m:s Y/O/h:m:s	Y/OTh:m:s.ms Y/O/h:m:s.ms
Y-O	Y-OTh:m:s Y-O-h:m:s	Y-OTh:m:s.ms Y-O- h:m:s.ms
Y,O	Y,OTh:m:s Y,O,h:m:s	Y,OTh:m:s.ms Y,O,h:m:s.ms
Y/M/D	Y/M/DTh:m:s Y/M/D/h:m:s	Y/M/DTh:m:s.ms Y/M/D/h:m:s.ms
Y-M-D	Y-M-DTh:m:s Y-M-D- h:m:s	Y-M-DTh:m:s.ms Y-M- D-h:m:s.ms
Y,M,D	Y,M,DTh:m:s Y,M,D,h:m:s	Y,M,DTh:m:s.ms Y,M,D,h:m:s.ms

Durations are of the form *[+]-###unit* where *+* is time forward and *-* is a time backwards from the reference time, *unit* are given below, and the *###* is an integer value.

Duration	Abbreviations
Seconds	s sec secs seconds
Minutes	m min mins minutes
Hours	h hrs hours
Days	d days
Weeks	w wk wks weeks
Months	mon months
Years	y yr yrs years
Decades	dec decades
Centuries	cent centuries

EVENT SEARCH

Broad search

Let's find all $M \geq 9$ earthquakes within the past century (100 years), assuming the date today is Oct-31-2019. This will search the USGS Earthquake catalog by default:

```
SAC> event time 2019-10-31 -100years mag 9
Origin          Lat.   Lon.   Depth  Mag.   Agency EventID
2011-03-11T05:46:24 38.30 142.37 29.00 9.10 mww US/official -
usgs:official20110311054624120_30
2004-12-26T00:58:53  3.29  95.98 30.00 9.10 mw  US/official -
usgs:official20041226005853450_30
```

```

1964-03-28T03:36:16 60.91 -147.34 25.00 9.20 mw iscgem/official -
usgs:official19640328033616_30
1960-05-22T19:11:20 -38.14 -73.41 25.00 9.50 mw iscgem/official -
usgs:official19600522191120_30
1952-11-04T16:58:30 52.62 159.78 21.60 9.00 mw iscgem/official -
usgs:official19521104165830_30

```

Using the NOW keyword

Using the *NOW* keyword in place of the first date is useful in scripts so one does not have to explicitly enter today's date. Let's use *NOW* and also change the catalog to Global CMT:

```

SAC> event time now -100years mag 9 gcmt
Origin          Lat.   Lon.   Depth  Mag.   Agency EventID
2011-03-11T05:47:32 37.52 143.05 20.00 9.10 MW GCMT/- GCMT gcmt:3279407
2004-12-26T01:01:09 3.09  94.26 28.60 9.00 MW GCMT/- GCMT gcmt:1916079

```

Note that the results from Global CMT, compared to the USGS, do not have the older events and the prefix for the event id is also different.

Narrow search

If we are interested in a particular event, an event time window, or depth range, that can be specified directly:

```

SAC> event time 1994/1 1994/365 mag 8 depth 600 1000 usgs
Origin          Lat.   Lon.   Depth  Mag.   Agency EventID
1994-06-09T00:33:16 -13.84 -67.55 631.30 8.20 mw US/HRV - usgs:usp0006dzc

```

STATION SEARCH

Rectangular Search in a Network

Looking for available stations is straightforward by using a region, network name, station name and channels. Here we only use a small region and the network *IU*:

```

SAC> station region -130 -110 45 48 net IU
Warning: Multiple instances of net.sta, likely mutiple epochs
Net Sta  Lat.   Lon.   Elev.  SiteName
IU  RAI0  46.0403 -122.8851 1.00 Rainier, Oregon, USA

```

Radial Search with a time window in a Network

If a particular time range or distance range is of interest, that may be specified as well:

```

SAC> station network IU channel ?HZ time 2011-03-11 1day radial 30 35
origin 143.05 37.53
Net Sta  Lat.   Lon.   Elev.  SiteName
IU  ADK    51.8823 -176.6842 130.00 Adak, Aleutian Islands, Alaska
IU  BILL   68.0653 166.4531 320.00 Bilibino, Russia
IU  DAV     7.0697 125.5791 150.00 Davao, Philippines
IU  MIDW   28.2156 -177.3698 20.00 Midway Island, USA
IU  TIXI   71.6341 128.8667 40.00 Tiksi, Russia

```

Using an EventID

The previous search was for stations operating during the 2011 Tohoku event with an epicentral distance 30 and 35 degrees. This search can be simplified by using and event id:

```
SAC> event mag 9 time 2011/1 +1y gcmt
Origin          Lat.   Lon.   Depth  Mag.   Agency EventID
2011-03-11T05:47:32 37.52 143.05 20.00 9.10 MW  GCMT/- GCMT gcmt:3279407
```

```
SAC> station net IU cha ?HZ gcmt:3279407 radial 30 35
Net Sta  Lat.   Lon.   Elev.  SiteName
IU  ADK    51.8823 -176.6842 130.00 Adak, Aleutian Islands, Alaska
IU  BILL   68.0653 166.4531 320.00 Bilibino, Russia
IU  DAV     7.0697 125.5791 150.00 Davao, Philippines
IU  MIDW   28.2156 -177.3698 20.00 Midway Island, USA
IU  TIXI   71.6341 128.8667 40.00 Tiksi, Russia
```

DATA SEARCH

Data for 1 hour after event origin

Once the event or time range and stations are identified, downloading data is straight forward. First, the event is determined; here we use the 1994 deep Bolivian event and a station named *DOOR*. Data requests need to specify where the data should be placed; this includes *miniSEED* for directly saving the miniSEED files, *SAC* for converting the miniSEED files into SAC binary format, and *READ* for placing the data directly into memory. Any combination of these options can be used. We also specify the duration of the data be downloaded using the *DURATION* option:

```
SAC> event time 1994/160 1d mag 8 gcmt
Origin          Lat.   Lon.   Depth  Mag.   Agency EventID
1994-06-09T00:33:45 -13.82 -67.25 647.10 8.20 MW  GCMT/- GCMT gcmt:369471
```

```
SAC> data sta DOOR cha BHZ gcmt:369471 duration +1h
## REQUEST 1/ 1
DATACENTER=IRISDMC,http://ds.iris.edu
XE DOOR -- BHZ 1994-06-09T00:33:45 1994-06-09T01:33:45
```

No Data Downloaded, use miniseed, sac, or read to download data

```
SAC> data sta DOOR cha BHZ gcmt:369471 dur +1h read
Data Center: IRISDMC,http://ds.iris.edu
          SourceID          Start sample  End sample          Gap
XFDSN:XE_DOOR__B_H_Z      1994-06-09T00:33:45.012000
1994-06-09T01:33:44.912000  ==
Working on file: XE.DOOR..BHZ.M.1994.160.003345.sac [ OK ]
```

Data download using a station file

If a stations file exists, data for these can be directly downloaded. Let's create the stations file, then download the data. We also save the data request using the *OUT* parameter in *DATA*, then immediately use that request file

Create a station file call net_BK.txt:

```

SAC> station net BK region -125 -110 30 40 sta B*,C* out net_BK.txt
Warning: Multiple instances of net.sta, likely mutiple epochs
Net Sta   Lat.     Lon.     Elev.   SiteName
BK BABI   36.5158 -120.8530 571.20 Babies Gulch, near Pinnacles National
Park, USA
...
BK CVS    38.3453 -122.4584 295.10 Carmenet Vineyards, Sonoma, CA, USA

```

Create a request file from the station file:

```

SAC> data time 2010,235T00:00:00 1hr in net_BK.txt cha BHZ out
net_BK_request.txt
Reading station file: net_BK.txt
## REQUEST 1/ 1
DATACENTER=NCEDC,http://www.ncedc.org
BK BDM -- BHZ 2010-08-23T00:00:00 2010-08-23T01:00:00
...
BK CVS 00 BHZ 2010-08-23T00:00:00 2010-08-23T01:00:00

No Data Downloaded, use miniseed, sac, or read to download data

```

Use the request file to download data, this will automatically download to miniSSED:

```

SAC> data request net_BK_request.txt
Data Center: NCEDC,http://www.ncedc.org
Writing data to fdsnws.2019.10.25.18.01.01.NCEDC.mseed [956.00 KiB]

```

Magic incantations to get the most recent miniSEED file and read it. These can be avoided by using there READ option in DATA:

```

SAC> sc to FILE ls -rt *.mseed | tail -1
SAC> getbb FILE
FILE = 'fdsnws.2019.10.25.18.01.01.NCEDC.mseed'
SAC> read %FILE

```

SourceID	Start sample	End sample	Gap
XFDSN:BK_BDM__B_H_Z	2010-08-23T00:00:00.010466		
	2010-08-23T00:59:59.985466	==	
XFDSN:BK_BKS__B_H_Z	2010-08-23T00:00:00.023145		
	2010-08-23T00:59:59.998145	==	
XFDSN:BK_BRIB_00_B_H_Z	2010-08-23T00:00:00.019538		
	2010-08-23T00:59:59.994538	==	
XFDSN:BK_BRK_00_B_H_Z	2010-08-23T00:00:00.019538		
	2010-08-23T00:59:59.994538	==	
XFDSN:BK_CMB__B_H_Z	2010-08-23T00:00:00.023616		
	2010-08-23T00:59:59.998696	==	
XFDSN:BK_CVS_00_B_H_Z	2010-08-23T00:00:00.019536		
	2010-08-23T00:59:59.994539	==	

```

SAC>

```

Applying metadata automatically

If data is downloaded and converted to sac binary or read into memory, SAC will try to insert metadata for the resulting data files.


```

SAC> data sta DOOR cha BHZ gcmt:369471 dur +1h read
Data Center: IRISDMC,http://ds.iris.edu
  SourceID          Start sample          End sample          Gap
XFDSN:XE_DOOR__B_H_Z  1994-06-09T00:33:45.012000
1994-06-09T01:33:44.912000  ==
Working on file: XE.DOOR..BHZ.M.1994.160.003345.sac [ OK ]

```

```

SAC> lh kevmn stlo stla evlo evla evdp

```

```

FILE: XE.DOOR..BHZ.M.1994.160.003345.sac - 1
-----

```

```

kevmn = gcmt:369471
stlo = -6.722330e+01
stla = -1.935380e+01
evlo = -6.725000e+01
evla = -1.382000e+01
evdp = 6.471000e+02

```

Applying metadata manually

If data is downloaded without metadata, it is simple to insert that metadata using the [METADATA](#) command:

```

SAC> data sta DOOR cha BHZ time 1994-06-09T00:33:45 1h read
Data Center: IRISDMC,http://ds.iris.edu
  SourceID          Start sample          End sample          Gap
XFDSN:XE_DOOR__B_H_Z  1994-06-09T00:33:45.012000
1994-06-09T01:33:44.912000  ==
Working on file: XE.DOOR..BHZ.M.1994.160.003345.sac [ OK ]

```

```

SAC> lh kevmn stlo stla evlo evla evdp

```

```

FILE: XE.DOOR..BHZ.M.1994.160.003345.sac - 1
-----

```

```

stlo = -6.722330e+01
stla = -1.935380e+01

```

```

SAC> metadata gcmt:369471
Working on file: XE.DOOR..BHZ.M.1994.160.003345.sac [ OK ]

```

```

SAC> lh kevmn stlo stla evlo evla evdp

```

```

FILE: XE.DOOR..BHZ.M.1994.160.003345.sac - 1
-----

```

```

kevmn = gcmt:369471
stlo = -6.722330e+01
stla = -1.935380e+01
evlo = -6.725000e+01
evla = -1.382000e+01
evdp = 6.471000e+02

```

INSTRUMENT RESPONSES

Responses for specific files is also simple with files read into memory. Responses can be obtained in sacpz or evalresp format.:

```
SAC> read XE.DOOR..BHZ.M.1994.160.003345.sac
```

```
SAC> response sacpz
```

```
Writing data to SAC_PZs_XE_DOOR__BHZ_1994-06-09T00:33:45_1994-06-09T01:33:44 [ 1.02 K
```

```
SAC> response evalresp
```

```
Writing data to RESP.XE.DOOR..BHZ [ 49.80 KiB]
```

COMPUTING TRAVELTIMES

Traveltimes can also be retrieved and set using the traveltime Web Service at IRIS. Just specific the *ONLINE* keyword to the **TRAVELTIME** command:

```
SAC> data sta CMB net BK cha BHZ gcmt:369471 duration 1h read
```

```
Data Center: NCEDC,http://www.ncedc.org
```

```
SourceID Start sample End sample Gap
```

```
XFDSN:BK_CMB__B_H_Z 1994-06-09T00:33:45.004000
```

```
1994-06-09T01:33:44.953000 ==
```

```
Working on file: BK.CMB..BHZ.D.1994.160.003345.sac [ OK ]
```

```
SAC> traveltime online picks 0 phase ttall
```

```
traveltime: depth: 647.100 km
```

```
traveltime: setting phase P at 618.4500 s [ t = 618.4500 s ] t0
```

```
traveltime: setting phase PKIKKIKP at 1782.5400 s [ t = 1782.5400 s ] t1
```

```
traveltime: setting phase PKIKPPKIKP at 2280.2400 s [ t = 2280.2400 s ] t2
```

```
traveltime: phase PKIKPPKIKP truncated in kt2
```

```
traveltime: setting phase PKPPKP at 2277.7900 s [ t = 2277.7900 s ] t3
```

```
traveltime: setting phase PKiKP at 974.6000 s [ t = 974.6000 s ] t4
```

```
traveltime: setting phase PP at 789.5900 s [ t = 789.5900 s ] t5
```

```
traveltime: setting phase PcP at 632.6300 s [ t = 632.6300 s ] t6
```

```
traveltime: setting phase S at 1128.5800 s [ t = 1128.5800 s ] t7
```

```
traveltime: setting phase SKIKKIKP at 1935.6899 s [ t = 1935.6899 s ] t8
```

```
traveltime: setting phase SKS at 1160.1700 s [ t = 1160.1700 s ] t9
```

```
SAC> lh picks
```

```
FILE: BK.CMB..BHZ.D.1994.160.003345.sac - 1
```

```
-----  
NPTS = 72000  
B = -3.960000e-01  
E = 3.599554e+03  
OMARKER = -2.1458e-09  
T0MARKER = 618.45 (P)  
T1MARKER = 1782.5 (PKIKKIKP)  
T2MARKER = 1997.2 (PKIKKIKS)  
T3MARKER = 2280.2 (PKIKPPKI)  
T4MARKER = 2277.8 (PKPPKP)
```

```

T5MARKER = 974.6          (PKiKP)
T6MARKER = 789.59        (PP)
T7MARKER = 632.63        (PcP)
T8MARKER = 1128.6        (S)
T9MARKER = 1935.7        (SKIKKIKP)
  KZDATE = JUN 09 (160), 1994
  KZTIME = 00:33:45.400

```

Teleseismic Traveltime Example

Given a region of interest to conduct a teleseismic traveltime experiment, first find the events 30 to 90 degrees from the region of interest, Hawaii (Lon: 158W, Lat: 20N). Here we only look for events within the last year and magnitude > 7 to limit the results, typically we use more expansive search parameters. As this uses a do loop, the commands need to be contained within a macro

```

% cat data.sm
event mag 7 radial -158 21 30 90 usgs time now -1years to events
outfile events_usgs_m7.xml

do ev list %events
  message "Event $ev"
  mkdir $ev
  cd $ev
  station $ev cha BHZ outfile stations.txt region -161 -152 18 22
  data $ev cha BHZ in stations.txt out request.txt duration 90m
  miniseed
  cd ..
enddo

% sac
SAC> macro data.sm

```

This should result in a set of directories named with eventids containing a station file, the data request, and miniseed files to be unpacked. Miniseed files can be unpacked, metadata applied, and traveltime data set using:

```

SAC> sc to PWD "basename @$ (pwd) "
SAC> getbb PWD
SAC> read *.mseed
SAC> metadata %PWD
SAC> traveltime online picks 0
SAC> write over

```

REDIRECTING SACLST IUTPUT

Let's say you have a list of sac files and you would like to generate a station list:

```

saclst knetwk kstnm stlo stlo stel f *.SAC | \
awk '{print $2,$3,$4,$5}' | \
sort | \
uniq > station_lists.txt

```

To make sure all event locations are consistent for a list of files:

```
sac1st evlo evla evdp f *.SAC | awk '{print $2,$3,$4}' | sort | uniq
```

Find file that are too short in duration with the longest first:

```
sac1st b e f *.SAC | \
    awk '{print $1,$2,$3,$3-$2}' | \
    sort -nr -k +4
```

Or removing files that are too short in duration:

```
mkdir too_short
for z in $( sac1st b e f *.SAC | awk '{if($3-$2<30.0){print $1}}' ) ; do
    mv $z too_short
done
```

Or using xargs:

```
sac1st b e f *.SAC | awk '{if($3-$2<30.0){print $1}}' | \
    xargs -I args mv args too_short
```

RDSEED - End of Life

Before 2010, the majority of data analysis was done on data sets commonly exchanged as SEED volumes, which could be downloaded from the IRIS Data Management Center over the Internet. For the SEED manual, go to <http://www.fdsn.org/seed_manual/SEEDManual_V2.4.pdf>.

Program RDSEED is a legacy utility for reading SEED volumes. This utility allows the user to read the contents of the SEED file and convert it to many different output formats (including SAC), extracting the data, station information, or instrument responses from the file. Although SEED volumes remains the FDSN-approved file format for representing and distributing digital seismic data, RDSEED is *End of Life* and no longer supported at IRIS. It has now been made an open-source project on *GitHub* <<https://github.com/iris-edu-legacy/rdseed>>. For further details on RDSEED, go to <<http://ds.iris.edu/ds/nodes/dmc/software/downloads/rdseed/5-3-1-eol/>>.

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. Let's say you wanted to read repeatedly the same three files, multiply each file by a different value, plot the results, and save the plot. QUITMACRO exits the macro. The macro file would look like this:

```
* a simple macro
datagen sub deep kev.z kev.r kev.t
rtr
mul 1.0 1.0 1.5
FILEID LOCATION UL TYPE LIST KSTCMP
title "mul 1.0 1.0 1.5"
p1
save macrotest-kev.pdf
quitmacro
```

If the file is named test.sm the command sequence for the above macro would be:

```
SAC> macro test.sm
```

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.

We chose to use data files read in through [DATAGEN](#) so that the lines above can be copied and pasted into a file named test.sm and run by the user. This is true for most of the macros given below.

Order-Dependent Arguments

The above example while simple is not very flexible. 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 allow for different multiplication factors:

```
datagen sub deep $1 $2 $3
rtr
mul $4 $5 $6
FILEID LOCATION UL TYPE LIST KSTCMP
title "mul $4 $5 $6"
p1
quitmacro
```

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> M test.sm kev.z kev.r kev.t 1.0 1.0 1.5
```

Note that the station name appears in the plot because of the *FILEID* command. The macro exits after the *P1* command in this case. The plot is still on the screen, so one can save it if one wants to.

Keyword-Driven Arguments

Keyword-driven arguments let one 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. Let's again modify our example to accept a list of files and also a list of multiplicative values:

```
$KEYS FILES VALUES
datagen sub deep $FILES
rtr
mul $VALUES
FILEID LOCATION UL TYPE LIST KSTCMP
title "mul $VALUES"
p1
quitmacro
```

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> m test.sm VALUES 1.0 1.0 1.5 FILES kev.z kev.r kev.t
```

Note that *VALUES* precedes *FILES* -- the order does not matter. We chose *DATAGEN* for the example because the waveforms are accessible. One could replace the *DATAGEN SUB DEEP* command with *READ*, if the user want to use other 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 1.0 1.0 1.5
datagen sub deep $FILES
mul $VALUES
FILEID LOCATION UL TYPE LIST KSTCMP
title "mul $VALUES"
p1
quitmacro
```

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> m test.sm FILES kev.z kev.r kev.t
```

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

```
SAC> m test.sm VALUES 1.5 1.5 1.2 FILES kev.z kev.r kev.t
```

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 test.sm  
FILES? kev.z kev.r kev.t
```

SAC does not query for a value until it first tries to evaluate the argument and finds that there is no default or input value. Note that SAC did not query for **VALUES** because it had a default set of values. 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 preceding its name with a percent sign, %, as shown below:

```
SAC> SETBB C1 2.45  
SAC> SETBB C2 4.94  
SAC> BANDPASS CORNERS %C1 %C2  
FILEID LOCATION UL TYPE LIST KSTCMP  
title "mul $VALUES"  
p1  
quitmacro
```

Now let's 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 1.0  
datagen sub deep $FILES  
EVALUATE TO VALUE2 $VALUE1 * 1  
EVALUATE TO VALUE3 %VALUE2 + 1.5  
MUL $VALUE1 %VALUE2 %VALUE3
```

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

```
SAC> m test.sm VALUE1 1 FILES kev.z kev.r kev.t
```

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 proceed 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 referred 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 similar 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 preceding 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 gets values from the sac header for each file individually using either the [LISTHDR](#) command or accessing the header values directly. This can be handled with a do loop:

```

DO J = 1, %SACNFILES
  lh fileS $J NPTS DELTA COLUMNS 2
ENDDO

DO J = 1, %SACNFILES
  message "$J          &$J$,NPTS          &$J$,DELTA          &$J$,FILENAME "
ENDDO

```

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
```

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
  ENDF
ENDDO
```

This **WHILE** loop in this macro is an example of a infinite 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 precede 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 referred 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
```

Arithmetic Functions

There are 22 arithmetic functions available. They correspond to the arithmetic functions found in the [EVALUATE](#) command. Each of these functions is described in the following table. Examples are then given.

Command	Syntax	Purpose
SINE	(SINE v)	Take the sine of a number.
ARCSINE	(ARCSINE v)	Take the arcsine of a number.
COSINE	(COSINE v)	Take the cosine of a number.
ARCCOSINE	(ARCCOSINE v)	Take the arccosine of a number.
TANGENT	(TANGENT v)	Take the tangent of a number.
ARCTANGENT	(ARCTANGENT v)	Take the arctangent of a number.
ADD	(ADD v1 v2 ... vn)	$v1 + v2 + \dots + vn$.
SUBTRACT	(SUBTRACT v1 v2 ... vn)	$v1 - v2 - \dots - vn$.
MULTIPLY	(MULTIPLY v1 v2 ... vn)	$v1 * v2 * \dots * vn$.
DIVIDE	(DIVIDE v1 v2 ... vn)	$v1 / v2 / \dots / vn$.
SQRT	(SQRT v)	Take the square root of v (for $v > 0$).
EXP	(EXP v)	Exponentiate a number.
POWER	(POWER v)	$10 ** v$.
ALOG	(ALOG v)	Take the natural logarithm of a number.
ALOG10	(ALOG10 v)	Take the log to base 10 of a number.
INTEGER	(INTEGER v)	Convert a number to an integer.
PI	(PI)	Return the value of pi. e.g., $2 * PI$
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.
GETVAL (1)	(GETVAL t)	Get value at time t for all files in memory
GETVAL (3)	(GETVAL File N t)	Get value for the Nth file at time t
GETTIME	(GETTIME MAX MIN [Val])	See discussion in Example 4 below

Example 1: Normalize Data Normalize a set of data files so that the absolute value of the extremum offset in the data set is 1.0:

```

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> rtr
SAC> lh depmax depmin
FILE: one.sac - 1
    depmax = 1.619626e+00    depmin = -1.470322e+00
FILE: two.sac - 2
    depmax = 3.239252e+00    depmin = -2.940643e+00
FILE: four.sac - 3
    depmax = 1.295701e+01    depmin = -1.176257e+01

```

```

SAC> setbb a (max &1, depmax &2, depmax &3, depmax)
==> setbb a 12.957
SAC> setbb b (min &1, depmin &2, depmin &3, depmin)
==> setbb b -11.7626
SAC> div (max %a (abs %b))
==> div 12.957
SAC> lh depmax depmin
FILE: one.sac - 1
    depmax = 1.250000e-01      depmin = -1.134769e-01
FILE: two.sac - 2
    depmax = 2.500000e-01      depmin = -2.269538e-01
FILE: four.sac - 3
    depmax = 1.000000e+00      depmin = -9.078154e-01

```

Example 2: Use of GETVAL

```

SAC> message (getval 10.0)
==> message -0.00011933 -0.000238659 -0.000954636
-0.00011933
-0.000238659
-0.000954636
SAC> message (getval file 1 10.0)
==> message -0.00011933
-0.00011933
SAC> message (getval file 3 10.0)
==> message -0.000954636
-0.000954636
SAC> setbb ab (getval file 2 10.0)
==> setbb ab -0.000238659
SAC> getbb ab
ab = -0.000238659
SAC>

```

Example 3: Simple Calculations

```

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.

Example 4: Use of GETTIME GETTIME returns the time offset (in seconds) relative to the time reference (O or B) for the first data point meeting the selection criteria.:

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

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

For example, to set the time in seconds to a blackboard variable 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 at time 41.87 seconds.

To return 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.

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 longer work. For example, to use **CONCATENATE** to set the station and event names in the title of a plot prior to 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. Through SAC version 101.6a, 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.

All reals in the SAC header and data sections are single-precision (32 bits, 4 bytes). As discussed in the section on precision in [TUTORIAL](#), although having single-precision for the data is sufficient, having only single-precision for time and distance variables in the header is no longer good enough to handle many modern data sets. SAC version 102.0 solves this problem. The header is unchanged, but a "footer" is added with 22 header variables in double-precision. Version 102.0 can handle "old" SAC data files with no footer or files with a footer, so there is complete compatibility. SAC v102.0 distinguishes between "old" and "new" files by the header variable NVHDR: if NVHDR = 6, the SAC file is in "old" format, if NVHDR = 7, it is in "new" format with a footer. See the section on *Floating-Point Precision in SAC* in [TUTORIAL](#) for a more detailed discussion.

SAC Binary Format for NVHDR = 6

The binary format is the most common one use. It is the default format in the [READ](#) and [WRITE](#) commands and in the subroutine library (RSAC1, RSAC2, WSAC1, WSAC2, WSAC0.) These are binary (unformatted) files.

Structure of SAC Binary Data File for NVHDR = 6

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	<ul style="list-style-type: none">• dependent variable• amplitude• real component	<ul style="list-style-type: none">• 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 referred 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	Auxiliary		Not really in the header. Derived from other header fields.

Header Variables

Word	Type	NAMES	o	o	o	o
0	F	DELTA	DEPMIN	DEPMAX	UNUSED	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	IBODY	UNUSED	UNUSED
100	I	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED
105	L	LEVEN	LPSPOL	LOVROK	LCALDA	UNUSED

... continued on next page

Word	Type	NAMES	o	o	o	o
110	K	KSTNM	KEVNM*			
116	K	KHOLE	KO	KA		
122	K	KT0	KT1	KT2		
128	K	KT3	KT4	KT5		
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, 16 bytes) long.

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

Structure of SAC Binary Data File for NVHD = 7

The only difference between a SAC binary data file with NVHDR = 6 and one with NVHDR = 7 is the addition of a "footer" following the data section(s). The footer is comprised of 22 double-precision (64-bit) floating point variables: DELTA, B, E, O, A, T0 ... T9, F, EVLO, EVLA, STLO, STLA, SB, and SDELTA. The names are the same as in the header; SAC distinguishes between the single- and double-precision variables in internal calculations by their location in the file.

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 80-character lines. For both NVHDR = 6 and NVHDR = 7, this is followed by one or two data sections. The data are in 5G15.7 format. The following table shows the line number, formats and names of the variables on the header section lines.

LINE	FORMAT	NAMES	o	o	o	o
01	(5G15.7)	DELTA	DEPMIN	DEPMAX	UNUSED	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	SB
12	(5G15.7)	SDELTA	DEPMEN	CMPAZ	CMPINC	XMINIMUM
13	(5G15.7)	XMAXIMUM	YMINIMUM	YMAXIMUM	ADJTM	UNUSED
14	(5G15.7)	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED
15	(5I10)	NZYEAR	NZJDAY	NZHOURL	NZMIN	NZSEC
16	(5I10)	NZMSEC	NVHDR	NORID	NEVID	NPTS

... continued on next page

LINE	FORMAT	NAMES	o	o	o	o
17	(5I10)	NSNPTS	NWFID	NXSIZE	NYSIZE	UNUSED
18	(5I10)	IFTYPE	IDEP	IZTYPE	UNUSED	IINST
19	(5I10)	ISTREG	IEVREG	IEVTYP	IQUAL	ISYNTH
20	(5I10)	IMAGTYP	IMAGSRC	IBODY	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		
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> CH NVDHR 7
SAC> WRITE ALPHA TEMP1
```

You can read this alphanumeric file directly into SAC with the following commands:

```
SAC> READ TEMP1
SAC> LH
```

SAC knows the difference between an alphanumeric and binary SAC files and should be able to read them directly. You can specify an ALPHA flag to the [READ](#) command for backwards compatibility:

```
SAC> READ ALPHA TEMP1
SAC> LH
```

Listed next are the first lines of TEMP1 -- header plus first 8 lines of data and ending with the 22 lines of the footer (one line per variable):

```
0.01000000    -1.569280     1.520640    -12345.00    -12345.00
 9.459999     19.450000    -41.430000     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     48.000000    -120.00000  -12345.00    -12345.00
 48.000000    -125.00000  -12345.00     15.000000    -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	7	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
....				
0.0099999997764825821				
9.4599990844726562				
19.449998861178756				
-41.430000305175781				
10.46399974822998				
-12345				
-12345				
-12345				
-12345				
-12345				
-12345				
-12345				
-12345				
-12345				
-12345				
-125				
48				
-120				
48				
-12345				
-1234				

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 7. Older version data (NVHDR < 6) are automatically updated when read into sac. In SAC v102.0, one can still use NVHDR = 6. To have double-precision capability, NVHDR = 7 [required]
B	F	Beginning value of the independent variable. [required]
E	F	Ending value of the independent variable. [required]
IFTYPE	I	<p>Type of file [required]:</p> <ul style="list-style-type: none"> • 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.
IDEP	I	<p>Type of dependent variable:</p> <ul style="list-style-type: none"> • IUNKN (Unknown) • IDISP (Displacement in nm) • IVEL (Velocity in nm/sec) • IACC (Acceleration in nm/sec/sec) • NOTE Others use SI units for IDISP, IVEL, IACC. • IVOLTS (Velocity in volts)
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 Day of the Year.
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.

... continued on next page

Name	Type	Description
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	<p>Reference time equivalence:</p> <ul style="list-style-type: none"> • 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.
NSNPTS	I	Number of points of original time series file (Stored NPTS).
SB	F	Begin value of original time series file (Stored B).
SDELTA	F	Time increment in original time series file (Stored DELTA).

Phase Picks

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

... continued on next page

Name	Type	Description
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 upward vertical; SEED/MINISEED uses dip: degrees from horizontal down).
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 (kilometers -- previously meters)
MAG	F	Event magnitude.
IMAGTYP	I	<p>Magnitude type:</p> <ul style="list-style-type: none"> • IMB (Bodywave Magnitude) • IMS (Surfacewave Magnitude) • IML (Local Magnitude) • IMW (Moment Magnitude) • IMD (Duration Magnitude) • IMX (User Defined Magnitude)

... continued on next page

Name	Type	Description
IMAGSRC	I	<p>Source of magnitude information:</p> <ul style="list-style-type: none"> • 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	<p>Type of event:</p> <ul style="list-style-type: none"> • 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 (LOCID).
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).

... continued on next page

Name	Type	Description
IBODY	I	<p>Body / Spheroid definition used in Distance Calculations</p> <ul style="list-style-type: none"> • undef / -12345 Sac Historical Spheroid version <ul style="list-style-type: none"> – a: 6378160.0 m – f: 0.00335293 – Important Note: For consistency with previous versions, it is best to leave the IBODY as undefined. This is not WGS-84, see IEARTH • ISUN / 98 Sun Meftah2015 Rozelot2001 <ul style="list-style-type: none"> – a: 696000000.0 m – f: 8.189e-6 • IMERCURY / 99 Mercury IAU2000 IAU-WGCCRE <ul style="list-style-type: none"> – a: 2439700.0 m – f: 0.0 • IVENUS / 100 Venus IAU2000 IAU-WGCCRE <ul style="list-style-type: none"> – a: 6051800.0 m – f: 0.0 • IEARTH / 101 WGS-84, EPSG:4325 <ul style="list-style-type: none"> – a: 6378137.0 m – f: 1.0 / 298.257223563 (= 0.0033528106647474805) • IMOON / 102 Moon IAU2000 IAU-WGCCRE <ul style="list-style-type: none"> – a: 1737400.0 m – f: 0.0 • IMARS / 103 Mars IAU2000 IAU-WGCCRE <ul style="list-style-type: none"> – a: 3396190.0 m – f: 1.0 / 169.89444722361179 (= 0.005886007555525457)

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	<p>Quality of data [not currently used]:</p> <ul style="list-style-type: none"> • IGOOD (Good data) • IGLCH (Glitches) • IDROP (Dropouts) • ILOWSN (Low signal to noise ratio) • IOTHER (Other)

... continued on next page

Name	Type	Description
ISYNTH	I	<p>Synthetic data flag [not currently used]:</p> <ul style="list-style-type: none"> • 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
iday	10
io	11
ia	12
it0	13
it1	14
it2	15
it3	16
it4	17
it5	18
it6	19
it7	20
it8	21

... continued on next page

Name	ID
it9	22
iradnv	23
itannv	24
iradev	25
itanev	26
inorth	27
ieast	28
ihorza	29
idown	30
iup	31
illbb	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
irltda	49
ivolts	50
imb	52
ims	53
iml	54
imw	55
imd	56
imx	57
ineic	58
ipdeq	59
ipdew	60
ipde	61
iisc	62
ireb	63
iusgs	64
ibrk	65

... continued on next page

Name	ID
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
iox	81
inu	82
inc	83
io_	84
il	85
ir	86
it	87
iu	88
ieq3	89
ieq0	90
iox0	91
iqc	92
iqb0	93
igey	94
ilit	95
imet	96
iodor	97
isun	98
imercury	99
ivenus	100
iearth	101
imoon	102
imars	103

REFERENCES

[IAU2000] <https://link.springer.com/article/10.1023/A:1013939327465>

Seidelmann, P., V. Abalakin, M. Bursa, M. Davies, C. De Bergh, J. Lieske, J. Oberst, J. Simon, E. Standish, P. Stooke, et al. (2002), Report of the IAU/IAG working group

on cartographic coordinates and rotational elements of the planets and satellites: 2000, *Celestial Mechanics and Dynamical Astronomy*, 82 (1), 83–111.

[IAU-WGCCRE] <https://astrogeology.usgs.gov/groups/IAU-WGCCRE>

[Meftah2015] <https://link.springer.com/article/10.1007%2Fs11207-015-0655-6>

Meftah, M., A. Irbah, A. Hauchecorne, T. Corbard, S. Turck-Chi'eze, J.-F. Hochedez, P. Boumier, A. Chevalier, S. Dewitte, S. Mekaoui, et al. (2015), On the determination and constancy of the solar oblateness, *Solar Physics*, 290(3), 673–687.

- 8.189e-6 (Flattening)

[Rozelot2001] <https://link.springer.com/article/10.1023/A:1005238718479>

Rozelot, J. P., Godier, S., & Lefebvre, S. (2001). On the theory of the oblateness of the Sun. *Solar Physics*, 198(2), 223-240.

- 8.33e-6 (Flattening)
- 6.50e-6 - 10.20e-6 (Flattening Range)

LATEST REVISION

Version 102.0

SAC Reading and Writing Routines

Overview

Using the SAC I/O library, `/${SACHOME}/lib/libscio.a`, one can write stand-alone codes in C or FORTRAN to read and write SAC formatted data files. The SAC I/O library is included in the sub-directory `/${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 in the online version and at `/$SACAUX/doc/examples`

Two distinct interfaces for reading and writing sac type files exist:

- Fortran-77 style SAC I/O interface (RSAC1, WSAC2, GETFHV, ...)
- C style SAC I/O interface (see end of this file)

For both styles, SAC uses C programming internally. Reading and writing files with the Fortran-77 stye interface is consistent with previous versions of the libscio library. The C style interface is documented at the end of this file.

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

```
gcc -o program source.c `sac-config --cflags --libs scio`  
f77 -o program source.f `sac-config --cflags --libs scio`
```

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 (REAL*4) header variables
- GETDHV gets Double (REAL*8) NVHDR=7 (v7) footer variables (new in SACv102.0)
- 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 (INTEGER) header variables.

For definitions of the SAC header variables, see [SAC data file format](#).

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

- SETFHV sets REAL*4 header variables
- SETDHV sets REAL*8 v7 footer 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 (v6 only)
- WSAC2 writes unevenly spaced and spectral files (v6 only)
- WSAC0 writes either format but has more comprehensive header files than the other two - including the ability to write a v7 file.

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, the call must be preceded by 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. 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.

NVHDR=6 is the default for WSAC0.

If one is reading this file from within SAC, the code for the programs is not visible, just the link to the file in `/${SACHOME}/doc/examples/`. The SAC manual, in either HTML or PDF, includes the code. If you do not have the manual on your computer, you can get the more complete version by going to URL http://ds.iris.edu/files/sac-manual/manual/input_output.html.

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
!       - Beginning 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
   - Beginning 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 &variable 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.

NVHDR=7: Use of `rsac1()` to read v7 files works, but values of calling arguments `b` and `delta` are the REAL*4 header variables. To obtain the full 64 bit v7 footer values, follow the `rsac1()` call with:

```

call getdhv("begin", begin, nerr)
call getdhv("delta", delta, nerr)

```

If there are other REAL*8 variables of interest in the v7 footer such as `t0`, one can get them by commands such as `call getdhv("t0",t0,nerr)`, where `t0` has been declared to be REAL*8.

Reading an Unevenly-Sampled or Spectral SAC File

In routine `rsac2()` is for reading in either:

- a spectral file, (Real + Imaginary or Amplitude + Phase)
- an unevenly spaced time series file (Amplitude + Time)

To determine the type of file, you must check the IFTYPE and LEVEN header variables:

- Evenly spaced file: IFTYPE = ITIME (1) and LEVEN = True
- Unevenly spaced file: IFTPYE = ITIME (1) and LEVEN = False
- Spectral file: (Real + Imaginary) IFTYPE = IRLIM (2)
- Spectrla file: (Amplitude + Phase) IFTYPE = IAMPH (3)

Both components are 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 &variable 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 %d\n", kname, nerr);
        exit(nerr) ;
    }

    /* Do some processing ... */

    exit(0);
}
```

NVHDR=7: If one uses `rsac2()` to read a v7 file, timing values stored in `xarray` are still REAL*4.

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
character*9 name
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)

!   .....
    name = ' '
    call getkhv('kstnm', name, nerr);
    if(nerr .ne. 0) then
        write(*,*)'Error reading variable: kstnm'
        call exit(-1)
    endif

    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 */
rsacl( 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);
}

```

NVHDR=7: If one wants the 64-bit version of one of the 22 floating-point header variables that are in the v7 footer, one should use GETDHV. If NVHDR=6, GETDHV will return 32-bit values promoted to 64 bit but does not increase their precision.

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 newhdr()
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, nerr
          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
*/
newhdr();
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 %d\n", kname, nerr);
    exit(-1);
}

exit(0);
}

```

NVHDR=7 - Writing a Version 7 file is **only** currently possible using WSAC0. Use of WSAC1 or WSAC2 create a v6 header with NEWHDR and are unable to write a v7-type file.

If you are writing a v7 file then, you will need to do either:

```

! To Generate a new file
integer :: nerr
real*8  :: t0
real*4  :: y(101), x(1)

y(:) = 0.0
y(50) = 1.0

call newhdr()
call setnhv("nvhdr", 7, nerr)
call setnhv("npts", 101, nerr)

t0 = 10.12
call setdhv("t0", t0, nerr)

call wsac0("filename.sac", x, y, nerr)
if(nerr .ne. 0) then
    write(*,*)'Error writing sac file',nerr
endif

```

or to read, update, and write the file over:

```

! Read, update and write a file
integer :: nerr, max, nlen
real*4  :: b, dt
real*8  :: t0
real*4  :: y(1000), x(1)
max = 1000

```

```

call rsac1("filename.sac", y, nlen, b, dt, max, nerr)

! Convert v6 to v7 if necessary
call setnhv("nvhdr", 7, nerr)

t0 = 15.12
call setdhv("t0", t0, nerr )

call wsac0("filename.sac", x, y, nerr)
if(nerr .ne. 0) then
    write(*,*)'Error writing sac file',nerr
endif

```

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 variables used in the call to wsac2()
character*11 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 newhdr()

```

```

        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 variables 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
*/
newhdr();
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 variables used in the call to wsac2()
character*11 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 Amplitude/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 variables 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 Amplitude/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 are 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 kevm, kstnm
real b, delta

```

```

real cmpaz, cmpinc
integer npts
integer nerr, j, i

DATA KNAME/' STAZ', ' STBZ', ' STCZ', ' STDZ', ' STEZ',
1      ' 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 <stdint.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 kevm[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(kevm, "Event Name");

setnhv("NPTS", &ndata, &nerr, strlen("NPTS"));
setfhv("EVLA", &evla, &nerr, strlen("EVLA"));
setfhv("EVLO", &evlo, &nerr, strlen("EVLO"));
setkhv("KEVM", &kevm[0], &nerr, strlen("KEVM"), 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;
}

```

C-Style Interface

The C-style version of the library is fully documented at: <https://savage13.github.io/sacio/html/index.html>. More functions than shown below are available. A simple example is provided here for reference:

```

#include <stdio.h>
#include <sacio.h>

int nerr    = 0;
int ival    = 0;
double dt   = 0.0;

```

```

char sta[18] = {0};

// Read a file named "file.sac"
sac *s = sac_read("filename.sac", &nerr);

// Set and get the delta
sac_set_float(s, SAC_DELTA, 0.25);
sac_get_float(s, SAC_DELTA, &dt);
printf("delta: %f\n", dt);

// Set the file type to: evenly spaced, time series
sac_set_int(s, SAC_EVEN, 1);
sac_set_int(s, SAC_FILE_TYPE, ITIME);
sac_get_int(s, SAC_EVEN, &ival);
printf("leven: %d\n", ival);
sac_get_int(s, SAC_FILE_TYPE, &ival);
printf("iftype: %d\n", ival);

// Set a string, SAC_KSTNM and SAC_STA are the same
sac_set_string(s, SAC_KSTNM, "PAS");
sac_get_string(s, SAC_STA, sta, sizeof sta);
printf("sta:  '%s'\n", sta);

// Most input strings are truncated at 8 characters
//     SAC_KNETWK and SAC_NET are the same
sac_set_string(s, SAC_KNETWK, "12345678");
sac_get_string(s, SAC_NET, sta, sizeof sta);
printf("net:  '%s'\n", sta);

// ... except for the Event name, it gets 16 characters
//     SAC_KEVNM and SAC_EVENT are the same
sac_set_string(s, SAC_EVENT, "1234567890123456");
sac_get_string(s, SAC_KEVNM, sta, sizeof sta);
printf("event: '%s'\n", sta);

// Set the file version to either v6 or v7
sac_set_v6(s);
sac_set_v7(s);
sac_get_int(s, SAC_HDR, &ival);
printf("hdr:   %d\n", ival);

// Amplitude (and time data) are accessed through
//     s->y and s->x
// Set the first five values
for(int i = 0; i < 5; i++) {
    s->y[i] = (float) i;
}
// Print out the first five data points
for(int i = 0; i < 5; i++) {
    printf("y[%d]: %e\n", i, s->y[i]);
}

// Write a sac file
sac_write(s, "filename.sac", &nerr);

```

```
if(nerr != 0) {  
    printf("Error writing sac file: %d\n", nerr);  
}  
  
// Free the sac file  
sac_free(s);  
s = NULL;
```

Using the SAC Libraries

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. The internal routines here are wrapped in an interface that should be more streamlined to use than previous versions to v102.0. The libraries `libsac.a` and `libsacio.a` are in `${SACHOME}/lib`. For more detailed examples, see `${SACHOME}/doc/examples` contained in the SAC distribution.

Callable in C and Fortran

All of these available functions are simplified wrappers around internally used functions within SAC with obscure, shortened and forgotten names and extra, usually unneeded, or confusing parameters. Each function documented below should be callable directly from C and Fortran. The Fortran wrappers should work simply for Fortran compilers that append underscores to function names internally within the program.

A difference between the C and Fortran versions is the calling convention of pass-by-value (default in C) and pass-by-reference (Fortran).

Compiling

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 for SAC. If you have the a C compiler or a Fortran compilers, try:

```
cc -o program main.c subs.c `sac-config --cflags --libs libsac libsacio`
gfortran -o program main.f `sac-config --cflags --libs libsac libsacio`
```

Fourier Transform (FFT)

Given below are both single- and double-precision routines for doing forward and inverse Fourier transforms. All transforms are performed in double precision, as all subroutine calls within SAC use the same internal code path. Single-precision versions internally convert/copy the input arrays to double precision as a prelude to performing the transform, and the results are then converted back to single precision on return. The internal calculations are done using a power-of-2 number of points. For a forward transform, n need not be a power of 2, but the output nf must be the next power of 2 greater than or equal to n . Parameter nf must be defined prior to calling any of these routines.

```
// Forward Transform - Single Precision
void  fft (float data, int n, float *re, float *im, int nf)
void  fftz(float data, int n, float complex *z, int nf)

// Forward Transform - Double Precision
void  dfft(double data, int n, double *re, double *im, int nf)
void  dfftz(double data, int n, double complex *z, int nf)

// Inverse Transform - Single Precision
void  ifft(float data, int n, float *re, float *im, int nf)
void  ifftz(float data, int n, float complex *z, int nf)

// Inverse Transform - Double Precision
```

```

void idfft(double data, int n, double *re, double *im, int nf)
void idfftz(double data, int n, double complex *z, int nf)

```

Compute the Fourier Transform (or inverse transform) of a data series.

Arguments

- **data** - Input time series for forward transform; output time series for inverse
- **n** - Length of time series on input for forward transform; number of points desired from the inverse transform
- **z** - Complex FFT Spectrum
- **re** - Real Component of the Fourier spectrum; calculated in forward transform, input for inverse transform.
- **im** - Imaginary Component of the Fourier spectrum.
- **nf** - Input length of **re**, **im**, and **z** for inverse transform; calculated in forward transform.

Normalization

Normalization of the transform by the length, **nf**, is done on the inverse transforms

Time Scaling

Time scaling is not performed within these functions, but can be accomplished by multiplying the Fourier spectrum by the sampling rate, **dt**. If the scaling is applied to the spectrum, make sure to remove the time shift to get back to the original time series.

Frequency Ordering

Amplitudes are ordered frequency starting with the zero frequency, through positive frequencies to the Nyquist $df \cdot (nf/2)$, then backwards through the negative frequencies

```

0, df, ... , df*(nf/2-1), df*(nf/2), -df*(nf/2-1), ... , -df

```

Examples

```

integer n, nf
real*8 :: data(16), data2(16)
real*8 :: re(64), im(64)
complex*16 :: z(16)

n = 10

! Find next power of 2
nf = 4
do while (nf < n)
  nf = nf * 2
enddo

! FFT with real/imaginary
call dfft(data, n, re, im, nf)
call idfft(data2, n, re, im, nf)

! FFT with complex number
call dfftz(data, n, z, nf)
call idfftz(data2, n, z, nf)

```

Remove Mean

```
void remove_mean (float *data, int n)
```

Remove the mean of a data series. The mean of the data series is automatically calculated and removed from the data series.

Arguments

- **data** - Input data series
- **n** - length of data

Note: Data is modified in place.

Examples

```
implicit none

integer,parameter :: nmax = 1776
integer :: npts, nerr
real*4 :: data(nmax), beg, dt

! Read in the data file
call rsac1('raw.sac', data, npts, beg, dt, nmax, nerr)

! Remove the mean of the data in place
call remove_mean(data, npts)
```

Effective SAC Commands

```
SAC> read raw.sac
SAC> rmean
```

Remove Trend

```
void remove_trend(float *data, int n, float delta, float b)
```

Remove the trend (along with the mean) of a data series in memory

Arguments

- **data** - Input data series, overwritten on output
- **n** - length of data
- **delta** - Time sampling of the data
- **b** - Initial time value of the data series

Note: Data is modified in place.

This calls internal routines `lifite()` and `rtrend()`.

Trend is removed as

```
y[i] = y[i] - yint - slope * (b + delta * i)
```

where y is the data

Examples

```
#define NMAX 1969

float y[NMAX], b, dt;
int nmax = NMAX;
int n, nerr;

// Read in the data file
rsacl("raw.sac", y, &n, &b, &dt, &nmax, &nerr, -1);

// Remove the trend of the data in place
remove_trend(y, n, dt, b);
```

Effective SAC Commands

```
SAC> read raw.sac
SAC> rtrend verbose
```

Filtering

Data is filtered using an Infinite Impulse Response Filter. See the [BANDPASS](#) command for definitions of the filter parameters and descriptions on how to use them.

```
void bandpass(float *data, int n, float dt, float low, float high)
void lowpass(float *data, int n, float dt, float corner)
void highpass(float *data, int n, float dt, float corner)

void filter(int prototype,
            int type,
            float *data, int n, float dt,
            float low, float high, int passes, int order,
            float transition,
            float attenuation)
```

Arguments

- *data* - Input and output data
- *n* - Length of data
- *dt* - Time sampling of the data (seconds)
- *low* - low frequency corner
- *high* - high frequency corner
- *corner* - corner of the filter for *lowpass* or *highpass*
- **passes** - **Number of passes**
 - 1 - forward pass only (causal)
 - 2 - forward and backward pass (zero-phase)

- *order* - Filter Order, not to exceed 10, 4-5 should be sufficient
- *transition* - Transition Bandwidth, only used in Chebyshev Type I and II Filters
- *attenuation* - Attenuation factor, amplitude reached at stopband edge, only used in Chebyshev Type I and II Filters
- **prototype - Filter Prototype**
 - 0 - Butterworth filter
 - 1 - Bessel filter
 - 2 - Chebyshev Type I filter
 - 3 - Chebyshev Type II filter
- **type - Filter Type**
 - 0 - Bandpass
 - 1 - Highpass
 - 2 - Lowpass
 - 3 - Bandreject

Examples

Bandpass filter in C

```
#define NMAX 2015
float y[NMAX], b, dt;
int n, nerr, nmax = NMAX;

// Read in the data file
rsac1("raw.sac", y, &n, &b, &dt, &nmax, &nerr, -1);

// bandpass filter from 0.10 Hz to 1.00 Hz
bandpass(y, n, dt, 0.10, 1.00);
```

Highpass filter in Fortran

```
implicit none
integer nmax, n, nerr, sac_compare
real*4 :: y(2012), b, dt
nmax = 2012

! Read in the data file
call rsac1("raw.sac", y, n, b, dt, nmax, nerr)

! highpass filter at 10.0 Hz
call highpass(y, n, dt, 10.0)

**Effective SAC Commands**

SAC> read raw.sac
SAC> bp co 0.10 1.0 p 2 n 4

SAC> read raw.sac
SAC> hp co 10.0 p 2 n 4
```

Further examples are given in `/${SACHOME}/doc/examples/filter/`. Because one uses FFT that pads with zeros, it is often prudent to precede the filter with `rtrend`; `taper`.

Cross Correlation

```
void correlate(float *f, int nf, float *g, int ng, float *c, int nc)
```

Compute the cross-correlation of two signals

Arguments

- *f* - First time series
- *nf* - Length of first time series
- *g* - Second time series
- *ng* - Length of second time series
- *c* - Cross correlation time series
- *nc* - Size of *c*, must be at least ($nf + ng - 1$)

Return: Cross correlation function, length: $nf + ng - 1$

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`

Examples

Effective SAC Commands

```
SAC> read file1.sac file2.sac
SAC> correlate
```

Cross Correlation Extras

```
int correlate_max(float *c, int nc)
```

Find the maximum of a correlation

Arguments

- *c* - float array (returned from `correlate` function)
- *nc* - length of *c*

Return: Index of maximum value in array

```
float correlate_time(float dt, float b, int i)
```

Compute the time of a data point given *dt* and begin time

Arguments

- *dt* - Time sampling
- *b* - Begin time

- *i* - data sample

Return: time value ($b + i * dt$)

```
float * correlate_time_array(float dt, float b, int n)
```

Compute a time array given dt and begin time

Arguments

- *dt* - Time sampling
- *b* - Begin time
- *n* - Length of data array

Return: time array

```
float correlate_time_begin(float dt, float n1, float _n2, float b1, float b2)
```

Compute begin time from a corealtion of two time series

Arguments

- *dt* - Time sampling
- *n1* - Length of first time series
- *n2* - Length of second time series (unused)
- *b1* - Begin time of first time series
- *b2* - Begin time of second time series

Return: $-dt * (n1 - 1) + (b2 - b1)$

This accounts for the possible differences in begin times of two time series

Envelope Calculation

```
void envelope(int n, float *in, float *out)
```

Compute the envelope of a time series using the Hilbert transform

Arguments

- *n* - Length of input and output time series
- *in* - Input time series
- *out* - Output time series with envelope applied

The envelope is applied as such where the $H(x)$ is the Hilbert transform:

```
out = sqrt( H( in(t) )^2 + in(t)^2 )
```

Examples

```
#define NMAX 1929
int nlen, nerr, nmax;
float yarray[NMAX], yenv[NMAX];
float beg, delta;

nmax = NMAX;

// Read in data file
rsac1("raw.sac", yarray, &nlen, &beg, &delta, &nmax, &nerr, SAC_STRING_LENGTH);

// Calculate Envelope of data
envelope(nlen, yarray, yenv);
```

Effective SAC Commands

```
SAC> read raw.sac
SAC> envelope
```

Because one uses FFT that pads with zeros, it is often prudent to precede the filter with [rtrend](#) ; [taper](#).

Differentiate

```
void dif2(float *array, int n, double delta, float *output)
```

Differentiate a data set using a two point differentiation

Arguments

- *array* - Input data to differentiate
- *n* - length of array
- *delta* - Time sampling of input data
- *output* - Output differentiated data, length n-1

This is the default scheme in the SAC program.

The output array will be 1 data point less than the input array.

Since this is not a centered differentiation, there is an implied shift in the independent variable by half the delta:

```
b_new = b_old + 0.5 * delta
```

Differentiation is performed as:

```
out[i] = (1/delta) * (in[i+1] - in[i])
```

Examples

```
integer,parameter :: nmax = 1000000
integer :: npts, nerr
real*4 :: data(nmax), out(nmax)
real*4 :: beg, dt

! Read in the data file
call rsac1("raw.sac", data, npts, beg, dt, nmax, nerr)

! Differentiate the data
call dif2(data, npts, dble(dt), out)

bnew = beg + 0.5 * delta
npts_new = npts - 1
```

Effective SAC Commands

```
SAC> read raw.sac
SAC> dif
```

Integerate

```
void int_trap(float *y, int n, double delta)
```

Integrate a data series using the trapezoidal method

Arguments

- *y* - Input data series, overwritten on output
- *n* - length of *y*
- *delta* - time sampling of the data series

Integration is performed as:

```
out[i] = out[i-1] + (delta/2) * (in[i] + in[i+1])
```

where the initial out value is 0.0.

The number of points on output should be reduced by 1

```
len(out) = len(in) - 1
```

and the beging value is shifted by 0.5 delta:

```
b_out = b_in + 0.5 * delta
```

Examples

```

#define NMAX 2012
float y[NMAX], b, dt;
int n, nerr, nmax = NMAX;

rsacl("raw.sac", y, &n, &b, &dt, &nmax, &nerr, -1);

int_trap(y, n, (double)dt);

```

Effective SAC Commands

```

SAC> read raw.sac
SAC> int

```

Taper Data

```

// Taper using points
void taper_points(float *data, int n, int taper_type, int ipts)
void taper(float *data, int n, int taper_type, int ipts)

// Taper using a duration in seconds
void taper_seconds(float *data, int n, int taper_type, float sec, float
delta)

// Taper using a percent of the data
void taper_width(float *data, int n, int taper_type, float width)

```

Arguments

- *data* - Input data series, overwritten on output
- *n* - Length of data
- *taper_type* - Type of Taper
 - 1 - Cosine - SAC_TAPER_COSINE
 - 2 - Hanning - SAC_TAPER_HANNING [Default in SAC]
 - 3 - Hamming - SAC_TAPER_HAMMING
- *ipts* - Points to use in the taper
- *sec* - Duration of the taper in seconds
- *delta* - Delta of the data
- *width* - Percent of the data to taper [SAC default is 5%]

Examples

```

#define MAX 1984
float data[MAX];
int nmax, npts, nerr, taper_type;
float beg, dt, width;

nmax = MAX;

```

```

// Read in the data file
rsacl("raw.sac", data, &npts, &beg, &dt, &nmax, &nerr, -1);

// Set up taper parameters
width = 0.05;    // Width to taper original data
taper_type = 2;  // HANNING taper

taper_width(data, npts, taper_type, width);

```

Effective SAC Commands

```

SAC> read raw.sac
SAC> taper TYPE HANNING WIDTH 0.05
(these are the defaults for taper in SAC)

```

Cut Data

```

void cut(float *y, int npts, float b, float dt,
         float begin_cut, float end_cut, int cuterr,
         float *out, int *nout)

```

Cut a time series at specified begin and end times

Arguments

- *y* - Input data to be cut
- *npts* - Length of *y*
- *b* - Begin time of data
- *dt* - time sampling (seconds)
- *begin_cut* - Start time of cut
- *end_cut* - End time of cut
- *cuterr* -
 - 1 - Fatal - SAC_CUT_FATAL
 - 2 - Use B and E Values - SAC_CUT_USEBE
 - 3 - Fill with Zeros - SAC_CUT_FILLZ
- *out* - Cut data on output
- *nout* - Length of out

Examples

```

integer,parameter :: nmax = 1776
real*4 :: y(nmax), out(nmax), b, dt, cutb, cute
integer :: nerr, n, nout

max = nmax

```

```
! Read in data
call rsac1("raw.sac", y, n, b, dt, max, nerr)

nout = max
cutb = 10.0
cute = 15.0
! Cut data from 10 to 15 or from B to E if window is too big
call cut(y, n, b, dt, cutb, cute, CUT_USEBE, out, nout)
```

Effective SAC Commands

```
SAC> read raw.sac
SAC> cut 10 15
SAC> read raw.sac
```

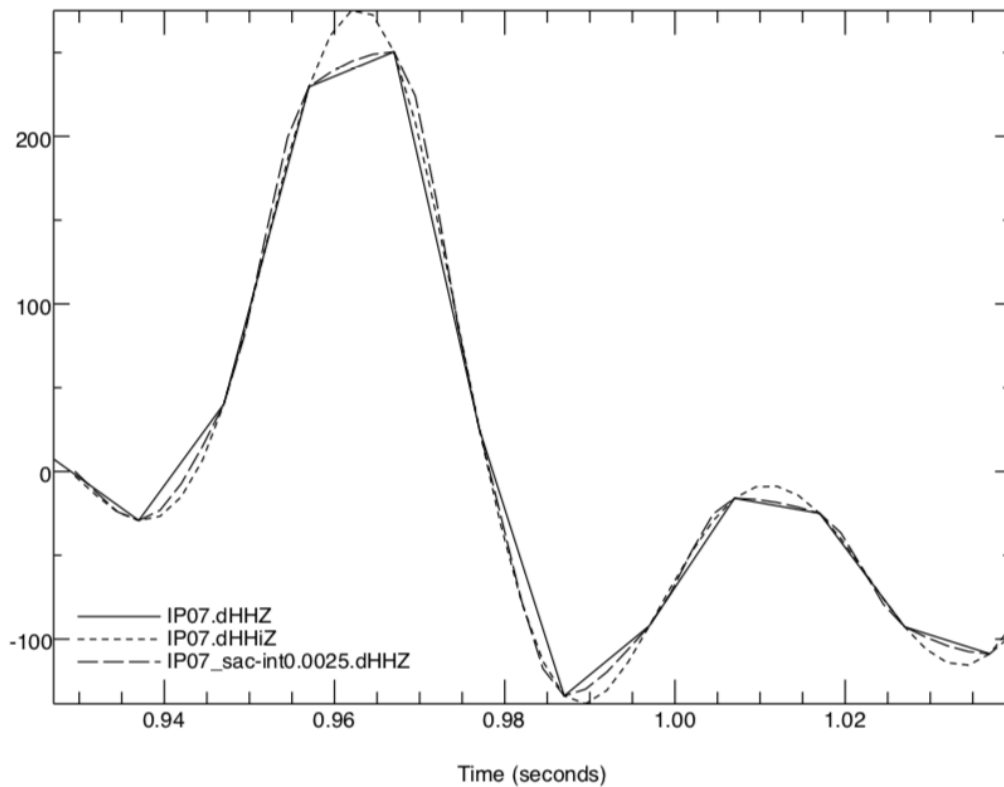
See `${SACHOME}/doc/examples/create_compare/` for an example.

Interpolation using cubic splines

In the pre-digital-data era, data extremes were relatively easy to see because the pen or light-beam velocity went to zero at them. The interpolation scheme used by the SAC interpolate routine uses a method popularized by Wiggins that took advantage of that feature so that cycle extrema could be at digitized points.

With digital data, the extrema may not be at digitized points, and for some studies it is desirable to get a better estimate of the maxima and minima (for example, estimating magnitudes based on amplitudes or when using amplitude ratios in focal-mechanism determinations. The routine below uses a pure cubic-spline interpolation published by Forsythe, et al., that can give significantly different results from a Wiggins interpolation. The program is in `${SACHOME}/doc/examples/interpolate`. The script `run_interpolate.sh` shows how the interpolation program is built and run, along with how interpolation would be accomplished in SAC.

The plot below shows the initial waveform, the result using Forsythe interpolation, and the result using SAC/Wiggins interpolation. (File `${SACHOME}/doc/examples/interpolate/interpolate.m` includes SAC calls to create the plot.)



Shown are the initial waveform (solid line), the result using Forsythe interpolation (short dashes), and the result using SAC interpolation (long dashes).

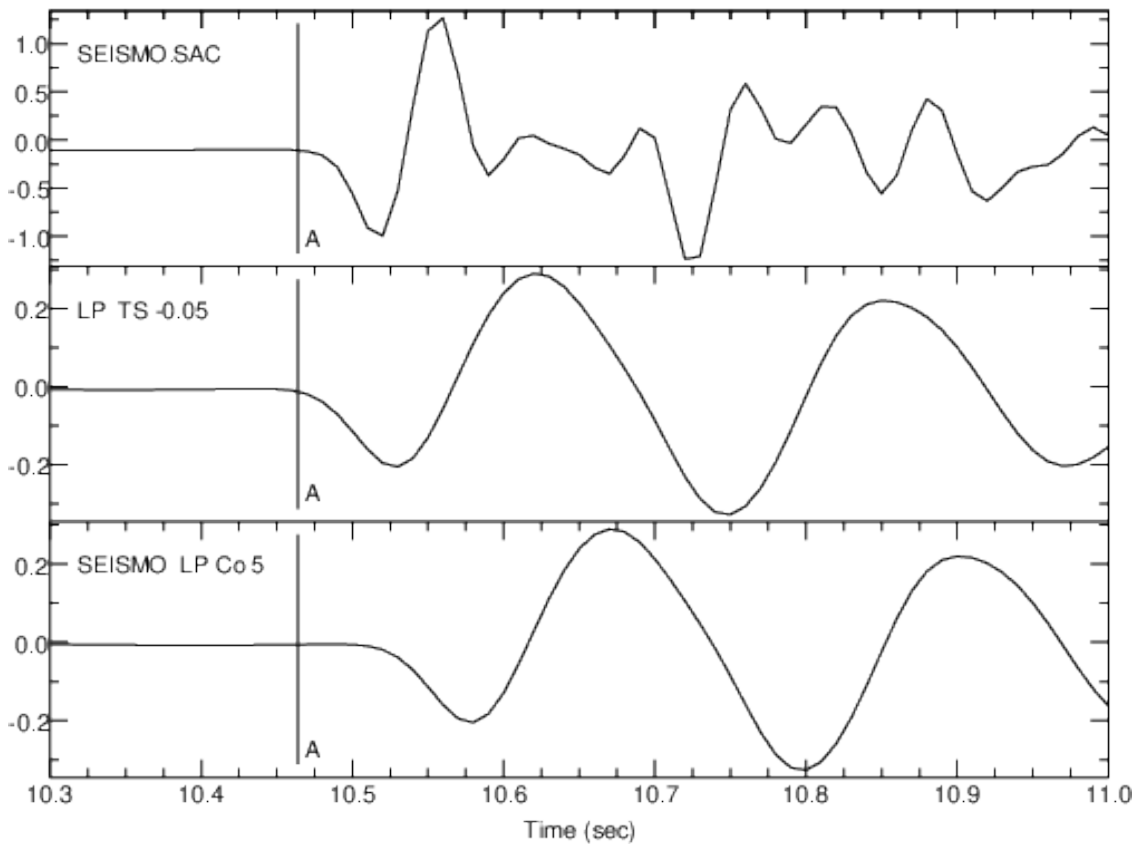
Time-Shift

There is no function in SAC that time-shifts a waveform, but as mentioned in the help file [lowpass](#) for the low-pass filter command, such filters time-shift the data, and one may want to correct for that time shift. One can use SAC to time-shift a waveform by changing the "b" header value for the SAC file. A (new as of v102.0) macro: `/${SACHOME}/macros/sac-ts.m` is an example of how to do this.

A Fortran program named `time_shift.f` in `/${SACHOME}/doc/examples/time_shift` does a time shift by taking the Fourier transform of the input time series then and doing the time shift in the frequency domain. Before taking the Fourier transform the waveform is prepared by taking out the mean/trend and then tapered to stabilize the Fourier transform. It is padded with zeros to minimize wrap-around.

All steps for an example are included in which a waveform is first low-pass filtered, which results in a time shift, and that time shift is taken out by the two methods: a call to the SAC macro and a run of program `time_shift`. The plot below shows the original plus the two methods for time-shifting the waveform for this case.

Time-Shift Summary



top: Initial waveform;
middle: `lp co 5 np 4` then time shift `-0.05`;
bottom: `seismo lp co 5 np 4`

Convolution

Prior to SAC v102.0, the SAC `CONVOLVE` command was effectively the same as the SAC `CORRELATE` command except for a sign change. For both the `CORRELATE` command and the previous version of `CONVOLVE`, the calculation is done in the frequency domain. The explicit method used for doing the convolution is called a "discrete" convolution. For many application that method is appropriate, but a *discrete* convolution has two features that potentially are undesirable when applied to time series:

- no scaling of the output by the digitizing interval, and
- no check on the start time for the pulse.

The more serious problem is the second one: If the "pulse" is centered at time zero, the old SAC `CONVOLVE` gave an incorrect waveform.

The directory `/${SACHOME}/doc/examples/convolve` has both FORTRAN and C programs with options for both *discrete* convolution and "time-series*" convolution, which treats convolution for a time series "correctly".

Sample Runs

Input for the convolution can be generated as:

```

SAC> fg triangle npts 8 delta 0.02 begin -0.08
SAC> write triangle_n8_d0.02.sac
SAC> fg impulse npts 12 delta 0.02 begin 0
SAC> write impulse_n12_d0.02.sac

```

Then run the convolvef program:

```

% ./convolvef
Usage: convolvef p_name wf_name c_name disc_conv
      where the first three arguments are filenames
      for pulse, waveform, and convolution output.
      If disc_conv is y, it uses a discrete convolution
      and the pulse begin time is set to zero.
      If disc_conv is n, pulse begin time is unchanged
      and the output is multiplied by delta, which is
      what one has in a time-series convolution.
% ./convolvef triangle_n8_d0.02.sac impulse_n12_d0.02.sac conv_y.sac y
% ./convolvef triangle_n8_d0.02.sac impulse_n12_d0.02.sac conv_n.sac n

```

The pulse file *triangle_n8_d0.02.sac* is symmetric around zero time, so comparing the last argument "y" and "n" for disc_conv demonstrates an important difference between *discrete* and *time-series* convolutions.

Convolution Primer

In these applications, f , is a waveform time series that is convolved with a pulse g . The equation for their convolution is

$$y(t) = f(t) \star g(t) = \int_{-\infty}^{\infty} dt' f(t') g(t-t')$$

Both f and g are functions of time, and their zero times are coupled through the term $g(t-t')$. From the above equation, it is easy to show that one can choose the time zero for y and f to be the same. In the applications discussed below, the zero time for $g(t)$ can be the same as or less than the zero time for $f(t)$.

To calculate the convolution, one discretizes both f and g and replaces the integral with a sum. In this discussion $\delta t = 0.02s$ is the digitizing interval for y , f , and g . One multiplies the sum by δt , which is not what is done in a *discrete* convolution, which also does not take into account any difference in zero time between $f(t)$ and $g(t)$.

Here, two applications of convolution are discussed. Both have the same $f(t)$ but different $g(t)$.

$f(t)$ is a synthetic waveform (produced using Haskell matrices or WKBJ) where vertical lines of calculated polarities and amplitudes are drawn at phase-arrival times. For these examples, the synthetic waveform is a vertical-component time series for an incident P-wave at an angle of 20° with the vertical through a 2-layer crust. There are $n_w = 2048$ points in the discretized f . (The large number is chosen to minimize wraparound.) The duration is $t_w = \delta t[n_w - 1]$. The first point in f is chosen to be at $t = 0$, so $f(t) = f(t)H(t)H(t_w - t)$, where

$$H(t) = \begin{cases} 1 & \text{if } t \geq 0 \\ 0 & \text{if } t < 0 \end{cases}$$

$g(t)$ is a pulse waveform with a far smaller duration than $f(t)$. Here, the pulse is either (1) an approximation of a P arrival so that the output of the convolution potentially models the data, or (2) a time-symmetric shape with a maximum at $t = 0$ to smooth out the Gibbs phenomenon that often accompanies arrivals in synthetics. Option (2) is often used for synthetics for receiver functions.

A "Brune" pulse for (1): $g(t) = U_0 H(t) t e^{-t/\kappa} H(t_p - t)$, where $\kappa = 0.1s$, $t_p = 1.26s$, and U_0 is a constant that only affects the amplitude, so is of no interest here.

For (2), the source pulse is a triangle function produced by fg in SAC:

fg triangle npts 8 delta 0.02 begin -0.08.

Note that the Brune pulse starts at the same time as f , while the triangle pulse starts at a negative time with a maximum at the start time for f . If the total time for the pulse is t_p , the general form for the pulse is $g(t) = g(t)H(t-t_1)H(t_2-t)$, where for the Brune pulse $t_1 = 0$, $t_2 = t_p$ and for the triangle pulse $t_1 = -0.08s$, $t_2 = 0.06s = t_p - t_1$. For either pulse, $t_p = \delta t [n_p - 1]$.

Given the above forms for f and g , the original convolution equation can be written

$$y(t) = \int_0^{t_w} dt' f(t') g(t-t') H(t-t'-t_1) H(t_2-t+t')$$

For the Brune pulse, $t_1 = 0$, so t cannot be negative, but for the triangle pulse the lower bound for t is -0.06 s. In Fortran, arrays are stored in the computer using positive integers, so to simplify the bookkeeping it is best to avoid negative times when we discretize the above equation. One can avoid negative times if one chooses $y(\tau) = y(t-t_1)$. With this choice the modified equation then reads

$$y(\tau) = y(t-t_1) = \int_0^{t_w} dt' f(t') g(\tau+t_1-t') H(\tau-t') H(t_2-\tau+t_1+t')$$

The discretized equation for the above is then

$$y_i = \delta t \sum_{j=1}^{n_w} f_j g_{i-j-j_1}$$

where $j_1 = -(t_1/\delta t) + 1$ and i runs from 1 to $n_w + n_p - 1$

The following Fortran code produces the correct result for the convolution for either pulse

```

j_1 = -nint(b_p/delta)+1
do i=1,n_w+n_p-1
  temp = 0.0
  do j=1,n_w
    if (i.ge.(j-j_1) .and. n_p.ge.(i-j+j_1)) then
      temp = temp + waveform(j)*pulse(i-j+j_1)
    endif
  end do
  conv(i) = delta*temp
end do

```

where $b_p = t_1$, $\delta t = \delta t$, and $\text{conv}(i) = y_i$.

Plots for the two pulse waveforms are shown in Figure 1, and the results of the convolution near the first synthetic arrival are shown in Figure 2 both for the Brune pulse and for the triangle pulse. The Gibbs phenomenon is quite pronounced at the arrival in the raw synthetic. Note that the peaks for the arrival in Figure 2 are at the same time for the triangle-pulse convolution and the synthetic, and the Brune-pulse convolution starts at the peak of the raw-synthetic arrival time. For display purposes, the waveforms are time-shifted so that the first arrival are at five seconds into the record. Hence the zero time for the display and the convolution calculation are not the same.

If one uses the SAC convolution for the two runs, one gets the "correct" result for the Brune pulse but a time shift of 0.08 seconds for the triangle pulse (which starts at -0.08 s). The convolution of the triangle pulse with itself is also time-shifted 0.08 s. If one uses the SAC CONVOLVE option "amplitude on" in program *convolvef* or *convolvec*, one gets the same amplitudes and times as for the discrete convolution output.

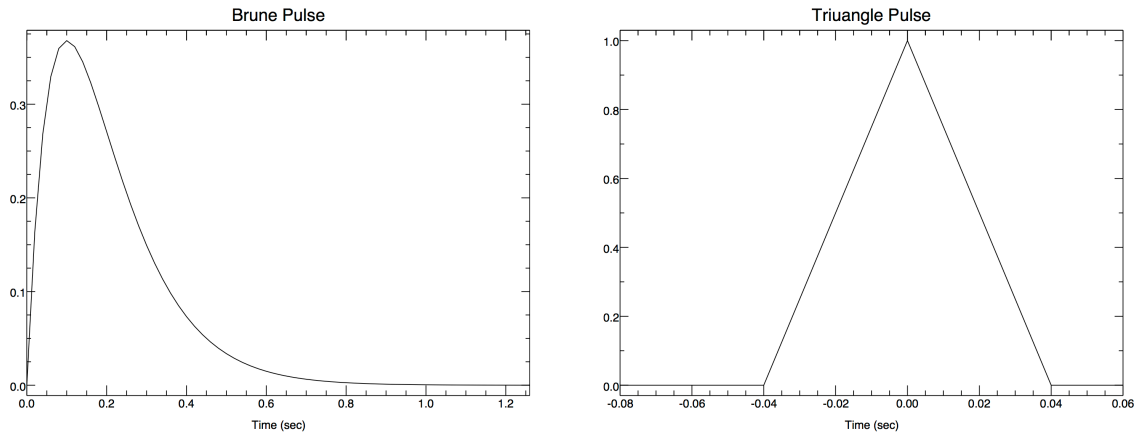


Figure 1. The two source pulses used for these convolution applications.

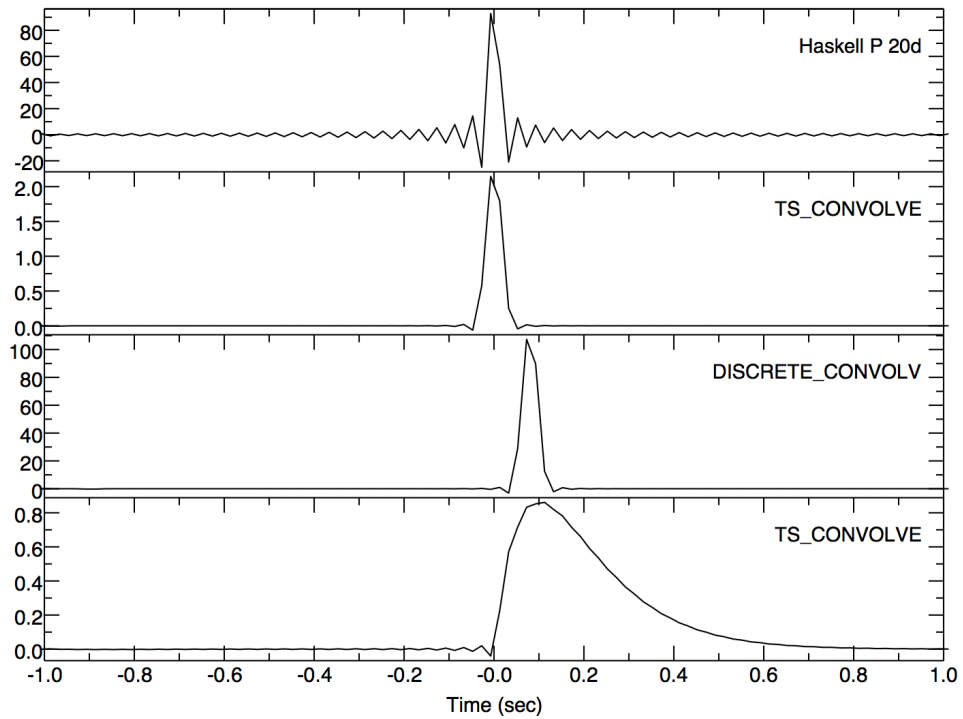


Figure 2: From top to bottom: unfiltered synthetic, time-series convolution with triangle pulse, discrete convolution with triangle pulse, time-series convolution with Brune pulse

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 preceding 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
SAC> ch kevn %TEMP%XYZ
    ch kevn %TEMP%XYZ
==> ch kevn 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 *kname, 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 *kname, 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 compiler, 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 terminator '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 scripts are provided to print the files and/or convert them to PDF format. See [sac/utls/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 similar 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.

PLOTG: 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 displayed 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
XFUDGE: 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 logarithmically 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, and `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 either endian.

The source code can be found in `sac/utlils/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 printer.

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
end
```

```
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

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.
0302	Memory manager links clobbered for block starting at:

... continued on next page

Number	Translation
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 calculation 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
1001	Bad command syntax at symbol

... continued on next page

Number	Translation
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
1113	Error starting program

... continued on next page

Number	Translation
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
1323	Undefined stop cut for file

... continued on next page

Number	Translation
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:
1381	Not a valid SORT parameter:

... continued on next page

Number	Translation
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
1618	Order = 0 in HQR

... continued on next page

Number	Translation
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
2106	Need free period and damping factor for LLL sub-type BB.

... continued on next page

Number	Translation
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.
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 =
2703	Can't evaluate logical expression:
2704	Reading macro file
2705	Searching macro file for

... continued on next page

Number	Translation
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.
5122	Distance out of range of data; blackboard variable not set:
5123	No blackboard variable name given, no variable set.

... continued on next page

Number	Translation
5124	TAUP and MODEL options are incompatible 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

Functional Command Listing

Executive Function: Information and Control

- abbrev:** Abbreviations for SAC Commands
- about:** Displays version and copyright information
- intro:** Provides a brief introduction to SAC
- help:** Prints information about SAC commands and features.
- printhelp:** Command for printing help pages from within SAC
- comcor:** Controls SAC's command correction option.
- inicm:** Reinitializes all of SAC's common blocks.
- production:** Controls the production mode option.
- report:** Informs the user about the current state of SAC.
- trace:** Controls the tracing of blackboard and header variables.
- echo:** Controls echoing of commands to the terminal.
- history:** Provides easy access to previous commands (like unix history)
- message:** Sends a message to the user's terminal.
- quit:** Terminates SAC.
- quitsub:** Terminates the currently active subprocess.
- pause:** Sends a message to the terminal and pauses.
- transcript:** Controls output to the transcript file.

Executive Function: Processing and Macros

- evaluate:** Evaluates simple arithmetic expressions.
- mathop:** Select the sequencing order for inline functions
- getbb:** Gets (prints) values of blackboard variables.
- setbb:** Sets (defines) values of blackboard variables.
- unsetbb:** Unsets (deletes) blackboard variables.
- installmacro:** Installs macro files in the global SAC macro directory.
- macro:** Executes a SAC macro file.
- setmacro:** Defines directories to search when executing a SAC macro file.
- systemcommand:** Executes system commands from SAC.

Data File Module

- data:** Search for, build requests, and download data
- event:** Search for events by region, magnitude, and date
- metadata:** Search for events by region, magnitude, and date
- response:** Download responses in sac-polezero or evalresp format
- station:** Search for station by region, identifier, channel, and date

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

write: Writes data in memory to disk.

readdb: Reads data from an Oracle database NO LONGER FUNCTIONAL

readcss: Reads CSS format data files into SAC.

writcss: Writes CSS format data files from SAC.

readgse: Reads GSE2.0 format data

writgse: Writes GSE2.0 format data

readsuds: Reads PC Suds data

readtable: Reads alphanumeric data files on disk into memory.

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

funcgen: Generates a function and stores it in memory.

convert: Converts data files from one format to another.

commit: Commits (copies) SAC's data to the I/O buffers

rollback: Replaces active data with data last committed to the I/O buffers

recalltrace: Rolls back waveforms and selected headers from I/O buffers and commits the rest of the headers.

readhdr: Reads headers from SAC data files into memory.

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

listhdr: Lists the values of selected header fields.

chnhdr: Changes the values of selected header fields.

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

readbbf: Reads a blackboard variable file into memory.

writbbf: Writes a blackboard variable file to disk.

cd: Change the working directory within SAC

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

cuterr: Controls errors due to bad cut parameters.

cutim: Cuts data after its already been read into memory

deletechannel: Deletes selected channels from memory

pickauthor: Controls author name when reading picks.

pickphase: Controls phase names to load into the SAC headers.

readerr: Controls errors that occur during the read command.

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

sort: Sorts waveforms based on selected header variables

wild: Sets wildcard characters used in read commands to expand

Graphics Environment Module

saveimg: Saves displayed graphics windows in a variety of formats.

xlim: Determines the plot limits for the x axis.

ylim: Determines the plot limits for the y axis.

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

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

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

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

xlin: Turns on linear scaling for the x axis.

xlog: Turns on logarithmic scaling for the x axis.

ylin: Turns on linear scaling for the y axis.

ylog: Turns on logarithmic scaling for the y axis.

xdiv: Controls the x axis division spacing.

ydiv: Controls the y axis division spacing.

xfull: Controls plotting of x axis full logarithmic decades.

yfull: Controls plotting of y axis full logarithmic decades.

xfudge: Changes the x axis "fudge factor."

yfudge: Changes the y axis "fudge factor."

axes: Controls the location of annotated axes.

border: Controls the plotting of a border around plots.

grid: Controls the plotting of grid lines in plots.

xgrid: Controls plotting of grid lines in the x direction.

ygrid: Controls plotting of grid lines in the y direction.

ticks: Controls the location of tick marks on plots.

title: Defines the plot title and attributes.

xlabel: Defines the x axis label and attributes.

ylabel: Defines the y axis label and attributes.

plabel: Defines general plot labels and their attributes.

filenumber: Controls plotting of file numbers next to each file.

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

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

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

tsize: Controls the text size attributes.

loglab: Controls labels on logarithmically scaled axes.

color: Controls color selection for color graphics devices.

line: Controls the linestyle selection in plots.

symbol: Controls the symbol plotting attributes.

qdp: Controls the "quick and dirty plot" option

width: Controls the width of data lines

beginframe: Turns off automatic new frame actions between plots.

endframe: Resumes automatic new frame actions between plots.

beginwindow: Begins plotting to a new graphics window.

window: Sets the location and shape of graphics windows.

xvport: Defines the viewport for the x axis.

yvport: Defines the viewport for the y axis.

floor: Puts a minimum value on logarithmically interpolated data.

null: Controls plotting of predefined null data values.

Graphics Control Module

- setdevice:** Defines a default graphics device to use in subsequent plots.
- begindevices:** Begins plotting to one or more graphics devices.
- enddevices:** Terminates one or more graphics devices.
- vspace:** Changes the maximum size and shape of plots.
- sgf:** Controls the SAC Graphics File device options.
- pause:** Tells SAC to pause for specified length of time.
- wait:** Tells SAC whether or not to pause between plots.
- print:** Print the most recent SGF file. Many plot commands also have a print option.

Graphics Action Module

- 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.
- plotpk:** Produces a plot for the picking of arrival times.
- plotpktable:** Table describing PLOTPK options.
- plotdy:** Plots data with error bars
- plotxy:** Plots one or more data files versus another data file.
- plotalpha:** Reads data from Tables and plots with labels
- plotc:** Annotates SAC plots and creates figures using cursor.
- plotctable:** Table describing PLOTc options.
- plotsp:** Plots spectral data in several different formats.
- plotpm:** Generates a "particle-motion" plot of pairs of data files.
- erase:** Erases the graphics display area.

Spectral Analysis Module

- fft:** Performs a discrete Fourier transform.
- keepam:** Keep only the amplitude component of the spectral file.
- ifft:** Performs an inverse discrete Fourier transform.
- plotsp:** Plots spectral data in several different formats.
- writesp:** Writes spectral files to disk as "normal" data files.
- readsp:** Reads spectral files written by @writesp@.
- lowpass:** Applies an IIR lowpass filter.
- highpass:** Applies an IIR highpass filter.
- bandpass:** Applies an IIR bandpass filter.
- bandrej:** Applies an IIR bandreject filter.
- wiener:** Applies an adaptive Wiener filter.
- fir:** Applies a finite-impulse-response filter.
- hanning:** Applies a "hanning" window to each data file.
- unwrap:** Computes amplitude and unwrapped phase.

khronhite: Applies a Khronhite filter to the data.
benioff: Applies a Benioff filter to the data.
divomega: Performs integration in the frequency domain.
mulomega: Performs differentiation in the frequency domain.
hilbert: Applies a Hilbert transform.
convolve: Computes the convolution of waveforms.
correlate: Computes the auto- and cross- correlation functions.
envelope: Computes the envelope function using a Hilbert transform.

Unary Operations Module

add: Adds a constant to each data point.
sub: Subtracts a constant from each data point.
mul: Multiplies each data point by a constant.
div: Divides each data point by a constant.
sqr: Squares each data point.
sqrt: Takes the square root of each data point. Error if any are negative
int: Performs integration using the trapezoidal rule.
abs: Takes the absolute value of each data point.
log: Takes the natural logarithm of each data point.
LOG10: Takes the base 10 logarithm of each data point.
exp: Computes the exponential of each data point.
EXP10: Computes the base 10 exponential ($10.**y$) of each data point.
dif: Differentiates data in memory.

Note: ADD, SUBTRACT, MULTIPLY, DIVIDE, SQRT, ABSolute, and EPS are also [INLINE](#) functions.

Binary Operations Module

merge: Merges (concatenates) a set of files to data in memory.
addf: Adds a set of data files to data in memory.
subf: Subtracts a set of data files from data in memory.
mulf: Multiplies a set of files by the data in memory.
divf: Divides data in memory by a set of data files.
binoperr: Controls errors that can occur during binary file operations.

Signal Correction Module

rq: Removes the seismic Q factor from spectral data.
rglitches: Removes glitches and timing marks.
rtrend: Removes the linear trend.
rmean: Removes the mean.
taper: Applies a symmetric taper to each end of data.
rotate: Rotates a pair of data components through an angle.

rotinc: Rotate a set of 3 perpendicular components.
interpolate: Interpolates evenly or unevenly spaced data to a new sampling rate.
quantize: Converts continuous data into its quantized equivalent.
stretch: Stretches (upsamples) data, including an optional interpolating FIR filter.
reverse: Reverse the order of data points.
smooth: Applies an arithmetic smoothing algorithm to the data.
decimate: Decimates (downsamples) data, including an optional anti-aliasing FIR filter.

Event Analysis Module

ohpf: Opens a HYPO formatted pick file.
chpf: Closes the currently open HYPO pick file.
whpf: Writes auxiliary cards into the @hypo@ pick file.
oapf: Opens a alphanumeric pick file.
capf: Closes the currently open alphanumeric pick file.
apk: Applies an automatic event picking algorithm.
plotpk: Produces a plot for the picking of arrival times.
plotpktable: Table describing PLOTPK options.
pickauthor: Controls author name when reading picks.
pickphase: Controls phase names to load into the SAC headers.

Signal Measurement Module

mtw: Determines the measurement time window for use in subsequent measurement commands.
markvalue: Searches for and marks values in a data file.
marktimes: Marks files with travel times from a velocity set.
markptp: Measures and marks the maximum peak to peak amplitude of each signal within the measurement time window.
rms: Computes the root mean square of the data within the measurement time window.

XYZ (3-D) Data Module

spectrogram: Calculates a spectrogram using all of the data in memory.
sonogram: Calculates a sonogram using all of the data in memory.
image: Plots 3D data files as color images
loadctable: Loads specified color table into 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 alphanumeric labels.
zcolors: Controls the color display of contour lines.

Instrument Correction Module

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

transfertable: Details for instrument types in transfer

Subprocess Invoking Module

spe: Initializes the Spectral Estimation Subprocess. SPE Commands

sss: Initializes the Signal Stacking Subprocess. SSS Commands

Analysis Tools

3C: GUI based three component processing tool

convolve: Computes the convolution of waveforms.

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

envelope: Computes the envelope function using a Hilbert transform.

filterdesign: Graphically display the filter's digital vs. analog characteristics.

linefit: Fits a line to data in memory and write results to blackboard

mat: Runs a MATLAB interactively or as a script on data in SAC

map: Uses GMT to create a MAP based on data in SAC

whiten: Whitens the data in memory

FK Spectrum

bbfk: Computes broadband frequency-wavenumber (FK) spectral estimate.

beam: Computes the beam.

Alphabetical Comamnd Listing

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

abbrev: Abbreviations for SAC Commands

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.

chpf: Closes the currently open HYPO pick file.

color: Controls color selection for color graphics devices.

comcor: Controls SAC's command correction option.

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

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.

data: Search for, build requests, and download data

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.

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.

event: Search for events by region, magnitude, and date

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.

filename: 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.

inbcm: 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-specified m-file or else get a Matlab prompt for interactive manipulation.

mathop: provides options for sequencing in inline expressions

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

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

metadata: Insert station and event metadata

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

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 an 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) from a user-defined preferences file, or interactively enter author list on

pickphase: Tell sac to read phase pick information (and possibly the author list) from a 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 the current output device.

plotc: Annotates SAC plots and creates figures using cursor.

plotcable:

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 column format on disk into memory.

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

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

response: Download responses in sac-polezero or evalresp format

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.

rotinc: Rotate a set of 3 perpendicular components.

rq: Removes the seismic Q factor from spectral data.

rtrend: Removes the linear trend.

saveimg: Saves displayed graphics windows in several formats

scallop: Calculate a spectrogram equal to the difference between two smoothed versions of 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 versions of 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.

station: Search for station by region, identifier, channel, and date

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 types 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 logarithmic 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 logarithmic 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)

ABBREV

SYNTAX

ABBREV

SAC Abbreviations

Abbreviation	Alternative	Alternative	Alternative	Alternative
AM	KEEPAM			
ARRAY	ARRAYMAP			
AS	ADDSTACK			
AXES	AXIS			
BD	BG	BEGG	BEGINDEVICES	
BEAM	BEAMFORM			
BF	BEGFR	BEGINFRAME		
BOEC	BINOPERR			
BP	BANDPASS			
BR	BANDREJ			
BW	BEGINWINDOW			
CAPF	CCIPF			
CH	CHNHDR			
COL	COLOR			
CONT	CONTOUR			
CONV	CONVERT			
CONVO	CONVOLVE			
COR	CORRELATE			
CS	CHANGESTACK			
DA	DISTANCEAXIS			
DC	DELETECHANNEL			
DEC	DECIMATE			
DFT	FFT			
DG	DATAGEN			
DS	DELETSTACK			
DW	DISTANCEWINDOW			
ED	EG	ENDG	ENDDEVICES	
EF	ENDFR	ENDFRAME		
ERA	ERASE			
EVAL	EVALUATE			
FD	FILTERDESIGN			
FG	FUNCGEN			
FN	FILENUMBER			
GS	GRAYSCALE			
GS	GLOBALSTACK			
GT	GTEXT			

... continued on next page

Abbreviation	Alternative	Alternative	Alternative	Alternative
H	HELP			
HAN	HANNING			
HCD	SGF			
HP	HIGHPASS			
IDFT	IFFT			
INICM	CLEARMEMORY			
INTERP	INTERPOLATE			
IS	INCREMENTSTACK			
LCT	CTABLE	COLORTABLE	LOADCTABLE	
LH	LISTHDR			
LP	LOWPASS			
LS	LISTSTACK			
M	MACRO			
MAP	GMAP	GMTMAP		
MARKP	MARKPTP			
MARKT	MARKTIMES			
MARKV	MARKVALUE			
MES	MESSAGE			
OAPF	OCIPF			
P	PLOT			
P1	PLOT1			
P2	PLOT2			
PA	PLOTALPHA			
PC	PLOT C			
PCOR	PLOT COR			
PH	PRINTHELP			
PICKA	PICKAUTHOR			
PICKPH	PICKPHASE			
PICKPR	PICKPREFS			
PPE	PLOTPE			
PPK	PLOTPK			
PPM	PLOT PM			
PROD	PRODUCTION			
PRS	PLOTRECORDSECTION			
PS	PLOTSTACK			
PSP	PLOTSP			
PSPE	PLOTSPE			
PXY	PLOTXY			
Q	END	DONE	EXIT	QUIT
QS	QUITSUB			
QS	QUITSUB			
R	READ			
R	READ			

... continued on next page

Abbreviation	Alternative	Alternative	Alternative	Alternative
RA	READALPHA			
RBBF	READBBF			
RCOR	READCOR			
RCSS	READCSS			
REP	REPORT			
RERR	READERR			
RGL	RGLITCHES			
RGSE	READGSE			
RH	READHDR			
ROT	ROTATE			
RS	MATPRS	RECORDSECTION		
RSDD	READSDD			
RSP	READSP			
RSUDS	READSUDS			
RTAB	READTABLE			
RTR	RTREND			
SAVE	SAVEIMG			
SC	SYSTEMCOMMAND			
SCP	SONO	SCALLOP	SONOGRAM	
SPG	SPECTROGRAM			
SS	SUMSTACK			
SYM	SYMBOL			
SYNC	SYNCH	SYNCHRONIZE		
TA	TIMEAXIS			
TRANS	TRANSFER			
TW	TIMEWINDOW			
VM	VELOCITYMODEL			
VR	VELOCITYROSETTE			
VSP	VSPACE			
W	WRITE			
WBBF	WRITEBBF			
WCOR	WRITECOR			
WCSS	WRITECSS			
WGSE	WRITEGSE			
WH	WRITEHDR			
WIN	WINDOW			
WNN	WRITENN			
WNR	WIENER			
WS	WRITESTACK			
WSDD	WRITESDD			
WSP	WRITESP			
WSPE	WRITESPE			
WT	WRITETABLE			

... continued on next page

Abbreviation	Alternative	Alternative	Alternative	Alternative
XLABL	XLABEL			
XVP	XWIND	XVPORT		
YLABL	YLABEL			
YVP	YWIND	YVPORT		
ZS	ZEROSTACK			

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

Note: ABS is also an [INLINE](#) function.

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:

```
SAC> READ F1 F2 F3
SAC> ADD 5.1 6.2
```

Note: ADD is also an [INLINE](#) function.

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:

```
SAC> READ FILE1 FILE2 FILE3
SAC> ADDF FILE4
```

To add two files to two other files:

```
SAC> READ FILE1 FILE2
SAC> 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. 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 constant 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:

```
SAC> AXES ON TOP
```

To turn off all axes annotation:

```
SAC> AXES OFF ALL
```

To turn axes annotation on for the bottom side and off for the rest:

```
SAC> 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 {range: 1-10}.

PASSES n: Set number of passes {n=1: causal, n=2: zero-phase}.

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 frequency. 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}
        {CORNER 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 v1 v2: Set corner frequencies to v1 and v2.

NPOLES n: Set number of poles {range: 1-10}.

PASSES n: Set number of passes {n=1: causal, n=2: zero-phase}.

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.:

```
SAC> BANDREJ NPOLES 4 CORNER 2 5
```

To apply a two-pole two-pass Bessel with the same corners.:

```
SAC> 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 $(1.0 + \text{EPS})$.

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.

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 $m \times n = 40,000$. 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.

REFERENCES

Nawab, SH, FU Dowla, and RT Lacos, Direction determination of wideband signals, IEEE Trans. Acous. Speech Sig. Proc., 33: (5), 1114-1122, 1985

LATEST REVISION

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 meridian 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 requisite 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.

LATEST REVISION

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 associated 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:

```
SAC> CUT A -0.2 N 512           (set up cut and read file)
SAC> READ FILE1
SAC> BEGINFRAME                 (turn off automatic framing)
SAC> XVPORT .1 .9               (define viewport and options)
SAC> YVPORT .7 .9
```

```

SAC> TITLE 'SEISMIC TRACE'
SAC> FILEID OFF          (turn off fileid and qdp option)
SAC> QDP OFF
SAC> PLOT                (plot the trace)
SAC> FFT WMEAN          (take transform of data)
SAC> XVPORT .1 .45      (second viewport and options)
SAC> YVPORT .15 .55
SAC> TITLE 'Amplitude Response (linlog)'
SAC> YLIM 1E-5 1
SAC> PLOTSP AM LINLOG   (plot the amplitude)
SAC> XVPORT .55 .9      (third viewport and options)
SAC> TITLE 'Amplitude Response (loglog)'
SAC> XLIM 1 60
SAC> PLOTSP AM LOGLOG   (plot amplitude again)
SAC> ENDFRAME           (resume automatic framing)
SAC> CUT OFF           (reset parameters used to default values)
SAC> FILEID ON
SAC> XLIM OFF
SAC> 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.:

```
SAC> BINOPERR NPTS FATAL  
SAC> READ FILE1  
SAC> ADDF FILE2  
SAC> ERROR: Header field mismatch: NPTS FILE1 FILE2
```

The file addition was not performed. Assume you now type:

```
SAC> BINOPERR NPTS WARNING
SAC> ADDF FILE2
SAC> 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 eight enumerated (I type) variables, 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 but 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 October 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 background 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:

```
SAC> COLOR RED INCREMENT
```

To select red data colors on a white background with a blue skeleton:

```
SAC> COLOR RED BACKGROUND WHITE SKELETON BLUE
```

To set up an incrementing data color list of red, white, and blue with an aquamarine (!!!) background:

```
SAC> 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.

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:

```
SAC> READ MYDATA
SAC> LISTHDR
FILE: MYDATA
      NPTS = 10000
      IFTYPE = GENERAL XYZ (3-D) FILE
      DEPMIN = 697.71
      DEPMAX = 1154.4
      NXSIZE = 100
      XMINIMUM = 82574.
```

```

XMAXIMUM = 86992.
NYSIZE = 100
YMINIMUM = 0.47439E+06
YMAXIMUM = 0.47720E+06
SAC> ZLEVELS RANGE 700 1150 INCREMENT 25
SAC> ZLINES LIST 1 2 3 4
SAC> TITLE 'Katmai topography from survey data [inc = 25 km]'
SAC> CONTOUR

```

The result of this example is shown in the figure below.

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:

```

SAC> READ MYDATA
SAC> ZLEVELS RANGE 700 1150 INCREMENT 25
SAC> ZLABELS ON LIST INT OFF OFF OFF
SAC> ZTICKS ON LIST 0 -1 -1 -1
SAC> ZLINES LIST 1
SAC> TITLE 'Katmai topography from survey data [labels and ticks]'
SAC> 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 Data File Format](#).

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 pulse shape with all other time series in memory.

SYNTAX

```
CONVOLVE {file} {PULSE n} {TRI v} {GAUSS v}
```

INPUT

PULSE n: The PULSE in filename n in memory (defaults to n=1)

file: Convolve with data in file (not in memory)

TRI v: Convolve with a triangle of half width v seconds

GAUSS v: Convolve with a Gaussian with standard deviation v seconds

CENTERED ON|OFF: Place zero time on center [ON] or beginning [OFF] of the pulse

DEFAULT VALUES

```
CONVOLVE PULSE 1 CENTERED OFF
```

DESCRIPTION

In seismology, convolution is used in two ways:

- 1) convolving a pulse shape with a time-series, and
- 2) convolving a response with a time series (or, more typically, deconvolving a response from a time series)

The equation for convolution takes a simple form in both the time and frequency domains; (1) is more easily described in the time domain, while (2) in the frequency domain. The CONVOLVE function here is oriented towards (1). Function [TRANSFER](#) is more appropriate for (2).

There are three ways to run CONVOLVE: (1) PULSE: read in N files and convolve one of them with all the others; (2) file: read in N files and convolve all of them with a file not among them; (3) Using a PREDEFINED FUNCTION (see below). Previous to SAC v102.0, the PULSE method was the only option, so we start there.

The pulse file is convolved with all other time-series files in memory using the discretized version of the following equation:

$$\text{conv}(\tau) = \text{INT} (f(t) g(\tau - t)) dt,$$

where f is the pulse file, g any from among the other signals in memory, and INT means integration (summation) over all times t. All signals in memory must have the same DELTA. The number of points in f, the pulse shape, cannot be greater than the number of points in any other file in memory. If NPTS(g) is the number of points in g and NPTS(f) the number in f, the number in conv is NPTS(g)+NPTS(f)-1. The start time for each output conv will be the same as B(g). Centering on the pulse shapes eliminates time shifts introduced by convolution. If CENTERED is OFF, B(f) = 0.0. CENTERED ON should only be used for time-symmetric pulses. As an example, let synthetic.sac be a synthetic time series created by a progra such as WKBJ, consisting of spikes with time offsets, amplitudes, and polarities representing phase arrivals. Let that file be called synthetic.sac. One wants to compare that synthetic with an observed sesimogram, and one creates a pulse file p_arrival.sac:

```
SAC> r p_arrival.sac synthetic.sac
SAC> PULSE 1 convolve CENTERED OFF
```

FILE OPTION

Given the same two files:

```
SAC> r synthetic.sac
SAC> convolve p_arrival.sac CENTER OFF
```

PREDEFINED FUNCTIONS

The option to convolve with TRI and GAUSS automatically computes these function with the correct time samplings. For GAUSS or TRI, there is no pulse-shape file read in; the predefined function takes the place of $f(t)$ in the above equation. Both functions are normalized such that they integrate to 1.0 and should not add amplitude (or "moment" in the language of sources). Both functions are time-symmetric. If the exponent in GAUSS(t) is $-0.5(t/v)^2$, v is the standard deviation.

Say one wants to look at file synthetic.sac simply as a spike series (no source pulse). The file may have Gibbs phenomena accompanying each arrival. Convoluting synthetic.sac with a predefined function will result in a cleaner looking seismogram:

```
SAC> r synthetic.sac
SAC> convolve GAUSS 0.04 CENTERED ON
or
SAC> r synthetic.sac
SAC> convolve TRI 0.04 CENTERED ON
```

All three cases (source, gauss, tri) are shown in the figure below. A Brune pulse is used as the p_arrival pulse. No relative scaling has been done except for the source pulse.

RELATION TO CORRELATE

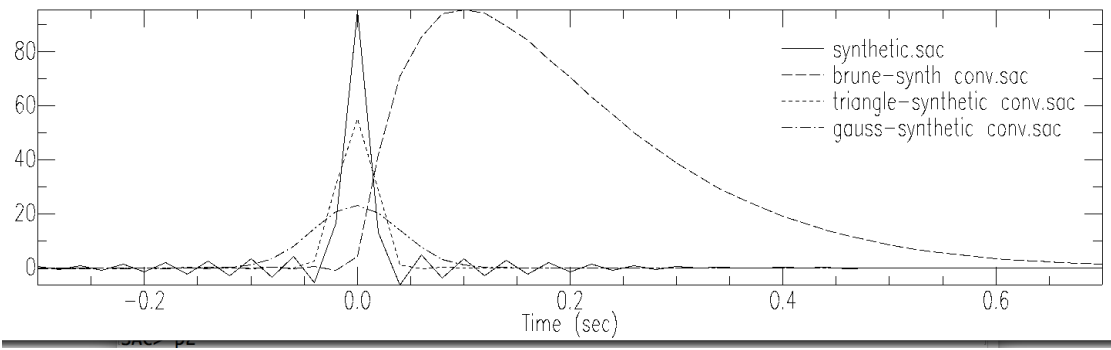
Note the similarity in the convolve equation and the one in [CORRELATE](#). Many convolve routines, including the original one in SAC, mistakenly use this similarity in deriving the procedure for convolution by simply perturbing the procedure used in [CORRELATE](#). The time series from a convolution (conv in the equation above) is typically a replacement time series for the time series g; for example, adding a pulse shapee to a synthetic seismogram. The relative times for f and g are not used in [CORRELATE](#), but in convolution they may play a role.

HEADER CHANGES

DEPMIN, DEPMAX, DEPMEN, NPTS

LATEST REVISION

2019 (Version 102.0)



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:

```
SAC> READ FILE1
SAC> PPK
SAC> ... use cursor to mark times T3 and T4.
SAC> READ MORE FILE2 FILE3
SAC> 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:

```
SAC> 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 OFF: No normalization

NORMALIZED ON: 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 NORMALIZED OFF NUMBER 1 LENGTH OFF TYPE RECTANGLE
```

DESCRIPTION

An auto-correlation is computed on the signal declared to be the master file, and a cross-correlation is calculated between the master file and each of the other signals in memory. This command computes correlations as defined by:

$$\text{corr}(\tau) = \text{int} (f(t) g(\tau + t)) dt,$$

where f is the master file, g is any from among the signals in memory, and int means summation over all times t . The number of points in the master file cannot be greater than the number of points in any other file in memory.

The input for SAC CORRELATE is files f and g . The output is the auto-correlation for f and the cross-correlation of f with g . The relative times for the two output files is most easily understood by examining the figure below. In this case, f is an isosceles triangle of duration 0.08 s starting at $t = 0$, which is produced by the commands:

```
SAC> fg triangle npts 9 begin -0.08 delta 0.02
SAC> cutim -0.04 0.04; ch b 0.0; write triangle.sac
```

In this example, *g* is a synthetic waveform with spikes at phase arrival times with amplitudes and polarity appropriate for that phase. The P arrival is at $t = 0$. The maximum amplitude for the auto-correlation always at output time $\tau = 0$. The P arrival spike has a high correlation with the triangle pulse. Note that the maximum for the output cross-correlation is at around -0.04 s, the onset time for the triangle auto-correlation. The commands for CORRELATE for *f* with *g* in this case are as follows:

```
SAC> r triangle.sac synthetic.sac
SAC> correlate normalized
SAC> write sac-autoc_triangle.sac sac-cc_triangle-synth.sac
SAC> r sac-autoc_triangle.sac
SAC> ch kevnrm "sac-autoc_triangle"
SAC> write over
SAC> r sac-cc_triangle-synth.sac
SAC> ch kevnrm "sac-cc_tri-synth"
SAC> write over
```

After *correlate normalized*, the filenames and KEVNM remain as they were before. Shown here is an example as to to label the output files.

If *N* is the number of points in *g* and *M* the number of points in *f*, the total number of points in *corr* is $N+M-1$. SAC does the calculation in the frequency domain so if *N* is not a power of 2, it is increased to N_2 , the next power of 2. Both *f* and *g* are padded with zeroes to N_2 points before the forward transforms. (It is wise to prepare all input files to minimize end effects.)

The windowing features of this command allow one 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.

ADDITIONAL EXAMPLES

To calculate the correlation functions using the third file in memory as the master file:

```
SAC> 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:

```
SAC> 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:

```
SAC> CORRELATE TYPE HANNING NUMBER 10 LENGTH 3.0
```

HEADER CHANGES

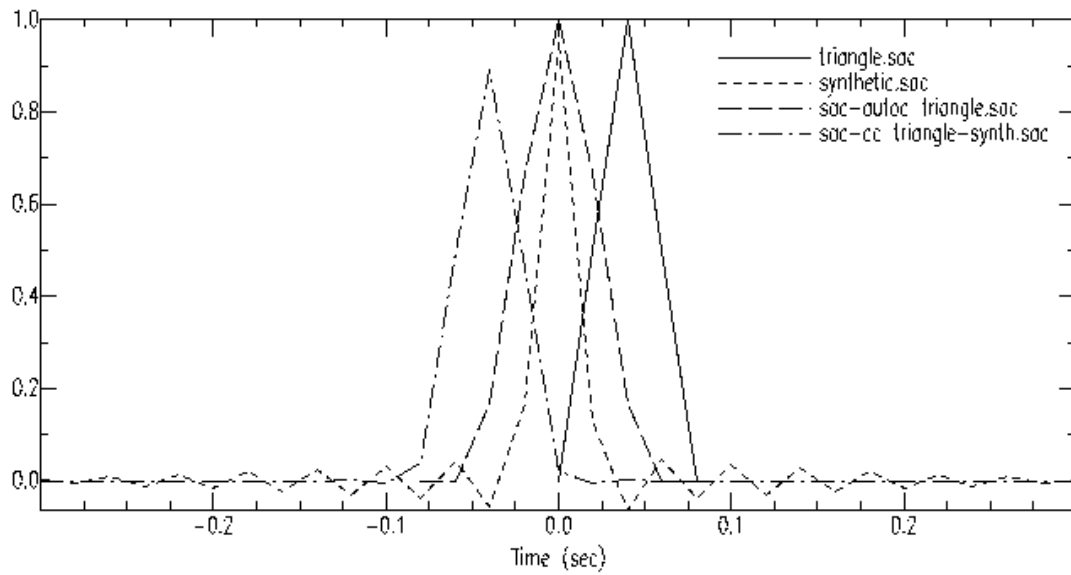
DEPMIN, DEPMAX, DEPMEN, NPTS, B

ACKNOWLEDGEMENTS

This command is based on an algorithm developed by Dave Harris (DBH)

LATEST REVISION

SAC v102.0 (added normalization)



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  KSOLE  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 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:

```

SAC> FG SEIS
SAC> RTR
SAC> P1
SAC> LH KSTNM KEVNM
SAC> CH KSTNM KAH KEVNM SOMEEVENT
SAC> ENVELOPE
SAC> PPK
SAC> user picks T1 at the change in frequency content
SAC> P1
SAC> LH KSTNM KEVNM
SAC> COMMIT          # replace with ROLLBACK or RECALLTRACE as appropriate
SAC> P1
SAC> 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](#), [DELETSTACK](#), [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 consistent 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](#) [READTABLE](#) [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 [DELETETSTACK](#) 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 [DELETETSTACK](#) for details.

Changing the COMMIT option in one delete command changes it in both for future calls. The COMMIT option for [DELETECHANNEL](#) and [DELETETSTACK](#) 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 the amount of a data file to be read. CUT does not act on data currently in memory: a call to CUT must be followed by a [READ](#) or [FUNCGEN](#) to take effect.

As discussed below, CUT can be preceded by [CUTERR](#), which controls errors if the chosen amount includes times outside B to E. The syntax will then be CUTERR {CUTERR option} ; CUT {CUT options}.

SYNTAX

```
CUT {ON|OFF|pdw|SIGNAL}
```

INPUT

ON: Turn cut option on but don't change pdw (see below).

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 the segment of a file one wishes to read. The most general form of a pdw is *ref offset ref offset*, where *ref* is a number or a reference value that is one of the following: B|E|O|A|F|T_n, 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 that is added to the reference value.

SIGNAL: Equivalent to typing: A -1 F +1.

DEFAULT VALUES

```
CUT OFF (equivalent to CUT b e)
```

```
CUTERR FILLZ for signal stacking subprocess, USEBE for others
```

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. 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 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 [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.

[CUTERR](#) is entered only if one wants to do something other than the default for an error.

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. The macro `cut_runs.m`, is in the SAC macros directory: `/${SACHOME}/macros/`. It, along with the results from entering `m ${SACHOME}/macros/cut_runs.m` after starting SAC. It is suggested that one runs this macro and compares the results with those from the macro in the help filefor [CUTIM](#):

```
fg seismo
write seismo.sac
echo on

* no cutting
lh b e a kztime

* begin to end---same as not cutting.
cut B E
read seismo.sac
lh b e a kztime

* First 3 secs of the file
cut B 0 3
read seismo.sac
lh b e a kztime

* From 0.5 secs before to 3 secs after first arrival
cut A -0.5 3
read seismo.sac
lh b e a kztime

* From 10 to 15 secs relative to zero
cut 10 15
read seismo.sac
lh b e a kztime

* From 0.5 to 5 secs relative to disk file start.
cut B 0.5 5
read seismo.sac
lh b e a kztime
```



```

* First 3 secs of the file and next 3 sec
cut b 0 3
read seismo.sac
write tmp.1

```

```

cut b 3 6
read seismo.sac
write tmp.2

```

```

cut off
read tmp.?
lh b e a kztime
title "top: cut b 0 3; bottom: cut b 3 6"
p1
save cut-test.pdf
\rm tmp.*

```

* Examples using CUTERR_

```

cut off
read seismo.sac
lh b e a npts kztime
cut a -0.5 15
read seismo.sac
lh b e a npts kztime
cuterr usebe ; cut a -0.5 15
read seismo.sac
lh b e a npts kztime
cuterr fillz ; cut a -0.5 15
read seismo.sac
lh b e a npts kztime

```

Note that one can 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.

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 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](#), [CUTIM](#)

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

CUTERR controls error conditions arising from bad **CUT** parameters. It is effectively an option of **CUT**, but is used with the following syntax: **CUTERR** {cuterr option} ; **CUT** {CUT options}. No action is taken until a subsequent **READ** or **FUNCGEN**. If the CUTERR option is FATAL, no data will be entered into memory. Data will be entered into memory for options USEBE or FILLZ with results as described above. The options are not case sensitive.

EXAMPLES

Command *fg seismo* reads into memory a seismogram:

```
SAC> CUT OFF
SAC> fg seismo
SAC> lh b, a, e, npts, kztime
b = 9.459999e+00          a = 1.046400e+01
e = 1.945000e+01        npts = 1000
kztime = 10:38:14.000
```

One gets the same result for *lh* if one precedes the *fg seismo* command with either *CUT A -5 E* or *CUTERR USEBE ; CUT A -5 E* because USEBE is the default for CUTERR. One gets a different result for the option FILLZ:

```
SAC> CUTEERR FILLZ ; CUT A -5 E
SAC> fg seismo
SAC> lh b, a, e, npts, kztime
b = 5.459999e+00          a = 1.046400e+01
e = 1.945000e+01        npts = 1400
kztime = 10:38:14.000
```

If on followed the CUTERR FILLZ example with *P1*, one would see an abrupt transition at time *B* because seismo has a nonzero offset at *B*. The following set of commands would produce a display with no offset: *fg seismo ; rtr ; taper ; write test.sac ; CUTEERR FILLZ ; CUT A -5 E ; read test.sac*.

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 pdw 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 and 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 wants 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 CUT. The macro `cutim_runs.m`, is in the SAC macros directory: `#{SACHOME}/macros/`. It, along with the results from entering `m #{SACHOME}/macros/cutim_runs.m` after starting SAC. It is suggested that one runs this macro and compares the results with those from the macro in the help file for [CUT](#):

```
echo on
fg seismo

* no cutting
lh b e a kztime

fg seismo
* 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 10 to 15 secs relative to zero
cutim 10 15
lh b e a kztime

fg seismo
* From 0.5 to 5 secs relative to disk file start.
cutim B 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

title "cutim b 0 3 b 3 6"
pl
save cutim_run.pdf
write *
```

Because CUTIM changes the file(s) in memory, the `fg seismo` is required between calls 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 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 reference value, in this case B.)

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

Version 102.0 Fixes behavior of [CUTIM](#) so it matches [CUT](#)

DATA

SUMMARY

Search for, build requests and download data

SYNTAX

```
DATA
{ catalog:event_id }
{ duration }
{ region w e s n }
{ origin lon lat }
{ radial min_radius max_radius }
{ time start end }
{ network net,to,use } { station sta,tions,to,get }
{ location 00,10,-- } { channel hhz,bh? }
{ in station_file }
{ out request_file }
{ max size_in_MB }
{ request request_file }
{ d | raw | quality | modified | best | merged | qc | unknown }
{ verbose }
{ ph5 }
{ sac }
{ read }
{ mseed | miniSEED }
{ more }
```

INPUT

avail | download: Display the Availability or Download data

catalog:event_id: Use an eventid to set the origin and start time. Catalogs may be usgs, isc, or gcmt. Use the [EVENT](#) command to get eventids

duration: Set the end time using a duration

region: Set the region to request stations within: west east south north

origin: Set the origin location: lon lat

radial: Set the radius limits in degrees: min_radius max_radius

time: Set the start and end times. End times can be set relatively

network: Set a collection of networks. Accepts lists and wildcards.

station: Set a collection of stations. Accepts lists and wildcards.

location: Set a collection of locations. Accepts lists and wildcards.

channel: Set a collection of channels. Accepts lists and wildcards.

in: Input station file for building the data request File format is space delimited with Network and Station in the 1st and 2nd columns. This file can be obtained from [STATION](#).

out: Output request file for saving the data request to run later

prefix: Prefix for miniSEED data files

ph5: Get additional data from the IRIS PH5 Web service. This option provides access to active-source data sets and their associated stations with high sample rates and time-limited deployments.

sac: Convert the miniSEED data into sac files and save. Station meta data is obtained for the stations. If an event is specified, event meta data is also used
mseed: (or miniSEED) Save the miniSEED files
read: Read the data following conversion from miniSEED to sac
more: Append to data existing data files in memory
max: Set the data download size for request: `size_in_MB`
verbose: Watch the details of the data search / download process

DEFAULT VALUES

```
DATA avail mseed max 200 loc * quality best
```

DESCRIPTION

Search for available data and possibly download the data. Data will be downloaded from multiple data repositories based on where the data is held. Requests are also limited in size, see *max size_in_MB*, to allow downloading data in reasonable sizes.

Time ranges can be set using a variety of formats, see examples below. End times can be set using a relative value with units. See [Data Access](#) for time formats.

EventID from [EVENT](#) can be used to identify origin times and locations. They will also be used to set sac file meta data for an event.

MiniSEED filenames will be defined when the request was made and the data center to which the request was made. Use *prefix* to set the starting portion of miniSEED files.

EXAMPLES

Let's get some data from station HELL, channels BHZ, from 2013/05/21 for a single hour:

```
SAC> data sta HELL cha BHZ net * loc * time 2013/05/21 +1h

## REQUEST 1/ 1
DATACENTER=NCEDC,http://www.ncedc.org
BK HELL 00 BHZ 2013-05-21T00:00:00 2013-05-21T01:00:00

No Data Downloaded, use miniSEED, sac, or read to download data
```

Note that the request is displayed to the screen but the data was not download. If you desire a saved, fuller version of the request, use the *out* option to save the request. If you want the data, run the same command again with

- *miniSEED* option to download miniSEED data
- *sac* option to get sac data
- *read* to place the data into memory

These options can be used individually or together. You can download data into SAC files and place those files into memory using the *sac* and *read* options together. For now, let's just get miniSEED data by appending the *miniSEED* option:

```
SAC> data sta HELL cha BHZ net * loc * time 2013/05/21 +1h miniSEED
Data Center: NCEDC,http://www.ncedc.org
Writing data to fdsnws.2019.07.12.10.38.15.NCEDC.mseed [276.00 KiB]
```

The data is written to a file with the prefix *fdsn*, the time of the request, the data center which provided the data, and a *mseed* extension. Note, the time in the file name has no relation to the data within the file.

Alternatively, if the request was saved to a file called *request.txt* the data can be obtained using:

```
SAC> data request request.txt
Data Center: NCEDC,http://www.ncedc.org
Writing data to fdsnws.2019.07.12.10.40.22.NCEDC.mseed [276.00 KiB]
```

Another option is to obtain data for a specific event using the duration and channel options. First we need an event:

```
SAC> event time 2019-05-14 +1d mag 7 cmt
Origin          Lat.   Lon.   Depth  Mag.   Agency EventID
2019-05-14T12:58:41 -3.96  152.53  18.80  7.60  MW    GCMT/- GCMT gcmt:11037207
```

We can then use the eventid at the end of the line to request data:

```
SAC> data gcmt:11037207 dur +10m net IU sta * cha BHZ rad 35 45 miniSEED
Data Center: IRISDMC,http://ds.iris.edu
Writing data to fdsnws.2019.07.12.10.53.30.IRISDMC.mseed [584.00 KiB]
```

An input station file using the *IN* keyword can also be used to request data:

```
SAC> station net YE sta SPOLE cha BHZ level station out southpole.txt
Net Sta  Lat.   Lon.   Elev.  SiteName
YE  SPOLE -89.9300  144.4400  2850.00  South Pole, Antarctica

SAC> data time 2014/03/01 +10m cha BHZ in southpole.txt miniSEED
Reading station file: southpole.txt
Data Center: IRISDMC,http://ds.iris.edu
Writing data to fdsnws.2019.10.30.19.31.34.IRISDMC.mseed [ 22.00 KiB]
```

It can be useful to see what may be download first; particularly if you forget to specify your desired channels and mistakenly download 10s of channels you are not interested in.

Large downloads will be divided into smaller chunks. The resulting miniSEED files will be merged back together during reading, provided gaps do not exist:

```
SAC> data net II sta BORG loc 00 cha BHZ time 2015/03/15 +1mo max 200

## REQUEST 1/ 3
DATACENTER=IRISDMC,http://ds.iris.edu
II BORG 00 BHZ 2015-03-15T00:00:00.000 2015-03-25T08:00:00.000

## REQUEST 2/ 3
DATACENTER=IRISDMC,http://ds.iris.edu
II BORG 00 BHZ 2015-03-25T08:00:00.000 2015-04-04T16:00:00.000
```

```
## REQUEST 3/ 3
DATACENTER=IRISDMC,http://ds.iris.edu
II BORG 00 BHZ 2015-04-04T16:00:00.000 2015-04-15T00:00:00.000
```

No Data Downloaded, use miniSEED, sac, or read to download data

Request files are written when requested using the *out* option or when data is requested/downloaded. The format of the request file starts with the request parameters followed by the individual data center requests. Lines starting with *#* are comments.

The request file is rewritten as individual requests are made allowing for larger requests to be stopped and restarted.

PH5 Example: If data originates from the PH5 Web service and metadata is desired in a resulting sac file, the PH5 option must be provided to tell sac to also search this repository for station metadata. The first example is able to download data, but results in an error when getting data. Adding the PH5 option in the second example below, allows the metadata to be obtained and added to the data:

```
SAC> data net 9A station 22770 time 2012-08-14 +10sec sac
Data Center: IRISPH5,http://ds.iris.edu
  SourceID                Start sample                End sample                Gap
XFDSN:9A_22770__D_P_Z    2012-08-14T00:00:00.000000 2012-08-14T00:00:09.996000 ==
Error 204 (HTTP):
```

Writing data to 9A.22770..DPZ.D.2012.227.000000.sac [10.38 KiB]

```
SAC> data net 9A station 22770 time 2012-08-14 +10sec sac PH5
Data Center: IRISPH5,http://ds.iris.edu
  SourceID                Start sample                End sample                Gap
XFDSN:9A_22770__D_P_Z    2012-08-14T00:00:00.000000 2012-08-14T00:00:09.996000 ==
Working on file: 9A.22770..DPZ.D.2012.227.000000.sac [ OK ]
Writing data to 9A.22770..DPZ.D.2012.227.000000.sac [ 10.38 KiB5
```

ERROR MESSAGES

SEE COMMANDS

[STATION](#), [EVENT](#), [RESPONSE](#), [METADATA](#), [DATA ACCESS](#)

LATEST REVISION

Version 102.0

DATAGEN

SUMMARY

Generates sample data files and stores them in memory.

SYNTAX

```
DATAGEN {MORE} {COMMIT|ROLLBACK|RECALLTRACE} {SUB name} {filelist}
```

where SUB is a datagen subdirectory and * name" is one of the following:

```
LOCAL  
REGIONAL  
TELESEISEM  
DEEP
```

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 | DEEP ,BREAK Specifics about the contents of these sub-directories is given below.

filelist: Each SUBdirectory has SAC files from a single event. A *filelist* is one or more SAC files in SUB.

DEFAULT VALUES

```
DATAGEN COMMIT SUB LOCAL cdv.z
```

DESCRIPTION

The DATAGUEN command operates much like the [READ](#) command except that for DATAGEN, any SAC file read into memory is from a subdirectory of `${SACHOME}/aux/datagen/SUB/`. Help file [TRAVELTIME](#) includes runs that use DATAGEN.

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:

```
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
```

DEEP EVENT

(new in SAC v102.0) The seismograms in DEEP are from the Sakhalin Island Event (May 12, 1990; M=6.5; Depth=611 km). The phases are very impulsive, and one can see many converted phases. There are vertical-, radial-, and transverse-component waveforms from eight broadband stations. The filenames are:

```
bla.r bla.t bla.z
ccm.r ccm.t ccm.z
cor.r cor.t cor.z
hrv.r hrv.t hrv.z
kev.r kev.t kev.z
kip.r kip.t kip.z
pas.r pas.t pas.z
tol.r tol.t tol.z
```

This data set was created as part of a focal-mechanism study, which used the polarity and amplitudes of the *SV* and *SH* phases. Horizontal components for a three-component seismograph must be rotated from *North-South* and *East-West* into *Radial* and *Transverse* to separate *SV* from *SH*. Positive *Radial* is forward and positive *Transverse* is to the right with respect to an observer facing the station on the epicenter-station line. To verify this:

```
SAC> datagen sub deep kev.*
  /usr/local/sac/aux/datagen/deep/kev.r ...kev.t ...kev.z
SAC> lh baz cmpaz
FILE: /usr/local/sac/aux/datagen/deep/kev.r - 1
  baz = 4.882252e+01          cmpaz = 2.288225e+02
FILE: /usr/local/sac/aux/datagen/deep/kev.t - 2
  baz = 4.882252e+01          cmpaz = 3.188225e+02
FILE: /usr/local/sac/aux/datagen/deep/kev.z - 3
  baz = 4.882252e+01          cmpaz = 0.000000e+00
SAC>
```

For the *Radial* direction, the *cmpaz* is in the opposition direction of the back azimuth, and, as discussed in [ROTATE](#), the *Transverse* is 90 degrees more than the *Radial*.

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.

LATEST REVISION

Version 102.0 (2020)

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:

```
SAC> READ FILE1
SAC> DECIMATE 7
SAC> 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
```

or:

```
[D]ELETE[C]HANNEL filename|filename|range {filename|filename|range ... }
```

INPUT

ALL: Deletes all files from memory. The user need not specify filenames or file numbers

filename: Name of a file in the file list.

filename: 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

```
SAC> dc 3 5 * deletes 3rd and 5th file.
SAC> dc S001.sz S002.sz * deletes named files.
SAC> dc 11-20 * deletes all the files from
* the 11th through the 20th,
* inclusive.
SAC> dc 3 5 11-20 S001.sz S002.sz * deletes all of the above.
```

ERROR MESSAGES

- 5106: File name not in file list
- 5107: File number not in file list

SEE COMMANDS

DELETSTACK, [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:

$$\text{OUT}(J) = (\text{DATA}(J+1) - \text{DATA}(J)) / \text{DELTA}$$

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:

$$\text{OUT}(J) = 1/2 * (\text{DATA}(J+1) - \text{DATA}(J-1)) / \text{DELTA}$$

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:

$$\text{OUT}(J) = 2/3 * (\text{DATA}(J+1) - \text{DATA}(J-1)) / \text{DELTA} - 1/12 * (\text{DATA}(J+2) - \text{DATA}(J-2)) / \text{DELTA}$$

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 divide 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:

```
SAC> READ F1 F2 F3  
SAC> DIV 5.1 6.2
```

Note: DIVIDE is also an [INLINE](#) function.

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:

```
SAC> READ FILE1 FILE2 FILE3
SAC> DIVF FILE4
```

To divide two files by two other files:

```
SAC> READ FILE1 FILE2
SAC> 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.

SYNTAX

DIVOMEGA value

DESCRIPTION

This command divides each point of a spectral file by its frequency given by:

$$\text{OMEGA} = 2.0 * \text{PI} * \text{FREQ}$$

This is analogous to integrating 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)

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 data should 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:

```
+ - * / **  
SQRT EXP ALOG ALOG10  
SIN COS TAN  
ASIN ACOS ATAN  
EQ NE LE GE LT GT  
ADD SUBTRACT MULTIPLY DIVIDE
```

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.

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.

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 slightly 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)

EVENT

SUMMARY

Search for events by region, magnitude and date

SYNTAX

```
EVENT {MAG min [max]}
      {TIME start end}
      {REGION w e s n}
      {RADIAL lon lat min_radius max_radius}
      {DEPTH min max}
      {OUTFILE filename}
      {TO blackboard-variable}
      {GCMT|ISC|NEIC|CMT|USGS|PDE}
      {VERBOSE}
```

INPUT

TIME: Set the start and end times. End times can be set relatively.

REGION west east south north: Set the geographic bounds of the search region

RADIAL min max: Set the center and radius limits of the search region in degrees

DEPTH min max: Set the minimum and maximum depth values in km.

MAG min [max]: Set the minimum and maximum magnitude values. A single value can be used to set a minimum magnitude and the maximum magnitude is set to 10.0.

GCMT: Search the Global Centroid Moment Tensor Catalog, CMT is an alias

ISC: Search the ISC Catalog

USGS: [Default] Search the USGS Catalog, PDE and NEIC are aliases

TO blackboard-variable: Save all event identifiers to a space separated list with a specific name

OUTFILE: Save the event search into a file. Files ending in .xml are saved as QuakeXML documents; those ending in .txt and others are saved as "csv" text

VERBOSE: [Not default] Watch the details of the event search process

DEFAULT VALUES

```
EVENT USGS
```

DESCRIPTION

Search an catalog for events. Results are returned to the screen or to a file if desired with the *OUTPUT filename* option.

Time ranges can be set using a variety of formats, see examples below. End times can be set using a relative value with units. See [DATA ACCESS](#) for time formats.

Event IDs are saved and can be used to identify origin times and locations in similar searching commands. They can also be used to set SAC file [METADATA](#) for an event.

EXAMPLES

To search for all events with Magnitudes ≥ 9.0 :

```
SAC> event mag 9
Origin          Lat.   Lon.   Depth  Mag.   Agency EventID
2011-03-11T05:46:24 38.30 142.37 29.00 9.10 mww US/official
usgs:official20110311054624120_30
2004-12-26T00:58:53  3.29  95.98 30.00 9.10 mw  US/official
usgs:official20041226005853450_30
1964-03-28T03:36:16 60.91 -147.34 25.00 9.20 mw  iscgem/official
usgs:official19640328033616_30
1960-05-22T19:11:20 -38.14 -73.41 25.00 9.50 mw  iscgem/official
usgs:official19600522191120_30
1952-11-04T16:58:30 52.62 159.78 21.60 9.00 mw  iscgem/official
usgs:official19521104165830_30
```

If you were expecting to find a specific event, it can be useful to search different catalogs. Let's use the UGSS / NEIC Catalog:

```
SAC> event mag 9 cmt
Origin          Lat.   Lon.   Depth  Mag.   Agency EventID
2011-03-11T05:47:32 37.52 143.05 20.00 9.10 MW  GCMT/- gcmt:3279407
2004-12-26T01:01:09  3.09  94.26 28.60 9.00 MW  GCMT/- gcmt:1916079
```

Let's try to focus into 1960 Chile by searching the southern hemisphere. Note, longitudes should be between -180 and 180:

```
SAC> event mag 9 usgs region -180 180 -90 0
Origin          Lat.   Lon.   Depth  Mag.   Agency EventID
1960-05-22T19:11:20 -38.14 -73.41 25.00 9.50 mw  iscgem/official
usgs:official19600522191120_30
```

Consider the deep 1994 Bolivian earthquake. Let's limit the depth and restrict the time range:

```
SAC> event time 1994-1 +1yr mag 8 usgs depth 610 1000
Origin          Lat.   Lon.   Depth  Mag.   Agency EventID
1994-06-09T00:33:16 -13.84 -67.55 631.30 8.20 mw  US/HRV usgs:usp0006dzc
```

Consider the 1989 Loma Prieta Earthquake in Northern California. This time we use a radial search between 0 and 5 degrees around 120 W, 38 N:

```
SAC> event time 1989-1 +1yr mag 6 radial -120 38 0 5
Origin          Lat.   Lon.   Depth  Mag.   Agency EventID
1989-10-18T00:04:15 37.04 -121.88 17.21 6.90 mh  NC/NC usgs:nc216859
```

Event Identifiers (eventid)

Events can be referenced in other commands using a unique identifier. The identifier is defined by *source:uniqueid*. Where the *source* is defined below and the unique identifier is the reference id provided by the agency. Most identifiers are less than 16 characters and will fit into the *KEVNM* header variable; the *official* catalog of significant earthquakes is a notable exception.

Source	Catalog	
isc	International Seismological Centre	http://www.isc.ac.uk/
usgs	United States Geological Survey	https://earthquake.usgs.gov/
gcmt	Global Centroid Moment Tensor	https://www.globalcmt.org/

ERROR MESSAGES

SEE COMMANDS

[STATION](#), [DATA](#), [RESPONSE](#), [METADATA](#), [DATA ACCESS](#)

LATEST REVISION

Version 102.0

EXP

SUMMARY

Computes the exponential of each data point.

SYNTAX

EXP

Note: EXP is also an [INLINE](#) function.

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.

FORTTRAN 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 spectral. 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

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:

```
SAC> FILEID LOCATION UL TYPE NAME
```

To define a special file id consisting of the station component, latitude, and longitude:

```
SAC> FILEID TYPE LIST KSTCMP STLA STLO
```

To include the name of the header field followed by a colon:

```
SAC> 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. These 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.:

```
SAC> 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.

REFERENCES

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 {frequency phase}
LINE {slope intercept}
QUADRATIC {a b c}
CUBIC {a b c d}
SEISMOGRAM
RANDOM {nfiles seed}
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 {frequency phase}: Sine wave with frequency in Hz and phase angle in degrees. Amplitude is one.

Note There is a factor of 2π in the phase argument:

$$\text{function} = 1.0 * \sin (2 * \text{Pi} * f * t)$$

LINE {slope intercept}: Linear function with slope given and intercept

QUADRATIC {a b c}: Quadratic function of the form:

$$a*t^{\{2\}} + b*t + c$$

CUBIC {a b c d}: Cubic function of the form:

$$a*t^{\{3\}} + b*t^{\{2\}} + c*t + d$$

SEISMOGRAM: Sample seismogram. This differs from other function options for fg in that there are no further options. Specifically, DELTA, NPTS, and BEGIN options are ignored for seismogram (abbreviated seis).

RANDOM {nfiles seed}: Random sequence (Gaussian white noise) generator. Number of random sequence files to generate is first and the "seed" used to generate the first random number is second. This seed value is stored in USER0 so that you can regenerate the same random sequence at a later time if desired.

Note This is not really random, not even psuedo-random.

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.  
FUNCGEN SINE 0.05 0.0  
FUNCGEN LINE 1 1  
FUNCGEN QUADRATIC 1 1 1  
FUNCGEN CUBIC 1 1 1  
FUNCGEN RANDOM 1 12357  
FUNCGEN SEISMO [there are no further options]
```

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 information when transferred 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
```

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 [SAC_MACROS](#) for details.

The options to this command let you control where the values are printed. One can print them to the terminal or append them to the end of a text file. One can include the variable name and an equals sign before the value or one can just have the value printed. One can have a newline placed after each value printed in a list or one can have them placed on a single line. One 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 [READ](#) command with the ALPHA option 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:

```
SAC> SETBB C1 2.45 C2 4.94
```

To later print their values you would use this command:

```
SAC> GETBB C1 C2
C1 = 2.45
C2 = 4.94
```

To print just their values on a single line:

```
SAC> GETBB NAMES OFF NEWLINE OFF C1 C2
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](#)

LATEST REVISION

Sept. 1, 1988 (Version 10.3E)

GRAYSCALE

SUMMARY

Produces grayscale images of data in memory.

Deprecation Warning

This command is deprecated as it relies on external software.

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-request@CS.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 be 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 manipulated and displayed 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
```

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 hardware 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:

```
SAC> 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

REFERENCES

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:

```
SAC> HELP
```

Now lets say you want information on several commands:

```
SAC> HELP READ CUT BEGINDEVICE PLOT
```

SAC begins displaying the **READ** help package. After a full page, it asks if you've seen enough:

```
MORE?  
SAC> YES
```

SAC displays the rest of the help package on **READ**, and then begins displaying the help package on the **CUT** command:

```
MORE?  
SAC> NO
```

SAC stops displaying the **CUT** help package and begins displaying the **BEGINDEVICE** help package:

```
MORE?  
SAC> 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},{CORNER 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 {range: 1-10}.

PASSES n: Set number of passes {n=1: causal, n=2: zero-phase}.

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.:

```
SAC> HIGHPASS NPOLES 4 CORNER 2
```

To apply a two-pole two-pass Bessel with the same corner.:

```
SAC> 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

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:

```
SAC> history
```

To repeat command 1:

```
SAC> !1
```

To repeat the last command:

```
SAC> !!
```

To repeat the 2nd-to-last command:

```
SAC> !-2
```

To repeat the command starting with ps:

```
SAC> !ps
```

ERROR MESSAGES

none

LATEST REVISION

March 03, 1997

Help Introduction

Documentation for SAC includes

A user manual w/tutorial, and command descriptions including commands in the spe and sss subprocesses. The general syntax for using SAC's integrated help package is: help "help-file", Where "help-file" is one of the files listed below.

Reference Manuals:

```
intro      tutorial  analysis  graphics  sac_macros
inline    blackboard  input_output  file_format  SAC_Library
SGF_format  Scripts      Error_messages
data_access
```

The user manual comes in 13 parts. The first part, intro, gives background information about SAC and includes a table of contents which briefly describes the other parts.

Alphabetical and functional listings

```
alphalist      funclist
```

Main SAC commands (ns means not supported)

```
3c

about          abs          absolutevalue  add          addf
apk           arraymap      axes

bandpass      bandrej      bbfk          beam         begindevices
beginframe    beginwindow  benioff       binoperr

border

capf          chnhdr       chpf          color
comcor       commit       contour       convert      convolve
copyhdr      correlate    cut           cuterr      cutim

datagen       decimate     deletechannel  deltacheck  divomega
dif          div

echo          enddevices   endframe      envelope     erase
evaluate     exp          exp10         external
external_interface

fft          fileid       filenumber    filterdesign
fir         floor

getbb        gmap        gmtmap       grayscale   grid
gtext

hanning      help        help_intro   highpass    hilbert
history     howto_external
```

ifft interpolate	image	inicm	installmacro	int
keepam	khronhite			
line listhdr loglab	linefit load loglin	linlin loadctable loglog	linlog log lowpass	log10
macro markvalue mulf	map merge mulomega	mat message	markptp mtw	marktimes mul
news	nplotc	null		
oapf	ohpf			
pause plot plotctable plotsp	pickauthor plot1 plotdy plotxy	pickphase plot2 plotpk print	picks plotalpha plotpktable printhelp	plabel plotc plotpm production
qdp	quantize	quit	quitsub	
read readgse readtable rmean rtrend	readbbf readhdr recalltrace rms	readcss readsdd report rollback	readdb (ns) readsp reverse rotate	readerr readsuds rglitches rq
scallop smooth sqr subf	setbb sonogram sqr symbol	setdevice sort stretch synchronize	setmacro spectrogram sub systemcommand	sgf
taper transfer	ticks transfertable	title tsize	trace tutorial	transcript
unsetbb	unwrap			
vspace				
wait window writehdr	whpf write writesdd	width writebbf writesp	wiener writecss	wild writegse
xdiv xlim	xfudge xlin	xfull xlog	xgrid xvport	xlabel
ydiv ylim	yfudge ylin	yfull ylog	ygrid yvport	ylabel
zcolors	zlabels	zlevels	zlines	zticks

Signal Stacking Subprocess Commands

addstack	changestack	deletestack	deltacheck	distanceaxis
distancewindow	globalstack	incrementstack	liststack	plotrecordsection
plotstack	sumstack	timeaxis	timewindow	traveltime
velocitymodel	velocityroset	writestack	zerostack	

Spectral Estimation Subprocess Commands

spe_cor	mem	mlm	pds	plotcor
plotspe	quitsub	read	writecore	writespe
spe				

Misc Notes

From within SAC, you can print any help file using the command `printhelp`:

```
SAC> printhelp ylin
```

would print the help file `ylin`.

You can also inspect the ascii documentation files under the help subdirectory of the `$SACAUX` directory. The contents of this file are in `$SACAUX/help/hlpintro`.

Please report any problems to sac-help@iris.washington.edu.

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. To minimize end-effects, it is suggested one precede a call to INT with [RMEAN](#); [TAPER](#).

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 line-drawing off. Figure below shows linestyles for n = 1 through 10. SOLID is n = 1, DOTTED is n = 8.

INCREMENT {ON}: For multiple data files in a plot, increment linestyle from linestyle list for each data file in the plot.

INCREMENT OFF: Do not increment linestyle for multiple data files.

LIST STANDARD: Change to the standard linestyle list (1 2 3 ..).

LIST nlist: Change the content of the linestyle list. Enter list of linestyle numbers (e.g., 3 1 2 ..).

FILL ON/OFF: Turn filling on/off.

FILL POS_COLOR/NEG_COLOR: Fill color for positive/negative section of the seismogram trace. Turns FILL on. See [COLOR](#) for color options.

FILL LIST STANDARD: Use Standard color list for Color Filling.

FILL LIST POS_COLOR/NEG_COLOR: Turn on color filling incrementing multiple colors in a list are available to set colors. Colors are specified in either Color Name or Number

DEFAULT VALUES

```
LINE SOLID INCREMENT OFF LIST STANDARD FILL OFF
```

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, and the [WIDTH](#) command controls the line-width. 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 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.

EXAMPLES

To select an incrementing linestyle using the STANDARD list - 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 excursions on a seismogram with red and the negative excursions with blue. If one leaves out the 0, the colored regions would be outlined with a black line:

```
SAC> fg seismo
SAC> line 0 fill red/blue
SAC> pl
```

SEE COMMANDS

[SYMBOL](#), [COLOR](#), [WIDTH](#)

LATEST REVISION

Version 102.0

01	_____
02	-----
03	-----
04	-----
05	-----
06	-----
07	-----
08	-----
09	-----
10	-----

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 is "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:

```
SAC> LISTHDR PICKS COLUMNS 2
```

To get a default listing of the third and fourth files in the data file list:

```
SAC> LISTHDR FILES 3 4
```

To list the values of the begin and end time only:

```
SAC> LISTHDR B E
```

To define a special list of the station parameters:

```
SAC> LISTHDR KSTNM STLA STLO STEL STDP
```

To reuse this special list later during the same execution:

```
SAC> LISTHDR SPECIAL
```

To set default two column output:

```
SAC> 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 section 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 precede 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 {range: 1-10}.

PASSES n: Set number of passes {n=1: causal, n=2: zero-phase}.

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.:

```
SAC> LOWPASS NPOLES 4 CORNER 2
```

To apply a two-pole two-pass Bessel with the same corner.:

```
SAC> 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.:

```
SAC> setbb wf $1  
SAC> setbb ts $2  
SAC> r %wf  
SAC> ch b (%ts + &1,b&)  
SAC> 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 for corner frequency larger than Nyquist frequency.** See Chapter 4 of Rabiner and 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.

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 macros directory: `${SACHOME}/macros/`.

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]

MAPSCALE: On | [Off] - Plot a Distance Scale on the Map

PLOTSTATIONS: [On] | Off - Plot all the Stations from seismograms

PLOTEVENTS: [On] | Off - 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 - 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
```

TOPOgraphy: On | [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
```

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:

```
SAC> MTW T4 T5  
SAC> 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:

```
SAC> MTW A 0 30  
SAC> MARKP L 3. TO T7
```

HEADER CHANGES

Tn, USER0, KTn, KUSER0

SEE COMMANDS

[MTW](#), [OAFP](#)

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 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.:

$$\text{time}(j) = \text{origin} + \text{distance}/\text{velocity}(j)$$

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:

```
SAC> MARKTIMES DISTANCE 340. TO T4
```

To select a different velocity set:

```
SAC> 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:

```
SAC> MARKT ORIGIN GMT 1984 231 12 43 17 237 TO T2
```

HEADER CHANGES

T_n, KT_n

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 occurrence 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:

```
SAC> MARKVALUE GE 3.4 TO T7
```

To later perform that same search in the measurement time window 10 seconds long beginning at T4:

```
SAC> MTW T4 0 10
```

```
SAC> MARKVALUE
```

HEADER CHANGES

T_n, K_{Tn}

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	igual		nxsize	
t3	knetwk	nzsec	isynt		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)`:

```
SAC> 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 filename`) 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 precedence. Normally, multiplication and division have a higher precedence 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 (concatenates) a set of files to data in memory.

SYNTAX

```
MERGE {VERBOSE} {GAP ZERO|INTERP} {OVERLAP COMPARE|AVERAGE}  
      {TOLERANCE_DELTA v} {filelist}
```

INPUT

GAP ZERO | INTERP: How to handle data gaps

- ZERO - Fill with 0.0 amplitude [default]
- 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

TOLERANCE_DELTA v: Tolerance between computed delta and delta reported in the header. Default tolerance is 1e-6

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 concatenated 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.

TOLERANCE_DELTA

For collections of files with high sample rates, long durations or both the computation of time becomes less precise due to the limited precision of the delta value in the header.

To avoid this, a calculated time-sampling is computed based on the difference in begin times and with sufficient precision. If the calculated time-sampling is within the tolerance specified by the TOLERANCE_DELTA, then the calculated time-sampling is used in place of the value reported in the header. Use VERBOSE for more information about the calculated and header-defined deltas:

```
SAC> merge verbose ./*.SAC  
merge calculating delta  
merge calculated delta: 4.000000000000000e-03 (assuming no-gaps)  
delta in header 4.000000189989805e-03 (first file)  
difference: 1.899898e-10 = abs( calc_delta - delta )  
tolerance: 1.000000e-06  
using calculated delta  
merging 24 files => 3600000 data points
```

The calculated delta assumes no gaps and no overlaps for the input files as is determined as

```
calc_delta = mean ( (file_{i+1}.b - file_{i}.b) / file_{i}.npts )
```

where the file_{i} comes before file_{i+1}.

To turn off computing the observed delta, use a value of ≤ 0.0

```
SAC r *SAC
SAC> merge verbose tol 0.0
merging 24 files => 3599999 data points
merge: Gap zero fill: [n: 899999 t: 3599.996171]
merge: Gap zero fill: [n: 1049999 t: 4199.996199]
.....
```

As soon as a significant gap or overlap is found between two files, all successive file boundaries will have a significant gap or overlap as the mismatch will increase with increasing n. In this data set, each file has 150000 points with a dt of 0.004s.

The output of Merge is a single file with the same name as the first file, so it was necessary to use the read statement.

EXAMPLES

To merge FILE3, FILE4, FILE1 and FILE2 into one file:

```
SAC> READ FILE1 FILE2
SAC> MERGE FILE3 FILE4
```

or:

```
SAC> READ FILE1 FILE2 FILE3 FILE4
SAC> MERGE
```

or:

```
SAC> DELETECHANNEL ALL
SAC> 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:

```
SAC> READ data/event1/elko.z
SAC> 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

Version 102.0

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:

```
SAC> MESSAGE FINISHED
FINISHED
```

To send a message with blanks, you must use single or double quotes:

```
SAC> MESSAGE 'Job has finished.'
Job has finished.
```

LATEST REVISION

January 8, 1983 (Version 8.0)

METADATA

SUMMARY

Metadata is data about data content and is critical for analysis of timeseries data such as seismic data. IRIS has various web-based tools that allow users to query, view, and download metadata. This command allows one to insert available metadata into SAC data files.

SYNTAX

```
METADATA {catalog:eventid}  
        {file metadata_file}  
        {ph5}  
        {verbose}
```

INPUT

catalog:eventid: Set the event to use when specifying event metadata. An eventid can be obtained using from the [EVENT](#) command

file metadata_file: Use the metadata from a file. The file may be either a text file or an xml file. There are *important* differences between xml and text files. Input metadata files must have channel level metadata, see the [STATION](#) command.

ph5: Get additional data from the IRIS PH5 Web service. This option provides access to active-source data sets and their associated stations with high sample rates and time-limited deployments.

VERBOSE: Watch the details of the metadata insertion process

DEFAULT VALUES

```
METADATA
```

DESCRIPTION

Insert metadata into files currently in memory. Metadata, i.e. sac headers, is only added for the station if the station id (Network Station Location and Channel) are defined in the sac header. Metadata for the event is only added if the event is specified as an command option.

Metadata for the station includes

- stla: Station Latitude
- stlo: Station Longitude
- stel: Station Elevation (in meters)
- stdp: Station Depth (always 0.0)
- cmpaz: Component Azimuth (degrees clockwise from North)
- cmpinc: Component Inclination (See discussion below)

Metadata for the event includes

- evla: Event Latitude
- evlo: Event Longitude
- evel: Event Elevation (always 0.0)

- evdp: Event Depth in km
- origin time: (kzdate and kztime)
- iztype: set to IO
- kevnm: (if short enough)

Setting of kzdate, kztime and iztype is equivalent to the following SAC commands:

```
SAC> ch o gmt 1994 160 00 33 16 230
SAC> ch iztype IO
SAC> ch allt (-1.0 * &1,o)
```

EXAMPLE

Adding Station Metadata

For data files missing station metadata, it is straightforward to insert this information by reading in the data and issuing the METADATA command. For this to work, station identifiers (KNETWK, KSTNM, KHOLE, KCMPNM) need to be defined:

```
SAC> read BK.BRK.00.BHZ.Q.2010.235.000000.sac
SAC> lh kcmpnm kstnm khole kcmpnm stlo stla stel stdp cmpaz cmpinc columns 2

FILE: BK.BRK.00.BHZ.Q.2010.235.000000.sac - 1
-----

kcmpnm = BHZ          kstnm = BRK
khole = 00           kcmpnm = BHZ

SAC> metadata
Working on file: BK.BRK.00.BHZ.Q.2010.235.000000.sac [ OK ]

SAC> lh kcmpnm kstnm khole kcmpnm stlo stla stel stdp cmpaz cmpinc columns 2

FILE: BK.BRK.00.BHZ.Q.2010.235.000000.sac - 1
-----

kcmpnm = BHZ          kstnm = BRK
khole = 00           kcmpnm = BHZ
stlo = -1.222610e+02 stla = 3.787352e+01
stel = 4.940000e+01  stdp = 2.700000e+00
cmpaz = 0.000000e+00 cmpinc = 0.000000e+00
```

Adding Station Metadata from a file

If the metadata exists in a .txt or .xml file, this information can be applied to existing data files. Input data files must have channel level metadata.

```
SAC> read YE.SPOLE..BHZ.M.2013.304.000000.sac
SAC> lh kcmpnm kstnm khole kcmpnm stlo stla stel stdp cmpaz cmpinc columns 2

FILE: YE.SPOLE..BHZ.M.2013.304.000000.sac - 1
-----
```

```
kcmpnm = BHZ          kstnm = SPOLE
khole =              kcmpnm = BHZ
```

```
SAC> metadata file station_info.xml
Working on file: YE.SPOLE..BHZ.M.2013.304.000000.sac [ OK ]
```

```
SAC> lh kcmpnm kstnm khole kcmpnm stlo stla stel stdp cmpaz cmpinc columns 2
```

```
FILE: YE.SPOLE..BHZ.M.2013.304.000000.sac - 1
```

```
-----
kcmpnm = BHZ          kstnm = SPOLE
khole =              kcmpnm = BHZ
  stlo = 1.444400e+02  stla = -8.993000e+01
  stel = 2.850000e+03  stdp = 0.000000e+00
  cmpaz = 0.000000e+00  cmpinc = 0.000000e+00
```

Adding Event Metadata

Event metadata can be added to a data file by issuing the metadata command with a specified eventid, here we use an eventid of gcmt:3279407. (See [EVENT](#) for a discussion of EventID.) Stationmetadata will also automatically be added:

```
SAC> event time now -10y mag 9 gcmt
Origin          Lat.   Lon.   Depth  Mag.   Agency EventID
2011-03-11T05:47:32 37.52 143.05 20.00 9.10 MW  GCMT/- GCMT gcmt:3279407
```

```
SAC> data gcmt:3279407 net II sta BORG cha BHZ loc '00' dur lmin miniseed
Data Center: IRISDMC,http://ds.iris.edu
Writing data to fdsnws.2019.11.01.15.56.00.IRISDMC.mseed [ 8.00 KiB]
```

```
SAC> read fdsnws.2019.11.01.15.56.00.IRISDMC.mseed
```

```
SAC> metadata gcmt:3279407
Working on file: II.BORG.00.BHZ.M.2011.070.054732.sac [ OK ]
```

```
SAC> lh stlo stla evlo evla kevm
```

```
FILE: II.BORG.00.BHZ.M.2011.070.054732.sac - 1
```

```
-----
stlo = -2.132680e+01  stla = 6.474740e+01
evlo = 1.430500e+02  evla = 3.752000e+01
kevm = gcmt:3279407
```

Alternatively, event metadata can be added at the time of downloading data by specifying an eventid and converting to SAC or reading the data into memory. The example below uses the eventid gcmt:3279407:

```
SAC> data gcmt:3279407 net II sta BORG cha BHZ loc '00' dur lmin read
Data Center: IRISDMC,http://ds.iris.edu
SourceID          Start sample          End sample          Gap
XFDSN:II_BORG_00_B_H_Z 2011-03-11T05:47:32.019900 2011-03-11T05:48:31.969900 ==
Working on file: II.BORG.00.BHZ.M.2011.070.054732.sac [ OK ]
```

```
SAC> lh stlo stla evlo evla kevnm
```

```
FILE: II.BORG.00.BHZ.M.2011.070.054732.sac - 1
```

```
-----  
stlo = -2.132680e+01          stla = 6.474740e+01  
evlo = 1.430500e+02          evla = 3.752000e+01  
kevnm = gcmt:3279407
```

PH5 Example: Given a SAC file missing station metadata where that metadata exists in the PH5 Web service and not in the IRIS Web service, the PH5 option must be specified to obtain and insert the metadata. The PH5 options is appropriate for active source stations:

```
SAC> read 9A.22770..DPZ.D.2012.227.000000.sac  
SAC> lh stlo stla
```

```
FILE: 9A.22770..DPZ.D.2012.227.000000.sac - 1
```

```
-----  
SAC> metadata  
Error 204 (HTTP):
```

```
SAC> metadata ph5  
Working on file: 9A.22770..DPZ.D.2012.227.000000.sac [ OK ]
```

COMPONENT INCLINATION

The component inclination is defined differently between SAC and SEED/miniSEED. Sac defined the inclination as degrees from vertical, what is typically thought of as an inclination. SEED/miniSEED defines the component inclination as a component dip, defined as degrees from horizontal down into the Earth.

Value	SEED / miniSEED	SAC
Horizontal	0.0	90.0
Up	-90.0	0.0
Down	90.0	180.0

Data retrieved automatically is handled correctly and the conversion is seamless. For metadata from a file specified by the *file metadata_file* command option, the type of file determines if the component is define as a dip (miniSEED) or inclination (SAC). This is the same convection used in mseed2sac <<https://seiscode.iris.washington.edu/projects/mseed2sac>> for consistency.

File Type	Separator	Convention	Conversion Value
text	, (comma)	SAC / Inclination	0
text	' ' (bar)	SAC / Inclination	90
xml	n/a	miniSEED / Dip	90

METADATA FORMAT

Metadata in *xml* format adheres to the FDSN Station format found at <http://www.fdsn.org/xml/station/>, but it is easier to create the metadata file format using the **STATION** command with a *channel* option.

Metadata in *text* format is defined as one channel per line separated by either ',' (commas) or '|' (bars). The value of the columns are in the following order (the same as mseed2sac)

- Network
- Station
- Location
- Channel
- Station Latitude
- Station Longitude
- Station Elevation
- Station Depth
- Component Azimuth (North = 0, East = 90)
- Component Inclination - Coordinate system is defined by the delimiter, see note above
- Instrument Name/Type
- Scale Factor (ignored)
- Scale Frequency (ignored)
- Scale Units (ignored)
- Sample Rate (ignored)
- Start time
- End time

SEE COMMANDS

[STATION](#), [DATA](#), [RESPONSE](#), [EVENT](#)

LATEST REVISION

Version 102.0

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:

```
SAC> READ F1 F2 F3  
SAC> MUL 5.1 6.2
```

Note: MULTILPLY is also an [INLINE](#) function.

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:

```
SAC> READ FILE1 FILE2 FILE3
SAC> MULF FILE4
```

To multiply two files by two other files:

```
SAC> READ FILE1 FILE2
SAC> 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.

SYNTAX

MULOMEGA

DESCRIPTION

This command multiplies each point of a spectral file by its frequency given by:

$$\text{OMEGA} = 2.0 * \text{PI} * \text{FREQ}$$

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)

PLOT C

SUMMARY

Annotates SAC plots and creates figures using cursor.

SYNTAX

```
PLOT C {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

```
PLOT C 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 **PLOT C** 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 **PLOT C** 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 `PLOT` command (in create or replay mode), and then execute the `ENDFRAME` command to resume automatic framing.

EXAMPLES

An example of the use of `PLOT` 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:

```
** generate filter response
SAC> FG IMPULSE NPTS 1024
SAC> LOWPASS C2 NPOLES 7 CORNER 0.2 TRANBW 0.25 A 10
SAC> FFT

** set up desired plot options
SAC> AXES ONLY LEFT BOTTOM
SAC> TICKS ONLY LEFT BOTTOM
SAC> BORDER OFF
SAC> FILEID OFF
SAC> QDP OFF
SAC> VSPACE 0.75
SAC> BEGINFRAME
SAC> PLOTSP AM LINLIN
SAC> PLOT CREATE FILE BANDPASS
SAC> ENDFRAME
```

`PLOTSP` was used to produce the curve of the filter response and the two axes. `PLOT` was used interactively to produce the annotation (i.e., the lines, arrows, and labels.) The viewspace command constrains the plot to 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:

```
SAC> BEGINDEVICES SGF
SAC> BEGINFRAME
SAC> PLOTSP
SAC> PLOT REPLAY
SAC> ENDFRAME
```

A SAC graphics file will be created containing the annotated plot. Two examples (one somewhat frivolous) of the use of `PLOT` 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](#)

Action Options Table

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 PLOT.C .
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

LATEST REVISION

October 11, 1984 (Version 9.1)

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:

```
SAC> 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
		<ul style="list-style-type: none">• "A" if an automatic pick (APK)
		<ul style="list-style-type: none">• "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 similar event location programs. Picks from the automatic picker (APK) and manual pick plot (PLOT PK) 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](#), [PLOT PK](#), [WHPF](#), [CHPF](#)

REFERENCES

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 essentially 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.

For example, 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 probably 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 disappeared. 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 essentially 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 description 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 transferred 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 transferred 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 the line if the nearest time point is in the top part of the plot, otherwise it is below the line.

CROSS: A vertical line at the time pick and a horizontal line at the nearest data point.

WIDTH v: For CROSS change width of pick display to v. Default: 0.1 of full plot width

HEIGHT v: For CROSS or HORIZONTAL, change height pick display to v. Default: 0.1 of full plot height

DEFAULT VALUES

```
PICK ON 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:

```
SAC> PICKS T4 C T5 C T6 C W 0.3 H 0.1
```

Other time pick displays 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:

```
SAC> PLABEL 'Sample seismogram' POSITION .12 .5
SAC> PLABEL 'from earthquake'
SAC> PLABEL 'on January 24, 1980'
SAC> PLABEL 'in Livermore Valley, CA'
```

An additional tiny label could be placed in the lower left hand corner:

```
SAC> 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 {n|OFF|ON}} {PRINT {pname} }
```

INPUT

ABSOLUTE: Plots files treating time as an absolute. Files with different begin times will be shifted relative to each other.

RELATIVE: Plots files relative to that file's begin time.

PERPLOT n: Plots n files per frame.

PERPLOT ON: Plots n files per frame. Use last value for n.

PERPLOT OFF: Plots all files on one 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 its 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 frequency domain, b, e, and delta are reset to 0, the Nyquist frequency, and df respectively. 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:

```
SAC> READ MNV.Z.AM KNB.Z.AM ELK.Z.AM
SAC> XLIM 0.04 0.16
SAC> YLIM 0.0001 0.006
SAC> LINLOG
SAC> SYMBOL 2 INCREMENT
```

```
SAC> TITLE 'Rayleigh Wave Amplitude Spectra for NESSEL'  
SAC> XLABEL 'Frequency (Hz)'  
SAC> PLOT2  
SAC> FFT  
SAC> XLIM off YLIM off  
SAC> line increment list 1 3  
SAC> 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** precede 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:

```
SAC> 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)

PLOT C

SUMMARY

Annotates SAC plots and creates figures using cursor.

SYNTAX

```
PLOT C {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

```
PLOT C 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 **PLOT C** 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 **PLOT C** 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 **PLOT** command (in create or replay mode), and then execute the **ENDFRAME** command to resume automatic framing.

EXAMPLES

An example of the use of **PLOT** 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:

```
SAC> FG IMPULSE NPTS 1024
SAC> LOWPASS C2 NPOLES 7 CORNER 0.2 TRANBW 0.25 A 10
SAC> FFT
SAC> AXES ONLY LEFT BOTTOM
SAC> TICKS ONLY LEFT BOTTOM
SAC> BORDER OFF
SAC> FILEID OFF
SAC> QDP OFF
SAC> VSPACE 0.75
SAC> BEGINFRAME
SAC> PLOTSP AM LINLIN
SAC> PLOT CREATE FILE BANDPASS
SAC> ENDFRAME
```

PLOTSP was used to produce the curve of the filter response and the two axes. **PLOT** was used interactively to produce the annotation (i.e., the lines, arrows, and labels.) The viewspace command constrains the plot to 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:

```
SAC> BEGINDEVICES SGF
SAC> BEGINFRAME
SAC> PLOTSP
SAC> PLOT REPLAY
SAC> ENDFRAME
```

A SAC graphics file will be created containing the annotated plot. Two examples (one somewhat frivolous) of the use of **PLOT** 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](#)

ACTION OPTIONS TABLE

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 PLOT.C .
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 [PLOT C](#).

- A line beginning with an asterisk is treated as a comment line.

LATEST REVISION

March 20, 1992 (Version 10.6e)

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.:

```
SAC> READALPHA CONTENT YY MYFILE
SAC> 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 n: Plots n files per frame.

PERPLOT ON: Plots n files per frame. Use last value for n.

PERPLOT OFF: Plots all files on one frame.

BELL OFF: keystrokes are silent.

BELL ON: Rings bell on each keystroke in active window.

ABSOLUTE: Plot files treating time as an absolute. Files with different begin times will be shifted relative to each other.

RELATIVE: Plot files relative to that file's begin time.

GMT: Display pick times in absolute (GMT) format.

ZERO: Display pick times relative to each files's zero time.

REFERENCE v: Turn reference line display on and change reference value to v.

REFERENCE ON: Turn reference line display on using last value of v.

REFERENCE OFF: Turn reference line display off.

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 OFF: Do not save pick locations in the blackboard.

SAVELOCS ON: Save pick locations (from I cursor command) to blackboard variables.

DEFAULT VALUES

```
PLOTPK PERPLOT OFF ABSOLUTE REFERENCE OFF MARKALL OFF SAVELOCS ON
```

DESCRIPTION

The format of the PLOTPK plot is similar to the [PLOT1](#) plot. When the crosshairs comes on, you position the crosshair center at a point on the displayed waveform and enter single-character keystrokes to perform various functions. Some but not all keystrokes produce graphic output on the

screen. Error and output messages are printed at the top of the plot window. Mouse clicks will identify the time and amplitude of the cursor position. A mouse drag will initiate a zoom operation.

If one tries to resize the plot window during PPK, SAC exits with an error message. Plot window resizing must be done before entering PPK.

Picks that are currently in the header are automatically displayed on the screen as vertical lines at the pick time with labels. Output from some keystrokes can be directed to the SAC header, to an alphanumeric file, or a hypo pick file if open (see [OAPF](#) and [OHPF](#)). If REFERENCE is on with a value v, if z is typed, there will be a "zero-level" horizontal line drawn at the cursor's vertical position plus two additional horizontal lines at +/- v.

If option SAVELOCS is on, cursor locations from the L cursor option are saved to blackboard variables as follows:

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 [PLOTPKTABLE](#) lists valid cursor options for plotpk.

EXAMPLE

The following example starts by opening an alphanumeric pick file, reads in a time-series file UOM1.Z (that is in directory doc/examples/), and calls ppk. In ppk I first expand the window to fill the screen, then it picks the P arrival and the P amplitude and then the S arrival time and amplitude. After exiting ppk, I display the picks from the header and the bb variables. I close the APF file and display its contents:

```
SAC> oapf name
SAC> r UOM1.Z
SAC> ppk
  ** Description                Key (case-insensitive)
  ** Zoom in to the P arrival:  x then x (or click and drag)
  ** Pick P arrival time:      i u 0 p (Impulsive, Up, Quality 0, P arrival)
  ** Pick P amplitude:         l
  ** Expand view:              o
  ** Zoom in to the S arrival:  x then x (or click and drag)
  ** Pick S arrival time:      i u 1 s (Impulsive, Up, Quality 1, S arrival)
  ** Pick S amplitude:         l
  ** Quit PPK:                 q
SAC> lh picks
  AMARKER = 0.25379 (IPU0)      TOMARKER = 0.44422 (ISU1)
SAC> getbb      ! Note that picks are not saved as bb variables, only 1
TLOC1 = 0.274248
TLOC2 = 0.465606
XLOC1 = 'NOV 04 (308), 2011 17:17:22.967'
XLOC2 = 'NOV 04 (308), 2011 17:17:23.159'
YLOC1 = 811.226
YLOC2 = 12580.1
SAC> capf
SAC> cat APF
  UOM1.Z      IPU0      2011308 17 17 22.95      18.04
```

```
UOM1.Z      LOC      2011308 17 17 22.97      811.2
UOM1.Z      ISU1     2011308 17 17 23.14      -8.896
UOM1.Z      LOC      2011308 17 17 23.16      1.258e+04
SAC>
```

The P pick is stored in header variable A, and the (IPU) in KA. The S pick is stored in T0 with (ISD1) in KT0.

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.
- 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.
 - Adjust cursor position and retry. Plot is always in ABSOLUTE mode.

SEE COMMANDS

[PLOT1](#), [OHPF](#), [OAPF](#), [APK](#), [PLOTPKTABLE](#)

LATEST REVISION

Version 102.0

PLOTPKTABLE

PLOTPK Cursor Options

char	meaning
a	Designate first arrival in header {1,7}.
b	Display previous plot (if there is one).
c	Evaluate first arrival and end of event {1,4,7}.
d	Set phase direction to DOWN (precedes P or S phase pick).
e	Set phase onset to EMERGENT (precedes P or S phase pick)
f	Define end of event {1,2,3,7}..
g	Display picks to terminal in HYPO format {4}.
h	If CHPF precedes call to PLOTPK , write picks to HYPO pick file {3,4}.
i	Set phase onset to IMPULSIVE (precedes P or S phase pick).
j	Set noise level {2,6,8}.
l	Saves and lists x and y values for cursor location {2,4}.
m	Compute maximum amplitude waveform {2,3,5}.
n	Display next plot (if there is one).
o	Zoom Out => Displays previous plot window. A maximum of five windows are saved.
p	Designate P wave arrival time {1,2,3,7}.
q	Terminate PLOTPK immediately.
s	Designate S wave arrival time {1,2,3,7}.
t	Designate user time Tn in header {1,2,7}. Next keystroke should be an integer between 0 and 9.
u	Set phase direction to UP (precedes P or S phase pick).
v	Define a Wood-Anderson waveform {2,5}.
w	Define a general waveform {2,5}.
x	<p>Set Zoom window</p> <ul style="list-style-type: none"> • Set beginning of zoomed plot • If x is second keystroke, set end of zoomed window and display • If s is second keystroke, defined zoomed plot limits are saved and used as xlims for subsequent plots
z	Set zero (reference) level {2,6,8}. If reference is on with a value f, two more horizontal lines are drawn at +/- f from the z line.
'+'	Set phase direction to PLUS (precedes P or S phase pick).

... continued on next page

char	meaning
'-'	Set phase direction to MINUS (precedes P or S phase pick).
' '	Set the phase direction to NEUTRAL (precedes P or S phase pick).
'n'	Set phase quality to n, n=0,1,2,3, or 4. (precedes P or S phase pick)

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 window.
- {5} Terminal echo is a box encompassing waveform.
- {6} Place horizontal cursor at designated level before typing command letter.
- {7} Terminal window echo is a labeled vertical line.
- {8} Terminal window echo is a labeled horizontal line.

LATEST REVISION

Version 102.0

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:

```
SAC> READ XYZ.T XYZ.R
SAC> XLABEL 'Radial component'
SAC> YLABEL 'Transverse component'
SAC> TITLE 'Particle-motion plot for station XYZ'
SAC> PLOTPM
```

If you wanted to plot only a small part of each file around the first arrival time, you could use the [XLIM](#) command:

```
SAC> XLIM A -0.2 2.0
SAC> 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:

```
SAC> BEGINWINDOW 2
SAC> PLOTPK
SAC> ... mark the portion you want using X and S
SAC> ... terminate PLOTPK with a Q
SAC> BEGINWINDOW 1
SAC> 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:

```
SAC> READ FILE1
SAC> FFT
SAC> 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:

```
SAC> READALPHA CONTENT YNNN MYFILE
SAC> LINE INCREMENT ON
SAC> 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 [BEGIN-DEVICES](#) to turn on the [SGF](#) device. Use the [SGF](#) command to set preferred 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:

```
SAC> PRINTHELP
```

Now lets say you want information on several commands:

```
SAC> 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 is 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:

```
SAC> READ FILE1 FILE2
SAC> BEGINDEVICES TERMINAL
SAC> 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:

```
SAC> BEGINDEVICES SGF
SAC> 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 Schaffer (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:

$$\text{ERROR} = 1/12 * (\text{DELTA} * \text{LEVEL}^2)$$

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:

$$(1/6) * \text{LEVEL}^2.$$

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. Schaffer; 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:

- [SPE](#) Spectral Estimation Subprocess
- [SSS](#) Signal Stacking Subprocess

SEE COMMANDS

[SPE](#), [SSS](#)

LATEST REVISION

October 11, 1984 (Version 9.1)

READ

SUMMARY

Reads data from 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
```

ALL options *must* precede 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 SEG-Y format. This format allows one waveform per file.

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.

AUTO-DETECTION: SAC will attempt to determine the types of files for reading. Known file types include SAC binary, SAC alphanumeric, SEG-Y, and miniSEED. If the type of file is specified using ALPHA or SEG-Y it will be assumed all files are of that type.

miniSEED: All data within a miniSEED file will be read into memory.

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 an absolute pathname is:

```
/disk/dir/subdir/file
```

Examples of a relative pathnames is:

```
./subdir/file  
../subdir2/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. The first relative-pathway example assumes the user is in subdir.

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, ROLL-BACK 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 transferring 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 pieces 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.

EXAMPLES

In the following examples it is assumed 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:

```
SAC> READ F01 F02 F03
```

The following command produces the same result using the wildcard operator:

```
SAC> READ F*
```

This command also produces the same result by using the concatenation operator:

```
SAC> READ F0[1,2,3]
```

To read the second, third, and fourth files:

```
SAC> R F02 ?03
```

The following examples show the use of the MORE option:

```
SAC> R F03 G03
```

files F03 and G03 are in memory:

```
SAC> R F01 F02
```

files F01 and F02 are in memory:

```
SAC> R MORE F03 G03
```

files F01, F02, F03, and G03 are in memory

This example uses the MORE option when a filename was misspelled:

```
SAC> R F01 G02 F03
WARNING: File does not exist: G02
Will read the remainder of the data files.
```

files F01 and F03 are in memory:

```
SAC> 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:

```
SAC> READ F01
SAC> HIGHPASS CORNER 1.3 NPOLES 6
SAC> READ MORE F01
SAC> 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":

```
SAC> READ DIR EVENT1 F01 F02
```

files in directory /me/data/event1 are read:

```
SAC> READ F03 G03
```

files in same directory are read:

```
SAC> READ DIR EVENT2 *
```

all files in /me/data/event2 are read:

```
SAC> 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

Version 102.0

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} {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.

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 [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 header 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

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 consistent 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 see [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 transferred 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 override 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 exhausted (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 michael 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 beginning 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](#), [CRR](#) (COMMIT, ROLLBACK, RECALLTRACE)

LATEST REVISION

See the [READ](#) command. Oct. 27, 1998 (Version 00.58)

READDB

SUMMARY

Reads data from Oracle database into memory.

Note 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, see [READCSS](#) and [PICKPREFS](#).

SYNTAX

```
READGSE {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.

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.

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 [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 gsefile(s).

filelist: The name(s) of one or more gse files.

DEFAULT VALUES

```
READGSE * MAXMEM 0.3 VERBOSE OFF COMMIT
```

DESCRIPTION

Any command which loads data into memory is monitored to maintain a level of confidence in the event information when transferred 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, see [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** precede 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 consistent 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** precede 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 information when transferred 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 information when transferred 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 information when transferred 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

Statient 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: "|p-wwssn"
4: "sp-dwwssn"
5: "|p-dwwssn"
6: "hglp-lamont"
7: "|p-hglp-lamont"
8: "sp-sro"
9: "|p-sro"
10: "sp-asro"
11: "|p-asro"
12: "sp-rstn"
13: "|p-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: "Kinematics-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: "magnitic 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 column 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** precede 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 information when transferred 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:

```
SAC> 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 precedes 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:

```
SAC>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:

```
SAC> 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:

```
SAC> READTABLE CONTENT XN6 FORMAT \ (5F10.20X,2F10.2\ ) FILEC
```

Another way that means less typing but is slightly less efficient is given below:

```
SAC> 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. Precede 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,  
WINDOW
```

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.

WINDOW: The current window sizes

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:

```
SAC> REPORT COLOR
COLOR option is ON
DATA color is YELLOW
INCREMENT data color is OFF
SKELETON color is BLUE
BACKGROUND color is NORMAL
```

To get the names of the HYPO and card image pick files:

```
SAC> REPORT APF HPF
Alphanumeric pick file is MYPICKFILE
HYPO pick file is HYPOPICKFILE
```

LATEST REVISION

Version 102.0

RESPONSE

SUMMARY

Download responses in sac-polezero or evalresp format

SYNTAX

```
RESPONSE {sacpz | polezero | pz | evalresp | evresp | resp }
          {network network} {station station}
          {location location} {channel channel}
          {time time} {time start end}
          {verbose}
```

INPUT

sacpz | polezero | pz: Download SAC polezero response files (default)

evalresp | evresp | resp: Download evalresp response files

network: Set the network to download the response of

station: Set the station to download the response of

location: Set the location to download the response of

channel: Set the channel to download the response of

time time-of-data: Set a specific time that the response includes

time start end: Set the time range to download responses

verbose: Watch the details of the response download process

DEFAULT VALUES

```
RESPONSE SACPZ
```

DESCRIPTION

Download responses for time-series data in memory or specified using the command options.

Data in Memory. To download responses for data in memory, the following header values must be defined:

- knetwk (Network)
- kstnm (Station name)
- khole (Location identifier)
- kcmpnm (Component)
- kzdate/kztime (Absolute time of the data, Optional)

An example of getting the response for XE.DOOR..BHZ in either sacpz or evalresp:

```
SAC> r XE.DOOR..BHZ.M.1994.160.003345.sac
SAC> response
Writing data to
SAC_PZs_XE_DOOR__BHZ_1994-06-09T00:33:45_1994-06-09T01:33:44 [ 1.02 KiB]
SAC> response evalresp
Writing data to RESP.XE.DOOR..BHZ [ 49.80 KiB]
SAC>
```


For specific responses, use the net, sta, loc, and cha options:

```
SAC> response resp net BK sta BKS loc * cha BHZ
Writing data to RESP.BK.BKS..BHZ [ 73.52 KiB]
```

Time Range: If no time period is specified, then the total duration is requested. See [DATA ACCESS](#) for time formats.

Wildcards are possible only in the location and channel command options.

Output file names are in the form of:

- SAC_PZs_net_sta_loc_cha_time
- SAC_PZs_net_sta_loc_cha_start_end
- RESP.net.sta.loc.cha

If any of the parts (net, sta, loc, or cha) are wildcards, they are left blank in the filename. If the file already exists, a period with a number is appended to the filename, e.g.:

- RESP.BK.BKS..BHZ
- RESP.BK.BKS..BHZ.0
- RESP.BK.BKS..BHZ.1

File [TRANSFER](#) discusses sac POLEZERO files and RESP files and shows how they are used to instrument-correct a time-series data file.

ERROR MESSAGES

- **3264: Response request either requires a data file with meta data** knetwm, kstnm, khole, kcmpnm [kzdate/kztime] or net, sta, loc, and cha [time/start/end]

SEE COMMANDS

[STATION](#), [DATA](#), [METADATA](#), [EVENT](#), [DATA ACCESS](#), [TRANSFER](#)

LATEST REVISION

Version 102.0

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 \geq 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 acquisition system and by timing marks produced by some data acquisition systems. It checks each data point to see if its 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

DEPMAX DEPMIN 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:

```
SAC> MTW T1 T2
SAC> RMS TO USER4
```

To compute the corrected root mean square using a noise window 5 seconds long ending at the header field T3:

```
SAC> MTW T1 T2
SAC> 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 (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 (clockwise rotation looking down).

REVERSED: Output (horizontal) components with the second lagging the first by 90 degrees (counterclockwise rotation looking down).

DEFAULT VALUES

```
ROTATE TO GCP NORMAL
```

DESCRIPTION

Pairs of data components are rotated in this command. Each pair must have the same station and event header variables, NPTS, B, and DELTA

TO option: Both components must be horizontals: CMPAZ must be defined and CMPINC must be 90 degrees. After the rotation is completed, the first component of each pair will be directed along the angle given after the TO keyword. If the TO GCP option is used, the station and header fields STLA, STLO, EVLA, and EVLO must be defined so that the backazimuth (BAZ) can be calculated. After the rotation, the first component will be directed along the angle given by the station-event backazimuth plus or minus 180 degrees (to keep the final angle between 0 and 360 degrees). This component therefore points from the event toward the station (the *radial* direction), and the second component is called "transverse" or "tangential". (The (upward) vertical, radial, and transverse directions form a left-handed coordinate system.)

The NORMAL and REVERSED options 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. ROTATE TO GCP REVERSED results in a transverse component in the opposite direction from ROTATE TO GCP NORMAL, a convention preferred by some researchers.

EXAMPLES

To rotate a pair of horizontals to a specified angle for the first component:

```
SAC> READ XYZ.N XYZ.E
SAC> ROTATE TO 123.43
```


To rotate two sets of horizontals so that the first component in each set along the great circle path and then write SAC files for the radial and transverse components:

```
SAC> READ ABC.N ABC.E DEF.N DEF.E
SAC> ROTATE TO GCP
SAC> W ABC.R ABC.T DEF.R DEF.T
```

HEADER CHANGES

CMPAZ, CMPINC, KCMPNM, DEP*

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)

ROTINC

SUMMARY

Rotate a set of 3 perpendicular components either around the vertical (V) direction (TO VRT or TO VNE) or around the Transverse (T) direction (TO LQT)

SYNTAX

Rotations around the vertical (cmpinc = 0 degrees) direction:

```
ROTINC {TO VRT|VNE}
```

Rotations around the transverse (cmpaz = baz-90) direction:

```
ROTINC {TO LQT} {ANGLE angle}
```

or:

```
ROTINC {TO LQT} {iP | iS} {vP alpha} {vS beta} {RAY p} {VERBOSE}
```

INPUT

TO VRT: rotate into vertical,radial, transverse coordinate system

TO VNE: rotate into system aligned with vertical, north, and east

TO LQT: rotate into P, SV, and SH coordinate system

ANGLE a: Rotation through an angle a (V to L) around T. L cmpinc = a

iP: incident P wave: calculated L cmpinc is the incident P apparent angle

iS: Incident S wave: calculated Q cmpinc is the incident SV apparent angle

vP v: v is P-wave velocity at the surface (default 5.8)

vS v: v is S-wave velocity at the surface (default 3.36)

RAY p: ray parameter, p in sec/km (no default value, so must be set)

VERBOSE: prints information to the screen for iP or iS options (default off)

DESCRIPTION

Given three perpendicular components of a recorded time series, ROTINC rotates the set into the selected coordinate system. It works its way through all records in memory until it fails to find three consecutive perpendicular traces. It assumes vertical up (cmpinc = 0.0), cmpaz measured clockwise from N. T to the right from the vertical-radial plane facing towards the station (baz - 90). Hence, VNE, VRT, and LQT are all left-handed coordinate systems.

For TO LQT, the "TO LQT" can be left out because that is the only option for which there are additional arguments. For TO LQT, there are two choices:

1 ROTINC ANGLE a: rotation around the T axis (a=0.0 is up)

2.ROTINC iP or ROTINC iS: rotation around T, through a calculated angle that is calculated assuming the P or SV wave is incident on the free surface (top of a half space). For iP, the output cmpinc for L is often called the apparent angle: the arctangent of the radial (R) amplitude divided by the vertical (V) amplitude. For iS, the output cmpinc for Q is the incident SV apparent angle. For ROTINC iS, the incident SV angle must be less than the critical angle: $vP * RAY < 1.0$.

EXAMPLES

Rotations around the vertical are straightforward: either TO VNE or TO VRT with no arguments.

Here are examples for TO LQT. For a rotation around T through an angle 24.44:

```
SAC> ROTINC ANGLE 24.44 (or ROTINC TO LQT ANGLE 24.44)
      Input: rotation angle from V to L
```

For a free-surface correction (Note: one must specify RAY.) Here $p = \sin(21.32)/5.8 = 0.063$, where 21.32 is the P incident angle:

```
SAC> ROTINC iP RAY 0.063 VERBOSE
      Incident P wave; free-surface response
      vP: 5.80 km/s vS: 3.36 km/s Ray Param: 0.063000 s/km
      Apparent angle: 24.44
SAC> message &1,cmpinc &1,kcmpnm
      24.4416
      L
```

Generally vP and vS are the velocities in the surface layer. If the wavelength of the dominant arrivals are larger than the thickness of that layer, one may have to average over two or more layers. This can be tested by plotting the vertical-radial particle motion ([PLOTPM](#)) as the output motion is along the apparent angle. The ray parameter will not be changed. Here is an example for the same ray but assuming the wavelengths are long enough to average over the whole crust:

```
SAC> ROTINC iP vp 8.04 vs 4.47 RAY 0.063 VERBOSE
      Incident P wave; free-surface response
      vP: 8.04 km/s vS: 4.47 km/s Ray Param: 0.063000 s/km
      Apparent angle: 32.71
```

Here is an example for iS ($p = \sin(23.12)/3.36 = 0.1169$, where 23.12 is the SV incident angle:

```
SAC> ROTINC iS RAY 0.1169 VERBOSE
      Incident S wave; free surface response
      vP: 5.80 km/s vS: 3.36 km/s Ray Param: 0.116900 s/km
      Apparent angle: 115.82
SAC> message &2,cmpinc &2,kcmpnm
      115.817
      Q
```

There is often interference with other arrivals for SV on the vertical, so the SV on V and R are generally not that similar. Hence V-R particle motion is less useful for SV than it is for P.

EQUATIONS

The relevant cmpinc (apparent angle) is computed using the following equations:

- iP:
$$L \text{ cmpinc} = \text{atan}(2*Vs*p*\text{sqrt}(1-(Vs*p)**2) / (1-2*(Vs*p)**2))$$
- iS:
$$Q \text{ cmpinc} = \text{atan}(\sqrt{p}*(1-2*(Vs*p)**2)) / (2*Vs**2*p*\text{sqrt}(1-(\sqrt{p}*p)**2)))$$

where p is the ray parameter. Equations are based on Problem 5.6 in Aki and Richards, 2002, pg. 184.

HEADER CHANGES

CMPINC, CMPAZ, KCMPNM, DEPMAX, DEPMIN, DEPMEN

AUTHOR

This command was originally contributed by Frederik Tilmann in 2015,

LATEST REVISION

May 2017 (Version 102.0)

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:

$$\text{AMP_corrected}(F) = \text{AMP_uncorrected}(F) * \text{Exp}((\text{pi}*R*F) / (Q*C))$$

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|VERBOSE]
```

INPUT

QUIET: If given, suppresses screen output

VERBOSE: If given, there is output to the screen about slope removed, etc.

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. The sequence RMEAN ; RTREND gives the same result as simply entering RTREND.

OUTPUT: The best-fitting straight line parameters for the last file in the data file list are written to blackboard variables beginning with RTR.

- RTR_SLP is the slope of the line.
- RTR_SDSLIP 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 coefficient.

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) [No longer included in the binary SAC package]
- 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 .xpm file is sufficient. Because of potential problems with portability, the PNG format is not enabled in the default builds of SAC.

The title in a save xxx.pdf is made with a font that is larger than that used in a title for p1 or p1 print because in producing the PDF file, a postscript font (with higher resolution) is used.

It is possible to produce plots in .png format if one builds SAC from sources (see `_${SACHOME}/README`).

Output .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 $X/Y = 11/8.5 = 1.2941$. To make the output .ps or .pdf file look most like a P1 or P2 output it is best to choose a display window aspect ratio of 1.2941.

As with .sgf files, the plots will not have a tight boundingbox. For .sgf files the script `_${SACHOME}/bin/sgftoeps.csh` will produce an EPS file with a tight boundingbox if the program Gh-sostscript (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. See `_${SACHOME}/aux/macros/demo` for scripts with examples.

EXAMPLES

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
```

The plots at the end of the [CONVOLVE](#), [CORRELATE](#), and [TUTORIAL](#) help file have an aspect ratio of 3.78 (because the larger aspect ratio takes less vertical space, which is useful if the plot is to be included in a document). The following commands can be used to produce such a plot file:

```
SAC> fg seismo
SAC> window 3 x .04 .94 y .48 .93 aspect 3.78
SAC> bw 3
SAC> p2
SAC> save seismo.xpm
```

LATEST REVISION

version 102.0

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, YMAX-
IMUM

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 preceding the name of the variable with a percent sign ("%"). If you want to concatenate 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:

```
SAC> SETBB C1 2.45 C2 4.94
```

To later use these variables in a command:

```
SAC> BANDPASS CORNERS %C1 %C2
```

To set a blackboard variable that contains spaces:

```
SAC> SETBB MYTITLE 'Sample filter response'
```

To check and make sure the value is correct:

```
SAC> GETBB MYTITLE
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:

```
SAC> TITLE '%MYTITLE%'
```

See the section on Macros in the SAC Users Manual for more examples of the use of blackboard variables in macros.

OUTPUT_FORMAT

Prior to SAC v102.0, floating-point numbers in [message](#), [listhdr](#), and [evaluate](#) displayed limited precision. OUTPUT_FORMAT, introduced in v102.0, allows one to see values with higher precision. The command syntax is `setbb output_format xxx`, where xxx can be default (%g), short (%.4f), long (%.15f), shortG (%.5g), longG (%.15g), shortE (%.4e), longE(%.15e), and off (reverts to old SAC format). For examples showing the utility of OUTPUT_FORMAT, see the section on *Floating-Point Precision in SAC* in [TUTORIAL](#).

SEE COMMANDS

[GETBB](#), [EVALUATE](#), [UNSETBB](#), [LISTHDR](#), [MESSAGE](#)

LATEST REVISION

Version 102,0

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 than 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:

```
SAC> SGF DIRECTORY /MYDIR/SGFSTORE FRAME 0
```

To set the x viewport plot size to 3 inches (i.e., wallet size):

```
SAC> SGF SIZE FIXED 3.0
```

For create a poster size plot to put on your wall:

```
SAC> SGF SIZE FIXED 30.0
```

To set the x viewport plot size to be 1 inch long for every 10 seconds of seismic data:

```
SAC> 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, YMAX-
IMUM

LATEST REVISION

May 26, 1995 (Version 00.31)

SORT

SUMMARY

Sorts files in memory by header fields.

SYNTAX

```
SORT header {ASCEND|DESCEND} {header {ASCEND|DESCEND} ... }
```

INPUT

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

- 1379: No [SORT](#) parameters given
- 1380: Too many [SORT](#) parameters:
- 1381: Not a valid [SORT](#) parameter:
- 1383: [SORT](#) failed

WARNING MESSAGES

- 1384

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
SQRT | NLOG | LOG10 | NOSCALING
YMIN v
YMAX v
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.

{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, YMAX-
IMUM

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

Note: SQRT is also an [INLINE](#) function.

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)

STATION

SUMMARY

Search for station by region, identifier, channel and date

SYNTAX

```
STATION {time start end}
  {network ne,tw,or,ks}
  {station stat,ions}
  {location lo,ca,ti,on,00,10,--}
  {channel cha,nne,els,BH?,HHZ,LHE}
  {outfile filename}
  {origin lon lat}
  {id:eventid}
  {radius min_radius max_radius}
  {ph5}
  {region w e s n}
  {level station|channel}
  {epochs on|off}
  {showtimes}
  {verbose}
```

INPUT

Time: Set the start and end times. End times can be set relatively. See [DATA ACCESS](#) for time formats.

Region west east south north: Set the geographic bounds of the search region

Radial min max: Set the radius limits of the search region. See [ORIGIN](#) and `id:eventid`

Origin lon lat: Set the origin (Lon, Lat) for a radial search

catalog:eventid: Set the origin and time from an event's origin time and location. This can be used with a region search.

Network: Set a collection of networks. Accepts lists and wildcards.

Station: Set a collection of stations. Accepts lists and wildcards.

Location: Set a collection of locations. Accepts lists and wildcards.

Channel: Set a collection of channels. Accepts lists and wildcards.

Level: Station or Channel, Station level metadata only includes Station location, Channel level metadata includes component orientation and can be used as input to the [METADATA](#) command

ph5: Get additional data from the IRIS PH5 Web service. This option provides access to active-source data sets and their associated stations with high sample rates and time-limited deployments.

Epochs:

- Off - Combine all stations with the same Network and Station, using the first value encountered

- On - Treat individual time epochs for stations separately, This will result in multiple stations with the same Network and Station, but with different epochs. This option changes how txt/csv output is written, not XML.

ShowTimes: Show start and end date-times in the output.

Outfile: Save the station search into a file. Files ending in .xml are saved as FDSNStationXML documents; those ending in .txt and others are saved as "csv" text. The Epochs option will modify how the output appears.

Verbose: Watch the details of the search process

DEFAULT VALUES

```
station epochs off
```

This command gets everything; you have been warned.

DESCRIPTION

Search a collection of stations meeting a set of criteria.

Searching can be done geographically using a Region or Radial search. For the Radial search the origin location (Lon, Lat) needs to be specified using either the Origin option or from an id:eventid. See [EVENT](#) for a discussion of eventid.

Setting the Time for a search can be done using two times, start and end, or using start and a relative offset.

If you want a particular Network or set of Stations, those can be entered directly in list form separated by commas ',' using wildcards and negation, see examples below. Channels are supported just as networks and stations.

Locations are the same as well, but unless you have specific knowledge it is best to ignore the Location option, see examples below.

Specifying id:eventid will set the time of the search and the origin location for a radial search. Timing and origin location information is obtained from online catalogs. Non-existent eventid will return an error. This can be useful for getting stations lists for specific stations.

EXAMPLES

The station database has intriguing station data. Lets see if there are stations on the Moon. The Moon Landing was in 1969 and the network code is XA, one of many temporary networks:

```
SAC> station net XA time 1969-1 +4y
Net Sta  Lat.    Lon.    Elev.  SiteName
XA S12   -3.0400 -23.4200 0.00  ALSEP 12, Oceanus Procellarum, Moon
XA S14   -3.6500 -17.4800 0.00  ALSEP 14, Fra Mauro, Moon
XA S15   26.0800  3.6600  0.00  ALSEP 15, Hadley Rille, Moon
XA S16   -8.9700 15.5100  0.00  ALSEP 16, Descartes, Moon
```

We would like stations close to the 2004 Sumatran M9 Event. First lets get the event id:

```
SAC> event time 2004-1 +1yr mag 9
Origin          Lat.    Lon.    Depth  Mag.    Agency EventID
2004-12-26T00:58:53  3.29   95.98  30.00  9.10  mw    US/official usgs:official2004122600
```

Using that eventid we can search for stations nearby:

```
SAC> station usgs:official20041226005853450_30 radial 0 5
Net Sta   Lat.     Lon.     Elev.   SiteName
PS  PSI     2.6938  98.9237  987.00 Parapat, Indonesia
SY  GSI     1.3039  97.5755  107.00 GSI synthetic
SY  LHMI    5.2288  96.9472   3.00 LHMI synthetic
SY  PSI     2.6938  98.9237  987.00 PSI synthetic
```

There are stations that are only contain synthetics, Network SY. Lets get rid of those:

```
SAC> station usgs:official20041226005853450_30 radial 0 5 net -SY
Net Sta   Lat.     Lon.     Elev.   SiteName
PS  PSI     2.6938  98.9237  987.00 Parapat, Indonesia
```

Epochs can be confusing as stations are serviced and instruments are changed, new epochs or time ranges are added. For example IU.ANMO has been operating since the mid-1990s and has multiple time periods:

```
SAC> station net IU sta ANMO epochs on
Net Sta   Lat.     Lon.     Elev.   SiteName
IU  ANMO   34.9459 -106.4572 1850.00 Albuquerque, New Mexico, USA
IU  ANMO   34.9459 -106.4572 1850.00 Albuquerque, New Mexico, USA
IU  ANMO   34.9502 -106.4602 1839.00 Albuquerque, New Mexico, USA
IU  ANMO   34.9459 -106.4572 1820.00 Albuquerque, New Mexico, USA
IU  ANMO   34.9459 -106.4572 1820.00 Albuquerque, New Mexico, USA
IU  ANMO   34.9459 -106.4572 1820.00 Albuquerque, New Mexico, USA
```

```
SAC> station net IU sta ANMO epochs off
Warning: Multiple instances of net.sta, likely mutiple epochs
Net Sta   Lat.     Lon.     Elev.   SiteName
IU  ANMO   34.9459 -106.4572 1850.00 Albuquerque, New Mexico, USA
```

```
SAC> station net IU sta ANMO epochs on showtimes
Net Sta   Lat.     Lon.     Elev.   TimeOn           TimeOff
IU  ANMO   34.9459 -106.4572 1850.00 1989-08-29T00:00:00 1995-07-14T00:00:00
IU  ANMO   34.9459 -106.4572 1850.00 1995-07-14T00:00:00 2000-10-19T16:00:00
IU  ANMO   34.9502 -106.4602 1839.00 2000-10-19T16:00:00 2002-11-19T21:07:00
IU  ANMO   34.9459 -106.4572 1820.00 2002-11-19T21:07:00 2008-06-30T00:00:00
IU  ANMO   34.9459 -106.4572 1820.00 2008-06-30T00:00:00 2008-06-30T20:00:00
IU  ANMO   34.9459 -106.4572 1820.00 2008-06-30T20:00:00 2599-12-31T23:59:59
```

(To fit on the page, the last column *SiteName* is left out. For all of them it is *Albuquerque, New Mexico, USA*.)

PH5 Example: Active-source stations are typically not available through the IRIS Web Service, but can be accessed through the PH5 Web Services. Station 9A.22770 is one of these stations. To access its information, you need to specify the PH5 option to tell SAC to also search within PH5:

```
SAC> station net 9A station 22770
Error 404 (HTTP): No Content
```

```
SAC> station net 9A station 22770 PH5
Net Sta   Lat.     Lon.     Elev.   SiteName
9A  22770  44.4122 -115.4911 1966.30 Deformation and magmatic modification ...
```

ERROR MESSAGES

SEE COMMANDS

[EVENT](#), [DATA](#), [RESPONSE](#), [METADATA](#), [DATA ACCESS](#)

LATEST REVISION

Version 102.0

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:

```
SAC> READ F1 F2 F3
SAC> SUB 5.1 6.2
```

Note: SUBTRACT is also an [INLINE](#) function.

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:

```
SAC> READ FILE1 FILE2 FILE3
SAC> SUBF FILE4
```

To subtract two files from two other files:

```
SAC> READ FILE1 FILE2
SAC> 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:

```
SAC> LINE OFF
SAC> SYMBOL 5
SAC> PLOT
```

To annotate four solid lines on a [PLOT2](#) plot using symbols 7, 4, 6, and 8, and a spacing of 0.3:

```
SAC> LINE SOLID
SAC> SYM SPACING .3 INCREMENT LIST 7 4 6 8
SAC> R FILE1 FILE2 FILE3 FILE4
SAC> 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:

```
SAC> READ FILE1 FILE2 FILE3
SAC> LINE LIST 1 0 0 INCREMENT
SAC> SYMBOL LIST 0 3 7 INCREMENT
SAC> PLOT2
```

SEE COMMANDS

[LINE](#)

LATEST REVISION

October 11, 1984 (Version 9.1) Summary of [SYMBOL](#) Command Attributes

Symbol Numbers:

1	•	9	✱
2	□	10	⊞
3	△	11	⊕
4	○	12	⊙
5	◇	13	⊠
6	▽	14	⊡
7	+	15	⊢
8	×	16	⊣

Default Values:

SYMBOL OFF SIZE 0.01 SPACING 0.

INCREMENT OFF LIST STANDARD

Symbol Sizes:

0.005	◻	0.020	◻
0.010	◻	0.025	◻
0.015	◻	0.030	◻

Symbol Spacings:

0.050	◻	◻	◻	◻	◻	◻	◻	◻	◻	◻	◻	◻	◻	◻
0.100	◻		◻		◻		◻		◻		◻		◻	
0.200	◻			◻			◻			◻			◻	
0.300	◻				◻					◻				◻

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 options 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:

```
SAC> READ FILE1 FILE2
SAC> LISTHDR B KZTIME KZDATE

FILE: FILE1
-----
      B = 0.
      KZTIME = 10:38:14.000
      KZDATE = MAR 29 (088), 1981

FILE: FILE2
-----
```

```
B = 10.00
KZTIME = 10:40:10.000
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:

```
SAC> SYNCHRONIZE
SAC> LISTHDR
```

```
FILE: FILE1
-----
      B = -126.00
KZTIME = 10:40:20.000
KZDATE = MAR 29 (088), 1981
```

```
FILE: FILE2
-----
      B = 0.
KZTIME = 10:40:20.000
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:

```
SAC> SYSTEMCOMMAND LS
... 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 COSINE option here is computed using the $\sin()$ function as:

$$\text{DATA}(J) = \text{DATA}(J) * \text{SIN}(\text{OMEGA}*J) \text{ where } \text{OMEGA} = \text{PI}/(2*N)$$

where N is the length of the taper on each end, and J runs from 1 to N.

The general form for the Hanning or Hamming taper is:

$$\text{DATA}(J) = \text{DATA}(J) * (F0 - F1 * \text{COS}(\text{OMEGA} * (J)))$$

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.

TYPE	OMEGA	F0	F1
HANNING	PI/N	0.50	0.50
HAMMING	PI/N	0.54	0.46

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)

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:

```
SAC> TICKS ON TOP
```

To turn off all tick marks (at least where there are no annotated axes):

```
SAC> TICKS OFF ALL
```

To turn tick marks on for the bottom side and off for the rest:

```
SAC> 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 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:

```
SAC> TRACE ON TEMP1 MYFILE,T0
TRACE (on) TEMP1 = 1.45623
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:

```
TRACE (mod) TEMP1 = 2.34293
TRACE (mod) MYFILE,T0 = 10.3451
```

Later in the processing TEMP1 may change again:

```
TRACE (mod) TEMP1 = 1.93242
```

When the tracing is turned off, SAC will print the current value one last time:

```
SAC> TRACE OFF TEMP1 MYFILE,T0  
TRACE (off) TEMP1 = 1.93242  
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:

```
SAC> 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:

```
SAC> 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:

```
SAC> 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:

```
SAC> TRANSCRIPT OPEN FILE ERRORTRAN CONTENTS ERRORS  
SAC> 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 current and future runs, use:

```
SAC> 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

INTRODUCTION

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: Instrument deconvolution using spectral division, EVALRESP, POLEZERO, or FAPfile
EVALRESP, POLEZERO, and FAPfile are described below.

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.

FREQLIMIS: Default is it is not used. See discussion below

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.

DEFAULT VALUES: TRANS FROM NONE TO NONE PREWHITENING OFF

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) - SAC's convention for displacement. If TRANSFER uses 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 (example below).

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. The evolution of data handling by IRIS is described in [DATA ACCESS](#).

The EVALRESP program calculates the complete system response from response (RESP) files produced by commands given in [RESPONSE](#) or programs such as RDSEED. The code used in SAC does

not recognize all the options available in the current version of EVALRESP (v5.0, August 2019). If one wants an option only available in the full program, one can first run EVALRESP with a FAP output option and use the FAP option (see below) to correct the SAC data file. The source code for program EVALRESP can be downloaded from URL <<http://ds.iris.edu/ds/nodes/dmc/software/downloads/>>.

The SAC sign convention for Laplace/Fourier transforms is the same as that used in SEED and EVALRESP: phase for a causal response decreases with increasing frequency. For displacements, the SAC convention is nm, while RESP files use meters. **NOTE:** The EVALRESP option in TRANSFER converts the output to the SAC convention. For other options (FAP, PZ) it may be necessary to manually 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 suggested rule-of-thumb is $f1 \leq f2/2$ and $f4 \geq 2*f3$.

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 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.

FROM EVALRESP

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 a RESP file. 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, please refer to 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 (KHOLE). Outside of SAC, 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 **CHNHDR**. 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/LOCID, 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 internally by a factor of 1.0e9 to nanometers, the SAC convention for displacement units.

EVALRESP EXAMPLE

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 deconvolution TO NONE because the instrument has zero response at zero frequency.

FROM POLEZERO

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:

$$H(s) = \frac{(s-z_1) * (s-z_2) * \dots * (s-z_{nz})}{(s-p_1) * (s-p_2) * \dots * (s-p_{np})}$$

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 IRIS DMC's SAC PZ web service <<http://service.iris.edu/irisws/sacpz/>> or the SEED manual for how it is defined.) 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), so the number of zeros lines can be less than nz . One may specify up to 30 poles and 30 zeros.

The original SAC polezero files only contained poles, zeroes, and a constant. About ten years ago it was decided that supplying formatted comments as a header in the polezero file helps users organize and understand the origins of the coefficients presented. For this reason, since 2011 SAC supports the *annotated* polezero file, produced by RDSEED (starting with v5.2, October 2011) or available for the IRIS SACPZ archive. Depending on the request format, a polezero file returned by sacpz may include multiple 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 annotation values that match the header values.

Options for specifying the NETWORK, STATION, LOCID, CHANNEL, DATE, TIME are available for the POLEZERO option as they are in the EVALRESP option.

The polezero file below was written by program RDSEED (v5.2):

```
* *****
* 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 six 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).

The INPUT UNIT for any polezero produced by sacpz or evalresp will be "M", so seismometers and accelerometers perform a simple conversion to address integration. An accelerometer will add two "zeros" to the beginning of a channel's zero response to step down to displacement. For velocity, the program just adds one "zero". The number of zeros in stage 0 in the RESP file may differ from the number in the polezero file because INPUT UNIT in the polezero is fixed while "Response in units" in the RESP file is not.

Note that when one compares the SAC waveform file to the polezero file, the SAC header value COMPINC differs by 90 degrees from the polezero DIP: COMPINC is degrees from the upward vertical, and DIP is the downward positive angle from horizontal. The polezero AZIMUTH uses the same convention as the SAC header CMPAZ.

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.

POLEZERO SEARCH

If a SUBTYPE with FILENAME is not set, SAC will search the current directory for SAC Polezero files named like:

```
SAC_PZs_NETWORK_STATION_CHANNEL_LOCID_*
```


where NETWORK, STATION, CHANNEL, LOCID are either those defined in the SAC Header or specified on the command like, e.g.:

```
TRANSFER from POLEZERO NETWORK BK STATION CMB to NONE
```

Matching files will be searched for the correct response using the associated metadata. The first file found with the correct metadata will be used for the response. Files with no associated metadata are assumed to be the correct response.

Use the DIR option to specify an alternative search directory, for example:

```
TRANSER from POLEZERO DIR resp to NONE
```

will search for responses like:

```
resp/SAC_PZs_NET_STAT_CHAN_LOC_*
```

An simple example of the automatic polezero search option would be to remove the responses for all files in memory, assuming the responses are in the current directory

```
SAC> READ *.XT*.BHZ.SAC
  XT.ATAT.01.BHZ.M.2001.255.084837.SAC
  XT.DAWA.01.BHZ.M.2001.255.084837.SAC
SAC>
SAC> rtrend
SAC> transfer from POLEZERO to NONE  FREQLIMITS (1/120) (1/60) 6 7
  Using polezero response for ATAT, BHZ, XT, 01 from SAC_PZs_XT_ATAT_BHZ_01_
  Station (ATAT  ), Channel (BHZ  )
  Using polezero response for DAWA, BHZ, XT, 01 from SAC_PZs_XT_DAWA_BHZ_01_
  Station (DAWA  ), Channel (BHZ  )
```

If the responses are instead in a directory named *responses*, a *DIR responses* should be added to the FROM part section, e.g.:

```
SAC> transfer from POLEZERO DIR responses to NONE  FREQLIMITS (1/120) (1/60) 6 7
  Using polezero response for ATAT, BHZ, XT, 01 from responses/SAC_PZs_XT_ATAT_BHZ_01_
  Station (ATAT  ), Channel (BHZ  )
  Using polezero response for DAWA, BHZ, XT, 01 from responses/SAC_PZs_XT_DAWA_BHZ_01_
  Station (DAWA  ), Channel (BHZ  )
```

POLEZERO EXAMPLES

The PZ file SAC_PZs_XC_OR075_LHZ is the correct one to remove the instrument response from waveform OR075_LHZ.SAC:

```
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
```

The MUL 1.0e9 command converts the displacement output from the POLEZERO meters to the SAC default of nanometers.

For 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 made a call to sacpz or concatenated 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

```

FROM FAPfile

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 than 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 FAPfile is consistent with that produced by the standalone program EVALRESP, but is different from the format used by (pre-2005) SAC2000.

FAPfile 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 from 0.006 HZ to 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.). As discussed above, if FAP comes from EVALRESP, the units need to be converted from SI units (meters for displacement) to the SAC convention.

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
LRSMS	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

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

FREPERIOD 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.

HEADER CHANGES

- idep
- depmin, depmax, depmen

LATEST REVISION

Version 102.0

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

Instrument	Subtype
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

Instrument	Options
ELMAG	<ul style="list-style-type: none"> • FREEPERIOD v where v = one of 15.0,30.0 • MAGNIFICATION n where n = one of 375,750,1500,3000,6000
GENERAL	<ul style="list-style-type: none"> • FREEPERIOD v • DAMPING v
LLL	<ul style="list-style-type: none"> • SUBTYPE BB • DAMPING v • FREEPERIOD v
REFTEK	<ul style="list-style-type: none"> • FREEPERIOD v • DAMPING v • CORNER v • HIGHPASS v

TRAVELTIME

SUMMARY

Computes traveltimes of selected phases for pre-defined velocity models. This routine has the same source code as SSS/TRAVELTIME. If one is in SSS, one can use different plotting routines. For an example, see [SSSTRAVELTIME](#).

SYNTAX

```
TRAVELTIME {MODEL string} {ONLINE} {VERBOSE | QUIET} {PICKS number}
           {PHASE phase list} {CLEAR} {ONRECORD ON|OFF} {M | KM}
```

INPUT

MODEL: *iasp91* [default], *ak135*. Traveltimes are calculated using the *iaspei-tau* procedures and traveltime tables. (For further informati, see documentation referenced below.)

ONLINE: Retrieve traveltimes from the IRIS Traveltime Web Service. This uses Crotwell's TauP Toolkit internally. In addition to *iasp91* and *ak135*, it includes model *prem*. Also, it includes additional phases, and some phase groupings are not the same as for the *iaspei-tau* tables used by SAC. (For further informati, see documentation referenced below.)

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 the first-point time (B). If instead, QUIET is among the options, nothing is displayed. VERBOSE is the default.

PICKS: There are 10 time picks in the SAC header: t0 to t9. If the number is n ($0 \leq n \leq 9$), the first phase will be at Tn. The default is n=0. If PICKS is included among the command-line options and if phases are within the taim range of and seismogram in memory, the picks will be included in the header for that seismogram and will be displayed in subsequent plots such as [PLOT1](#) and [PLOTPK](#). The default is not to have PICKS on, but if PICKS is turned on, it will remain on for subsequent runs with the same data set. Traveltimes are calculated relative to the origin (O) but the displayed and stored Tn times are relative to (B).

PHASE: List of phases for which times are displayed (unless QUIET is turned on). Phase names are generally case sensitive. As of SAC v102.0, *verbose ... phase* displays phases in chronological order, and the sorting is done before comparison with PICKS list. If *PICKS n* is among the command-line options, the phase arrival times and their times and labels will be added to the header starting at Tn. If n is 0, one can have up to 10 phases put in the header and available for subsequent plots. If n is 8, one can have at most 2 new phases added to the header.

ONRECORD ON|OFF: If the command line has ONRECORD ON, phases are displayed (VERBOSE) and/or included as a PICK if the arrival time is between (B) and (E) for each station. ONRECORD OFF is the Default

CLEAR: Clears the phase list from previous runs. An empty phase list will be populated with the DEFAULT phases (see below) or values taken from the environment variable SAC_TRAVELTIME_PHASES. (See below.) In the command line, CLEAN must come before PHASE

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. M was

the default in SAC until 2005. For all runs of *TRAVELTIME VERBOSE*, the depth is printed to the screen so it is obvious whether meters or kilometers were used

DEFAULT VALUES: MODEL iasp91 ONRECORD OFF KM PHASE P S Pn Pg Sn Sg

DESCRIPTION

All waveforms in memory must have event and station locations defined as well as the event depth and the origin time.

The traveltimes tables use the stored degree-distance measure (GCARC), which is calculated from the event and station latitudes and longitudes using spherical-triangle geometry after converting geographic latitudes to geocentric.

A quick way to find out all the possible phases that could be visible on a seismogram according to *TRAVELTIME*, use *phase all*. Eg.:

```
SAC> read my-seismogram
SAC> traveltimes verbose phase all
SAC> traveltimes verbose onrecord on clear phase all
```

The second traveltimes command displays only arrivals that are between (B) and (E) for *my-seismogram*. The *clear* option removes phase data from memory for the previous call .

As discussed in [DATAGEN](#), seismograms from a deep-focus event have been added: `/${SACHOME}/aux/datagen/deep/`. Arrivals are sharp, and several converted phases can be seen. We use that event to demonstrate how to use *TRAVELTIME*:

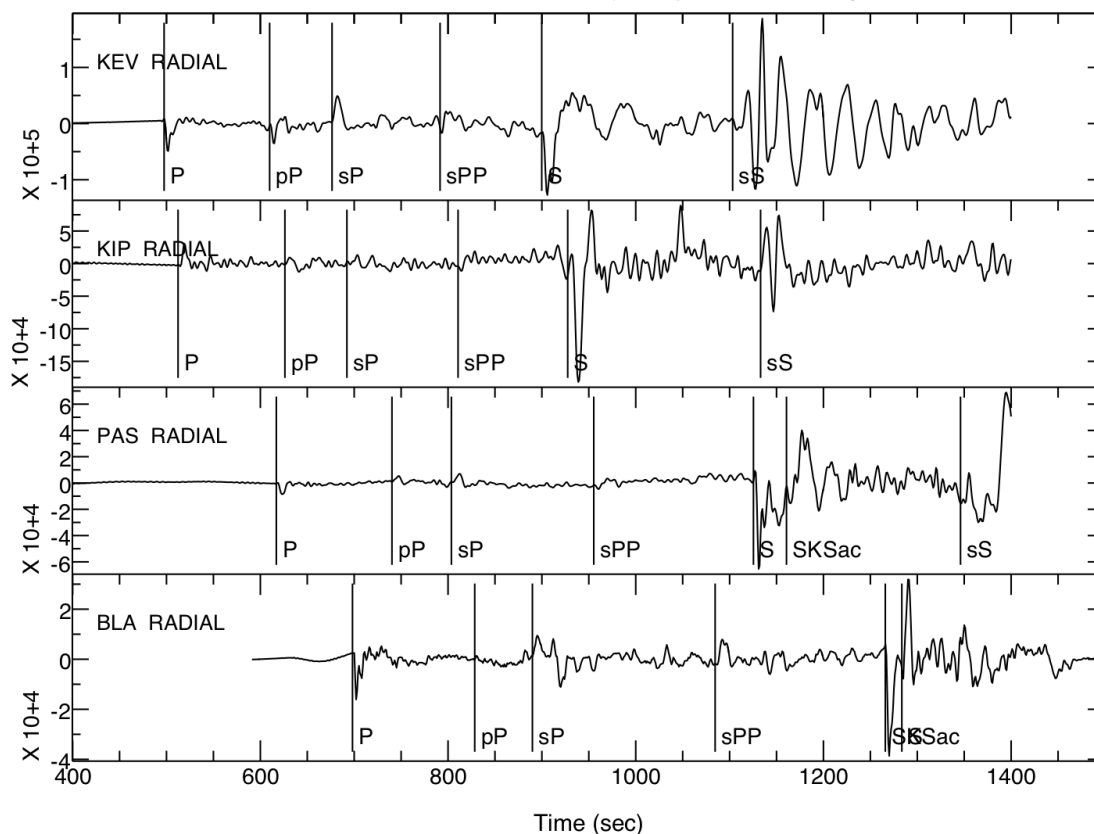
```
SAC> datagen sub deep bla.r
SAC> lh b e o gcarc evdp
      b = 5.911000e+02                e = 1.499100e+03
      o = 0.000000e+00                gcarc = 8.627435e+01
      evdp = 6.110000e+02
SAC> traveltimes verbose picks 0 onrecord on phase all
      traveltimes: depth: 611.000 km
      ...
      traveltimes: setting phase SKSac    at 1265.9326 s [ t = 1265.9326 s ] t9
      ...
SAC> traveltimes picks 9 clear phase SKS
      traveltimes: depth: 611.000 km
      traveltimes: error finding phase SKSdf
      traveltimes: setting phase SKSac    at 1265.9326 s [ t = 1265.9326 s ] t9
```

In the traveltimes call, arrival SKSac was the 10th arrival so would be the last stored pick. If *clear* had not been included in the second command, the *VERBOSE* output would have been incorrect. (Try it.) Prior to SAC v102.0, *phase SKS* would have returned *error finding phase SKS*.

Next are the commands and an output plot for a run using the radial component for four stations and six phases. Because plot command [PLOT1](#) plots traces from top-to-bottom in the order read in, we manually re-ordered the files so that distance increases from first to last. Phase sPP is not in the *iasp-tau* tables but is in the *ONLINE tau-p* tables. (A nice feature for *SAC TRAVELTIME* is the ability to combine phases from multiple traveltimes table in a single plot:

```
SAC> datagen sub deep kev.r kip.r pas.r bla.r
SAC> rtr
SAC> traveltimes quiet online picks 0 phase sPP
SAC> traveltimes picks 1 phase P pP sP S sS SKS
SAC> FILEID LOCATION UL TYPE LIST KSTCMP
SAC> TITLE 'Sakhalin Island Event: &l,KZDATE&, Depth is &l,EVDP& KM'
```

Sakhalin Island Event: MAY 12 (132), 1990, Depth is 611 KM



```
SAC> p1
SAC> save sakhalin-picks_p1.pdf
```

In [SSSTRAVELTIME](#) a plot is created for the same seismograms and phases using [PLOTRECORD-SECTION](#).

DEFAULT PHASES

The default phases for TRAVELTIME are set as local crustal phases: "P S Pn Pg Sn Sg". If a different set of default phases is desired the environment variable SAC_TRAVELTIME_PHASES can be set to override the default. The variable is comma or space delimited. For teleseismic data, one might use "P pP sP S pS sS PP SS". An alternative is to use "all" to capture as many phases as possible.

Note: Environmental variables must be set outside of SAC. In the *sh* shell, for *all*, enter `export SAC_TRAVELTIME_PHASES="all"`. For shell *csh*: `setenv SAC_TRAVELTIME_PHASES "all"`.

REFERENCES

laspei-tau: <https://seiscode.iris.washington.edu/projects/iaspei-tau/>

TauP Toolkit: <http://www.seis.sc.edu/downloads/TauP/taup.pdf> Section 4: Phase naming in TauP

IRIS DMC IRISWS travelttime Web Service Documentation <https://service.iris.edu/irisws/traveltime/1/>

LATEST REVISION

Version 2.0

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 [PLOT](#) 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 viewport.
- B Number of lines of text in full viewport.
- **C Number of lines of text in a normal viewport.** Normal means 0. to 1. in x and 0. to 0.75 in y.
- D Minimum number of characters per normal viewport line.
- **E Average number of characters per normal viewport 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:

```
SAC> TSIZE MEDIUM 0.35
SAC> TITLE 'Rayleigh Wave Spectra' SIZE MEDIUM
SAC> PLOT2
```

To reset this (and any other) size definitions to their default values:

```
SAC> TSIZE NEW
```

SEE COMMANDS

[TITLE](#), [XLABEL](#), [FILEID](#), [PLOT2](#)

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:

```
SAC> UNSETBB C1 C2 X
```

To unset all blackboard variables:

```
SAC> 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 2π 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.

REFERENCES

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 [PLOT](#) 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:

```
SAC> READ FILE1 FILE2 FILE3 FILE4
SAC> PLOT
    waiting           {plot of FILE1 to terminal}
SAC> (return)
SAC> waiting         {plot of FILE2}
SAC> kill            {user has seen enough}
    (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 resultant 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.

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:

```
SAC> WIDTH 1 INCREMENT
```

To set up an incrementing data width list of 1, 3, and 5 with an skeleton of 2:

```
SAC> 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 $\mu = 1.95 / \text{Rho}(0)$. Where $\text{Rho}(0)$ 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 $\text{epsilon} == 0$ 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:

```
SAC> 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:

```
SAC> 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 duplicate 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.:

```
SAC> READ S*
  STA01E STA01N STA01Z STA02E STA02N STA02Z STA03Z
SAC> READ *Z
  STA01Z STA02Z STA03Z
SAC> READ ???
  ABC DEF
SAC> READ STA01[Z,N,E]
  STA01Z STA01N STA01E
SAC> READ *[Z,N,E]
  STA01Z STA02Z STA03Z STA01N STA02N STA01E STA02E
SAC> READ *1[Z,N,E] *2[ ]
  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, size, and aspect ratio of graphics windows.

SYNTAX

```
WINDOW n {XSIZE xmin xmax} {YSIZE ymin ymax}  
        {ASPECT [ value | ON | OFF ]}
```

INPUT

n: The graphics window number of interest. There are a total of nine possible graphics windows [1 - 9].

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. $xsize = xwmax - xwmin$.

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. $ysize = ywmax - ywmin$.

ASPECT VALUE: If ASPECT is ON (OFF is 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 not explicitly given. The default for VALUE is $11.0/8.5 = 1.294$.

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

The default is $n = 1$.

For versions 101.5 through 101.6a, the default for ASPECT was ON, so *xwmax* was not used in the default windows. If you prefer the 2011 - 2018 windows, use a `init.m` macro (see `${SACHOME}/README`) with the following lines:

```
window 1 x 0.05 0.65 y 0.45 0.95 aspect on  
window 2 x 0.07 0.67 y 0.43 0.93 aspect on  
window 3 x 0.09 0.69 y 0.41 0.91 aspect on  
window 4 x 0.11 0.71 y 0.39 0.89 aspect on  
window 5 x 0.13 0.73 y 0.37 0.87 aspect on
```

DESCRIPTION

The default window set is useful if one wants to create and compare several plots in a single SAC session. For an example about how to produce a screen with several separate but related plots, see ``${SACHOME}`/aux/macros/demo`.

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, the moving of a window will result in the current plot being redraw 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.

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 or 16:9.

`SAVEIMG` has instructions how to create plot files from displayed windows, but for PDF or PS plots, the aspect ratio is fixed to 11.0/8.5 (1.294). To get the best resolution for such plots, it is best to have the window aspect ration at or near 1.294.

For screens with large aspect ratios, width/height > 2.0 or < 0.5, e.g. multiple monitors and high-end displays and if `ASPECT` is `OFF`, the `ASPECT` ratio will automatically be turned `ON` and set to $11/8.5 = 1.294$. This behavior attempts to avoid ultra-wide display windows.

Prior to 2011 (when `ASPECT` was introduced), the most recent revision to `WINDOW` was in 1986, and computer monitors have evolved considerably since then.

EXAMPLES

To see the current window values:

```
SAC> report window
```

To change from the default (`n = 1`) to give 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.

Note that the order matters:

```
SAC> window 1 ASPECT 1.33 x 0.25 0.85
SAC> bw 1
```

will not give the same answer as

```
SAC> window 1 x 0.25 0.85 ASPECT (4.0 / 3.0)
SAC> bw 1
```

which ignores `xwmax` and fixes the aspect ratio to $4/3 = 1.33$ because later items on the command line take precedence

LATEST REVISION

2019 (v 102.0)

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** precede any element in the namingoptions:

```
OVER  
APPEND text  
PREPEND text  
DELETE text  
CHANGE text1 text2  
filelist
```

Only one of these naming options 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 SEG Y format.

This format allows one waveform per file. **Note** only evenly-spaced, time-series files will be written in SEG Y.

Since SAC stores the waveform as a series of real*4 numbers, and SEG Y stores the waveform as a series of long integers, the datapoints from SAC are normalized to the maximum allowable integer. The scale field in SEG Y 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 SEG Y](#) command.

The following SAC header fields are saved as the following SEG Y header fields

SAC	SEG Y
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
DEPMAX	max, corrected by SEGYP's scale.
DEPMIN	min, corrected by SEGYP'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 SEGYP 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 track 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:

```
SAC> READ D1 D2 D3
SAC> LOWPASS BUTTER NPOLES 4
SAC> WRITE F1 F2 F3
```

This could have also been done using the CHANGE option:

```
SAC> READ D1 D2 D3
SAC> LOWPASS BUTTER NPOLES 4
SAC> 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:

```
SAC> 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} {DIR name} filename
```

INPUT

TYPE: Determines whether 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.

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
```

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

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](#)

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** precede any element in the naming options:

```
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:

```
SAC> READ FILE1
SAC> FFT AMPH
SAC> WRITESP OVER
```

SAC will then write out two files, FILE1.AM and FILE1.PH. Now you perform the operations on the amplitude file:

```
SAC> READ FILE1.AM
SAC> ...perform operations.
SAC> 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:

```
SAC> READSP FILE1
SAC> IFFT
SAC> 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 similar 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 logarithmic 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 xmin xmax
```

INPUT

xvmin: X axis viewport minimum value. Must be in the range 0.0 to xmax.

xvmax: X axis viewport maximum. Must be in the range xmin 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](#)

REFERENCES

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 similar 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:

```
SAC> YLIM 0.0 30.0 ALL OFF
SAC> READ FILE1 FILE2 FILE3
SAC> 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 logarithmic 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](#)

REFERENCES

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
```

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
```

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:

```
SAC>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:

```
SAC>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 traveltimes called from within SSS. Computes traveltimes curves for pre-defined 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	PLOT	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 beginning 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 amount 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 consistent 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:

```
SAC> GLOBALSTACK DELAY 1.0 INCREMENT 0.03
SAC> ADDSTACK FILEA DELAY 2.0
SAC> ADDSTACK FILEB DELAY 3.0 INCREMENT 0.01 REVERSED
SAC> ADDSTACK FILEC
SAC> 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:

```
SAC> 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:

```
SAC> INCREMENTSTACK
SAC> CHANGESTACK FILEC NORMAL
SAC> 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:

```
SAC> DELETSTACK FILED
SAC> INCREMENTSTACK
SAC> 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](#), [DELETSTACK](#)

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  
[D]ISTANCE v  
[B]EGINTIME v  
[E]NDTIME v  
[D]ELAY 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.

[D]ISTANCE v: Station to epicenter distance in kilometers for this file. This is used to calculate dynamic time delays.

[B]EGINTIME v: Time of the beginning of the event.

[E]NDTIME v: Time of the end of the event.

[D]ELAY 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 amount 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.

ROUND OFF: 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 ROUND OFF
```

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:

```
SAC> 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  
[D]ISTANCE v  
[D]ELAY 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.

[D]ISTANCE v: Global station to epicenter distance in kilometers.

[D]ELAY v [[S]ECONDS|[P]OINTS]: 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::

```
SAC> ADDSTACK FILEA
SAC> ADDSTACK FILEB
SAC> ADDSTACK FILEC
SAC> ADDSTACK FILED
SAC> VELOCITYMODEL 1 REFR VAPP 7.9 VAPPI 0.1 TOVM CALC DIST 320. TVM 45.
SAC> SUMSTACK
SAC> WRITESTACK STACK1
SAC> INCREMENTSTACK
SAC> SUMSTACK
SAC> WRITESTACK STACK2
SAC> INCREMENTSTACK
SAC> SUMSTACK
SAC> 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, TOVM, 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]ABELS 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]ELOLOCITY velocity_value  
[A]SPECT ON | OFF  
[ORIE]NT [P]ORTRAIT | [L]ANDSCAPE  
[T]IME 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 traveltimes curves. See [SSSTRAVELTIME](#).

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 traveltimes curves on. Traveltimes curves must have been computed with the traveltimes 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 separate 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 section plots with different time windows and you want each 2 seconds on these plots to be 1 cm long:

```
SAC> 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

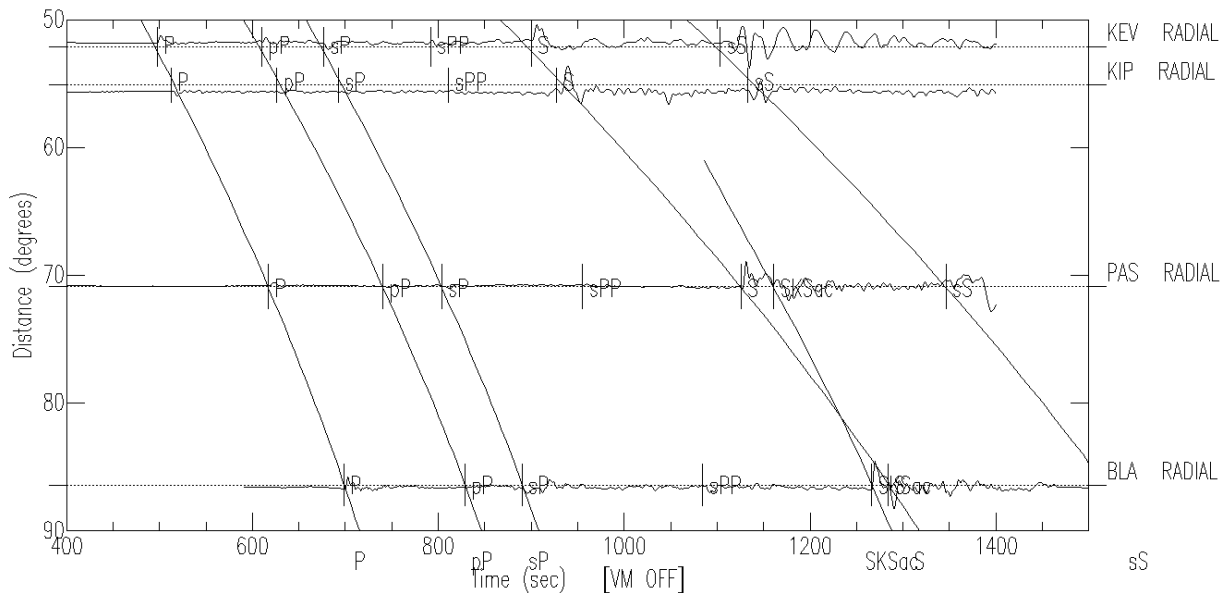
Computes traveltimes of selected phases for pre-defined velocity models. Description and examples assume the calls are made from within SSS. The source file is the same as the non-SSS version, and this file has only an example that uses SSS plotting. For the full description, see [TRAVELTIME](#).

EXAMPLE

This example is the same traveltimes run as in [TRAVELTIME](#), but here for the final plot we use [PLOTRECORDSECTION](#):

```
SAC> datagen sub deep bla.r kev.r kip.r pas.r ; rtr
SAC> SSS
SAC/SSS> traveltimes quiet picks 0 phase sPP
SAC/SSS> traveltimes quiet picks 1 phase P pP sP S sS SKS
SAC/SSS> dw units degrees fixed 50 90 ; tw 400 1500
SAC/SSS> prs orient port ttime on labels kstcmp
SAC/SSS> save sakhalin-picks-prs.xpm
```

One must put SAC files in memory before entering SSS. We use the default aspect ratio in *prs*, so if we had used [SAVEIMG](#) to create a PDF file, the output plot would have errors.



See Commands

[TRAVELTIME](#) [PLOTRECORDSECTION](#) [TIMEWINDOW](#) [DISTANCEWINDOW](#) [SAVEIMG](#)

LATEST REVISION

Version 102.0

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  
TOVM v|CALCULATE  
DVM v1 [v2]  
TVM v1 [v2]  
VAPPI v  
TOVMI 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 move-out" 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.

TOVM v: Set the time axis intercept to v.

TOVM 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 amount each time the INCREMENTSTACK command is executed.

TOVMI v: Set the time axis intercept increment to v. The time axis intercept is incremented by this amount 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:

$$\begin{aligned}TDELAY &= TVM(1) - (T0VM + DIST / VAPP) \\TDELAY &= TVM(1) - \text{SQRT} (T0VM**2 + (DIST / VAPP)**2)\end{aligned}$$

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:

```
SAC> 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
PLOTG	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

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:

```
[H]AMMING | [H]ANNING | [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 preferred, 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 preferred 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 calculated.

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 peculiarities 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.