Title
LISA USER'S TUTORIAL - RUDIMENTARY GUIDE TO LISA

Authors
Liu, J.
Matis, H.

Publication Date
1988-02-01

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

LISA User's Tutorial - Rudimentary Guide to LISA

J. Liu and H. Matis

February 1988

Prepared for the U.S. Department of Energy under Contract DE-AC03-76SF00098

LISA User's Tutorial - Rudimentary Guide to LISA

James Liu and Howard Matis
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

ABSTRACT

LISA, a software package, which at this time is available on computer nodes: DLS, BEVAX, 88VAX and CSA, can be used for both on-line and off-line data analysis. This software package, designed to use the VAX VMS operating system, provides research groups with histogramming capability, live scatterplots, event analysis, and other important functions. LISA, written in FORTRAN, utilizes VAX global memory, screen management facilities and other run-time library routines. LISA also has compatibility with the HBOOK histogramming package.

February 11, 1988

LISA User's Tutorial - Rudimentary Guide to LISA

James Liu and Howard Matis

Lawrence Berkeley Laboratory
Berkeley, CA 94720

The following is a user's guide to using the functions available in the LISA software package. A table of various LISA functions is listed for quick reference while a glossary of each function in alphabetical order describes each function in more depth. Also documented is a short troubleshooting section which is recommended reading for first time LISA users. Users, particularly experts, are encouraged to report problems with this documentation and submit additions to this document which would help other users (especially in the troubleshooting and glossary sections). The data file for this text uses troff text processing package with the ms macros and is stored in directory: local_root:[lisa.lislib]lisa.doc.

Table of Contents

COMMAND SUMMARY

Control Commands	
BEgin	*Start reading events.*
ENd	*Stop reading events.*
CLEar	*Clear spectra and CLIst.*
CLFile	*Close channel to output file.*
CLPar	*Clear PLIst.*
DMP	*Dump spectra, etc. to disk (create an out_file).*
SDMP	*Dump etc. to disk.*
INDmp	*Reads DMP or SDMP out_files into Lisa.*
HBCON	*Convert LISA DMP out_file to HBOOK out_file.*
HDMP	*Write LISA histograms into HBOOK out_file.*
HGET	*Read in HBOOK histograms.*
MODe	*Lin-log, toggle for pp, event ptr.*
MOUNT_tape	*Mount a magnetic tape for input.*
MOVE_tape	*Move a magnetic tape to a run for analysis.*
NEW	*Open input.*
SPIn	*Spectrum in.*
SPOut	*Spectrum out.*
STOPLisa	*Stop Lisa.*
SDElete	*Spectrum delete.*
WARTE	*Wait.*
Queue Histogram Commands	
DEFM1	*Counts vs. Mod. No.*
DEFM2	*Channel No. vs. Mod. No.*
DEFMM	*Mod. No. vs. Mod. No.*
DEFQ1	*Counts vs. Channel No.*
DEFQ2	*Channel No. vs. Channel No.*
QUEue	*Define queue.*
KBUF Histogram Commands	
DEF1	*Counts vs. Channel No.*
DEF12	*Set up 2-D histograms from defined 1-D*
DEF2	*Channel No. vs. Channel No.*
DEFBit	*Counts vs. Bit No.*
DEFS	*Set up multiple display*
Live Display	
LIVe	*starts point plot live display*
Conditions and Parameters	
CONDition	*Set a logical condition.*
PARameter	*Set a parameter value.*
KNAme	*Define a name.*

Display Commands	
SPectrum	*Display a spectrum.*
SPEC	*Display a series of spectra.*
MSPectrum	*Display Multiple SPEctra simultaneously (DEFS dependent).*
SPMult	*Display Multiple SPEctra simultaneously(DEFS indep'nt).*
SPVT	*Display spectra on any VT100 compatible terminal.*
Next	*Next spectrum.*
AGain	*Same spectrum.*
PREvious	*Previous spectrum*
MAP	*Make a contour map.*
CUrsor	*Activate cursor.*
X	*Expand.*
MODe	*Lin-log, toggle for pp, event pointer.*
PAge	*Clear the screen.*
SPPrint	*Spectrum print.*
WINdow	*Set a window.*
WINShow	*Show a window.*
WINX	*Set a window on x.*
WINY	*Set a window on y.*
RECORD	*Direct graphic output to an Laser printer data file.*
ENDRECORD	*Resume graphic output to terminal.*
Arithmetic Commands	
ADD	*Add spectra.*
SUBTract	*Subtract spectra.*
MULtiply	*Multiply spectra.*
DIVide	*Divide spectra.*
ADDConstant	*Add constant to spectrum.*
BKG	*Background fit.*
LINFit	*Linear fit.*
GAUssian	*Gaussian fit.*
INTegrate	*Integrate.*
MEAn	*Sum, mean, and FWHM.*
MULConstant	*Multiply spectrum by a constant.*
MULParameter	*Multiply spectrum by a parameter.*
CMPress	*Compress a spectrum.*
SHIft	*Shift a spectrum.*
PROject	*Project 2D to 1D.*
Linearization Commands	
LINE	*Define a linearization.*
LIDiss	*Display and change a linearization.*
LINIo	*Input and output of linearizations.*

Information Commands	
HILfe	*Help (in German).*
SLIst	*Spectrum list.*
CLIst	*Condition list and frequency.*
KLIst	*Name list.*
PLIst	*Parameter list.*
QLIst	*Queue list.*
STATus	*Status report of Lisa.*
SSTatus	*Short status report.*
FM	*Front end monitor.*
FILes	*List data files in default directory.*
BUG	*Examine LISA variables in debug mode.*

Note that the capitalized letters at the front of a command indicate the allowed abbreviation for the command. Note also that some of the commands may not work when the full, proper command is typed. In this case, use the given abbreviation instead.

GLOSSARY OF LISA COMMANDS

ADD

Adds the contents of two spectra and then stores the sum result in a third. Note that the third spectra must be initially defined. This can be done using DEF1, DEF2, etc. Using the ADD function is simple. An example is:

N1 + N2 = N3 N1,N2,N3? 1,2,16 <cr>

In the above example, the contents of spectra 1 and 2 were added by summing corresponding bins (i.e. like adding vectors) and the resultant was put into spectrum 16. Note that the resultant spectra, 16 in our case, must be defined with equal or smaller x and y limits and greater binsize than either of the two spectra which were added.

ADDConstant

Similar to the ADD function except that in this case, ADDConstant adds a constant to all channels in a spectrum much like adding a constant scalar to every component of a vector. For example:

N = N + CONST N,CONST? 3,21.4 <cr>

In this example, the value 21.4 was added to every channel in spectrum 3.

AGain

Replots the last spectrum that was plotted during the current session of LISA.

BEgin

The BEgin command causes LISA to begin reading events from an event file. Any histgograms which have been defined will begin to be filled. If you have any completed histograms, it is advisable to dump them to an output file if they have already been dumped (use DMP). If you do not care whether or not you lose the data in the previous spectra, then you may simply clear all the channels (using CLEar). If you do not clear the channels, then LISA will continue to add to all the channels in every spectrum. Note that before LISA can begin to read, a file must first be opened or specified using the NEW command.

BKG

BKG performs a polynomial fit of a histogram through at least 3 cursor-points. To use the BKG function, do the following:

step	*what to do*
1	-display the histogram to be fitted
2	-type bkg
3	-use the shift key and cursor keys to position the graphic cross hairs that appear on a specific point.
4	-hit the space bar and repeat for any more points along the histogram.
5	-on the last point, do not hit the space bar. Instead hit F to begin fitting.

Output concerning probability and chi squared will be outputted to the screen and also to output file FOR003.DAT.

BUG

BUG examines the contents of Lisa global common area while Lisa is running using the VAX debugger. The global common is examined dynamically. Therefore it may be desirable to end the current run.

step	*command*	*description*
1	$bug	this invokes the debugger
2	DEBUG> go	tell process to startup
3	$debug	this invokes the debugger again
4	DEBUG> ...	now, issue standard debugger commands

CLEar

The CLEar command clears the channels of one or more spectra from LISA memory. To use the the CLEar just type CLEar and then enter the spectra numbers according to the instructions in the prompt.

Examples:

(a)	Clear spectrum , N N1,N2 or "ALL" : 12 <cr>
(b)	Clear spectrum , N N1,N2 or "ALL" : all <cr>
(c)	Clear spectrum , N N1,N2 or "ALL" : 4,8 <cr>

In example (a), the channels in spectrum 12 are cleared. In example (b) the channels in every spectra are cleared. And in (c), the channels in spectra 4 through 8, inclusive, are cleared.

CLFile

The CLFile command stands for "close file" and therefore closes an output file. Only one of two output channels is available when using the CLF command. These channels are either 1 or 2 depending on which one was selected using the OUT command. (See OUT for more information).

Examples:

(a)	$CLF 1 <cr>
(b)	$clf <cr>
	Close output file number (1/2)> 2 <cr>

In example (a), both command and the parameter are issued on one command line. In example (b), first the CLF command is issued. Then the subroutine automatically prompts for an output file unit number when it cannot detect one on the same command line.

CLIst

CLIst represents "condition" list and therefore types a list of all defined conditions and how often they were fulfilled.

Examples:

(a)	$CLI 1,2 <cr>
(b)	$CLIst 5 <cr>
(c)	$clist <cr>

Example (a) causes CLIst to only list the first and second defined conditions.; example (b) causes CLIst to only list the fifth condition; and example (c) causes all the conditions to be listed.

CLPar

Clears the parameter-array

CMPress

Compresses spectrum N into spectrum M (should be smaller). That is, the channels or bins in spectrum N are reduced to a lower number by combining channels. This is extremely useful for situations calling for better statistics when certain bins or channels in a histogram have too few counts. To use the CMP function, one must first define a histogram with a larger binsize or channel with using DEF1, DEF2, or other histogram definition commands. (See DEF1, DEF2, etc). Use of the CMPress command itself is simple.

Examples:

(a)	CMP <cr>
	N1 -> N2: N1,N2? 2,16 <cr>
(b)	cmpr 2,16 <cr>

In example (a) the user first types CMP and carriage returns. The program then automatically prompts for input. In this case, the user compresses spectrum 2 into spectrum 16. In example (b), the exact same result is achieved except this time, the user enters all the data on

one command line. This avoids a prompt and thus saves a few seconds.

CONDition

Defines or changes a condition, which is used to determine if an event meets the criteria to be included in a spectra. (i.e. analogous to a voltage discrimator which requires a signal to have at least a minimum voltage to ensure that it is counted). To use the CONDition command, the following parameter formats may be used. The word COND must be first typed and then followed by one of the four parameter formats listed below.

Parameter formats:

(1)	COOR,IMIN,IMAX,NAME
(2)	ICON1,.op.,ICON2,NAME
(3)	.op.,ICON1,NAME
(4)	COOR,.op.,MASK,NAME

Where COOR represents coordinate (integer or character), IMIN represents an integer lower limit, IMAX an integer upper limit, and NAME is just a character string name for the condition. ICON represents an integer index of the condition and MASK represents a bitmask. In the examples below, the first version is a simple window check. The second version uses as a separator between the ICONs one of the following characters:

Separator	*meaning*	*comments*
.O	.OR.	
.A	.AND.	
.X	.XOR.	
.N	.NOT.	is used in the 3) version
.B	.AND.	with a bitmask is the 4) version

Examples:

(a)	COND 2,100,350,EFEN	tests a window
(b)	COND 3,500,600,TFEN	tests another window
(c)	COND 10,.A,11,BOTH	result is .AND. of previous two
(d)	COND 102,.B,4096,BIT	tests if bit 2**1 is set in KOO102
(e)	COND 102,.B,%O10000,BIT	which has the same effect

CURSOR

draws a dashed marker in a histogram and labels the coordinate. To use cursor, the user must first display a histogram. Then type CURsor command followed by either nothing or an xvalue. If no parameter is supplied in addition to the command, then a graphic cursor will appear on screen (i.e. a set of cross-hairs). The cursor can be positioned using

the cursors and shift keys. Once the correct position is selected, depressing the 1 key will cause the dashed marker to be printed.

Example

(a)	CUR 800 <cr>
(b)	curs <cr>
	{move cursor, then hit 1}

In example (a), the program draws a vertical dashed marker at x=800.0 In example (b), the program will draw a vertical dashed line where the x-coordinate of the graphic cursor was last placed.

DEF1

DEF1 defines a one-dimensional spectrum. That is, it opens or allots a memory location which will be used to store histogram data. Several parameters must be associated with the definition which give the histogram any meaning. These parameters include the x-limits or range of the bins or channels (just like defining channels in a multi-channel analyzer); they also include the width of the channel or binsize; and also the associated event-coordinate which identifies the a specific type of trigger used to detect the event; and finally the condition which is a parameter required to be met before an event is accepted (i.e. like a software discriminator).

Parameters: (for non-interactive mode)
Xmin,Xmax,Xbin,Xcoor,Cond,Name

Notes:

(a)	Cond is specified by either the condition-number or name or -1 for no conditions (or always incrementing) or 0 if the incrementing is done in the user routine
(b)	Xmin, Xmax are the values of the x-limits
(c)	Xbin is the width of a channel or bin
(d)	Xcoord is an event cordinate (i.e. index in KBUF; can also be a name in Knam)
(e)	Name is a character string title for the histogram

DEF1 can be utilized in both interactive or non-interactive modes. In non-interactive mode, DEF1 allows the user to enter all the parameters on one command line by typing DEF1 followed by the parameters separated by commas. The non-interactive mode for defining a 1-dimensional histogram is fast if only a few definitions need to be entered. However, for those users who are not too familiar with LISA or who have many definitions to enter, the interactive mode can simplify and clarify the spectra definition process. To get into the interactive mode, just type DEF1 without any trailing parameters.

DEF2

DEF2 defines a two-dimensional spectrum in much the same way as DEF1 defines a one-dimensional spectrum. The required parameters are similar to the one-dimensional case, except in the 2-D case, the y-limits and y event-coordinate are also required. Note that DEF2 also has interactive and non-interactive modes.

Parameters: (non-interactive mode)
Xmin,Xmax,Xbin,Xcoor,Ymin,Ymax,Ybin,Ycoor,Cond,Name

Fig 1. Below is a sample session using the interactive mode.

```
                    DEF2 DATA ENTRY EDITOR

 PF1=Clear Screen  PF2=Clear Line   MESSAGES:
 PF3=Save          PF4=Help
 Ctrl/R=Restore    Ctrl/Z=Quit
 Ctrl/F=Change Filename
 Ctrl/W=Refresh    Buffer DEF2.COM          (R=REAL, I=INT, C=CHAR)

X-LOWER LIMIT (R): 0.
X-UPPER LIMIT (R): 4100.
X-BINWIDTH    (R): 5.
X-COORDINATE(C/I): thbl1_1
Y-LOWER LIMIT (R): 0.
Y-UPPER LIMIT (R): 100.
Y-BINWIDTH    (R): 1.
Y-COORDINATE(C/I): 37
CONDITION   (C/I): -1
A NAME        (C): thbl1.vs.37
```

DEF12

Defines a two-dimensional spectrum from previously define 1-D spectra

Parameters: Sp#1,Sp#2,Cond,Name

where the range of histograms are from spectra number 1 to Spectra 2 Note: The other parameters have the same meaning as in DEF1 and two non/interactive modes available.

DEFB

Defines a bit-spectrum (16bits). Also has interactive and non-interactive modes.

Parameters: the same as in DEF1

DEFM1

Defines the incrementing of a 1-dim spectrum from a queue spectrum of module numbers. Two non & interactive modes available.

Parameters: (for non-interactive mode)
Istart,Istop,Xbin,Ique#,Ipos,Ithreshold,Cond,NAME

Notes:

(a)	increments for each element found in queue Ique# with a module number between Istart and Istop.
(b)	Ipos is the offset to the beginning of the queue element
(c)	increments only if value larger than Ithreshold
(d)	Ique# may be a symbolic name of a queue

DEFM2

Defines the incrementing of a 2-dim spectrum from a queue contents of all modules versus module number. Two non/interactive modes available.

Parameter: (for non-interactive mode)
Xmin,Xmax,Xbin,Ique#,Istart,Istop,Ipos,Cond,NAME

Notes:

(a)	increments for each element found in queue Ique# with a module number between Istart and Istop.
(b)	Ipos is the offset to the beginning of the queue element
(c)	The Y-axis will be the module number.
(d)	Ique# may be a symbolic name of a queue

DEFMM

Defines the incrementing of a 2-dim spectrum from a queue module number versus module number with threshold on contents. Two non/interactive modes available.

Parameters: (for non-interactive mode)
Istartx,Istopx,Xbin,Iposx,Istarty,Istopy,Ybin,Iposy,
Ique#,Ixthresh,Iythresh,Cond,Name

Notes:

(a)	increments for each pair of elements found in queue Iquenr with module numbers between Istartx and Istopx, and Istarty and Istopy.
(b)	Iposx and Iposy are the offsets to the beginning of the queue element
(c)	Ixthresh and Iythresh are the lower thresholds for the values.
(d)	Ique# may be a symbolic name of a queue.

DEFQ1

Defines the incrementing of a 1-dim spectrum from a queue contents for all modules 'OR'ed. Two, non & interactive modes.

Parameter: (for non-interactive)
Xmin,Xmax,Xbin,Ique#,Istart,Istop,Ipos,Cond,NAME

Notes:

(a)	increments for each element found in queue Ique# with a module number between Istart and Istop.
(b)	Ipos is the offset of the value to the begin of the queue element
(c)	Ique# may be a symbolic name of a queue

DEFQ2

Defines the incrementing of a 2-dim spectrum from a queue contents of module versus contents of module, 'OR'ed for all modules. Two, non & interactive modes available.

Parameter:
Xmin,Xmax,Xbin,Iposx,Ymin,Ymax,Ybin,Iposy,
Ique#,Istart,Istop,Cond,NAME

Notes:

(a)	increments for each element found in queue Ique# with a module number between Istart and Istop.
(b)	Iposx and Iposy are the offsets of the value to the
(c)	begin of the queue element
(d)	Ique# may be a symbolic name of a queue

DEFS

DEFines Screen layout for multiple spectra on one screen. [i.e. allows lisa users to set up a graphics (tektronix) screen with multiple histograms (up to 30) on a single "screen."] The main purpose of DEFS is to provide users the means to display a lot of histogram information in a vary compact and concise way. This can only be achieved if a user can display more than one histogram on a screen. In this capacity, DEFS is a utility program which allows the user to specify which spectra are displayed where on screen and how many on a screen. DEFS has many features suchs as: create, edit, and delete functions, execution of DCL commands within DEFS, and outfile storage of screen numbers. DEFS has also many other convenience features and an extensive help menu. Asynchronous access to the help menu during execution of DEFS allows DEFS to be self-explanatory. To use DEFS, one only needs to understand basic histogramming principles. In short, LISA stores histograms by allocating a certain quantity of memory for each specific plot, and then assigning a number to this quantity of memory. This number is

the spectrum or histrogram number of that quantity. Therefore, manipulation of a specific spectrum or histogram always requires that the spectrum number be entered to direct the program to access the correct block of data. DEFS allows users to specify the number of histograms to be fitted horizontally and vertically, thus making a grid. Then DEFS also allows users to specify which histograms are to be plotted in which grid space and what portion of the entire histogram. For more assistance in the use of DEFS, execute DEFS and obtain asynchronous help from DEFS.

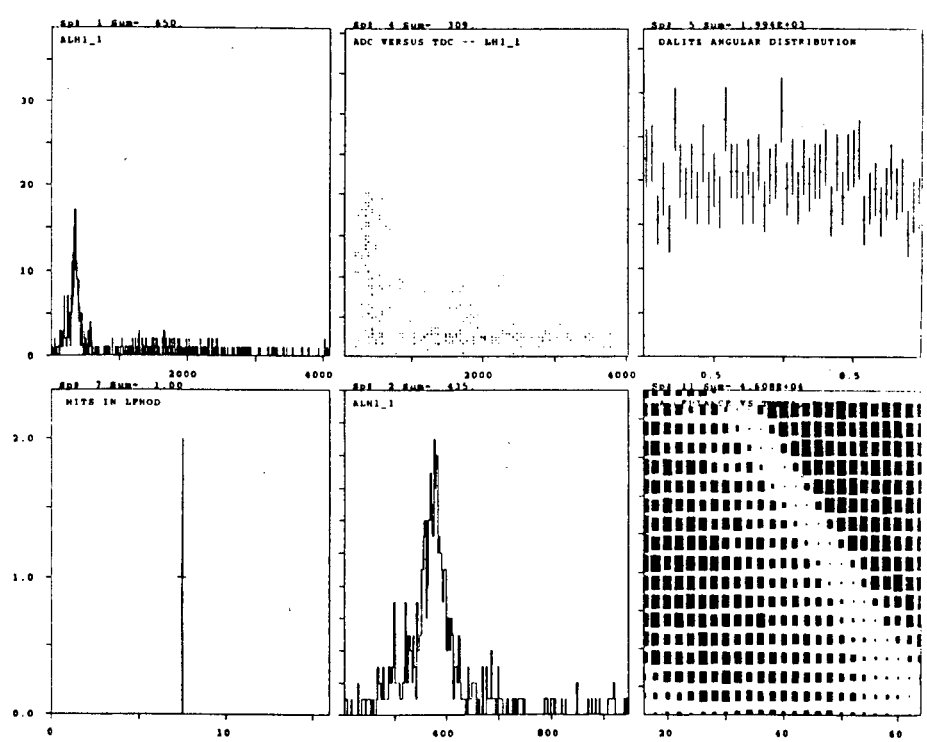
Fig. 2. Shown below is a sample of the DEFS editor.

DEFS — DEFINES SPECTRA FOR MSPE

CURRENT SCREENS IN MEMORY Ctrl/H - HELP, Ctrl/Z - EXIT

SCRN#	XX	YY	SPECTRA NUMBERS	CURRENTLY 5 SCREENS
1	2	2	1 2 3 4	
2	2	2	5 6 7 8	
3	3	2	9 10 11 12 13 14	
4	1	1	15	
5	3	2	1 4 5 7 2 11	

Fig. 3. The graphics screen output for screen 5.



DIVide

Divides one spectrum by another. That is, it divides each channel or bin in the first spectrum by the coresponding channel or bin in the second spectrum and places the result into a third spectrum.

Examples:

(a)	DIV 1,2,16 <cr>
(b)	divide <cr>
	N1 / N2 = N3: N1,N2,N3? 1,2,16 <cr>

Both example (a) and (b) do the same thing. (i.e. they both divide spectrum 1 by spectrum 2 and put the result in spectrum 16.

DMP

Dumps almost all LISA information into a disk file, including spectra.

Parameter: filename

ENd

Causes LISA to stop reading events.

Parameter: none

Example: EN

Note: This command does not close the file, so you can BEgin again

ENDRECORD

Used to stop graphic input from being written to data file. Used after command RECORD was started.

ENQ

Define the type of terminal for graphics. You can tell LISA what graphics terminal in two ways. In the first method type, type ENQ. If an answerback message is set on the terminal, then it will define the terminal type. If not you be requested to enter terminal type. Valid answers are:

VT240	VT640	GRAPHON	4010
4014	4107	4115	IMAGEN

NOTE: The GRAPHON definition is only for Graphon terminals for the VT100 series.

Another way to define graphics terminal type is to use the following logical name definition:

$DEFINE TERM_TYPE xxxx

where xxxx is one of the valid terminal names defined above.

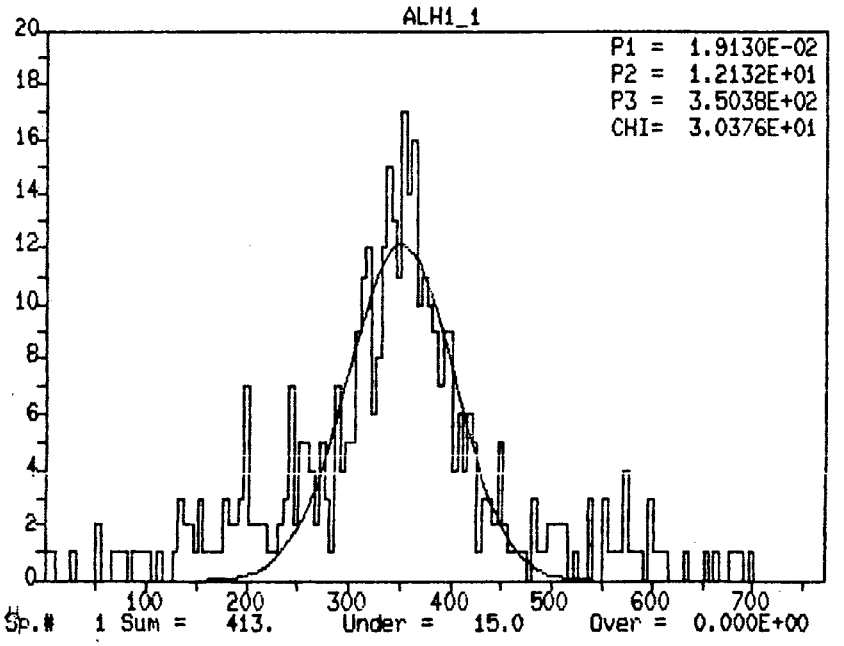
FILes

Display currently available files of events on the default data account.

GAUssian

GAUSS 1, up to GAUSS 4 fits up to 4 gaussians. To use, move cursor keys to position the crosshairs at first approximations to left side, center, and right side of peak. Hit a key after each approximation is made. Output will be both on screen and in FOR003.DAT.

Fig.4. Below is a sample of a guassian fit done on a typical spectrum.



HCONvert

Translates spectra data stored in a LISA DMP file into HBOOK DAT file. This allows LISA users to copy LISA data into HBOOK format and utilize the HBOOK plotting package.(i.e. HCONvert is the inverse of HGET).

Parameter: FILENAME [.DMP]

Notes:

(a)	HDMP and HCONvert are similar. However, HCONvert reads from a DMP file and NOT Lisa memory, hence HCONvert does not require that you dump spectra first into LISA memory using the INDmp command.
(b)	Also, ALL spectra in the DMP file are translated to the DAT file.
(c)	The user cannot specify specific spectra to be stored.

Notes: (cont'd)

(d)	When entering Lisa DMP filename, the .DMP is optional.
(e)	The data is copied in HBOOK format into the same filename except with a .DAT specification (i.e. FILENAME.DMP --> FILENAME.DAT)
(f)	For converting spectra directly from LISA memory to HBOOK DAT file, see [HDMP].

HDMP

Translates spectra data stored in current LISA memory into HBOOK DAT file. This allows LISA users to copy LISA data into HBOOK format and utilize the HBOOK plotting package. (i.e. HDMP is the HBOOK equivalent of DMP)

Parameters: OUTPUT_FILENAME[.DAT] [,Hfirst] [,Hlast]

Notes:

(a)	the .DAT is optional.
(b)	OUTPUT_FILENAME[.DAT] is the character name of the output file.
(c)	Hfirst = number of first histogram in LISA memory to be stored.
(d)	Hlast = number of last histogram in LISA memory to be stored.

If Hfirst and Hlast are not specified then the default is to store ALL the histograms. If only Hfirst is specified, then all the histograms after (inclusive) the Hfirst histogram will be stored.

HGET

Reads histograms from an HBOOK file written with HSTORE, and then translates and adds the histograms to LISA memory. (i.e. allows HBOOK histograms to be plotted using the LISA graphics package.)

Parameter: hbook_filename [X]

Notes:

(a)	If you specify the optional X you will not get the errorbars on 1-dim spectra.
(b)	Asks for HIST-ID, either give the ID of the desired hist, or a start- and a stop-id, or 0 for all.

HIlfe

HELP is used to read the system help-files HILFE is the help-command for LISA (Deutsche ya!) (just in case you're curious, "hilfe" is German for "help")

INDmp

Command in LISA which reads in a DMP-file (also known as a LISA output_file). LISA output_files are created with the DMP command. See Note.

Parameter: filename.

Notes:

(a)	you must have generated this dump with the commands [DMP] or [SDMp]
(b)	If you don't specify the filename with the command, you are prompted for the filename and asked if and how you want to read in the various pieces in the dump.(i.e. add spectra)

INTegrate

Integrates area between cursor marks when LISA is displaying a spectrum. To use the integrate function, first display a spectrum using the SPectrum command and then type INTegrate. A graphic cursor (i.e. a set of cross hairs) will appear. After setting the cursor in one position, hit the 1 key. Then repeat this time hitting the 2 key. The INTegrate function will sum the counts in all the channels between given x-values.

KNAm

KNAm reads in names of event coordinates. This function allows users input character names for the type of trigger used in a detector. This allows users to refer to either the index or id-number of the trigger which is stored in KBUF memory, or to refer to the character KNAm in many functions which require that an event coordinate be supplied.

KLIst

KLIst lists the names of event coordinates. Users can specify what range range of names they wish to view. For example:

Examples:

(a)	KLI 10,100 <cr>
(b)	KLI 16 <cr>
(c)	KLI <cr>

Example (a) demonstrates how to list only the KNames from 10 to 100 inclusive; (b) shows how to list only the individual kname, 16; and (c) shows how to list all the knames in memory.

LIDis

Displays an existing linearization.

Parameter: number of linearization.

You then can redefine a line or only the value of a line.

LINEarize

Defines a new linearization. A linearization is a process which functionally reduces a curved 2-dim plot into a straight line.

Procedure:

(1)	define a 2-D plot either by plotting a stored density plot or live plot.
(2)	Type LINE. LINE asks if the plot is a Density or Live plot.
(3)	Enter a capital D or capital L.
(4)	Then use cursor to define lines.
(5)	End a line with E, and type in the required value for line. Lines have to be uniquely defined in X, with ascending x-values. Lines are not allowed to cross in Y. For positive coordinate numbers two lines are used, for negative numbers four.

LINIo

Input and output of linearizations

LIVe

Defines a point-scatter-plot for live display. Also, multi-live display capability available. New screen managed text editor makes for easy data entry and self explanatory plot specifications. For more help, run LIV and get help.

MAP

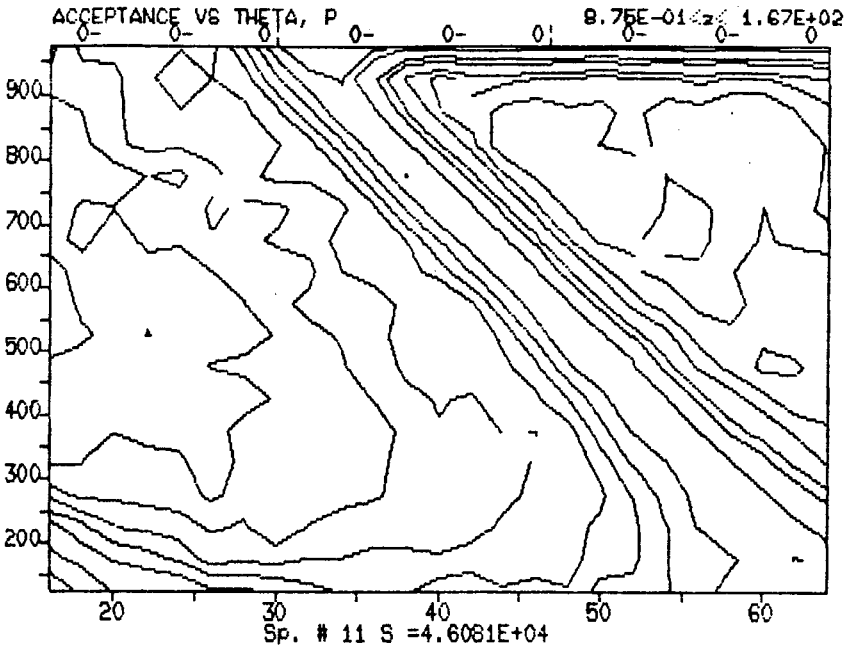
Plots a contour map of a two dim memory spectrum.

Parameters:

Sp#,left_coor,rt_coor,down_coor,up_coor,zmin,zmax,Ismooth,nlines

Default for NLINES is 10, for ISMOOTH = 0. Ismooth determines whether smoothing is done to the contour plot. Default 0 is not to smooth; 1 to smooth.

Fig. 5. Below is a sample contour plot made with the MAP function.



MEAn

Types sum, mean and FWHM of spectra

Parameters: ISTART, ISTOP

Note: If no parameters, it does it for all spectra. For 2-Dim, only sum is correct.

MODe

MODe is like a multiple parameter switch which allows a user to toggle between different LISA functions. MODe can:

(a)	define shared disk read mode.
(b)	switch between linear and log scaling.
(c)	define pixel toggle mode for live-display.
(d)	define type of Tektronix emulation.

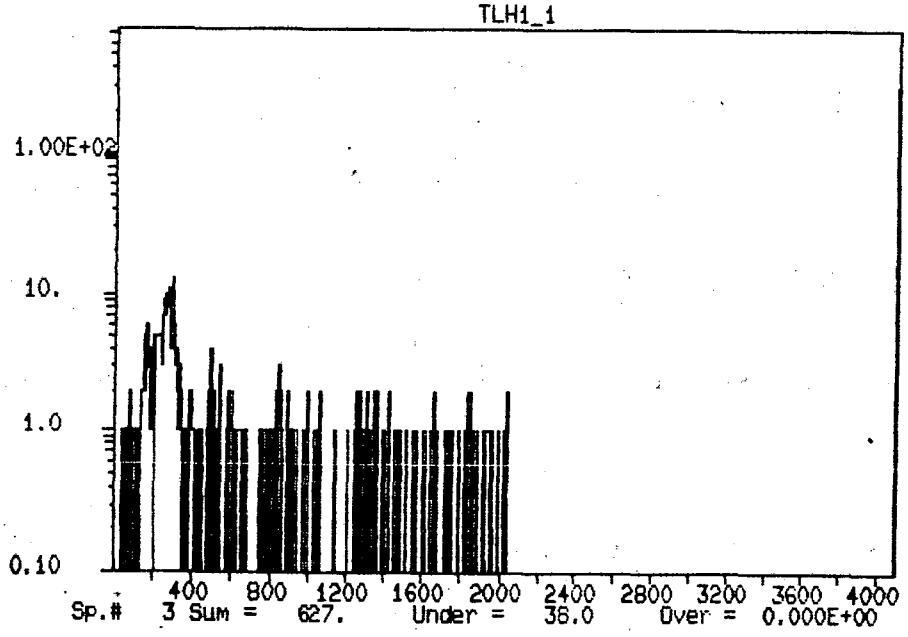
To use mode, type MODe followed by one of the following:

(a)	number parameter (see below)
(b)	LIN or LOG (default is LINear)
(c)	TOG or NOT (default is No Toggle)
(d)	TEK,1 for non-retrographics
(e)	TEK,3 for retrographics (VT640)

Function (a) defines shared disk read mode. (number parameters defined below) Parameter definitions are listed below.

parameter:	*defines:*
get_f_stat_new = 1	flag to start at latest event
get_f_nomiss = 2	flag to ret err if data lost
get_f_skiptobig = 4	flag set if skip large events quietly
get_f_nowait = 8	flag set if not to wait for new events
get_f_stat_cur = 16	stay near the front of the data
get_f_stat_old = 32	start at the oldest (shared data only)
get_f_latest_only = 64	read only the very latest data

Fig. 6. An example of logarithmic scaling.



MOUNT_tape

Command used to mount a tape for LISA reply. This command prompts for the name of the tape drive. This needs only to be done once per tape.

MOVE_tape

After using the command MOUNT_TAPE, this command is used to position the tape to the correct file. It assumes that the a begin run record is id=11. If you have a different number, then you will have to make your own version. See the topic Reading_tapes for further information.

MSPEctrum

MSPE is a LISA command which plots multiple spectra screens previously defined using DEFS. Also see SPMult

Parameters:
Screen1 [,Screen2 or E]

Notes:

(a)	a screen consists of an array of histograms (i.e. Multiple SPEctra) which were previously defined using DEFS.
(b)	it is not necessary to exit DEFS to utilize MSPE. In fact, it is recommended that LISA users utilize the plotting feature in DEFS.
(c)	for more help, exit this help, then type DEFS and get HELP on viewing and plotting. For multi-plot capabilities independent of DEFS see [SPMult].
(d)	Screen1 - integer - first screen to plot. If screen2 is not specified, then only screen1 alone will be plotted.
(e)	Screen2 - integer - if screen2 is specified then all the screens from screen1 to screen2 will be plotted (inclusive).
(f)	E - edit mode - if "E" is specified instead of screen2, the user has the option of setting either an individual range or scale for a single histogram or a global scale for any group of histograms in the screen.
(g)	Screen2 must be either zero or greater than or equal to screen1.
(h)	Edit mode only works when plotting a single screen and will not work when both screen1 and screen2 are specified.
(i)	initial parameters must all be specified with commas as delimiter symbols.

MULConstant

Multiplies all channels of a spectrum with a constant.

MULParameter

Multiplies all channels of a spectrum with a parameter.

Parameter:
Spectrum #, Parameter #

MULtiply

Multiplies two spectra and places the product into a third spectrum. See ADD and DIVide for format and examples.

NEW

Opens an existing disk file for input.

Parameter: file-name

Example: $NEW LIST01

Note: The device, directory and file-extension are assigned with the logical name LIS_EVT: (dts_data_default for shared disk systems)

Example:

```
$ass/table=lisa disk:[user.data].evt lis_evt <cr>
$new list01 <cr>
```

The above example results in opening the file disk:[user.data]list01.evt

Next

Displays the next spectrum

Example: N ! this will display the next spectrum

Note: the previous display-boundaries are assumed.

OUT

Opens one of two possible list-mode-output-files.

Parameter: 1 or 2

Note: defaults are provided by lis_out:. The user has to call explicitly in his user-routine the subroutine OUTVAR1 or 2 with the proper parameters (see source listing)

PAge

Erases the tektronix screen and sets to transparent or alphanumeric mode.

PARameter

Changes the value of a parameter Parameter: IPAR,VALUE or just VALUE

PLIst

Types all non-zero parameters

PROjection

Projects out of a 2-dim spectrum into a existing 1-d spectrum

Parameter: Axis, Sp#2, Sp#1, Ymin, Ymax

Notes:

Axes	axis onto which projection is made (X or Y)
Sp#2	number of the 2-d spectrum to project
Sp#1	number of the 1-d spectrum to store projection
Ymin,Ymax	range of values in the other axis

Note that the projection is added to the defined 1-d spectrum if the bins of the 1-d spectrum have not been cleared.

Examples:

(a)	PRO X,1,13,500.,600.
(b)	PRO X,1,13,600
(c)	PRO Y,1,13

Example (a) projects onto x, from sp#1 to sp#13 in the Y-range from 500 to 600. Example (b) does the same, but only the single row 600. Example (c) projects all columns from sp#1 to sp#13. Note that the 1-dim spectrum can be defined with different binsizes, but different binsizes may result in spikes in the new histogram.

QUEue

Defines a detector queue for decoding and use in automatic incrementing. If you're curious what a queue is, simply put, a queue defines multiple parameters for testing events during data collection. If you're still confused, then you could say that a queue tells LISA to gather information in a histogram for one event with more than one piece of information.

A simple analogy of a queue is a standardized GRE examination. Each exam provides information on the test-taker's personal background (race, sex, etc.), quantitative ability, analytical ability, and verbal ability. When all the exams have been completed, much data has been collected. Each exam can be considered an event, while each person taking the exam can be considered a parameter or event coordinate. Thus each parameter (person) yields a specific amount of information about each event (exam).

When all the data is compiled, we can look at the data in different ways. For example, we can examine a two dimensional scatterplot of race versus total sum of individual test scores; or we can examine a histogram of total points of all test takers for each exam versus exam number. In each example we might get different, but valuable information. For instance, we might find from a scatterplot that a particular group of test-takers all scored the same (and therefore all cheated off the same exam maybe?) or that one of the quantitative exams was the easiest exam because everyone scored well on it. The combinations of ways to utilize this "queued" data are vast. Anyone who needs more information on queues should contact their local LISA expert for help.

Parameter: Ique#,Len_elemt,Max_elemt,FROM,TO,NEW-NR,NAME

Notes:	
(a)	Len_elemt is the number of words per element
(b)	Max_elemt is the maximum number of elements
(c)	FROM,TO are the range of event coordinates used which is calculated using: New_ind=index_to_R_kbuf - FROM + NEW_NR.
(d)	If NEW-NR is 0, no automatic filling is done.
(e)	If NEW-NR is -1, a combined queue is filled. In those cases FROM is the first queue number or name and TO refers to the first condition number or name.

Examples:	
(a)	$QUE 1,2,400,EADC(1),EADC(815),1,EADC
(b)	$QUE 13,4,200,EADC,COND1,-1,COMBINED

Que 13 is defined with 4 words per element, max 200 elements filled by queues EADC and the two following queues. Condition COND1 and the two following ones are used. Elements are only created if the three conditions applied to the respective R_kbuf's are all true.

QLIst

Lists all defined queues

Reading_tapes

To read data tapes directly, you may use the procedures listed in the following table. Note that in this example, the tape drive is located on MTB0:

step	*command*	*description*
(1)	$MOUNT_TAPE MTB0:	!mount the tape drive
(2)	$MOVE_TAPE	!procedure to position tape
(3)	$NEW MTB0:	!tell lisa to read from tape
(4)	$BEGIN	!begin a run

If your program reads an end-of-file, then you might have to do instruction 3 in addition to using 2 to position the tape to the next run.

RECORD

Direct graphic output to a file which can later be sent to a laser printer. Use the command ENDRECORD to close the file and enable graphic output to the terminal.

SDElete

Deletes one or more spectra definitions.

parameters: spectrum #1 [,spectrum #2]

Notes:

(a)	if no parameters are supplied, then LISA prompts for parameters
(b)	spectrum #1 = integer number of the first spectra to delete.
(c)	If spectrum #2 is not specified only spe1 will be deleted.
(d)	spe2 = integer last spectra to delete. If spe2 is specified, then all the spectra from spe1 to spe2 (inclusive) will be deleted.

SDMp

Dumps almost every information into a disk file, except spectra.

Parameter: filename

SLIst

Types a list of all spectra definitions giving type of spectra, limits, binwidths and character titles.

SPectrum

Plots a single one-dim or 2-dim histogram on the logical device TEKTRONIX The default logical device is the graphics screen. However, graphics output may be directed to an outfile using RECORD.

Parameter:
NSPE,XMIN,XMAX,YMIN,YMAX,Y-TEXTSTRING (for 1-dim)
NSPE,XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX (for 2-dim)

Examples:

(a)	sp 5
(b)	SPE 3,200,1500,10,40,LLHC_3

Example (a) plots a histogram of spectrum 5 using the default dimensions already in the memory. Example (b) plots a histogram of spectrum 3 with specific X and Y limits and a y-axis title.

Notes:

(a)	if you omit all parameters except NSPE scaling is done automatically. The parameters XMIN and XMAX refer to the data-space prior to binning.
(b)	linear or log-scale is selected with the MODe command.
(c)	for Multiple histogram plots, see [DEFS], [MSPE] or [SPMult]

SPEC

Command procedure to display histograms in sequential file. A spectrum can be updated continuously or else all spectra can be automatically displayed. This command can be used to store the graphic output on a file which can be later printed on a laser printer.

SPOut

Writes one spectrum to disk.

Parameter: NSp,filename

SPIn

Reads one spectrum from disk

Parameter: filename

SPPrint

Prints a spectrum on LP

SPMult

Displays spectra automatically. Multiple spectra can be displayed with this command. The command prompts for needed information. This is a very fast way to display all spectra.

SPTop

Convert a LISA histogram to a form that can be used by TOPDRAW to make fancy plots.

SPVT

Displays one dimensional spectra on any vt100 compatable terminal. A graphic terminal is not needed.

SUBTract

Subtracts spectra

SHIft

Shifts all channels in a spectrum

STATus

Starts the process STATUS which displays in 5 second intervals the status-report and rings a bell when Lisa stops for an end-run event. To kill this process and clear the screen type Ctrl/C

Fig. 7. Below is an example of STATus output during a run

```
***LISA SYSTEM REPORT***

Experiment:  23-DEC-86 10:28:38     486  4.9    P      17-DEC-1987 13:32:03
Status: running                 Insert: v9.9   Evbev: v3.2    Run No.:    486

        STATISTICS                          I/O - FILES

Total elapsed time :    0 02:55:16   Input : RUN486            rec:      8
Total cpu time(sec):            52   Out 1 :                   rec:
Performance (%)    :            40   Out 2 :                   rec:
Event #            :           190   Displ :
Events/sec         :             2
Events processed   :           190   Spectra:                 15
                                     Conditions:               4
                                     Linearizations:           0
                                     Event Structure:  simple unpacking

Ctrl/C to exit                       Mode:  0 Reading data sequentially.
```

SSTatus

Displays a short status message about file and record

STOPLisa

Stops the execution of LISA. This halts LISA in mid-execution. However, this method is NOT PREFERRED. If an exit is desired, then typing LOgout stops LISA and then cleans-up, allowing plenty of time to avoid a real logout.

TPAR

Changes one parameter using the arrow keys on a VT100

WARTE

Blocks the terminal or the execution of command-files as long as LISA is in the running state.

Parameter: none

Note: block running state means either wait for EOF or NREC, or wait up to the desired records, or -ITIM waits the specified time in seconds.

WINdow

Sets a window condition using the cursor in a spectrum display

Parameters: COOR,COND_NAME

Notes:

(a)	COOR is the KBUF index of an event coordinate
(b)	COND_NAME is the name of a condition
(c)	mark with the cross-hair cursor first the lower then the upper boundary for the test.

WINX

Sets a window condition using cursor on x-axis in a 2-dim display

Parameter: COOR,COND_NAME

Note: same as WIN

WINShow

WINShow is the inverse command of WIN.

Parameter: ICON

Notes:

(a)	ICON is the integer index of the condition
(b)	The boundaries of the window-condition are displayed in the spectrum which is on the display.

WINY

Sets a window condition using the cursor on the y-axis in a 2-dim display.

Parameter: COOR,COND_NAME

Note: only the horizontal line of the cursor is relevant, mark first the lower then the upper boundary

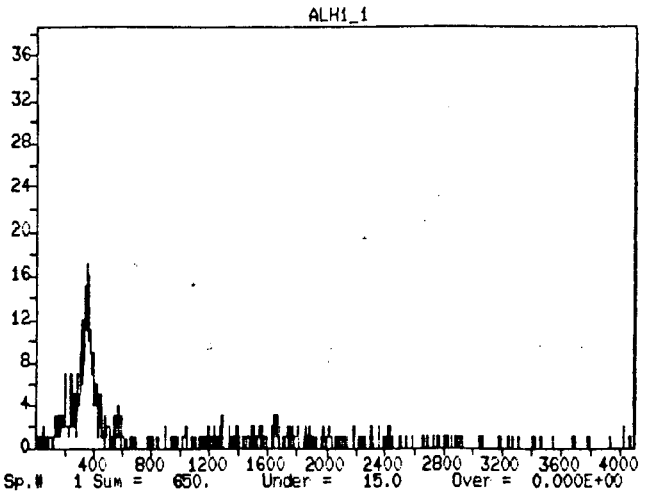
X

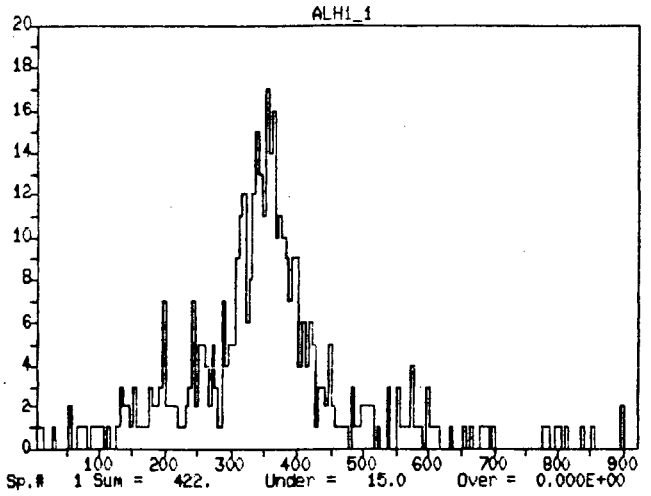
Expands a one or two dimensional histogram with the cursor. To use, first display a spectrum (SPectrum) and then type X <cr>. Using the cursor keys, position the graphic crosshairs at opposite corners of the section to be expanded, striking a key to confirm the position each time.

Parameter: none on the issuing terminal, but typing an [H] in response to the cursor uses also the y-coordinate for scaling

Note: works with two-dimensional display too.

Fig. 8. Shown below are the same histogram; however, the right one has been expanded.





LISA GENERAL FACTS AND OPERATIONS

ASSIGNMENTS

LISA data used in asynchronous processes for both on-line and off-line analyses require the use of output files. The list below shows the logical name assignments which direct output to the correct directory or location. Subroutines in LISA already use these locations automatically. The default memory location is the home directory or the graphics screen terminal as is the case of LIS_LIF. However, assignments may be changed to reflect an alternate location. This can be done during the session using a DCL logical name assignment. Those strong in LISA background may also wish to write their own I/O subroutines in which case the use of LISA assignements in in a fortran OPEN statement is encouraged for storing data in an output file.

LIS_SEC for GLOBAL-SECTION-File
LIS_DMP for Dumps
DTS_DATA_DEFAULT for input of events
LIS_EVT for List-mode-files (input)
LIS_OUT for List-mode-files (output)
LIS_SPE for Spectrum-output(TEKTRONIX:)
LIS_LIF for Live-display

Example Session on LISA

The following table lists a typical sequence of commands that a LISA user might execute after LISA has been started. Those who are not familiar with LISA are welcome to start-up LISA (see start-up) and then go through the sequence below. Note, however, that the datafile RUN15 may not exist in your file directory.

Example Session on LISA	
command	description
DEF1 1,2001,20,190,1,TDC_BULL	define a spectrum which contains the data from 1 to 2001 in 100 bins (bin-size 20), increment the data of coordinate 190 under condition 1
CONDition 8,.B,16384,PULS	define a condition,which is true when the pulser-bit is set in coordinate 8
NEW RUN15	opens diskfile RUN15 with the defaults provided by the logical name dts_data_default. For example: dts_data_default: == disk_a:[user.data].evt. Use the FILes command to obtain a listing of
BEgin	starts to read events from file RUN15.
ENd	stop reading events. Use this command when you think enough data has been collected.

Example Session on LISA (cont'd)	
command	description
SSTatus	performs a Short STatus, (i.e. tells the current filename and record that is being analyzed)
SP 1	displays a histogram of the first spectrum.
X	invokes the cursor to expand the display. See X in the glossary on how to position the cursor.
MODe LOG	switches all following displays to log scales.
AGain	this plots the histogram of spectrum 1 again, only now, the y-axis has been changed to log.
BEgin	continue reading
LIVe 190,0,2200,1190,0,2200,0,BULL	define point plot display, full screen on x-axis coordinate 190 vs 1190 on y-axis, no condition
^c	interrupt live display (this is a subcommand in the LIVe subroutine).
ex	to exit the live display (this is also a subcommand in the LIVe subroutine).
ENd	stop reading events
DMP TEST	this writes to disk all spectra,etc. in file TEST.DMP. The data can be restored at a later session using the INDmp command.
LOgout	logout and stop lisa with cleanup

Extension or File type specifications

The default extensions for certain types of LISA files are listed below. When user name assignments fail to give the extension, then the default extensions are used.

.DMP for Dumps
.EVT for List-mode-data on disk
.LIF for Live-display-diskfiles

Flow Chart of LISA program

The program LISA on VAX computers utilizes the possibility to communicate between different processes via global sections (COMMON-blocks) and event flags.

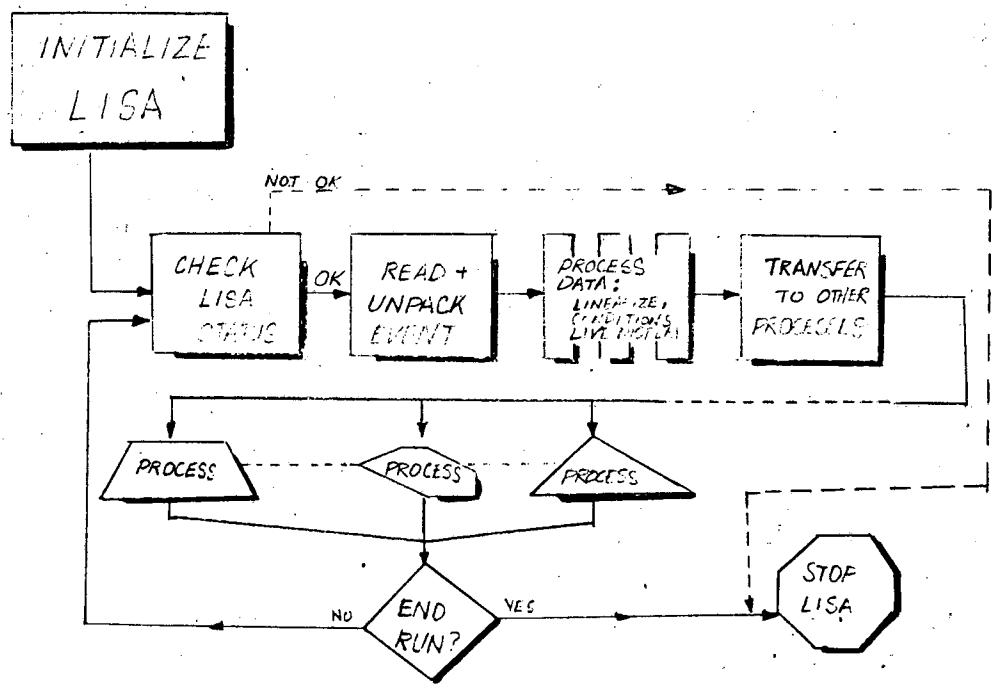
Program LISA itself contains only the process which reads event data from disk, unpacks, analyses, increments spectra, generates point plots of live data and writes sorted data back to disk. LISA is a subprocess started with DCL commands and has no connection to a terminal.

To monitor what LISA is doing a second subprocess STATUS reads all relevant status information and broadcasts it to the requesting terminal.

All other LISA commands start an interactive process at the users terminal, do something (I hope so) and then exit to the monitor.

Normally all those programs reside in one single directory with the logical name LIS_LIBRARY:. Of course every user is free to implement his own commands in his own directory, but in creating those programs it is recommended to use some coding conventions, e.g. use INCLUDE to get all the information about COMMON blocks and the mapping routines.

Program LISA is a simple FORTRAN code to analyze list-mode data. The flow of data is fixed in this eventloop :



Limits

Below is a list of the maximum limits for certain variables in LISA:

LISA maximum limits	
Maximum number of coordinates	10000
Queues	200
Space for queues	20000 words
Integer parameters	100
Real parameters	100
Conditions	500
Linearizations(2*)	25
Linearizations(4*)	1+1* (2*)
Spectra	500
Space for spectra	500000 Real*4 Words
Input file	old data from any disk or new data from the shared disk
Output files	2 separate list-mode-diskfiles

Organization

The LISA software package attempts to incorporate consistent conventions in terms of both the user aspect and the programmer aspect. In terms of the user aspect, LISA usually follows some standard conventions. These conventions include:

a) interactive and non-interactive modes of data entry.
b) logical abbreviations for functions and commands
c) screen management and one-button functions; and
d) error checking.

What is meant by interactive and non-interactive modes of data entry is that when using a LISA command which requires the user to supply some data, the data can be submitted on the same command line as the command, and thus the user bypasses all the unneeded prompts. However, if a LISA command is called, but no data is provided after the command, then LISA automatically prompts for the needed information. A good example of this is the SDElete command. If the user types SDE 1,10 and then hits return, the first through tenth spectra are deleted without a prompt. However, if just SDE is typed, then a sizeable prompt will appear.

The use of logical abbreviations for almost all LISA commands is an aid to users who hate to type out full commands to get something done. Use of logical abbreviations allows for speedier, less annoying access to functions, without having to memorize entirely new vocabulary.

Screen management (SMG) was only recently (in 1987) incorporated into LISA. Use of SMG makes nice looking and easy to read displays (like STATus). SMG also permits "push-button" computing which means computing without having to hit return after every answer.

Incorporation of lots of error-checking decreases the possibility of making errors which propagate through an entire session. Errors are usually caught early, usually when they're entered. Also, the use of error checking in SMG routines allows users to enter multiple data with or without commas. LISA-commands normally can be typed in one line, e.g.:

DEF1 0,2200,10,22,-1,Title of spectrum

or if you don't know the parameters of the call just type the command and answer the questions. When you specify the parameters use always a <BLANK> to separate the keyword from the parameters and use always <COMMA> to separate several parameters. In general parameters can be typed in any format i.e. integer, floating with or without exponent, or as hexadecimal or octal numbers (precede with %X or %O). In most cases where coordinate numbers are requested, it is allowed to give the symbolic name if it was defined with a KName command. You can index names. In commands involving queues and conditions the symbolic names can be used (No index). Keywords can be

abbreviated as specified in the definition.

In terms of the programming aspects, LISA software follows these conventions:

a) consistent use of global sections
b) separation of task and object files, modularization
c) consistent naming of subprograms
d) use of FORTRAN and adequate documentation

Consistent use of global sections involves both using the least amount of global memory as practical and using the correct global memory sections as consistent with other subroutines which use it.

Object files are usually kept separate from the task, or executable, file. This allows storage of long object files in one of LISA's libraries. VAX/VMS allows creation and linking of object libraries with multiple executable tasks and thus saves recompilation time and memory space.

Consistent naming of task and associated subprograms is used. This aids in locating source code when bugs are encountered. For example a sample user program may look like:

```
      PROGRAM SUB
      INCLUDE 'LIS_LIBRARY:LISINC(LISTSK)'  ! this does the
mapping
      CALL SUBS
      END

      SUBROUTINE SUBS
      INCLUDE 'LIS_LIBRARY:LISINC(DATCOM)'   ! COMMON
block
      do something                           ! source code
      RETURN
      END
```

A sample LINK :

```
      LINK FILE,LIS_LIBRARY:LINK/OPT
        be sure to define as link-libraries:
      LIS_LIBRARY:LISLIB
      PLT_LIBRARY:PLTLIB
      HISSLIB is needed for thse programs accessing the shared disk.
```

Note that LISA attempts to use a consistent naming convention for its subroutines. In the above example, the main program (or task) is labelled program SUB while the subroutine is called SUBS. The standard LISA convention is to insert an extra "S" at the end of of a program name to designate that the program is a subroutine. Another example is the program name MDIS, which requests LISA to begin a single or multiple live plot when a user types LIV. In this example

MDIS is the main program associated with the LIV command, but the subroutine which contains all the algorithms is called MDISS, with two "S"s.

All of LISA's code is written in FORTRAN. Why? Because FORTRAN is known world-wide and is an easily learned higher level language. Adequate documen- tation of the written code is encouraged.

LISA commands are organized in a command file which uses symbol definitions to define commands. It is suggested that as more and more features are added to LISA, new commands be also placed in the start-up command files as well. (See Start-up for more information on start-up files.)

Spectra

Principally all spectra are REAL*4 words. The default weighting for all histograms is 1.0. If a different weight is desired, then incrementing of the weighted histogram must be done by the user. Note that the boundaries and the binsize of spectra are REAL numbers!!

There are several types of spectra:

DEF1	normal 1-d incremented from R_KBUF
DEFB	bit-spectrum incremented from R_KBUF
DEF2	normal 2-d incremented from R_KBUF
DEFM1	1-d modulenumber spectrum from queue
DEFM2	2-d contents of modules vs mod.number from queue
DEFQ1	1-d OR of contents of modules inc from queue
DEFQ2	2-d contents vs contents OR of modules incremented from combined queue
DEFMM	2-d module vs module inc from combined queue with threshold on contents
DEF12	2-d from previously defined 1-D

When referring to the parameters of spectra definitions as outlined in the glossary, MIN and MAX refer to begining of first channel and end of last. Thus, channel MAX is included. START and STOP refer to the first and last elements of a queue. Thus, STOP is included.

Start-up

To run LISA many symbols must be defined. These symbols are all contained in one command file. See your friendly LISA programmer for the location of this file for your experiment. A default version is located in LIS_LIBRARY:LISA.COM.

To run your own version of LISA you must begin by defining all these symbols. This can be done by creating a command file in your area which includes a command to execute the LISA command file. For

example, your command file may include this command:
$@local_root:[lisa.lislib]lisa.com or this command:
$@lis_library:lisa.com

Your own specific command file may also include additional symbol definitions, specific commands, and/or assignments. Once this command file is executed, all the definitions and symbols will be set. Then only one task is left, and that is to start LISA. To get the standard bare bones task, just type LISA. This version is only useful for manipulating and viewing histograms and not for processing data events.

The following are some examples of start-up command files:

```
==========================================
$!  RUN.COM start-up command file
$!  Version 1.1                    H. Matis
$!
$! version of lisa to run standard analysis package
$! the call to initialization is machine dependent and can be removed
$! if initialization is done before
$!
$context=""
$set term/nowrap
$lisa_name=f$trnlnm("lis_procnam")
$if lisa_name .nes. "" then goto start
$@local_root:[lisa.dlslib]lisa        !initialization of lisa done here
$goto begin
$!
$start:                               !is lisa running
$pid=f$pid(context)
$if pid .eqs. "" then goto begin
$procnam:='f$getjpi(pid,"prcnam")
$if procnam .eqs. lisa_name then goto found
$goto start
$!
$found:
$stopl
$!
$begin:
$dlsrun
$wait_lisa
$ind dls_library:dls_standard         !dump file
==========================================
```

```
$!  RUNTEST.COM start-up command file
$!  Version 1.1                    H. Matis
$!
$! used to run non-standard lisa programs
$!
$! to use this file you will have to change the definition of test_version
```

```
$! you might also want to change the call to the command file
LISA.COM
$! you may also change the .dmp file call
$!
$context=""
$set term/nowrap
$lisa_name=f$trnlnm("lis_procnam")
$if lisa_name .nes. "" then goto start
$@local_root:[lisa.dlslib]lisa        !initialization of lisa done here
$                                        !machine dependent call
$goto begin
$!
$start:  !is lisa running?
$pid=f$pid(context)
$if pid .eqs. "" then goto begin
$procnam:='f$getjpi(pid,"prcnam")
$if procnam .eqs. lisa_name then goto found
$goto start
$!
$found:
$stopl
$!
$begin:
$test_version:==sys_user:[matis.dls]dls   !change this to your test ver-
sion
$dlstestrun
$wait_lisa
$ind dls_library:dls_standard                    !dls_standard dump
==========================================================
$! LISA.COM start-up command file
$! this command file is used to startup lisa
$!
$! machine dependent calls
$!
$data_disk:==sys_data:[dls_pool]
$lislib$:==local_root:[lisa.lislib]
$pltlib$:==local_root:[lisa.pltlib]
$dlslib$:==local_root:[lisa.dlslib]
$goolib$:==local_root:[goolib.exe]  !remove if you don't need ethernet
$!                                  !shared disk stuff.
$!
$! machine independent calls
$!
$@'lislib$'lisa_setup                !main setup for lisa
$@'dlslib$'lisa_dls                  !dls specific lisa commands
$@'goolib$'mini_goosy                !needed to get ethernet shared disk
stuff
```

LISA TROUBLESHOOTING

First Step

The first step to take when LISA exits during a session for unknown reason, is to type:

$type LISA.PRO

If problems are encountered in start-up, then check if you have:

-adequate diskspace (minimum 6000+ blocks free)
-current version of start-up command procedures
-current version of operating system has not been changed

Funny Printout When Using Graphics

LISA is designed for use in VT100 and compatible terminals. Sometimes, however, when both graphics and alphanumeric modes of screen output are used, funny characters and lots of garbage may occur on the screen. The problem should not occur on actual DEC VT terminals. However, with Graphons and other compatibles, these terminals may not be able to react to software instructions which cause the DEC VT100 to toggle back and forth between graphics and alphanumerics. Therefore, on these terminals, the toggling between graphics and alphanumerics must be done manually with the terminal set-up keys. Make sure VMS knows what terminal type you are using. Use the command:

$SET TERMINAL/INQUIRE

Another common cause of funny graphics output is the terminal "wrap" parameter. The terminal should be set to "nowrap." Sometimes, however, when a "set host" is done to another account, the terminal may be switched to "wrap." When a graphics screen starts drawing a picture and suddenly draws random characters on the screen, a thing to check is the terminal wrap parameter. To set the wrap parameter, type:

$SET TERMINAL/NOWRAP

Subroutine Crash, Failure, Mix-up

LISA subprograms have been carefully tested and are assumed to work. If an error message appears, attempt to ascertain the nature of the error. Check the LISA documentation for correct input format or if the subprogram has interactive help, run the subprogram again and get the help files. If it is determined that the correct instructions are being carried out, and the error is still occuring, then take note of the error and go to your LISA expert, or better yet, go to the person who wrote the subroutine if the person is available.

Occasionally, a Mix-up will occur. A Mix-up error occurs only when a

LISA plotting or filing routine is used. Usually, either the plots are all mislocated on the screen, or the index numbers you just entered have mysteriously changed, or things just plain ain't the way you entered them. What is happening is that multiple subroutines are accessing the same LISA global commons and therefore, over-writing each other. Since a user can only use one LISA function (and therefore one subroutine) at a time, the only time a mix-up occurs is when multiple terminals are logged on to the same account using LISA. The mix-up is caused by over-writing the same global commons from different process names on the same account. Although much has been done to protect LISA against mix-ups caused by multiple process names, the very nature of global memory prohibits full separation of LISA for individual process names on the same user account.

Sudden Unexpected Termination and Clean-up

Sudden unexpected terminations of the LISA processes can sometimes occur when not so desired. There are several types of errors which may cause this to occur. When sudden termination does occur, the first thing to do is NOT to go bother your local LISA expert. (The expert may make you look like a fool.) The first thing to do is check the last LISA logout status. LISA's logout status is located in a file called LISA.PRO. To find out the cause of logout, type: $TYPE LISA.PRO <cr> If still cannot ascertain the reason for termination, then it may be necessary to contact your local LISA expert. (Warning: the expert may still make you look like a fool)

In most cases, sudden termination of LISA usually occurs within a minute or two of initialization. In these cases, the problem has almost always something to do with an alteration in the LISA start-up command procedures or a lack of available diskspace to run LISA. The alteration of a command procedure may be a simple change of location of a revised start-up command file, or a revised start-up data file. The solution is to reinitialize LISA using the correct start-up files. Lack of diskspace is always accompanied by some message which says: [ACP file create failed]. Purging unnecessary files or obtaining more diskspace will solve this problem.

A second, less common cause of sudden termination is a bug in a LISA subroutine. LISA subroutines aren't all perfect, and therefore, will have bugs in them. This is especially true of new subroutines. Luckily, however, most bugs in subroutines will cause the subroutine to crash, but not LISA. A subroutine which causes LISA to crash is a serious error and should come to the immediate attention of a local LISA expert. (In this case, the expert who wrote the subroutine is the fool.) A subroutine which causes LISA termination is identified by the fact that LISA crashes only when a particular function is used. This type of error usually requires rewriting some or all the code in the subroutine.

A third and worst cause of sudden LISA termination is the change to a newer version of the VAX/VMS operating system. When LISA is working fine at the current version of VMS (i.e. version 4.6) and a change to a new version of the operating system causes unexpected termination in LISA, then it's time to contact as many LISA experts you can find. The cause of LISA's incompatibility with a new operating system almost always has something to do with the new operating system's inability to support software written using the older operating system.

This manual has been carefully checked. However, as with LISA, there are some hidden bugs in this document. Please report all errors and suggestions for improvements to the authors.

LAWRENCE BERKELEY LABORATORY
TECHNICAL INFORMATION DEPARTMENT
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720