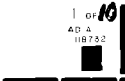
AD-A118 732 PAR TECHNOLOGY CORP NEW HARTFORD NY F/G 9/2
ON-LINE PATTERN ANALYSIS AND RECOGNITION SYSTEM, OLPARS VI. SOF--ETC(U)
JUN 82 S E HAEHN, D MORRIS
UNCLASSIFIED PAR-82-20 NL



(2)

PAR TECHNOLOGY CORPORATION

AD A118732

DTIC FILE COPY

DTIC ELECTE AUG 31 1982 D H

Approved for public release:
distribution unlimited

82 08 24 025

(2)

OLPARS Software Reference Manual

June 18, 1982

DTIC
ELECTE
AUG 3 1 1982
D

DTIC
COPY
INSPECTED
2

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD A115 732	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) OLPARS VI (On-Line Pattern Analysis and Recognition System)		5. TYPE OF REPORT & PERIOD COVERED OLPARS Software Reference Manual
		6. PERFORMING ORG. REPORT NUMBER 82-20
7. AUTHOR(s) Mr. Steven E. Haehn Ms. Donna Morris		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS PAR Technology Corporation Route #5, Seneca Plaza New Hartford, New York 13413		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Department of Defense Washington, D. C.		12. REPORT DATE June 18, 1982
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same as Block 11		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

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

18. SUPPLEMENTARY NOTES

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

Pattern Recognition, Structure Analysis, Discriminant Analysis
Data Transformation, Feature Extraction, Feature Evaluation
Cluster Analysis, Classification Computer Software

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

There are 2 manuals referred to under this title: A Programmers Reference and Maintenance Manual (PRM), and a Software Reference Manual (SRM). The PRM gives a detailed view on OLPARS and data structures and concepts. The SRM gives detailed information on all the OLPARS commands and their subprograms (e.g., calling sequences, function).

OLPARS Software Reference Manual

Submitted By

PAR TECHNOLOGY CORPORATION
Route 5, Seneca Plaza
New Hartford, New York 13413

Authors

Mr. Steven E. Haehn
Ms. Donna A. Morris

SECTION 1

INTRODUCTION

This document presents detailed designs of the programs necessary to carry out the OLPARS functions. The format of the description for each program is given in Figure 1-1. If a particular item of the format does not pertain to a program (e.g., a subroutine has no output arguments), it is omitted from the description of that program. The description of the programs appear alphabetically in Section 2. Appendix A contains a list of programs that are system dependent and, hence, must be revised on each separate installation of OLPARS. Also included is a list of system-dependent parameters, in Appendix B, and a list of named COMMON blocks, in Appendix C. A cross reference listing of all the currently existing OLPARS programs can be found in Appendix D.

The description of the portable OLPARS filing system is contained in the OLPARS Programmer and System Maintenance Manual (also referred to as the OLPARS Programmer's Reference Manual). The specifications of the programs constantly refer to this document for these descriptions, and it is assumed that a person using these Program Specifications also has a copy of the Programmer's Reference Manual. An overview of the entire system is given in the OLPARS V Final Report.

PROGRAM NAME:

The name used to activate the program.

CATEGORY:

The program categorization types:

OLPARS Executive
OLPARS Programmer Aids
Utility command
Measurement Evaluation command (subsidiary)
Structure Analysis command (subsidiary)
Logic Design command (subsidiary)
Transformation command
(INTERNAL SUBROUTINES)

File Manipulation (level I or II)
File Access (level I or II)
Data Tree File Access
Display File Access
Tree List File Access
Logic List File Access
Communication File Access

Measurement Evaluation Routine
Structure Analysis Routine
Logic Design Routine
Numerical Computation Algorithm
Terminal I/O (character or graphics)
Terminal Display Routine
Utility Subroutine

NOTE

When any of the above are "system dependent," that fact should be mentioned too.

PROGRAM DESCRIPTION:

A functional description of the program

PROGRAM USAGE:

The calling sequence of subroutines, or the possible program usage and type of functions

INPUT ARGUMENT(S):

All input arguments to a SUBROUTINE, with a description of each one, would appear here.

Figure 1-1 Specification Format

OUTPUT ARGUMENT(S):

All output arguments of a SUBROUTINE, with a description of each one, would appear here.

WORKING SPACE ARGUMENT(S):

Any working space argument to be used by a SUBROUTINE, with a description of each one, would appear here.

USER INTERACTION:

Program requests, responded to by user

ALGORITHM / NOTES:

The basic algorithm, describing the program, using a block structured, English-program language notation

FILES:

Names of the files accessed by this program

REFERENCED BY:

If the program is a SUBPROGRAM, the name(s) of the calling program(s) go here.

If the program is a main program (i.e., an OLPARS command), "OLPARS user types 'program name'" goes here. Note, this section may not be complete. For a total list of references, look up program name in Appendix D.

SUBPROGRAM(S) REFERENCED:

The name(s) of the subprogram(s) called

DIAGNOSTICS:

Mention any error diagnostics a SUBROUTINE may return to its calling program

SEE ALSO:

Mention other programs, not specified in the above 'reference' lists, that are of a similar nature, or work in conjunction, with this program

HISTORY:

Person(s) who designed the program, along with the date the program was designed and written.

Figure 1-1 Specification Format (cont'd)

PROGRAM SPECIFICATIONS

PROGRAM NAME: ADDMCV

CATEGORY: Data Tree File Access Routine

PROGRAM DESCRIPTION:

ADDMCV adds the mean and covariances of two nodes. It is passed the file descriptor (in FDTI) of the TI file, the entry number of the destination node (in DESNOD) and the dimension of the data set (in NDIM). The results are placed into entry DESNOD.

PROGRAM USAGE:

```
CALL ADDMCV(FDTI,DESNOD,NDIM,NVECB,MEANB,COVB)
```

INPUT ARGUMENTS:

FDTI - INTEGER; the file descriptor of the TI file

DESNOD - INTEGER; entry number of DESTINATION node

NDIM - INTEGER; dimension of the data set

NVECB - INTEGER; the number of vectors in the SOURCE node

MEANB - REAL ARRAY(NDIM); mean vector of the SOURCE node

COVB - REAL ARRAY(NDIM*(NDIM+1)/2); the covariance matrix of the SOURCE node

ALGORITHM/NOTES:

The computation is done as follows:

Let mA(i), mB(i) = the ith element of the means for node A and B, respectively.

mAB(i) = the ith element of the mean for the combination of nodes A and B.

nA, nB, nAB = the number of vectors contained in nodes A, B, and A+B.

cA(i,j) cB(i,j) cAB(i,j) = the i,jth element of the covariance matrix of nodes A, B, and A+B.

Then

EQ. 1 $nAB = nA + nB$

$$\langle \text{mean for meas}(i) \text{ of } A+B \rangle = \frac{\langle \text{sum meas}(i) \text{ in } A \rangle + \langle \text{sum meas}(i) \text{ in } B \rangle}{\langle \text{total number of vectors in } A+B \rangle}$$

EQ. 2 $$mAB(i) = \frac{nA * mA(i) + nB * mB(i)}{nAB}$$

$$\langle \text{covariance of meas}(i) \text{ and meas}(j) \text{ of } A+B \rangle = \left(\frac{\langle \text{sum of meas}(i)*\text{meas}(j) \text{ in } A \rangle + \langle \text{sum of meas}(i)*\text{meas}(j) \text{ in } B \rangle}{\langle \text{total number of vectors in } A+B \rangle} \right) - \langle \text{meanAB}(i)*\text{meanAB}(j) \rangle$$

EQ. 3 $$covAB(i,j) = \left[\frac{nA*(cA(i,j)+mA(i)*mA(j)) + nB*(cB(i,j)+mB(i)*mB(j))}{nAB} \right] - mAB(i)*mAB(j)$$

Equation 1 (EQ. 1) is the sum of the vectors at both nodes. Equation 2 (EQ. 2) is the sum of the mean vectors of both nodes. Equation 3 (EQ. 3) is the sum of the covariance matrices of both nodes.

FILES:

TI - tree information

REFERENCED BY:

ADUPCM, COMNOD

SUBPROGRAMS REFERENCED:

INSINT, INSPGM, INSRET, TIGCOP, TIGCOV, TIGMN, TIPCOV, TIPMN

HISTORY:

designed by DAVID BirnBaum June 12, 1978

programmed by Steven Haehn Aug. 18, 1980

PROGRAM NAME: ADUPCM

CATEGORY: Data Tree File Access Routine

PROGRAM DESCRIPTION:

ADUPCM traverses the tree, via parent pointers, and adds the mean vector and covariance matrix of the given node to the mean vectors and covariance matrices of all the nodes above it.

PROGRAM USAGE:

```
CALL ADUPCM(FDTI,ENTABL,NODE)
```

INPUT ARGUMENTS:

```
  FDTI - INTEGER; file descriptor of the TI file
ENTABL - INTEGER array (TABSIZ); the entry table
                                 of the TI file
 NODE - INTEGER; slot number of the given node
```

ALGORITHM / NOTES:

```
WHILE (the present node is not the senior node)

    Obtain the parent pointer of the present node.

    Add the mean vectors and covariance matrices
    of the two nodes (the parent and the child),
    storing the result in the parent node.

    Make the parent node the present node.

ENDWHILE
```

FILES:

TI - tree information

REFERENCED BY:

APPEND

SUBPROGRAMS REFERENCED:

ADDMCV, INSINT, INSPGM, INSRET, TIGCOP

HISTORY:

designed by Steven Haehn June 8, 1978

programmed by Steven Haehn Aug. 19, 1980

PROGRAM NAME: ALNCPN

CATEGORY: Data Tree File Access Routine

PROGRAM DESCRIPTION:

ALNCPN (add lowest node counts, pointers and name) inserts a lowest node (minus means and covariances) into a tree. The file descriptor (FDTI) points to the tree information file that the new node is to be added. The slot numbers in the node entry table of the PARENT node and the NEWNOD(E) are given as arguments (i.e., NEWNOD is to be inserted under PARENT). The node name (NODNAM), the number of vectors (NVEC) contained in the node, and the vector pointer (VECPTR) that points to the node's vectors within the tree vector file, are also given as arguments.

NOTE: Parent node must have at least one child before this routine will work properly.

PROGRAM USAGE:

```
CALL ALNCPN(FDTI,ENTABL,PARENT,NEWNOD,NODNAM,NVEC,
            VECPTR)
```

INPUT ARGUMENTS:

FDTI - INTEGER; file descriptor of the TI file

ENTAB - INTEGER ARRAY(TABSIZ); the TI file entry table

PARENT - INTEGER; slot number of the new node's parent

NEWNOD - INTEGER; slot number of the new node to be added to the data tree

NODNAM - INTEGER ARRAY(NODLEN); name of the new node that is being added to the data tree

NVEC - INTEGER; the number of vectors residing at the new node

VECPTR - INTEGER; pointer to the new node's vectors found in the TV file

ALGORITHM / NOTES:

Obtain the entry table number of the parent's last child and the node level of the child

Set the sibling pointer of the retrieved child to NEWNOD.

WHILE (the current child is not a lowest node)

    Locate the youngest child node that is a lowest node

ENDWHILE

Set the new node's lowest-node-link-pointer to the child's lowest-node-link-pointer.

Set child's lowest-node-link-pointer to NEWNOD.

Set the new node's lowest node back pointer to the child's entry table number.

IF (the new node is not the last lowest node)

    Set the lowest-node-back-pointer of the node pointed to by the new node's lowest-node-link-pointer, so it points to the new node.

ENDIF

Set the TV vector pointer and the number of vectors count, within the new node, to their given values.

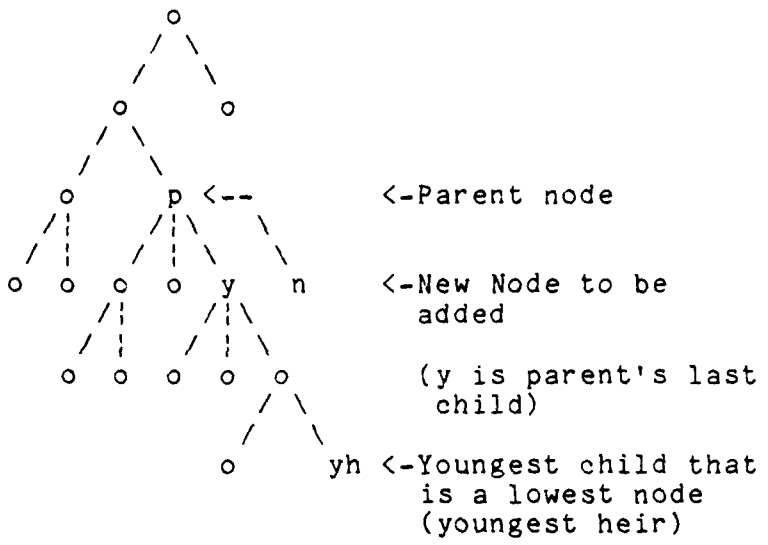
Set the first child pointer, number of lowest nodes beneath, and number of children of the new node to zero (initialized at compile time).

Set the entry table back pointer of the new node to NEWNOD and the new node's parent pointer to PARENT.

Set the new node name to NODNAM.

RETURN

OLPARS Tree figure, with node name references made in ALNCPN.



FILES:

TI - tree information

REFERENCED BY:

APPEND, CLNODE

SUBPROGRAMS REFERENCED:

INSCHR, INSINT, INSPGM, INSRET, TIPCOP, TIGCOP, TIPCAP, TIPNAM

SEE ALSO:

CLNODE

HISTORY:

designed by Steven Haehn June 7, 1978

programmed by Steven Haehn Aug. 13, 1980

PROGRAM NAME: ANYTHING

CATEGORY: Utility Command (system dependent)

PROGRAM DESCRIPTION:

ANYTHING displays a text file containing all the available OLPARS commands.

PROGRAM USAGE:

User types in 'ANYTHING'.

ALGORITHM / NOTES:

The file (ANYTHING.TXT) to be displayed at the user's terminal will reside in the OLPARS directory. It can be created and modified by any text editing program (e.g., EDI).

FILES:

ANYTHING.TXT
CM - communications

REFERENCED BY:

OLPARS user.

SUBPROGRAMS REFERENCED:

CMGDIR, CONCAT, ERASE, FILGET, INSINI, INSRET, OEXIT, OPENFX, OPENS, MENU, TRMPUT

HISTORY:

designed by Steven Haehn July 31, 1978

programmed by Steven Haehn April 21, 1980

PROGRAM NAME: APPEND

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

APPEND appends a node from one tree to another. It obtains a lowest node from one tree and attaches it under an intermediate node of another tree (or same tree).

PROGRAM USAGE:

User types in 'APPEND'.

USER INTERACTION:

The user is asked for the node names of two trees (the names may be the same). The first name (tree 1) the user gives to the program specifies the source tree (the tree from which the new node is to be obtained). The second name (tree 2) specifies the receiving tree (the tree to which the node is to be appended).

The user is then asked for the names of two nodes. Node1 (source node) must be a lowest node. Node2 (receiving node) must be a intermediate node. The senior node can be the receiving node, but not the source node.

The user is now asked for a new name for the node that is being appended. Lastly, the user is asked if he wants to resequence the vector identifiers of the appended node.

ALGORITHM/NOTES:

Erase screen and home cursor (ERASE).

Obtain user interaction information (UIAPND)

Obtain a slot number in the entry table (tree 2) and an entry number for the new node (GNXTAN).

IF (entry table is full) THEN

Display error message to user: 'tree is full: no more nodes can be added.'

ELSE

Copy vectors from source tree to receiving tree (COPYVC).

Add the nodename, counts and pointers of the new node to receiving tree (ALNCPN).

Copy mean and covariance of source node in source tree to new node in receiving tree (COPYMN,COPYCV).

Add the means and covariances of this new node to all nodes directly above it (ADUPCM).

Adjust the number of children at the recieving node, and the number lowest nodes in all the intermediate nodes above the node appended. (FIXKLV)

ENDIF

Put up option list at user's terminal (MENU).

END

FILES:

CM - communication
HS - history
TI - tree information
TL - tree list
TV - tree vector
OLPARS option file
Instrmentation file

SUBPROGRAMS REFERENCED:

ADUPCM, ALNCPN, CMGOTH, COPYCV, COPYMN, COPYVC, ERASE
FIXKLV, GNXTAN, INSINI, INSINT, INSRET, MENU, OEXIT
OPENFX, TIGCOP, TRMPUT, UIAPND

SEE ALSO:

COMNOD, DSUBSTRC, CREATREE, COPYDS

HISTORY:

designed by Steve Haehn June 9, 1978

programmed by Mark Maginn August 1, 1980

PROGRAM NAME: ASCDEC

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

ASCDEC converts ascii characters to a decimal number

PROGRAM USAGE:

CALL ASCDEC(STRING,STRPTR,STRLEN,NCHAR,NUMBER)

INPUT ARGUMENTS:

STRING - LOGICAL *1 ARRAY; character string to be converted

STRLEN - INTEGER; the length of the character string to be converted. this is the length of the string from beginning to end, not from 'STRPTR' to end, unless, of course, 'STRPTR' equals 1.

STRPTR - INTEGER; index in the character string at which to start conversion

NCHAR - INTEGER; the number of characters to convert. if the calling program does not know the number of characters to be converted, then nchar can be set to a negative number and a maximum of 5 characters will be converted. conversion will stop as soon as a non-numeric character is encountered.

OUTPUT ARGUMENTS:

NUMBER - INTEGER; the decimal number that is the result of the conversion. 'number' is set to -1 under the following conditions:

(1) if a 5-character conversion results in an integer overflow

(2) if 'nchar' is positive and a non-numeric character is encountered before 'nchar' is exhausted

(3) if 'nchar' is greater than 5

REFERENCED BY:

GEN

HISTORY:

designed by Donna Morris January 30,1979

programmed by Donna Morris March 2,1979

PROGRAM NAME: BINWIDTH

CATEGORY: UTILITY COMMAND

PROGRAM DESCRIPTION:

BINWIDTH is used for modifying the scale of a one-space display. The following list shows the portions of a one-space display scale that can be altered.

1. Change the starting point of the display scale.

2. Change the number of bins on the scale.

3. Change the interval size (bin size).

PROGRAM USAGE:

User types in 'BINWIDTH'

USER INTERACTION:

The user is first asked if the minimum scale value is to be changed. Then the user is asked if he wants to change the interval size and/or the number of bins that are on the scale. If these values are to be changed, he is then asked for the new values.

ALGORITHM / NOTES:

```
Ask user if he wants to change the minimum
projection scale value.

Ask user if he wants to change the number of
bins

Ask user if he wants to change the interval
size (size of a bin).

IF (the minimum scale value is to be changed)

    Change the minimum scale value.

    IF (the number of bins was user specified)

        Change the number of bins to user's
        value.

        Change the maximum scale value

    ENDIF
```

ELSE

IF (the number of bins was user specified)

Change the number of bins to user's value.

Change the maximum scale value

ENDIF

ENDIF

Write out new values to DI header

Turn the screen coordinate flag in the DV file header 'off' so that the display subroutine "knows" that it must recalculate a new set of screen coordinates for the one-space display

Redisplay the user's one-space display (MICMAC).

NOTE

Actual file changes are not made until user has completely answered all queries. The maximum scale value will have to be altered when the number of bins value is recalculated, because the value of the number of bins is required to be an integer number.

e.g., if XMAX = 5, XMIN = 0, BINSIZE = 2

using integer arithmetic,

No. bins = (XMAX - XMIN)/BINSIZE

= (5 - 0)/2

= 2

Therefore, XMAX in the DI file must be adjusted.

XMAX = (no. bins*BINSIZE) + XMIN

= (2*2) + 0

= 4

OLPARS Program Specifications
BINWIDTH

FILES:

CM - communications file
DI - display Information file
DV - display vecor file
PV - projection vector file
S1 - scratch 1 file
Instrumentation file

SUBPROGRAMS REFERENCED:

CLOSFX	CMGOTH	DIGHDI	DIGMAX	DIPHDI	DIPMAX	DVPHDR
EQUALA	ERASE	INSFLT	INSINI	INSINT	INSRET	MENU
MICMAC	OEXIT	OPENFX	PROMPT	TRMGET	TRMPUT	

HISTORY:

designed by Mark Maginn November 21, 1980

programmed by Mark Maginn November 21, 1980

PROGRAM NAME: BOUND

CATEGORY: UTILITY ROUTINE

PROGRAM DESCRIPTION:

BOUND converts a screen coordinate to a projected data value. It is used to convert graphic input to a usable real number.

PROGRAM USAGE:

```
REAL BOUND

VALUE=BOUND(SCRN,MINV,MAXV,WIDTH,BCOORD)
```

INPUT ARGUMENTS:

SCRN - INTEGER; screen coordinate to be converted

MINV - REAL; minimum data value

MAXV - REAL; maximum data value

WIDTH - INTEGER; width (or height) of display field minus width of one display character

BCOORD - INTEGER; screen coordinate of lower left corner of display rectangle plus one character

OUTPUT ARGUMENTS:

BOUND - REAL; the data value corresponding to the screen coordinate

ALGORITHM / NOTES:

```
IF (screen coord less than boundary coord) THEN

    BOUND = MINV
    SCRN = BCOORD

ELSEIF (screen coord greater than upper boundary) THEN

    BOUND = upper boundary

ELSE

    BOUND = appropriate data value

ENDIF
```

REFERENCED BY:

DRAWBNDY, OPTDSP

SEE ALSO:

DRAWBNDY

HISTORY:

designed by David J. Tipton Sept. 7, 1979

programmed by David J. Tipton Sept. 7, 1979

PROGRAM NAME: BYEOLP

CATEGORY: Utility Command (system dependent)

PROGRAM DESCRIPTION:

BYEOLP "signs" a user off of OLPARS.

PROGRAM USAGE:

User types in 'BYEOLP'.

ALGORITHM / NOTES:

For now, this will simply allow the CIP to exit. The function is implemented using an RSX-11M command file (see appendix A in OLPARS VI Programmer's Reference Manual).

REFERENCED BY:

OLPARS user.

SEE ALSO:

HELOLP

HISTORY:

designed by Steven Haehn July 28, 1978

programmed by Steven Haehn June 28, 1981

PROGRAM NAME: CDEFAULT

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

CDEFAULT puts up a list of options and asks the user to select one. By selecting the appropriate option, the user may (1) change the cluster/scatter cut-off value, (2) change the one-space bin factor, (3) change from long prompts to short prompts or vice versa, (4) change the instrumentation flag from 'ON' to 'OFF' or vice versa, or (5) set default values for the CM file parameters mentioned in 1,2,3 and 4 above. The effects of changes in these options are as follows:

Option 1: If the number of vectors in a data set to be projected on a two-space plot is greater than the cut-off value, a cluster plot is displayed, otherwise a scatter plot is shown. If option 1 is selected, the program requests that a new value be input. Example: if the cut-off value were set to zero, all two-space plots would initially be displayed in the cluster mode.

Option 2: The initial number of bins for a histogram plot is determined by dividing the total number of vectors by the number of classes times the one-space bin factor. If option 2 is selected, the program requests that a new bin factor value be entered. Example: if the number of classes is 1, the bin factor is set to 2, and the total number of vectors is 100, these vectors will be placed in 100/(1*2) = 50 bins.

Option 3: The default value of the prompt flag is 0 - use short prompts. By changing to long prompts (flag = 1), the user receives longer messages requesting information.

Option 4: The default value of the instrumentation flag is 0 - disable (turn off) instrumentation. By changing the instrumentation flag to 1, the instrumentation is

enabled (turned on).

Option 5: The default value of the instrumentation threshold is 5. If instrumentation is 'ON', a program must have completed successfully at least 5 times in order turn off the printing of debug information.

Option 6: If option 6 is selected, the cluster/scatter cut-off value is set to 500, the one-space bin factor is set to 5, the prompt flag is set to 0, the instrumentation flag is set to zero, and the instrumentation threshold is set to 5.

PROGRAM USAGE:

User types in 'CDEFAULT'.

USER INTERACTION:

CDEFAULT asks the user to select changes from the following list:

1. Cluster/scatter cut-off value

2. One-space bin factor

3. Change prompt flag from current one.

4. Change instrumentation flag from current one.

5. Change instrumentation threshold value from current one.

6. Set default values as follows:
two-space cut-off value = 500
one-space bin factor = 5
prompt flag = 0 (short prompt mode)
instrumentation flag = 0 (disable)
instrumentation threshold value = 5

OLPARS Program Specifications
CDEFAULT

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Get current default information from CM file (CMGOTH).

REPEAT

Prompt user for a change.

Request appropriate value.

Enter value in memory.

UNTIL (user quits)

Write new information back into CM file (CMPOTH).

Put up option list at user's terminal (MENU).

EXIT

FILES:

CM - communications

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CMGOTH, CMPOTH, ERASE, MENU, TRMGET, TRMPUT, PROMPT

HISTORY:

designed by Dave Birnbaum September 20, 1978

programmed by Donna Morris September 26, 1979

PROGRAM NAME: CDISPLAY

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

If the current display is a two-space scatter plot, CDISPLAY changes the display to a two-space cluster plot, and vice versa. If the current display is a one-space macro plot, CDISPLAY changes the display to a one-space micro plot, and vice versa.

PROGRAM USAGE:

User types in 'CDISPLAY'.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Open the Communications file and the display files

IF (the display file contains a two-space display, i.e., the display code must be 1 or 2) THEN

Zero out the screen coordinate flag in the DV file header

Call CLSCAT to create the new display.

ELSEIF (the display file contains a one-space display, i.e., the display code must be 5 or 6)THEN

Zero out the screen coordinate flag in the DV file header

Call MICMAC to create new display.

ELSE

Tell user that the display file does not contain a one or two-space display.

ENDIF

Put up option list at user's terminal (MENU).

END

FILES:

CM - communications
DI - display information
DV - display value
PV - projection vector
S1 - scratch1

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

DIGDC, DVPHDR, ERASE, CLSCAT, CLOSFX, CMGCDS, CMGOTH, MENU, MICMAC, OPENFX, TRMPUT

HISTORY:

designed by DAB and SEH September 29, 1978
programmed by Donna Morris October 1,1979

PROGRAM NAME: CDSCHK

CATEGORY: UTILITY ROUTINE

PROGRAM DESCRIPTION:

CDSCHK checks the current data set names in the CM and DI file headers to see if they match. CDSCHK returns a true value if the names match and a false value if the names do not match.

PROGRAM USAGE:

```
LOGICAL CDSCHK
        .
        .
        .
IF(CDSCHK(FDCM,FDDI))
```

INPUT ARGUMENTS:

FDCM - INTEGER; the file descriptor of the CM file
FDDI - INTEGER; the file descriptor of the DI file

OUTPUT ARGUMENTS:

CDSCHK - LOGICAL; 'true' if the current data set names match; 'false' if the current data set names do not match.

FILES:

CM - Communication file
DI - Display Information file

REFERENCED BY:

Display routines (or routines that call display routines) that must check to make sure the current data set has not been changed.

SUBPROGRAMS REFERENCED:

CMGCDS, DIGNAM, EQUALA, INSPGM, INSRET

OLPARS Program Specifications
CDSCHK

HISTORY:

designed by Donna Morris January 21, 1981

programmed by Donna Morris January 21, 1981

PROGRAM NAME: CHARFL

CATEGORY: Utility Subroutine

PROGRAM DESCRIPTION:

CHARFL fills a buffer with a specified character from its current length to a desired length.

PROGRAM USAGE:

CALL CHARFL (BUFFER, CURLEN, DESLEN, FILCHR)

INPUT ARGUMENTS:

BUFFER - INTEGER; the buffer to be filled (starting at the index CURLEN+1)

CURLEN - current length of BUFFER

DESLEN - desired length of BUFFER

FILCHR - the character with which to fill BUFFER to the desired length

OUTPUT ARGUMENTS:

BUFFER - same as input but 'filled' to the desired length (DESLEN) with the fill character (FILCHR)

REFERENCED BY:

COMNOD

HISTORY:

designed by David J. Tipton October 31, 1980

programmed by David J. Tipton October 31, 1980

PROGRAM NAME: CHKEXS

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

CHKEXS is called by routines (commands) that cannot operate on a data set in excess measurement mode. If the dimensionality of the data set is greater than the maximum dimensionality allowed in OLPARS when not in excess measurement mode (currently 50), an error message is printed, the menu is redisplayed, and CHKEXS calls a routine to exit ('oexit').

PROGRAM USAGE:

CALL CHKEXS(FIDCM,NDIM,PGMNAM)

INPUT ARGUMENTS:

FIDCM - INTEGER; the file descriptor of the CM file

NDIM - INTEGER; the vector dimensionality of the data set being used by the calling routine

PGMNAM - INTEGER array (10); the name of the calling routine

ALGORITHM / NOTES:

This subroutine should be called before a main routine alters the contents of any display files, because the current menu is redisplayed.

FILES:

CM - Communications

REFERENCED BY:

All routines (commands) that cannot operate on a data set in excess measurement mode.

SUBPROGRAMS REFERENCED:

INSPGM, INSRET, MENU, OEXIT, TRMPUT

HISTORY:

designed by Donna Morris October 29, 1980

programmed by Donna Morris October 29, 1980

OLPARS Program Specifications
CIP

PROGRAM NAME: CIP

CATEGORY: OLPARS Executive (system dependent)

PROGRAM DESCRIPTION:

CIP (command input processor) partially implements the executive function of OLPARS. It accepts an OLPARS command, searches for the command name in the OLPARS program dictionary (for verification of program existence), and then creates a command file which will be used to initiate (spawn) the desired command.

PROGRAM USAGE:

Program must be "installed" with the name "...CIP" on RSX11M operating system and is initiated with the line "CIP [ggg,mmm]" where "ggg" and "mmm" are the group and member number of the OLPARS system directory.

ALGORITHM / NOTES:

The user is allowed to type in an initial substring if the OLPARS program desired. The substring must have enough characters that will make the spelling of the program name unique. Otherwise, the user may invoke a program that was really not wanted.

The progam dictionary is kept in alphabetical order. Searching for a program name always starts at the beginning of the dictionary. An added goodie to the search algorithm (for speed): If the first letter of the program dictionary name is greater than the first letter of the user's substring command request, the search for the user's substring has failed (i.e., the user has not requested a known OLPARS program).

```
REPEAT
   Put out OLPARS prompt (TEXT)
   REPEAT
      Wait for user to type in a command (TRMGET)
      REPEAT
         Read the OLPARS program dictionary and
```

```
            try to match the user's command with
            some initial substring of a command
            found in the program dictionary. (FILGET,EQUALS)

         UNTIL(command is validated or not found)

         IF(the command is known and is not 'BYEOLP')THEN

            Create command file used to invoke OLPARS
            command

         ENDIF

      UNTIL(the command is known)

   UNTIL('BYEOLP' is typed by user)

   END
```

USER INTERACTION:

The OLPARS user types in the name of the OLPARS program that is wanted for execution.

FILES:

OLPARS program dictionary
OLPARS command invocation command file (OLPCOM.CMD)

REFERENCED BY:

OLPARS user through HELOLP.CMD

SUBPROGRAMS REFERENCED:

CLOSE, CONCAT, EQUALS, EXIT, FILGET, FLUSHF, GETMCR, INDEX, LENGTH, OPEN$, TRMGET, TRMPUT, VALUES

SEE ALSO:

"HELOLP.CMD" and OLPARS programmer and maintenance manual

HISTORY:

designed by Steven Haehn October 25, 1978

programmed by Steven Haehn April 27, 1979

PROGRAM NAME: CLNODE

CATEGORY: Data Tree File Access Routine

PROGRAM DESCRIPTION:

Clnode creates a new lowest node in a data tree. If the node being created at this parent is a first lowest node, the parent node is updated and all ancestors' first lowest node pointers are checked and adjusted.

PROGRAM USAGE:

```
CALL CLNODE(FDTI,ET,PARENT,NODSLT,ONEKID,NODNAM,
            NVECS,VECPTR)
```

INPUT ARGUMENTS:

FDTI - INTEGER; file descriptor of the TI file

ET - INTEGER array (TABSIZ); the TI file entry table

PARENT - INTEGER; the pointer to the parent node of the lowest node being created

NODSLT - INTEGER; the pointer to the lowest node being created

ONEKID - LOGICAL; indicates whether the lowest node being created is the first lowest node at this parent or not

OUTPUT ARGUMENTS:

NODNAM - INTEGER array (NODLEN); the name of the node being created

NVECS - INTEGER; the number of vectors at the node being created

VECPTR - INTEGER; the TV file pointer to the vectors which lie at the new lowest node

ALGORITHM / NOTES:

Get the parent node counts and pointers from the TI file (TIGCAP).

IF (the lowest node being created is the first lowest node at this parent) THEN

Create the TI file entry for the new lowest node (TIPCAP, TIPNAM).

Fix the lowest nodes which should now include the new lowest node in their low node link pointers and low node back pointers (TIPCOP).

Update the parent's TI file entry (TIPCOP).

Adjust the first lowest node pointer of all ancestors to the new first lowest node (TIPCOP).

ELSE

Create the TI file entry for the new lowest node (ALNCPN).

ENDIF

RETURN

FILES:

TI - tree information file
instrumentation file

REFERENCED BY:

RESTRUCT

SUBPROGRAMS REFERENCED:

ALNCPN, INSPGM, INSRET, TIGCAP, TIGCOP, TIPCAP, TIPCOP, TIPNAM

SEE ALSO:

ALNCPN

HISTORY:

designed by Jill King February 13, 1981

programmed by Jill King February 13, 1981

OLPARS Program Specifications
CLOSBL

PROGRAM NAME: CLOSBL

CATEGORY: File I/O (system dependent)

PROGRAM DESCRIPTION:

CLOSBL closes the "system" file opened by OPENBL. The file to be closed is pointed to by a file descriptor. If the current file block (of the file being closed) has been changed, CLOSBL will write the block to the file before closing it. The logical unit portion of File Access and Control Table (FACT), pointed to by the given file descriptor, is made available for future use. CLOSBL has the ability to save or delete the "system" file from the operating system's filing structure, upon closure of the file.

PROGRAM USAGE:

CALL CLOSBL(FID,DISPOS)

INPUT ARGUMENTS:

FID - INTEGER; file descriptor of file to be closed
DISPOS - INTEGER; disposition of the file upon closing
(0 - save the file, 1 - delete file)

ALGORITHM / NOTES:

CLOSBL accesses the PRBUFF region of the FACT, thus array checking must be turned off (use /-CK switch to DEC F4P compiler) when compiling this program.

REFERENCED BY:

FMNCOV, GETVEC

SUBPROGRAMS REFERENCED:

CLOSE, INSPGM, OEXIT, TRMPUT, WRITEB

DIAGNOSTICS:

When CLOSBL gets an invalid file descriptor or has a problem writing out to the file before closing, it tells the user and halts the program.

SEE ALSO:

OPENBL

HISTORY:

designed by Steven Haehn June 16, 1979

programmed by Steven Haehn June 16, 1979

PROGRAM NAME: CLOSE

CATEGORY: OLPARS utility (system dependent)

PROGRAM DESCRIPTION:

CLOSE will close the file associated with logical unit number LUN, the first parameter to CLOSE, via the file control service macro CLOSE$.

If specified by the second parameter, SAVFLG, the file will also be deleted from the user file directory via the file control service routine .DLFNB, or printed to the local line printer (if one exists).

PROGRAM USAGE:

CALL CLOSE (LUN,SAVFLG)

INPUT ARGUMENTS:

LUN - INTEGER; logical unit number of file to be closed

SAVFLG - INTEGER; 0 = file to be saved after closing
1 = file to be deleted upon closing
2 = file to be closed, spooled for printing, and subsequently deleted

REFERENCED BY:

Level I subroutines and utility programs

HISTORY:

designed by Steve Haehn January 19, 1979

programmed by Dave Tipton March 13, 1979

PROGRAM NAME: CLOSFX

CATEGORY: Level II File Manipulation Subroutine

PROGRAM DESCRIPTION:

CLOSFX closes an OLPARS "fixed" file. The file descriptor (FID) specifies which file is being closed. The file descriptor was obtained via a call to OPENFX or CREAFX.

PROGRAM USAGE:

CALL CLOSFX(FID)

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the file to be closed

FILES:

All the OLPARS "fixed" files

REFERENCED BY:

OLPARS commands

SUBPROGRAMS REFERENCED:

OCLOSE

SEE ALSO:

OPENFX, CREAFX, DELEFX

HISTORY:

designed by Steve Haehn June 26, 1978

programmed by Donna Morris March 1,1979

PROGRAM NAME: CLOSTR

CATEGORY: Level II File Manipulation Subroutine

PROGRAM DESCRIPTION:

CLOSTR removes the File Access and Control Table (FACT) entries pointed to by the file descriptors, IFID and VFID. When only one of the files of a tree happens to be opened, then only one of the file descriptors should contain a legitimate value. The other file descriptor should be set to zero. (File descriptors are obtained through calls to OPENTR or CREATR.)

PROGRAM USAGE:

CALL CLOSTR(IFID,VFID)

INPUT ARGUMENTS:

IFID - INTEGER; file descriptor of the TI (or LI) file
VFID - INTEGER; file descriptor of the TV (or LV) file

SUBPROGRAMS REFERENCED:

OCLOSE

SEE ALSO:

OPENTR, DELETR, CREATR

HISTORY:

designed by Steve Haehn June 22, 1978

programmed by Donna Morris July 27,1979

PROGRAM NAME: CLSCAT

CATEGORY: Terminal Display Routine

PROGRAM DESCRIPTION:

CLSCAT produces a two-space display. It assumes that the vectors have already been projected onto a two-dimensional space and that the projected vectors are stored in the PV file. CLSCAT determines whether a scatter or cluster plot is called for and calls either SCAT or CLUS to produce the display.

PROGRAM USAGE:

```
CALL CLSCAT(FIDCM,FIDDI,FIDDV,FIDPV,CF)
```

INPUT ARGUMENTS:

FIDCM - INTEGER; the file descriptor of the CM file

FIDDI - INTEGER; the file descriptor of the DI file

FIDDV - INTEGER; the file descriptor of the DV file

FIDPV - INTEGER; the file descriptor of the PV file

CF - INTEGER; (change flag) if CF = 0, no change made in type of plot. If CF=1, scatter is changed to cluster or vice versa.

ALGORITHM / NOTES:

Erase screen (ERASE)

Get screen coordinate information from CM file (CMGSCN).

Let DC = the display code from DI header (DIGDC).

```
IF (DC = 0) THEN

    IF (total number of vectors in DI header is less than
    the two-space cutoff value) THEN

        Create a scatter plot (SCAT).

        Set DC = 2.

    ELSE

        Create a cluster plot (CLUS).

        Set DC  1.

    ENDIF

ELSEIF (CF = 0 and DC = 1) THEN

    Create a cluster plot.

ELSEIF (CF = 0 and DC = 2) THEN

    Create a scatter plot.

ELSEIF (CF = 1 and DC = 1) THEN

    Create a scatter plot.

    Set DC = 2.

ELSEIF (CF = 1 and DC = 2) THEN

    Create a cluster plot.

    Set DC = 1.

ELSE

    Print an error message.

ENDIF
```

Write DC out to DI,DV,PV file headers.

Draw rectangle on screen (RCTNGL).

Display list of class symbols in left-hand corner of screen with an '*' by those classes that are to be displayed (DISPCL).

Read Xmin, Xmax, Ymin, Ymax for displaying.

Print information on top and bottom of display rectangle.

RETURN

FILES:

DI - display information
DV - display value
PV - projection vector
CM - communications
Instrumentation files

REFERENCED BY:

All two-space projection routines.

SUBPROGRAMS REFERENCED:

CMGSCN, DISPCL, RCTNGL, DIGHOI, SCAT, CLUS, TEXT, ERASE, DIGMAX, DIGDC

HISTORY:

designed by Dave Birnbaum July 18, 1978

programmed by David Tipton August 10, 1979

PROGRAM NAME: CLUS

CATEGORY: Terminal Display Routine

PROGRAM DESCRIPTION:

After a data set or set of vectors at a logic node is projected onto a pair of projection vectors, CLUS is called to produce a cluster plot. CLUS computes the cluster screen coordinates of each vector, stores them in the DV file and creates a grid matrix of display symbols. The grid matrix is then displayed on the screen.

PROGRAM USAGE:

```
CALL CLUS(FIDDI,FIDDV,ISCREN)
```

INPUT ARGUMENTS:

FIDDI - INTEGER; the file descriptor of the DI file

FIDDV - INTEGER; the file descriptor of the DV file

ISCREN - INTEGER array (SC2NUM); the set of two-space screen coordinates

ALGORITHM / NOTES:

Get original min/max values, number of classes, and type of scaling from DI file (DIGMAX,DIGHDI).

Compute the X and Y ranges (depending on whether the type of scaling is SQUARE or RECTANGULAR) to be used in the calculation of screen coordinates.

Rewrite the current min/max values so that they reflect the type of scaling being used.

```
IF (screen coordinate flag from DV header is off)
THEN

    Obtain pointers to decision boundary points
    (DIGHDI)

    IF (there are any decision boundaries) THEN

        Recompute the screen coordinates for boundary
        points.

    ENDIF

    DO WHILE (there are more classes)

        Get next class from DI file (DIGENT).

        Compute scatter screen coordinates for
        the mean vector and store them in DV file.

        IF (class is ON) THEN

            DO WHILE (there are more vectors in
            the class)

                Get next vector from DV.

                Compute and store in DV the
                cluster screen coordinates for
                that vector.

                IF (both screen coordinates are
                >0) THEN

                    Place class symbol for that
                    vector in proper place in
                    grid matrix (if a different
                    symbol is already there, put
                    an asterisk in that place).

                ENDIF

            ENDDO
```

```
            ELSE

                DO WHILE (there are more vectors in the
                class)

                    Get next vector from DV.

                    Compute and store in DV the cluster
                    screen coordinates for that vector.

                ENDDO

            ENDIF

        ENDDO

    ELSE

        DO WHILE (there are more classes)

            Get next class from DI file (DIGENT).

            IF (class is ON) THEN

                DO WHILE (there are more vectors in
                the class)

                    Get next vector from DV.

                    IF (both screen coordinates are
                    >0) THEN

                        Place class symbol for that
                        vector in proper place in the
                        grid matrix.

                    ENDIF

                ENDDO

            ENDIF

        ENDDO

    ENDIF

    Display grid matrix on screen (GRIDIS).

    Set screen coordinates flag in DV header to 1
    (screen coordinates computed).

    RETURN
```

FILES:

DI - display information
DV - display value

REFERENCED BY:

CLSCAT

SUBPROGRAMS REFERENCED:

DIGMAX, DIGENT, GRIDIS, DVPHDR, DVGVEC

SEE ALSO:

SCAT

HISTORY:

designed by Dave Birnbaum July 17, 1978

programmed by David Tipton June 13,1979

PROGRAM NAME: CMDISP

CATEGORY: Terminal Display Routine

PROGRAM DESCRIPTION:

CMDISP displays a confusion matrix at the screen. If the matrix is too big to be displayed, it displays just the summary.

PROGRAM USAGE:

CALL CMDISP(FIDDI,FIDCM)

INPUT ARGUMENTS:

FIDDI - INTEGER; the file descriptor of the DI file

FIDCM - INTEGER; the file descriptor of the CM file

ALGORITHM / NOTES:

```
Erase screen.
IF (display code = 4, confusion matrix display) THEN
    IF (confusion matrix will fit on screen) THEN
        Display it.
        Display summary.
    ELSE
        Display summary.
        Inform user that the confusion matrix is too big
        to be displayed and that he should request a line
        printout of it.
    ENDIF
ELSE
    Print error message.
ENDIF
RETURN
```

REFERENCED BY:

PEPAIR, PECLDB, PENMV, RDISPLAY

SUBPROGRAMS REFERENCED:

CMGSCN, DIGCME, DIGCMH, ERASE, INSINT, INSPGM, INSRET, TRMPUT

HISTORY:

designed by John W. Tenney May 1, 1981

programmed by John W. Tenney May 3, 1981

PROGRAM NAME: CMGCDS

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

CMGCDS gets the current data set from the CM file. It returns the information of the current data set in COUNT (the FCD of the TI file, the FCD of the TV file, the entry slot number of the current node, and NDIM). Also returned are TREE (the treename), and NODE (the nodename).

PROGRAM USAGE:

```
CALL CMGCDS(FIDCM,COUNT,TREE,NODE)
```

INPUT ARGUMENTS:

FIDCM - INTEGER; the file descriptor of the CM file

OUTPUT ARGUMENTS:

```
COUNT - INTEGER (4); count information
 TREE - INTEGER (TRELEN); the treename
 NODE - INTEGER (NODLEN);the nodename
```

FILES:

CM - communications
Instrumentation files

REFERENCED BY:

COPY,S2EIGV,DDATATREE

SUBPROGRAMS REFERENCED:

FGET,PACKW,TRMPUT

HISTORY:

designed by Dave Birnbaum May 25,1978

programmed by Donna Morris May 4,1979

PROGRAM NAME: CMGDIR

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

CMGDIR retrieves OLPARS directory path string (place where all the OLPARS tasks and files are located) from the Communications (CM) file.

PROGRAM USAGE:

CALL CMGDIR (CMFD, DIRSTR, STRLEN)

INPUT ARGUMENTS:

CMFD - INTEGER; file descriptor of the CM file

OUTPUT ARGUMENTS:

DIRSTR - INTEGER ARRAY(FMNLEN); storage location for the retrieved directory path string.

STRLEN - INTEGER; actual length of the directory path string.

FILES:

CM - Communications

REFERENCED BY:

ANYTHING, GETOPT, GOPTNM, HELP, SETOPT

SUBPROGRAMS REFERENCED:

FGET, PACKW

SEE ALSO:

CMPDIR

HISTORY:

designed by Steven Haehn April 15, 1980

programmed by Steven Haehn April 18, 1980

PROGRAM NAME: CMGLOG

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

CMGLOG obtains the current logic information found in the communications (CM) file. It returns LogNam(the logic tree name), FcdLI(the LI file code), FcdLV(the LV file code), and IncLog(the number of incomplete logic nodes).

PROGRAM USAGE:

```
CALL CMGLOG(FIDCM,LOGNAM,FCDLI,FCDLV,INCLOG)
```

INPUT ARGUMENTS:

FIDCM - INTEGER; the file descriptor of the CM file.

OUTPUT ARGUMENTS:

LOGNAM - INTEGER array (TRELEN) ; the current logic name

FCDLI - INTEGER; the file code of the LI portion of the logic file

FCDLV - INTEGER; the file code of the LV portion of the logic file

INCLOG - INTEGER; the number of incomplete logic nodes

FILES:

CM - communications

SUBPROGRAMS REFERENCED:

FGET, INSCHR, INSINT, INSPGM, INSRET, OEXIT, PACKW, TRMPUT

HISTORY:

designed by Steve A. Haehn August 2, 1978

programmed by Jill M. King September 4, 1980

PROGRAM NAME: CMGOPT

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

CMGOPT gets the last major OLPARS option number and the last major OLPARS option name from the CM file

PROGRAM USAGE:

CALL CMGOPT(FIDCM,OPTNO,OPTNAM)

INPUT ARGUMENTS:

FIDCM - INTEGER; the file descriptor of the CM file

OUTPUT ARGUMENTS:

OPTNO - INTEGER; the last major OLPARS option number in the CM file

OPTNAM - INTEGER array (10); the last major OLPARS option name in the CM file

FILES:

CM - communications

REFERENCED BY:

MENU

SUBPROGRAMS REFERENCED:

FGET, PACKW, TRMPUT

SEE ALSO:

CMPOPT

HISTORY:

designed by Donna Morris October 31, 1979
(Happy Halloween)

programmed by Donna Morris October 31, 1979

PROGRAM NAME: CMGOTH

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

CMGOTH gets other information from the CM file. It is passed the file ID (FID) of the CM file and a code (ICODE) indicating what piece of information to return as the function value. This scheme operates according to the following table. CMGOTH is set equal to the required piece of information.

ICODE	CMGOTH SET EQUAL TO
1	Last major OLPARS option number
2	Prompt flag
3	One-space bin factor
4	Two-space cutoff value
5	Instrumentation flag
6	Instrumentation threshold value

PROGRAM USAGE:

```
INTEGER CMGOTH
      .
      .
      .
N = CMGOTH(FID,ICODE)
```

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the CM file
ICODE - INTEGER; the code

OUTPUT ARGUMENTS:

CMGOTH - INTEGER; the information requested by user

FILES:

CM - communications

SUBPROGRAMS REFERENCED:

FGET

SEE ALSO:

CMPOTH

HISTORY:

designed by David Birnbaum June 23, 1978

programmed by David Tipton June 6, 1979

modified by Steven Haehn Sept. 5, 1979

(I added the table information that gets the Instrumentation flag)

modified by Donna Morris November 28, 1979

(I added the instrumentation threshold value access capability)

PROGRAM NAME: CMGSCN

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

CMGSCN gets screen information from the CM file. It is passed the file ID (FID) of the CM file. It returns the screen information into the array ISCREN.

PROGRAM USAGE:

```
CALL CMGSCN(FID,ISCREN)
```

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the CM file

OUTPUT ARGUMENTS:

ISCREN - INTEGER array (NOSCRN); the screen coordinate information

FILES:

CM - communications
Instrumentaion Files

SUBPROGRAMS REFERENCED:

FGET

HISTORY:

designed by Dave Birnbaum June 23, 1978

programmed by David Tipton May 25, 1979

PROGRAM NAME: CMINIT

CATEGORY: Utility (system dependent)

PROGRAM DESCRIPTION:

CMINIT is used in conjunction with the RSX11M HELOLP.CMD command file to implement the "hello olpars" command on the VAX computer at PAR.

PROGRAM USAGE:

Within HELOLP.CMD -

```
ASN PAR1:'$OLPDIR'CMINIT.EXE=CMINIT
CMINIT '$OLPDIR'
```

INPUT ARGUMENTS:

'$OLPDIR' is the OLPARS directory path name obtained from an RSX command file

USER INTERACTION:

User is asked for the name of the terminal at which he/she is working.

ALGORITHM / NOTES:

CMINIT reads its RSX command line for the OLPARS directory path and places the path name into the 'CM' file. It also greets the user and asks her/him for the name of the terminal being used. After the user answers this question, the screen information for the user's terminal is placed in the user's 'CM' file.

FILES:

CM - Communications

TERMINALS.TXT - the names of the terminals known to OLPARS

<terminal name>.TXT - the screen info. for the given terminal

REFERENCED BY:

HELOLP.CMD, an RSX-11M command file

SUBPROGRAMS REFERENCED:

CLOSE, CLOSFX, CMPDIR, CMPSCN, CONCAT, FILGET, FLUSHF, GETMCR, INSSET, LENGTH, OEXIT, OPENFX, OPENS, PROMPT, TRMGET, TRMPUT

HISTORY:

designed by Steven Haehn April 15, 1980

programmed by Steven Haehn April 18, 1980

PROGRAM NAME: CMNCOV

CATEGORY: Numerical Computation Algorithm

PROGRAM DESCRIPTION:

CMNCOV computes the mean and covariance of all vectors in a class. These vectors are retrieved one at a time to save storage space, so instead of using the standard formulas for computing the mean and covariance arrays, a 'running' factor variation array is computed so that it can be updated after retrieving each vector. This routine has the capability of writing the vectors retrieved into a specific file. If the vectors are not to be written, the output file descriptor and the pointer to that file should be set to zero.

PROGRAM USAGE:

```
CALL CMNCOV(FDINTV,FDOUTV,NDIM,NVECS,PTREAD,PTRITE,
            MEAN,COV)
```

INPUT ARGUMENTS:

FDINTV - INTEGER; the file descriptor of the file from from which the vectors are to be read

FDOUTV - INTEGER; the file descriptor of the file to which the vectors are to be written

NDIM - INTEGER; the dimension of the data set

NVECS - INTEGER; the number of vectors used in the computations

PTREAD - INTEGER; the file pointer indicating the entry number of the first vector to be read from the input file

PTRITE - INTEGER; the file pointer indicating the entry number of the first vector to be written to the output file

OUTPUT ARGUMENTS:

MEAN - REAL array (Ndim); the 'running' mean vector

COV - REAL array ((Ndim*(Ndim+1))/2), the lower triangular portion of the 'running' covariance matrix, stored as a vector

ALGORITHM/NOTES:

Initialize the factor variation and mean arrays.

DO I=1,number of vectors in class

Retrieve the next vector from the input file (TVGVEC).

IF (the output file descriptor.ne.0) THEN

Place the vector into the output file (TVPVEC).

ENDIF

Update the 'running' factored variation arrays
to include the new vector.

ENDDO

Compute the final covariance array using the finished
mean vector.

RETURN

FILES:

TV - tree vector file
instrumentation file

REFERENCED BY:

RESTRUCT

SUBPROGRAMS REFERENCED:

INSFLT, INSPGM, INSRET, TVGVEC, TVPVEC

HISTORY:

designed by Jill King January 23, 1981

programmed by Jill King January 23, 1981

PROGRAM NAME: CMPCDS

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

CMPCDS stores information on the current data set in the CM file. It is passed the file ID of the CM file (FIDCM), the FCDs, the entry table slot number of the current node and the dimension of the data set (all in COUNT). It is also passed the treename/nodename pair in the variables TREE and NODE. It places all this information into the CM file.

PROGRAM USAGE:

```
CALL CMPCDS(FIDCM,COUNT,TREE,NODE)
```

INPUT ARGUMENTS:

FIDCM - INTEGER; the file descriptor of the CM file
COUNT - INTEGER array (4); count information on the current data set
TREE - INTEGER array; treename of the current data set
NODE - INTEGER array; nodename of the current data set

FILES:

CM - communications
Instrumentation files

REFERENCED BY:

SETDS, S2EIGV, DDATATREE

SUBPROGRAMS REFERENCED:

FPUT,ITREAL

HISTORY:

designed by Dave Birnbaum May 22, 1978

programmed by Donna Morris August 7,1979

OLPARS Program Specifications
CMPDIR

PROGRAM NAME: CMPDIR

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

CMPDIR should be used to place the OLPARS directory path string into the Communications (CM) file.

PROGRAM USAGE:

CALL CMPDIR (CMFD, DIRSTR, STRLEN)

INPUT ARGUMENTS:

CMFD - INTEGER; file descriptor of the CM file

DIRSTR - INTEGER ARRAY(FMNLEN); storage location of the directory path string to be placed in th CM file.

STRLEN - INTEGER; actual length of the directory path string.

FILES:

CM - Communications

REFERENCED BY:

HELOLP

SUBPROGRAMS REFERENCED:

FPUT, ITREAL

SEE ALSO:

CMGDIR

HISTORY:

designed by Steven Haehn April 15, 1980

programmed by Steven Haehn April 18, 1980

PROGRAM NAME: CMPLOG

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

CMPLOG writes current logic information into the communications (CM) file.

PROGRAM USAGE:

CALL CMPLOG(FID,LOGNAM,LIFCD,LVFCD,INCLOG)

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the CM file

LOGNAM - INTEGER array (TRELEN); the current logic name

LIFCD - INTEGER; the file code of the LI portion of the logic file

LVFCD - INTEGER; the file code of the LV portion of the logic file

INCLOG - INTEGER; the number of incomplete logic nodes

ALGORITHM / NOTES:

If you only want to write part of the current logic information out to CM, put a -1 in the integer arguments you do not want to write out, and a '$' in the character argument.

FILES:

CM - communications

REFERENCED BY:

NAMELOG, SETLOG

SUBPROGRAMS REFERENCED:

FPUT, INSFLT, INSPGM, INSRET, ITREAL, TRMPUT

HISTORY:

designed by Steven Haehn August 2, 1978

programmed by Mark Maginn July 17, 1980

OLPARS Program Specifications
CMPOPT

PROGRAM NAME: CMPOPT

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

CMPOPT writes the last major OLPARS option number and the last major OLPARS option name to the CM file

PROGRAM USAGE:

CALL CMPOPT(FIDCM,OPTNO,OPTNAM)

INPUT ARGUMENTS:

FIDCM - INTEGER; the file descriptor of the CM file

OPTNO - INTEGER; the last major OLPARS option number to write to the CM file

OPTNAM - INTEGER array (10); the last major OLPARS option name to write to the CM file

FILES:

CM - communications

REFERENCED BY:

SETOPT

SUBPROGRAMS REFERENCED:

FPUT, ITREAL, TRMPUT

SEE ALSO:

CMGOPT

HISTORY:

designed by Donna Morris October 31, 1979
(Happy Halloween)

programmed by Donna Morris October 31, 1979

PROGRAM NAME: CMPOTH

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

CMPOTH puts "other" information into the CM file. It is passed the file ID (FIDCM) of the CM file, a code (ICODE) and an integer value in INFO. CMPOTH inserts the value in INFO into the CM file according to the code ICODE.

ICODE	INFO INSERTED INTO
1	Last major OLPARS option number
2	Prompt flag
3	One-space bin factor
4	Two-space cutoff value
5	Instrumentation flag
6	Instrumentation threshold

PROGRAM USAGE:

```
CALL CMPOTH(FIDCM,ICODE,INFO)
```

INPUT ARGUMENTS:

```
FIDCM - INTEGER; the file descriptor of the CM file
ICODE - INTEGER; the code
 INFO - INTEGER; the information
```

FILES:

CM - communications

SUBPROGRAMS REFERENCED:

FPUT

SEE ALSO:

CMGOTH

OLPARS Program Specifications
CMPOTH

HISTORY:

designed by Dave Birnbaum June 23, 1978

programmed by Donna Morris August 7,1979

modified by Steven Haehn September 4, 1979

(I added the Instrumentation flag access capability)

modified by Donna Morris November 28, 1979
(I added the Instrumentation threshold value access capability)

PROGRAM NAME: CMPRT

CATEGORY: Line Printer Subroutine

PROGRAM DESCRIPTION:

CMPRT produces a printout of a confusion matrix that is stored in the DI file.

PROGRAM USAGE:

CALL CMPRT(FIDDI,MSG)

INPUT ARGUMENTS:

FIDDI - INTEGER; the file descriptor of the DI file

MSG - INTEGER array (MAXLIN); message to be printed.

FILES:

DI - display information

REFERENCED BY:

PEPAIR, PRTCM, PENMV, PECLDB

SUBPROGRAMS REFERENCED:

CLOSE, DIGCME, DIGCMH, FILPUT, INSPGM, INSRET, OPENS, PRINTR, TRMPUT

HISTORY:

designed by John W. Tenney May 13, 1981

programmed by John W. Tenney May 23, 1981

OLPARS Program Specifications
CMPSCN

PROGRAM NAME: CMPSCN

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

CMPSCN puts screen information into the CM file. It is passed the file ID (FIDCM) of CM and a set of screen coordinates in ISCREN. It places ISCREN into the CM file.

PROGRAM USAGE:

CALL CMPSCN(FIDCM,ISCREN)

INPUT ARGUMENTS:

FIDCM - INTEGER; the file descriptor of the CM file
ISCREN - INTEGER array (NOSCRN); the screen coordinates

FILES:

CM - communications

REFERENCED BY:

Only the CIP or HELLOLP

SUBPROGRAMS REFERENCED:

FPUT,ITREAL

HISTORY:

designed by Dave Birnbaum June 23, 1978

programmed by Donna Morris August 8,1979

PROGRAM NAME: CNT1SP

CATEGORY: Utility Subroutine

PROGRAM DESCRIPTION:

CNT1SP (count 1 space) determines the bins into which the vectors of a projected data set fall (i.e., create a bin count entry in the S1 file for each class to be displayed). This program is used in one space projection programs only.

PROGRAM USAGE:

```
CALL CNT1SP(FDDI,FDDV,FDS1,DIHDR,SCOORD,SCRN,MINMAX,
            NCLASS,NBNDRY,NBINS,DSPTYP,FLSTBN,CLSBCT)
```

INPUT ARGUMENTS:

FDDI - INTEGER; file descriptor of DI file

FDDV - INTEGER; file descriptor of DV file

FDS1 - INTEGER; file descriptor of S1 file

DIHDR - INTEGER ARRAY(DINITM); DI header

SCOORD - INTEGER; screen coordinates flag (indicates if screen coords. have been calculated for the given data set

SCRN - INTEGER ARRAY(19); OLPARS screen coordinates

MINMAX - INTEGER ARRAY(4); range of display values (only first to values are used in one-space proj.)

NCLASS - INTEGER; the number of classes in the given data set

NBNDRY - INTEGER; number of data partitions made on the current data set

NBINS - INTEGER; number of bins in the current one space display

DSPTYP - INTEGER; display type (MACRO or MICRO display code)

OUTPUT ARGUMENTS:

FLSTBN - INTEGER; fullest bin - maximum number of vectors any one class put in a single bin

WORKING SPACE ARGUMENTS:

CLSBCT - REAL ARRAY(MAXBIN); the bin count of a single class in the data set

ALGORITHM / NOTES:

```
IF (the bin counts need to be computed) THEN

    IF (any boundaries exist) THEN

        Recompute boundary screen coordinates

    ENDIF

    Compute the bin numbers for the vectors in the
    data class. (BINDET)

ELSE

    Retrieve fullest bin from S1 header (S1GHDR)

ENDIF

RETURN
```

FILES:

DI - display information
DV - display value
S1 - scratch 1

REFERENCED BY:

MACROP, MICROP

SUBPROGRAMS REFERENCED:

BINDET, DIGENT, DIPENT, DVGVEC, DVPVEC, INSCHR, INSINT, INSPGM, INSRET, OEXIT, SCATR, S1GHDR, S1PBC, S1PHDR, TRMPUT

DIAGNOSTICS:

If the first vector of each data class (in the DV file) is not the mean vector, CNT1SP will indicate this and force the program to exit. (If this happens the display files may be corrupted.)

HISTORY:

designed by Steven Haehn Oct. 5, 1980

programmed by Steven Haehn Oct. 16, 1980

OLPARS Program Specifications
COLAPS

PROGRAM NAME: COLAPS

CATEGORY: Support Routine

PROGRAM DESCRIPTION:

COLAPS uses the measurement vector in the PV file to eliminate from the covariance matrix rows and columns corresponding to unused measurements.

PROGRAM USAGE:

```
CALL COLAPS(OCOV,NCOV,MVEC,NDIM,NMUSED)
```

INPUT ARGUMENTS:

OCOV - REAL array ((NDIM * (NDIM + 1)) / 2); the lower triangular portion of the covariance matrix before rows and columns have been eliminated (the old covariance matrix)

MVEC - REAL array (NDIM); the measurement vector in the PV file. rows and columns corresponding to unused measurements will be eliminated from the covariance matrix.

NDIM - INTEGER; the actual vector dimensionality

NMUSED - INTEGER; the number of measurements used

OUTPUT ARGUMENTS:

NCOV - REAL array ((NMUSED * (NMUSED + 1)) / 2); the lower triangular portion of the covariance matrix after rows and columns corresponding to unused measurements have been eliminated (the new covariance matrix)

REFERENCED BY:

EIGNPV, S2FSHP

HISTORY:

designed by Donna Morris June 1,1979

programmed by Donna Morris June 1,1979

PROGRAM NAME: COMBIN

CATEGORY: Utilities Subroutine

PROGRAM DESCRIPTION:

COMBIN combines two covariance or scatter arrays and their two corresponding mean vectors, along with the total number of vectors used to compute the combined arrays. The formulas are given below:

LET MeanA,MeanB = the first and second mean vectors
MeanAB = the combined mean vector
A,B = the first and second covariance or scatter arrays
AB = the combined covariance or scatter array
NvecA,NvecB = the number of vectors used to compute the first and second arrays
NvecAB = the number of vectors used to compute the combined array

COMBINED MEAN VECTOR:

$$MeanAB = ((NvecA*MeanA) + (NvecB*MeanB)) / NvecAB$$

COMBINED COVARIANCE OR SCATTER ARRAY:

$$AB = [NvecA * (A(i,j) + MeanA(i) * MeanA(j)) + NvecB * (B(i,j) + MeanB(i) * MeanB(j))] / NvecAB - [MeanAB(i) * MeanAB(j)]$$

NOTE

If the number of vectors in both classes is equal to 0, then COMBIN exits without further ado, thus leaving the output arguments unchanged.

PROGRAM USAGE:

```
CALL COMBIN(MEANA,MEANB,A,B,NVECA,NVECB,NDIM,MEANAB,
            AB,NVECAB)
```

INPUT ARGUMENTS:

MEANA - REAL array (Ndim); the first mean vector

MEANB - REAL array (Ndim); the second mean vector

A - REAL array ((Ndim*(Ndim+1))/2); the lower triangular portion of the first covariance or scatter array, stored as a vector

B - REAL array ((Ndim*(Ndim+1))/2); the lower triangular portion of the second covariance or scatter array, stored as a vector

NVECA - INTEGER; the number of vectors used to compute the first covariance or scatter array

NVECB - INTEGER; the number of vectors used to compute the second covariance or scatter array

NDIM - INTEGER; the dimension of the vectors

OUTPUT ARGUMENTS:

MEANAB - REAL array (Ndim); the combined mean vector

AB - REAL array ((Ndim*(Ndim+1))/2); the lower triangular portion of the combined covariance or scatter array, stored as a vector

NVECAB - INTEGER; the number of vectors used to compute the combined covariance or scatter array

FILES:

instrumentation file

REFERENCED BY:

L1ASDG, L2ASDG, L2FSHP, S2FSHP

SUBPROGRAMS REFERENCED:

INSPGM, INSRET

HISTORY:

designed by Jill King October 22, 1980

programmed by Jill King October 22, 1980

PROGRAM NAME: COMNOD

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

COMNOD combines nodes in a data tree. It is used for combining nodes (the children nodes) under a common node (the parent node). The children combined must be lowest nodes, i.e., they have no children. COMNOD will not allow all nodes under a single node to be combined; DSUBSTRC must be used for this purpose.

PROGRAM USAGE:

User types in 'COMNOD'.

USER INTERACTION:

COMNOD does the following:

- Asks for the tree name.
- Asks for the parent node name.
- Prints a 'class select list' and asks for the class symbols of the nodes to combine.
- Asks for a new nodename (which may be the same name as one of the nodes to combine).
- Asks if the id's of the vectors of the nodes to combine should be resequenced.
- Asks for the starting id if resequencing is desired.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Request a tree name. Check if the dimensionality of the tree is within the maximum limit (excess measurement mode is illegal).

Request the parent node name. Check if the parent has at least three children and if at least two of its children are lowest nodes.

Create a list of names of the children that are lowest

nodes and a list of their entry table slot numbers.

Request from the list of names the class symbols of the nodes to combine (SELCLS).

Create a list of slot numbers of the nodes to combine.

Request a nodename for the new combined node.

Check it (LGLNOD) (UNIQND).

Request if resequencing is desired; if it is, get the starting id.

Create a new TV file with a dummy name (CREATR).

Initialize the header of the dummy tree vector file (TVPHDR).

Copy vectors from combined nodes into the dummy TV file (COPYVC).

From the list of entry slots of the combined nodes, determine the slot of the 'new' node. Set the vector pointer of the 'new' node to 1. Make all necessary adjustments to the linked lists (counts and pointers) of the TI file and to the mean vector and covariance matrix of the mean vector and covariance matrix of the 'new node' in order to make the TI entries of the other combined nodes 'go away' (NODCOM).

Set the new nodename in the TI entry of the new node (TIPNAM).

Copy rest of vectors from tree into the dummy TV file (COPYVC) making sure to adjust the necessary vector pointers in the TI entries of the nodes copied (TIPCOP).

Add entries of the combined nodes, excluding the entry of the new node, to the garbage (free) list of the TI file (GARBND).

Replace the original TV file with the dummy TV file (RENAMT).

Put up option list at user's terminal (MENU).

END

OLPARS Program Specifications
COMNOD

FILES:

TL - tree list
TI - tree information
TV - tree vector
CM - communication

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLOSTR, CMGOTH, COPYVC, CREATR, ERASE
GARBND, INSCHR, INSINI, INSINT, INSLOG
INSRET, MENU, NODCOM, OEXIT, OPENFX
RENAMT, TIGCAP, TIGCOP, TIPCOP, TIPNAM
TRMPUT, TVPHDR, UICMND

The following subroutines are referenced by 'UICMND':

CHARFL, CLOSFX, EQUALA, INSCHR, INSINT
INSLOG, INSPGM, INSRET, LENGA, LGLNOD
OPENFX, OPENTR, PROMPT, SELCLS, TIGCAP
TIGET, TIGHDR, TIGNAM, TISRCH, TLSRCH
TRMGET, TRMPUT, UNIQND

HISTORY:

designed by David Tipton August 19, 1980

programmed by David Tipton October 27, 1980

PROGRAM NAME: CONCAT

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

CONCAT moves 'Nchar' characters from 'Source' to 'Destin', starting at 'Sstart' and 'Dstart', respectively. 'Dlen' and 'Slen' are used for array bounds checking, so a user doesn't inadvertently transfer too many characters to 'Destin' (thus clobbering other program areas). Incorrect methods of calling CONCAT follow:

```
logical*1 a(25),b(25)

call concat(a(15),1,25,b,1,25,25)
```

or:

```
call concat(a(15),15,25,b,1,25,25)
```

PROGRAM USAGE:

CALL CONCAT(DESTIN,DSTART,DLEN,SOURCE,SSTART,SLEN,NCHAR)

INPUT ARGUMENTS:

DESTIN - LOGICAL *1 ARRAY; character string destination

DSTART - INTEGER; position in destination string to start placing characters

DLEN - INTEGER; maximum length of destination

SOURCE - LOGICAL *1 ARRAY; character string source

SSTART - INTEGER; position in source array to start retrieving characters

SLEN - INTEGER; maximum length of source string

NCHAR - INTEGER; number of characters to transfer

REFERENCED BY:

GEN,GETHST

OLPARS Program Specifications
CONCAT

HISTORY:

designed by Steve Haehn January 1,1979

programmed by Steve Haehn January 1,1979

PROGRAM NAME: COPYCV

CATEGORY: Data Tree File Access Routine

PROGRAM DESCRIPTION:

COPYCV performs a direct transference of the covariance matrix of one tree node into another tree node. The two FIDs may be of the same tree or different trees. The nodes within the tree are pointed to by the node entry numbers (ENTNOS, ENTNOR). NDIM is the dimension size of the data vectors.

NOTE

Since the whole covariance matrix, i.e. the upper triangular portion, will not be in memory at one time, this routine must coordinate the access to the covariance matrix within the TI file.

PROGRAM USAGE:

CALL COPYCV(FDTIS,FDTIR,ENTNOS,ENTNOR,NDIM)

INPUT ARGUMENTS:

FDTIS - INTEGER; the file descriptor of the source tree

FDTIR - INTEGER; the file descriptor of the object tree

ENTNOS - INTEGER; entry number of node in source tree

ENTNOR - INTEGER; entry number of node in object tree

NDIM - INTEGER; dimension of the data sets

FILES:

TI - tree information (files of the two nodes to be accessed)

REFERENCED BY:

APPEND

OLPARS Program Specifications
COPYCV

SUBPROGRAMS REFERENCED:

INSPGM, INSRET, TIGCOV, TIPCOV

SEE ALSO:

COPYMN

HISTORY:

designed by Steve Haehn June 7, 1978

programmed by Mark Maginn August 21, 1980

PROGRAM NAME: COPYMN

CATEGORY: Data Tree File Access Routine

PROGRAM DESCRIPTION:

COPYMN performs a direct transfer of the mean vector of one tree node into another tree node. The two FIDs may be of the same tree or different trees. The nodes within the trees are pointed to by the node entry numbers (ENTNOS, ENTNOR). NDIM is the dimension size of the data vectors.

PROGRAM USAGE:

```
CALL COPYMN(FDTIS,FDTIR,ENTNOS,ENTNOR,NDIM)
```

INPUT ARGUMENTS:

FDTIS - INTEGER; file descriptor of the source tree

FDTIR - INTEGER; file descriptor of the object tree

ENTNOS - INTEGER; entry number of the node in source tree

ENTNOR - INTEGER; entry number of the node in object tree

NDIM - INTEGER; dimension of the vectors in the data set

FILES:

TI - tree information (files of the two nodes to be accessed)

REFERENCED BY:

APPEND

SUBPROGRAMS REFERENCED:

INSINT, INSPGM, INSRET, TIGMN, TIPMN

SEE ALSO:

COPYCV

OLPARS Program Specifications
COPYMN

HISTORY:

designed by Steven Haehn June 7, 1978

programmed by Mark Maginn August 20, 1980

PROGRAM NAME: COPYVC

CATEGORY: Data Tree File Access Routine

PROGRAM DESCRIPTION:

COPYVC retrieves vectors from TV(1) and copies them into TV(2), and does this NUMVEC times in sequence. It then updates the TV(2) header by writing in the next open entry. If the number of vectors to copy are greater than the space available, no coping takes place, and the 'ABORT' flag is set equal to QUIT.

If the display symbol argument (DSPSYM) is non zero, that value is placed into the copied vector as the new display symbol.

If the vector identifiers of the copied vectors are to be resequenced, the resequencing flag (RENMBR) should be set to 'true', and a starting class identifier (SCLSID) should be given.

PROGRAM USAGE:

```
CALL COPYVC(FDTV1,FDTV2,VECPTR,NUMVEC,NDIM,DSPSYM,RENMBR,
            SCLSID,ABORT)
```

INPUT ARGUMENTS:

FDTV1 - INTEGER; the file descriptor of the first TV file(TV(1))

FDTV2 - INTEGER; the file descriptor of the second TV file(TV(2))

VECPTR - INTEGER; the entry position in TV(1) from which copying begins

NUMVEC - INTEGER; the number of vectors (entries) to be copied

NDIM - INTEGER; the dimension of the data set

DSPSYM - INTEGER; the display symbol of the vector

RENMBR - INTEGER; the resquence flag

SCLSID - INTEGER; the starting class id

OUTPUT ARGUMENTS:

ABORT - INTEGER; flag that gets set if there are too many vectors to copy

VECPTR - INTEGER; the entry position in TV(2) to which the vectors from TV(1) were copied

ALGORITHM / NOTES:

REMEMBER that the vector pointer (VECPTR) is modified upon program exit.

FILES:

TV - tree vector

REFERENCED BY:

APPEND, COMNOD, COPYND

SUBPROGRAMS REFERENCED:

INSINT, INSPGM, INSRET, TVGHDR, TVGVEC, TVPHDR, TVPVEC

HISTORY:

designed by David Birnbaum May 26, 1978

programmed by Mark Maginn August 17, 1980

PROGRAM NAME: CRANDTS

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

CRANDTS creates a random test set from the current data set. The current data set must be the senior node, and the data set must have only one level under the senior node (only lowest nodes). Using the current data set as the "old" tree, this routine creates a new tree by extracting a user - specified percentage of the vectors from the "old" tree. When all the specified vectors have been removed, the old tree name will be changed. The original tree will no longer exist.

PROGRAM USAGE:

User types 'CRANDTS'

USER INTERACTION:

The user enters a "new" tree name for the tree to be created. The user now enters a percentage of vectors to be extracted from the "old" (current data set) tree. Then user then enters a treename to be used to rename the current data set.

ALGORITHM / NOTES:

Obtain the current data set from communications file.

Check to make sure senior node is the current node, and the tree contains only lowest nodes beneath the senior node.

```
REPEAT
   Ask for a "new" treename.
   IF (treename is not legal) THEN
      Ask for another treename.
   ELSE
      Open tree list file
```

```
            WHILE (new name does not exist)

               Name exists.  Ask user whether or not to
               destroy the existing tree.

               IF ( not to destroy) THEN

                  Ask for another treename.

               ENDIF

            ENDWHILE

         ENDIF

      UNTIL

      REPEAT

         Ask for the percentage to extract.

         IF (percentage not in range) THEN

            Ask for another percentage.

         ENDIF

      UNTIL

      REPEAT

         Ask for a name to rename the current data set.

            WHILE (new name does not exist)

               Name exists.  Ask user whether or not to
               destroy the existing tree.

               IF ( not to destroy) THEN

                  Ask for another treename.

               ENDIF

            ENDWHILE

         ENDIF

      UNTIL
```

```
IF (new tree couldn't be created) THEN  (CREATR)

   Print message and go to BYEBYE.

ENDIF

Copy the tree information header and entry from the old
tree into the new tree.  (TIPHDR,TIPENT)

WHILE (not at end of the tree)

   Obtain number of vectors at a node, vector pointer
   to the TV file, and the slot number of the current
   node.(GNXTND)

   Calculate the nuber of vectors to be extracted.

   Initialize mean and covariance arrays.

   DO 1 to number of vectors to be extracted.

      Determine random number and vector to be
      extracted fro old tree.

      Obtain vector from "old" file and put in "new"
      file.  Increment new vector count.

      Update 'running' means, factored variations and
      covariances to include current vector.

      Replace vector just extracted with the last
      vector in the class.

   ENDDO

   Compute the final covariance array using the
   finished mean.

   Clean up both trees.
      o  subtact entries from "old" tree. (SUBMNC)
      o  write entries into "new" tree.
         (TIPCOV, TIPMN, TIPCAP, TIPNAM)
      IF ( first node to be written) THEN

         Write entry to senior node.

      ELSE
```

```
                Add up means and covariances at the senior
                node.  (ADUPCM)

             ENDIF

          ENDWHILE

          Write out new TV header. (TVPHDR)

          Close "old" TI file. (CLOSTR)

          IF (current tree name does not equal "rename) THEN
             Rename "old" tree to the name specified by
             the user.  (RENAMT)

             IF (tree couldn't be renamed ) THEN

                Print message and goto BYEBYE.

             ELSE

                Set the current data set to be the renamed
                tree. (CMPCDS)

             ENDIF

          ENDIF

       ENDIF

       Put up the option list at user's terminal (MENU).

       END
```

FILES:

```
            HS - History file
            CM - Communications file
            TL - Tree List file
       TI "old" - Tree Information file of old tree
       TI "new" - Tree Information file of new tree
       TV "old" - Tree Vector file of old tree
       TV "new" - Tree Vector file of new tree
       OPTIONS  - OLPARS option file
       Instrumention file
```

REFERENCED BY:

OLPARS user.

AD-A118 732 PAR TECHNOLOGY CORP NEW HARTFORD NY F/G 9/2
ON-LINE PATTERN ANALYSIS AND RECOGNITION SYSTEM. OLPARS VI. SOF--ETC(U)
JUN 82 S E HAEHN, D MORRIS
UNCLASSIFIED PAR-82-20 NL

2 OF 10
AD A
118732

SUBPROGRAMS REFERENCED:

ADUPCM, CLOSFX, CHKEXS, CLOSTR, CMGCDS, CMGOTH, CMPCDS,
CREATR, EQUALA, ERASE, GNXTND, INSFLT, INSINI, INSINT,
INSLOG, INSRET, LGLTRE, MENU, OEXIT, OPENFX, OPENTR,
PROMP, RAN, RENAMT, SUBMNC, TIGCAP, TIGCOP, TIGET,
TIGHDR, TIPCAP, TIPCOV, TIPET, TIPHDR, TIPMN, TIPNAM,
TLSRCH, TRMGET, TRMPUT, TVGVEC, TVPHDR, TVPVEC

HISTORY:

designed by Kermit Klingbail March 14, 1978

programmed by Mark Maginn July 28, 1981

PROGRAM NAME: CREAFX

CATEGORY: Level II File Manipulation Routine

PROGRAM DESCRIPTION:

CREAFX creates and opens an OLPARS "fixed" file. The fixed file is referenced by an integer variable which is the file code of the fixed file. The file code is used to determine a 2-character mnemonic for the file. The file descriptor (FID) of the opened file is returned.

PROGRAM USAGE:

```
IF (CREAFX(FID,FCD,HNE,LENE))

   LOGICAL FUNCTION CREAFX
```

INPUT ARGUMENTS:

FCD - INTEGER; the file code of the fixed file

HNE - INTEGER; the length (in elements) of the header of a fixed file

LENE - INTEGER; the length (in elements) of a logical entry of the fixed file

OUTPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the file

CREAFX - LOGICAL; if true, the fixed file was created successfully

ALGORITHM / NOTES:

The following list of file codes of the fixed files is known to this routine:

CM = 1	PV = 6
TL = 2	S1 = 7
LL = 3	SV = 8
DI = 4	SM = 9
DV = 5	HS = 10

FILES:

All the OLPARS "fixed" files.

SUBPROGRAMS REFERENCED:

OCREAT

DIAGNOSTICS:

If the fixed file cannot be created, CREAFX will return with a 'false' value.

SEE ALSO:

OPENFX, CLOSFX, DELEFX

HISTORY:

designed by Steve Haehn June 26, 1978

programmed by Donna Morris April 25,1979

PROGRAM NAME: CREATLOG

CATEGORY: Logic Design Command (subsidiary)

PROGRAM DESCRIPTION:

CREATLOG creates and stores one or two space group logic. After the logic is created a partial evaluation of the vectors present at the node is performed and the results are displayed to the user.

This command is utilized after one of the L1 or L2 commands have been called to create a projection and one or two boundaries have been drawn on the user's data display by DRAWBNDY.

PROGRAM USAGE:

User types 'CREATLOG'.

USER INTERACTION:

User is asked for the classes present in each of the defined data partitions, whether or not (s)he is satisfied with the logic results, and if the misclassified vectors should be printed out at the line printer.

ALGORITHM / NOTES:

Erase screen and home cursor.

Make sure projection has been created by a logic design routine.

Make sure the display code is a one- or two-space code.

Make sure data set has been partitioned (boundaries exist).

Make sure logic has not already been defined for current logic node.

Ask user for the classes present in each of the defined data partitions.

IF (we have a 1-space projection) THEN

    Create a 1-SPACE logic node (CRLOG1).

ELSE

Create a 2-SPACE logic node (CRLOG2).

ENDIF

Evaluate the vectors of the classes found at the current logic node (CREVAL).

Put up MENU.

END

FILES:

CM - communications	HS - history
DI - display information	OLPARS options
DV - display value	Instrumentation
PV - projection vector	MISCLASSifications
LI - logic information	TI - tree information
LV - logic value	TV - tree vector

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLOSFX, CMGCDS, CREVAL, CRLOG1, CRLOG2, ERASE, GTRGNI, GTRGNR, INSINI, INSINT, INSRET, KILLND, LIGCNM, LIGCOP, MENU, OEXIT, OPENS, OPENTR, TRMPUT, UICRLG

HISTORY:

designed by Kermit Klingbail September 28, 1978

programmed by Steven Haehn April 15, 1981

PROGRAM NAME: CREATR

CATEGORY: Level II File Manipulation Subroutine

PROGRAM DESCRIPTION:

CREATR creates "system" files that will make up an OLPARS tree. The calling program is required to pass NDIM (tree dimensionality), the NAME of the tree, the TYPE of tree (data or logic), and the ACCESS to the tree. The file descriptors of the newly created files will appear in IFID and/or VFID upon return from CREATR. IFID is the file descriptor for the tree/logic information file, and VFID is the file descriptor for the tree vector/logic value file. CREATR searches the tree/logic list file to see if the tree already exists. If it does, the data within the trees is clobbered when the tree files are opened.

Value of tree TYPE - 1 - data tree
2 - logic tree

Value of ACCESS - 1 - create information file only
2 - create vector/value file only
3 - create both files of tree

CREATR creates a new TL or LL file entry for the new tree.

PROGRAM USAGE:

```
LOGICAL CREATR
      .
      .
      .

IF (CREATR(IFID,VFID,NDIM,NAME,TYPE,ACCESS,DSNAME))
```

INPUT ARGUMENTS:

ACCESS - INTEGER; specifies which files to create

DSNAME - INTEGER array (TRELEN + NODLEN);
the design data set name (used
(only when creating a logic tree)

NAME - INTEGER array (TRELEN); new treename

NDIM - INTEGER; dimension of data set

TYPE - INTEGER; specifies whether to create a data
or logic tree

OUTPUT ARGUMENTS:

IFID - INTEGER; file descriptor of the created TI
or LI file

VFID - INTEGER; file descriptor of the created TV
or LV file

FILES:

TI - tree information
TV - tree vector
TL - tree list
LI - logic information
LV - logic value
LL - logic list

SUBPROGRAMS REFERENCED:

CLOSFX,LLGENT,LLGHDR,LLPENT,LLPHDR,LLSRCH,MAXO,
OCREAT,OPENFX,TLGENT,TLGHDR,TLPENT,TLPHDR,TLSRCH,
TRMPUT

DIAGNOSTICS:

If any portion of the tree cannot be created, CREATR
returns with a 'false' value.

SEE ALSO:

OPENTR, CLOSTR, DELETR

HISTORY:

designed by Steve Haehn June 22, 1978

programmed by Donna Morris July 26,1979

PROGRAM NAME: CREVAL

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

CREVAL (create logic's partial evaluation routine) is used by CREATLOG to evaluate the vectors of the classes that lie at the current logic node. CREVAL displays a confusion matrix (NOT stored in the Display information file) at the user's terminal.

PROGRAM USAGE:

```
INTEGER CREVAL
  .
  .
IF(CREVAL (FDCM,FDLI,FDTI,FDTV,ERRLUN,NREGN,KIDS,CLSPRS,
           NCLSPR,CLSNMS,NENTRY,ENTNOS,DSPTYP,CRNTLG,
           NBOUND) .EQ. QUIT)
```

INPUT ARGUMENTS:

FDCM - INTEGER; communications file descriptor

FDLI - INTEGER; logic information file descriptor

FDTI - INTEGER; tree information file descriptor

FDTV - INTEGER; tree vector file descriptor

ERRLUN - INTEGER; misclassification file's logical unit

NREGN - INTEGER; the number of regions (kids) that are to be created

KIDS - INTEGER array(NREGN); logic nodes to which vectors are to be classified (2 or 3 kids, depending on the number of boundaries drawn by user. These nodes are children of the current logic node.

CLSPRS - INTEGER array(MAXCLS,NREGN); indices to classes present at the current logic node

NCLSPR - INTEGER array(NREGN); number of classes present at each of the current node's children

CLSNMS - INTEGER array(NODLEN,MAXCLS); names of the classes in the design data set

NENTRY - INTEGER; the number of classes present at the current logic node

ENTNOS - INTEGER array(NENTRY); entry numbers of the classes present at the current logic node

DSPTYP - INTEGER; display type (1-space, 2-space)

CRNTLG - INTEGER; current logic node number

NBOUND - INTEGER; number of user created boundaries

OUTPUT ARGUMENTS:

CREVAL - INTEGER; tells calling program to QUIT or continue

FILES:

CM - communications MISCLASS(ifications)
LI - logic information
TI - tree information
TV - tree vector

REFERENCED BY:

CREATLOG

SUBPROGRAMS REFERENCED:

CLOSE, CMGCDS, CMGLOG, CMGOTH, CMGSCN, CMPLOG, EQUALA, ERASE, EV1SP, EV2SP, FILPUT, INSCHR, INSINT, INSPGM, INSRET, LIPCOP, PRINTR, PROMPT, TIGCOP, TIGNAM, TRMGET, TRMPUT, TVGVEC, TVPLOG, WRTNOW

HISTORY:

designed by Steven Haehn April 6, 1981

programmed by Steven Haehn April 12, 1981

OLPARS Program Specifications
CRLOG1

PROGRAM NAME: CRLOG1

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

CRLOG1 creates decision logic at the current logic node and two to three logic nodes below the current logic node for one-space projections (between-group logic).

PROGRAM USAGE:

```
CALL CRLOG1(FDLI, FDLV, FDDI, FDDV, FDPV, CLSPRS, NCLSPR,
            NREGN, ENTNOS, NENTRY, CRNTLG, LVPTR)
```

INPUT ARGUMENTS:

FDLI - INTEGER; file descriptor for LI file

FDLV - INTEGER; file descriptor for LV file

FDDI - INTEGER; file descriptor for DI file

FDDV - INTEGER; file descriptor for DV file

FDPV - INTEGER; file descriptor for PV file

CLSPRS - INTEGER array(MAXCLS,MAXRGN);
indices into the table of names representing the classes that are supposed to be present at the new nodes being created by this routine.

NCLSPR - INTEGER array(MAXRGN);
the number of classes present in (CLSPRS) each of the new nodes being created

NREGN - INTEGER; the number of regions (new nodes to be created) created by user

ENTNOS - INTEGER array(MAXCLS);
the data tree entry numbers of the classes present at the current logic node

NENTRY - INTEGER; the number of classes present at (ENTNOS) the current logic node

CRNTLG - INTEGER; the node number of the current logic node

OUTPUT ARGUMENTS:

LVPTR - INTEGER; logic value pointer - points to first entry of the newly created 1-space group logic

FILES:

LI - logic information
LV - logic value
DI - display information
DV - display value

REFERENCED BY:

CREATLOG

SUBPROGRAMS REFERENCED:

CRLV1, EVALBM, GNLIAE, INSPGM, INSRET, LIGCNM, LIGC
LIGENT, LIGSTR, LIPENT

HISTORY:

designed by Kermit Klingbail October 3, 1978

programmed by Steven Haehn March 11, 1981

PROGRAM NAME: CRLOG2

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

CRLOG2 creates decision logic at the current logic node and two to three logic nodes below the current logic node for two-space projections (between-group logic).

PROGRAM USAGE:

```
CALL CRLOG2(FDLI, FDLV, FDDI, FDDV, FDPV, CLSPRS, NCLSPR,
            NREGN, ENTNOS, NENTRY, CRNTLG, LVPTR)
```

INPUT ARGUMENTS:

FDLI - INTEGER; file descriptor for LI file

FDLV - INTEGER; file descriptor for LV file

FDDI - INTEGER; file descriptor for DI file

FDDV - INTEGER; file descriptor for DV file

FDPV - INTEGER; file descriptor for PV file

CLSPRS - INTEGER array(MAXCLS,MAXRGN);
indices into the table of names representing the classes that are supposed to be present at the new nodes being created by this routine.

NCLSPR - INTEGER array(MAXRGN);
the number of classes present in (CLSPRS) each of the new nodes being created

NREGN - INTEGER; the number of regions (new nodes to be created) created by user

ENTNOS - INTEGER array(MAXCLS);
the data tree entry numbers of the classes present at the current logic node

NENTRY - INTEGER; the number of classes present at (ENTNOS) the current logic node

CRNTLG - INTEGER; the node number of the current logic node

OUTPUT ARGUMENTS:

LVPTR - INTEGER; logic value pointer - points to first entry of the newly created 2-space group logic

FILES:

LI - logic information
LV - logic value
DI - display information
DV - display value

REFERENCED BY:

CREATLOG

SUBPROGRAMS REFERENCED:

CRLV2, EVALBM, GNLIAE, INSPGM, INSRET, LIGCNM, LIGCOP, LIGENT, LIGSTR, LIPENT

HISTORY:

designed by Kermit Klingbail October 3, 1978

programmed by Steven Haehn March 11, 1981

PROGRAM NAME: CRLV1

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

CRLV1 constructs the LV file entries which comprise the logic block for one-space group (projection) logic. The data (coordinates of boundary lines and the projection vector) for doing this is found in the DI, DV, and PV files which resulted from a one-space display upon which the user has drawn one or two boundaries.

PROGRAM USAGE:

```
CALL CRLV2 (FDLV, FDDI, FDDV, FDPV, RGTRGN, LFTRGN,
           MIDRGN, LVETSZ, LVPTR)
```

INPUT ARGUMENTS:

FDLV - INTEGER; file descriptor of logic value file

FDDI - INTEGER; display information file descriptor

FDDV - INTEGER; file descriptor of display value file

FDPV - INTEGER; projection vector file descriptor

RGTRGN - INTEGER; logic node associated with right region of user partitioned 1-space display

LFTRGN - INTEGER; logic node associated with left region of user partitioned 1-space display

MIDRGN - INTEGER; logic node associated with middle region of user partitioned 1-space display

LVETSZ - INTEGER; logic value file's entry size

OUTPUT VARIABLES:

LVPTR - INTEGER; pointer to first entry of 1-space group logic block

ALGORITHM / NOTES:

Obtain 1-space boundaries from DV file (DVGVEC)

Obtain projection vector from PV file (PVGENT)

Place group logic block header into region (PTRGNR)

Place the projection vector into the group logic region (PTRGNR)

FILES:

LV - logic value
DI - display information
DV - display value
PV - projection vector

REFERENCED BY:

CRLOG1

SUBPROGRAMS REFERENCED:

DIGHDI, DVGVEC, INSPGM, INSRET, PTRGNR, PVGENT

HISTORY:

designed by Kermit KlingBail September 29, 1978

programmed by Steven Haehn March 23, 1981

PROGRAM NAME: CRLV2

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

CRLV2 constructs the LV file entries which comprise the logic block for two-space group (projection) logic. The data (coordinates of boundary line segments and the projection vectors) for doing this is found in the DI, DV, and PV files which resulted from a two-space display upon which the user has drawn one or two boundaries.

PROGRAM USAGE:

```
CALL CRLV2 (FDLV, FDDI, FDDV, FDPV, XSRLND, CR1LND,
            CR2LND, LVETSZ, LVPTR)
```

INPUT ARGUMENTS:

FDLV - INTEGER; file descriptor of logic value file

FDDI - INTEGER; display information file descriptor

FDDV - INTEGER; file descriptor of display value file

FDPV - INTEGER; projection vector file descriptor

XSRLND - INTEGER; logic node associated with excess region of user partitioned 2-space display

CR1LND - INTEGER; logic node associated with first convex region of user partitioned 2-space display

CR2LND - INTEGER; logic node associated with second convex region of user partitioned 2-space display

LVETSZ - INTEGER; logic value file's entry size

OUTPUT VARIABLES:

LVPTR - INTEGER; pointer to first entry of 2-space group logic block

ALGORITHM / NOTES:

Obtain 2-space boundary segments and convex region points from DV file (DVGVEC)

Obtain projection vectors from PV file (PVGENT)

Obtain first entry of group logic region (GNLVAE) (space for the group logic block header)

For each line segment of the boundary(ies), compute a discriminant vecor and threshold value (DSCVTH) and place the discriminant vector into the group logic region, saving the threshold value (PTRGNR).

Place the saved discriminant thresholds into the group logic region (PTRGNR)

Finally, place group logic block header into region (PTRGNI)

FILES:

LV - logic value
DI - display information
DV - display value
PV - projection vector

REFERENCED BY:

CRLOG2, PAIRMOD

SUBPROGRAMS REFERENCED:

DIGHDI, DSCVTH, DVGVEC, GNLVAE, INSPGM, INSRET, PTRGNI, PTRGNR, PVGENT

HISTORY:

designed by Kermit KlingBail September 29, 1978

programmed by Steven Haehn March 23, 1981

PROGRAM NAME: CRPROJ

CATEGORY: Display File Access Routine

PROGRAM DESCRIPTION:

CRPROJ stores in the DV file the selected coordinate(s) for each vector from a data set and sets up the DI file entry for each class in the data set. CRPROJ is passed the entry table slot number of the current data node in the current data set, and it projects the vectors in the lowest nodes underneath that node.

PROGRAM USAGE:

```
CALL CRPROJ(FIDTI,FIDTV,FIDDI,FIDDV,ENTABL,SLTNO,COORDS)
```

INPUT ARGUMENTS:

FIDTI - INTEGER; the file descriptor of the TI file

FIDTV - INTEGER; the file descriptor of the TV file

FIDDI - INTEGER; the file descriptor of the DI file

FIDDV - INTEGER; the file descriptor of the DV file

ENTABL - INTEGER array (100); the entry table of the data tree

SLTNO - INTEGER; the entry table slot number of the current data node

COORDS - INTEGER array (2); the coordinates to project on. COORDS (2) = 0 means this is a one-space projection.

ALGORITHM / NOTES:

Get the entry table slot numbers of the set of lowest nodes (GLONOD).

Call LRPROJ to project the vectors. Be sure to set ALLVEC = .TRUE.

RETURN

FILES:

TI - tree information
TV - tree vector
DI - display information
DV - display value

REFERENCED BY:

S1CRDV, S2CRDV

SUBPROGRAMS REFERENCED:

GLONOD, LRPROJ

HISTORY:

designed by Dave Birnbaum July 7, 1978

programmed by Donna Morris August 12, 1980

OLPARS Program Specifications
CSCALE

PROGRAM NAME: CSCALE

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

CSCALE changes the 'type of scaling' element in the DI file header for one and two-space displays. For one-space displays, CSCALE alters the meaning of the area under a histogram of a micro display. (Under the count option, each histogram column is proportional in area to the number of vectors in each displayed class which falls into a given bin; therefore, the total area under an entire class histogram is indicative of the number of vectors in each class. The probability option produces histogram columns representing the percentage of each class within a given bin; therefore, the total areas under all class histograms are equal to one another (that is, totals to 100 percent).) When the count option is in effect, CSCALE changes the display to feature the probability option, and vice versa. For two-space displays, CSCALE changes the scaling of a cluster or scatter plot from rectangular to square and vice versa. When square scaling is used, the x and y screen coordinates are scaled to the largest of the two ranges (X and Y). When rectangular scaling is used, the x screen coordinates are scaled to the X range, and the y screen coordinates are scaled to the Y range. Thus, square scaling uses the same scale for both axes, and rectangular scaling uses two different scales for the X and Y axes.

PROGRAM USAGE:

User types in 'CSCALE'.

ALGORITHM / NOTES:

```
Open the CM file and the display files
if(the display code = macro or micro)then
   if(type of scaling element = probabilities)then
      set type of scaling element to = counts
   else
      set type of scaling element to = probabilities
   endif
```

```
      zero out the screen coordinate flag in the DV file
      header.  write out the type of scaling element to
      the DI file header.

      display a one-space plot

   elseif(the display code = scatter or cluster)then

      if(we are in global mode (zooming is not in effect))

         set the original min/max values to be the
         current min/max values

         if(the type of scaling element = square)then

            set the type of scaling to = rectangular

         else

            set the type of scaling to = square

         endif

      elseif(we are in zoom mode)then

         if(type of scaling = rectangular)then

            set the type of scaling to = square

            change the current rectangular zoom
            min/max values to be square scaled
            coordinates

         elseif(type of scaling = square)then

            give the user a message that this
            option is not available and ask her
            if she wishes to continue.  (see note below)

            if(the user does not wish to continue)

               go to BYEBYE

            endif

         endif

      endif
```

```
            if(the type of scaling was changed)then

               zero out the screen coordinate flag in the
               DV file header.  write the type of scaling
               element to the DI file header.

            endif

            put up a two-space plot

         else

            notify the user that the display is not a one
            or two-space display

         endif

BYEBYE   display the menu

         over and out

         NOTE:  we cannot change from square scaling to
                rectangular scaling in zoom mode, as we
                do not have enough information stored
                in the DI file header.  we do not know
                the value of the original zoom coordinates
                or the difference that was added to the
                X or Y maximum value when we changed from
                rectangular to square scaling at some
                previous point in time.  to incorporate
                this option, it would be necessary to
                add elements to the DI file header).

FILES:

         CM - Communications
         DI - Display Information
         DV - Display Vector
         PV - Projection Vector
         S1 - Scratch 1
         HS - History (indirectly used)
         OP - Option (indirectly used)
         instru.dat (record i/o instrumentation file)
```

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLOSFX, CLSCAT, DIGHDI, DIPHDI, DVPHDR, INSINI, INSRET, MICMAC, OEXIT, OPENFX, TRMPUT

HISTORY:

designed by Steve Haehn October 2, 1978

programmed by Donna Morris November 4, 1980

PROGRAM NAME: DBNDY

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

DBNDY deletes all existing display boundaries from a one- or two-space display file. It then redisplays the projection with the menu.

PROGRAM USAGE:

User types in 'DBNDY'.

ALGORITHM / NOTES:

Obtain the display code from the DI file.

IF (the display code shows that the file does not contain information for a one- or two-space display)

    Tell user "Not a one- or two-space display"

ELSE

    Remove the boundary from the DI file.
    Redisplay the projection.

ENDIF

Put up MENU

FILES:

CM - communications
DI - display information
DV - display value
PV - projection vector
S1 - scratch 1
Instrumentation file

REFERENCED BY:

OLPARS user.

SUBPROGRAMS REFERENCED:

CLOSFX, CLSCAT, CMGOTH, DIGHDI, DIPHDI, INSINI, INSINT, INSRET, MENU, MICMAC, OEXIT, OPENFX, TRMPUT

SEE ALSO:

DRAWBNDY

HISTORY:

designed by Steven Haehn August 9, 1978

programmed by Mark Maginn July 10, 1980

PROGRAM NAME: DCRIM

CATEGORY: Numerical Computation Algorithm

PROGRAM DESCRIPTION:

DCRIM computes the Fisher discriminant and an orthogonal discriminant for two groups of classes from the current data set. These vectors are defined as:

LET normc = the normalizing constant
A = the sum of the within group covariance or scatter matrices, inverted
Mndif = the difference between the two group mean vectors

FISHER discriminant formula:

$$fishdr = normc * (A * Mndif)$$

ORTHOG discriminant formula:

$$orthdr = normc*\left(A - \frac{[Mndif * A^{2} * Mndif]}{[Mndif * A^{3} * Mndif]} * A^{2}\right)*Mndif$$

PROGRAM USAGE:

```
CALL DCRIM(CSOPT,NCLASS,SLOTN1,SLOTN2,FDPV,FDTI,FDTV,ET,
           ALLVEC,NDIM,NODNUM,FISHDR,ORTHDR)
```

INPUT ARGUMENTS:

CSOPT - INTEGER; the scatter or covariance matrix option
CSOPT = 0 means covariance option
CSOPT = 1 means scatter option

NCLASS - INTEGER array(2); the number of classes in each group

SLOTN1 - INTEGER array(MAXCLS); the entry table slot numbers corresponding to the classes in group 1

SLOTN2 - INTEGER array(MAXCLS); the entry table slot numbers corresponding to the classes in group 2

FDPV - INTEGER; the file descriptor of the PV file

FDTI - INTEGER; the file descriptor of the TI file

FDTV - INTEGER; the file descriptor of the TV file

ET - INTEGER array(TABSIZ); the TI file entry table

ALLVEC - LOGICAL; TRUE means all vectors used in the computation; FALSE means only those vectors which lie at the specified logic node are to be used

NDIM - INTEGER; the dimension of the current data set

NODNUM - INTEGER; the logic node number

NOTE

For structure analysis, ALLVEC must be set to TRUE and NODNUM will be disregarded.

OUTPUT ARGUMENTS:

FISHDR - REAL array(Ndim); the Fisher discriminant

ORTHDR - REAL array(Ndim); the orthogonal discriminant

OLPARS Program Specifications
DCRIM

ALGORITHM / NOTES:

Compute the sum of the within group covariance or scatter matrices and place in matrix A (DCRIM1).

Invert the matrix computed (INVERT, DITHER).

Compute the normalized Fisher and normalized orthogonal discriminants (DCRIM2).

IF (measurements have been eliminated) THEN

Expand the vectors back to their original dimension

ENDIF

RETURN

REFERENCED BY:

L2FSHP, S2FSHP

SUBPROGRAMS REFERENCED:

DCRIM1, DCRIM2, DITHER, INSFLT, INSINT, INSPGM, INSRET, INVERT, PVGENT

HISTORY:

designed by David Birnbaum August 10, 1978

programmed by Jill King November 14, 1980

PROGRAM NAME: DCRIM1

CATEGORY: Numerical Computation Algorithm

PROGRAM DESCRIPTION:

DCRIM1 computes the combined covariance or scatter array and the two corresponding group mean vectors from two groups of classes from the current data set.

PROGRAM USAGE:

```
CALL DCRIM1(CSOPT,NCLASS,SLOTN1,SLOTN2,FDPV,FDTI,FDTV,ET,
            ALLVEC,NDIM,NODNUM,NEWDIM,A,MNDIF)
```

INPUT ARGUMENTS:

CSOPT - INTEGER; the scatter or covariance matrix option
CSOPT = 0 means covariance option
CSOPT = 1 means scatter option

NCLASS - INTEGER array(2); the number of classes in each group

SLOTN1 - INTEGER array(MAXCLS); the entry table slot numbers corresponding to the classes in group 1

SLOTN2 - INTEGER array(MAXCLS); the entry table slot numbers corresponding to the classes in group 2

FDPV - INTEGER; the file descriptor of the PV file

FDTI - INTEGER; the file descriptor of the TI file

FDTV - INTEGER; the file descriptor of the TV file

ET - INTEGER array(TABSIZ);entry table of the TI file

ALLVEC - LOGICAL; TRUE means all vectors used in the computations; FALSE means only those following at the particular logic node

NDIM - INTEGER; the dimension of the current data set

NODNUM - INTEGER; the logic node number

OUTPUT ARGUMENTS:

NEWDIM - INTEGER; the dimension of the vectors after measurements have been eliminated

A - REAL array ((Newdim*(Newdim+1))/2); the sum of the within group covariance or scatter matrices, stored as a vector

MNDIF - REAL array (Newdim); the difference between the two individual group mean vectors

ALGORITHM / NOTES:

```
Compute or retrieve the mean and covariance arrays for
the first class of the first group (LOGMNC,TIGMN,TIGCOV)

DO groupno=1,2

   WHILE (there are more classes left in the group)

      Compute or retrieve the covariance and mean array
      for the present class (LOGMNC,TIGMN,TIGCOV)

      Combine the 'running' covariance and mean arrays
      with the arrays for the present class  (COMBIN)

   ENDWHILE

   Save the individual group mean array

ENDDO

IF (the scatter matrix option has been choosen) THEN

   Compute the scatter matrix from the combined
   covariance matrix

ENDIF

IF (measurements are to be eliminated) THEN

   Eliminate measurements in the combined covariance
   or scatter array (COLAPS)

   Eliminate measurements in each individual group mean
   vector (COLAPS)

ENDIF
```

Compute the difference between the two group mean vectors

RETURN

FILES:

PV - projection vector file
TI - tree information file
TV - tree vector file
instrumentation file

REFERENCED BY:

L2FSHP, S2FSHP

SUBPROGRAMS REFERENCED:

COLAPS, COMBIN, INSFLT, INSINT, INSPGM, INSRET, LOGMNC, PVGENT, PVGHDR, TIGCOP, TIGCOV, TIGMN

HISTORY:

designed by David Birnbaum August 10, 1980

programmed by Jill King November 11, 1980

PROGRAM NAME: DCRIM2

CATEGORY: Numerical Computation Algorithm

PROGRAM DESCRIPTION:

DCRIM2 computes the normalized Fisher discriminant and a normalized orthogonal discriminant for two groups of classes from the current data set. These vectors are defined below:

LET normc = the normalizing constant
A = the sum of the within group covariance or scatter matrices, inverted
Mndif = the difference between the two group mean vectors

FISHER discriminant formula:

$$fishdr = normc * (A * Mndif)$$

ORTHOG discriminant formula:

$$orthdr = normc*\left(A - \frac{[Mndif * A^{2} * Mndif]}{[Mndif * A^{3} * Mndif]} * A^{2}\right)*Mndif$$

PROGRAM USAGE:

CALL DCRIM2(NDIM,A,MNDIF,FISHDR,ORTHDR)

INPUT ARGUMENTS:

NDIM - INTEGER; the dimension of the vectors. If measurements have been eliminated, Ndim represents the dimension of the collapsed vectors.

A - REAL array ((Ndim*(Ndim+1))/2); the lower triangular portion of the combined and inverted covariance or scatter matrix

MNDIF - REAL array (Ndim); the difference in the two individual mean vectors from each group of classes

OUTPUT ARGUMENTS:

FISHDR - REAL array (Ndim); the fisher discriminant

ORTHDR - REAL array (Ndim); the orthogonal discriminant

FILES:

instrumentation file

REFERENCED BY:

S2FSHP

SUBPROGRAMS REFERENCED:

IDX, INSFLT, INSPGM, INSRET, $SQRT

HISTORY:

designed by David Birnbaum August 10, 1978

programmed by Jill King November 11, 1980

PROGRAM NAME: DDATANOD

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

DDATANOD deletes one or more lowest nodes from a data set. If the node to be deleted has only one sibling, then the sibling replaces the parent node in the tree structure. To delete a tree with only one lowest node, DDATATREE must be used.

PROGRAM USAGE:

User types in 'DDATANOD'.

USER INTERACTION:

User is asked for the class symbol of a node (in the current tree) to be deleted.

ALGORITHM / NOTES:

```
Erase screen and home cursor (ERASE).

REPEAT

   Retrieve the TI file entry table (TIGET).

   Retrieve the list of lowest nodes from the TI file
   (GLONOD).

   IF (THE CURRENT DATA SET HAS LESS THAN TWO LOWEST
       NODES)

      BREAK

   ELSE

      REPEAT

         Prompt user for a node name to be deleted
         (SELCLS).

         IF (user has selected more than one class) THEN

            Print error message and prompt again
            (TRMPUT).

         ENDIF
```

```
            UNTIL (user enters a correct node name)

            Rearrange counts and pointers so as to remove the
            node and make changes to means and covariances to
            all nodes above deleted node (REARR).

         ENDIF

      UNTIL (user indicates (s)he is done, or there are no
                lowest nodes to delete)

      END
```

NOTE:

The vectors of the node that is deleted are not actually removed from the TV file. However, since there is no pointer that points to them remaining in the TI file, these vectors are effectively deleted.

FILES:

CM - communications file
HS - history file
TV - tree vector file
TI - tree information file
instrumentation file
option file

REFERENCED BY:

OLPARS user.

SUBPROGRAMS REFERENCED:

CMGCDS, CMGOTH, ERASE, GLONOD, INSINI, INSRET,
MENU, OEXIT, OPENFX, OPENTR, REARR, SELCLS,
TIGET, TRMPUT

HISTORY:

designed by David Birnbaum July 28, 1978

programmed by Jill King May 13, 1981

OLPARS Program Specifications
DDATATREE

PROGRAM NAME: DDATATREE

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

DDATATREE deletes a datatree from the user's directory and the treelist file.

PROGRAM USAGE:

User types in 'DDATATREE'.

USER INTERACTION:

Prompts for the treename.

ALGORITHM / NOTES:

```
Erase screen and home cursor (ERASE).
Retrieve number of entries in tree list (TLGHDR).
IF (tree list is empty) THEN
    Send message to user and exit program.
ENDIF
DO UNTIL (user enters a correct treename or quits)
    Prompt for treename (PROMPT).
    Delete tree (DELETR).
    Send message to user if treename does not exist.
    IF (deleted tree is current data set) THEN
        Change current data set to 'NOTATREE', and
        the node name to 'NONE'(CMPCDS, CMGCDS).
    ENDIF
ENDDO
Put up option list at user's terminal (MENU).
END
```

FILES:

CM - communication
HS - history
TI - tree information (to be deleted)
TL - tree list
TV - tree vector (to be deleted)
instrumentation file
option file

REFERENCED BY:

OLPARS user.

SUBPROGRAMS REFERENCED:

CLOSFX, CMGCDS, CMGOTH, CMPCDS, DELETR, EQUALA, ERASE, INSINI, INSRET, MENU, OEXIT, OPENFX, PROMPT, TLGHDR TRMGET, TRMPUT

HISTORY:

designed by David Birnbaum August 28, 1978

programmed by Jill M. King August 29, 1980

OLPARS Program Specifications
DDSUBSTR

PROGRAM NAME: DDSUBSTR

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

Deletes substructures under a node in a chosen data set.

PROGRAM USAGE:

User types in 'DDSUBSTR'.

USER INTERACTION:

User is asked:

1) for a name of a data tree. Here the user can enter a colon (:) before the first character of the tree name and/or a star (*) for the tree name. The colon represents the senior node as the starting node. The star represents the current data set.
2) for a name of an "intermediate" node in the data tree if the colon was not entered in 1. (sub - structure underneath this node will be deleted).
3) whether or not to resequence the vector id's of the vectors lying at nodes in the substructure being deleted. If vector resequencing occurs, the user is also requested a starting vector id number.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

REPEAT

Prompt user for treename. (PROMPT).

Check for a colon.
Check for star.

UNTIL (valid tree name)

```
REPEAT
   IF (colon not typed above) THEN
      Ask for nodename.
      Search TI file for node (TISRCH).
      IF (node is a lowest node) THEN
         Send error message back to user.
      ENDIF
   ENDIF
   IF (nodename = senior node) THEN
        Request confirmation.
    ENDIF
UNTIL (valid nodename and treename)
```

Prompt user for resequencing strategy. (PROMPT).

Determine beginning and end of the set of lowest node underneath the user-supplied node.

Check to see if existing node name or class symbol is unique as a low node.

IF (name or symbol is not unique) Prompt user for new name and check this one.

Create a new TV file (CREATR) to transfer vectors into.

For each lowest node in the tree that is not beneath the user-supplied node:

1. transfer vectors into new TV file.

2. change vector pointer in TI file.

Transfer vectors from lowest nodes underneath user-supplied node into new TV file and set VP in user-supplied node.

Adjust counts and pointers in user-supplied node and in all nodes above that one. These include:

- o LNLP of the user-supplied node.
- o LNLP of the node that points to the first lowest node beneath the user-supplied node.
- o LNBP of the node that is pointed to by the LNLP of the last lowest node beneath the user-supplied node.
- o FCP of the user-supplied node.
- o FLNP of the user-supplied node.
- o Number of lowest nodes beneath the user-supplied node and all nodes above it.
- o FLNP of any node above the user-supplied node if this pointer pointed to a lowest node beneath the user-supplied node.

Add all nodes beneath the user supplied node to the garbage list of the TI file (GARBND).

Delete the old tree and rename the dummy tree to the old tree name (RENAMT).

Tell user that substructure has been deleted.

UNTIL (User is finished deleting substructures)

END

NOTE

DDSUBSTR needs 9 logical unit numbers allocated, rather than the 8 which appear to be indicated. This is due to the fact that CREATR opens the tree list file.

FILES:

CM - communications
TV - tree vector
TI - tree information
TL - tree list
instrumentation
Olpars option file
HS - History
TV2 - new tree vectors

REFERENCED BY:

OLPARS user.

SUBPROGRAMS REFERENCED:

CHARFL	CLOSTR	CMGOTH	CMGCDS	COPYVC	CREATR	EQUALA
ERASE	FIXKLV	GARBND	GLONOD	GNXTND	INSCHR	INSINI
INSINT	INSRET	LENGA	MENU	OEXIT	OPENFX	OPENTR
PROMPT	RENAMT	TIGCAP	TIGCOP	TIGET	TIGHDR	TIPCAP
TIPCOP	TIPNAM	TISRCH	TRMGET	TRMPUT	TVPHDR	UNIQND

HISTORY:

designed by David A. Birnbaum July 31, 1978

programmed by John W. Tenney February 18, 1981

OLPARS Program Specifications
DELETE

PROGRAM NAME: DELETE

CATEGORY: OLPARS utility (system dependent)

PROGRAM DESCRIPTION:

DELETE will delete any file, whether open or closed, using the .DLFNB routine from the SYSLIB.OLB library under RSX-11M V3.1.

PROGRAM USAGE:

```
INTEGER DELETE

N = DELETE(FILNAM,ERRCOD)
```

INPUT ARGUMENTS:

FILNAM - LOGICAL*1 array; file name represented by ASCII characters

OUTPUT ARGUMENTS:

ERRCOD - INTEGER; the file control service routine error code returned on an error from .DLFNB

ALGORITHM / NOTES:

The function value of DELETE will be set to -1 if an error has occurred. If the error was caused by FCS functions .CLOSE or .DLFNB, the error code parameter will be set to the error code found in offset F.ERR in the fdb used by DELETE. If the error was caused by an invalid lun (calling $FCHNL) or by an invalid syntax in the file name (invoking CSI$1) or by an appended switch to the file name (invoking CSI$2) then the error code will not be set (left at 0).

REFERENCED BY:

Level I subroutines and utility programs

HISTORY:

designed by Steve Haehn January 19, 1979

programmed by Dave Tipton March 12, 1979

modified by D. T. June 14, 1979

PROGRAM NAME: DELETR

CATEGORY: Level II File Manipulation Subroutine

PROGRAM DESCRIPTION:

DELETR removes the files that make up an OLPARS tree from the operating system file structure. The calling program supplies the NAME of the tree and the TYPE of tree (data or logic). The TYPE tells DELETR which list file, logic or tree, to search through for the tree NAME.

Value of TYPE - 1 - data tree
2 - logic tree

PROGRAM USAGE:

```
LOGICAL DELETR
     .
     .
     .
   IF (DELETR(TRNAME,TRTYPE)) THEN
```

INPUT ARGUMENTS:

TRNAME - CHARACTER STRING; 'name' of OLPARS tree to be deleted
TRTYPE - INTEGER; tree type

FILES:

TI - tree information
TL - tree list
TV - tree vector
LI - logic information
LL - logic list
LV - logic value

SUBPROGRAMS REFERENCED:

CLOSFX, INSINT, INSLOG, INSPGM, INSRET, LLGENT, LLGHDR, LLPENT, LLPHDR, LLSRCH, ODELET, OPENFX, TLGENT, TLGHDR, TLPENT, TLPHDR, TLSRCH, TRMPUT

DIAGNOSTICS:

If the tree NAME does not exist in the tree/logic list file, DELETR will return a 'false' value.

SEE ALSO:

RENAMT

HISTORY:

designed by Steve E. Haehn June 22, 1978

programmed by Jill M. King August 29,1980

PROGRAM NAME: DIGCME

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DIGCME gets a DI file entry for confusion matrix displays

PROGRAM USAGE:

CALL DIGCME(FID,ENTNO,NASCLS,NAME,NOS,PRCENT,ASSCLS)

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the DI file

ENTNO - INTEGER; the entry number

NASCLS - INTEGER; the number of assigned classes

OUTPUT ARGUMENTS:

NAME - INTEGER array (NODLEN); the tree class name

NOS - INTEGER array (4);

NOS(1) = number of vectors
NOS(2) = number of correct
NOS(3) = number of errors
NOS(4) = number of reject

PRCENT - REAL array (3);

PRCENT(1) = percent correct
PRCENT(2) = percent errors
PRCENT(3) = percent reject

ASSCLS - INTEGER array (50); numbers in each assigned class

OLPARS Program Specifications
DIGCME

FILES:

DI - display information

SUBPROGRAMS REFERENCED:

FGET

HISTORY:

designed by David A. Birnbaum October 3, 1978

programmed by John W. Tenney March 25, 1981

PROGRAM NAME: DIGCMH

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DIGCMH retrieves a DI file header in confusion matrix format. DIGCMH is the opposite of DIPCMH.

PROGRAM USAGE:

```
CALL DIGCMH(FID,CODPTR,TOTALS,PRCENT,LOGNAM,CLSNAM,DCODE,
            DSNAME)
```

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the DI file

OUTPUT ARGUMENTS:

CODPTR - INTEGER array (7);

```
CODPTR(1) = OLPARS option number
CODPTR(2) = number of classes
CODPTR(3) = NDIM
CODPTR(4) = number of vectors
CODPTR(5) = logic node number
CODPTR(6) = number of assigned classes
CODPTR(7) = logic type
```

TOTALS - INTEGER array (3);

```
TOTALS(1) = number correct
TOTALS(2) = number errors
TOTALS(3) = number reject
```

PRCENT - REAL array (3);

```
PRCENT(1) = percent correct
PRCENT(2) = percent errors
PRCENT(3) = percent reject
```

LOGNAM - INTEGER array (TRELEN); the logic name

CLSNAM - INTEGER array (NODLEN,MAXCLS); the array of reassigned class names

DCODE - INTEGER; display code

DSNAME - INTEGER array (TRELEN+NODLEN); the data set name

OLPARS Program Specifications
DIGCMH

SUBPROGRAMS REFERENCED:

FGET

SEE ALSO:

DIPCMH

HISTORY:

designed by John W. Tenney April 28, 1981

programmed by John W. Tenney May 5, 1981

PROGRAM NAME: DIGDC

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DIGDC gets the display code from the DI file.

PROGRAM USAGE:

```
INTEGER DIGDC
        .
        .
        .
DC = DIGDC(FIDDV)
```

INPUT ARGUMENTS:

FIDDI - INTEGER; the file descriptor of the DI file

OUTPUT ARGUMENTS:

DIGDC - INTEGER; the display code

REFERENCED BY:

RDISPLAY

SUBPROGRAMS REFERENCED:

FGET

HISTORY:

designed by David A. Birnbaum September 20, 1978

programmed by Dave Tipton June 18, 1979

PROGRAM NAME: DIGEFS

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DIGEFS gets the display flag symbol from either a one or two-space DI file entry.

PROGRAM USAGE:

CALL DIGEFS (FDDI, ENTNUM, DSPFSY)

INPUT ARGUMENTS:

FDDI - INTEGER; the file descriptor of the DI file

ENTNUM - INTEGER; entry number of the DI file entry to be accessed

OUTPUT ARGUMENTS:

DSPFSY - INTEGER; the display flag symbol

FILES:

DI - display information Instrumentation files

REFERENCED BY:

DISPCL

SUBPROGRAMS REFERENCED:

FGET INSCHR INSINT INSPGM INSRET OEXIT TRMPUT

SEE ALSO:

DIPEFS

HISTORY:

designed by Mark Maginn December 4,1980

programmed by Mark Maginn December 4,1980

PROGRAM NAME: DIGENT

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DIGENT gets information from a DI file entry for one-space or two-space displays.

PROGRAM USAGE:

```
CALL DIGENT(FID,ENTNO,ONETWO,NAME,NUMVEC,DVPTR,DISPLA,
            INFLAG,S1PTR)
```

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the DI file

ENTNO - INTEGER; the entry number in the DI file

ONETWO - INTEGER; equals 1 for a one-space DI file
equals 2 for a two-space DI file

OUTPUT ARGUMENTS:

NAME - INTEGER array (NODLEN); the classname

NUMVEC - INTEGER; the number of vectors

DVPTR - INTEGER; the DV file pointer

DISPLA - INTEGER; the display flag

INFLAG - INTEGER; the intensity flag (set to zero for two-space DI file)

S1PTR - INTEGER; the S1 file pointer (set to zero for two-space DI file)

FILES:

DI - display information

SUBPROGRAMS REFERENCED:

FGET,PACKW,TRMPUT

OLPARS Program Specifications
DIGENT

HISTORY:

designed by Steve Haehn October 24, 1978

programmed by Donna Morris May 20,1979

PROGRAM NAME: DIGHDI

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DIGHDI retrieves those portions of the DI file that should be represented by integers to the rest of OLPARS (for one- and two-space displays only).
DIGHDI is passed the file descriptor of the DI file and an information array (INFO). The information array is both an input and an output array. Non-negative values in INFO are replaced with the item referred to by the INFO array subscript (code). The coding scheme for DIGHDI is shown under ALGORITHM/NOTES.

PROGRAM USAGE:

CALL DIGHDI(FID,INFO)

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the DI file

INFO - INTEGER array (DINITM); array indicating which items to retrieve. This array is overwritten with values of the the items retrieved.

ALGORITHM / NOTES:

CODE (SUBSCRIPT)	ITEM RETRIEVED	ELEMENT NUMBER OF ITEM IN DI HEADER
1	Display Code	1
2	OLPARS Option Number	2
3	Dimension of Data Set	15
4	Number of Classes	16
5	Number of Boundaries	17
6	Number of Points in First Boundary	18
7	First Boundary Pointer	19
8	Number of Points in Second Boundary	20
9	Second Boundary Pointer	21
10	First Proj. Vector No.	30
11	Second Proj. Vector No.	31
12	Logic Node Number	40
13	Total Number of Vectors	41
14	Number of Bins	42
15	Type of Scaling	43
16	Zoom Flag	44
17	One-Space Intensity Flag	45

FILES:

DI - display information

SUBPROGRAMS REFERENCED:

FCET,TRMPUT

HISTORY:

designed by Dave Birnbaum June 15, 1978

programmed by Donna Morris May 4,1979

PROGRAM NAME: DIGMAX

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DIGMAX gets the current or original min and max values from the DI file header for one- and two-space displays. It is passed the file descriptor (FID) of the DI file and a flag I. If I=0, DIGMAX returns the original min, max values in MINMAX. If I=1, DIGMAX returns the current min,max values in MINMAX.

PROGRAM USAGE:

CALL DIGMAX(FID,I,MINMAX)

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the DI file

I - INTEGER; a flag which determines whether to get the original or current values

OUTPUT ARGUMENTS:

MINMAX - REAL array (4);
the min and max values in the DI file

FILES:

DI - display information

SUBPROGRAMS REFERENCED:

FGET

HISTORY:

designed by Dave Birnbaum June 21, 1978

programmed by David Tipton June 7, 1979

PROGRAM NAME: DIGNAM

CATEGORY: Level II File Access Subrcutine

PROGRAM DESCRIPTION:

DIGNAM gets the name of a data set or a logic from the DI file header for all types of displays. It is passed the file descriptor (FID) of the DI file and a flag (NAMFLG). If NAMFLG=0, the data set (treename, nodename pair) is returned in NAMES. If NAMFLG=1, the logic name is returned in NAMES.

PROGRAM USAGE:

CALL DIGNAM(FID,NAMFLG,NAMES)

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the DI file
NAMFLG - INTEGER; the name flag; 0 for data set; 1 for logic

OUTPUT ARGUMENTS:

NAMES - INTEGER(12); the names of the data set or logic

FILES:

DI - display information

SUBPROGRAMS REFERENCED:

FGET

HISTORY:

designed by David A. Birnbaum June 21, 1978

programmed by Dave Tipton June 15, 1979

PROGRAM NAME: DIGROE

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DIGROE retrieves all or a portion of the DI file rank order format entry.

PROGRAM USAGE:

```
CALL DIGROE(FIDDI,BEGIN,NUMBER,BUFFER)
```

INPUT ARGUMENTS:

```
 FIDDI - INTEGER; file descriptor for DI file
 BEGIN - INTEGER; the number of the first pair
                  of elements to be retrieved
NUMBER - INTEGER; the number of pairs of elements
                  to be retrieved
```

OUTPUT ARGUMENTS:

```
BUFFER - REAL array(2*NUMBER); the ranked values/
                               associated indices that are returned
```

FILES:

DI - display information

REFERENCED BY:

PROBCONF, SLCTMEAS, TRANSFRM, UNION

SUBPROGRAMS REFERENCED:

FGET, INSPGM, INSRET, OEXIT, TRMPUT

SEE ALSO:

DIPROE

HISTORY:

designed by Kermit Klingbail October 4, 1978

programmed by Donna Morris October 16, 1980

PROGRAM NAME: DIGROH

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DIGROH reads elements from the header of the Display Information file, in rank order format.

PROGRAM USAGE:

CALL DIGROH(FIDDI,ICODE,INFO1,INFO2)

INPUT ARGUMENTS:

FIDDI - INTEGER; file descriptor for DI file

ICODE - INTEGER; code to specify which elements are to be read

OUTPUT ARGUMENTS:

INFO1 - See Algorithm/Notes

INFO2 - See Algorithm/Notes

OLPARS Program Specifications
DIGROH

ALGORITHM / NOTES:

ICODE	INFO1	INFO2	Content
0	Integer Array (8 + MAXCLS)	Not Used	Entire Header
1	Integer	Not Used	Display Code
2	Integer	Not Used	Option Number
3	Integer	Not Used	ndim
4	Integer	Not Used	nclas
5	Integer	Not Used	Rank Option
6	Integer	Integer	Rank Option Parameters
7	Integer	Not Used	Sort Order Flag
8	Integer Array (MAXCLS)	Integer	Display Character List, nclas

NOTE: MAXCLS = the maximum number of classes allowed

FILES:

DI - display information (rank order format)

REFERENCED BY:

RODISP

SUBPROGRAMS REFERENCED:

FGET, INSPGM, INSRET, PACKW

SEE ALSO:

DIPROH

HISTORY:

designed by Kermit Klingbail October 4, 1978

programmed by Donna Morris October 17, 1980

PROGRAM NAME: DIGROI

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DIGROI reads the best class/best class pair indices array from the DI file header

PROGRAM USAGE:

```
CALL DIGROI(FIDDI,NDIM,INDEXS)
```

INPUT ARGUMENTS:

FIDDI - INTEGER; the file descriptor of the DI file
NDIM - INTEGER; the vector dimensionality

OUTPUT ARGUMENTS:

INDEXS - INTEGER array(2*NDIM); the best class/best class pair indices array

FILES:

DI - display information file (rank order format)

REFERENCED BY:

RODISP

SUBPROGRAMS REFERENCED:

FGET, INSPGM, INSRET, PACKW

SEE ALSO:

DIPROI

HISTORY:

designed by Donna Morris December 10, 1980

programmed by Donna Morris December 10, 1980

OLPARS Program Specifications
DIPCME

PROGRAM NAME: DIPCME

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DIPCME puts a DI file entry for confusion matrix displays

PROGRAM USAGE:

CALL DIPCME(FID,ENTNO,NASCLS,NAME,NOS,PRCENT,ASSCLS)

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the DI file

ENTNO - INTEGER; the entry number

NASCLS - INTEGER; the number of assigned classes

NAME - INTEGER array (NODLEN); the tree class name

NOS - INTEGER array (4);

NOS(1) = number of vectors
NOS(2) = number of correct
NOS(3) = number of errors
NOS(4) = number of reject

PRCENT - REAL array (3);

PRCENT(1) = percent correct
PRCENT(2) = percent errors
PRCENT(3) = percent reject

ASSCLS - INTEGER array (50); numbers in each assigned class

FILES:

DI - display information

SUBPROGRAMS REFERENCED:

FPUT

HISTORY:

designed by David A. Birnbaum October 3, 1978

programmed by John W. Tenney March 25, 1981

PROGRAM NAME: DIPCMH

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DIPCMH writes the DI file header for confusion matrix displays.

PROGRAM USAGE:

```
CALL DIPCMH(FID,CODPTR,TOTALS,PRCENT,LOGNAM,CLSNAM,DCODE,
            DSNAME)
```

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the DI file

CODPTR - INTEGER array (7);

```
CODPTR(1) = OLPARS option number
CODPTR(2) = number of classes
CODPTR(3) = NDIM
CODPTR(4) = number of vectors
CODPTR(5) = logic node number
CODPTR(6) = number of assigned classes
CODPTR(7) = logic type
```

TOTALS - INTEGER array (3);

```
TOTALS(1) = number correct
TOTALS(2) = number errors
TOTALS(3) = number reject
```

PRCENT - REAL array (3);

```
PRCENT(1) = percent correct
PRCENT(2) = percent errors
PRCENT(3) = percent reject
```

LOGNAM - INTEGER array (TRELEN); the logic name

CLSNAM - INTEGER array (NODLEN,MAXCLS); the array of reassigned class names

DCODE - INTEGER; the display code

DSNAME - INTEGER array (TRELEN+NODLEN); the data set name

SUBPROGRAMS REFERENCED:

FPUT

HISTORY:

designed by John W. Tenney April 29, 1978

programmed by John W. Tenney May 5, 1981

OLPARS Program Specifications
DIPEFS

PROGRAM NAME: DIPEFS

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DIPEFS puts the display flag symbol into either a one or two-space DI file entry.

PROGRAM USAGE:

CALL DIPEFS (FDDI, ENTNUM, DSPFSY)

INPUT ARGUMENTS:

FDDI - INTEGER; the file descriptor of the DI file

ENTNUM - INTEGER; entry number of the DI file entry to be accessed

DSPFSY - INTEGER; the display flag symbol

FILES:

DI - display information
Instrumentation files

REFERENCED BY:

DSPROJ, LCPROJ

SUBPROGRAMS REFERENCED:

FPUT INSCHR INSINT INSPGM INSRET

SEE ALSO:

DIGEFS

HISTORY:

designed by Mark Maginn December 4,1980

programmed by Mark Maginn December 4,1980

PROGRAM NAME: DIPENT

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DIPENT puts information into a DI file entry for one-space or two-space displays. It is passed the file descriptor of the DI file (FIDDI), an entry number in ENTNO and an array IPUT of negative ones and zeros. DIPENT only writes when IPUT entries are on as follows:

	ACTION
If IPUT(1) = 0	The contents of NAME are written into the classname.
If IPUT(2) = 0	NUMVEC is written into the number of vectors.
If IPUT(3) = 0	DVPTR is written into the DV file pointer.
If IPUT(4) = 0	ONOFF is written into the display flag.
If IPUT(5) = 0	INFLAG is written into the intensity flag.
If IPUT(6) = 0	S1PTR is written into the S1 pointer.

PROGRAM USAGE:

```
CALL DIPENT(FIDDI,ENTNO,IPUT,NAME,NUMVEC,DVPTR,ONOFF,
            INFLAG,S1PTR)
```

INPUT ARGUMENTS:

```
 FIDDI - INTEGER; the file descriptor of the DI file
 ENTNO - INTEGER; the entry number in DI to write to
  IPUT - INTEGER array (6); action code storage
  NAME - INTEGER; the classname
NUMVEC - INTEGER; the number of vectors
 DVPTR - INTEGER; the DV file pointer
 ONOFF - INTEGER; the display flag
INFLAG - INTEGER; the intensity flag
 S1PTR - INTEGER; the S1 file pointer
```

OLPARS Program Specifications
DIPENT

FILES:

DI - display information

SUBPROGRAMS REFERENCED:

FPUT, ITREAL

HISTORY:

designed by Dave Birnbaum September 12, 1978

programmed by Donna Morris April 1,1979

PROGRAM NAME: DIPHDI

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DIPHDI puts integer information into the one- or two-space DI file header. DIPHDI is passed the file descriptor of the DI file and an array of values (INFO). The INFO array subscript is the code of an item that can be referenced by DIPHDI. A negative value in the array indicates that the item specified by the code (subscript) is not to be written. The codes are the same as in DIGHDI.

PROGRAM USAGE:

CALL DIPHDI(FID,INFO)

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the DI file
INFO - INTEGER array (DINITM); the array of values

FILES:

DI - display information

SUBPROGRAMS REFERENCED:

FPUT

SEE ALSO:

DIGHDI

HISTORY:

designed by Dave Birnbaum June 15, 1978

programmed by Donna Morris May 4, 1979

PROGRAM NAME: DIPMAX

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DIPMAX puts the current or original min and max values into the DI file header for one-space or two-space displays.

DIPMAX is passed the file descriptor (FID) of the DI file, a flag OCFLAG, set equal to 0 or 1, and MINMAX (a real array that contains the original or current min,max values).

If OCFLAG=0, DIPMAX places MINMAX into the original min, max area in the DI file header. If OCFLAG=1, DIPMAX places MINMAX into the current min,max area in DI. For one-space displays the y min and max values should be set to 0.

PROGRAM USAGE:

```
CALL DIPMAX(FID,OCFLAG,MINMAX)
```

INPUT ARGUMENTS:

```
   FID - INTEGER; the file descriptor of the DI file
OCFLAG - INTEGER; original/current flag
MINMAX - REAL array (4); the min,max values
```

FILES:

DI - display information

SUBPROGRAMS REFERENCED:

FPUT

HISTORY:

designed by Dave Birnbaum June 23, 1978

programmed by David Tipton June 7, 1979

PROGRAM NAME: DIPNAM

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DIPNAM puts the name of a data set or a logic into the DI file header for all types of displays. It is passed the file descriptor (FID) of the DI file, a flag NAMFLG and the names in NAMES. IF NAMFLG=0, NAMES is placed in the data set portion of the DI header. If NAMFLG=1, the first eight characters of NAMES are placed in the logic portion of the DI header.

PROGRAM USAGE:

CALL DIPNAM(FID,NAMFLG,NAMES)

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the DI file
NAMFLG - INTEGER; the name flag; 0= put data set names, 1= put logic name
NAMES - INTEGER array (12); the data set or logic tree names

FILES:

DI - display information

SUBPROGRAMS REFERENCED:

FPUT

HISTORY:

designed by David A. Birnbaum June 22, 1978

programmed by Dave Tipton June 15, 1979

PROGRAM NAME: DIPROE

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DIPROE writes all or a portion of the DI file rank order format entry.

PROGRAM USAGE:

```
CALL DIPROE(FIDDI,BEGIN,NUMBER,BUFFER)
```

INPUT ARGUMENTS:

```
 FIDDI - INTEGER; file descriptor for DI file
 BEGIN - INTEGER; the number of the first pair
                  of elements to be written
NUMBER - INTEGER; the number of pairs of elements
                  to be written
BUFFER - REAL array(2*NUMBER); ranked values and
                  associated indices to be written
```

FILES:

DI - display information

REFERENCED BY:

DSCRMEAS, PROBCONF, RANK, SLCTMEAS, UNION

SUBPROGRAMS REFERENCED:

FPUT, INSPGM, INSRET

SEE ALSO:

DIGROE

HISTORY:

designed by Kermit Klingbail October 4, 1978

programmed by Donna Morris October 16, 1980

PROGRAM NAME: DIPROH

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DIPROH puts elements into the header of the Display Information file, in rank order format.

PROGRAM USAGE:

CALL DIPROH(FIDDI,ICODE,INFO1,INFO2)

INPUT ARGUMENTS:

FIDDI - INTEGER; file descriptor for DI file

ICODE - INTEGER; code to specify which elements are to be written

INFO1 - See Algorithm/Notes

INFO2 - See Algorithm/Notes

ALGORITHM / NOTES:

ICODE	INFO1	INFO2	Content
0	Integer Array (8 + MAXCLS)	Not Used	Entire Header
1	Integer	Not Used	Display Code
2	Integer	Not Used	Option Number
3	Integer	Not Used	ndim
4	Integer	Not Used	nclas
5	Integer	Not Used	Rank Option
6	Integer	Integer	Rank Option Parameters
7	Integer	Not Used	Sort Order Flag
8	Integer Array (MAXCLS)	Integer	Display Character List, nclas

NOTE: MAXCLS = the maximum number of classes allowed

FILES:

DI - display information (rank order format)

REFERENCED BY:

DSCRMEAS, PROBCONF, RANK

SUBPROGRAMS REFERENCED:

FPUT, INSPGM, INSRET, ITREAL

SEE ALSO:

DIGROH

HISTORY:

designed by Kermit Klingbail October 4, 1978

programmed by Donna Morris October 17, 1980

PROGRAM NAME: DIPROI

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DIPROI writes the best class/best class pair indices as part of the DI file header.

PROGRAM USAGE:

```
CALL DIPROI(FIDDI,NDIM,INDEXS)
```

INPUT ARGUMENTS:

```
 FIDDI - INTEGER; the file descriptor of the DI file
  NDIM - INTEGER; the vector dimensionality
INDEXS - INTEGER array(2*NDIM); the best class/best
                                class pair indices array
```

FILES:

DI - display information file (rank order format)

REFERENCED BY:

DSCRMEAS, PROBCONF, RANK

SUBPROGRAMS REFERENCED:

FPUT, INSPGM, INSRET, ITREAL

SEE ALSO:

DIGROI

HISTORY:

designed by Donna Morris December 10, 1980

programmed by Donna Morris December 10, 1980

OLPARS Program Specifications
DISPCL

PROGRAM NAME: DISPCL

CATEGORY: Terminal Display Routine

PROGRAM DESCRIPTION:

DISPCL is passed the file descriptor (FID) of the DI file. DISPCL first erases the screen and moves the cursor to the upper left-hand portion of the screen. It then goes through the DI file, entry by entry, displaying the class symbols (two symbols per line) and placing an * (asterisk) next to those classes where the display flag is on.

PROGRAM USAGE:

CALL DISPCL(FID)

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the DI file

FILES:

DI - display information

SUBPROGRAMS REFERENCED:

ERASE, DIGHDI, DIGENT, TRMPUT

HISTORY:

designed by Dave Birnbaum September 20, 1978

programmed by David Tipton June 6, 1979

PROGRAM NAME: DISTEU

CATEGORY: Numerical Computation Algorithm

PROGRAM DESCRIPTION:

DISTEU returns as its value the square of the Euclidean distance between two vectors. This is defined as:

$$dist = \sum_{i=1}^{NDIM} (x(i) - y(i))^{**2}$$

Where **2 = [raised to the second power]

$\sum_{i=1}^{NDIM}$ = [the summation as i goes from 1 to NDIM]

PROGRAM USAGE:

```
REAL FUNCTION DISTEU

D = DISTEU(VEC1,VEC2,NDIM)

Where VEC1 = [x(1) ,..., x(NDIM)]

      VEC2 = [y(1) ,..., y(NDIM)]
```

INPUT ARGUMENTS:

VEC1 - REAL array (NDIM); the first vector
VEC2 - REAL array (NDIM); the second vector
NDIM - INTEGER; the dimension of the vectors

REFERENCED BY:

NMDIST

HISTORY:

designed by David A. Birnbaum September 22, 1978

programmed by John W. Tenney March 11, 1981

PROGRAM NAME: DISTMA

CATEGORY: Numerical Computation Algorithm

PROGRAM DESCRIPTION:

DISTMA returns as its value the Mahalanobis distance squared between two vectors.

PROGRAM USAGE:

```
    REAL FUNCTION DISTMA

D = DISTMA(VEC1,VEC2,NDIM,MATRIX)
```

INPUT ARGUMENTS:

```
  VEC1 - REAL array (NDIM); the first vector
  VEC2 - REAL array (NDIM); the second vector
  NDIM - INTEGER; the vector dimensionality
MATRIX - REAL array ((NDIM * (NDIM+1))/2); the matrix
```

ALGORITHM NOTES:

The Mahalonobis distance is defined by:

dist = DVT * COV(-1) * DV

where DV = distance vector (VEC1 - VEC2)

DVT = transpose of DV

COV(-1) = The inverse of the covariance matrix

The following example shows the derivation of the algorithm. This example is for NDIM=3:

```
          |c(1,1) c(1,2) c(1,3)| (A
(A B C)   |c(2,1) c(2,2) c(2,3)|  B
          |c(3,1) c(3,2) c(3,3)|  C)

=         (Ac(1,1) + Bc(2,1) + Cc(3,1)) * A +
          (Ac(1,2) + Bc(2,2) + Cc(3,2)) * B +
          (Ac(1,3) + Bc(2,3) + Cc(3,3)) * C

=         AAc(1,1) + ABc(2,1) + ACc(3,1) +
          ABc(1,2) + BBc(2,2) + BCc(3,2) +
          ACc(1,3) + BCc(2,3) + CCc(3,3)
```

Since the inverse of the covariance is a symmetrical matrix, any element with (x,y) as coordinates is equal to any with (y,x).

Therefore:

```
dist = AAc(1,1) + 2ABc(1,2) + 2ACc(1,3) +
       BBc(2,2) + 2BCc(2,3) +
       CCc(3,3)
```

REFERENCED BY:

NMDIST

HISTORY:

designed by John W. Tenney March 11, 1981

programmed by John W. Tenney March 11, 1981

PROGRAM NAME: DISTQC

CATEGORY: Numerical Computation Algorithm

PROGRAM DESCRIPTION:

DISTQC returns as its value the value of the quadratic classifier discriminant function between a vector and a class in NMV logic.

PROGRAM USAGE:

```
REAL FUNCTION DISTQC

D = DISTQC(VEC1,VEC2,NDIM,MATRIX,DET,PROB)
```

INPUT ARGUMENTS:

VEC1 - REAL array (NDIM); the vector

VEC2 - REAL array (NDIM); the mean vector

NDIM - INTEGER; the dimension

MATRIX - REAL array ((NDIM (NDIM+1))/2); the packed inverse of the covariance matrix

DET - REAL; the determinant of the covariance matrix

PROB - REAL; the a priori probability

SUBPROGRAMS REFERENCED:

DISTMA

REFERENCED BY:

NMDIST

HISTORY:

designed by John W. Tenney May 14, 1981

programmed by John W. Tenney May 21, 1981

PROGRAM NAME: DISTVA

CATEGORY: Numerical Computation Algorithm

PROGRAM DESCRIPTION:

DISTVA returns as its value the square of a weighted Euclidean distance between two vectors. This is defined as

$$dist = \sum_{i=1}^{Ndim} \frac{(x(i) - y(i))^{**2}}{V(i)}$$

where **2 = [raised to the second power]

$\sum_{i=1}^{Ndim}$ = [the summation as i goes from 1 to Ndim]

V = [V(1) ,..., V(NDIM)] are the weights.

V is the vector of variances when used by NMV routines.

PROGRAM USAGE:

```
D = DISTVA(VEC1,VEC2,NDIM,V)

    REAL FUNCTION DISTVA
```

Where: VEC1 = [x(1) ,..., x(NDIM)] is the first vector,

VEC2 = [y(1) ,..., y(NDIM)] is the second vector,

INPUT ARGUMENTS:

```
  VEC1 - REAL array (NDIM); the first vector
  VEC2 - REAL array (NDIM); the second vector
     V - REAL array (NDIM); the weights
  NDIM - INTEGER; the vector dimensionality
```

REFERENCED BY:

NMDIST

HISTORY:

designed by David A. Birnbaum September 29, 1978

programmed by John W. Tenney March 11, 1981

OLPARS Program Specifications
DITHER

PROGRAM NAME: DITHER

CATEGORY: Numerical Computation Algorithm

PROGRAM DESCRIPTION:

This routine is used in conjunction with the INVERT routine. When a linearly dependent row is found in a symetric matrix, this implies that the matrix is singular, ie. no inverse exists. The function of the DITHER routine is to slightly change the diagonal element in the linearly dependent row of the matrix to be inverted, and hence eliminate the linear dependence. Thus the inverse computed is actually the inverse of a matrix slightly different from the original.

PROGRAM USAGE:

CALL DITHER(MATRIX,ROW,NDIM)

INPUT ARGUMENTS:

MATRIX - REAL array ((Ndim*(Ndim+1))/2); the lower triangular portion of the symetric matrix to be inverted, stored as a vector

ROW - INTEGER; the row containing the linear dependency

NDIM - INTEGER; the order of the matrix

OUTPUT ARGUMENTS:

MATRIX - REAL array ((Ndim*(Ndim+1))/2); the lower triangular portion of the symetric matrix, stored as a vector, after the pivot element has been changed

AGORITHM/NOTES:

```
DO i = 1,Ndim

   IF (i = row number of linearly dependent row) THEN

      Locate the 'bad' diagonal pivot element in the row
      and store it's vector location (IDX).

   ELSE

      Locate the diagonal element of minimum value in the
      array, not equal to zero (excluding the 'bad' pivot
      element.

   ENDIF

ENDDO

Clear the error flag.

Slightly alter the pivot element in the linearly
dependent row to eliminate the linear dependency.
The formulas are as follows:
   IF (Matrix(KK).eq.0)  Matrix(KK)=(.5)*minimum
   IF (Matrix(KK).ne.0)  Matrix(KK)=(1.001)*Matrix(KK)

RETURN
```

FILES:

instrumentation file

REFERENCED BY:

L1ASDG, L2ASDG, L2FSHP, S2FSHP

SUBPROGRAMS REFERENCED:

IDX, INSFLT, INSINT, INSPGM, INSRET

HISTORY:

designed by Jill King September 29, 1980

programmed by Jill King September 29, 1980

PROGRAM NAME: DLOGTREE

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

DLOGTREE deletes a user-specified logic tree from the user's directory.

PROGRAM USAGE:

User types in 'DLOGTREE'.

ALGORITHM / NOTES:

```
Erase screen and home cursor (ERASE).
Retrieve number of entries in tree list (LLGHDR).
IF (tree list is empty) THEN
    Send message to user and exit program.
ENDIF
DO UNTIL (user enters a correct logic name or quits)
    Prompt for treename (PROMPT).
    Delete logic (DELETR).
    Send message to user if logic name does not exist.
    IF (deleted logic is the current logic) THEN
        Change current logic to 'NOLOGIC' (CMGLOG,
        CMPLOG).
    ENDIF
ENDDO
Put up option list at user's terminal (MENU).
END
```

FILES:

CM - communication
HS - history
LI - logic information
LL - logic list
LV - logic value
instrumentation file
option file

REFERENCED BY:

OLPARS user.

SUBPROGRAMS REFERENCED:

CLOSFX, CMGLOG, CMGOTH, CMPLOG, DELETR, EQUALA, ERASE, INSINI, INSRET, LLGHDR, MENU, OEXIT, OPENFX, PROMPT, TRMGET, TRMPUT

HISTORY:

designed by David A. Birnbaum March 7, 1978

programmed by Jill M. King September 8, 1980

PROGRAM NAME: DLREGN

CATEGORY: Logic Tree File Access Routine

PROGRAM DESCRIPTION:

DLREGN enters a logic block (i.e., the set of LV file entries that correspond to a logic node or a reject strategy of a logic node) into the LV free list (it deletes a logic block region).

PROGRAM USAGE:

```
CALL DLREGN(FDLV,RGNPTR)
```

INPUT ARGUMENTS:

```
  FDLV - INTEGER; the file descriptor of the LV file
RGNPTR - INTEGER; the starting entry in the LV of the
                  logic block
```

ALGORITHM / NOTES:

```
Get LV header (LVGHDR).

WHILE (We've not reached the end of the region
       (a completed region's last link is zero) or
       beginning of the 'free list' (an incomplete
       region's last link will point to the head
       of the free list - see program 'GNLVAE'))

       Take entries out of active use, that is, negate
       the entry's current link.

ENDWHILE

Write LV header (LVPHDR).
```

FILES:

LV - logic value

REFERENCED BY:

KILLND

SUBPROGRAMS REFERENCED:

INSINT, INSPGM, INSRET, LVGHDR, LVGLNK, LVPHDR, LVPLNK, OEXIT, TRMPUT

HISTORY:

designed by David BirnBaum September 19, 1978

programmed by Steven Haehn March 5, 1981

PROGRAM NAME: DLSUBSTR

CATEGORY: UTILITY COMMAND

PROGRAM DESCRIPTION:

DLSUBSTR deletes the logic at a specified logic node.

PROGRAM USAGE:

User types in 'DLSUBSTR'.

USER INTERACTION:

The user is asked for the name of the logic tree, the logic node number for which logic is to be deleted, and what type of logic should be deleted:
(1) all logic at and below the logic node specified,
(2) independent reject strategy only, or
(3) decision logic only (excluding independent reject strategy).

ALGORITHM / NOTES:

Erase the screen and home the cursor (ERASE).

REPEAT

Prompt the user for the logic tree name from which logic is to be deleted (PROMPT,TRMGET).

If a colon was entered before the tree name (indicating the senior node will be the starting node), the node number will be set to 1 (EQUALA).

If an asterisk was entered (indicating that the current logic tree is wanted), retrieve the current logic tree name (EQUALA,CMGLOG).

UNTIL (correct response is given)

Retrieve the name of the design data set (LIGDSN).

Make sure the logic name at the design data set is the same as the user specified logic name (TVGHDR,EQUALA).

```
REPEAT

   IF (a colon was not entered before the tree name) THEN

      Prompt the user for the logic node number for
      which logic is to be deleted (PROMPT,
      TRMGET).

   ENDIF

   Make sure the specified logic node is within the
   range of valid node numbers and is not in the free
   list (LIGLOG).

   REPEAT

      Ask the user for the type of logic to be deleted:
      (1) All logic at and below the logic node
      (2) Independent Reject Strategy only
      (3) Decision Logic only (excludes Independent
          Reject Strategy)

   UNTIL (correct response is entered)

   IF (the user has selected the senior logic node,
       and has indicated that all logic at and below
       that node should be deleted) THEN

      Make sure the user wants to delete the entire data
      tree structure (PROMPT,TRMGET).

   ENDIF

   IF (All logic at and below the logic node is to
       be deleted) THEN

      Delete logic substructure and fix up design
      data set (KILLND)

   ELSEIF (Independent Reject Stategy only is to be
           deleted) THEN

      Delete the Independent Reject Strategy (DLREGN)
```

```
            ELSEIF (Decision Logic only is to be deleted) THEN

               Save pointer in LV tc Independent Reject Strategy
               Write a zero in LI file as pointer to Ind. Rej.
               Strategy
               Delete Decision Logic and fix up design data set
               (call to KILLND; KILLND will think there is no Ind.
               Rej. Strat. so it won't delete it)
               Rewrite the correct pointer (in LV) to the
               Ind. Rej. Strat.

            ENDIF

            Notify the user that logic has been deleted.

         UNTIL  (user quits via OLPARS 'exit command' character)

         Update number of incomplete logic nodes in the logic
         tree.

         IF (logic tree is incomplete) Set the reassociated names
                                       flag to zero (not set).

         IF (the current logic tree was specified) update the
            number of incomplete logic nodes in the CM file.

         Put up the user's option list (MENU).

         END

FILES:

         CM - communications file
         HS - history file
         LI - logic information file
         LV - logic vector file
         TI - tree information file
         TV - tree vector file
         instrumentation file
         option file

SUBPROGRAMS REFERENCED:

         CMGLOG, CMGOTH, CMPLOG, DLREGN, EQUALA, ERASE,
         GETLST, INSINI, INSRET, KILLND, LIGCOP, LIGDSN,
         LIGLOG, LIGSTR, LIPCOP, LIPSTR, MENU, OEXIT,
         OPENFX, OPENTR, PROMPT, TRMGET, TRMPUT, TVGHDR
```

HISTORY:

designed by Jill King June 12, 1981

programmed by Jill King June 12, 1981

OLPARS Program Specifications
DRAWBNDY

PROGRAM NAME: DRAWBNDY

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

DRAWBNDY lets an OLPARS user draw a partition on a data projection.

PROGRAM USAGE:

User types in 'DRAWBNDY'.

USER INTERACTION:

Using the graphics cursor, the user is asked to enter a threshold point for one-space displays (maximum of two thresholds). For two-space displays he is asked to enter a boundary and the convex region of the boundary (maximum of two boundaries, maximum of 6 points per boundary).

During one-space entry:

- Typing any alphanumeric character, except 'Q', enters a threshold value.
- Typing a 'Q' quits the program (automatic exit occurs after entering two threshold values).

During two-space boundary entry mode:

- Typing any alphanumeric character, except 'Q', enters a boundary point.
- Typing a 'Q' quits the entering of the boundary. (If 'Q' is typed before entering any points, the program exits.) After six boundary points have been entered, the program automatically leaves boundary entry mode.

During two-space convex point entry mode:

- Typing any alphanumeric character, except 'Q', enters a convex point.
- Typing a 'Q' exits the program. (Note: Boundary is not entered into the display files when a 'Q' is typed in this mode.)

ALGORITHM / NOTES:

Obtain current data set info. from CM file.
Obtain the display code from the DI file.

CHECK FOR:

- 'excess measurement' mode
- the existence of a current data set
- one or two space display code
- OLPARS long prompt mode

Get the current min's and max's from the DI file.

IF (this is a TWO-SPACE plot) THEN

Set the number of boundaries to zero.

REPEAT

Display entry mode option (BOUNDARY:)

Read a characater and set of coordinates from terminal.

Set the number of boundary points to 1.

IF the character was a 'Q', BREAK.

Scale the coordinates.

REPEAT

Read a character and set of coordinates from terminal.

IF the character was a 'Q,' BREAK.

Draw a line from the previous point to the new point.

Scale new point.

Add 1 to the number of boundary points.

UNTIL the number of boundary points >= 6.

```
        IF the number of boundary points < 2.

            Set EXIT number to 1.

            Tell user that the boundary was not put
            in DI file (because only 1 point was
            read during boundary entry mode).

            BREAK (out of outer REPEAT loop).

        ELSE

            Display the entry mode option.
            (CONVX PT:)

            Read character and set of coordinates
            from the terminal.

            IF the character was a 'Q.'

                Set the EXIT number to 2.

                Tell user that the boundary was not
                put in DI file (because the convex
                point was not entered).

                BREAK (out of outer REPEAT loop).

            ELSE

                Determine boundary coordinates in
                projected space.

                Place the boundary, convex point,
                and the number of points in the
                boundary in the DI and DV files.

                Add 1 to the number of boundaries.

            ENDIF

        ENDIF

    UNTIL the number of boundaries is >= 2.

    IF the number of boundaries >0

        Place the number of boundaries in the DI file.

    ENDIF
```

```
        ELSEIF (this is a ONE-SPACE plot) THEN

            Display the entry mode option (THRESHOLD:)

            REPEAT

                Read a character and set of coordinates
                from terminal

                IF the character was  'Q,' BREAK (exit
                REPEAT loop).

                Convert the coordinates to a threshold
                value and store in the DV file
                (set pointer in DI file).

                Add 1 to the threshold counter.

            UNTIL the threshold counter is > 2.


        IF (there is more than one threshold) THEN

            Make sure that the left most threshold
            is the first threshold.

        ENDIF

        Place boundary points into the Display files.

        Put up the Menu (only if there a previous
                         screen erasure)

        END
```

FILES:

CM - communications
DI - display information
DV - display value

REFERENCED BY:

OLPARS user.

OLPARS Program Specifications
DRAWBNDY

SUBPROGRAMS REFERENCED:

BNDEXP, BOUND, CHKEXS, CLOSFX, CMGCDS, CMGOTH,
CMGSCN, DIGDC, DIGMAX, DIPHDI, DVGHDR, DVPVEC,
EQUALA, ERASE, GIN, INSINI, INSINT, INSLOG,
INSRET, LINSEG, MENU, OEXIT, OPENFX, TEXT,
TRMPUT

SEE ALSO:

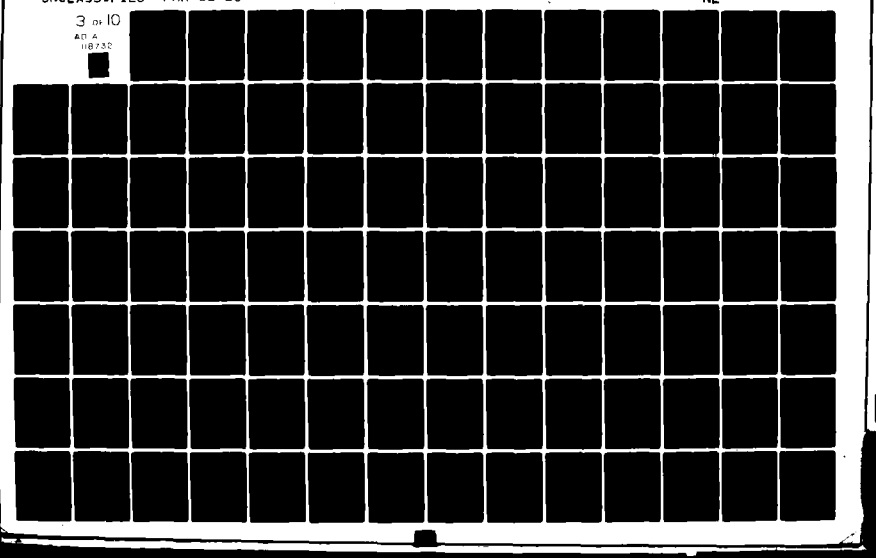
REDRAW, RDISPLAY

HISTORY:

designed by Steven Haehn July 27, 1978

programmed by Dave Tipton September 26,1979

AD-A118 732 PAR TECHNOLOGY CORP NEW HARTFORD NY F/G 9/2
ON-LINE PATTERN ANALYSIS AND RECOGNITION SYSTEM. OLPARS VI. SOF--ETC(U)
JUN 82 S E HAEHN, D MORRIS
UNCLASSIFIED PAR-82-20 NL



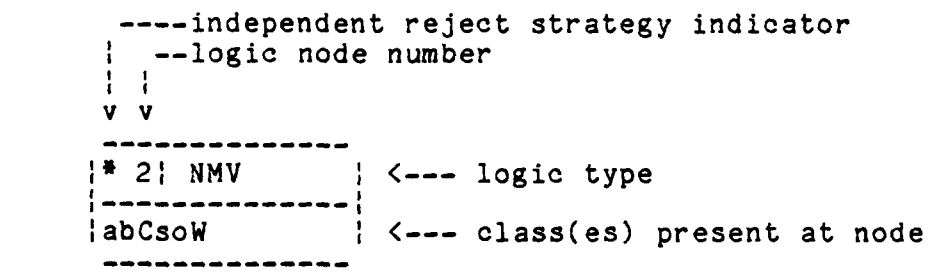
PROGRAM NAME: DRAWLOG

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

DRAWLOG produces a pictorial display of a portion (or all) of a logic tree at any stage in the development of the logic. Logic nodes are displayed as partitioned boxes with interconnecting lines to illustrate their relationship to each other.

EXAMPLE LOGIC NODE



The class(es) present at a logic node are listed in the bottom portion of the box of every logic node (except REJECT nodes). If the node is a completed logic node, there is only one class present, and its name appears in this portion of the box. Otherwise the class display symbols of the classes present at the node appear in this region.

In the upper left portion of every logic node box is the logic node number preceded by either a star or a blank, depending upon whether an independent Boolean reject strategy is associated with the node or not.

The upper right portion of the logic node box contains the type of logic at the node (e.g. NMV), 'INCMPLT' for incomplete logic nodes, or '*REJECT*' for reject nodes.

If a given logic tree is too large to display on the screen, only a portion of the tree is displayed; with the user option of displaying more.

PROGRAM USAGE:

User types in 'DRAWLOG'.

USER INTERACTION:

Asks user to input a logic tree name. A star (*) represents current logic.

Asks user to input a logic node number. A colon (:) preceding the logic tree name (or star) means the senior node is selected.

The selected logic node and all logic nodes below it, or following it on the same level are displayed as described above.

If the entire structure has not been displayed, ask user for another logic node number (it is likely that '0' or '-1' may be acceptable for paging here).

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Ask user for name of logic tree to be drawn (PROMPT).

IF (a star is entered) then

   Get current logic (CMGLOG).

   IF (current logic does not exist) then

      Print error message (TRMPUT) and exit.

   ENDIF

ENDIF

Find class names and number of all low nodes in the design data set (LIGCNM).

Find maximum number which user may specify as a logic node number (LIGCOP).

Find design data set name (LIGDSN).

Find dimensionality of the design data set (LIGCOP).

REPEAT

   Ask user to input a logic node number (PROMPT).

   IF (logic node number entered is negative) then

      Negate the logic node number.

```
   Find parent's structure (LIGSTR).

   IF (parent's first child was not chosen) then

      Retrieve array of next level nodes below
      parent (GLSTRC).

      Find previous sibling of selected node and
      use it as the selected node (the previous
      sibling will be displayed as '-1').

   ENDIF

ENDIF

Obtain logic tree structure to be drawn (GLSTRC).

WHILE (structure has too many levels - too deep, or
      too many low nodes -too wide)

   IF (structure is too deep) then

      Set new cutoff level to maximum level that
      can be displayed.

   ENDIF

   IF (structure is too wide) then

      Locate the first level that is not too wide
      to be displayed.

      IF (we are at the first level)

         Save the second level logic node numbers
         to be used for paging.

         Update the count of the second level nodes
         to be the maximum that can be displayed.

         Break

      ELSE

         Set cutoff level to the level that can be
         displayed.

      ENDIF

      Obtain structure with a new cutoff level
      (GLSTRC).

ENDWHILE
```

```
IF (another page exists before this one) then

   Set minus one (-1) as the logic node number on
   the bottom of the second level.

   IF (no more pages exist) set page flag false

ENDIF

Compute the display positions for each lowest node
in the logic tree structure.

Compute the display positions for each intermediate
node in the logic tree structure.

WHILE (there are more levels to be displayed)

   Compute the number of nodes at this level.

   WHILE (nodes still exist at this level)

      Position cursor to correct position (MOVE).

      Output reject strategy and logic node number
      (TRMPUT).

      IF (logic exists at this node) then

         Get the option name from the option file
         (GOPTNM).

         Get the list of classes present (GCLIST).

         Print out option name and list of classes
         on line below (TRMPUT).

         IF (undisplayed children exist) then

            Print fan to right of box (LINSEG).

         ENDIF

      ELSEIF (logic is incomplete or complete) then

         Print out reject strategy and logic node
         number (TRMPUT).

         Get list of classes present (GCLIST).

         IF (number of classes is .GT 1) THEN

            Print 'INCMPLT' in upper box (TRMPUT).
```

```
            Print class list in lower box (TRMPUT).

         ELSEIF (number of classes is .EQ. 1) then

            Get the design node name (LIGDCN).

            Print the design node name in the lower
            partition (TRMPUT).

         ELSE

            Print '*REJECT*' in upper partition
            (TRMPUT).

         ENDIF

      ELSE

         Print nothing (the logic node is in the
         free list of the LI file).

      ENDIF

      If (box should be drawn) draw it (LINSEG).

      If (line should be drawn back to parent) then

         Draw it (LINSEG).

      ENDIF

   ENDWHILE

ENDWHILE

Print name of logic tree, time, and classes present
at top of display.

Print design data set tree and node name, class
count, and dimensionality at bottom of display.

UNTIL (the whole logic tree was displayed only on the
      first display)

Put up option list at user's terminal (MENU).

RETURN
```

OLPARS Program Specifications
DRAWLOG

FILES:

CM - communications	LI - logic information
HS - history	LV - logic value
Instrumentation	OLPARS options

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CMGLOG	CMGOTH	CMGSCN	EQUALA	ERASE
GCLIST	GLSTRC	GOPTNM	INSFLT	INSINI
INSINT	INSRET	LIGCNM	LIGCOP	LIGDCN
LIGDSN	LIGLOG	LIGSTR	LINSEG	MENU
MOVE	OEXIT	OPENFX	OPENTR	PROMPT
TRMGET	TRMPUT	WRTNOW		

SEE ALSO:

DRAWTREE

HISTORY:

designed by David A. Birnbaum September 19, 1978

programmed by Dave Tipton May 29, 1981

PROGRAM NAME: DRAWTREE

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

DRAWTREE displays a selected OLPARS data tree, showing the structural relationship between the various nodes of the tree.

PROGRAM USAGE:

User types in 'DRAWTREE'.

USER INTERACTION:

The user is asked for the name of the data set to be drawn.

ALGORITHM / NOTES:

```
Obtain a tree name and a class name from user.

Obtain a full tree structure.

Determine a cutoff level for drawing.

IF (no error has occurred in the structure)

    IF (cutoff level is at the senior node)

        Show as much of the structure as possible

    ELSE

        Determine the horizontal separation distance
        between each level in the tree.

        Determine the vertical separation distance
        between the lowest nodes.

        Determine the position of nodes in the display
        and lines to be drawn between the nodes.

        Display the node names.

        Draw the connecting lines.

        IF (the whole tree has not been displayed)

            Display the incomplete node lines.
```

```
            ENDIF

        ENDIF

    ELSE

        Tell user "tree structure in error."

    ENDIF

    Put up option list.
```

FILES:

```
    CM - communications          Instrumentation
    HS - history
    TI - tree information
    TL - tree list    ---------  OLPARS option file
```

REFERENCED BY:

OLPARS user.

SUBPROGRAMS REFERENCED:

CHARFL, CMGOTH, CMGSCN, EQUALA, ERASE, GDSTRC, INSFLT, INSINI, INSINT, INSRET, LENGA, LINSEG, MENU, MOVE, OEXIT, OPENFX, OPENTR, PROMPT, TIGET, TIGHDR, TISRCH, TRMGET, TRMPUT, WRTNOW

SEE ALSO:

DRAWLOG

HISTORY:

designed by Steven Haehn August 7, 1978

programmed by Mark Maginn and Steven Haehn Dec. 16, 1980

PROGRAM NAME: DSCRMEAS

CATEGORY: Measurement Evaluation Command

PROGRAM DESCRIPTION:

DSCRMEAS opens the TI file for the current data set and gets the number and list of names of the lowest nodes. The DI and DV file headers are created. From the means and variances of each of the lowest node classes, the M(ij) discriminant values are computed and stored as entries in the DV file. From the M(ij) discriminants, the M(i) and M discriminant values are computed and stored as entries in the DV file.

Next, the asterisk array in DI is initialized and the discriminant values are searched to find the class and class pair best discriminated by each measurement. This information is stored as part of the DI file header. Then the overall discriminant values (the M's) are ranked and stored in DI entry 1 (there is only 1 DI file entry in rank order format).

RODISP is called and an overall ranking is displayed. DSCRMEAS concludes by calling MENU to put up the list of measurement evaluation subsidiary commands.

NOTE: This command can run in excess measurement mode.

PROGRAM USAGE:

User types in 'DSCRMEAS'.

ALGORITHM / NOTES:

Some Definitions:

SUM = the sum of ... (addition)

NCLASS = the number of classes in the data set

NDIM = the vector dimensionality

X = all vectors in a class

X(k) = the kth component (measurement) of all vectors in a class

N(I) = the number of vectors in class I

MEAN(X(k),I) = the mean value of the kth component of the vectors in class I

STDDEV(X(k),I) = the standard deviation of the kth component of the vectors in class I

VAR(X(k),I) = the variance of the kth component of the vectors in class I
= [STDDEV (X(k), I)] **2
NOTE: (this value corresponds to the diagonal position (k,k) in the covariance matrix).

The following discriminant measures will be computed by 'DSCRMEAS'.

M(I,J,X(k)) = the discriminant measure for differentiating class I from class J using measurement X(k)

M(I,X(k)) = the discriminant measure for differentiating class I from all other classes using measurement X(k)

M(X(k)) = the discriminant measure for distinguishing all classes using measurement X(k)

Discriminant Measure Computations:

(1) M(I,J,X(k)) =

$$\frac{[\ MEAN(X(k),I) - MEAN(X(k),J)\ ]\ **2}{[(N(I)-1) * VAR(X(k),I) + (N(J)-1) * VAR(X(k),J)]}$$

for I = 1 to NCLASS - 1
J = I + 1 to NCLASS
k = 1 to NDIM

Note: this computation results in (NCLASS * (NCLASS - 1)) /2 discriminant measure vectors each of length NDIM

(2) M(I,X(k)) = SUM [M(I,J,X(k))]

for I = 1 to NCLASS
J = 1 to NCLASS; J not equal to I
k = 1 to NDIM

Note: this computation results in NCLASS discriminant measurement vectors each of length NDIM

(3) M(X(k)) = SUM [M(I,X(k))]

for I = 1 to NCLASS
k = 1 to NDIM

Note: this computation results in 1 discriminant measurement vector of length NDIM

FILES:

CM - communication
DI - display information
DV - display values
HS - history (indirectly used)
INSTRU.DAT (instrumentation dump) - (indirectly used)
OP - option (indirectly used)
TI - tree information (for CDS)
TL - tree list (indirectly used)
TV - tree vector (for CDS)

REFERENCED BY:

OLPARS User.

SUBPROGRAMS REFERENCED:

CMGCDS, CMGOTH, DIPNAM, DIPROE, DIPROH, DIPROI, DVGROE, DVPROE, DVPROH, ERASE, GLONOD, INSCHR, INSFLT, INSINI, INSINT, INSRET, MENU, OEXIT, OPENFX, OPENTR, PRECPT, RODISP, RSORTD, SETOPT, TIGCOP, TIGET, TIGMN, TIGVAR, TRMPUT, TVGVEC, UIDCMS

HISTORY:

designed by Kermit Klingbail July 31, 1978

programmed by Donna Morris December 11, 1981

PROGRAM NAME: DSCVTH

CATEGORY: Numerical Computation Algorithm

PROGRAM DESCRIPTION:

DSCVTH computes an N-dimensional vector and a threshold from the two 2-dimensional end points of a boundary line segment drawn on a two-dimensional subspace projection of an N-dimensional data set. This discriminant vector and threshold value are utilized in determining which side of a boundary a sample point falls.

PROGRAM USAGE:

```
CALL DSCVTH(X1,Y1,X2,Y2,XCONVX,YCONVX,NDIM,XPROJ,YPROJ,
            THRESH,DSCRMV)
```

INPUT ARGUMENTS:

X1, Y1 - REAL; the x and y coordinates of the first endpoint of the boundary line segment

X2, Y2 - REAL; the x and y coordinates of the second endpoint of the boundary line segment

XCONVX - REAL; the x coordinate of the convex point

YCONVX - REAL; the y coordinate of the convex point

NDIM - INTEGER; the dimensionality of the data set

XPROJ - REAL array (NDIM); x projection vector

YPROJ - REAL array (NDIM); y projection vector

OUTPUT ARGUMENTS:

THRESH - REAL; the threshold value

DSCRMV - REAL array (NDIM); the discriminant vector

ALGORITHM / NOTES:

Compute the vector d = (dx,dy) such that d is perpendicular to the boundary line segmen' For two lines to be perpendicular, one slope must be the inverted negation of the other slope. Since, by definition, the slope of the boundary line is (Y2-Y1)/(X2-X1), then the slope of the perpendicular must be -(X2-X1)/(Y2-Y1). Using a point-slope formula, we can find the equation of the vector. The x and y coordinates are in the form:

dx = Y2 - Y1
dy = - (X2 - X1)

Project the convex point and boundary endpoint onto d. The formula for the projection of c onto d is:

$$((d:c) / norm(d)^2) * d$$

where : represents the dot product and norm is the length of the vector. For our use,

$norm(d)^2 * d$ is a constant used in all computations, and thus can be ignored everywhere without affecting the results. Thus, the projection formula becomes a simple dot product, and we have:

pConvx = (Xconvx * dx) + (Yconvx * dy)

and

pEndPt = (X2 * dx) + (Y2 * dy)

To assure that the convex point actually falls within the convex boundary, the projected convex point value must be greater than or equal to the projected end-point value. If this is not the case, the slope of the vector d is negated and the projected endpoint is negated. Thus,

IF (projected convex point.lt.projected endpoint) THEN

Replace dx by -dx.

Replace dy by -dy.

Replace pEndPt by -pEndPt.

ENDIF

Place the projected endpoint value into the threshold value.

Compute the discriminant vector using the formula:
dscrm(i) = (dx * Xproj(i)) + (dy * Yproj(i))

RETURN

END

FILES:

instrumentation file

REFERENCED BY:

Two-space group logic and structure analysis subroutines

SUBPROGRAMS REFERENCED:

INSFLT, INSPGM, INSRET

HISTORY:

designed by Kermit Klingbail September 28, 1978

programmed by Jill King January 14, 1981

PROGRAM NAME: DSPROJ

CATEGORY: Display File Access Routine

PROGRAM DESCRIPTION:

DSPROJ projects a data set onto a projection vector or projection vectors for one-space or two-space displays. DSPROJ is used only in the S1 and S2 structure analysis commands. It operates class by class, projecting the vectors from a class onto the chosen projection vector(s). The results are stored in the DV file, the DI file entry for each class is updated and the original minimum and maximum projection coordinates are computed and stored in the DI file.

PROGRAM USAGE:

```
CALL DSPROJ(FIDTI,FIDTV,FIDPV,FIDDI,FIDDV,ET,SLTNO,
            ONETWO)
```

INPUT ARGUMENTS:

FIDTI - INTEGER; the file descriptor of the TI file

FIDTV - INTEGER; the file descriptor of the TV file

FIDPV - INTEGER; the file descriptor of the PV file

FIDDI - INTEGER; the file descriptor of the DI file

FIDDV - INTEGER; the file descriptor of the DV file

ET - INTEGER array (TABSIZ); the entry table of the data tree

SLTNO - INTEGER; the slot number of the current data node

ONETWO - INTEGER; 1 for one-space, 2 for two-space

ALGORITHM / NOTES:

Get the projection vector(s) from the PV file (PVGHDR, PVGENT)

Get the number of lowest nodes (classes), the entry slot numbers, and the names of the nodes (GLONOD).

DO WHILE (there are more classes to be projected).

Get a class (TIGCOP) and its mean vector (TIGMN).

Project the mean vector onto the projection vector(s) and store results in DV file.

DO WHILE (there are more vectors in the class).

Get a vector from that class (TVGVEC).

Project it onto the projection vector(s).

Update x(min), x(max), (y(min), y(max)).

Store projected vector in next DV file entry (DVPVEC).

ENDDO

Create DI file entry for this class (DIPENT).

ENDDO

Store x(min), x(max), (y(min), y(max)) values in DI file as original and current (DIPMAX).

Store total number of vectors in DI header (DIPHDI).

Update the next available entry pointer in the DV file header.

RETURN

FILES:

TI - tree information
TV - tree vector
PV - projection vector
DI - display information
DV - display value

REFERENCED BY:

All S1 and S2 commands except S1CRDV, S2CRDV

SUBPROGRAMS REFERENCED:

TIGCOP, TVGVEC, DVPVEC, DIPMAX, DIPHDI, DIPENT, PVGHDR, PVGENT, TIGMN, GLONOD, PROJCT, TIGNAM

HISTORY:

designed by Dave Birnbaum July 17, 1978

programmed by Donna Morris June 1,1979

PROGRAM NAME: DTRENAME

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

Change the name of an OLPARS data tree.

PROGRAM USAGE:

User types in 'DTRENAME'.

USER INTERACTION:

The name of the data tree to be changed is requested from the user, along with the new name of the tree.

FILES:

CM - communications
TL - tree list
TI - tree information
TV - tree vector
HS - history
OLPARS option
Instrumentation

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLOSFX, CLOSTR, CMGOTH, EQUALA, ERASE, INSINI, INSRET.
LGLTRE, MENU, OEXIT, OPENFX, OPENTR, PROMPT, RENAMT,
TLSRCH, TRMGET, TRMPUT

SEE ALSO:

LTRENAME

HISTORY:

designed by Steven Haehn Aug. 15, 1981

programmed by Steven Haehn Sept. 28, 1981

PROGRAM NAME: DVEC

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

DVEC allows the user to delete vectors (in a lowest node) under the current data set.

PROGRAM USAGE:

User types in 'DVEC'.

USER INTERACTION:

The user is first asked to name a lowest node from which vectors are to be deleted. Then the user is asked if

(1) all vectors
(2) a range of vectors
(3) or individual vectors

should be deleted.

If (2), an initial and last vector ID are requested to be entered.

If (3), a vector ID (for each vector to be deleted) is requested.

ALGORITHM / NOTES:

REPEAT

Erase screen and home cursor (ERASE).

Print the list of lowest nodes contained in the current data set (TRMPUT).

Ask user for the name of a lowest node (PROMPT).

Check the name against the list of lowest nodes for validity.

UNTIL (a correct name is entered or user quits)

Ask user for option (1), (2), or (3).

```
IF (option (2) is entered) THEN

    Ask user for two vector IDs representing the range of
    vectors to be deleted.

ELSEIF (option (3) is entered) THEN

    Ask user for individual vector IDs.

ENDIF

IF (option (1) is entered) THEN

    Delete all vectors (and data node) (REARR)

ELSEIF (option (2) or (3) is entered) THEN

    Locate vectors to be deleted.

    Delete the vector(s) by compressing the TV file entry

    IF (all vectors are deleted from node) THEN

        The data node must be deleted (REARR).

    ENDIF

    Change the following information in the node from
    which the vectors were deleted and in all nodes
    above that one:

    o   Number of vectors
    o   Means
    o   Covariances

ENDIF

Put up option list at user's terminal (MENU).

END
```

NOTE

$$nCov(ij) = \frac{(oCov(ij)+oMn(i)*oMn(j))*nVec-V(i)*V(j)}{nVec - 1} - nMn(i)*nMn(j)$$

where,

- nCov(ij) = new covariance of measurements i and j after vector deletion
- oCov(ij) = old covariance of measurements i and j before vector deletion
- nMn(i) = new class mean measurement after vector deletion
- oMn(i) = old class mean measurement before vector deletion
- nVec = number of vectors in data class
- V(i) = vector measurement

REFERENCED BY:

OLPARS user.

SUBPROGRAMS REFERENCED:

ADUPCM, CHKEXS, CMGCDS, CMGOTH, EQUALA, ERASE, GLONOD, INSFLT, INSINI, INSINT, INSRET, MENU, OEXIT, OPENFX, OPENTR, PROMPT, REARR, SBUPMC, SELCLS, TIGCOP, TIGCOV, TIGET, TIGMN, TIPCOP, TIPCOV, TIPMN, TRMGET, TRMPUT, TVGHDR, TVGVEC, TVPHDR, TVPVEC

HISTORY:

designed by David Birnbaum July 31, 1978

programmed by Steven Haehn July 24, 1981

PROGRAM NAME: DVGHDR

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DVGHDR gets the DV file header for one-space, two-space and rank order displays. It is passed the file descriptor (FID) of the DV file and an information array (INFO). The information array is both an input and output array. Non-negative values in INFO are replaced with the item referred to by the INFO array subscript (code). The codes are as follows:

CODE	ELEMENT IN DV HEADER
1	Display Code
2	OLPARS Option Number
3	Screen Coordinate Flag
4	Next Available Entry

PROGRAM USAGE:

CALL DVGHDR(FID,INFO)

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the DV file

INFO - INTEGER array (4); array indicating which items to retrieve. this array is overwritten with the values of the items retrieved.

FILES:

DV - display value

SUBPROGRAMS REFERENCED:

FGET,TRMPUT

SEE ALSO:

DVPHDR

HISTORY:

designed by Dave Birnbaum June 21, 1978

programmed by Donna Morris May 17,1979

PROGRAM NAME: DVGROE

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DVGROE retrieves an entry from the DV file when in rank order format.

PROGRAM USAGE:

```
CALL DVGROE(FIDDV,LENTNO,NDIM,VECTOR)
```

INPUT ARGUMENTS:

```
 FIDDV - INTEGER; the file descriptor of the DV file
LENTNO - INTEGER; entry number to be retrieved
  NDIM - INTEGER; data set dimensionality
```

OUTPUT ARGUMENTS:

VECTOR - REAL array (NDIM); DV file entry elements

FILES:

DV - display value

REFERENCED BY:

DSCRMEAS, PROBCONF, RANK, UNION

SUBPROGRAMS REFERENCED:

FGET, INSPGM, INSRET

SEE ALSO:

DVPROE

HISTORY:

designed by Kermit Klingbail October 4, 1978

programmed by Donna Morris October 16, 1980

PROGRAM NAME: DVGROH

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DVGROH gets elements of DV file header when in rank order format.

PROGRAM USAGE:

```
CALL DVGROH(FIDDV,DSPCOD,OPTNO)
```

INPUT ARGUMENTS:

FIDDV - INTEGER; the file descriptor of the DV file

OUTPUT ARGUMENTS:

DSPCOD - INTEGER; display code
OPTNO - INTEGER; option number

FILES:

DV - dislay value

REFERENCED BY:

PROBCONF

SUBPROGRAMS REFERENCED:

FGET, INSPGM, INSRET, OEXIT, TRMPUT

SEE ALSO:

DVPROH

HISTORY:

designed by Kermit Klingbail October 4, 1978

programmed by Donna Morris October 16, 1980

PROGRAM NAME: DVGVEC

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DVGVEC gets a DV file entry for one- or two-space displays. It is passed the file descriptor (FID) of the DV file, the value 1 or 2 in IDISP (for one-space or two-space) and an entry number (ENTNO). DVGVEC retrieves the entry in the DV file corresponding to ENTNO and places:

- The vector ID in ID.
- The x coordinate in XCOORD.
- The x screen coordinate in XSCRN.
- The y coordinate in YCOORD (for two-space only).
- The y screen coordinate in YSCRN (for two-space only).

PROGRAM USAGE:

```
CALL DVGVEC(FID,IDISP,ENTNO,ID,XCOORD,XSCRN,YCOORD,
     YSCRN)
```

INPUT ARGUMENTS:

```
   FID - INTEGER; the file descriptor of the DV file
 IDISP - INTEGER; one-space or two-space code
 ENTNO - INTEGER; the entry number
```

OUTPUT ARGUMENTS:

```
    ID - INTEGER; the vector descriptor
XCOORD - REAL;    the x coordinate
 XSCRN - INTEGER; the x screen coordinate
YCOORD - REAL;    the y coordinate
 YSCRN - INTEGER; the y screen coordinate
```

FILES:

DV - display value

SUBPROGRAMS REFERENCED:

FGET

OLPARS Program Specifications
DVGVEC

HISTORY:

designed by Dave Birnbaum June 21,1978

programmed by Donna Morris June 5,1979

PROGRAM NAME: DVPHDR

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DVPHDR places elements in a DV file header for one-space, two-space and rank order displays. It is passed the file descriptor (FID) of the DV file and an information array (INFO). The INFO array subscript represents the code of an item that can be referenced by DVPHDR. A negative value in the array indicates that the item specified by the code (subscript) is not to be written. The codes are as follows:

CODE	INFO INSERTED INTO
1	Display Code
2	OLPARS Option Number
3	Screen Coordinate Flag
4	Next Available Entry

PROGRAM USAGE:

CALL DVPHDR(FID,INFO)

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the DV file
INFO - INTEGER array(4); the information

FILES:

DV - display value

SUBPROGRAMS REFERENCED:

FPUT

HISTORY:

designed by Dave Birnbaum June 28, 1978

programmed by Donna Morris May 7,1979

PROGRAM NAME: DVPROE

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DVPROE stores data in an entry of the DV file in rank order format.

PROGRAM USAGE:

```
CALL DVPROE(FIDDV,LENTNO,NDIM,VECTOR)
```

INPUT ARGUMENTS:

```
 FIDDV - INTEGER; file descriptor of the DV file
LENTNO - INTEGER; entry number in which to store data
  NDIM - INTEGER; data set dimensionality
VECTOR - REAL array (NDIM); elements to be stored
```

FILES:

DV - display value

REFERENCED BY:

DSCRMEAS, PROBCONF

SUBPROGRAMS REFERENCED:

FPUT, INSPGM, INSRET

SEE ALSO:

DVGROE

HISTORY:

designed by Kermit Klingbail October 4, 1978

programmed by Donna Morris October 16, 1980

PROGRAM NAME: DVPROH

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DVPROH puts elements into header of DV file in rank order format.

PROGRAM USAGE:

```
CALL DVPROH(FIDDV,DSPCOD,OPTNO)
```

INPUT ARGUMENTS:

```
 FIDDV - INTEGER; file descriptor of the DV file
DSPCOD - INTEGER; display code
 OPTNO - INTEGER; option number
```

FILES:

DV - display value

REFERENCED BY:

DSCRMEAS, PROBCONF

SUBPROGRAMS REFERENCED:

FPUT, INSPGM, INSRET

SEE ALSO:

DVGROH

HISTORY:

designed by Kermit Klingbail October 4, 1978

programmed by Donna Morris October 16, 1980

PROGRAM NAME: DVPVEC

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

DVPVEC puts a DV file entry into a DV file for one- or two-space displays. It is passed the following information:

- FID - the file descriptor of the DV file.
- IDISP - A one or two indicating one-space or two-space.
- ENTNO - The entry number in DV of the entry to be written into.
- ID - The vector ID of the vector.
- XCOORD - The x coordinate projection.
- XSCRN - The x screen coordinate.
- YCOORD - The y coordinate.
- YSCRN - The y screen coordinate.

YCOORD and YSCRN are ignored for one-space displays.

PROGRAM USAGE:

```
CALL DVPVEC(FID,IDISP,ENTNO,ID,XCOORD,XSCRN,YCOORD,YSCRN)
```

INPUT ARGUMENTS:

```
   FID - INTEGER; the file descriptor of the DV file
 IDISP - INTEGER; one-space or two-space flag
 ENTNO - INTEGER; the DV file entry number
    ID - INTEGER; the vector descriptor
XCOORD - REAL; the x coordinate
 XSCRN - INTEGER; the x screen coordinate
YCOORD - REAL; the y coordinate
 YSCRN - INTEGER; the y screen coordinate
```

FILES:

DV - display value
Instrumentation files

SUBPROGRAMS REFERENCED:

FPUT

SEE ALSO:

DVGVEC

HISTORY:

designed by Dave Birnbaum June 21, 1978

programmed by David Tipton June 4, 1979

PROGRAM NAME: EIGENJ

CATEGORY: Numerical Computation Algorithm

PROGRAM DESCRIPTION:

EIGENJ computes the eigenvalues and eigenvectors for a symmetric matrix. It uses a threshold Jacobi method for symmetric matrices. The maximum dimension acceptable to EIGENJ is MAXDIM.

PROGRAM USAGE:

CALL EIGENJ(MATRIX,MDIM,NDIM,VECTOR,VALUES)

INPUT ARGUMENTS:

MATRIX - REAL array ((NDIM x (NDIM + 1))/2); the upper triangular part of the symmetric matrix read in the order: row 1, row 2, ... row NDIM

MDIM - INTEGER; the maximum order of the matrix

NDIM - INTEGER; the order of the matrix

OUTPUT ARGUMENTS:

VECTOR - REAL array (MDIM,NDIM); the eigenvectors

VALUES - REAL array (NDIM); the eigenvalues

ALGORITM/NOTES:

For further reference, see the chapter entitled "Jacobi's Method for Symmetrical Matrices" ((Carnahan, Luther, and Wilkes, 'APPLIED NUMERICAL METHODS' (Publisher: Wiley) Section 4.8, p.250 - 251).

REFERENCED BY:

EIGNLG, EIGNPV

HISTORY:

designed by David Birnbaum September 6, 1978

programmed by Donna Morris June 4, 1979

PROGRAM NAME: EIGLPV

CATEGORY: Display File Access Routine

PROGRAM DESCRIPTION:

EIGLPV calls EIGNLG to return in arrays the eigenvalues and eigenvectors at an incomplete logic node. EIGLPV then puts the eigenvectors and eigenvalues into the PV file.

PROGRAM USAGE:

```
CALL EIGLPV (FIDTI, FIDTV, FIDPV, NCLASS, CLIST,
             ENTAB, ALLVEC, NODNUM)
```

INPUT ARGUMENTS:

FIDTI - INTEGER; the file descriptor of the TI file

FIDTV - INTEGER; the file descriptor of the TV file

FIDPV - INTEGER; the file descriptor of the PV file

NCLASS - INTEGER; the number of classes at the incomplete logic node

CLIST - INTEGER array (NCLASS); the list of entry table slot numbers of the classes in the TI file

ET - INTEGER array (TABSIZ); the TI entry table

ALLVEC - LOGICAL; TRUE means all vectors, FALSE means only those vectors at the logic node

NODNUM - INTEGER; the logic node number

ALGORITHM / NOTES:

Get the dimensionality and number of measurements used (PVGHDR).

Get the measurement vector (PVGENT).

Find the eigenvalues and eigenvectors (EIGNLG).

Expand the first dimension of the eigenvector array to the length of the measurement vector if needed, putting zeroes in the entries corresponding to unused measurements.

Store eigenvectors in the PV file (PVPENT).

Sort eigenvalues in descending order and expand the array with zeroes if any measurements were eliminated (SORT).

Store eigenvalues in the PV file (PVPENT).

FILES:

PV - projection vector
TI - tree information
TV - tree vector

REFERENCED BY:

L1EIGV, L2EIGV, RECTLV

SUBPROGRAMS REFERENCED:

EIGNLG, PVGENT, PVGHDR, PVPENT, SORT

HISTORY:

designed by David J. Tipton January 9, 1981

programmed by David J. Tipton January 9, 1981

PROGRAM NAME: EIGNLG

CATEGORY: Display File Access Routine

PROGRAM DESCRIPTION:

EIGNLG computes the covariance of the set of vectors at an incomplete logic node. It then calls EIGENJ to compute the eigenvalues and eigenvectors and passes them back in EVAL and EVEC.

PROGRAM USAGE:

```
CALL EIGNLG (FIDTI, FIDTV, NDIM, NCLASS, CLIST, ET,
             ALLVEC, NUSED, MVEC, EVAL, EVEC)
```

INPUT ARGUMENTS:

FIDTI - INTEGER; the file descriptor of the TI file

FIDTV - INTEGER; the file descriptor of the TV file

NDIM - INTEGER; the dimension of the data set

NCLASS - INTEGER; the number of classes

CLIST - INTEGER array (NCLASS); the list of entry table slot numbers of the classes in the TI file

ET - INTEGER array (TABSIZ); the TI entry table

ALLVEC - LOGICAL; TRUE means all vectors, FALSE means only those vectors at the logic node

NODNUM - INTEGER; the logic node number

NUSED - INTEGER; the number of measurements not eliminated within the measurement vector

MVEC - INTEGER array (NDIM); the measurement vector

OUTPUT ARGUMENTS:

EVAL - REAL array (NDIM); the eigenvalues

EVEC - REAL array (NDIM x NDIM); the eigenvectors

ALGORITHM / NOTES:

```
IF (all vectors from the classes at the logic node are
to be used in the computation) THEN

   Get the mean of the class (TIGMN).

   Get covariance matrix of the class (TIGCOV).

   Get the number of vectors in the class (TIGCOP).

ELSE

   Compute covariance matrix of the class vectors
   that lie at the logic node (LOGMNC).

ENDIF

WHILE (there are more classes from the logic node)

   IF (all vectors from the classes at the logic node
       are to be used in the computation) then

      Get the mean of the class (TIGMN)

      Get the covariance matrix of the class (TIGCOV)

      Get the number of vectors in the class (TIGCOP)

   ELSE

      Compute covariance matrix of the class vectors
      that lie at the logic node (LOGMNC).

   ENDIF

   Add to covariance matrix computation for entire set
   of classes (COMBIN).

ENDWHILE

Eliminate the rows and columns of the covariance
matrix that correspond to the measurements to be
eliminated (COLAPS).

Compute eigenvalues and eigenvectors (EIGENJ).

RETURN
```

REFERENCED BY:

L1EIGV, L2EIGV, RECTLV .

SUBPROGRAMS REFERENCED:

COLAPS, COMBIN, EIGENJ, LOGMNC
TIGCOP, TIGCOV, TIGMN

HISTORY:

designed by David A. Birdbaum September 13, 1978

programmed by David Tipton January 7, 1981

PROGRAM NAME: EIGNPV

CATEGORY: Display File Access Routine

PROGRAM DESCRIPTION:

EIGNPV is passed the file descriptors of the TI and PV files and the entry number of the node in the data set where the covariance matrix can be found. EIGNPV retrieves the covariance matrix, computes the eignevalues and eigenvectors (via EIGENJ) and sets up the PV file for the subroutines that will follow.

PROGRAM USAGE:

```
CALL EIGNPV(FIDTI,FIDPV,ENTNO)
```

INPUT ARGUMENTS:

FIDTI - INTEGER; file descriptor of the TI file
FIDPV - INTEGER; file descriptor of the PV file
ENTNO - INTEGER; the entry number in the TI file of the data set

ALGORITHM / NOTES:

Determine what measurements are to be included in the computation.

Get the covariance matrix. If measurements are to be eliminated from the computation, remove corresponding rows and columns from the covariance matrix (COLAPS).

Compute eigenvalues and eigenvectors (EIGENJ).

Place them into the PV file (PVPENT), putting zeros in the ignored measurements.

RETURN

FILES:

TI - tree information
PV - projection vector

REFERENCED BY:

S1EIGV, S2EIGV

SUBPROGRAMS REFERENCED:

COLAPS,EIGENJ,PVGENT,PVGHDR,PVPENT,TIGCOV

HISTORY:

designed by Dave Birnbaum September 6, 1978

programmed by Donna Morris May 17,1979

PROGRAM NAME: EIGNXFRM

CATEGORY: Transformation Command

PROGRAM DESCRIPTION:

EIGNXFRM generates a new tree of equal or lower dimensionality by transforming the selected data set, using the eigenvectors which correspond to the selected eigenvalues (the selected eigenvalues consist of the threshold eigenvalue and all eigenvalues above it in the list).

PROGRAM USAGE:

User types in 'EIGNXFRM'.

USER INTERACTION:

The user is asked if he wants a line printer copy of the eigenvalues and eigenvectors. He is asked to type in a threshold and a new treename.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Call ESETUP to open the CM, DI, TL, PV, and current data set tree files. ESETUP will call UIXFRM to ask the user for the name of the new data tree to be created.

Compute the eigenvalues and eigenvectors (EIGNPV).

Display the decreasing list of eigenvalues on the screen and ask the user if (s)he wants a lineprinter listing (EIGPRT).

Obtain threshold value from user (GTHRSH) and determine which eigenvectors to project on.

Call EXFRM to perform the transformation.

Save the transformation matrix if the user so desires (SAVMAT).

Put up option list at user's terminal (MENU).

END

To compute new mean vectors and covariance matrices let:

n = dimension of the original data set

m = number of eigenvalues chosen (the dimension of the new data set)

x' = a mean vector of a node in the old tree

C = the covariance matrix of a node in the old tree

A = the m by n matrix whose rows consist of the m eigenvectors corresponding to the m chosen eigenvalues

Then the new mean vector and new covariance matrix of the node are given by:

Ax' and AC times the transpose of A

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLOSFX, CLOSTR, EIGNPV, EIGPRT, ERASE, ESETUP
EXFRM, GTHRSH, INSINI, INSRET, MENU, OEXIT, OPENFX
PVGENT, SAVMAT, TRMPUT

SEE ALSO:

MATXFRM, NORMXFRM

HISTORY:

designed by Dave Birnbaum September 8, 1978

programmed by Donna Morris January 1, 1982

OLPARS Program Specifications
EIGPRT

PROGRAM NAME: EIGPRT

CATEGORY: Display File Access Routine

PROGRAM DESCRIPTION:

EIGPRT is passed the file descriptors of the PV and CM files. It displays the eigenvalues from the PV file and asks the user if he wishes a printout of them. If EIGPRT is set to TRUE, the calling program quits.

PROGRAM USAGE:

```
LOGICAL EIGPRT
        .
        .
        .
IF(EIGPRT(FIDPV,FIDCM))
```

INPUT ARGUMENTS:

FIDPV - INTEGER; the file descriptor of the PV file
FIDCM - INTEGER; the file descriptor of the CM file

ALGORITHM / NOTES:

Display eigenvalues on terminal screen (PVGHDR, PVGENT, TRMPUT)

IF (printout of eigenvalues and eigenvectors is required (PROMPT, TRMGET)) THEN

    Set up output file for printout (OPENS, PRINTR, FILPUT)

ENDIF

RETURN

FILES:

PV - projection vector

REFERENCED BY:

S1EIGV, S2EIGV, L1EIGV, L2EIGV, REPROJECT

SUBPROGRAMS REFERENCED:

CLOSE, CMGCDS, CMGOTH, CMGSCN, EQUALA,
ERASE, FILPUT, INSPGM, INSRET, OPENS,
PRINTR, PROMPT, PVGENT, PVGHDR, TRMGET,
TRMPUT

HISTORY:

designed by Dave Birnbaum June 19, 1978

programmed by Donna Morris June 10, 1979

PROGRAM NAME: ELIFRE

CATEGORY: Logic Tree File Access Routine

PROGRAM DESCRIPTION:

ELIFRE enters a Logic Information entry into the free list of the LI file.

PROGRAM USAGE:

```
CALL ELIFRE (FDLI, LOGNDS, NODCNT)
```

INPUT ARGUMENTS:

FDLI - INTEGER; Logic Information file descriptor

LOGNDS - INTEGER array(MAXCLS*(MAXCLS+1)/2);
list of logic nodes to be entered
into 'free' list

NODCNT - INTEGER; the number of logic nodes in 'LOGNDS'

FILES:

LI - logic information

REFERENCED BY:

KILLND

SUBPROGRAMS REFERENCED:

INSINT, INSPGM, INSRET, LIGCOP, LIPCOP, LIPLOG, OEXIT, TRMPUT

SEE ALSO:

DLREGN

HISTORY:

designed by Steven Haehn Mar. 5, 1981

programmed by Steven Haehn Mar. 11, 1981

PROGRAM NAME: ELIMEA

CATEGORY: Display File Access Routine

PROGRAM DESCRIPTION:

ELIMEA is called by certain S1 and S2 commands to ask the user if measurements are to be ignored in the computation of the projection vectors. ELIMEA sets the measurement vector in the PV file.

PROGRAM USAGE:

```
LOGICAL ELIMEA
IF(ELIMEA(FIDPV,FIDCM,NDIM))
```

INPUT ARGUMENTS:

FIDPV - INTEGER; the file descriptor of the PV file
FIDCM - INTEGER; the file descriptor of the CM file
NDIM - INTEGER; the dimension of the data set

ALGORITHM / NOTES:

IF (user wishes to eliminate measurements) THEN

Ask for the numbers of the measurements to be eliminated (PROMPT).

Set the measurement vector in the PV file (PVPENT).

ELSE

Set the coordinates of the measurement vector to all ones (PVPENT).

ENDIF

Set the number of measurements used in the PV file header.

SUBPROGRAMS REFERENCED:

CMGOTH,EQUALA,PVPENT,PVPHDR,PROMPT,TRMGET

DIAGNOSTICS:

ELIMEA is set to TRUE if the user wishes to quit the program or if the user eliminates all of the measurements.

HISTORY:

designed by Dave Birnbaum August 9, 1978

programmed by Donna Morris May 22,1979

PROGRAM NAME: EQUALA

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

EQUALA compares two strings for equality. If the strings are equal, a true value is returned as the function value. If the strings are not equal, i.e., don't match in length or characters, a false value is returned as the function value.

LEN1 and LEN2 specify the length of the character strings being tested for equality. If both LEN1 and LEN2 are zero, then the corresponding strings (see below) must have an end-of-string (EOS) symbol attached to them (EOS in OLPARS is a zero). In order for the strings to be considered equal by equala, the EOS must occur in the same position in both strings.

EXACT is a logical switch indicating whether or not the 'case' of the characters being compared is important.

Note, the characters are stored one per integer word.

PROGRAM USAGE:

```
LOGICAL EQUALA
      .
      .
      .
IF(EQUALA(STRNG1,LEN1,STRNG2,LEN2,EXACT))
```

INPUT ARGUMENTS:

STRNG1 - INTEGER array (MAXLIN + 1); the first character string

LEN1 - INTEGER; the length of the first character string

STRNG2 - INTEGER array (MAXLIN + 1); the second character string

LEN2 - INTEGER; the length of the second character string

EXACT - LOGICAL; 'true' means the 'case' of the alphabetic characters must match too (i.e., lower case a is not equal to upper case A).
'false' means the case of the alphabetic characters being compared is unimportant (i.e., lower case a equals upper case A).

OUTPUT ARGUMENTS:

EQUALA - LOGICAL; 'true' when the two character strings are equal, 'false' when they are not equal

REFERENCED BY:

UPPER LEVEL ROUTINES

SUBPROGRAMS REFERENCED:

VALUEA

HISTORY:

designed by Steve Haehn February 26,1979

programmed by Steve Haehn February 26,1979

PROGRAM NAME: EQUALS

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

EQUALS compares two strings for equality. If the strings are equal, a true value is returned as the function value. If the strings are not equal, i.e., don't match in length or characters, a false value is returned as the function value.

LEN1 and LEN2 specify the length of the character strings being tested for equality. If both LEN1 and LEN2 are zero, then the corresponding strings (see below) must have an end-of-string (EOS) symbol attached to them (EOS in OLPARS is a zero). In order for the strings to be considered equal by equals, the EOS must occur in the same position in both strings.

EXACT is a logical switch indicating whether or not the 'case' of the characters being compared is important.

Note, the characters in the string are stored in a FORTRAN Hollerith field. Under RSX-11M FORTRAN IV, a Hollerith character string is stored 2 characters to one integer word.

PROGRAM USAGE:

```
LOGICAL EQUALS
        .
        .
        .
IF(EQUALS(STRNG1,LEN1,STRNG2,LEN2,EXACT))
```

INPUT ARGUMENTS:

STRNG1 - INTEGER array (MAXLIN + 1); the first character string

LEN1 - INTEGER; the length of the first character string

STRNG2 - INTEGER array (MAXLIN + 1); the second character string

LEN2 - INTEGER; the length of the second character string

EXACT - LOGICAL; 'true' means the 'case' of the alphabetic characters must match too (i.e., lower case a is not equal to upper case A).
'false' means the case of the alphabetic characters being compared is unimportant (i.e., lower case a equals upper case A).

OUTPUT ARGUMENTS:

EQUALS - LOGICAL; 'true' when the two character strings are equal, 'false' when they are not equal

REFERENCED BY:

UPPER LEVEL ROUTINES

SUBPROGRAMS REFERENCED:

VALUES

HISTORY:

designed by Steve Haehn February 26,1979

programmed by Steve Haehn February 26,1979

PROGRAM NAME: ERASE

CATEGORY: Terminal I/O (graphic, system dependent)

PROGRAM DESCRIPTION:

ERASE erases the terminal screen and homes the cursor if not automatically done. (Homing the cursor means to move it to the upper left-hand corner of the screen.)

PROGRAM USAGE:

CALL ERASE

HISTORY:

designed by Steven Haehn April 5, 1978

programmed by Steven Haehn April 1, 1981

OLPARS Program Specifications
EV1SP

PROGRAM NAME: EV1SP

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

EV1SP (evaluate one-space logic) takes an individual vector and "passes" it against the logic resulting from boundaries drawn on a one-space projection. It is also used by RESTRUCT to determine which region a vector falls in.

PROGRAM USAGE:

```
CALL EV1SP(NBOUND,NDIM,VECTOR,DSCRMV,THRSHL,THRSHR,
           REGION)
```

INPUT ARGUMENTS:

NBOUND - INTEGER; the number of boundaries drawn

NDIM - INTEGER; the dimension of the current data set

VECTOR - REAL array (NDIM); the vector to be evaluated

DSCRMV - REAL array (NDIM); the discriminant vector

THRSHL - REAL; the left-most threshold value

THRSHR - REAL; the right-most threshold value

OUTPUT ARGUMENTS:

REGION - INTEGER; the region in which the vector lies
REGION = LEFT (or 1) if the vector falls to the left of the left-most boundary
REGION = RIGHT (or 2) if the vector falls to the right of the right-most boundary
REGION = MIDDLE (or 3) if the vector falls between the two boundaries
(ignored for one boundary)

ALGORITHM / NOTES:

```
IF (right threshold value .lt. left threshold value) THEN
   Reverse the two values.
ENDIF
Compute the dot product of vector and discriminant.
IF (dot product .ge. right threshold value) THEN
    The vector lies in the right region.
ELSE
   IF ((number of boundaries.eq.2) and (the dot product
        .gt. the left threshold value)) THEN
      The vector lies in the middle region.
   ELSE
      The vector lies in the left region.
   ENDIF
ENDIF
RETURN
```

FILES:

instrumentation file

REFERENCED BY:

CRLOG1, RESTRUCT

SUBPROGRAMS REFERENCED:

INSINT, INSPGM, INSRET

SEE ALSO:

SU1SP

HISTORY:

designed by Kermit Klingbail October 3, 1978

programmed by Jill King January 26, 1981

PROGRAM NAME: EV2SP

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

EV2SP (evaluate two-space logic) takes an individual vector and "passes" it against the logic resulting from boundaries drawn on a two-space projection. EV2SP is also used by RESTRUCT to determine in which region a vector lies.

PROGRAM USAGE:

```
CALL EV2SP(NBOUND,NSEGM1,TNSEGM,NDIM,VECTOR,DSCRMV,
           THRESH,REGION)
```

INPUT ARGUMENTS:

NBOUND - INTEGER; the number of boundaries drawn

NSEGM1 - INTEGER; the number of line segments in the first boundary

TNSEGM - INTEGER; the total number of line segments contained in both boundaries combined

NDIM - INTEGER; the dimension of the current data set

VECTOR - REAL array (NDIM); the vector to be evaluated

DSCRMV - REAL array (EXMLIM,MAXSEG); the discriminant vector for each line segment of each boundary, where MAXSEG is the maximum number of boundary line segments allowed in two boundaries and thus is equal to 10

THRESH - REAL array (TNSEGM); the threshold values for each line segment of each boundary

OUTPUT ARGUMENTS:

REGION - INTEGER; the region in which the vector lies
REGION = LEFT (or 1) if the vector falls on the convex side of the first boundary
REGION = RIGHT (or 2) if the vector falls on the convex side of the second boundary
REGION = MIDDLE (or 3) if vector falls in the excess region

ALGORITHM / NOTES:

DO i=1,number of line segments in the first boundary

Compute the dot product of the vector and the discriminant of the ith boundary line segment.

IF (the dot product is less than its corresponding threshold value) THEN

BREAK

ENDIF

ENDDO

IF (each dot product is greater than or equal to its corresponding threshold value) THEN

The vector lies in the LEFT region.

ELSEIF (the number of boundaries equals one) THEN

The vector lies in the RIGHT region.

ELSE

DO i=1,number of line segments in the second boundary

Compute the dot product of the vector and the discriminant of the ith boundary line segment.

```
            IF (the dot product is less than its corresponding
               threshold value)  THEN

               The vector lies in the MIDDLE region.

               BREAK

            ELSEIF (this is last time through loop) THEN

               The vector lies in the RIGHT region.

            ENDIF

         ENDDO

      ENDIF

      RETURN
```

FILES:

instrumentation file

REFERENCED BY:

CRLOG2, RESTRUCT

SUBPROGRAMS REFERENCED:

INSINT, INSPGM, INSRET

SEE ALSO:

SU2SP

HISTORY:

designed by Kermit Klingbail October 3, 1978

programmed by Jill King January 14, 1981

PROGRAM NAME: EVALBM

CATEGORY: Logic Design Routine (system dependent)

PROGRAM DESCRIPTION:

EVALBM evaluates the bit map, found in a logic information (LI) file entry, that represents the number of classes present (residing) at a specific logic node. Depending on the directional flag setting, EVALBM will either create a list of subscript pointers from the bit map, or create the bit map from a list of subscript pointers.

PROGRAM USAGE:

```
CALL EVALBM(BITMAP,DIRECF,CPRSNT,NCPRS)
```

INPUT ARGUMENTS:

BITMAP - REAL array (2); bit map representing the classes residing at a logic node (input argument when DIRECF is 'true')

DIRECF - LOGICAL; directional flag indicating which direction evaluation should occur (true; BITMAP->CPRSNT, false; BITMAP<-CPRSNT)

CPRSNT - INTEGER array (50); a list of subscripts (pointers) designating classes present at a logic node (input argument when DIRECF is 'false')

NCPRS - INTEGER; the number of classes present at a node (the number of bits turned on in BITMAP when DIRECT is 'true' or the number of subscripts found in CPRSNT when DIRECF is 'false')

OUTPUT ARGUMENTS:

BITMAP - REAL array (2); same as above (however, is output argument when DIRECF is 'false')

CPRSNT - INTEGER array (50); same as above (however, is output argument when DIRECF is 'true')

ALGORITHM/NOTES:

Users of this program should use the variables 'TO' and 'FROM' to represent the logical constants 'true' and 'false,' respectively. This will enable you to read your program easier (i.e., know which direction the evaluation is occurring), e.g.,

```
CALL EVALBM(BITMAP,TO,CPRSNT,NCPRS)
CALL EVALBM(BITMAP,FROM,CPRSNT,NCPRS)
```

Also, the bitmap is interpreted differently when only one class is present at the given logic node. The first element of the bitmap points directly to the class name found in the header.

REFERENCED BY:

NAMELOG, GCLIST

HISTORY:

designed by Steve Haehn September 15, 1978

programmed Donna Morris August 10, 1980

PROGRAM NAME: EVALU8

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

EVALU8 evalutes a vector against a logic. It returns the logic node number to which the vector was assigned and any pertinent error information.

PROGRAM USAGE:

```
LOGICAL EVALU8

        .
        .
        .

IF(EVALU8(FDLI,FDLV,NDIM,VECTOR,NODNUM,INDREJ,REJNOD,
            ERRINF))
```

INPUT ARGUMENTS:

FDLI - INTEGER; the file descriptor of the LI file

FDLV - INTEGER; the file descriptor of the LV file

NDIM - INTEGER; the vector dimensionality

VECTOR - REAL array (MAXDIM); the vector to be evaluated

OUTPUT ARGUMENTS:

NODNUM - INTEGER; the node number to which the vector was assigned

INDREJ - LOGICAL; TRUE means that the vector was rejected by an independent reject strategy at NODNUM. Otherwise 'INDREJ' is FALSE.

REJNOD - LOGICAL; TRUE means that NODNUM is a reject node

ERRINF - REAL array (MAXCLS + 1); the error information passed by back routines PAIRLG, NMDIST, and CLDBVC. ERRINF(1) will be the logic code indicating the type of logic creating the error information.

EVALU8 - LOGICAL; returns FALSE if logic evaluation cannot continue.

ALGORITHM / NOTES:

NOTE

9/16/81 - EVALU8 is not complete. BOOLEAN logic, CLOSED DECISION BOUNDARY logic, and NEAREST NEIGHBOR logic sections need to be written, along with some of the subroutines that these sections call. The algorithm below, however, will refer to these logics in order to provide a design for future coding of these sections.

```
Set NODNUM = 1 (NODNUM will contain the logic node
number at which the vector resides at any point in
the evaluation process.  It is initialized at 1
corresponding to the senior node of the logic tree).

Initialize INDREJ and REJNOD to FALSE.

Get the LI file entry for NODNUM.

REPEAT

    IF (there is an independent reject strategy at
        NODNUM) THEN

        Get its first entry (GTRGNI).

        Get the Boolean statement (GBOOLS).

        Test the vector (INTRPB).

        IF (the vector was rejected by the strategy) THEN

            Set INDREJ = TRUE.

            RETURN

        ENDIF

    ENDIF

    IF (logic type = 1 (pairwise)) THEN

        IF (the logic information for NODNUM is not
            currently stored in the Fisher common area
            FSHBLK) THEN

            Set up the fisher pairwise common area
            (SUFISH).
```

```
            Indicate that the information for NODNUM is
            currently in the fisher common area.

        ENDIF

        Call PAIRLG to evaluate the pairwise logic on the
        vector.  PAIRLG returns an index into the array
        of logic node numbers (assigned logic nodes)
        underneath the fisher node.  If the node to which
        the vector was assigned is a reject node, PAIRLG
        will set REJNOD to TRUE.  PAIRLG also returns, as
        error information, the vote counts (one vote
        count for each class at the fisher node).

        Use the index returned by PAIRLG for determining
        the logic node number to which the vector was
        assigned.  set NODNUM to this number.

        ERRINF(1) = FISHER PAIRWISE LOGIC

        BREAK

    ELSEIF (logic type = 2 (group logic) and subtype = 1
    (one-space)) THEN

        IF (the logic information for NODNUM
            (discriminant and thresholds) is not
            currently stored in the one-space group
            logic common area GP1BLK) THEN

            Set up the one-space group logic common
            area (SU1SP)

            Indicate that the information for NODNUM is
            currently in the one-space group logic
            common area.

        ENDIF

        Call EV1SP to determine which region the vector
        falls in.

        Determine which node number is associated with
        this region (look at 'logic nodes associated
        with regions 1,2, and 3 array' in the common
        area).

        Set NODNUM = this node number.

    ELSEIF (logic type = 2 (group logic) and subtype = 2
            (two-space)) THEN
```

```
    IF (the logic information for NODNUM
        (discriminant and threshold) is not
        currently stored in the two-space group
        logic common area GP2BLK) THEN

        Set up the two-space group logic common
        area (SU2SP)

        Indicate that the information for NODNUM is
        currently in the two-space group logic
        common area.

    ENDIF

    Call EV2SP to determine which region the vector
    falls in.

    Determine which node number is associated with
    this region (look at 'logic nodes associated
    with regions 1,2, and 3 array' in the common
    area).

    Set NODNUM = to this node number.

ELSEIF (logic type = 2 (group logic) and subtype = 3
(Boolean)) THEN

    Get Boolean statement from logic block (GTRGNI,
    GBOOLS).

    Evaluate vector against it (INTRPB) to get TRUE
    or FALSE.

    Set NODNUM = corresponding node number.

    IF (NODNUM is a reject node) THEN

        Set REJNOD = TRUE.

    ENDIF

ELSEIF (logic type = 3 (NMV)) THEN

    IF (the logic information for NODNUM is not
        currently stored in the NMV common area
        NMVBLK) THEN

        Set up the NMV common area (SUNMV).

        Indicate that the information for NODNUM is
        currently in the NMV common area.

    ENDIF
```

Call NMDIST to evaluate the NMV logic on the vector. NMDIST returns an index into the array of logic node numbers (assigned logic nodes) underneath the NMV node. If the node to which the vector was assigned is a reject node, NMV will set REJNOD to TRUE. NMV also returns, as error information, the 'distances' of the vector from each class at the NMV node.

ELSEIF (logic type = 4 (closed decision boundary)) THEN

Call CLDBVC to evaluate the closed decision boundary logic. CLDBVC returns the logic node number and an array of distances.

ELSEIF (logic type = 5 (Nearest Neighbor)) THEN

CALL EVALNN which does the following:

IF (the logic information for NODNUM is not currently stored in the nearest neighbor logic block)THEN

Write logic information to the nearest neighbor common area.

Open the reference pattern TI file (OPENTR).

Check that the reference pattern classes are in the same order as the classes at the logic node.

Write TI reference pattern counts and pointers to the nearest neighbor common area.

Close the reference pattern TI file and open the TV file (CLOSTR and OPENTR).

ENDIF

Call NNBREV to evaluate the vector.

End of EVALNN algorithm.

ENDIF

Get LI file entry for logic node NODNUM.

UNTIL (vector resides at a lowest logic node, i.e., one with only one assigned class, or a reject node)

```
        IF (there is an independent reject at NODNUM) THEN

            Get its first entry (GTRGNI).

            Get its Boolean statement (GBOOLS).

            Test the vector (INTRPB).

            IF (vector is rejected) THEN

                Set INDREJ = TRUE.

            ENDIF

        ENDIF

        RETURN
```

SUBPROGRAMS REFERENCED:

CLDBVC, EV1SP, EV2SP, EVALNN, GBOOLS, GTRGNI, INSINT, INSPGM, INSRET, INTRPB, LIGENT, NMDIST, NNCLAS, PAIRLG, SUFISH, SUNMV, SU1SP, SU2SP

HISTORY:

designed by Dave Birnbaum October 13, 1978

programmed by Donna Morris September 16, 1981

PROGRAM NAME: EXFRM

CATEGORY: TRANSFORMATION ROUTINE

PROGRAM DESCRIPTION:

EXFRM produces a new data tree which is an eigenvector transformation of the current data set. The transformation procedure involves multiplying each vector by the eigenvector transformation matrix and adjusting the means and covariances of all nodes.

PROGRAM USAGE:

```
LOGICAL EXFRM
        .
        .
        .
IF (EXFRM(CDTREE,CRSLT,NDTREE,NDIM,NOEIGV,MATRIX))
```

INPUT ARGUMENTS:

CDTREE - INTEGER array (TRELEN); the current data tree name

CRSLT - INTEGER; the TI file entry table slot number of the current node

NDTREE - INTEGER array (TRELEN); the new data tree name

NDIM - INTEGER; the dimension of the current data set and the new tree

NOEIGV - INTEGER; the number of eigenvalues used to compute the eigenvector transformation matrix being used in the creation of the new data tree

MATRIX - REAL array (MAXDIM,NOEIGV); the eigenvector transformation matrix which will be used to transform the current data set

OUTPUT ARGUMENTS:

EXFRM - LOGICAL; if 'true', indicates the transformation was successful. if 'false' indicates the calling routine should exit.

ALGORITHM / NOTES:

A key feature of the following algorithm is that the entry table slot numbers with which nodes are identified do not change in the new tree. This way a minimum number of pointers in the TI file entries need to be adjusted. (See section in Programmer's Reference Manual on 'Creating New Trees from Old Trees')

Open the current data set tree files and create the new data tree files (OPENTR, CREATR).

Read in the counts and pointers of the current node (senior node in the new tree).

Save the total number of vectors in the current tree.

INTIALIZE:

```
number of nodes = 0
node level = 1
save current node level
vector pointer in the new tree = 1
set senior node name to be '****'
parent pointer = 0
sibling pointer = 0
```

REPEAT

```
Number of nodes = number of nodes + 1

Transfer the TI file entry of the node.

Adjust the node level if necessary.

Entry table slot number of the current
node = Number of nodes.

IF (node is a lowest node) THEN

   DO I = 1,number of vectors at this node

      Retrieve and transform the vector
      (TVGVEC,TVPVEC).

    ENDDO

    Set vector pointer.
```

IF (this is the first lowest node encountered)

Set the lowest node back pointer to -1.

ENDIF

ENDIF

Retrieve and adjust the mean array (TIGMN,TIPMN).

Retrieve the covariance array and transform it into a variance array (TIGCOV).

Perform an eigenvector transformation on the variance array (MXFRM).

Transform the new variance array into a covariance array and store it (TIPCOV).

UNTIL (there are no more nodes to be transformed)

Set the lowest node link pointer of the last lowest node to -1.

Write the new TI and TV file headers (TIPHDR,TVPHDR).

Senior node pointer in TI header equals the slot number of the current data set node.

FILES:

TI - Tree Information (current and new tree) file
TV - Tree Vector (current and new tree) file
instrumentation file

REFERENCED BY:

MATXFRM, NORMXFRM

SUBPROGRAMS REFERENCED:

CLOSTR, CREATR, GNXTND, INSFLT, INSINT, INSPGM,
INSRET, MXFRM, OPENTR, TIGCAP, TIPET, TIPHDR
TIPMN, TIPNAM, TRMPUT, TVGVEC, TVPHDR, TVPVEC

HISTORY:

designed by Jill King April 22, 1981

programmed by Jill King April 22, 1981

PROGRAM NAME: EXPAND

CATEGORY: UTILITY SUBROUTINE

PROGRAM DESCRIPTION:

EXPAND takes a lower triangular matrix that has been COLAPSed, and expands it out to the specified size. The measurements added are zeros. They are added in the rows and columns which are indicated by zeros in the mask vector.

PROGRAM USAGE:

```
CALL EXPAND(OLDMAT,NEWMAT,OLDDIM,NEWDIM,MVEC)
```

INPUT ARGUMENTS:

OLDMAT - REAL array((OLDDIM*(OLDDIM+1))/2); The old matrix to be expanded.

OLDDIM - INTEGER; the exact dimensionality of the old matrix.

NEWDIM - INTEGER; the exact dimensionality of the new matrix.

MVEC - INTEGER array(NEWDIM); The mask vector.
1 - use old measurement
0 - add a zero

OUTPUT ARGUMENTS:

NEWMAT - REAL array((NEWDIM*(NEWDIM+1))/2); The NEW expanded matrix. NEWMAT and OLDMAT can be the same.

OLPARS Program Specifications
EXPAND

ALGORITHM / NOTES:

```
        DO UNTIL (all new values have been put in)
           IF (the row or column is not in the old matrix) THEN
              NEWMAT(index) = 0.0
           ELSE
              NEWMAT(index) = OLDMAT(oldindex)
           ENDIF
        ENDDO
```

REFERENCED BY:

NMVBSU

SEE ALSO:

COLAPS

HISTORY:

designed by John W. Tenney March 26, 1981

programmed by John W. Tenney March 26, 1981

PROGRAM NAME: EXT1ST

CATEGORY: Utility Subroutine

PROGRAM DESCRIPTION:

EXT1ST is passed two arrays containing the 'x' and 'y' points of a boundary (BNDYX,BNDYY), the border points in screen coordinate value (XMIN,YMIN,XMAX, YMAX), and an index into the array of points (PNTPOS).

Using the index to the present point, a new first point is found that lies on a border - an extension of the present boundary segment.

PROGRAM USAGE:

```
CALL EXT1ST (NEWX,NEWY,BNDYX,BNDYY,XMIN,YMIN,XMAX,YMAX,
             FIRSTX,FIRSTY,PNTPOS)
```

INPUT ARGUMENTS:

BNDYX - REAL; array of 'x' boundary points in a boundary

BNDYY - REAL; array of 'y' boundary points in a boundary

XMIN - REAL; minimum 'x' screen coordinate

YMIN - REAL; minimum 'y' screen coordinate

XMAX - REAL; maximum 'x' screen coordinate

YMAX - REAL; maximum 'y' screen coordinate

PNTPOS - INTEGER; the current point in the boundary

OUTPUT ARGUMENTS:

NEWX - REAL; 'x' coordinate returned from 'rXint'

NEWY - REAL; 'y' coordinate returned from 'rYint'

FIRSTX - REAL; the 'x' coordinate that has been extended to the border

FIRSTY - REAL; the 'y' coordinate that has been extended to the border

OLPARS Program Specifications
EXT1ST

ALGORITHM / NOTES:

Determine where the 'x' coordinates are in relation to the border (window)

Determine where the 'y' coordinates are in relation ot the border (window)

Calculate the new first point (x,y)

FILES:

Instrumentation file

REFERENCED BY:

REDRAW

SUBPROGRAMS REFERENCED:

INSINT, INSPGM, INSRET, RXINT, RYINT

HISTORY:

designed by Mark Maginn September 15, 1980

programmed by Mark Maginn September 15, 1980

PROGRAM NAME: EXTLST

CATEGORY: Utility Subroutine

PROGRAM DESCRIPTION:

EXTLST is passed two arrays containing the 'x' and 'y' points of a boundary (BNDYX,BNDYY), the border points in screen coordinate value (XMIN,YMIN,XMAX, YMAX), and an index into the array of points (PNTPOS).

Using the index to the present point, a new last point is found that lies on a border - an extension of the present boundary segment.

PROGRAM USAGE:

```
CALL EXTLST (NEWX,NEWY,BNDYX,BNDYY,XMIN,YMIN,XMAX,YMAX,
             FINALX,FINALY,PNTPOS)
```

INPUT ARGUMENTS:

BNDYX - REAL; array of 'x' boundary points in a boundary

BNDYY - REAL; array of 'y' boundary points in a boundary

XMIN - REAL; minimum 'x' screen coordinate

YMIN - REAL; minimum 'y' screen coordinate

XMAX - REAL; maximum 'x' screen coordinate

YMAX - REAL; maximum 'y' screen coordinate

PNTPOS - INTEGER; the current point in the boundary

OUTPUT ARGUMENTS:

NEWX - REAL; 'x' coordinate returned from 'rXint'

NEWY - REAL; 'y' coordinate returned from 'rYint'

FINALX - REAL; the 'x' coordinate that has been extended to the border

FINALY - REAL; the 'y' coordinate that has been extended to the border

ALGORITHM / NOTES:

Determine where the 'x' coordinates are in relation to the border (window)

Determine where the 'y' coordinates are in relation ot the border (window)

Calculate the new last point (x,y)

FILES:

Instrumentation file

REFERENCED BY:

REDRAW

SUBPROGRAMS REFERENCED:

INSINT, INSPGM, INSRET, RXINT, RYINT

HISTORY:

designed by Mark Maginn September 15, 1980

programmed by Mark Maginn September 16, 1980

PROGRAM NAME: FCGENT

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

FCGENT reads a logical entry from the file code table

PROGRAM USAGE:

CALL FCGENT(FID,LEN,FLPTR,NAME)

INPUT ARGUMENTS:

FID - INTEGER; file descriptor of the file code table

OUTPUT ARGUMENTS:

LEN - INTEGER; file code table logical entry number

FLPTR - INTEGER; file code table free list pointer

NAME - LOGICAL*1 ARRAY; filename portion of a file code table entry

FILES:

FCT - File Code Table

REFERENCED BY:

GEN

SUBPROGRAMS REFERENCED:

FGET

SEE ALSO:

FCPENT

HISTORY:

designed by Steven Haehn January 30,1979

programmed by Donna Morris March 2,1979

OLPARS Program Specifications
FCGHDR

PROGRAM NAME: FCGHDR

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

FCGHDR reads the file code table header

PROGRAM USAGE:

CALL FCGHDR(FID,NXTAVL,NUSED,LSTFEN,LSTENT)

INPUT ARGUMENTS:

FID - INTEGER; file descriptor of the file code table

OUTPUT ARGUMENTS:

NXTAVL - INTEGER; next available entry in the file code table

NUSED - INTEGER; number of filled entries in the file code table

LSTFEN - INTEGER; last free entry in the file code table

LSTENT - INTEGER; last entry in the file code table

FILES:

FCT - File Code Table

REFERENCED BY:

GEN

SUBPROGRAMS REFERENCED:

FGET

SEE ALSO:

FCPHDR

HISTORY:

designed by Steven Haehn January 30,1979

programmed by Donna Morris March 2,1979

PROGRAM NAME: FCPENT

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

FCPENT writes a logical entry to the file code table

PROGRAM USAGE:

CALL FCPENT(FID,LEN,FLPTR,NAME)

INPUT ARGUMENTS:

FID - INTEGER; file descriptor of the file code table

LEN - INTEGER; file code table logical entry number

FLPTR - INTEGER; file code table free list pointer

NAME - LOGICAL*1 ARRAY; filename portion of a file code table entry

FILES:

FCT - File Code Table

REFERENCED BY:

GEN

SUBPROGRAMS REFERENCED:

FPUT

SEE ALSO:

FCGENT

HISTORY:

designed by Steven Haehn January 30,1979

programmed by Donna Morris March 2,1979

OLPARS Program Specifications
FCPHDR

PROGRAM NAME: FCPHDR

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

FCPHDR writes out the file code table header

PROGRAM USAGE:

CALL FCPHDR(FID,NXTAVL,NUSED,LSTFEN,LSTENT)

INPUT ARGUMENTS:

FID - INTEGER; File descriptor of the file code table

NXTAVL - INTEGER; Next available entry in the file code table

NUSED - INTEGER; The number of filled entries in the file code table

LSTFEN - INTEGER; Last free entry in the file code table

LSTENT - INTEGER; Last entry in the file code table

FILES:

FCT - File Code Table

REFERENCED BY:

GEN

SUBPROGRAMS REFERENCED:

FPUT

SEE ALSO:

FCGHDR

HISTORY:

designed by Steven Haehn January 30,1979

programmed by Donna Morris March 2,1979

PROGRAM NAME: FCREAT

CATEGORY: OLPARS utility (system dependent)

PROGRAM DESCRIPTION:

FCREAT will create a block I/O file using file control service routines under RSX-11M V3.1.

PROGRAM USAGE:

```
INTEGER FCREAT
N = FCREAT (FILNAM,SAVFLG)
```

INPUT ARGUMENTS:

FILNAM - LOGICAL*1 array; file name represented by ASCII characters

SAVFLG - INTEGER; 0 = file to be saved after closing
1 = file to be deleted upon closing

ALGORITHM / NOTES:

The logical unit number is the function value returned.

If an error is found, the value returned is -1.

REFERENCED BY:

Level I subroutines and utility programs

HISTORY:

designed by Steve Haehn January 19, 1979

programmed by Dave Tipton March 12, 1979

PROGRAM NAME: FCTOPN

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

FCTOPN will open the file code table and fill in the file access and control table with the proper parameters. The file descriptor (FID) of the file code table is returned.

PROGRAM USAGE:

CALL FCTOPN(FID,ACCESS)

INPUT ARGUMENTS:

ACCESS - INTEGER; read, write, or update access to the file code table

OUTPUT ARGUMENTS:

FID - INTEGER; file descriptor of the file code table

ALGORITHM / NOTES:

This routine has the following named common block:

FACT - The File Access and Control Table

FILES:

FCT - File Code Table

REFERENCED BY:

GEN

SUBPROGRAMS REFERENCED:

FOPEN, GETFID

HISTORY:

designed by Steve Haehn January 30,1979

programmed by Donna Morris March 2,1979

PROGRAM NAME: FGET

CATEGORY: Level I File Access Subroutine (system dependent)

PROGRAM DESCRIPTION:

FGET returns to the calling program the number of FORTRAN REAL elements requested by the program from an OLPARS user file. The first parameter of FGET, FID, is used to determine which FACT entry describes the file to be accessed. From the parameters specifying the logical entry of the file and the first element, plus information in the FACT entry (header number of elements and logical entry number of elements), it is possible for FGET to compute the "absolute" element address in the file of the first element to be retrieved. Dividing this address by the number of elements per physical record (constant for all files) will give the physical record number containing the first element. Comparing the required physical record number to the physical record number currently contained in the memory buffer determines if the element requested is already in memory; if it is, it is returned to the calling program; if it is not, a new physical record must be read into the buffer. However, before reading the new physical record, a check of PF (put flag) determines if any elements of the current physical record were changed by a write (FPUT) operation. If so, the memory buffer is written to the file before the new physical record is read into the buffer.

PROGRAM USAGE:

```
INTEGER FGET
N = FGET(FID,LEN,FEN,NOE,RBUF)
```

INPUT ARGUMENTS:

FID - INTEGER; FACT entry for the file

LEN - INTEGER; the logical entry number within file (0 indicates header)

FEN - INTEGER; first element, within the logical entry, to be returned

NOE - INTEGER; the number of elements to be returned

OLPARS Program Specifications
FGET

OUTPUT ARGUMENTS:

RBUF - REAL array (NOE); an array large enough to store NOE elements

FGET - INTEGER; the number of real elements returned to the calling program. FGET returns -1 if an end-of-file is encountered

ALGORITHM / NOTES:

This routine has the following named COMMON blocks:

FACT - the File Access and Control Table which is described in the EASSON Final Report, Section 3.3.

REFERENCED BY:

All level II file reading routines.

SEE ALSO:

FPUT

HISTORY:

designed by Steve Haehn July 4, 1978

programmed by Donna Morris February 10, 1979

PROGRAM NAME: FILEIN

CATEGORY: Utility Command (system dependent)

PROGRAM DESCRIPTION:

FILEIN takes a "system" file in the user's directory of a specified format (see below) and uses it to create an OLPARS tree within an OLPARS user's directory.

PROGRAM USAGE:

User types in 'FILEIN'.

USER INTERACTION:

The user is asked:

1) for a file name - the name of the file to be converted to an OLPARS data tree.

2) for a tree name - the name of the OLPARS data tree to be created.

3) whether or not there is a numeric vector identifier on each vector.

4) for the number of keywords found on each vector.

5) for the dimensionality of the data set.

ALGORITHM / NOTES:

Erase screen.

Ask user for a tree name and a file name.

WHILE (the tree name is not unique within the TL file)

Tell the user about it and ask for a new name.

END WHILE

Ask user if a vector id. is present on each vector.

Ask user for the number of keywords present on each vector.

Ask user for the dimensionality of the vectors.

```
REPEAT

    Obtain a class name from the raw vector.

    Ignore any optional vector id. or optional keywords.

    Read a vector and write it to the class name file.

    IF (the class name is not the same as the previous
        class name)

        Close the previous class name file.

        Open new class name file

        IF ('open' fails)

            Create a new class name file.

            Set the number of vectors counter to zero.

        ENDIF

    ENDIF

    Increment the number of vectors counter.

    Write vector to class name file

UNTIL(end of file on user's input)


Close the last opened class name file.

Create a tree file pair.

Create a senior node (without means and covs.)
in the TI file.

Create lowest nodes (without means and covs.)

WHILE (not all class name files have been used)

    Obtain a class name.

    Open a class name file

    Put vectors in the TV file.

    Compute means and covariances for a class
    and the senior node.

    Place means and covariances in TI file entry
```

Close and delete class name file.

ENDWHILE

Place the means and covs. for the senior node in TI file.

Put up menu.

END

NOTE

Since vectors may be read in any order, a separate file is used to gather the vectors of a class into a contiguous group. (Each vector is appended to its appropriate CLASSNAME file.) Once the OLPARS tree is created, the CLASSNAME files should be destroyed. Also, FILEIN can currently create what can be considered an 'invalid' OLPARS data tree. The tree consists of a senior node and one lowest node (usually there must be at least two lowest nodes under any intermediate node). Throughout OLPARS there is no other way to obtain a tree with only one lowest node under an intermediate node. This type of tree is being allowed so that single vectors or classes may be added to existing trees via the APPEND command.

OLPARS data file input format

```
------> 1-4 character class name,
|       optional vector identifier,
|       optional keyword 1,
|                .            (Keywords have an arbitrary
|                .             length and may be imbedded
|                .             with blanks)
|       optional keyword N,
|              X1,
|                .
|                .       (vector elements)
|                .
one vector ->     Xndim;
                  .
                  .
                  .
                  /*      (end of file indicator)
                          (optional under RSX-11)
```

OLPARS Program Specifications
FILEIN

FILES:

TI - tree information	'N' classname files
TV - tree vector	CM - communication
TL - tree list	user input file
HS - history	

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLOSBL, CLOSE, CLOSFX, CLOSTR, CMGOTH, CREATR, FMNCOV, GETVEC, INSINI, INSINT, INSRET, MENU, OEXIT, OPENBL, OPENFX, TIPCAP, TIPET, TIPHDR, TIPNAM, TRMPUT, TVGVEC, TVPHDR, TVPVEC, USRIO1

DIAGNOSTICS:

Only the first 5 vectors with invalid dimensionalities will be reported to the user. Any errors occurring after the first five will not be listed, but the total number of "bad" vectors will be given.

SEE ALSO:

FILEOUT

HISTORY:

designed by Steven Haehn July 17, 1978

programmed by Steven Haehn June-July 1979

PROGRAM NAME: FILEOUT

CATEGORY: Utility Command (system dependent)

PROGRAM DESCRIPTION:

FILEOUT takes an OLPARS data tree structure (2 files) and converts it to a "system" file of a specified format (see below). The "system" file will be left in the OLPARS user's directory.

PROGRAM USAGE:

User types 'FILEOUT'

USER INTERACTION:

The user specifies:

1. the tree to be converted to a "system" file
2. the "system" file name
3. the field width of a vector measurement
4. the number of decimal positions in the measurement output field (precision)

ALGORITHM / NOTES:

OLPARS data file output format

```
------> 1-4 character class name,
|               x1
|               .       (vector elements)
|               .
|               .
one vector ->   xndim;
               .
               .
               .
               /*   (end of file indicator)
```

FILES:

TI - tree information	user vector text file
TV - tree vector	HS - history
TL - tree list	Instrumentation
OLPARS option	

OLPARS Program Specifications
FILEOUT

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLOSTR, CLOS$, CMGCDS, CMGOTH, EQUALA, ERASE, ERRSET,
GNXTND, GTLUN, INSINI, INSINT, INSRET, MENU, OEXIT,
OPENFX, OPENTR, OPEN$, PACKB, PROMPT, TIGET, TIGHDR,
TRMGET, TRMPUT, TVGVEC

SEE ALSO:

FILEIN

HISTORY:

designed by Steven Haehn July 19, 1978

programmed by Steven Haehn September 30, 1981

PROGRAM NAME: FILGET

CATEGORY: FILE I/O (system dependent)

PROGRAM DESCRIPTION:

FILGET reads a "system" file and interprets the data within it via a format string. Its functions are identical with TRMGET, except that the input comes from a file instead of a terminal.

PROGRAM USAGE:

```
INTEGER FILGET
K = FILGET(LUN,'format string',arg1,arg2,...)
```

INPUT ARGUMENTS:

LUN - INTEGER; logical unit number of input file

format string - CHARACTER; string containing conversion specifications for variables to be read

arg1, arg2, ... - addresses of arguments to be read

OUTPUT ARGUMENTS:

FILGET - INTEGER; the number of successfully matched and assigned input items

-1 ; end of file reached

-2 ; universal help symbol was read

ALGORITHM / NOTES:

See ALGORITHM/NOTES section of TRMGET

REFERENCED BY:

FILEIN

SUBPROGRAMS REFERENCED:

$FCHNL

DIAGNOSTICS:

FILGET will return a -1 value on end-of-file.

SEE ALSO:

FILPUT, TRMGET, TRMPUT

HISTORY:

designed by	Steven Haehn	July 24, 1978
programmed by	Curry Bartlett	Feb. 22, 1979
modified by	Dave Tipton	May 14, 1979

KNOWN BUGS:

For each logical unit that FILGET accesses, there must be a separate input buffer (e.g., buff1, buff2...) and set of buffer character pointers (e.g., lcou1-bcou1... see code). Thus, currently, FILGET can only read 1 input file at a time, without problems.

(SOLUTION: Create a separate .PSECT region for these buffers, which is expandable at task build time. Then use an indexing function that will return the address of the appropriate buffer when needed.)

PROGRAM NAME: FILPUT

CATEGORY: FILE I/O (system dependent)

PROGRAM DESCRIPTION:

FILPUT formats and writes data to a "system" file. Its functions are identical to TRMPUT, except that the output goes to a file instead of a terminal.

PROGRAM USAGE:

CALL FILPUT(LUN,'format string',arg1,arg2,...)

INPUT ARGUMENTS:

LUN - INTEGER; logical unit number of output file

format string - HOLLERITH character string

arg1,arg2,... - address of arguments to be written out

ALGORITHM / NOTES:

See ALGORITHM/NOTES section of TRMPUT

REFERENCED BY:

FILEOUT

SUBPROGRAMS REFERENCED:

$FCHNL

SEE ALSO:

FILGET, TRMGET, TRMGLB, TRMPUT

HISTORY:

designed by Steven Haehn July 24, 1978

programmed by Curry Bartlett Jan. 15, 1979

OLPARS Program Specifications
FISH

PROGRAM NAME: FISH

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

FISH projects an unknown vector on the Fisher direction and determines which class of a class pair the vector should receive a vote for.

PROGRAM USAGE:

```
    INTEGER FUNCTION FISH

I = FISH(VEC,NUM,THRSHD,NDIM,FISHDR)
```

INPUT ARGUMENTS:

VEC - REAL array (NDIM); the vector

NUM - INTEGER; the number of thresholds to use

THRSHD - REAL array (5); the five Fisher thresholds

NDIM - INTEGER; the dimension

FISHDR - REAL array (NDIM); the coefficients of the Fisher direction

OUTPUT ARGUMENTS:

FISH - INTEGER; 1 = class 1
2 = class 2
3 = neither class

REFERENCED BY:

PAIRLG

SEE ALSO:

FISHER

HISTORY:

designed by John W. Tenney July 2, 1981

programmed by John W. Tenney Oct. 9, 1981

PROGRAM NAME: FISHER

CATEGORY: Logic Design Command

PROGRAM DESCRIPTION:

FISHER constructs and evaluates the Fisher pairwise discriminant logic. It calculates the pairwise discriminant vector as well as the orthogonal discriminant vector for each pair of classes. The latter is saved for possible future use. Five thresholds for each pair of classes are also calculated at this time. The Fisher discriminant may be calculated using either the sum of the scatter matrices or the sum of the covariance matrices; also, any measurements may be eliminated from the calculation. After the logic has been evaluated, the user may choose to change the number of thresholds or the minimum vote count and reevaluate the logic. A description of the Fisher pairwise logic follows.

Fisher pairwise discriminant logic (fisher) is constructed by computing optimal linear discriminants and thresholds to distinguish between every pair of classes (subclasses) within a designated group. The linear discriminant is the Fisher linear discriminant given by:

$$d(ij) = \text{alpha} * \text{INVERSE}(W(ij)) * \text{delta}(ij)$$

where

$$\text{delta}(ij) = \text{Mean}(i) - \text{Mean}(j)$$

$$W(ij) = (\text{Nvec}(i)-1) * \text{Cov}(i) + (\text{Nvec}(j)-1) * \text{Cov}(j)$$
$$= \text{sum of the within-class scatter matrices}$$

or

$$W(ij) = \text{Cov}(i) + \text{Cov}(j)$$
$$= \text{sum of the within-class covariance matrices}$$

$$\text{Mean}(i) = \text{mean vector of class I}$$

$$\text{Cov}(i) = \text{covariance matrix for class I}$$

$$\text{Nvec}(i) = \text{number of vectors in class I}$$

alpha = a normalizing constant such that the length of d(ij) is 1. The numeric sign of 'alpha' is chosen such that the projected mean of class(i) is greater than the projected mean of class(j) for all i<j

(e.g. for 3 classes C1, C2, and C3 there are 3 possible class pairings; C1 vs C2, C1 vs C3, and C2 vs C3 (C2 vs C1 is not allowed). Note, first class subscript is always less than second class subscript. The numeric sign of 'alpha' is such that the projected mean of C1, (d(1,2),Mean(C1)), is greater than the projected mean of class C2, (d(1,2),Mean(C2)), that is, the projected mean of the first class of a class pair is greater than the projected mean of the second class of the class pair.

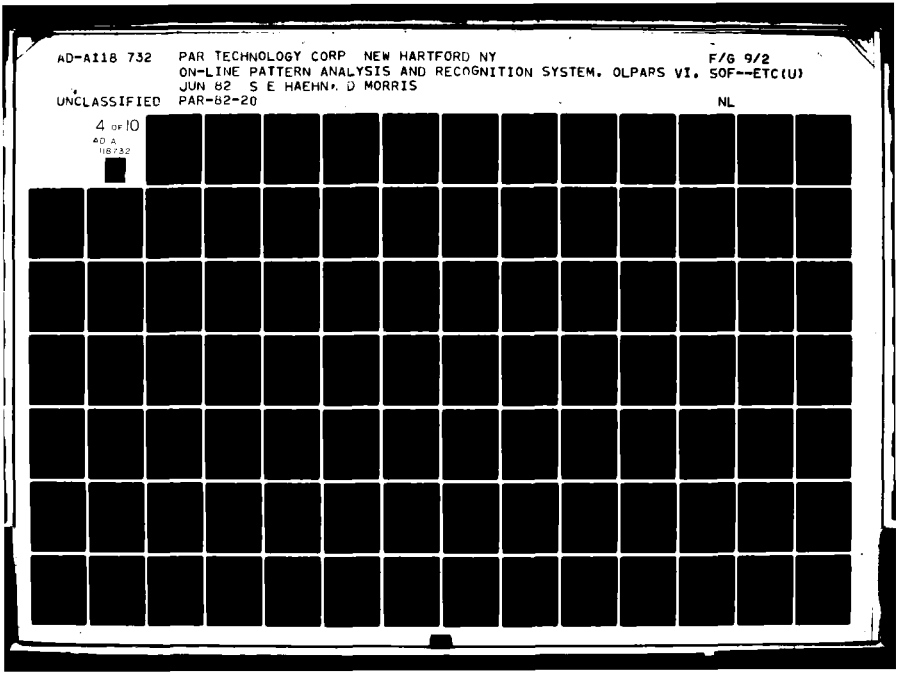
To continue:

(d(1,3),Mean(C1)) > (d(1,3),Mean(C3))
(d(2,3),Mean(C2)) > (d(2,3),Mean(C3))
.
.
(d(ij),Mean(i)) > (d(ij),Mean(j)))

Once the within-group pairwise discrimination is complete, the pairwise decisions are combined to produce a final decision. The group of classes might be the original K classes of the "current data set," or the group might be composed of a subset of K. In the case where the user does not subdivide the K classes,

(s)he would compute K[(K - 1)/2] pairwise discriminants. The resultant logic for this example is shown in Figure 1.

The output from the 'ij' box for a vector X is either a zero or a one, depending on the value of (d(ij), X) and the threshold option chosen. A one is a vote for class 'i' and a zero is a vote for class 'j'. If (d(ij), X) lies outside the specified region, a vote for neither class is obtained.


AD-A118 732
PAR TECHNOLOGY CORP NEW HARTFORD NY
F/G 9/2
ON-LINE PATTERN ANALYSIS AND RECOGNITION SYSTEM. OLPARS VI. SOF--ETC(U)
JUN 82 S E HAEHN, D MORRIS
UNCLASSIFIED
PAR-82-20
NL
4 OF 10
AD A
118732

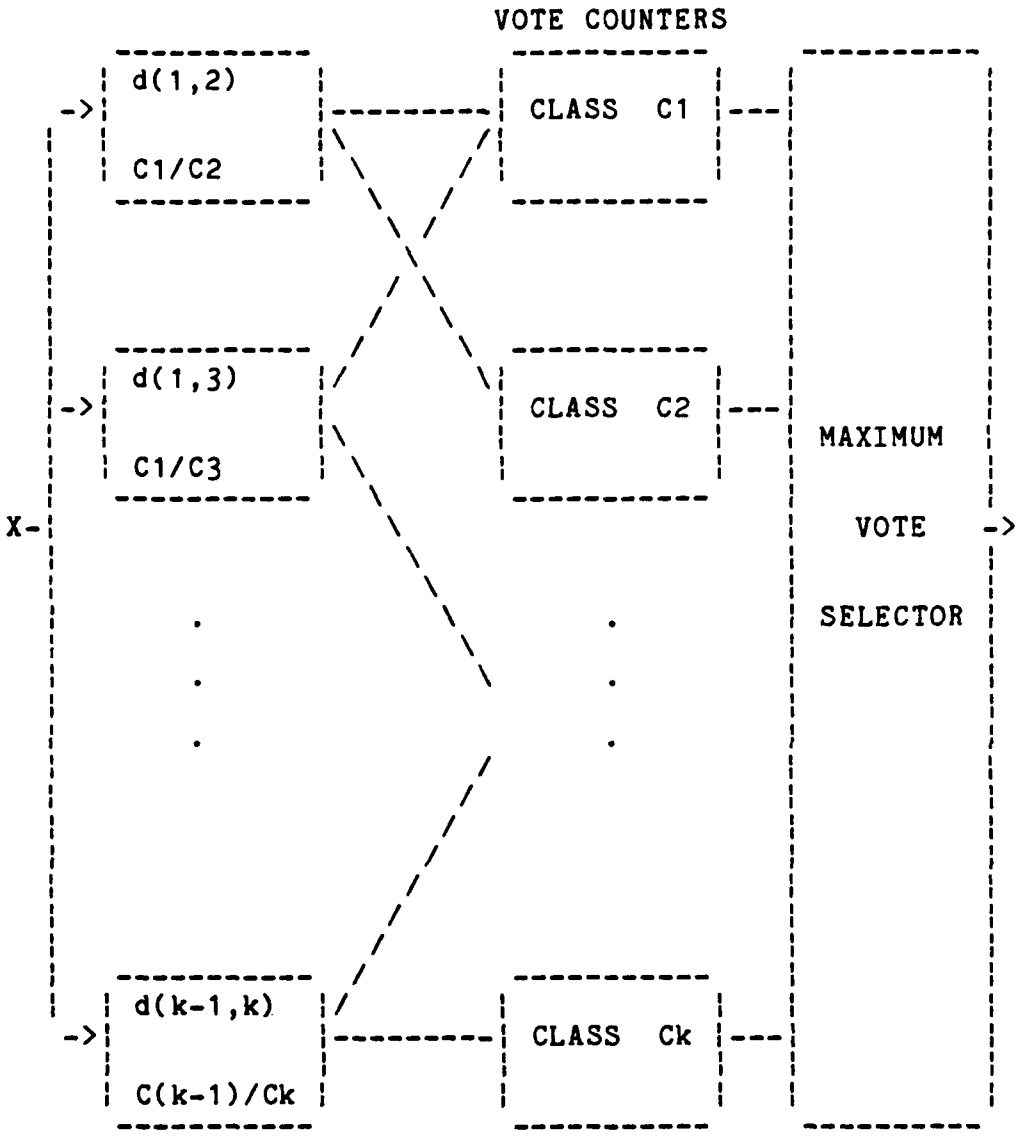


Figure 1 - Pairwise Logic

The votes for each class are collected and a final decision is made according to the class with the most votes. If there is a tie, the decision is made for the class with the largest apriori probability. If the apriori probabilities of the tied classes are equal, the vector is rejected. The vector is also rejected if the maximum class vote is less than a minimum vote count.

Under the Fisher discriminant logic option, the user can select 1, 2, 3, or 4 threshold options which result in the different boundaries shown in Figure 2.

In Figure 2, 'm(0)' is the estimated mean of class '0' projected onto the discriminant direction 'D', i.e.,

$$m(0) = \text{transpose}(D) * \text{Mean}(0)$$

$$= (d(ij), \text{Mean}(0))$$

If 'E' is considered the first class and '0' considered the second class of a class pair, OLPARS will automatically set the thresholds:

$$t(1) = \text{Mean}(0) - \text{delta}(E,0)/2$$

$$t(2) = \text{Mean}(0) + \text{delta}(E,0)/3$$

$$t(3) = \text{Mean}(0) + \text{delta}(E,0)/2$$

$$t(4) = \text{Mean}(E) - \text{delta}(E,0)/3$$

$$t(5) = \text{Mean}(E) + \text{delta}(E,0)/2$$

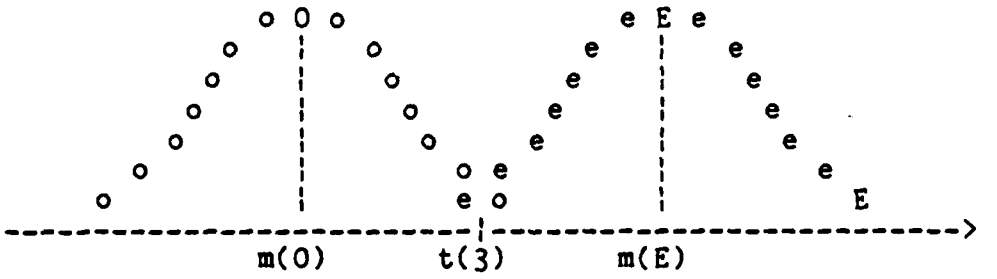
==========

Notational Description for Figure 2:

m(0)	- mean of class '0'
t(1)	- first threshold (5 thresholds total)
d(ij)	- discriminant vector of the ith and jth class
D	- d(ij)
X	- arbitrary class vector
(D,X)	- vector X evaluated against discriminant
>	- relational operator 'greater than'
=>	- relational operator 'greater than or equal to'
<	- relational operator 'less than'
<=	- relational operator 'less than or equal to'

==========

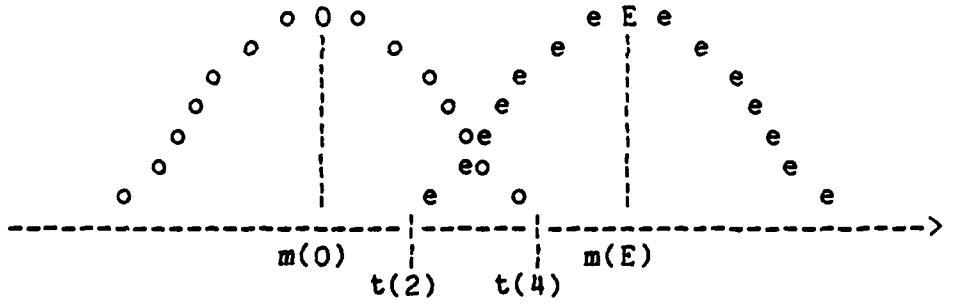
a) One Threshold for class pair (E vs O)



Vote for class 'E' if (D,X) > t(3)

Vote for class 'O' if (D,X) <= t(3)

b) Two Thresholds for class pair (E vs O)



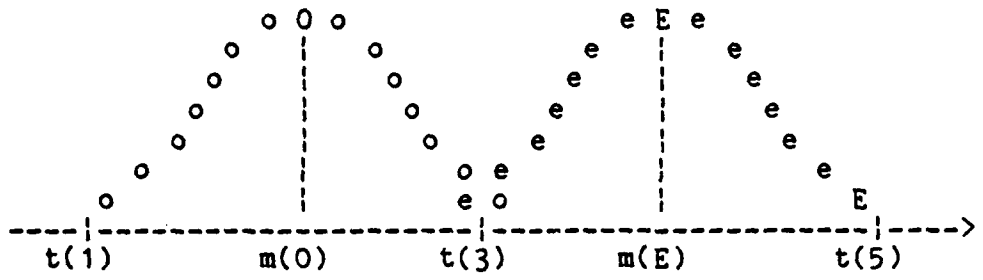
Vote for class 'E' if (D,X) > t(4)

Vote for class 'O' if (D,X) < t(2)

NO VOTE if t(2) <= (D,X) <= t(4)

Figure 2 - Threshold Options for Fisher Pairwise Logic (Here class pair E vs O)

c) Three Thresholds for class pair (E vs O)

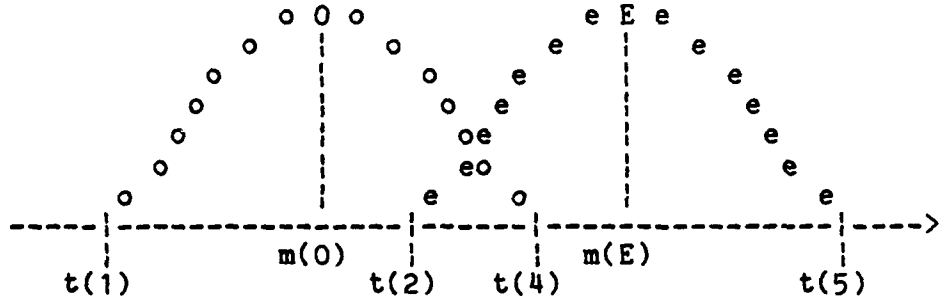


Vote for class 'E' if t(3) < (D,X) < t(5)

Vote for class 'O' if t(1) < (D,X) <= t(3)

NO VOTE if (D,X) <= t(1) or (D,X) => t(5)

d) Four Thresholds for class pair (E vs O)



Vote for class 'E' if t(4) < (D,X) < t(5)

Vote for class 'O' if t(1) < (D,X) < t(2)

NO VOTE if (D,X) <= t(1) or t(2) <= (D,X) <= t(4)
or (D,X) => t(5)

Figure 2 - Threshold Options for Fisher Pairwise Logic (continued)

PROGRAM USAGE:

User types in 'FISHER'.

ALGORITHM / NOTES:

Erase screen and home the cursor (ERASE).

Get set up for logic operation (SETUP).

Set up logic block for Fisher pairwise. This includes determining which measurements and vectors to use, computing, if necessary, the covariance matrices of the classes, computing the Fisher and orthogonal directions and storing these quantities in the pairwise logic block (FISHSU,FISHSL).

Prompt user for number of thresholds and insert this number in the proper entry for each class pair (FISHTH).

Prompt user for minimum vote count and insert in pairwise logic block (entry 1) (FISHVC).

Reset the logic elements in the TV file (KILLND).

Put up option list at user's terminal (MENU).

END

SUBPROGRAMS REFERENCED:

CHKEXS, CMGCDS, CMGLOG, CMPLOG, ERASE, FISHSL, FISHSU, FISHTH, FISHVC, INSINI, INSRET, KILLND, LIPCOP, MENU, OEXIT, OPENFX, SETUP, TRMPUT

HISTORY:

designed by John W. Tenney Sept. 2, 1981

programmed by John W. Tenney Oct. 9, 1981

OLPARS Program Specifications
FISHMOD

PROGRAM NAME: FISHMOD

CATEGORY: Logic Design Command

PROGRAM DESCRIPTION:

FISHMOD is designed to modify an existing FISHER logic node. The user is given a list of logic nodes with FISHER logic. After the node is selected, the user is prompted for a new minimum vote count and number of thresholds.

To evaluate the modified logic, use PWEVAL.

PROGRAM USAGE:

User types in 'FISHMOD'.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Display a list of PAIRWISE logic nodes (GETLST,TRMPUT).

Ask user to enter a PAIRWISE logic node number (PROMPT).

Save old minimum vote count (GTRGNI).

Prompt user for a new minimum vote count and set corresponding flag in LV file (FISHVC).

Prompt user for number of thresholds and store in logic block (FISHTH).

IF (user wishes to quit) THEN
save old minimum vote count (PTRGNI).

Fix temporary logic elements in the TV file (KILLND).

Put up option list at user's terminal (MENU)

END

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CHKEXS, CMGCDS, CMGLOG, CMGOTH, ERASE, FISHTH, FISHVC
GETLST, GTRGNI, INSINI, INSRET, KILLND, LIGCOP, LIGSTR
MENU, OEXIT, OPENFX, OPENTR, PROMPT, PTRGNI, TRMGET
TRMPUT

SEE ALSO:

FISHER, THRESHMOD, OPTIMLMOD, PWEVAL

HISTORY:

designed by John W. Tenney Sept. 2, 1981

programmed by John W. Tenney Oct. 9, 1981

OLPARS Program Specifications
FISHSL

PROGRAM NAME: FISHSL

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

FISHSL sets up the LI portion of the Logic Block, and also stores the node numbers in the LV file.

PROGRAM USAGE:

SUBROUTINE FISHSL (FIDLI,FIDLV,NODNUM,TRUCLS,LVNEL,ENT1)

INPUT ARGUMENTS:

FIDLI - INTEGER; file descriptor for logic information

FIDLV - INTEGER; file descriptor for logic value file

NODNUM - INTEGER; current logic node number

TRUCLS - INTEGER; number of classes at the logic node

LVNEL - INTEGER; number of elements in LV file entry

ENT1 - INTEGER; pointer to first LV entry

ALGORITHM / NOTES:

Get LI entries for the logic node corresponding to each class plus the reject node (GNLIAE).

Store the node numbers in the LV file (PTRGNI)

DO WHILE (there are more classes + 1 for the reject node) (reject node is done first)

In this entry set the following (LIPENT):

Logic code type = 0 (undefined).

Logic subtype = 0

Option Number = 0

Node level = node level of pairwise node plus 1

Number of nodes below = 0

Entry number of parent = NODNUM

Entry of first child = 0

Entry in LV of start of decision logic = 0

Entry of reject strategy = 0

Data class name

Reassociated name = data class name

Modified logic flag = 0

Number of classes = 1

Bit Map (EVALBM)

ENDDO

RETURN

FILES:

LI - logic information
LV - logic value

REFERENCED BY:

FISHER

SUBPROGRAMS REFERENCED:

EVALBM, GNLIAE, GTRGNI, INSINT, INSPGM, INSRET, LIGCLP, LIGCNM, LIGENT, LIGSTR, LIPENT, PTRGNI

SEE ALSO:

FISHSU

HISTORY:

designed by John W. Tenney Sept. 10, 1981

programmed by John W. Tenney Oct. 9, 1981

PROGRAM NAME: FISHSU

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

FISHSU sets up the LV portion of the Logic Block for FISHER logic.

PROGRAM USAGE:

```
SUBROUTINE FISHSU (FIDCM,FIDLI,FIDLV,FIDTI,FIDTV,NODNUM,
                   ALLVEC,ET,NDIM,TRUCLS,CLIST,PMODE,
                   QFLAG,LVNEL,ENT1)
```

INPUT ARGUMENTS:

FIDCM - INTEGER; file descriptor for CM file

FIDLI - INTEGER; file descriptor for LI file

FIDLV - INTEGER; file descriptor for LV file

FIDTI - INTEGER; file descriptor for TI file

FIDTV - INTEGER; file descriptor for TV file

NODNUM - INTEGER; current logic node number

ALLVEC - LOGICAL; true if user wants to use all vectors that lie at the logic node.

ET - INTEGER array (TABSIZ); the entry table

NDIM - INTEGER; the dimensionality

TRUCLS - INTEGER; number of classes at the logic node

CLIST - INTEGER array (TRUCLS); entry table slot numbers

PMODE - INTEGER; the prompt mode flag

OUTPUT ARGUMENTS:

QFLAG - LOGICAL; False if the user wants to quit

LVNEL - INTEGER; number of elements in LV file entry

ENT1 - INTEGER; pointer to first LV entry

ALGORITHM / NOTES:

Save the number of classes for the logic block.

Allocate an LV entry for the first entry in the block. (GNLVAE)

Allocate space for node numbers and pointers (PTRGNI).

Set first four elements of first entry

DO WHILE (there are more class pairs)

Set up pointers to the classes

Get the fisher and orthogonal directions (DCRIM)

Retrieve the means and compute the thresholds (LOGMN, TIGMN,PROJCT)

Store the LV entries for this class pair (PTRGNR)

ENDDO

Terminate the LV region (LVPLNK).

Store the first LV entry (PTRGNI).

RETURN

FILES:

CM - communications
LI - logic information
LV - logic value
PV - projection vector
TI - tree information
TV - tree vector

REFERENCED BY:

FISHER

SUBPROGRAMS REFERENCED:

CLOSFX, DCRIM, ELIMEA, GNLVAE, INSPGM, INSRET, LIGCOP, LOGMN, LVPLNK, OPENFX, PROJCT, PTRGNI, PTRGNR, SETOPT, TIGMN, TRMPUT

SEE ALSO:

FISHSL

HISTORY:

designed by John W. Tenney Sept. 10, 1981

programmed by John W. Tenney Oct. 9, 1981

PROGRAM NAME: FISHTH

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

FISHTH prompts the user for the number of thresholds and inserts this number into the pairwise logic block for every class pair.

PROGRAM USAGE:

LOGICAL FUNCTION FISHTH

QUIT = FISHTH(FIDLV,ENT1,LVNEL,TRUCLS,PMODE)

INPUT ARGUMENTS:

FIDLV - INTEGER; the file descriptor of the LV file

ENT1 - INTEGER; the entry number in LV of the first class pair

LVNEL - INTEGER; number of elements in LV file entry

TRUCLS - INTEGER; number of classes at logic node

PMODE - INTEGER; the prompt mode flag

USER INTERACTION:

Asks user for the number of thresholds.

FILES:

LV - logic value

REFERENCED BY:

FISHER, FISHMOD

SUBPROGRAMS REFERENCED:

DLREGN, GTRGNI, GTRGNR, INSPGM, INSRET, PROMPT, PTRGNR, TRMGET

DIAGNOSTICS:

If the user wishes to quit, FISHTH is set equal to TRUE.

HISTORY:

designed by David A. Birnbaum September 27, 1978

programmed by John W. Tenney Oct. 9, 1981

OLPARS Program Specifications
FISHVC

PROGRAM NAME: FISHVC

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

FISHVC prompts the user for a minimum vote count and inserts it into the first entry in a pairwise logic block.

PROGRAM USAGE:

LOGICAL FUNCTION FISHVC

QUIT = FISHVC(FIDLV,ENT1,LVNEL,PMODE)

INPUT ARGUMENTS:

FIDLV - INTEGER; the file descriptor of the LV file

ENT1 - INTEGER; the entry number in LV of entry 1 of the pairwise logic block

LVNEL - INTEGER; the number of elements in an LV entry

PMODE - INTEGER; the prompt mode flag

FILES:

LV - logic value

REFERENCED BY:

FISHER, PAIRMOD

SUBPROGRAMS REFERENCED:

GTRGNI, INSPGM, INSRET, PROMPT, PTRGNI, TRMGET, TRMPUT

DIAGNOSTICS:

FISHVC returns a value of TRUE if the user wishes to quit

HISTORY:

designed by David A. Birnbaum September 27, 1978

programmed by John W. Tenney Oct. 9, 1981

PROGRAM NAME: FIXKLV

CATEGORY: Data Tree Access Routine

PROGRAM DESCRIPTION:

FIXKLV is passed the file descriptor of the TI file being updated, the entry table(EnTble), the slot number of the new node(NewNod), the number of lowest nodes to add(NlowNd), the number of kids to add(Nkids) and the number of vectors to add (Nvecs).

It adds 'Nkids' to the number of children at the parent node of the new node, then traverses the tree via parent pointers adding 'NlowNd' to the number of lowest nodes beneath the present node, and 'Nvecs' to the number of vectors at the present node.

PROGRAM USAGE:

CALL FIXKLV(FDTI, ENTBLE, NEWNOD, NLOWND, NKIDS, NVECS)

INPUT ARGUMENTS:

FDTI - INTEGER; the file descriptor of the TI file

ENTBLE - INTEGER; the entry table of the updated tree

NEWNOD - INTEGER; the slot number of the new node

NLOWND - INTEGER; the number of lowest nodes to be added

NKIDS - INTEGER; the number of children to be added

NVECS - INTEGER; the number of vectors to be added

OLPARS Program Specifications
FIXKLV

ALGORITHM / NOTES:

Get the parent pointer of the new node

Update the number of children at the parent node

WHILE(parent ptr. ne. no more parents)

Update the number of lowest nodes at current node

Update the number of vectors at the current node

Obtain a new parent

ENDWHILE

END

FILES:

TI - tree information

REFERENCED BY:

APPEND, COMNOD

SUBPROGRAMS REFERENCED:

INSINT, INSPGM, INSRET, TIGCOP, TIPCOP

HISTORY:

designed by Mark Maginn August 22, 1980

programmed by Mark Maginn August 22, 1980

PROGRAM NAME: FLUSHB

CATEGORY: Utility subroutine (system dependent)

PROGRAM DESCRIPTION:

FLUSHB (flush block buffer) is used where it's important to cause a file's 'in memory' buffer to be written to the file.

PROGRAM USAGE:

CALL FLUSHB(FD)

INPUT ARGUMENTS:

FD - INTEGER; file descriptor of the block I/O file to be flushed.

ALGORITHM / NOTES:

This program should be compiled with the F4P compiler's /-CK switch (i.e., that is array checking is turned off). This allows access to the PRBUFF of the File Access Control Table outside of its declared size (because it can be expanded at task build time).

REFERENCED BY:

GETHST, PUTHST

SUBPROGRAMS REFERENCED:

WRITEB

DIAGNOSTICS:

Bad file descriptors and write failures are mentioned to user.

HISTORY:

designed by Steven Haehn July 20, 1979

programmed by Steven Haehn Aug 21, 1979

OLPARS Program Specifications
FLUSHF

PROGRAM NAME: FLUSHF, FLUSHT

CATEGORY: File, Terminal I/O (system dependent)

PROGRAM DESCRIPTION:

FLUSHF flushes (cleans out, clobbers, essentially eradicates) the file I/O input buffer accessed by the FILGET subroutine (useful when ignoring current lines are desired).

FLUSHT flushes (and all the above) the terminal I/O input buffer accessed by the TRMGET subroutine (usefulness, same).

PROGRAM USAGE:

```
CALL FLUSHF(LUN)
CALL FLUSHT
```

INPUT ARGUMENTS:

for FLUSHF -- LUN - INTEGER; the logical unit of the file line to be flushed

ALGORITHM / NOTES:

These calls are global entry points into the TRMGET module.

REFERENCED BY:

Any file or terminal accessing routine, in particular, MAKOPT.

HISTORY:

designed by Steven Haehn Aug. 28, 1979

programmed by Steven Haehn Sept. 4,1979

KNOWN BUGS:

SEE FILGET SPECIFICATIONS

PROGRAM NAME: FOPEN

CATEGORY: OLPARS utility (system dependent)

PROGRAM DESCRIPTION:

FOPEN will use RSX-11M V3.1 file control service routines to open a block I/O file for read, update, or append.

PROGRAM USAGE:

```
INTEGER FOPEN
N = FOPEN (FILNAM,ACCESS)
```

INPUT ARGUMENTS:

FILNAM - LOGICAL*1 array; file name represented by ASCII characters

ACCESS - INTEGER; 0 = file open for read only
1 = file open for write (update under RSX-11M)
2 = file open for append

ALGORITHM / NOTES:

The logical unit number is the function value returned.

If an error is found, the value returned is -1.

REFERENCED BY:

Level I subroutines and utility programs

HISTORY:

designed by Steve Haehn January 19, 1979

programmed by Dave Tipton March 12, 1979

OLPARS Program Specifications
FPUT

PROGRAM NAME: FPUT

CATEGORY: Level I File Access Subroutine (system dependent)

PROGRAM DESCRIPTION:

FPUT writes out to an OLPARS user file the number of FORTRAN REAL elements given to it by the calling program.

PROGRAM USAGE:

CALL FPUT(FID,LEN,FEN,NOE,RBUF)

INPUT ARGUMENTS:

FID - INTEGER; FACT entry for the file

LEN - INTEGER; the logical entry number within file (0 indicates header)

FEN - INTEGER; first element, within the logical entry, to be written out

NOE - INTEGER; the number of elements to be written out

RBUF - REAL array (NOE); an array large enough to store NOE elements

ALGORITHM / NOTES:

This routine has the following named COMMON blocks:

FACT - the File Access and Control Table which is described in the EASSON Final Report, Section 3.3.

REFERENCED BY:

All level II file writing routines.

SEE ALSO:

FGET

HISTORY:

designed by Steven Haehn July 4, 1978

programmed by Donna Morris February 10, 1979

OLPARS Program Specifications
FXLTMP

PROGRAM NAME: FXLTMP

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

FXLTMP is called to adjust (fix) the temporary logic elements in the vectors stored in the TV file. It is needed any time a modification to a logic at a node takes place.

FXLTMP searches the TV file for all vectors whose temporary logic elements are equal to logic nodes below the given node. It changes their elements to the given logic node number.

PROGRAM USAGE:

CALL FXLTMP(FDLI,FDLV,FDTI,FDTV,CRNTLG,NODLST,NODCNT)

INPUT ARGUMENTS:

FDLI - INTEGER; the file descriptor of the LI file
FDLV - INTEGER; the file descriptor of the LV file
FDTI - INTEGER; the file descriptor of the TI file
FDTV - INTEGER; the file descriptor of the TV file
CRNTLG - INTEGER; the current logic node number

OUTPUT ARGUMENTS:

NODLST - INTEGER array(MAXCLS*(MAXCLS+1)/2);
list of logic nodes below current logic node

NODCNT - INTEGER; number of logic nodes below current logic node

ALGORITHM / NOTES:

Get a list of logic node numbers below the given logic node (GNXTLN).

Get a list of classes at the given logic node (GCLIST)

For each class present at the logic node make changes to all those vectors of that class that lie at a lower logic node (TVGLOG,TVPLOG)

RETURN

FILES:

LI - logic information
LV - logic value
TI - tree information
TV - tree vector

REFERENCED BY:

KILLND

SUBPROGRAMS REFERENCED:

EQUALA, GCLIST, GNXTLN, INSPGM, INSRET, TIGCOP, TIGET, TIGHDR, TIGNAM, TVGLOG, TVPLOG

HISTORY:

designed by David A. Birnbaum September 25, 1978

programmed by Steven Haehn April 1, 1981

PROGRAM NAME: GARBND

CATEGORY: Data Tree File Access Routine

PROGRAM DESCRIPTION:

GARBND (garbage node) adds a node entry of the TI file to the TI file free list.

PROGRAM USAGE:

CALL GARBND(FID, SLTNO, ENTTBL)

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the TI file

SLTNO - INTEGER; the entry table slot number of the entry to be added to the TI file free list

ENTTBL - INTEGER ARRAY; the entry table of the TI file header

FILES:

TI - tree information

REFERENCED BY:

COMNOD

SUBPROGRAMS REFERENCED:

INSINT, INSPGM, INSRET, TIGHDR, TIPCOP
TIPET, TIPHDR

HISTORY:

designed by David A. Birnbaum August 23, 1978

programmed by David J. Tipton October 31, 1980

PROGRAM NAME: GCLIST

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

GCLIST gets the list of class names that are at a logic node.

PROGRAM USAGE:

```
CALL GCLIST(FDLI,NODNUM,CLIST,NCLAS)
```

INPUT ARGUMENTS:

FDLI - INTEGER; the file descriptor of the LI file
NODNUM - INTEGER; the logic node number

OUTPUT ARGUMENTS:

CLIST - INTEGER array (NODLEN,MAXCLS); the class list
NCLAS - INTEGER; the number of classes in this list

ALGORITHM / NOTES:

The algorithm used in this program relies on the evaluating bitmap routine generating its 'classes-present' list in numerical ascending order.

FILES:

LI - logic information

REFERENCED BY:

SETUP

SUBPROGRAMS REFERENCED:

EVALBM, INSINT, INSPGM, INSRET, LIGCLP, LIGCNM

HISTORY:

designed by David BirnBaum Sept. 14, 1978

programmed by Steven Haehn Dec. 6, 1980

OLPARS Program Specifications
GDSTRC

PROGRAM NAME: GDSTRC

CATEGORY: Data Tree File Access Routine

PROGRAM DESCRIPTION:

GDSTRC obtains the tree structure of an OLPARS data set (treename, nodename pair). The elements of the tree structure, returned, are those that are necessary for making a display of the tree structure at the user's terminal. GDSTRC returns the lowest level of nodes found in the tree as its function value.

PROGRAM USAGE:

```
INTEGER GDSTRC
   .
   .
LOWEST = GDSTRC(FDTI,ENTBLE,STRTND,CUTOFF,NAMES,LEVELS,
                NODPTR, PARENT, NUMVEC, NODCNT, LOWCNT)
```

INPUT ARGUMENTS:

FDTI - INTEGER; the file descriptor of the TI file being accessed

ENTBLE - INTEGER array(TABSIZ); the entry table of TI file

STRTND - INTEGER; slot number of the starting node in the traversal of the data set

CUTOFF - INTEGER; the lowest level of the structure to be returned

OUTPUT ARGUMENTS:

NAMES - INTEGER array (NODLEN,MAXCLS*2); names of all the nodes in the structure returned

LEVELS - INTEGER array (LEVMAX +1); pointers into NODPTR (see ALGORITHM/NOTES)

NODPTR - INTEGER array (MAXCLS*2); pointers into NAMES, NUMVEC, PARENT arrays (see ALGORITHM/NOTES)

PARENT - INTEGER array (MAXCLS*2); parent pointers of the nodes within the desired tree structure

NODCNT - INTEGER; total number of nodes found

LOWCNT - INTEGER; total number of lowest nodes found

ALGORITHM / NOTES:

If a cutoff level has been specified by setting 'CUTOFF' to a value other than zero, any nodes at the cutoff level, as well as any lowest nodes above that level, will be returned as lowest nodes.

The value of the index into the 'LEVELS' array represents the node level of the data tree being drawn.

Example: LEVELS(2), shows we are dealing with the second level of nodes in the data tree.

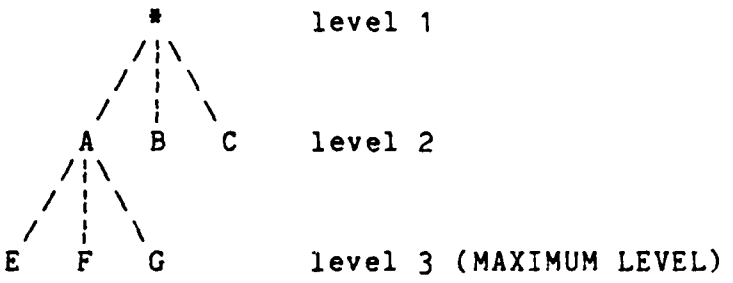
Note, the level being considered will only agree with the actual level value found in the node if the 'current node' is the senior node of the tree. Therefore, the index into the 'LEVELS' array may be considered the 'relative' level of the current data set.

The 'LEVELS' array contains pointers to the 'NODPTR' array. The values in 'LEVELS' point to the first node found at the given level.
The expression 'LEVELS(i+1) - LEVELS(i)' for values of i from 1 to MAXIMUM LEVEL + 1 will yield the number of nodes found at 'LEVELS(i)'.

The 'NODPTR' array contains pointers to the 'tree information' arrays ('NAMES', 'NUMVEC', 'PARENT') The contents of the 'tree information' arrays are in the order as obtained by the 'gnxtnd' subprogram. The 'NODPTR' values can be considered pointers to a record containing the following information about a node in a data tree:

1) name of a data node
2) parent pointer of a node
3) number of vectors that lie at a data node
   (0 if not a lowest node)

The following figure shows the relationship between the 'LEVELS', 'NODPTR', and 'tree information' arrays for the given tree.



LEVELS	1	2	5	8

1 2 5 8 (LEVELS VALUES)

NODPTR	1	2 - 6 - 7	3 - 4 - 5	

	1	2	3	4	5	6	7
NAMES	*	A	E	F	G	B	C
	*	A	E	F	G	B	C
	*	A	E	F	G	B	C
	*	A	E	F	G	B	C
NUMVEC	0	0	6	8	4	6	8
PARENT	0	1	2	2	2	1	1

FILES:

TI - tree information

REFERENCED BY:

DRAWTREE

SUBPROGRAMS REFERENCED:

GNXTND, INSINT, INSPGM, INSRET, TIGCAP, TIGNAM

DIAGNOSTICS:

If the tree structure of the data tree is corrupted, GDSTRC will return a -1 as its function value.

SEE ALSO:

GLSTRC

HISTORY:

designed by Steven Haehn August 4, 1978

programmed by Mark Maginn and Steve Haehn Dec. 16, 1980

PROGRAM NAME: GEN

CATEGORY: OLPARS Programmer Aids (system dependent)

PROGRAM DESCRIPTION:

The GEN utility will create all the necessary OLPARS 'fixed' files that are required by an OLPARS user and set up the user's File Code Table which specifies the total number of trees (data logic) an OLPARS user is allowed to have in his directory.

PROGRAM USAGE:

See Algorithms/Notes

ALGORITHM / NOTES:

Initiating syntax for GEN is:

GEN	/EX:n	expand old user by n-trees
	/NW:n	create new user with n-trees
	/LI	print out a user's tree capacity

NOTE

Gen must be used while in an OLPARS user's directory.

FILES:

Sets up a user's File Code Table and
Creates all OLPARS 'fixed' files

CM - communications
TL - tree list
LL - logic list
DI - display information
DV - display value
PV - projection vector
S1 - scratch 1
SV - saved vector
SM - saved transformation matrix
HS - history file

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

ASCDEC,CLOSE,CLOSFX,CREAFX,EQUALS,FCGENT,FCGHDR,
FCPENT,FCPHDR,FCREAT,FCTOPN,FPUT,GETMCR,GETSTR,
HSPHDR,HSPREC,INDEX,LENGTH,NBLANK,OCLOSE,OEXIT,TRMPUT

SEE ALSO:

Appendix C of the OLPARS Programmer and System
Maintenance Manual

HISTORY:

designed by Steven Haehn September 25, 1978

programmed by Donna Morris Feb. 28, 1979

OLPARS Program Specifications
GETFID

PROGRAM NAME: GETFID

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

GETFID searches the file access and control table for a free file descriptor

PROGRAM USAGE:

CALL GETFID(FID)

OUTPUT ARGUMENTS:

FID - INTEGER; file descriptor to the file access and control table

ALGORITHM / NOTES:

This routine has the following named common block:

FACT - The File Access and Control Table

REFERENCED BY:

GEN

SUBPROGRAMS REFERENCED:

TRMPUT, OEXIT

DIAGNOSTICS:

If there are no logical unit numbers available, GETFID prints out a message and exits.

HISTORY:

designed by Steve Haehn January 30,1979

programmed by Donna Morris March 2,1979

PROGRAM NAME: GETHST

CATEGORY: Utility subroutine (system dependent)

PROGRAM DESCRIPTION:

GETHST retrieves a history file record from the instrumentation package history file, increases the tally portion of the record (this signifies that the program referred to by the given 'NAME' argument, has been entered or is now executing), and writes this record back into the history file.

PROGRAM USAGE:

```
CALL GETHST(NAME,NAMLEN)
```

INPUT ARGUMENTS:

NAME - LOGICAL*1 ARRAY; a Hollerith string representing the name of the program entered.

NAMLEN - INTEGER; length of the 'NAME' character string

ALGORITHM / NOTES:

The folowing variables within the common area 'HISTRY' are changed.

HSLINK(HSP), HSNMLN(HSP), HSTALY(HSP), HSTNAM(1-9,HSP), HSTPTR(HSP), HSP

REFERENCED BY:

INSPGM

SUBPROGRAMS REFERENCED:

CONCAT,EQUALS,FLUSHB,HASH,HSGHDR,HSGREC,HSPHDR,HSPLNK, HSPREC,OEXIT,TRMPUT

SEE ALSO:

PUTHST

OLPARS Program Specifications
GETHST

HISTORY:

designed by Steven Haehn Feb. 23, 1979

programmed by Steven Haehn Apr. 6, 1979

PROGRAM NAME: GETLST

CATEGORY: Logic Tree File Access Routine

PROGRAM DESCRIPTION:

GETLST retrieves a list of logic nodes of a particular type according to the value of TYPE.

TYPE	PROPERTY OF LOGIC NODE
0	Incomplete
1	Pairwise
2	Group logic (one-space, two-space, Boolean)
3	NMV
4	Closed decision boundary
5	Nearest Neighbor
6	Lowest

An incomplete logic node is one that has no decision logic and has more than one class assigned to it. A lowest logic node has no children below it in the tree structure.

PROGRAM USAGE:

CALL GETLST(FDLI,TYPELN,NODLST,NODCNT)

INPUT ARGUMENTS:

FDLI - INTEGER; the file descriptor of the LI file

TYPELN - INTEGER; the type of logic node to be retrieved

OLPARS Program Specifications
GETLST

OUTPUT ARGUMENTS:

NODLST - INTEGER array (MAXCLS); the list of nodes (node numbers) that have been requested

NODCNT - INTEGER; the number of nodes retrieved

ALGORITHM / NOTES:

This routine traverses the logic tree using GNXTLN to pick out the set of nodes specified by TYPELN.

FILES:

LI - logic information

REFERENCED BY:

FORTLOGC, NMVMOD, SETUP

SUBPROGRAMS REFERENCED:

GNXTLN, INSINT, INSPGM, INSRET, LIGLOG, LIGSTR

HISTORY:

designed by David BirnBaum September 25, 1978

programmed by Steven Haehn Nov. 25, 1980

PROGRAM NAME: GETOPT

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

GETOPT returns to the calling program the list of options of the current OLPARS command. The OPTION array will contain the option names retrieved from the OLPARS option file. OPTCNT will be the total number of options in the OPTION list. OPTNO is the option number of an OLPARS program. MENUNO is the menu number of the option given. The combined numbers, OPTNO and MENUNO, determine the option list to be returned to the calling program. If the given option number does not have an option list, OPTCNT will equal 1 and the option name 'ANYTHING' will be returned in OPTION.

PROGRAM USAGE:

```
CALL GETOPT(OPTION,OPTCNT,OPTNO,MENUNO)
```

INPUT ARGUMENTS:

```
 OPTNO - INTEGER; option number of an OLPARS command
MENUNO - INTEGER; the menu number of the given option
```

OUTPUT ARGUMENTS:

```
OPTION - LOGICAL *1 (10,15); the array of option names
OPTCNT - INTEGER; total number of options in the option
                  list
```

FILES:

OLPARS option file (OPTION.OLP)

REFERENCED BY:

MENU

SEE ALSO:

Appendix B, OLPARS Programmer and System Maintenance Manual, for format of option file

OLPARS Program Specifications
GETOPT

HISTORY:

designed by Steve Haehn June 28, 1978

programmed by Donna Morris August 31,1979

PROGRAM NAME: GETPR

CATEGORY: Utility Routine

PROGRAM DESCRIPTION:

GETPR displays a list of class symbols on the screen and prompts the user for a pair of symbols. The user may also enter a '*' for the next class pair in sequence, (if this is the first time the routine is called, the first pair is returned), or a ',' to exit (successfully).

PROGRAM USAGE:

```
INTEGER GETPR

NPAIR=GETPR(TRUCLS,NAMLST)
```

INPUT ARGUMENTS:

TRUCLS - INTEGER; The number of classes at the logic node

NAMLST - INTEGER array (NODLEN,TRUCLS); the list of class names

OUTPUT ARGUMENTS:

NPAIR - INTEGER; the index to the class pair in a lower triangular matrix

REFERENCED BY:

THRESHMOD, OPTIMLMOD

SUBPROGRAMS REFERENCED:

EQUALA, ERASE, FLUSHT, INSPGM, INSRET, TRMGET, TRMPUT

HISTORY:

designed by John W. Tenney Sept. 10, 1981

programmed by John W. Tenney Oct. 9, 1981

OLPARS Program Specifications
GETSTR

PROGRAM NAME: GETSTR

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

GETSTR reads a character string from the terminal

PROGRAM USAGE:

```
INTEGER GETSTR
N = GETSTR(LINE)
```

OUTPUT ARGUMENTS:

LINE - LOGICAL *1 ARRAY; character string read from the terminal

ALGORITHM / NOTES:

The value of the function is the number of successfully matched and assigned input items. A negative one indicates a null line or carriage return.

REFERENCED BY:

GEN

SUBPROGRAMS REFERENCED:

TRMGET

HISTORY:

designed by Steven Haehn January 30,1979

programmed by Donna Morris March 2,1979

PROGRAM NAME: GIN

CATEGORY: Terminal I/O (graphic, system dependent)

PROGRAM DESCRIPTION:

GIN places the graphic terminal into graphics input mode and waits for the return of a graphics screen coordinate and the character that was typed in to send the coordinate.

PROGRAM USAGE:

```
CALL GIN(X,Y,CHAR)
```

INPUT ARGUMENTS:

```
   X - INTEGER; the X graphics screen coordinate
   Y - INTEGER; the Y graphics screen coordinate
CHAR - INTEGER; the character
```

REFERENCED BY:

DRAWBNDY, SCALZM

HISTORY:

designed by Steven E. Haehn April 6, 1978

programmed by David J. Tipton June 28, 1979

PROGRAM NAME: GLONOD

CATEGORY: Data Tree File Access Routine

PROGRAM DESCRIPTION:

GLONOD gets the list of lowest nodes underneath a given node (whose entry slot number is passed in SLTNO). GLONOD is also passed the file descriptor of the TI file and its entry table (in ET). It returns the number of lowest nodes (NLOWND), the entry slot numbers (in Sna) and the names of the nodes (in NMA).

GLONOD operates as follows. It gets the number of lowest nodes underneath the given node directly from element 6 of the TI file entry for the given node. From element 12 of that entry it goes to the first lowest node beneath the given node. Using the lowest node link pointers (element 13), the rest of the lowest nodes can be retrieved.

PROGRAM USAGE:

CALL GLONOD(FIDTI,ET,SLTNO,NLOWND,SNA,NMA)

INPUT ARGUMENTS:

FIDTI - INTEGER; the file descriptor of the TI file

ET - INTEGER array; the entry table of the TI file

SLTNO - INTEGER; entry table slot number of the given node

OUTPUT ARGUMENTS:

NLOWND - INTEGER; number of lowest nodes

SNA - INTEGER array; slot numbers of the lowest nodes

NMA - INTEGER array(4,50); names of the lowest nodes

FILES:

TI - tree information
Instrumentation files

REFERENCED BY:

COMNOD,DSPROJ

SUBPROGRAMS REFERENCED:

TIGCOP, TIGNAM

HISTORY:

designed by Dave Birnbaum June 5, 1978

programmed by Donna Morris June 1,1979

OLPARS Program Specifications
GLSTRC

PROGRAM NAME: GLSTRC

CATEGORY: Logic Tree File Access Routine

PROGRAM DESCRIPTION:

GLSTRC retrieves from the logic information (LI) file the desired structural information to be used in drawing a portion of a logic tree.

PROGRAM USAGE:

```
INTEGER GLSTRC
   .
   .
LOWEST =GLSTRC (FDLI, STRTND, CUTOFF, NXSTRT, LEVELS,
        NODPTR, PARENT, LOGNOD, KIDFLG, OPTION, LTYPE,
        REJECT, NODCNT, LOWCNT)
```

INPUT ARGUMENTS:

FDLI - INTEGER; the file descriptor of the LI file being accessed

STRTND - INTEGER; the entry (logic node) number of the node that represents the root of the tree (or substructure) desired

CUTOFF - INTEGER; the lowest level of the structure to be returned

NXSTRT - INTEGER; the entry (logic node) number of the first node to be returned on the next level after STRTND

OUTPUT ARGUMENTS:

LEVELS - INTEGER array (LEVMAX +1); pointers into NODPTR (see ALGORITHM/NOTES)

NODPTR - INTEGER array (MAXCLS*2); pointers into PARENT, LOGNOD, KIDFLG, REJECT, OPTION, and LTYPE arrays (see ALGORITHM/NOTES)

PARENT - INTEGER array (MAXCLS*2); parent pointers of the logic nodes within the desired logic tree structure

LOGNOD - INTEGER array (MAXCLS*2); logic node numbers

KIDFLG - INTEGER array (MAXCLS*2); numbers of children

or MORE if children exist but the logic node is on the CUTOFF level

OPTION - INTEGER array (MAXCLS*2); option numbers in OLPARS option file associated with the logic nodes (logic nodes without logic have 0 in their entry)

LTYPE - INTEGER array; (MAXCLS*2); logic types (logic nodes that are incomplete or complete have 0 for their entry)

REJECT - INTEGER array; (MAXCLS*2); independent Boolean reject logic indicators ('*' or blank)

NODCNT - INTEGER; total number of nodes found

LOWCNT - INTEGER; total number of lowest nodes found

ALGORITHM / NOTES:

If a cutoff level has been specified by setting 'CUTOFF' to a value other than zero, any logic nodes at the cutoff level, as well as any lowest logic nodes above that level, will be returned as lowest nodes.

The value of the index into the 'LEVELS' array represents the logic node level of the logic tree being drawn.

Example: LEVELS(2), shows we are dealing with the second level of nodes in the logic tree.

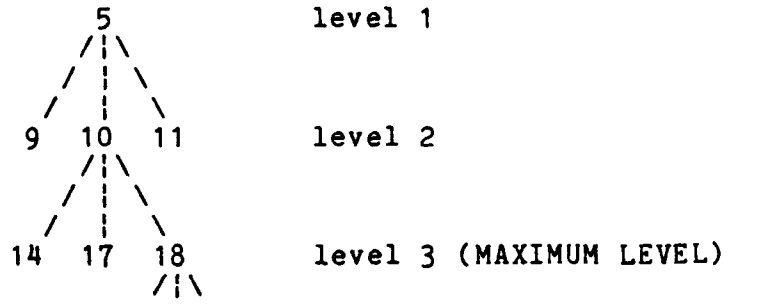
Note, the level being considered will only agree with the actual level value found in the node if the 'current node' is the senior node of the tree. Therefore, the index into the 'LEVELS' array may be considered the 'relative' level of the current data set.

The 'LEVELS' array contains pointers to the 'NODPTR' array. The values in 'LEVELS' point to the first logic node found at the given level.
The expression 'LEVELS(i+1) - LEVELS(i)' for values of i from 1 to MAXIMUM LEVEL + 1 will yield the number of nodes found at 'LEVELS(i)'.

The 'NODPTR' array contains pointers to the 'logic tree information' arrays ('PARENT', 'LOGNOD', 'KIDFLG', 'OPTION', 'LTYPE', 'REJECT'). The contents of the 'tree information' arrays are in the order as obtained by the 'gnxtnd' subprogram. The 'NODPTR' values can be considered pointers to a record containing the following information about a node in a logic tree:

1) parent pointer of a logic node
2) logic node number
3) number of children (or 'MORE' flag)
4) option number (if logic exists)
5) logic type (if logic exists)
6) independent Boolean reject logic indicator

The following figure shows the relationship between the 'LEVELS', 'NODPTR', and 'logic tree information' arrays for the structure beginning at logic node 5. STRTND is set to 5, CUTOFF is set to 3, and NXSTRT is 0.



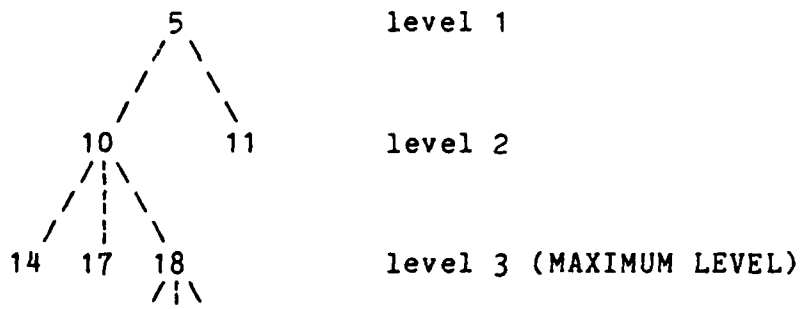
LEVELS	1	2	5	8

(LEVELS VALUES)

NODPTR	1	2 - 3 - 7	4 - 5 - 6	

	1	2	3	4	5	6	7
PARENT	0	1	1	3	3	3	1
LOGNOD	5	9	10	14	17	18	11
KIDFLG	3	0	3	0	0	-1	0
OPTION	230	0	205	0	0	310	0
LTYPE	2	0	2	0	0	3	0
REJECT	*		*				

The following example shows how 'NXSTRT' may be used in order to retrieve a specific node on the level below 'STRTND'. Using the same start logic node (STRTND = 5), CUTOFF again set to 3, and NXSTRT now at 10, the following will be returned by 'GLSTRC'.



LEVELS	1	2	4	7

(LEVELS VALUES)

NODPTR	1	2 - 6	3 - 4 - 5	

	1	2	3	4	5	6
PARENT	0	1	2	2	2	1
LOGNOD	5	10	14	17	18	11
KIDFLG	3	3	0	0	-1	0
OPTION	230	205	0	0	310	0
LTYPE	2	2	0	0	3	0
REJECT	*	*				

OLPARS Program Specifications
GLSTRC

FILES:

LI - logic information

REFERENCED BY:

DRAWLOG

SUBPROGRAMS REFERENCED:

GNXTLN INSINT INSPGM INSRET LIGLOG LIGSTR TRMPUT

DIAGNOSTICS:

If the tree structure of the logic tree is corrupted, GLSTRC will return a -1 as its function value.

SEE ALSO:

GDSTRC

HISTORY:

designed by Steven Haehn December 16, 1980

programmed by Dave Tipton May 29, 1981

PROGRAM NAME: GNLIAE

CATEGORY: Logic Tree File Access Routine

PROGRAM DESCRIPTION:

GNLIAE gets the next available entry in the LI file. It returns the entry number as its value and updates the free entry list.

PROGRAM USAGE:

```
INTEGER GNLIAE
   .
   .
NXTAVN = GNLIAE(FIDLI)
```

INPUT ARGUMENTS:

FIDLI - INTEGER; the file descriptor of the LI file

OUTPUT ARGUMENTS:

GNLIAE - INTEGER; the entry number of the next available logic node

ALGORITHM / NOTES:

Get head and tail pointers to LI free list

Set GNLIAE = head ptr.

IF (next available node entry (head) equals next open entry at end of file (tail)) THEN

Increment head and tail by 1 and write back into file (LIPCOP).

ELSE

Get entry link from the entry pointed to by head (LIGLOG).

Set head equal to the negative value of this entry link and write into LI header (LIPCAP).
(Remember, free list links are negative values. That's why we have to use negative value above)

ENDIF

RETURN

OLPARS Program Specifications
GNLIAE

FILES:

LI - logic information

SUBPROGRAMS REFERENCED:

LIGCAP, LIPCOP, LIGLOG

HISTORY:

designed by David Birnbaum September 19, 1978

programmed by Steven Haehn March 15, 1981

PROGRAM NAME: GNLVAE

CATEGORY: Logic Tree File Access Routine

PROGRAM DESCRIPTION:

GNLVAE gets the next available entry (or entries) in the LV file. It returns a pointer to the first entry it has obtained and the next available entry (link of the last entry obtained). The free list is updated and all the obtained entries are automatically linked together.

PROGRAM USAGE:

```
INTEGER GNLVAE
  .
  .
NXTAVR = GNLVAE (FDLV, WANTED, LSTLNK)
```

INPUT ARGUMENTS:

```
  FDLV - INTEGER; the file descriptor of the LV file
WANTED - INTEGER; the number of entries wanted
LSTLNK - INTEGER; pointer to the next available entry
                  (link of last entry in obtained region)
```

OUTPUT ARGUMENTS:

```
GNLVAE - INTEGER; pointer to first entry of obtained
                  entries
```

ALGORITHM / NOTES:

```
Obtain the free list control variables and immediately
set our function value to the head of the free list
(LVGHDR)

WHILE (there are more entries to be obtained)

    IF (there are no entries in the free list) THEN

        Obtain the entry at the end of the file

    ELSE

        Pull next entry from free list

    ENDIF

    Put next available entry value into entry itself
    (LVPLNK)
```

ENDWHILE

Now its time to update the LV header with the number of entries in use and the new free list pointers

RETURN

FILES:

LV - logic value

SUBPROGRAMS REFERENCED:

INSINT, INSPGM, INSRET, LVGHDR, LVGLNK, LVPHDR, LVPLNK

HISTORY:

designed by David BirnBaum September 19, 1978

programmed by Steven Haehn April 1, 1981

PROGRAM NAME: GNSMAE

CATEGORY: Level II File Access Routine

PROGRAM DESCRIPTION:

GNSMAE gets the next available entry in the saved transformation matrix (SM) file. It returns the vector's entry number as its value and updates the free list pointers.

PROGRAM USAGE:

```
INTEGER GNSMAE

I = GNSMAE (FDSM, WANTED, LSTLNK)
```

INPUT ARGUMENTS:

FDSM - INTEGER; the file descriptor of the SM file

WANTED - INTEGER; the number of available entries to be retrieved

OUTPUT ARGUMENTS:

LSTLNK - INTEGER; the last link pointer retrieved

GNSMAE - INTEGER; the first link pointer retrieved

ALGORITHM / NOTES:

```
Retrieve the SM file header (SMGHDR).

DO i=1, number of available entry numbers to return

   IF (there are no entries in the free list) THEN

      Obtain the entry at the end of the file

   ELSE

      Retrieve the next entry from the free list
      (SMGLNK).

   ENDIF

   Fix the free list links (SMPLNK).

ENDDO
```

OLPARS Program Specifications
GNSMAE

Update the SM file header (SMPHDR).

RETURN

FILES:

SM - saved transformation matrix file
instrumentation file

SUBPROGRAMS REFERENCED:

INSINT, INSPGM, INSRET, SMGHDR, SMGLNK, SMPHDR,
SMPLNK

HISTORY:

designed by Jill King March 13,1981

programmed by Jill King March 13,1981

PROGRAM NAME: GNXTAN

CATEGORY: Data Tree File Access Routine

PROGRAM DESCRIPTION:

GNXTAN returns the slot number of the next available node within the entry table (ENTBLE) of a tree information file. The entry table slot will be filled, by GNXTAN, with the next available node (initially, the slot contains zero). If there is no available room in the entry table, GNXTAN returns zero.

PROGRAM USAGE:

```
INTEGER GNXTAN
        .
        .
        .
SLOTNO = GNXTAN(FDTI,ENTBLE)
```

INPUT ARGUMENTS:

FDTI - INTEGER; file descriptor of the TI file

ENTBLE - INTEGER array (TABSIZ); the entry table

OUTPUT ARGUMENTS:

GNXTAN - INTEGER; next available slot number within the entry table

ENTBLE - INTEGER array (TABSIZ); the next available node number is written to the first empty slot

ALGORITHM / NOTES:

```
DO, slot pointer = 1, TABSIZ (100)

    Look for an empty slot.

    IF (at end of table) RETURN.

ENDDO
```

IF(HEAD POINTER .EQ. TAIL POINTER) THEN

Obtain the next available node entry from the TI file and put in the empty slot.

Update the node free list in the TI file.

ELSE (HEAD POINTER DID NOT EQUAL TAIL POINTER)

Fill the entry table slot with the pointer to the head of the free list

Update the node free list in the TI file

ENDIF
RETURN

FILES:

TI - tree information

REFERENCED BY:

APPEND

SUBPROGRAMS REFERENCED:

INSINT, INSPGM, INSRET, TIGCOP, TIGHDR, TIPHDR

HISTORY:

designed by Steven Hæhn June 8, 1978

programmed by Mark Maginn August 8, 1980

PROGRAM NAME: GNXTLN

CATEGORY: Logic Tree File Access Routine

PROGRAM DESCRIPTION:

GNXTLN obtains the structure of the "next" node within the given logic tree. The "next" node is defined to be the child of the current node; the sibling of the current node, when there are no children; or the sibling of the parent node, when the current node has no sibling and is not the "starting" node. The "starting" node is the first node in the tree structure that GNXTLN is to look under for more nodes.

PROGRAM USAGE:

```
LOGICAL GNXTLN
   .
   .

IF(GNXTLN(FDLI,STRTND,CRNTND,STRUCT))
```

INPUT ARGUMENTS:

FDLI - INTEGER;	file descriptor of the LI file being accessed
STRTND - INTEGER;	the starting node number (entry number) of the logic tree traversal
CRNTND - INTEGER;	the node number of the previously retrieved (or initial)) node; represents the current node in the tree traversal algorithm
*STRUCT - INTEGER	array (LSTRSZ); the structure pointer variables of the current node

---

* The table is optionally initialized, see Algorithms/Notes.

OUTPUT ARGUMENTS:

CRNTND - INTEGER; the entry (node) number of the current node just retrieved

STRUCT - INTEGER array (LSTRSZ); the structure pointer variables of the current node

ALGORITHM / NOTES:

```
GNXTLN = .TRUE.

IF (the current node has a value less than zero) THEN

    Retrieve the starting node before continuing on.

ENDIF

IF (children exist) THEN

    Make first child present node.

ELSEIF (present node is not starting node and a
        sibling exists)

    Make sibling present node.

ELSE

    WHILE (present node is not starting node and
           present node has no siblings)

        Make parent of present node, the present node.

    ENDWHILE

    IF (present node is not starting node) THEN

        Make sibling present node.

    ELSE

        GNXTLN = .FALSE.

    ENDIF

ENDIF

RETURN
```

NOTE

If the calling program doesn't happen to have the structure elements of the starting node, it should set the current node variable equal to the negative value of the starting node, so that GNXTLN will obtain the starting node structure elements.

FILES:

LI - logic information

REFERENCED BY:

GLSTRC

SUBPROGRAMS REFERENCED:

INSINT, INSLOG, INSPGM, INSRET, LIGSTR

DIAGNOSTICS:

If there is no next node, i.e., all nodes after the starting node have been seen, GNXTLN will return a 'false' value.

SEE ALSO:

GNXTND (for pictorial example of tree traversal algorithm)

HISTORY:

designed by Steven Haehn September 19, 1978

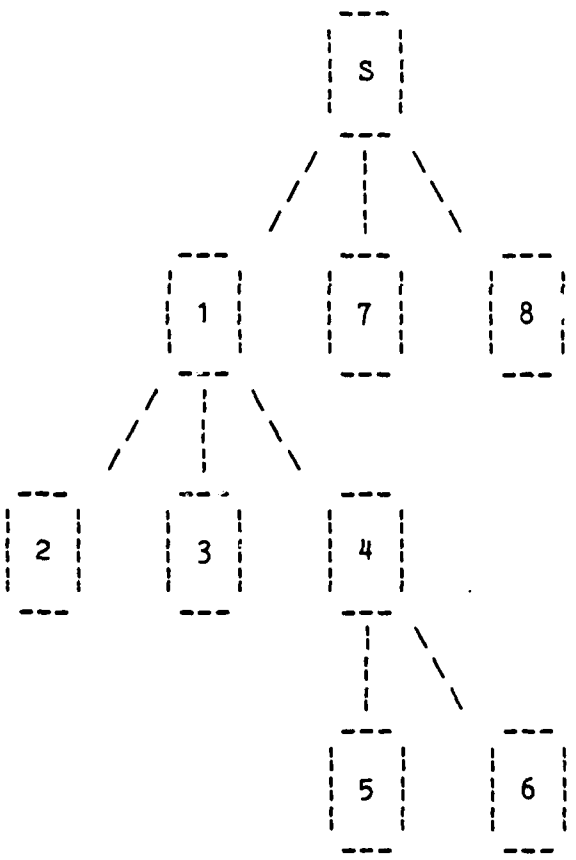
progammed by Steven Haehn Nov. 19, 1980

PROGRAM NAME: GNXTND

CATEGORY: Data Tree File Access Routine

PROGRAM DESCRIPTION:

GNXTND obtains the name and the counts and pointers of the "next" node within the given data tree. The "next" node is defined to be the child of the current node; the sibling of the current node, when there are no children; or the sibling of the parent node, when the current node has no sibling and is not the "starting" node. The "starting" node is the first node in the tree structure that GNXTND is to look under for more nodes.



Nodes are returned in the order indicated above

PROGRAM USAGE:

```
LOGICAL GNXTND
      .
      .
      .
IF (GNXTND(TIFD,ENTAB,STRTND,CRNTND,NODNAM,CAPTBL))
```

INPUT ARGUMENTS:

TIFD - INTEGER; file descriptor of the TI file being looked at

ENTAB - INTEGER ARRAY (TABSIZ); entry table of the current TI file being accessed

STRTND - INTEGER; an entry table slot number representing the starting point of the tree traversal

CRNTND - INTEGER; an entry table slot number representing the current node to be used

CAPTBL - INTEGER ARRAY (CAPSIZ); counts and pointers of the current node (NOTE: The table is optionally initialized, see NOTES.)

OUTPUT ARGUMENTS:

CRNTND - INTEGER; an entry table slot number representing the current node obtained

NODNAM - INTEGER ARRAY; node name of the current node

CAPTBL - INTEGER ARRAY (CAPSIZ); counts and pointers of the current node

ALGORITHM / NOTES:

```
GNXTND = .TRUE.

IF (the current node has a value less than zero) THEN

    Retrieve the starting node before continuing on.

ENDIF
```

```
IF (children exist) THEN

    Make first child present node.

ELSEIF (present node is not starting node and a sibling
       exists)

    Make sibling present node.

ELSE

    WHILE (present node is not starting node and present
          node has no siblings)

        Make parent of present node, the present node.

    ENDWHILE

    IF (present node is not starting node) THEN

        Make sibling present node.

    ELSE

        GNXTND = .FALSE.
        (No more nodes in the tree)

    ENDIF

ENDIF

RETURN
```

NOTE

If the calling program does not happen to have the counts and pointers of the starting node, it should set the current node variable equal to the negative value of the starting node, so that GNXTND will obtain the starting node counts and pointers.

FILES:

TI - tree information

REFERENCED BY:

TISRCH, GDSTRC, UNIQND

SUBPROGRAMS REFERENCED:

TIGCAP, TIGNAM

DIAGNOSTICS:

If there is no next node, i.e. all nodes after the starting node have been seen, GNXTND will return with a 'false' value.

HISTORY:

designed by Steven Haehn August 3, 1978

programmed by Steven Haehn Sept. 13, 1979

PROGRAM NAME: GOPTNM

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

GOPTNM returns to the calling program the option name in the OLPARS option file header that corresponds to the option number passed to GOPTNM. If the given option number can not be found, the option name will be set to blanks.

PROGRAM USAGE:

CALL GOPTNM (FDCM, OPTNUM, OPTNAM)

INPUT ARGUMENTS:

FDCM - INTEGER; file descriptor of the communications (CM) file
OPTNUM - INTEGER; option number of an OLPARS command

OUTPUT ARGUMENTS:

OPTNAM - INTEGER ARRAY (PGMLEN+1); option name that corresponds to OPTNUM

FILES:

OLPARS option file (OPTION.OLP)

SUBROUTINES REFERENCED:

CLOSBL CMGDIR FGET INSINT INSPGM
INSRET OEXIT OPENBL TRMPUT

REFERENCED BY:

GLSTRC OLTDMP

SEE ALSO:

Appendix B, OLPARS Programmer and System Maintenance Manual, for format of option file

HISTORY:

designed by Dave Tipton April 28, 1981

programmed by Dave Tipton April 28, 1981

PROGRAM NAME: GRIDIS

CATEGORY: Terminal Display Routine

PROGRAM DESCRIPTION:

GRIDIS is passed the set of screen coordinates in ISCREN and the grid matrix in GRIDMX. It displays the grid matrix on the screen.

PROGRAM USAGE:

CALL GRIDIS(ISCREN,GRIDMX)

INPUT ARGUMENTS:

ISCREN - INTEGER array (S"2NUM); screen coordinates at user's terminal

GRIDMX - INTEGER array (number of rows in the cluster plot grid,number of columns in the cluster plot grid); cluster matrix to be displayed at user's terminal

SUBPROGRAMS REFERENCED:

MARK

REFERENCED BY:

CLUS

HISTORY:

designed by Dave Birnbaum June 23, 1978

programmed by David Tipton June 13, 1979

PROGRAM NAME: GTLUN

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

GTLUN searches through the logical unit number table (LUNTBL) found in the CALUN (currently available logical unit numbers) common area for an available logical unit that can be used by the requesting program. Once it finds an available unit, GTLUN marks it as being used.

PROGRAM USAGE:

CALL GTLUN(LUN)

OUTPUT ARGUMENTS:

LUN - INTEGER; an available logical unit number

REFERENCED BY:

VCREAT, FCREAT, VOPEN, FOPEN

DIAGNOSTICS:

GTLUN will return a -1 (an invalid LUN) when there are no available logical units

SEE ALSO:

RELUN

HISTORY:

designed by Steven Haehn Dec. 15, 1978

programmed by Steven Haehn Feb. 28, 1979

PROGRAM NAME: GTRGNI

CATEGORY: UTILITY SUBROUTINE

PROGRAM DESCRIPTION:

GTRGNI (get region, convert retrieved elements to integers) retrieves a region (a series of entries) from a Logic Value file. The regions obtained represent specific entities used by the individual calling programs (i.e., pointers to linked lists).

PROGRAM USAGE:

```
CALL GTRGNI(FDLV,ELEMCT,LVETSZ,REGION,RGNPTR,NXTRGN)
```

INPUT ARGUMENTS:

FDLV - INTEGER; File descriptor of the LV file

ELEMCT - INTEGER; the number of elements to be read from the LV region.

LVETSZ - INTEGER; logic value entry size (found in Logic Information file header)

RGNPTR - INTEGER; pointer to the region to be read

OUTPUT ARGUMENTS:

REGION - INTEGER array(ELEMCT); the place to store the region found in the LV file

NXTRGN - INTEGER; pointer to the next region following the current region (or an end-of-region link)

FILES:

LV - logic value

REFERENCED BY:

CREATLOG

SUBPROGRAMS REFERENCED:

INSINT, INSPGM, INSRET, LVGENT

OLPARS Program Specifications
GTRGNI

SEE ALSO:

GTRGNR, PTRGNI, PTRGNR

HISTORY:

designed by Steven Haehn March 5, 1981

programmed by Steven Haehn April 1, 1981

PROGRAM NAME: GTRGNR

CATEGORY: UTILITY SUBROUTINE

PROGRAM DESCRIPTION:

GTRGNR (get region of real elements) retrieves a region (a series of entries) from a Logic Value file. The regions obtained represent specific entities used by the individual calling programs (i.e., group logic discriminant vectors).

PROGRAM USAGE:

```
CALL GTRGNR(FDLV,ELEMCT,LVETSZ,REGION,RGNPTR,NXTRGN)
```

INPUT ARGUMENTS:

FDLV - INTEGER; File descriptor of the LV file

ELEMCT - INTEGER; the number of elements to be read from the LV region.

LVETSZ - INTEGER; logic value entry size (found in Logic Information file header)

RGNPTR - INTEGER; pointer to the region to be read

OUTPUT ARGUMENTS:

REGION - REAL array(ELEMCT); the place to store the region found in the LV file

NXTRGN - INTEGER; pointer to the next region following the current region (or an end-of-region link)

FILES:

LV - logic value

REFERENCED BY:

CREATLOG

SUBPROGRAMS REFERENCED:

INSINT, INSPGM, INSRET, LVGENT

SEE ALSO:

GTRGNI, PTRGNI, PTRGNR

HISTORY:

designed by Steven Haehn March 5, 1981

programmed by Steven Haehn April 1, 1981

PROGRAM NAME: HASH

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

HASH computes an integer number in the range of 1 to TBLSIZ (table size) by 'hashing' the given character string using a division-remainder algorithm.

PROGRAM USAGE:

```
INTEGER HASH
    .
    .
    .

IPTR = HASH(STRING,LNGTH,TBLSIZ)
```

INPUT ARGUMENTS:

STRING - LOGICAL *1 ARRAY(LNGTH); character string to be hashed (system dependent data type)

LNGTH - INTEGER; length of string to be hashed

TBLSIZ - INTEGER; a prime number representing the maximum number 'HASH' can generate

OUTPUT ARGUMENTS:

HASH - INTEGER; an integer number in the range of 1 to TBLSIZ

ALGORITHM / NOTES:

The FORTRAN found on the RSX-11M operating system (in this case, FORTRAN IV PLUS) gives a user the capability of addressing a single character (clearly compiler dependent). This feature is used in the HASH algorithm. The algorithm consists of treating each character as a number, creating a sum from the characters (numbers) in the string (including information about the position of the character), and finally creating a number that lies between 1 and the given table size, using modular arithmetic. Note, the character (byte) data is coerced into an integer data type. Sign extension (i.e., the sign of the byte data (the eighth bit) is propagated thru the high order byte of the integer into the sign portion of the integer) occurs in

the integer during the data type conversion, but a negative number will not appear because all legitimate ASCII characters use only 7 bits out of an eight bit byte (the eighth bit is the sign bit and it has a value of zero). For further details, consult an RSX-11M FORTRAN user's guide.

Using the fact that the maximum legitimate ASCII character (MAXCHAR) value is equivalent to the decimal number 127, and the maximum size positive decimal number a 16 bit integer (using 2's complement notation) can contain is 32767, it was estimated that a character string consisting of only MAXCHARs could hold 256 characters without causing an overflow condition during the summation portion of the algorithm (i.e., 32767/128 is approximately equal to 256). This is definitely an adequate length for the character strings being used. (Under RSX-11M FORTRAN, a literal string can only have a maximum length of 256).

The table size input variable should be a prime number because this will reduce the frequency of 'collisions' (or repetitions) of the values generated by HASH.

REFERENCED BY:

GETHST

HISTORY:

designed by Steven Haehn Dec. 2, 1978

programmed by Steven Haehn Apr. 4, 1978

PROGRAM NAME: HELOLP

CATEGORY: Utility Command (system dependent)

PROGRAM DESCRIPTION:

HELOLP will "sign" a user on to OLPARS, i.e. it will validate his/her presence on OLPARS, place the user in their own directory, and perform file initializations within that directory.

PROGRAM USAGE:

User types in 'HELOLP'.

USER INTERACTION:

The user types in a "login id.", the terminal being used, and any other initializing information needed to start up OLPARS.

ALGORITHM / NOTES:

The HELOLP function is completely described in the OLPARS VI Programmer's Reference Manual in appendix A. It is implemented via an RSX-11M command file.

REFERENCED BY:

OLPARS user.

SEE ALSO:

BYEOLP

HISTORY:

designed by Steven Haehn July 28, 1978

programmed by Steven Haehn June 19, 1981

PROGRAM NAME: HELP

CATEGORY: Utility Command (system dependent)

PROGRAM DESCRIPTION:

HELP gives the OLPARS user a helpful description of an OLPARS subject or of a specific OLPARS command. The user may also obtain a listing of all available help files. All help is written to the terminal.

PROGRAM USAGE:

User types in 'HELP'.

USER INTERACTION:

The user types in a character string which may be one of the following:
(1) a command name
(2) the name of an OLPARS subject
(3) the initial characters (sub-string) of a command name or subject name
(4) 'ALL'

Typing 'ALL' will give the user a listing of all available help files. To leave the HELP program, the user types in the OLPARS 'exit command' character.

ALGORITHMS / NOTES:

Read the OLPARS directory string from the CM file.

Use the OLPARS directory string to create the complete pathname to the help dictionary file (HELP.TXT) and to the file containing the directory string and file extension of the OLPARS help files (HELPDIR.TXT).

If (HELP.TXT and HELPDIR.TXT are successfully opened)

Read the character string in HELPDIR.TXT. It should contain a directory string indicating where the help files are located, along with the filename extension of the help files.

Search for a right bracket (]) in the character string, which indicates the end of the directory string.

```
   Starting at the position after the right bracket
   (in case the directory string contains a period),
   search for a period (.), which indicates the
   beginning of the filename extension.

   DONE = .FALSE.

   REPEAT
      Get the request for help from the user.

      IF( the user wants to exit) break

      Create the pathname to the help file to be
      accessed:

      IF(the user-typed string is in help dictionary)
         the user may have typed a sub-string of a
         subject - get the entire name from the help
         dictionary and append it to the help directory
         string.

      ELSE
         append the user-typed character string to
         the help directory string

      ENDIF

      To complete creation of the pathname of the
      help file to be accessed, append the help
      filename extension plus a null byte to the
      end of the character string.

      IF(help file is opened successfully)then

         copy each line of the help file to the
         terminal.

         NEXT

      ELSE

         give user a message - 'NO HELP AVAILABLE
                               FOR THE USER REQUEST'

      ENDIF

   UNTIL(DONE)

ENDIF
```

OLPARS Program Specifications
HELP

FILES:

Block I/O	Record I/O
CM	Terminal
OPTION.OLP	HELP.TXT
	HELPDIR.TXT
	the help file

REFERENCED BY:

OLPARS user.

DIAGNOSTICS:

If a legitimate 'RSX' filename cannot be created from the user-typed character string, or if the help file does not exist or cannot be opened, the user will be notified that no help is available for the given request.

HISTORY:

designed by Steve Haehn June 30, 1978

programmed by Donna Morris July 29, 1981

PROGRAM NAME: HSGHDR

CATEGORY: Utility subroutine (system dependent)

PROGRAM DESCRIPTION:

HSGHDR retrieves the header portion of the instrumentation package's historic records file. The header currently contains the pointer to the overflow (collision) area of the historic records file.

PROGRAM USAGE:

CALL HSGHDR(HISTFD,NXTREC)

INPUT ARGUMENTS:

HISTFD - INTEGER; historic records file descriptor

OUTPUT ARGUMENTS:

NXTREC - INTEGER; next available record in the historic file's overflow area (i.e., the end of the file)

ALGORITHM / NOTES:

Integer 'OVF' is equivalenced over real variable 'OVFLPT' (system dependent) (i.e., we're mapping an integer variable onto a real variable so there is no real conversion step; FGET can only read real elements).

REFERENCED BY:

GETHST

SUBPROGRAMS REFERENCED:

FGET

SEE ALSO:

HSPHDR

HISTORY:

designed by Steven Haehn Feb. 7 1979

programmed by Steven Haehn Apr. 6 1979

OLPARS Program Specifications
HSGREC

PROGRAM NAME: HSGREC

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

HSGREC retrieves a record from the instrumentation historic records file

PROGRAM USAGE:

CALL HSGREC(HSFD,RECPTR,LINK,TALLY,NAMLEN,NAME)

INPUT ARGUMENTS:

HSFD - INTEGER; historic records file descriptor

RECPTR - INTEGER; record pointer to historic record to be read

OUTPUT ARGUMENTS:

LINK - INTEGER; collision linkage pointer (when zero, no collisions have occurred)

TALLY - INTEGER; number of times to print out debug information

NAMLEN - INTEGER; length of 'name'

NAME - LOGICAL *1 ARRAY; name of historic record

ALGORITHM / NOTES:

NOTE: equivalences of real and integer variables are being done. This is a system dependent attribute.

a history record looks like this:

overflow

<Link>	<Tally>	<Name length>	<Name>
1 REL.	1 REL.		3 REL.

REL. = Real element

FILES:

HS - Instrumentaton History

REFERENCED BY:

GEN, GETHST

SUBPROGRAMS REFERENCED:

FGET

HISTORY:

designed by Steve Haehn February 7,1979

programmed by Steve Haehn February 7,1979

PROGRAM NAME: HSPHDR

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

HSPHDR writes the header portion of the instrumentation debug package's historic records file.

PROGRAM USAGE:

```
CALL HSPHDR(HISTFD,NXTREC)
```

INPUT ARGUMENTS:

HISTFD - INTEGER; historic records file descriptor

NXTREC - INTEGER; next available record in the historic files overflow area

ALGORITHM / NOTES:

NOTE: integer array 'ovf' is equivalenced over real variable 'ovflpt' (system dependent) (i.e.,we're mapping an integer variable into a real variable so no time is taken for a 'integer to real' conversion)

FILES:

HS - Instrumentation Historic File

REFERENCED BY:

GEN, GETHST

SUBPROGRAMS REFERENCED:

FPUT

HISTORY:

designed by Steve Haehn February 7,1979

programmed by Steve Haehn February 7,1979

PROGRAM NAME: HSPLNK

CATEGORY: Utility subroutine (system dependent)

PROGRAM DESCRIPTION:

HSPLNK writes out the linkage portion of a historic record in the instrumentation package history file.

PROGRAM USAGE:

```
CALL HSPLNK(HISTFD,HRECPT,LINK)
```

INPUT ARGUMENTS:

```
HISTFD - INTEGER; historic records file descriptor
HRECPT - INTEGER; historic record pointer
  LINK - INTEGER; link to be written into the historic
                  record pointed to by 'HRECPT'
```

ALGORITHM / NOTES:

Integer array 'ILINK' is equivalenced over real variable 'RLINK' (system dependent) (i.e., we're mappping an integer variable onto a real variable so that there is no integer to real conversion).

REFERENCED BY:

GETHST

SUBPROGRAMS REFERENCED:

FPUT

HISTORY:

designed by Steven Haehn Feb. 24, 1979

programmed by Steven Haehn Apr. 6, 1979

OLPARS Program Specifications
HSPREC

PROGRAM NAME: HSPREC

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

HSPREC writes a record to the instrumentation historic records file.

PROGRAM USAGE:

CALL HSPREC(HSFD,RECPTR,LINK,TALLY,NAMLEN,NAME)

INPUT ARGUMENTS:

HSFD - INTEGER; historic records file descriptor

RECPTR - INTEGER; record pointer to historic record to be read

OUTPUT ARGUMENTS:

LINK - INTEGER; collision linkage pointer (when zero, no collisions have occurred)

TALLY - INTEGER; number of times to print out debug information

NAMLEN - INTEGER; length of 'name'

NAME - LOGICAL *1 ARRAY; name of historic record

ALGORITHM / NOTES:

NOTE: equivalences of real and integer variables are being done. This is a system dependent attribute.

a history record looks like this:

overflow

<Link>	<Tally>	<Name length>	<Name>
1 REL.	1 REL.		3 REL.

REL. = Real element

FILES:

HS - Instrumentaton History

REFERENCED BY:

GEN, GETHST

SUBPROGRAMS REFERENCED:

FGET

HISTORY:

designed by Steve Haehn February 7,1979

programmed by Steve Haehn February 7,1979

PROGRAM NAME: IDCHK

CATEGORY: Utility Routine

PROGRAM DESCRIPTION:

IDCHK checks the validity of the character string (NAME), according to the options given within its argument list. The character string MUST be terminated by a zero (the end-of-string character) when 'LNGTH' is equal to zero.

If the length of the character string is important, i.e., the character string in 'NAME' must contain 1 to 'MXLEN' characters in order to be a valid identifier, then the 'REQUIR' switch is set to true. When the 'REQUIR' switch is set to false, the length of the character string maybe less than or equal to 'MXLEN' for 'NAME' to be a valid identifier.

'FCHTYP' and 'RESTYP' are numeric codes indicating the valid type of characters allowed in the identifier. FCHTYP is the code used for the first character in the string 'NAME'. 'RESTYP' is the code used for the rest of the character string. The type codes are as follows:

1 - alphabetic
2 - alphanumeric
3 - any visible ASCII character (includes 'blank')

PROGRAM USAGE:

```
LOGICAL IDCHK
   .
   .
IF(IDCHK(NAME,LNGTH,REQUIR,MXLEN,FCHTYP,RESTYP))
```

INPUT ARGUMENTS:

```
  NAME - INTEGER (MAXLIN) ARRAY; the character string
LNGTH  - INTEGER; the length of the character string

REQUIR - LOGICAL; .true. indicates that given 'LNGTH'
                  must be equal to 'MXLEN', .false.
                  means that the the given 'LNGTH'
                  maybe less than or equal to 'MXLEN'

 MXLEN - INTEGER; the maximum length 'NAME' may have
FCHTYP - INTEGER; first character of string type code
RESTYP - INTEGER; rest of string type code
```

ALGORITHM / NOTES:

```
Assume that the "NAME" given is not a legal identifier
(IDCHK = false).

IF (the length of the "NAME" is OKEY-DOKEY) THEN

    IF (the first character of "NAME" is of the type
        "FCHTYP") THEN

        IF (the rest of the characters in "NAME" are of
            type "RESTYP") THEN

            "NAME" is a legal identifier
            (IDCHK = true).

        ENDIF

    ENDIF

ENDIF

RETURN
```

REFERENCED BY:

LGLTRE, LGLNOD

SUBPROGRAMS REFERENCED:

TYPE

HISTORY:

designed by Steven Haehn June 1, 1978

programmed by Steven Haehn April 8, 1980

PROGRAM NAME: IDX

CATEGORY: Utilities Subroutine

PROGRAM DESCRIPTION:

Given the row and column values of a position in a symetric square matrix, IDX computes the corresponding position of the same matrix when stored as a vector using only the lower triangular elements.

PROGRAM USAGE:

```
INTEGER IDX
  .
  .
  .
POSITION = IDX (ROW,COL,NDIM)
```

INPUT ARGUMENTS:

ROW - INTEGER; the row value of the position requested

COL - INTEGER; the column value of the position requested

NDIM - INTEGER; the order of the matrix

OUTPUT ARGUMENTS:

IDX - INTEGER; the position of the requested element in the lower triangular portion of the symetric square matrix, stored as a vector

ALGORITHM / NOTES:

```
Check for valid input data

IF (row value.ge.column value) THEN

   IDX = Col * Ndim - Col * (Col-1) / 2 - Ndim + Row

ELSE

   IDX = Row * Ndim - Row * (Row-1) / 2 - Ndim + Row

ENDIF

RETURN
```

REFERENCED BY:

L2FSHP, S2FSHP

SUBPROGRAMS REFERENCED:

TRMPUT

HISTORY:

designed by Jill King November 3, 1980

programmed by Jill King November 3, 1980

PROGRAM NAME: INDEX

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

INDEX finds a character 'Char' in string 'String'. If 'Char' is found, a character position pointer is returned. Otherwise, a zero is returned. If the character string being scanned does not have an end-of-string symbol (0) appended to it, INDEX looks at MAXLIN-1 characters.

Note: INDEX will return 0 if 'Char' is the eos symbol. (i.e., you are not allowed access to the eos symbol).

PROGRAM USAGE:

```
INTEGER INDEX
     .
     .
     .
N=INDEX(STRING,CHAR)
```

INPUT ARGUMENTS:

STRING - LOGICAL *1 ARRAY; character string for which to find the position of a given character

CHAR - LOGICAL *1; character for which to find the position in a character string

OUTPUT ARGUMENTS:

INDEX - LOGICAL; character position pointer. index will return zero if the given character is not found in the character string.

REFERENCED BY:

Level I Routines

HISTORY:

designed by Steve Haehn January 1,1979

programmed by Steve Haehn January 1,1979

PROGRAM NAME: INITD

CATEGORY: Display File Access Routine (system dependent)

PROGRAM DESCRIPTION:

INITD is passed the ID's of the DI, DV, PV, S1 and CM files. INITD initializes the headers of the DI, DV, PV and S1 files according to the following table.

FILE	ITEM	ACTION
DI, DV, PV, S1	Display Code	Set to 0 (invalid display code)
DI, DV, S1	OLPARS Option No.	Set to option no. of calling command
DI, PV	Data Set	Insert data set names in file
DI, PV	Dimension	Insert dimension of data set in file
DI	Number of Boundaries	Set equal to 0
DI	Type of Scaling	Set equal to 1 (counts/square)
DI	Zoom Flag	Set equal to 0
DI	Intensity Flag	Set equal to 0
DV	Next Available Entry	Set equal to 1
DV	Screen coordinate flag	Set to 0 (off)

PROGRAM USAGE:

```
CALL INITD(FIDDI,FIDDV,FIDPV,FIDS1,FIDCM,PGMNAM)
```

OLPARS Program Specifications
INITD

INPUT ARGUMENTS:

FIDDI - INTEGER; the file descriptor of the DI file
FIDDV - INTEGER; the file descriptor of the DV file
FIDPV - INTEGER; the file descriptor of the PV file
FIDS1 - INTEGER; the file descriptor of the S1 file
FIDCM - INTEGER; the file descriptor of tne CM file
PGMNAM - LOGICAL *1 array (CMDLEN); program name of calling program

FILES:

DI - display information
DV - display value
PV - projection vector
S1 - scratch 1
CM - communication

REFERENCED BY:

Structure analysis and logic design commands

SUBPROGRAMS REFERENCED:

CMGCDS,DIPHDI,DIPNAM,DVPHDR,PVPHDR,PVPNAM,SETOPT,S1PHDR

HISTORY:

designed by Dave Birnbaum June 26, 1978

programmed by Donna Morris May 7,1979

PROGRAM NAME: INSCHR

CATEGORY: Programmer's Aid (system dependent)

PROGRAM DESCRIPTION:

INSCHR (instrumentation character variable dump) prints out the given variable name and the character string stored there. There should only be one character stored per integer.

PROGRAM USAGE:

CALL INSCHR(NAME,VALUE,LNGTH)

INPUT ARGUMENTS:

NAME - LOGICAL*1 ARRAY(VRNLEN+1); HOLLERITH field containing the name of the variable to be printed.

VALUE - INTEGER ARRAY(MAXLIN); character string to be printed.

LNGTH - INTEGER; length of character string to be printed. If less than or equal to zero then a null character string is expected to terminate the string.

REFERENCED BY:

Any OLPARS system independent programs

SUBPROGRAMS REFERENCED:

FILPUT, OEXIT, TRMPUT

SEE ALSO:

INSDBL, INSFLT, INSINT, INSLOG

HISTORY:

designed by Steven Haehn Mar. 19, 1979

programmed by Steven Haehn Apr. 20, 1979

OLPARS Program Specifications
INSDBL

PROGRAM NAME: INSDBL

CATEGORY: Programmer's Aid (system dependent)

PROGRAM DESCRIPTION:

INSDBL (instrumentation double precision floating point variable dump) prints out the given variable name and the value of the floating point number stored there.

PROGRAM USAGE:

CALL INSDBL(NAME,VALUE)

INPUT ARGUMENTS:

NAME - LOGICAL*1 ARRAY(VRNLEN+1); HOLLERITH string containing name of the variable to be printed.

VALUE - REAL; floating point number to be printed.

REFERENCED BY:

All OLPARS system independent programs

SUBPROGRAMS REFERENCED:

FILPUT, OEXIT, TRMPUT

SEE ALSO:

INSCHR, INSFLT, INSINT, INSLOG

HISTORY:

designed by Steven Haehn Mar. 19, 1979

programmed by Steven Haehn Apr. 20, 1979

PROGRAM NAME: INSFLT

CATEGORY: Programmer's Aid (system dependent)

PROGRAM DESCRIPTION:

INSFLT (instrumentation single precision floating point variable dump) prints out the given variable name and the value of the floating point number stored there.

PROGRAM USAGE:

CALL INSFLT(NAME,VALUE)

INPUT ARGUMENTS:

NAME - LOGICAL*1 ARRAY(VRNLEN+1); HOLLERITH string containing name of the variable to be printed.

VALUE - REAL; floating point number to be printed.

REFERENCED BY:

All OLPARS system independent programs

SUBPROGRAMS REFERENCED:

FILPUT, OEXIT, TRMPUT

SEE ALSO:

INSCHR, INSDBL, INSINT, INSLOG

HISTORY:

designed by Steven Haehn Mar. 19, 1979

programmed by Steven Haehn Apr. 20, 1979

OLPARS Program Specifications
INSINI

PROGRAM NAME: INSINI

CATEGORY: Programmer's aid (system dependent)

PROGRAM DESCRIPTION:

INSINI initializes the disabling flag and indentation level of the OLPARS programmer's instrumentation package. When instrumentation is enabled, two files will be opened; the instrumentation output (debug) file and the instrumentation history file.

PROGRAM USAGE:

```
CALL INSINI(NAME)
```

INPUT ARGUMENTS:

NAME - LOGICAL*1 ARRAY; Hollerith string containing the name of the program that has called INSINI.

ALGORITHM / NOTES:

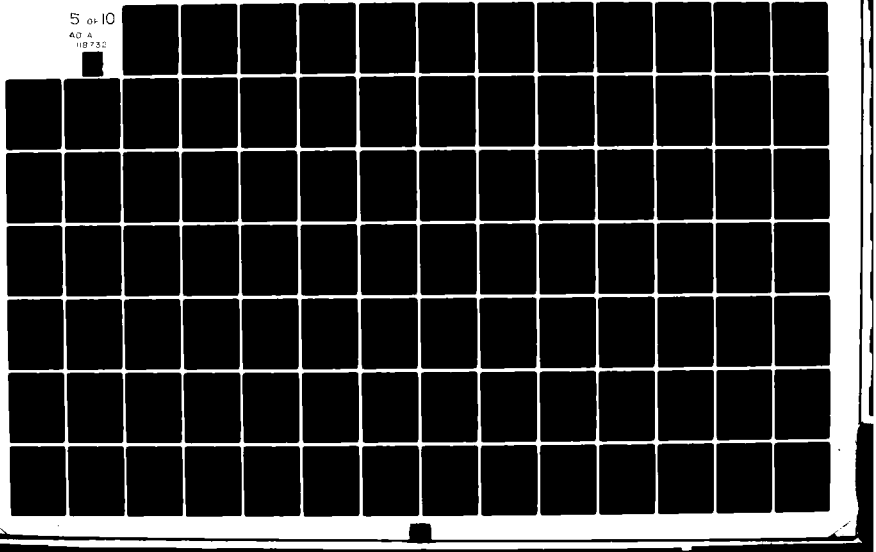
Under RSX-11M, programs do not always abort after an error has occurred. Therefore, INSINI sets the 'continuation' bit in DEC's FORTRAN OBJECT TIME SYSTEM (OTS) error facility to 'false' for all errors that will not halt program execution. (see RSX-11M FORTRAN IV (F4P) user's guide, under 'ERRSET'.)

FILES:

CM - communications
HS - history
'INSTRU.DAT' the instrumentation output (debug info.) file

AD-A118 732 PAR TECHNOLOGY CORP NEW HARTFORD NY F/G 9/2
ON-LINE PATTERN ANALYSIS AND RECOGNITION SYSTEM, OLPARS VI. SOF--ETC(U)
JUN 82 S E HAEHN, D MORRIS
UNCLASSIFIED PAR-82-20 NL

5 of 10
AD A
118732



REFERENCED BY:

All OLPARS commands

SUBPROGRAMS REFERENCED:

CLOSFX, CMGOTH, EQUALS, ERRSET, INSPGM, OEXIT, OPENFX, OPENS, TRMPUT

DIAGNOSTICS:

Will notify programmer when it can't open any of the files mentioned above.

HISTORY:

designed by Steven Haehn Dec. 4, 1978

programmed by Steven Haehn Apr. 20, 1979

modified by Donna Morris Nov. 28, 1979

(I removed the user interaction section which asked the user if instrumentation was to be 'ON', and replaced it with code that reads the instrumentation flag from the CM file).

OLPARS Program Specifications
INSINT

PROGRAM NAME: INSINT

CATEGORY: Programmer's Aid (system dependent)

PROGRAM DESCRIPTION:

INSINT (instrumentation INTEGER variable dump) prints out the given variable name and the value of the integer number stored there.

PROGRAM USAGE:

CALL INSINT(NAME,VALUE)

INPUT ARGUMENTS:

NAME - LOGICAL*1 ARRAY(VRNLEN+1); HOLLERITH string containing name of the variable to be printed.

VALUE - INTEGER; integer number to be printed.

REFERENCED BY:

All OLPARS system independent programs

SUBPROGRAMS REFERENCED:

FILPUT, OEXIT, TRMPUT

SEE ALSO:

INSCHR, INSDBL, INSFLT, INSLOG

HISTORY:

designed by Steven Haehn Mar. 19, 1979

programmed by Steven Haehn Apr. 20, 1979

PROGRAM NAME: INSLOG

CATEGORY: Programmer's Aid (system dependent)

PROGRAM DESCRIPTION:

INSLOG (instrumentation logical variable dump) prints out the given variable name and the value of the given variable as "true" or "false".

PROGRAM USAGE:

CALL INSLOG(NAME,VALUE)

INPUT ARGUMENTS:

NAME - LOGICAL*1 ARRAY(VRNLEN+1); HOLLERITH field containing the name of the variable to be printed. (VRNLEN = variable name length, a PARAMETER)

VALUE - LOGICAL; logical value to be printed.

REFERENCED BY:

Any OLPARS system independent programs.

SUBPROGRAMS REFERENCED:

FILPUT, OEXIT, TRMPUT

SEE ALSO:

INSCHR, INSDBL, INSFLT, INSINT

HISTORY:

designed by Steven Haehn Mar. 19, 1979

programmed by Steven Haehn Apr. 20, 1979

OLPARS Program Specifications
INSPGM

PROGRAM NAME: INSPGM

CATEGORY: Programmer's aid (system dependent)

PROGRAM DESCRIPTION:

INSPGM increments the indentation level of the debug printout and prints out the name (preferably the subroutine name of the routine entered before INSPGM) given to it. INSPGM also retrieves the history record, associated with the given NAME, from the user's History file and updates it.

PROGRAM USAGE:

CALL INSPGM(NAME,PGMTYP)

INPUT ARGUMENTS:

NAME - CHARACTER; Hollerith string, terminated by a null (0) character, containing the name of the subroutine previously entered.

PGMTYP - INTEGER; Type of program that is using the instrumentation package. Zero (0), means calling program is of type 'MAIN'. One (1), means program is of type 'SUBPROGRAM'.

FILES:

HS - instrumentation history

REFERENCED BY:

All OLPARS system independent programs

SUBPROGRAMS REFERENCED:

GETHST,FILPUT

DIAGNOSTICS:

An error message will appear at the user's terminal indicating that the programmer has called INSPGM with a name that has too many characters.

SEE ALSO:

INSRET

HISTORY:

designed by Steven Haehn Feb. 1, 1979

programmed by Steven Haehn Feb. 1, 1979

PROGRAM NAME: INSRET

CATEGORY: Programmer's aid (system dependent)

PROGRAM DESCRIPTION:

INSRET writes out the history record (indirectly) of the current instrumentation level. It also decrements the indentation level of the debug printout.

PROGRAM USAGE:

CALL INSRET

FILES:

HS - history
'INSTRU.DAT' the instrumentation (debug) output file

REFERENCED BY:

All OLPARS system independent programs

SUBPROGRAMS REFERENCED:

FILPUT, OEXIT, PUTHST, TRMPUT

DIAGNOSTICS:

Tells user that the instrumentation package is not initialized. Warns user of instrumentation underflow (when INSRET is called more than INSPGM).

SEE ALSO:

INSPGM

HISTORY:

designed by Steven Haehn Feb. 20, 1979

programmed by Steven Haehn Apr. 19, 1979

PROGRAM NAME: INSSET

CATEGORY: Programmer's aid (system dependent)

PROGRAM DESCRIPTION:

INSSET is used to force the instrumentation package to a desired state (i.e., either 'ON' or 'OFF'). The previous state of the package is returned to the calling program. This allows the program to reset the old state of the instrumentation package.

PROGRAM USAGE:

```
LOGICAL INSSET
    .
    .
    .

STATE = INSSET(NWSTAT,THRSH)
```

INPUT ARGUMENTS:

NWSTAT - LOGICAL; New state of the instrumentation package (false=OFF, true=ON)

THRSH - INTEGER; threshold for debug printout to start at (and of times to print info. out, once the threshold is exceeded).

OUTPUT ARGUMENTS:

INSSET- LOGICAL; value of the previous state of the instrumentation package.

ALGORITHM/NOTES:

This routine should not be permanently left in any of the OLPARS programs. It's usage is of a temporary nature only, forcing debug printout where normally it would not be seen.

REFERENCED BY:

Any OLPARS system independent program.

SUBPROGRAMS REFERENCED:

OEXIT, OPENFX, TRMPUT

OLPARS Program Specifications
INSSET

HISTORY:

designed by Steven Haehn Mar. 20, 1979

programmed by Steven Haehn Apr. 20, 1979

PROGRAM NAME: INTENSIFY

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

INTENSIFY highlights a class or classes currently displayed by drawing a solid outline around the given class distributions. It should be noted that, unless classes are well separated, if more than two classes are concurrently intensified, then the display itself becomes cluttered with lines and it is hard to observe the distributions. This routine is applicable to one-space micro displays only. (If used on a one-space macro plot, a micro-plot will be generated on the terminal.)

PROGRAM USAGE:

User types 'INTENSIFY'.

USER INTERACTION:

User specifies the classes (via class display symbol) to be intensified. (OBTCLS)

ALGORITHM / NOTES:

. Make sure display file contains only a one-space display plots.

REPEAT

Ask user for classes to be intensified (OBTCLS)

Mark classes to be intensified.

Put up display with the chosen classes (MICMAC)

Ask user if he wants to intensify again

UNTIL (intensifing is complete)

Put up option list only if 'erase' was called(MENU)

END

OLPARS Program Specifications
INTENSIFY

FILES:

CM - communications file
DI - display information
DV - display value
PV - projection vector
S1 - scratch 1
OLPARS option file
Instrumentation file

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLOSFX, CMGOTH, DIGHDI, DIPENT, EQUALA, ERASE, INSINI, INSINT, INSRET, MENU, MICMAC, OBTCLS, OEXIT, OPENFX, TRMPUT

SEE ALSO:

SELECT, CDISPLAY

HISTORY:

designed by Steve Haehn October 3, 1978

programmed by Mark Maginn October 15, 1980

PROGRAM NAME: INVERT

CATEGORY: Numerical Computation Algorithm

PROGRAM DESCRIPTION:

This routine calculates the inverse of a symetric square matrix A, which is stored as a vector containing the lower triangular portion of the matrix. If a linearly dependent row is encountered, the program ends and the row at which this occurred is returned along with the original matrix, unchanged. At this point, the DITHER routine may be called to slightly alter the pivot element in the linearly dependent row. As an added dividend, the determinate of the matrix is returned as the value of the function.

PROGRAM USAGE:

```
REAL FUNCTION INVERT

DET=INVERT(A,NDIM,THRESH,ERROW)
```

INPUT ARGUMENTS:

A - REAL array ((Ndim*(Ndim+1))/2); the lower triangular portion of the symmetric square matrix to be inverted, stored as a vector

NDIM - INTEGER; the order of matrix A

THRESH - REAL; the minimum allowable value for the pivot element

OUTPUT ARGUMENTS:

A - REAL array ((Ndim*(Ndim+1))/2); the lower triangular portion of the inverse matrix, stored as a vector

ERROW - INTEGER; the row number of the linearly dependent row found, 0 if there was none

INVERT - REAL; if ERROW is 0, then invert is the value of the determinant of the matrix, otherwise it is set equal to 0.

OLPARS Program Specifications
INVERT

ALGORITHM/NOTES:

Expand the matrix and place it into a working array.

DO I=1,Ndim

Locate the element of largest magnitude in row I.

IF (pivot element.lt.threshold value) THEN

a linear dependency occurs in this row, and thus no inverse matrix exists. Goto BYEBYE.

ENDIF

Save the pivot column.

Move the Ith column to the pivot column.

Compute the partial column inverse.

Adjust the Ith row and save it.

Adjust the remaining matrix with the row multiplier.

ENDDO

Reorder the matrix columns.

Collapse the inverse matrix into a vector.

RETURN

FILES:

instrumentation file

SUBPROGRAMS REFERENCED:

INSINT, INSPGM, INSRET

REFERENCED BY:

L2FSHP, NMVBSU, S2FSHP

HISTORY:

designed by Jill King September 29, 1980

programmed by Jill King September 29, 1980

PROGRAM NAME: ITREAL

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

ITREAL moves integer words to real elements. Transfer of words continues until the number of integers to transfer is exhausted or until the first null character has been transferred, depending on whether or not the stop transferring flag has been set. The actual number of integers transferred is returned.

PROGRAM USAGE:

```
CALL ITREAL(INTGRS,REALS,NUMBER,NXFERD,STOP)
```

INPUT ARGUMENTS:

INTGRS - INTEGER array (NUMBER + 1); integers to transfer

REALS - REAL array (NUMBER + 1); output buffer for integers

NUMBER - INTEGER; number of integers to transfer

STOP - LOGICAL; stop transferring flag. if true, stop transferring integers after the first null character has been transferred. if false, continue moving integers until 'NUMBER' is exhausted.

OUTPUT ARGUMENTS:

NXFERD - INTEGER; the number of integers actually transferred

REFERENCED BY:

Level I Routines

SEE ALSO:

PACKB,PACKW

HISTORY:

designed by Donna Morris March 9,1979

programmed by Donna Morris May 9,1979

PROGRAM NAME: KILLND

CATEGORY: Utility Subroutine

PROGRAM DESCRIPTION:

KILLND deletes logic nodes below a give logic node (in Logic Information file) and any decision logic regions (Logic Value file entries) associated with logic nodes. The decision logic for the given node is deleted, if the kill-current-decision-logic flag is set to 'true'.
The data vectors that lie at the logic nodes (Tree Vector file entries) below the given logic node are sent back to the given logic node (i.e., the temporary logic elements found in the data vectors of the classes present at the given logic node are set to the value of the given logic node).

(Example use: when user does not finish creating decision logic for a logic node, the altered data set and uncompleted logic node must be cleaned up.)

PROGRAM USAGE:

```
CALL KILLND(FDLI,FDLV,FDTI,FDTV,CRNTLG,KILLDL)
```

INPUT ARGUMENTS:

```
  FDLI - INTEGER; Logic Information file descriptor
  FDLV - INTEGER; Logic Value file descriptor
  FDTI - INTEGER; Tree Information file descriptor
  FDTV - INTEGER; Tree Vector file descriptor
CRNTLG - INTEGER; current logic node
KILLDL - LOGICAL; kill-decision-logic-at-current-
                  logic-node flag
```

FILES:

```
LI - logic information
LV - logic value
TI - tree information
TV - tree vector
```

REFERENCED BY:

All Logic Design Commands

SUBPROGRAMS REFERENCED:

DLREGN, ELIFRE, FXLTMP, GTRGNI, INSINT, INSPGM, INSRET, LIGCOP, LIGLOG, LIGSTR, LIPLOG, LIPSTR

HISTORY:

designed by Steven Haehn Mar. 5, 1981

programmed by Steven Haehn Apr. 1, 1981

OLPARS Program Specifications
L1ASDG

PROGRAM NAME: L1ASDG

CATEGORY: Logic Design Command

PROGRAM DESCRIPTION:

L1ASDG projects a data set onto the Fisher direction associated with two algebraically-assigned groupings of data classes at an incomplete logic node. It is the one-space analogue of L2ASDG and is the same except for obvious one-space modifications.

PROGRAM USAGE:

User types in 'L1ASDG'.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Make sure the current data set is not in excess measurement mode (CHKEXS).

Get set up for the projection of the set of vectors at a logic node (SETUP).

Initialize DI, DV, PV and file S1 headers (INITD).

Call UILASD to

Retrieve or compute the class mean vectors (TIGMN, LOGMN).

Divide the classes at the specified logic node into two groups, determined by the euclidean distance between the mean vectors. First, all class mean vectors are compared, and the two classes with the largest euclidean distance between the means are saved. Then, one at a time, each class mean vector is compared with each of the two saved class mean vectors. If the mean is closer to the mean of saved class 1, then the class belongs to the first group. Otherwise, the class belongs to the second group (DISTEU).

Display the two groups of classes (TRMPUT).

```
            IF (the user is not satisfied) THEN

               Request group modifications (SELCLS).

               Redisplay modified groups (TRMPUT).

            ENDIF

            Ask the user if measurements are to be eliminated
            (ELIMEA).

            Ask the user if scatter or covariance matrices are
            to be used in the computations (PROMPT).

         Compute optimal discriminant plane (DCRIM).

         Store them in the PV file (PVPENT, PVPHDR).

         Project the vectors (LNPROJ).

         Create the display (MICMAC).

         Put up option list at user's terminal (MENU).

         END
```

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

```
CHKEXS, CLOSFX, CMGCDS, DCRIM,  DIPHDI, DIPNAM
ERASE,  INITD,  INSINI, INSINT, INSRET, LNPROJ
MENU,   MICMAC, MODDFS, OEXIT,  OPENFX, PVPENT
PVPHDR, SETUP,  TRMPUT, UILASD
```

HISTORY:

designed by David Birnbaum September 13, 1978

programmed by Jill King May 6, 1981

OLPARS Program Specifications
L1CRDV

PROGRAM NAME: L1CRDV

CATEGORY: Logic Design Command

PROGRAM DESCRIPTION:

L1CRDV is the one-space coordinate projection routine for use in logic design. It is the same as L2CRDV except for obvious one-space modifications.

PROGRAM USAGE:

User types in 'L1CRDV'.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Get set up for the projection of the set of vectors at a logic node (SETUP).

Ask user to select one coordinate on which to project the data set.

Initialize DI, DV, PV and S1 files (INITD).

Enter the coordinate in DI file (DIPHDI).

For each vector, store the projected coordinate in the DV file and set up DI file entries (LCPROJ).

Create one-space plot (MICMAC).

Put up option list at user's terminal (MENU).

END

FILES:

CM - communications
DI - display information
DV - display value
PV - projection vector
S1 - scratch 1
TL - tree list
LL - logic list
HS - history
TI - tree information
TV - tree vector
OLPARS option file
Instrumentation
LI - logic information
LV - logic value

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLOSFX, CLOSTR, CMGCDS, CMGOTH, DIPHDI, DIPNAM, ERASE,
INITD, INSINI, INSRET, LCPROJ, MENU, MICMAC, OEXIT,
OPENFX, PROMPT, SETUP, TRMGET, TRMPUT

SEE ALSO:

L2CRDV

HISTORY:

designed by David BirnBaum September 28, 1978

programmed by Steven Haehn Dec. 9, 1980

OLPARS Program Specifications
L1EIGV

PROGRAM NAME: L1EIGV

CATEGORY: Logic Design Command

PROGRAM DESCRIPTION:

L1EIGV projects the set of vectors at an incomplete logic node onto an eigenvalue of that data set that is selected by the user. It is the same as L2EIGV except for obvious one-space modifications.

PROGRAM USAGE:

User types in 'L1EIGV'.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Get set up for the projection of the set of vectors at a logic node (SETUP).

Initialize DI, DV, PV and S1 headers (INITD).

Measurements to be ignored? (ELIMEA).

Compute eigenvalues and eigenvectors and insert into PV file (EIGLPV).

Printout of eigenvectors desired (EIGPRT)?

Ask user to select one eigenvalue for use in the projection (SLTEIG).

Project the data set (LNPROJ).

Create display (MICMAC).

Put up option list at user's terminal (MENU).

END

FILES:

PV - projection vector

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

ERASE, SETOPT, SETUP, INITD, ELIMEA,
EIGLPV, EIGPRT, SLTEIG, LNPROJ, MICMAC,
MENU, PVPHDR, PVPENT

HISTORY:

designed by David A. Birdbaum September 13, ,1978

programmed by David J. Tipton December 22, 1981

PROGRAM NAME: L2ASDG

CATEGORY: Logic Design Command

PROGRAM DESCRIPTION:

L2ASDG projects a data set on to the optimal discriminant plane associated with two algebraically-assigned groupings of data classes at an incomplete logic node. The calculations proceed as in S2ASDG.

PROGRAM USAGE:

User types in 'L2ASDG'.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Make sure the current data set is not in excess measurement mode (CHKEXS).

Get set up for the projection of the set of vectors at a logic node (SETUP).

Initialize DI, DV, PV and file S1 headers (INITD).

Call UILASD to

Retrieve or compute the class mean vectors (TIGMN, LOGMN).

Divide the classes at the specified logic node into two groups, determined by the euclidean distance between the mean vectors. First, all class mean vectors are compared, and the two classes with the largest euclidean distance between the means are saved. Then, one at a time, each class mean vector is compared with each of the two saved class mean vectors. If the mean is closer to the mean of saved class 1, then the class belongs to the first group. Otherwise, the class belongs to the second group (DISTEU).

Display the two groups of classes (TRMPUT).

```
        IF (the user is not satisfied) THEN

           Request group modifications (SELCLS).

           Redisplay modified groups (TRMPUT).

        ENDIF

        Ask the user if measurements are to be eliminated
        (ELIMEA).

        Ask the user if scatter or covariance matrices are
        to be used in the computations (PROMPT).

     Compute optimal discriminant plane (DCRIM).

     Store them in the PV file (PVPENT, PVPHDR).

     Project the vectors (LNPROJ).

     Create the display (CLSCAT).

     Put up option list at user's terminal (MENU).

     END
```

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CHKEXS, CLOSFX, CLSCAT, CMGCDS, DCRIM, DIPHDI, DIPNAM, ERASE, INITD, INSINI, INSINT, INSRET, LNPROJ, MENU, MODDFS, OEXIT, OPENFX, PVPENT, PVPHDR, SETUP, TRMPUT, UILASD

SEE ALSO:

S2ASDG

HISTORY:

designed by David Birnbaum September 13, 1978

programmed by Jill King May 5, 1981

OLPARS Program Specifications
L2CRDV

PROGRAM NAME: L2CRDV

CATEGORY: Logic Design Command

PROGRAM DESCRIPTION:

L2CRDV is the two-space coordinate projection routine for use in logic design.

PROGRAM USAGE:

User types in 'L2CRDV'.

USER INTERACTION:

Prompts for coordinates to project on.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Get set up for the projection of the set of vectors at a logic node (SETUP).

Initialize DI, DV, PV and S1 files (INITD).

Ask user to select two coordinates to project on.

Enter these coordinates in DI file (DIPHDI).

For each vector, store those coordinates in the DV file and set up DI file entries (LCPROJ).

Create two-space plot (CLSCAT).

Put up option list at user's terminal (MENU).

END

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLOSFX, CLOSTR, CLSCAT, CMGCDS, CMGOTH, DIPHDI
DIPNAM, ERASE, INITD, INSINI, INSRET, LCPROJ
MENU, OEXIT, OPENFX, OPENTR, PROMPT, PVPENT
PVPHDR, SETUP, TIGCOP

HISTORY:

designed by David A. Birnbaum September 28, 1978

programmed by David J. Tipton February 18, 1981

OLPARS Program Specifications
L2EIGV

PROGRAM NAME: L2EIGV

CATEGORY: Logic Design Command

PROGRAM DESCRIPTION:

L2EIGV projects the set of vectors at an incomplete logic node onto two eigenvalues of that data set that are selected by the user.

PROGRAM USAGE:

User types in 'L2EIGV'.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Get set up for the projection of the set of vectors at a logic node (SETUP).

Initialize DI, DV, PV and S1 headers (INITD).

Measurements to be ignored? (ELIMEA).

Compute eigenvalues and eigenvectors and insert into PV file (EIGLPV)

Printout of eigenvectors desired (EIGPRT)?

Ask user to select two eigenvalues for use in the projection (SLTEIG).

Project the data set (LNPROJ).

Create display (CLSCAT).

Put up option list at user's terminal (MENU).

END

FILES:

DI - display information
DV - display vector
PV - projection vector
TI - tree information
TV - tree vector

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLSCAT, EIGLPV, EIGPRT, ELIMEA, ERASE, INITD
LNPROJ, MENU, PVPENT, PVPHDR, SETUP, SLTEIG

SEE ALSO:

S2EIGV

HISTORY:

designed by David A. Birnaum September 13, 1978

programmed by David J. Tipton Januray 27, 1981

PROGRAM NAME: L2FSHP

CATEGORY: Logic Design Command

PROGRAM DESCRIPTION:

L2FSHP projects the vectors at an incomplete logic node on two Fisher directions, which correspond to two pairs of data classes at the node.

PROGRAM USAGE:

User types in 'L2FSHP'.

USER INTERACTION:

The user is prompted for:

1. An incomplete logic node number (done in SETUP).
2. The first class pair.
3. The second class pair.
4. Scatter or covariance matrix options.
5. Whether or not to eliminate some measurements in the computation, and the number of these measurements.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Call UIL2FS to

Get set up for the projection of vectors at an incomplete logic node (SETUP).

Initialize DI, DV, PV, and S1 file headers (INITD).

Call SLPAIR to

Display list of classes that lie at the logic node (these were retrieved in SETUP).

Request first pair of classes (SELCLS).

Request second pair of classes (SELCLS).

Scatter or covariance option (PROMPT).

Eliminate any measurements (ELIMEA).

Compute Fisher discriminant for first pair of classes (DCRIM).

Compute Fisher discriminant for second pair of classes (DCRIM).

Orthogonalize the two discriminants.

Store the discriminants in the PV file making sure pointers to the projection vectors ar correct (PVPENT).

Project set of classes (LNPROJ).

Modify the display flag symbol (MODDFS).

Create two-space display (CLSCAT).

Put up option list at user's terminal (MENU).

END

FILES:

CM - communication file
DI - display information file
DV - display vector file
HS - history file
LI - logic information file
LV - logic vector file
PV - projection vector file
TI - tree information file
TV - tree vector file
instrumentation file
option file

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CHKEXS, CLSCAT, CMGCDS, DCRIM, DIPHDI, DIPNAM, ERASE
INSINI, INSINT, INSLOG, INSRET, LNPROJ, MENU, MODDFS
OEXIT, OPENFX, PVPENT, PVPHDR, TRMPUT, UIL2FS, $SQRT

HISTORY:

designed by David Birnbaum September 14, 1978

programmed by Jill King December 11, 1980

PROGRAM NAME: LCPROJ

CATEGORY: Display Files Access Routine

PROGRAM DESCRIPTION:

If ALLVEC = TRUE, LCPROJ stores in the DV file the selected coordinate(s) for each vector from the set of classes whose entry table slot numbers are passed in SLOTS. If ALLVEC = FALSE, LCPROJ stores only those vectors that lie at the logic node number passed in NODNUM.

PROGRAM USAGE:

```
CALL LCPROJ(FIDTI,FIDTV,FIDDI,FIDDV,ENTABL,NCLASS,
SLOTS,ALLVEC,NODNUM,COORDS)
```

INPUT ARGUMENTS:

FIDTI - INTEGER; the file descriptor of the TI file

FIDTV - INTEGER; the file descriptor of the TV file

FIDDI - INTEGER; the file descriptor of the DI file

FIDDV - INTEGER; the file descriptor of the DV file

ENTABL - INTEGER; array (100); the entry table

NCLASS - INTEGER; the number of classes

SLOTS - INTEGER; array (NCLASS); the list of entry table slot numbers of the classes at the logic node

ALLVEC - LOGICAL; TRUE means use all vectors; FALSE means use only those vectors that lie at the logic node

NODNUM - INTEGER; the logic node number

COORDS - INTEGER array (2); the coordinates (measurement numbers); if COORDS (2) = 0 then this is a one-space projection

ALGORITHM / NOTES:

```
DO WHILE (there are more classes to be projected)

    Get class counters and pointers (TIGCOP) and
    class name (TIGNAM)

    IF (all vectors of the classes are to be used) THEN

        IF(not in excess measurement mode)THEN

            Get mean vector of the class (TIGMN).

        ENDIF

        DO WHILE (there are more vectors in the class)

            Get a vector and store selected coordinates
            in DV file.

            IF(we are in excess measurement mode)THEN

                Update the mean vector computation.

            ENDIF

            Update x(min), x(max), (y(min), y(max)).

        ENDDO

        Store selected coordinate of mean vector
        in DV file.

        Create a DI file entry for the class.

    ELSE

        DO WHILE (there are more vectors in the class
        that lie at the specified logic node)

            Get a vector and store selected coordinates
            in DV file.

            Update mean vector computation and the vector
            count.

            Update x(min), x(max), (y(min), y(max)).

        ENDDO
```

Calculate the new mean vector.

Store selected coordinates of mean vector in DV file.

Create a DI file entry for the class.

Update the total vector count

ENDIF

ENDDO

Store x(min), x(max), (y(min), y(max)) values in DI file as original and current (DIPMAX).

Store total number of vectors in DI file header (DIPHDI).

Update the next available entry in the DV file header.

RETURN

FILES:

TI - tree information
TV - tree vector
DI - display information
DV - display value

REFERENCED BY:

L1CRDV, L2CRDV, CRPROJ

SUBPROGRAMS REFERENCED:

DIPENT, DIPHDI, DIPMAX, DVPHDR, DVPVEC, INSINT, INSPGM, INSRET, TIGCOP, TIGHDR, TIGMN, TIGNAM, TVGVEC, TVPVEC

HISTORY:

designed by Dave Birnbaum September 26, 1978

programmed by Donna Morris October 1, 1980

OLPARS Program Specifications
LENGA

PROGRAM NAME: LENGA

CATEGORY: Utility Subroutine

PROGRAM DESCRIPTION:

LENGA determines the length of a character string. The maximum size string allowed is determined by 'MAXLIN'. The string must have and end-of-string symbol (0) appended to its end. If no eos symbol is found, LENGTH will return 'MAXLIN-1'.

PROGRAM USAGE:

```
INTEGER LENGA
     .
     .
     .

N=LENGA(STRING)
```

INPUT ARGUMENTS:

STRING - INTEGER array (MAXLIN); the character string for which to determine the length

OUTPUT ARGUMENTS:

LENGA - INTEGER; the length of the character string

REFERENCED BY:

SETDS

HISTORY:

designed by Steve Haehn September 7,1979

programmed by Steve Haehn September 7,1979

PROGRAM NAME: LENGTH

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

LENGTH determines the length of a character string. The maximum size string allowed is determined by 'MAXLIN'. The string must have and end-of-string symbol (0) appended to its end. If no eos symbol is found, LENGTH will return 'MAXLIN-1'.

PROGRAM USAGE:

```
INTEGER LENGTH
        .
        .
        .
N = LENGTH (STRING)
```

INPUT ARGUMENTS:

STRING - LOGICAL *1 ARRAY; the character string for which to determine the length

OUTPUT ARGUMENTS:

LENGTH - INTEGER; the length of the character string

REFERENCED BY:

GEN

HISTORY:

designed by Steve Haehn January 1,1979

programmed by Steve Haehn January 1,1979

PROGRAM NAME: LESS

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

LESS determines the lexigraphical order of two given character strings. When the first string is 'less than' the second string, LESS returns a 'true' value. Otherwise, LESS returns 'false'. The 'case' of the alphabetical characters may or may not be used in determining the lex. order.

PROGRAM USAGE:

```
LOGICAL LESS
     .
     .
     .
IF(LESS(STR1,LEN1,STR2,LEN2,CASE))
```

INPUT ARGUMENTS:

STR1 - LOGICAL *1 ARRAY(MAXLIN); string 1 in 'less' comparison

LEN1 - INTEGER: length of the first string

STR2 - LOGICAL *1 ARRAY(MAXLIN); string 2 in 'less' comparison

LEN2 - INTEGER; length of the second string

CASE - LOGICAL; determines whether or not the case of the alphabetic characters is important in the string comparison. (true - case important
false - case unimportant)

OUTPUT ARGUMENTS:

LESS - LOGICAL; 'TRUE' if STR1 is less than STR2
'FALSE' if STR1 is greater than or equal to STR2.

REFERENCED BY:

MAKOPT

SUBPROGRAMS REFERENCED:

LENGTH, VALUES

SEE ALSO:

EQUALS

HISTORY:

designed by Steven Haehn Aug. 17, 1979

programmed by Steven Haehn Aug. 24, 1979

OLPARS Program Specifications
LGLNOD

PROGRAM NAME: LGLNOD

CATEGORY: Utility Subroutine

PROGRAM DESCRIPTION:

Given the character string contained in the variable NODNAM, LGLNOD determines if NODNAM is a legal OLPARS node name. Legal node names may not start with a blank or have any commas in them. Other printable ASCII characters are OK.

PROGRAM USAGE:

```
LOGICAL LGLNOD
   .
   .
IF(LGLNOD(NODNAM))
```

INPUT ARGUMENTS:

NODNAM - INTEGER ARRAY(NODLEN); the node name

OUTPUT ARGUMENTS:

LGLNOD - LOGICAL; true if NODNAM is a legal OLPARS node name; otherwise false.

ALGORITHM / NOTES:

```
IF (the node name contains any digit, letter, or
    other printable ASCII characters) THEN

    IF (the display symbol is not a blank) THEN

        IF(there are no commas in the node name) THEN

           NODNAM is a valid OLPARS node name
           (LGLNOD = true).

        ENDIF
    ENDIF
ELSE

    NODNAM is not a valid OLPARS node name
    (LGLNOD = false).

ENDIF
RETURN
```

REFERENCED BY:

APPEND

SUBPROGRAMS REFERENCED:

EQUALA, IDCHK, INSLOG, INSPGM, INSRET

HISTORY:

designed by Steven Haehn June 1, 1978

programmed by Steven Haehn April 8, 1980

PROGRAM NAME: LGLTRE

CATEGORY: Utility Subroutine

PROGRAM DESCRIPTION:

Given the character string contained in the variable TRENAM, LGLTRE determines if TRENAM is a legal OLPARS tree name.

PROGRAM USAGE:

```
LOGICAL LGLTRE
   .
   .
IF(LGLTRE(TRENAM))
```

INPUT ARGUMENTS:

TRENAM - INTEGER array (TRELEN + 1); the tree name

ALGORITHM / NOTES:

```
IF (the "TRENAM"'s first character is alphabetic
    and the rest of the name is alphanumeric) THEN

    "TRENAM" is a valid OLPARS tree name (LGLTRE = true).

ELSE

    "TRENAM" is not a valid OLPARS tree name
     (LGLTRE = false).

ENDIF

RETURN
```

REFERENCED BY:

APPEND

SUBPROGRAMS REFERENCED:

IDCHK

HISTORY:

designed by Steven Haehn June 1, 1978

programmed by Steven Haehn April 8, 1980

PROGRAM NAME: LIGCAP

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LIGCAP obtains from the Logic Information (LI) file header the counts and pointers of the logic tree.

PROGRAM USAGE:

```
CALL LIGCAP(FIDLI,NDIM,NLOWND,NXTLND,NOPENT,NLNOD,
            CRNTND,ALIAS,NLVELM,NILN)
```

INPUT ARGUMENTS:

FIDLI - INTEGER; file descriptor of LI file to be accessed

OUTPUT ARGUMENTS:

NDIM - INTEGER; dimensionality of vectors in design data set

NLOWND - INTEGER; number of lowest nodes in the design data set

NXTLND - INTEGER; next available logic node entry in LI file

NOPENT - INTEGER; the 'next open entry on end of file' element in the LI file

NLNOD - INTEGER; number of logic nodes currently in logic tree

CRNTND - INTEGER; the current logic node (for group 1 and 2-space logic only)

ALIAS - INTEGER; the reassociated class name flag (for evaluation) determines whether or not the class name to be used is the original class name or its alias

NLVELM - INTEGER; number of elements in an LV file entry

NILN - INTEGER; number of incomplete logic nodes

FILES:

LI - logic information

SUBPROGRAMS REFERENCED:

FGET, INSPGM, INSRET

SEE ALSO:

LIPCAP

HISTORY:

designed by Steve Haehn September 13, 1978

programmed by Donna Morris August 12, 1980

OLPARS Program Specifications
LIGCLP

PROGRAM NAME: LIGCLP

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LIGCLP returns to the calling program the bit map representing the classes present at a logic node in a logic tree.

PROGRAM USAGE:

CALL LIGCLP(FDLI,ENTNO,CPRSNT,NCPRS)

INPUT ARGUMENTS:

FDLI - INTEGER; file descriptor of the LI file being accessed

ENTNO - INTEGER; entry number of the logic entry that contains the classes-present bit map

OUTPUT ARGUMENTS:

CPRSNT - REAL array (BMPSIZ); bit map designating the number of classes present (residing) at a logic node

NCPRS - INTEGER; the number of classes present at a logic node

FILES:

LI - logic information

REFERENCED BY:

GCLIST

SUBPROGRAMS REFERENCED:

FGET, INSINT, INSPGM, INSRET, OEXIT, TRMPUT

SEE ALSO:

LIPCLP

HISTORY:

designed by Steven Haehn September 14, 1978

programmed by Steven Haehn Nov. 25, 1980

PROGRAM NAME: LIGCNM

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LIGCNM obtains the class names of the lowest nodes of a design data tree from the logic information (LI) file header of a logic tree. It will also return the number of class names of these lowest nodes found in the design data set.

PROGRAM USAGE:

CALL LIGCNM(FIDLI,CLSNAM,NCLAS)

INPUT ARGUMENTS:

FIDLI - INTEGER; file descriptor of the LI file being accessed

OUTPUT ARGUMENTS:

CLSNAM - INTEGER array (4,50); storage area for class names to be read from LI file header

NCLAS - INTEGER; the number of class names found in the LI file header

FILES:

LI - logic information

SUBPROGRAMS REFERENCED:

FGET, INSPGM, INSRET

SEE ALSO:

LIPCNM

HISTORY:

designed by Steve Haehn September 12, 1978

programmed by Donna Morris August 11, 1980

PROGRAM NAME: LIGCOP

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LIGCOP retrieves from an LI file header a count or pointer data value. The count or pointer to be referenced is specified by a code. (The code information can be found in the program description section of LIPCOP.)

PROGRAM USAGE:

```
INTEGER LIGCOP
     .
     .
NUMRET = LIGCOP(FDLI,CPCODE)
```

INPUT ARGUMENTS:

FDLI - INTEGER; file descriptor of LI file being accessed

CPCODE - INTEGER; count or pointer code representing data to be accessed

OUTPUT ARGUMENTS:

LIGCOP - INTEGER; count or pointer retrieval from the LI file header

FILES:

LI - logic information

SUBPROGRAMS REFERENCED:

FGET, INSINT, INSPGM, INSRET, OEXIT, TRMPUT

SEE ALSO:

LIPCOP

OLPARS Program Specifications
LIGCOP

HISTORY:

designed by Steven Haehn September 18, 1978

programmed by Mark Maginn July 17, 1980

PROGRAM NAME: LIGDCN

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LIGDCN gets the original design data set class name and the reassociated class name from a logical information file entry.

PROGRAM USAGE:

CALL LIGDCN (FID, ENTNO, ODCN, RCN)

INPUT ARGUMENTS:

FID - INTEGER; LI file descriptor

ENTNO - INTEGER; LI file entry number (or logic node number)

OUTPUT ARGUMENTS:

ODCN - INTEGER (NODLEN) array; the original design data set class name for this logic node (ENTNO)

RCN - INTEGER (NODLEN) array; the reassociated class name for this logic node (ENTNO)

SUBPROGRAMS REFERENCED:

FGET

HISTORY:

designed by David Tipton September 20, 1978

programed by David Tipton May 7, 1981

OLPARS Program Specifications
LIGDSN

PROGRAM NAME: LIGDSN

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LIGDSN obtains the design data set name from the logic information (LI) file header of a user's logic tree.

PROGRAM USAGE:

```
CALL LIGDSN(FIDLI,DSNAME)  -
```

INPUT ARGUMENTS:

FIDLI - INTEGER; file descriptor of the LI file being accessed

OUTPUT ARGUMENTS:

DSNAME - INTEGER array(12); name of the logic tree's design data set, which is a treename, nodename pair

FILES:

LI - logic information

SUBPROGRAMS REFERENCED:

FGET, INSPGM, INSRET

SEE ALSO:

LIPDSN

HISTORY:

designed by Steve Haehn September 12, 1978

programmed by Donna Morris August 11, 1980

PROGRAM NAME: LIGENT

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LIGENT retrieves a logic information (LI) file entry from a logic tree.

PROGRAM USAGE:

```
CALL LIGENT(FIDLI,ENTNO,LOGCD,STRUCT,ODCNAM,RCNAM,
            MLFLAG,NUMCLS,BITMAP)
```

INPUT ARGUMENTS:

FIDLI - INTEGER; file descriptor of LI file to be accessed

ENTNO - INTEGER; entry number of the LI entry to be retrieved

OUTPUT ARGUMENTS:

LOGCD - INTEGER array (3); logic code array:

- o element 1 - logic type code of this node (i.e. logic that produced the nodes immediately under this node)
- o or - entry link (to a deleted node) in free node list
- o element 2 - logic subtype code
- o element 3 - option number of creating routine

STRUCT - INTEGER array (7); storage area for the logic node structure pointers

ODCNAM - INTEGER array(4); original design data set class name for the node entry

RCNAM - INTEGER array(4); reassociated class name (ALIAS) is used when the reassociate name flag (RNF or ALIAS flag, in the LI file header) is turned on

MLFLAG - INTEGER; modified logic flag

NUMCLS - INTEGER; number of classes present (residing) at this node

BITMAP - REAL array (2); bit map representing the classes present (residing) at this node

FILES:

LI - logic information

SUBPROGRAMS REFERENCED:

FGET, INSPGM, INSRET, PACKW.

SEE ALSO:

LIPENT

HISTORY:

designed by Steve Haehn September 13, 1978

programmed by Donna Morris August 11, 1980

PROGRAM NAME: LIGLOG

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LIGLOG retrieves the logic code information and the modified logic flag of a logic entry from the logic information (LI) file. (The type code portion of the logic code information may be a "free" list link pointer.)

PROGRAM USAGE:

```
CALL LIGLOG(FDLI,ENTNO,TYPECD,SUBTYP,OPTNO,MODLF)
```

INPUT ARGUMENTS:

FDLI - INTEGER; file descriptor of the LI file being accessed

ENTNO - INTEGER; entry number of the logic entry desired

OUTPUT ARGUMENTS:

TYPECD - INTEGER; logic type code of this node (i.e., logic that produced the nodes immediately under this node)

or: entry link (to a deleted node) in free node list

SUBTYP - INTEGER; logic subtype code

OPTNO - INTEGER; option number of creating routine

MODLF - INTEGER; modified logic flag

FILES:

LI - logic information

REFERENCED BY:

ELIFRE

OLPARS Program Specifications
LIGLOG

SUBPROGRAMS REFERENCED:

FGET, INSINT, INSPGM, INSRET, OEXIT, TRMPUT

SEE ALSO:

LIPLOG

HISTORY:

designed by Steven Haehn September 19, 1978

programmed by Steven Haehn Dec. 3, 1980

PROGRAM NAME: LIGPRB

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LIGPRB obtains the a priori probabilities, found in the logic information (LI) file header, of the lowest nodes of a design data set. It also returns the number of classes that have a probability number.

PROGRAM USAGE:

CALL LIGPRB(FIDLI,PROB,NCLAS)

INPUT ARGUMENTS:

FIDLI - INTEGER; file descriptor of the LI file being accessed

OUTPUT ARGUMENTS:

PROB - REAL array (50); storage area for the a priori probability numbers found in the LI file header

NCLAS - INTEGER; the number of classes that have a probability number (or the number of probabilities) to be read from the LI file header

FILES:

LI - logic information

SUBPROGRAMS REFERENCED:

FGET, INSPGM, INSRET

SEE ALSO:

LIPPRB

HISTORY:

designed by Steve Haehn September 12, 1978

programmed by Donna Morris August 11, 1980

OLPARS Program Specifications
LIGSTR

PROGRAM NAME: LIGSTR

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LIGSTR gets structural information from an LI file entry. It places them in an array STRUCT as follows:

POSITION IN STRUCT	DATA	ELEMENT IN LI ENTRY
1	Node level	4
2	Number of nodes below this node at next level	5
3	Entry number of parent node	6
4	Entry of first child	7
5	Entry number of next sibling	8
6	Entry number in LV for start of decision logic	9
7	Entry number in LV for reject strategy	10

PROGRAM USAGE:

CALL LIGSTR(FDLI,ENTNO,STRUCT)

INPUT ARGUMENTS:

FDLI - INTEGER; the file descriptor of the LI file
ENTNO - INTEGER; the entry number

OUTPUT ARGUMENTS:

STRUCT - INTEGER array (LSTRSZ); the structural info.

SUBPROGRAMS REFERENCED:

FGET, INSPGM, INSRET, OEXIT, PACKW, TRMPUT

HISTORY:

designed by David BirnBaum September 20, 1978

programmed by Steven Haehn Dec. 4, 1980

OLPARS Program Specifications
LINSEG

PROGRAM NAME: LINSEG

CATEGORY: Terminal I/O (graphic, system dependent)

PROGRAM DESCRIPTION:

LINSEG draws a line having texture ITEX between the points (X1,Y1) and (X2,Y2). These points are actual screen coordinates. (ITEX will probably not be used, but is included to be compatible with the PLTMAP FORTRAN subroutine.)

PROGRAM USAGE:

```
CALL LINSEG(X1,Y1,X2,Y2,ITEX)
```

INPUT ARGUMENTS:

```
  X1 - INTEGER; the X coordinate of the first point
  Y1 - INTEGER; the Y coordinate of the first point
  X2 - INTEGER; the X coordinate of the second point
  Y2 - INTEGER; the Y coordinate of the second point
ITEX - INTEGER; the texture of the line to be drawn
```

HISTORY:

designed by Steve Haehn April 13, 1978

programmed by Steve Haehn April 1, 1981

PROGRAM NAME: LIPCAP

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LIPCAP writes the counts and pointers of a logic tree into the logic information (LI) file header. This program will also calculate the number of elements in an LV file entry if the parameter NLVELM is set to zero (the dimensionality, i.e. NDIM, is used in this calculation).

PROGRAM USAGE:

```
CALL LIPCAP(FIDLI,NDIM,NLOWND,NXTLND,NOPENT,NLNOD,
            CRNTND,ALIAS,NLVELM,NILN)
```

INPUT ARGUMENTS:

FIDLI - INTEGER; file descriptor of LI file to be accessed

NDIM - INTEGER; dimensionality of vectors in design data set

NLOWND - INTEGER; number of lowest nodes in the design data set

NXTLND - INTEGER; next available logic node entry in LI file

NOPENT - INTEGER; the 'next open entry on end of file' element in the LI file

NLNOD - INTEGER; number of logic nodes currently in logic tree

CRNTND - INTEGER; the current logic node (for group 1 and 2-space logic only)

ALIAS - INTEGER; the reassociated class name flag (for evaluation) determines whether or not the class name to be used is the original class name or its alias

NLVELM - INTEGER; number of elements in an LV file entry

NILN - INTEGER; number of incomplete logic nodes

OLPARS Program Specifications
LIPCAP

FILES:

LI - logic information

SUBPROGRAMS REFERENCED:

FPUT, INSPGM, INSRET

SEE ALSO:

LIGCAP

HISTORY:

designed by Steve Haehn September 13, 1978

programmed by Donna Morris July 17, 1980

PROGRAM NAME: LIPCLP

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LIPCLP writes out the classes-present bit map of a logic entry to a logic tree.

PROGRAM USAGE:

CALL LIPCLP(FDLI,ENTNO,CPRSNT,NCPRS)

INPUT ARGUMENTS:

FDLI - INTEGER; file descriptor of the LI file being accessed

ENTNO - INTEGER; entry number of the logic entry to which the classes-present bit map will be written

CPRSNT - REAL array (BMPSIZ); bit map designating the classes present (residing) at a logic node

NCPRS - INTEGER; the number of classes present at a logic node

FILES:

LI - logic information

SUBPROGRAMS REFERENCED:

FPUT, INSINT, INSPGM, INSRET

SEE ALSO:

LIGCLP

HISTORY:

designed by Steven Haehn September 14, 1978

programmed by Steven Haehn Dec. 4, 1980

OLPARS Program Specifications
LIPCNM

PROGRAM NAME: LIPCNM

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LIPCNM writes out the class names of the lowest nodes of a design data set to the logic information (LI) file header of a logic tree.

PROGRAM USAGE:

CALL LIPCNM(FIDLI,CLSNAM,NCLAS)

INPUT ARGUMENTS:

FIDLI - INTEGER; file descriptor of the LI file being accessed

CLSNAM - INTEGER array (4,NCLAS+1); storage area for class names to be written out to the LI file header (+1 to fool array checking if ever NCLAS equals zero)

NCLAS - INTEGER; the number of class names being written out to the LI file header

FILES:

LI - logic information

SUBPROGRAMS REFERENCED:

FPUT, INSPGM, INSRET

SEE ALSO:

LIGCNM

HISTORY:

designed by Steve Haehn September 12, 1978

programmed by Donna Morris July 17, 1980

PROGRAM NAME: LIPCOP

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LIPCOP writes out to an LI file header a count or pointer data value. The count or pointer to be written is given by one of the following codes:

CODE	INFORMATION REQUESTED	ELEMENT IN LI HEADER
1	Dimensionality of design data set	13
2	Number of lowest nodes in design data set	14
3	Next available node entry	15
4	Next open node entry at end of file	16
5	Number of nodes actively used in logic tree	17
6	Current logic node	18
7	Alias flag	19
8	Number of elements in LV file entry	20
9	Number of incomplete logic nodes	21

PROGRAM USAGE:

```
CALL LIPCOP(FDLI,CPCODE,CNTPTR)
```

OLPARS Program Specifications
LIPCOP

INPUT ARGUMENTS:

FDLI - INTEGER; file descriptor of the LI file being accessed

CPCODE - INTEGER; count or pointer code representing data to be accessed

CNTPTR - INTEGER; count or pointer to be written to LI file header

FILES:

LI - logic information

SUBPROGRAMS REFERENCED:

FPUT, INSPGM, INSRET

SEE ALSO:

LIGCOP

HISTORY:

designed by Steven Haehn September 18, 1978

programmed by Steven Haehn Dec. 4, 1980

PROGRAM NAME: LIPDCN

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LIPDCN writes the design data set class name and the reassociated class name into an LI file entry.

PROGRAM USAGE:

```
CALL LIPDCN(FIDLI,ENTNO,ODCN,RCN)
```

INPUT ARGUMENTS:

```
FIDLI - INTEGER; the file descriptor of the LI file
ENTNO - INTEGER; the entry number
 ODCN - INTEGER array (NODLEN); the design data set name
  RCN - INTEGER array (NODLEN); the reassociated name
```

SUBPROGRAMS REFERENCED:

FPUT, INSPGM, INSRET, ITREAL

HISTORY:

designed by David Birnbaum September 20, 1978

programmed by Donna Morris July 2, 1981

OLPARS Program Specifications
LIPDSN

PROGRAM NAME: LIPDSN

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LIPDSN writes the design data set name into the logic information (LI) file header of a user's logic tree.

PROGRAM USAGE:

CALL LIPDSN(FIDLI,DSNAME)

INPUT ARGUMENTS:

FIDLI - INTEGER; file descriptor of the LI file being accessed

DSNAME - INTEGER(12)array; name of the logic tree's design data set, which is a treename, nodename pair

FILES:

LI - logic information

SUBPROGRAMS REFERENCED:

FPUT, INSPGM, INSRET

SEE ALSO:

LIGDSN

HISTORY:

designed by Steve Haehn September 12, 1978

programmed by Donna Morris July 17, 1980

PROGRAM NAME: LIPENT

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LIPENT writes out a logic information (LI) file entry into a logic tree.

PROGRAM USAGE:

```
CALL LIPENT(FIDLI,ENTNO,LOGCD,STRUCT,ODCNAM,RCNAM,
            MLFLAG,NUMCLS,BITMAP)
```

INPUT ARGUMENTS:

FIDLI - INTEGER; file descriptor of LI file to be accessed

ENTNO - INTEGER; entry number of the LI entry to be written

LOGCD - INTEGER array (3); logic code array:

- o element 1 - logic type code of this node (i.e. logic that produced the nodes immediately under this node)
- o or - entry link (to a deleted node) in free node list
- o element 2 - logic subtype code
- o element 3 - option number of creating routine

STRUCT - INTEGER array (7); storage area for the logic node structure pointers

ODCNAM - INTEGER array (4); original design data set class name for the node entry

RCNAM - INTEGER array (4); reassociated class name (ALIAS) is used when the reassociate name flag (RNF or ALIAS flag, in the LI file header) is turned on

MLFLAG - INTEGER; modified logic flag

NUMCLS - INTEGER; number of classes present (residing) at this node

BITMAP - REAL array (2); bit map representing the classes present (residing) at this node

FILES:

LI - logic information

SUBPROGRAMS REFERENCED:

FPUT, INSPGM, INSRET, ITREAL

SEE ALSO:

LIGENT

HISTORY:

designed by Steve Haehn September 13, 1978

programmed by Donna Morris July 28, 1980

PROGRAM NAME: LIPLOG

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LIPLOG writes out the logic code information and the modified logic flag of a logic entry to the logic information (LI) file. (The type code portion of the logic code information may be a "free" list link pointer.)

PROGRAM USAGE:

```
CALL LIPLOG(FDLI,ENTNO,TYPECD,SUBTYP,OPTNO,MODLF)
```

INPUT ARGUMENTS:

FDLI - INTEGER; file descriptor of the LI file being accessed

ENTNO - INTEGER; entry number of the logic entry desired

TYPECD - INTEGER; logic type code of this node (i.e., logic that produced the nodes immediately under this node)

or: entry link (to a deleted node) in free node list

SUBTYP - INTEGER; logic subtype code

OPTNO - INTEGER; option number of creating routine

MODLF - INTEGER; modified logic flag

FILES:

LI - logic information

REFERENCED BY:

ELIFRE

SUBPROGRAMS REFERENCED:

FPUT, INSINT, INSPGM, INSRET

OLPARS Program Specifications
LIPLOG

SEE ALSO:

LIGLOG

HISTORY:

designed by Steven Haehn September 19, 1978

programmed by Steven Haehn Dec. 3, 1980

PROGRAM NAME: LIPPRB

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LIPPRB writes out the a priori probabilities of the lowest nodes of a design data set to the LI file header.

PROGRAM USAGE:

CALL LIPPRB(FIDLI,PROB,NCLAS)

INPUT ARGUMENTS:

FIDLI - INTEGER; file descriptor of the LI file being accessed

NCLAS - INTEGER; the number of classes that have a probability number (or the number of probabilities) to be written in the LI file header

PROB - REAL array (NCLAS+1); storage area for the a priori probability numbers to be written to the LI file header (+1 to fool array checking if ever NCLAS equals zero

FILES:

LI - logic information

SUBPROGRAMS REFERENCED:

FPUT, INSPGM, INSRET

SEE ALSO:

LIGPRB

HISTORY:

designed by Steve Haehn September 12, 1978

programmed by Donna Morris July 17, 1980

PROGRAM NAME: LIPSTR

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LIPSTR puts structural information into the LI file entry. It places them in an array STRUCT as follows:

POSITION IN STRUCT	DATA	ELEMENT IN LI ENTRY
1	Node level	4
2	Number of nodes below this node at next level	5
3	Entry number of parent node	6
4	Entry of first child	7
5	Entry number of next sibling	8
6	Entry number in LV for start of decision logic	9
7	Entry number in LV for reject strategy	10

PROGRAM USAGE:

CALL LIPSTR(FDLI,ENTNO,STRUCT)

INPUT ARGUMENTS:

FDLI - INTEGER; the file descriptor of the LI file
ENTNO - INTEGER; the entry number
STRUCT - INTEGER array (LSTRSZ); the structural info.

SUBPROGRAMS REFERENCED:

FPUT, INSPGM, INSRET, ITREAL

HISTORY:

designed by David BirnBaum September 20, 1978

programmed by Steven Haehn Dec. 4, 1980

PROGRAM NAME: LISTLOGS

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

LISTLOGS displays the list of logics along with their design sets.

PROGRAM USAGE:

User types in 'LISTLOGS'.

ALGORITHMS / NOTES:

Erase screen and home cursor (ERASE).

IF (number of logics = 0) THEN

Print message (TRMPUT).

ELSE

Display all logic names together with the design sets of those logics and whether or not logic is complete.

ENDIF

Put up option list at user's terminal (MENU).

END

FILES:

LI - logic information
LL - logic list
CM - communication

REFERENCED BY:

OLPARS user.

SUBPROGRAMS REFERENCED:

ERASE, INSCHR, INSINI, INSRET, LLGENT, LLGHDR, MENU, OEXIT, OOPEN, OPENFX, TRMPUT, WRTNOW

HISTORY:

designed by Dave Birnbaum

programmed by Mark Maginn July 22, 1980

OLPARS Program Specifications
LISTREES

PROGRAM NAME: LISTREES

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

LISTREES displays, at the terminal, the names of all the trees in the user's TREELIST file.

PROGRAM USAGE:

User types in 'LISTREES'.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

IF (number of trees in TL file = 0) THEN

Print message (TRMPUT).

ELSE

Display at terminal all treenames in the TL file.

ENDIF

Put up option list at user's terminal (MENU).

END

FILES:

CM - communication
TL - treelist

REFERENCED BY:

OLPARS user.

SUBPROGRAMS REFERENCED:

ERASE, INSINI, INSRET, MENU, OEXIT, OPENFX, TLGENT, TLGHDR, TRMPUT, WRTNOW

HISTORY:

designed by Dave Birnbaum March 7, 1978

programmed by Mark Maginn July 21, 1980

PROGRAM NAME: LLGENT

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LLGENT obtains an entry within the Logic List file. LLGENT is passed the file descriptor of the Logic List (LL) file and the number of an entry within the LL file. It returns to the calling program the file codes for the Logic Information and Logic Value files, the logic name, the number of dimensions in the design data set, the link pointer, the tree and class name of the design data set, and the incomplete logic flag.

PROGRAM USAGE:

```
CALL LLGENT(FIDLL,ENTNO,FDCLI,FCDLV,LOGNAM,NDIM,LNKPTR,
            DSNAME,INCLOG)
```

INPUT ARGUMENTS:

FIDLL - INTEGER; file descriptor of the logic list file

ENTNO - INTEGER; entry number of the LL entry desired

OUTPUT ARGUMENTS:

FCDLI - INTEGER; file code of the logic information file

FCDLV - INTEGER; file code of the logic value file

LOGNAM - INTEGER array (TRELEN); name of the logic tree

NDIM - INTEGER; the dimensionality of the design data set

LNKPTR - INTEGER; the link pointer - points to the next entry in the alphabetcally linked list

DSNAME - INTEGER array (TRELEN + NODLEN); design data set name

INCLOG - INTEGER; incomplete logic flag

OLPARS Program Specifications
LLGENT

FILES:

LL - logic list
Instrumentation files

REFERENCED BY:

LLSRCH, DELETR

SUBPROGRAMS REFERENCED:

FGET, PACKW

SEE ALSO:

TLGENT

HISTORY:

designed by Steve Haehn September 1, 1978

programmed by Donna Morris July 27,1979

PROGRAM NAME: LLGHDR

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LLGHDR obtains the number of entries element (NOE) and the alphabetic link pointer element (LNKPTR) from the LL file header.

PROGRAM USAGE:

```
CALL LLGHDR(FIDLL,NOE,LNKPTR)
```

INPUT ARGUMENTS:

FIDLL - INTEGER; file descriptor of the LL file

OUTPUT ARGUMENTS:

NOE - INTEGER; the number of entries

LNKPTR - INTEGER; the link pointer which points to the first entry in the alphabetically linked list

FILES:

LL - logic list
Instrumentation files

REFERENCED BY:

LLSRCH, DELETR, DLOGTREE

SUBPROGRAMS REFERENCED:

FGET,TRMPUT

HISTORY:

designed by Steve Haehn September 5, 1978

programmed by Donna Morris July 27,1979

PROGRAM NAME: LLPENT

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LLPENT writes an entry into the Logic List (LL) file. It is passed the file descriptor of the LL file (FIDLL), the logical entry number (ENTNO), and the information on the tree to be placed in the LL file.

PROGRAM USAGE:

```
CALL LLPENT(FIDLL,ENTNO,FCDLI,FCDLV,LOGNAM,NDIM,LNKPTR,
            DSNAME,INCLOG)
```

INPUT ARGUMENTS:

FIDLL - INTEGER; file descriptor of the logic list file

ENTNO - INTEGER; entry number of the LL entry to be written

FCDLI - INTEGER; file code of the logic information file

FCDLV - INTEGER; file code of the logic value file

LOGNAM - INTEGER array (TRELEN); name of the logic tree

NDIM - INTEGER; the dimensionality of the design data set

LNKPTR - INTEGER; the link pointer - points to the next entry in the alphabetically linked list

DSNAME - INTEGER array(TRELEN + NODLEN); design data set name

INCLOG - INTEGER; incomplete logic flag

FILES:

LL - logic list
Instrumentation files

SUBPROGRAMS REFERENCED:

FPUT, ITREAL

SEE ALSO:

TLPENT

HISTORY:

designed by Steve Haehn September 5, 1978

programmed by Donna Morris July 27,1979

OLPARS Program Specifications
LLPHDR

PROGRAM NAME: LLPHDR

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LLPHDR writes the number of entries element (NOE) and the alphabetic link pointer element (LNKPTR) to the LL file header.

PROGRAM USAGE:

CALL LLPHDR(FIDLL,NOE,LNKPTR)

INPUT ARGUMENTS:

FIDLL - INTEGER; file descriptor of the logic list file

NOE - INTEGER; the number of entries in the logic list file

LNKPTR - INTEGER; the link pointer which points to the first entry in the alphabetically linked list

FILES:

LL - logic list

SUBPROGRAMS REFERENCED:

FPUT

HISTORY:

designed by Steve Haehn September 5, 1978

programmed by Donna Morris July 27,1979

PROGRAM NAME: LLSRCH

CATEGORY: Logic List File Access Routine

PROGRAM DESCRIPLION:

LLSRCH is passed the file descriptor (in FIDLL) of the logic list file and the name of a tree (in NAME). LLSRCH then searches the logic list for a tree with the same name. If a match is found, LLSRCH is set equal to true and the file codes and dimension of the tree are placed in FCDLI, FDCLV and NDIM, respectively. If no match is found, NDIM and ENTNO are set to zero and FCDLI and FCDLV are set to equal the entry number and link pointer, respectively, needed to update the logic list file alphabetic linked list when a new entry with the given name (NAME) is added to the logic list.

PROGRAM USAGE:

```
LOGICAL LLSRCH
        .
        .
        .
IF (LLSRCH(FIDLL,ENTNO,NAME,FCDLI,FCDLV,NDIM))
```

INPUT ARGUMENTS:

FIDLL - INTEGER; the file descriptor of the LL file

NAME - INTEGER array (TRELEN); name of the logic tree to search for

OUTPUT ARGUMENTS:

ENTNO - INTEGER; if llsrch is true, ENTNO contains the entry number of the LL entry found.

FCDLI - INTEGER; if llsrch is true, FCDLI is the file code of the LI file. if llsrch is false, FCDLI is the entry number of the treename that alphabetically precedes the position where NAME belongs in the linked list and is used by the calling routine to update the list if the entry NAME is added to the logic list.

FCDLV - INTEGER; if llsrch is true, FCDLV is the file code of the LV file. if llsrch is false, FCDLV is the link pointer that should be written to the entry NAME in the event that the calling routine adds the entry NAME to the logic list.

NDIM - INTEGER; if llsrch is true, NDIM is the dimension of the data set. if llsrch is false, NDIM is set to zero.

ALGORITHM / NOTES:

LLsrch searches the tree alphabetically by following the linked list pointers.

Read in the LL file header to get the first link pointer

I = link pointer

while (I is not equal to the last link pointer (-1))

get entry I (we will use its name and link pointer)

if (treenames match) then

llsrch =.true.
break

elseif (NAME > the LL entry name) then

I = link pointer
stay in the loop to get the next entry

else

llsrch = .false.
return the link information in FCDLI and FCDLV
set NDIM to zero
break

endif

endwhile

```
if (link pointer = -1)

     llsrch = .false.
     return the link information in FCDLI and FCDLV
     set NDIM to zero

endif
```

FILES:

LL - logic list

REFERENCED BY:

Any routine that must check to see that a logic tree exists or must use a logic set.

SUBPROGRAMS REFERENCED:

LLGHDR, LLGENT

HISTORY:

designed by Dave Birnbaum May 19, 1978

programmed by Donna Morris July 10, 1979

OLPARS Program Specifications
LNPROJ

PROGRAM NAME: LNPROJ

CATEGORY: Display File Access Routine

PROGRAM DESCRIPTION:

LNPROJ projects the vectors at a logic node (or all of the vectors in the classes at that logic node) onto a projection vector or projection vectors for one-space or two-space displays. LNPROJ is called by the L1 and L2 commands (except CRDV). The results are stored in the DI and DV files and original maximum, minimum values are computed.

PROGRAM USAGE:

```
CALL LNPROJ(FIDTI,FIDTV,FIDDI,FIDDV,FIDPV,CLIST,NCLASS,
            ALLVEC,ET,PROJTY,LOGNOD)
```

INPUT ARGUMENTS:

FIDTI - INTEGER; the file descriptor of the TI file

FIDTV - INTEGER; the file descriptor of the TV file

FIDDI - INTEGER; the file descriptor of the DI file

FIDDV - INTEGER; the file descriptor of the DV file

FIDPV - INTEGER; the file descriptor of the PV file

CLIST - INTEGER array (MAXCLS); the slot numbers corresponding to each of the classes

NCLASS - INTEGER; the number of classes which lie at the logic node

ALLVEC - LOGICAL; TRUE means all vectors to be projected
FALSE means only vectors at logic node

ET - INTEGER array (TABSIZ); the TI file entry table

PROJTY - INTEGER; the projection type
1 for one-space, 2 for two-space

LOGNOD - INTEGER; the logic node number

ALGORITHM / NOTES:

```
Get the first projection vector (PVGENT).

IF (two-space display) get the second projection
vector (PVGENT).

DO I=1, number of classes to be projected

   Get the number of vectors in the class (TIGCOP).

   IF (not all the vectors are to be used) THEN

      Initialize the mean vector to zero.

   ELSE

      Get the mean vector (TIGMN).

   ENDIF

   DO J=1, number of vectors in the class

      Get the next vector.

      IF (not all vectors are to be used) THEN

         IF (this vector lies at the logic node) THEN

            Compute a 'running' mean vector.

         ELSE

            Next

         ENDIF

      ENDIF

      Project the vector onto the projection
      vector(s) (PROJCT).

      Update x(min), x(max), y(min), y(max).

      Store the projected vector in the DV file (DVPVEC).

   ENDDO

   Project the mean vector onto the projection
   vector(s) (PROJCT).
```

Store the projected mean vector in the DV file (DVPVEC).

Create a DI file logical entry for this class (DIPENT).

Put blanks in the display flag symbol (DIPEFS).

Calculate the total number of vectors used in the projection.

ENDDO

Store xmin, xmax, ymin, ymax values in the DI file header as original and current (DIPMAX).

Store the total number of vectors in the DI file header (DIPHDI).

Update the next available entry in the DV file header (DVPHDR).

RETURN

FILES:

DI - display information file
DV - display value file
PV - projection vector file
TI - tree information file
TV - tree vector file
instrumentation file

REFERENCED BY:

L1ARBV, L2ARBV, L1ARDG, L2ARDG, L1ASDG, L2ASDG, L1EIGV, L2EIGV, L2FSHP, L2GNDV, L2GNDV

SUBPROGRAMS REFERENCED:

DIPEFS, DIPENT, DIPHDI, DIPMAX, DVPHDR, DVPVEC, INSINT, INSPGM, INSRET, PROJCT, PVGENT, PVGHDR, TIGCOP, TIGMN, TVGVEC

HISTORY:

designed by David Birnbaum September 12, 1978

programmed by Jill King December 12, 1980

PROGRAM NAME: LOGEVAL

CATEGORY: Logic Design Command

PROGRAM DESCRIPTION:

LOGEVAL enables the user to test any logic, complete or incomplete, against a data set or vector having the same dimensionality. The logic used is the current logic, and the data set used is the current data set. When LOGEVAL works with a data set, it always sets the temporary logic element of each vector it evaluates to the value of the logic node at which the vector was placed.

LOGEVAL operates in four modes which are described as follows:

1. Complete. In this mode a data set (the test set) is run against a completed logic and a confusion matrix display is created. Vectors from a class are incorrectly classified if they are assigned to a logic node of a different class. In the event that the class names of the test set are not the same as the class names of the data set on which the logic was designed (the design set), the command REASNAME may be invoked before proceeding with logic evaluation. The user may have the confusion matrix and error information output on a line printer.

2. Restore. The restore mode is used to restore the temporary logic elements of the vectors in the TV file of the design data set to the correct logic node number in an incomplete logic (although it is possible to restore a data set to a completed logic). It is envisioned that this mode would be called in the following circumstances. Suppose that a user begins a logic design on a data set but does not complete it. Suppose further, that a different logic is designed or evaluated against the same data set. Before the first logic is completed, the user must reset the temporary logic elements so that the logic design programs will know what vectors fall at what logic nodes. This is the purpose of the restore mode of LOGEVAL. No confusion matrix or error information is available for this mode.

3. Classifier. In this mode a test data set is evaluated against a complete or incomplete logic to determine the number of vectors assigned to each node in the logic. The classes that the vectors might have come from in the data set are disregarded. The output is a table, which lists (for each lowest logic node) the logic node number, the class associated with that node number (**** if the node is a reject node or 'INCOMPLETE' if the node is incomplete) and the number of vectors from the data set falling at the node. This table can be reproduced on the line printer. A listing of vector ID's, along with the logic nodes to which the vectors were assigned, may also be printed.

4. Single Vector. In single vector mode the user may pass a single vector through a logic and have the result displayed on the screen. The vector may come from the current data set, in which case it would be identified by its vector ID, or the vector may be given directly to LOGEVAL by the user.

All four modes call the program EVALU8 to perform the evaluation of a vector against a logic. EVALU8 returns the logic node number to which the vector was assigned and any pertinent error information.

PROGRAM USAGE:

User types in 'LOGEVAL'.

USER INTERACTION:

See Algorithm/Notes.

ALGORITHM / NOTES:

Erase the screen and home the cursor (ERASE).

Retrieve the current data set and current logic information from the CM file (CMGCDS,CMGLOG).

Prompt user for the mode of operation (UILGE1).

Get the option number of 'LOGEVAL' for the DI confusion matrix file (SETOPT).

Open the logic file (OPENTR) and get the dimension and name of the design data set from its LI file header (LIGCOP, LIGDSN).

```
IF (single vector mode was chosen) THEN

    Find out if the vector to be used is coming
    from the current data set or will be typed
    in by the user.  This is done so we can
    determine whether we need the current data set.

ENDIF

IF (we will be using the current data set) THEN

    Open the current data tree files (OPENTR)

    IF (the dimension of the current data set does not
    equal the dimension of the logic) THEN

        Print error message and exit

    ENDIF

LNDIF

Initialize common variables to be used in the
logic evaluation.


COMPLETE MODE:
IF (complete mode is chosen) THEN

    IF (there are reassociated class names in the LI
        header)

        Ask user if he wishes to use the reassociated
        class names.

    ENDIF

    Open the DI file for a confusion matrix display.

    Open an error (misclassification) file (OPENS).
    Write out a header to the error file - include
    current data set name, design data set name, and
    current logic name (FILPUT).

    IF (we will not be using the reassociated class
        names)

        Read the design data set class names in the LI
        header (LIGCNM).
```

```
        ELSE

            Search the logic tree for lowest nodes in order
            to get their reassociated class names (LIGCOP,
            GNXTLN, LIGCLP, LIGDCN).

        ENDIF

        Get the number of lowest nodes, their entry table
        slot numbers, and their names (TRUE CLASS NAMES)
        (GLONOD).

        DO WHILE (there are more classes in the data set)

            Get the number of vectors and the vector pointer
            for the current lowest node (TIGCOP).

            Write the name of the data class to the error
            file (FILPUT).

            DO WHILE (there are more vectors in the class)

                Get a vector (TVGVEC).

                Evaluate it against the logic (EVALU8).

                Write new temporary logic element for the
                vector into the TV file (TVPVEC).

                IF (the vector was rejected) THEN

                    Update the reject counts.

                    Place vector ID and any pertinent error
                    information in the error file (FILPUT).

                ELSEIF (the vector was assigned to the
                        correct class) THEN

                    Update the 'number of correctly assigned
                    vectors' count.

                ELSE

                    Update the 'number of incorrectly
                    assigned vectors' count.

                    Place vector ID and any pertinent error
                    information in the error file (FILPUT).

                ENDIF

            ENDDO
```

```
            Calculate, for this data class, the percent
            correct, the percent error, and the percent
            rejects.

            Write a DI logical entry for the confusion matrix
            display for this class (DIPCME).

            Write 'total number of errors' and 'total number
            of rejects' for this class to the error file
            (FILPUT).

        ENDDO

        Write the logic name and logic tree file codes to the
        TV file (TVPHDR).

        Calculate the total percents for correct, error, and
        reject vectors and write the DI file header for the
        confusion matrix display (DIPCMH).

        Present the confusion matrix at the terminal
        (CMDISP).

        IF (user wishes a printout of the confusion matrix or
        the error listing) THEN

            Print them (CMPRT, PRINTR).

        ENDIF


RESTORE MODE:
ELSEIF (restore mode is chosen) THEN

        IF (the current data set name is not the same as the
            design data set name) THEN

            Print an error message and exit.

        ENDIF

        Get the number of lowest nodes, their entry table
        slot numbers, and their names (GLONOD).

        DO WHILE (there are more classes in the data set)

            Get the number of vectors and the vector pointer
            for the current lowest node (TIGCOP).
```

```
            DO WHILE (there are more vectors in the class)

                Get a vector (TVGVEC).

                Evaluate it against the logic (EVALU8).

                Write new temporary logic element for the
                vector into the TV file (TVPVEC).

            ENDDO

        ENDDO


    CLASSIFIER MODE:
    ELSEIF (classifier mode is chosen) THEN

        IF (there are reassociated class names in the LI
            header)

            Ask user if he wishes to use the reassociated
            class names.

        ENDIF

        Get the number of lowest nodes, their entry table
        slot numbers, and their names (GLONOD).

        Get a list of lowest logic node numbers (GETLST).

        DO WHILE (there are more classes in the data set)

            Get the number of vectors and the vector pointer
            for the current lowest node (TIGCOP).

            DO WHILE (there are more vectors in the class)

                Get a vector (TVGVEC).

                Evaluate it against the logic (EVALU8).

                Write new temporary logic element for the
                vector into the TV file (TVPVEC).

                Update the count of the number of vectors
                assigned to the logic node number returned
                by 'EVALU8'.

            ENDDO

        ENDDO
```

```
Present tabular output at terminal (TRMPUT).
Include the logic node number, associated class
(or **** (for reject node) or INCOMPLETE (for
more than 1 class at a node)), and the number
of vectors from the data set assigned to the
logic node.

IF (user desires a printout of the table) THEN

    Open a sequential file (OPENS).

    Write table into it (FILPUT).

    Print the file (PRINTR).

ENDIF

IF (user desires a printout of vector ID's and the
logic node numbers to which the vectors were
assigned) THEN

    Open a sequential file (OPENS).

    DO WHILE (there are more classes in the data set)

        Get number of vectors and vector pointer of
        the class (TIGCOP).

        DO WHILE (there are more vectors in the
                  class)

            Get the vector ID and temporary logic
            element of the vector (TVGVEC).

            Determine if the logic node is complete
            (LIGCLP).

            IF (the logic node is complete) THEN

                Get the associated class name from
                the LI file.

            ENDIF

            Write the vector ID, the temporary logic
            element, and the associated class name
            (or 'INCOMPLETE') to the sequential file
            (FILPUT).

        ENDDO

    ENDDO
```

```
            Print the file (PRINTR).

        ENDIF


    SINGLE VECTOR MODE:
    ELSEIF (single vector mode chosen) THEN

        IF (there are reassociated class names in the LI
            header)

            Ask user if he wishes to use the reassociated
            class names.

        ENDIF

        Get the vector to be used (UILGE3).  The vector
        will either come from the current data set or will
        be typed in by the user.
            ALGORITHM FOR MODULE UILGE3:

            IF (vector to be used is from current data set)

                REPEAT

                    Prompt for the class symbol.
                    Prompt for the vector id.

                UNTIL (the vector is found)

            ELSE

                REPEAT

                    REPEAT

                        Prompt for a vector element.

                    UNTIL (vector element is valid)

                UNTIL (the number of elements entered equals
                       the logic tree dimensionality)

            ENDIF

        END OF UILGE3 ALGORITHM.

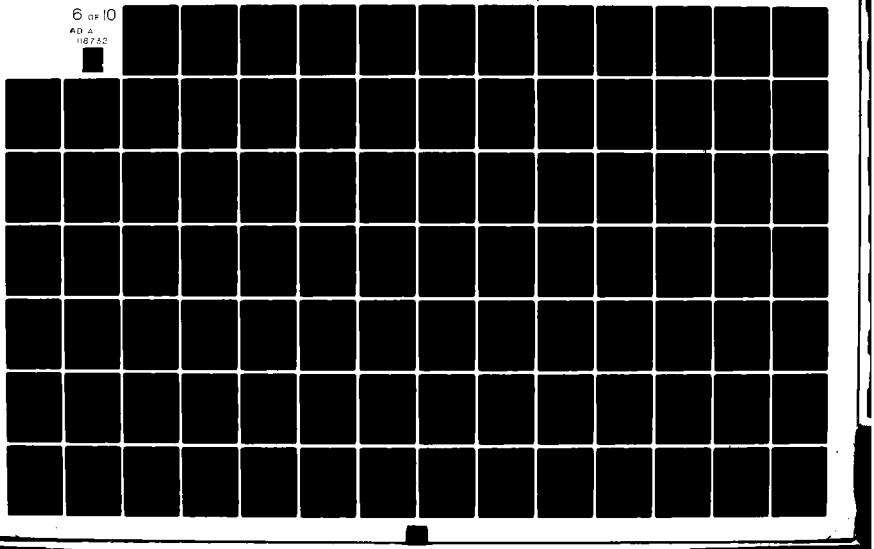
        Evaluate the vector against the logic (EVALU8).

        Display the results at the user's terminal.

    ENDIF
```

AD-A118 732 PAR TECHNOLOGY CORP NEW HARTFORD NY F/G 9/2
ON-LINE PATTERN ANALYSIS AND RECOGNITION SYSTEM. OLPARS VI. SOF--ETC(U)
JUN 82 S E HAEHN, D MORRIS
UNCLASSIFIED PAR-82-20 NL

6 OF 10
AD A
118732



Put up option list at user's terminal (MENU).

END

FILES:

BLOCK I/O FILES	RECORD I/O FILES
HS - history	INSTRU.DAT
CM - communications	MISCLASS. (error file)
OPTION.OLP - option file	CONMAT. (confusion matrix)
LI - logic information	
LV - logic value	CLASIFY1. (classifier
TI - tree information (CDS)	CLASIFY2. mode files)
TV - tree vector (CDS)	
DI - display file	
TI - reference pattern data tree	
TV - reference pattern data tree	

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLFYMD, CLOSTR, CMPMOD, ERASE, INSINI, INSRET, LSETUP, MENU, OEXIT, OPENFX, RSTORE, SINGLE, TRMPUT

HISTORY:

designed by Dave Birnbaum October 12, 1978

programmed by Donna Morris April 15, 1981

OLPARS Program Specifications
LOGMN

PROGRAM NAME: LOGMN

CATEGORY: Numerical Computation Algorithm

PROGRAM DESCRIPTION:

LOGMN computes the mean array of all vectors in a class that lie at a particular node. These vectors are retrieved one at a time to save storage space, so instead of using the standard formula for computing the mean array, a 'running' mean array is computed in order to update it after retrieving each vector.

PROGRAM USAGE:

CALL LOGMN(FDTI,FDTV,LOGNOD,NDIM,ENTNO,MEAN,NVEC)

INPUT ARGUMENTS:

FDTI - INTEGER; the file descriptor of the TI file

FDTV - INTEGER; the file descriptor of the TV file

LOGNOD - INTEGER; the logic node number

NDIM - INTEGER; the dimension of the data set

ENTNO - INTEGER; the entry number of the class in the TI file

OUTPUT ARGUMENTS:

MEAN - REAL array (Ndim); the mean vector

NVEC - INTEGER; the actual number of vectors used in the computations

ALGORITHM/NOTES:

Retrieve the number of vectors and the entry number of the first vector in the class (TIGCOP).

Initialize the mean array.

DO I=1,number of vectors in class

Retrieve the next vector from the TV file (TVGVEC).

```
      IF (the vector is at the node specified) THEN

         Update the 'running' mean array to include the
         new vector.

      ENDIF

   ENDDO

   RETURN
```

FILES:

```
   TI - tree information file
   TV - tree vector file
   instrumentation file
```

REFERENCED BY:

L1ASDG, L2ASDG

SUBPROGRAMS REFERENCED:

INSFLT, INSINT, INSPGM, INSRET, TIGCOP, TVGVEC

HISTORY:

designed by Jill King May 4, 1981

programmed by Jill King May 4, 1981

OLPARS Program Specifications
LOGMNC

PROGRAM NAME: LOGMNC

CATEGORY: Numerical Computation Algorithm

PROGRAM DESCRIPTION:

LOGMNC computes the mean and covariance of all vectors in a class that lie at a particular node. These vectors are retrieved one at a time to save storage space, so instead of using the standard formulas for computing the mean and covariance arrays, a 'running' factor variation array is computed so that it can be updated after retrieving each vector.

PROGRAM USAGE:

CALL LOGMNC(FDTI,FDTV,LOGNOD,NDIM,ENTNO,MEAN,COV,NVEC)

INPUT ARGUMENTS:

FDTI - INTEGER; the file descriptor of the TI file

FDTV - INTEGER; the file descriptor of the TV file

LOGNOD - INTEGER; the logic node number

NDIM - INTEGER; the dimension of the data set

ENTNO - INTEGER; the entry number of the class in the TI file

OUTPUT ARGUMENTS:

MEAN - REAL array (Ndim); the mean vector

COV - REAL array ((Ndim*(Ndim+1))/2), the lower triangular portion of the covariance matrix, stored as a vector

NVEC - INTEGER; the actual number of vectors used in the computations

ALGORITHM/NOTES:

Retrieve the number of vectors and the entry number of the first vector in the class (TIGCOP).

Initialize the factor variation and mean arrays.

DO I=1,number of vectors in class

Retrieve the next vector from the TV file (TVGVEC).

IF (the vector is at the node specified) THEN
update the 'running' factored variation arrays
to include the new vector.

ENDIF

ENDDO

Compute the final covariance array using the finished mean vector.

RETURN

FILES:

TI - tree information file
TV - tree vector file
instrumentation file

REFERENCED BY:

NMVBSU, S2FSHP, L2FSHP

SUBPROGRAMS REFERENCED:

INSFLT, INSINT, INSPGM, INSRET, TIGCOP, TVGVEC

HISTORY:

designed by David Birnbaum September 27, 1978

programmed by Jill King October 16, 1980

OLPARS Program Specifications
LTRENAME

PROGRAM NAME: LTRENAME

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

Change the name of an OLPARS logic tree.

PROGRAM USAGE:

User types in 'LTRENAME'.

USER INTERACTION:

The name of the logic tree to be changed is requested from the user, along with the new name of the tree.

FILES:

CM - communications
LL - logic list
LI - logic information
LV - logic value
HS - history
OLPARS option
Instrumentation

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLOSFX, CLOSTR, CMGOTH, EQUALA, ERASE, INSINI, INSRET, LGLTRE, LLSRCH, MENU, OEXIT, OPENFX, OPENTR, PROMPT, RENAMT, TRMGET, TRMPUT

SEE ALSO:

DTRENAME

HISTORY:

designed by Steven Haehn Aug. 15, 1981

programmed by Steven Haehn Sept. 28, 1981

PROGRAM NAME: LVGENT

CATEGORY: Level II File Access Routine

PROGRAM DESCRIPTION:

LVGENT is used to retrieve an entire Logic Value entry.

PROGRAM USAGE:

```
CALL LVGENT(FDLV,ENTNO,NBRELM,ENTRY,LNKVAL)
```

INPUT ARGUMENTS:

```
  FDLV - INTEGER; file descriptor of Logic Value file
 ENTNO - INTEGER; entry to be accessed
NBRELM - INTEGER; number of elements to access in the
                  entry (less than or equal to entry
                  size)
```

OUTPUT ARGUMENTS:

```
 ENTRY - REAL array(NBRELM); elements to access
LNKVAL - INTEGER; link value of entry
```

FILES:

LV - logic value

REFERENCED BY:

GTRGNI, GTRGNR

SUBPROGRAMS REFERENCED:

FGET, INSINT, INSPGM, INSRET, OEXIT, TRMPUT

SEE ALSO:

LVPENT

HISTORY:

designed by Steven Haehn Mar. 5, 1981

programmed by Steven Haehn Mar. 11, 1981

PROGRAM NAME: LVGHDR

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LVGHDR retrieves the logic value (LV) file header for the calling program.

PROGRAM USAGE:

```
CALL LVGHDR(FIDLV,NEU,NOE,NETU)
```

INPUT ARGUMENTS:

FIDLV - INTEGER; the file descriptor of the LV file

OUTPUT ARGUMENTS:

```
   NEU - INTEGER; the total number of entries actively
                  used
   NOE - INTEGER; next open entry at end of file
  NETU - INTEGER; next entry to use
```

FILES:

LV - logic value

SUBPROGRAMS REFERENCED:

FGET, INSPGM, INSRET

SEE ALSO:

LVPHDR

HISTORY:

designed by Dave Birnbaum September 19, 1978

programmed by Donna Morris August 11, 1980

PROGRAM NAME: LVGLNK

CATEGORY: Level II File Access Routine

PROGRAM DESCRIPTION:

LVGLNK retrieves the link element found in a Logic Value file entry.

PROGRAM USAGE:

CALL LVGLNK(FDLV,ENTRY,LNKVAL)

INPUT ARGUMENTS:

FDLV - INTEGER; file descriptor for Logic Value file
ENTRY - INTEGER; entry to be accessed

OUTPUT ARGUMENTS:

LNKVAL - INTEGER; link value

FILES:

LV - logic value

REFERENCED BY:

PTRGNI, PTRGNR

SUBPROGRAMS REFERENCED:

FGET, INSINT, INSPGM, INSRET, OEXIT, TRMPUT

SEE ALSO:

LVPENT

HISTORY:

designed by Steven Haehn Mar. 5, 1981

programmed by Steven Haehn Mar. 11, 1981

OLPARS Program Specifications
LVPENT

PROGRAM NAME: LVPENT

CATEGORY: Level II File Access Routine

PROGRAM DESCRIPTION:

LVPENT is used to write an entire Logic Value entry. (does not alter link value of an entry)

PROGRAM USAGE:

```
CALL LVPENT(FDLV,ENTNO,NBRELM,ENTRY)
```

INPUT ARGUMENTS:

```
  FDLV - INTEGER; file descriptor of Logic Value file
 ENTNO - INTEGER; entry to be accessed
NBRELM - INTEGER; number of elements to access in the
                  entry (less than or equal to entry
                  size)

 ENTRY - REAL array(NBRELM); elements to access
```

FILES:

LV - logic value

REFERENCED BY:

PTRGNI, PTRGNR

SUBPROGRAMS REFERENCED:

FPUT, INSINT, INSPGM, INSRET

SEE ALSO:

LVGENT

HISTORY:

designed by Steven Haehn Mar. 5, 1981

programmed by Steven Haehn Mar. 11, 1981

PROGRAM NAME: LVPHDR

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

LVPHDR writes the Logic Value File header.

PROGRAM USAGE:

```
CALL LVPHDR(FIDLV,NEU,NOE,NETU)
```

INPUT ARGUMENTS:

```
 FIDLV - INTEGER; the file descriptor of the LV file
   NEU - INTEGER; the total number of entries actively
                  used
   NOE - INTEGER; next open entry at end of file
  NETU - INTEGER; next entry to use
```

FILES:

LV - logic value

SUBPROGRAMS REFERENCED:

FPUT, INSPGM, INSRET

SEE ALSO:

LVGHDR

HISTORY:

designed by Dave Birnbaum September 19,1978

programmed by Donna Morris July 28, 1980

OLPARS Program Specifications
LVPLNK

PROGRAM NAME: LVPLNK

CATEGORY: Level II File Access Routine

PROGRAM DESCRIPTION:

LVPLNK writes the link element found in a Logic Value fi_ entry.

PROGRAM USAGE:

CALL LVPLNK(FDLV,ENTRY,LNKVAL)

INPUT ARGUMENTS:

FDLV - INTEGER; file descriptor for Logic Value file
ENTRY - INTEGER; entry to be accessed
LNKVAL - INTEGER; link value

FILES:

LV - logic value

REFERENCED BY:

PTRGNI, PTRGNR, NMVBSU

SUBPROGRAMS REFERENCED:

FPUT, INSINT, INSPGM, INSRET

SEE ALSO:

LVGENT

HISTORY:

designed by Steven Haehn Mar. 5, 1981

programmed by Steven Haehn Mar. 11, 1981

PROGRAM NAME: MACROP

CATEGORY: Terminal Display Routine

PROGRAM DESCRIPTION:

MACROP produces a one-space MACRO plot. MACROP computes screen coordinates, if necessary, and then displays the projected vectors one class at a time.

PROGRAM USAGE:

CALL MACROP(FDCM,FDDI,FDDV,FDS1,SCRN,PAGING,NXTCLS)

INPUT ARGUMENTS:

FDCM - INTEGER; the file descriptor of the CM file

FDDI - INTEGER; the file descriptor of the DI file

FDDV - INTEGER; the file descriptor of the DV file

FDS1 - INTEGER; the file descriptor of the S1 file

SCRN - INTEGER array (19); the OLPARS screen coordinates

NXTCLS - INTEGER; next class to be displayed on the macro plot page

OUTPUT ARGUMENTS:

PAGING - LOGICAL; set to 'true' if more classes are to be displayed and there is no more room on the current page.

NXTCLS - INTEGER; next class to be displayed on the macro plot page (used only if 'PAGING' is true)

ALGORITHM / NOTES:

```
Determine the bins into which the vectors of the given
data set fall (CNT1SP)

WHILE (there are classes to be displayed)

    Retrieve next class entry from DI file (DIGENT)

    IF (the class should be displayed) THEN

        IF (the current display page is full) THEN

            Indicate more pages exist.

            BREAK out of this display loop.

        ELSE

            IF (probabilities are requested) THEN

                Convert bin counts to probablities.

                Display class bin probabilities.

            ELSE

                Display class bin counts

            ENDIF

        ENDIF

    ENDIF

ENDWHILE

RETURN
```

FILES:

CM - communications
DI - display information
DV - display value
S1 - scratch 1

REFERENCED BY:

MICMAC

SUBPROGRAMS REFERENCED:

CMGOTH, CNT1SP, DIGENT, DIGHDI, DIGMAX, DIPHDI, DVGHDR
DVPHDR, INSINT, INSPGM, INSRET, LINSEG, S1GBC, TEXT
TRMPUT

HISTORY:

designed by David BirnBaum July 18, 1978

programmed by Steven Haehn Oct. 16, 1980

PROGRAM NAME: MAKETREE

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

MAKETREE is used to create a new tree from nodes of existing trees.

To COMBINE trees:

MAKETREE can create a new data tree with copies of the lowest nodes of up to ten trees. In the process, if any display symbols are duplicated, new display symbols will be substituted (display symbols of lowest data nodes must always be unique).

To MERGE trees:

MAKETREE can create a new data tree by merging all of the vectors from similarly-named nodes of various trees into nodes (with the same names) in the new tree. A maximum of ten trees can be merged.

PROGRAM USAGE:

User types in 'MAKETREE'.

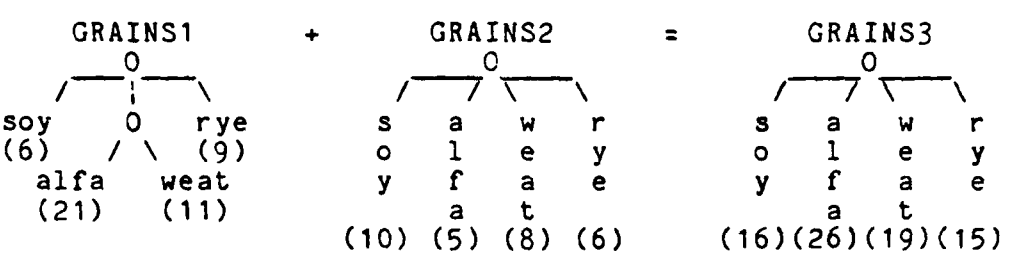
USER INTERACTION:

- User specifies the tree name of the new tree to be created.
- User specifies how the tree is to be created (combined or merged).
- User specifies the names of the trees that will be used in making up the new tree.
- User specifies whether or not the vectors are to be resequenced.

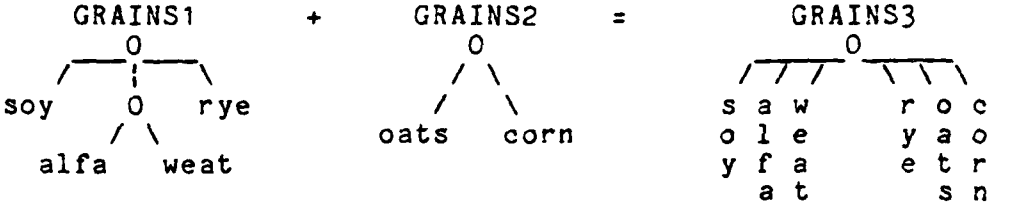
ALGORITHM / NOTES:

In the following diagram, the data trees GRAINS1 and GRAINS2 are going to be merged into the new tree, GRAINS3. Note that the node names and the number of lowest nodes (4) are identical in both GRAINS1 and GRAINS2 (GRAINS1 has a total of 6 nodes, including the intermediate nodes (marked '0'), while GRAINS2 has only 5; this example shows that the

structure of the two data trees being merged is irrelevant to the merging operation). In the diagram, the numbers in parenthesis represent the number of vectors residing at each of the data nodes.



In the next example, the two data trees, GRAINS1 and GRAINS2, are to be combined to form the new data tree, GRAINS3 (see diagram). The symbol 'O', occurring by itself, represents an intermediate data node (i.e., not a lowest node). The data structure of the component trees is irrelevant to MAKETREE.



==========

Ask user for the name of the new tree to be created.

Ask user if (s)he wants to:

1. Place all the lowest nodes of different trees under the senior node of the new tree (combination mode).

2. Same-named lowest nodes of different trees are merged together and placed under the senior node of the new tree (merge mode).

Ask user for the number of trees to be used in making the new tree.

```
REPEAT

    Ask user for the name of a tree to be used in making
    up the new tree.

    Optain the dimensionality of the tree from the tree
    list file (TLSRCH).

    IF (this is the first tree name typed in) THEN

        Save the tree dimensionality.

        IF (the dimensionality of the tree exceeds the
        maximum allowed for covariance and mean calcula-
        tion) THEN

            The tree is the "excess measurement" mode.
            (Indicate a 'NO GO' situation.)

        ENDIF

    ELSEIF (the dimensionality of this tree is not equal
            to the dimensionality of the first tree)

        Tell user that this tree will not be used in the
        tree creation process because of mismatch in the
        vector dimensionality.

        Decrement the number of trees to be used in the
        tree creation process by 1.

    ENDIF

UNTIL (all the tree names have been typed in).

IF (the user chose merge mode) THEN

    Ask user if the vector ID numbers should be
    sequenced (given new ID numbers) as they are
    processed.

    IF (user asked for new ID numbers) THEN

        Ask user if (s)he wants a printer listing of the
        vector number changes.

    ENDIF

ENDIF
```

```
IF (user requested combination mode) THEN (MAKCMB)

    Create senior node of new tree (with empty means
    and covariances)

    WHILE (there are still nodes to process from
           existing trees)

        Get next node from an existing tree.

        Make sure class display symbol is unique.

        Adjust counts and pointers of new node.
        (indicating a lowest node under a senior node)

        Copy means, covariances and vectors of existing
        node to new tree.

    ENDWHILE

    Create TI and TV headers for new tree.

ELSE  (MAKMRG)

    Create senior node of new tree (with empty means
    and covariances)

    WHILE (there are lowest nodes in the first data tree)

        Save next node name (to be compared against later
        tree node names) and vector counts.

        Create new tree's counts and pointers.

        Copy existing tree's means and covariances.

    ENDWHILE

    WHILE (there are more trees)

        Make sure next tree has same number of lowest
        nodes as first tree.

        WHILE (there are lowest nodes in present tree)

            Obtain name of next node and verify that
            this node was found in the first tree.

            IF (node is present in first tree) THEN

                Save vector count.

            ENDIF
```

```
            ENDWHILE

            IF (tree is OK) THEN

                Add saved vector counts to previously
                collected vector counts.

            ENDIF

        ENDWHILE

        From summed vector count information, compute vector
        pointers for new data tree (We save and sum vector
        counts, to compute vector pointers, because
        the vectors of a node in a data tree must be
        contiguous).

        Now copy vectors from old data tree(s) to new
        data tree.

        'Add in' means and covariances of each node of all
        but the first data tree (that has already been taken
        care of by a previous copy).

        Create TI and TV headers for new tree.

    ENDIF

    END
```

FILES:

old TI - tree information	Instrumentation
old TV - tree vector	OLPARS option
new TI - tree information	VECIDS - resequenced
new TV - tree vector	vector ids.
TL - tree list file	
HS - history	

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

ADUPCM, CLOSFX, CMGOTH, CREATR, EQUALA, ERASE, FILPUT,
INSINI, INSRET, LGLTRE, MAKCMB, MAKMRG, MENU, OEXIT,
OPENFX, OPENS, PROMPT, TLSRCH, TRMGET, TRMPUT

HISTORY:

designed by Steven Haehn September 24, 1978

programmed by Steven Haehn August 31, 1981

OLPARS Program Specifications
MAKOPT

PROGRAM NAME: MAKOPT

CATEGORY: OLPARS Programmer Aids (system dependent)

PROGRAM DESCRIPTION:

MAKOPT creates the OLPARS option file (OPTION.OLP) within the OLPARS directory. THIS PROGRAM SHOULD BE USED ONLY BY OLPARS PROGRAMMERS/MAINTAINERS TO CREATE THE PROPER OPTION FILE.

PROGRAM USAGE:

An OLPARS programmer/maintainer places him(her)self in the OLPARS directory and types 'RUN MAKOPT'.

ALGORITHM / NOTES:

There are 2 'option' files opened by MAKOPT. They are the text input (OPTION.TXT) file and the OLPARS option output (OPTION.OLP) file.

(IF FOR ANY REASON the number of maximum programs (MAXPGM parameter) is altered, then the programs 'GETOPT' and 'SETOPT' must be recompiled with the new MAXPGM parameter.)

The format of the source text is:

PGM. NAME : OPT. # : MENU # : [OPT1] ... [OPT15] /

([ ] means optional)

The program name refers to the OLPARS command. The maximum length of a program name allowed in OPTION.OLP (OPTION.TXT, too) is 10 characters.

The option number is an arbitrarily assigned number.

NOTE: Once the option number is assigned to a program, that program should ALWAYS retain the same option number. (See Appendix A section 3.2 of OLPARS final report.)

The menu number represents the different possible menus a single OLPARS command might have.

The option list is a list of OLPARS commands that may be used after 'PROGRAM NAME' has completed computations.

The command names in the option list must be separated by blanks or tabs. The last option must be followed by a space or tab and a slash (/) character.

Comments are denoted by a ';' in column one of the input record.

FILES:

OPTION.OLP	(output)
OPTION.TXT	(input)
HS	history
INSTRU.OLP	instrumentation

REFERENCED BY:

OLPARS programmer or system maintainer

SUBPROGRAMS CALLED:

CONCAT, EQUALS, FILGET, FLUSHF, FPUT, INSCHR, INSINI, INSINT, MAX (FORTRAN FUNC.), OEXIT, OPENBL, PSORT, TRMPUT

HISTORY:

designed by Steven Haehn October 25, 1978

programmed by Steven Haehn Aug. 20, 1979

PROGRAM NAME: MARK

CATEGORY: Terminal I/O (graphic, system dependent)

PROGRAM DESCRIPTION:

MARK places a given marker at screen coordinates x,y. The marker codes (i.e., values of MARKER) can be:

1 for plus sign
2 for cross
3 for triangle
4 for square
5 for dot

To obtain ascii character, the decimal number representing the ascii code should be used. INDIC is ignored, but presently for compatibility with Tektronix PLTMAP routines.

PROGRAM USAGE:

```
CALL MARK(X,Y,MARKER,INDIC)
```

INPUT ARGUMENTS:

```
     X - INTEGER; the X screen coordinate
     Y - INTEGER; the Y screen coordinate
MARKER - INTEGER; the marker code
 INDIC - INTEGER; ignored
```

HISTORY:

designed by Steve Haehn April 6, 1978

programmed by Steve Haehn April 1, 1981

PROGRAM NAME: MATRIX

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

MATRIX enables the user to save or delete transformation matrices from the saved transformation matrix (SM) file or to obtain information about that file.

PROGRAM USAGE:

User types in 'MATRIX'.

USER INTERACTION:

MATRIX gives the user the five following options (with additional interactions for each option, if necessary).

1. Delete a saved matrix. (Additional interaction: Enter the name of the matrix to be deleted.)

2. Print a line printer listing of one or all matrices in the SM file. (Additional interaction: Enter the name of the matrix(s) to be listed.)

3. Dump at the terminal one or all matrices in the SM file. (Additional interaction: Enter the name of the matrix(s) to be dumped.)

4. List at the terminal the names, number of row vectors, and dimensions of all saved matrices in the SM file. (Additional interaction: none).

5. Enter a user-supplied matrix. (Additional interaction: Enter the name, type, dimension, and matrix to be saved in the SM file.)

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Display option list (TRMPUT).

```
        REPEAT

           Request option number (PROMPT, TRMGET).

           IF (option is to QUIT) THEN

              Exit program.

           ELSEIF (option is DELETE) THEN

              Delete the matrix(s) specified (MTRXDE).

           ELSEIF (option is PRINT) THEN

              Print the matrix(s) specified (MTRXDP).

           ELSEIF (option is DUMP) THEN

              Dump the matrix(s) specified (MTRXDP).

           ELSEIF (option is LIST) THEN

              List the saved matrix descriptions (MTRXLI).

           ELSEIF (option is ENTER) THEN

              Enter a matrix to be saved (MTRXEN).

           ENDIF

        UNTIL (user asks to quit)

        END
```

FILES:

```
        CM - communication file
        HS - history file
        SM - saved transformation matrix file
        instrumentation file
        option file
```

SUBPROGRAMS REFERENCED:

```
        CMGOTH, EQUALA, ERASE, INSINI, INSRET, MENU,
        MTRXDE, MTRXDP, MTRXEN, MTRXLI, OEXIT, OPENFX,
        PROMPT, TRMGET, TRMPUT
```

HISTORY:

designed by David Birnbaum September 30, 1978

programmed by Jill King March 16, 1981

PROGRAM NAME: MATXFRM

CATEGORY: Transformation Command

PROGRAM DESCRIPTION:

MATXFRM asks the user for the name of a saved transformation matrix stored in the saved transformation matrix (SM) file. It then transforms the current data set into a new tree using that saved matrix.

PROGRAM USAGE:

User types in 'MATXFRM'.

USER INTERACTION:

The user is asked for the name of a saved transformation matrix and a new tree name.

ALGORITHM / NOTES:

Erase the screen and the home cursor (ERASE).

Make sure the current data set is not in excess measurement mode (CHKEXS).

REPEAT

Ask the user for the name of the saved transformation matrix to be used in computations (PROMPT).

DO i = 1,MAXMTR

Retrieve the i-th matrix description (SMGMDS).

IF (this is the specified matrix) THEN

IF (dimen. of current data set is not equal to dimension of specified matrix) THEN

Print error message and reprompt.

ELSEIF (the specified matrix is an eigenvector transformation matrix using only one eigenvalue) THEN

```
                    Print message informing user that the
                    transformed matrix will be a simple
                    scalar value.

                 ENDIF

              ELSEIF (this is last time through loop) THEN

                 Print error message and reprompt.

              ENDIF

           ENDDO

        UNTIL (transformation matrix has been chosen)

        Retrieve specified matrix from the SM file (SMGENT).

        Ask user for name of the new data tree (UIXFRM).

        Perform the matrix transformation (EXFRM,NXFRM).

        Put up option list at user's terminal (MENU).

        END
```

FILES:

```
        CM - communications file
        HS - history file
        SM - saved transformation matrix file
        TI - tree information file
        TL - tree list file
        TV - tree vector file
        instrumentation file
        option file
```

REFERENCED BY:

        OLPARS user

SUBPROGRAMS REFERENCED:

        CHARFL, CHKEXS, CLOSFX, CMGCDS, CMGOTH, EQUALA,
        ERASE, EXFRM, INSFLT, INSINI, OPENFX, PROMPT,
        SMGENT, SMGMDS, TRMGET, TRMPUT, UIXFRM

HISTORY:

        designed by David Birnbaum September 30, 1978

        programmed by Jill King April 22, 1981

PROGRAM NAME: MEAN1

CATEGORY: Terminal Display Routine

PROGRAM DESCRIPTION:

MEAN1 displays the set of mean vectors for the classes that are being displayed on a one-space plot.

PROGRAM USAGE:

```
CALL MEAN1(FDDI,FDDV,SCREN)
```

INPUT ARGUMENTS:

FDDI - INTEGER; the file descriptor of the DI file

FDDV - INTEGER; the file descriptor of the DV file

SCREN - INTEGER array (NOSCRN); the one-space screen parameters

ALGORITHM / NOTES:

```
Get the number of bins in the display and the number of
classes in the DI file (DIGHDI).

DO i=1, the number of classes

    Get the  DI file entry for the next class (DIGENT).

    IF (this class is to be projected) THEN

        Get the screen coordinate (bin number) of the
        mean vector (DVGVEC).

        IF (the bin number is > 0) THEN

            Adjust the screen coordinates to reflect the
            actual coordinates instead of the bin number.

            Move the cursor to the position at which the
            class symbol is to appear (MOVE).

            Display the class symbol at the bottom of the
            proper bin (TRMPUT).

        ENDIF
```

```
            ENDIF

        ENDDO

        RETURN

        END
```

FILES:

DI - display information file
DV - display vector file
instrumentation file

REFERENCED BY:

PROJMN

SUBPROGRAMS REFERENCED:

DIGENT, DIGHDI, DVGVEC, INSINT, INSPGM, INSRET, MOVE, TRMPUT

HISTORY:

designed by David Birnbaum July 18, 1978

programmed by Jill King January 1, 1981

PROGRAM NAME: MEAN2

CATEGORY: Terminal Display Routine

PROGRAM DESCRIPTION:

MEAN2 displays the set of mean vectors for the classes that are being displayed on a two-space plot.

PROGRAM USAGE:

```
CALL MEAN2(FDDI,FDDV,SCREN)
```

INPUT ARGUMENTS:

FDDI - INTEGER; the file descriptor of the DI file

FDDV - INTEGER; the file descriptor of the DV file

SCREN - INTEGER array (NOSCRN); the two-space screen parameters

ALGORITHM / NOTES:

```
Get the number of classes in the DI file (DIGHDI).

DO i=1, the number of classes

    Get the DI file entry for the next class (DIGENT).

    IF (this  class is to be projected) THEN

        Get the screen coordinates of the mean vector
        (DVGVEC).

        IF (the coordinates are > 0) THEN

            Display the class symbol at those screen
            coordinates (MARK).

            Draw a rectangle around the mean vector
            symbol (RTANGL).

        ENDIF
```

```
            ENDIF

        ENDDO

        RETURN

        END
```

FILES:

DI - display information file
DV - display vector file
instrumentation file

REFERENCED BY:

PROJMN

SUBPROGRAMS REFERENCED:

DIGENT, DIGHDI, DVGVEC, INSINT, INSPGM, INSRET, MARK, RTANGL

HISTORY:

designed by David Birnbaum July 18, 1978

programmed by Jill King January 1,1981

PROGRAM NAME: MEASXFRM

CATEGORY: Transformation Command (system dependent)

PROGRAM DESCRIPTION:

MEASXFRM gives the user the capability of transforming a given data set into a new data set. This transformation takes the form of FORTRAN statements which are a function of the measurements from the original data set (e.g., NM(1) = OM(1) + OM(2) states that measurement 1 of the new data set equals the sum of measurements 1 and 2 of the old data set). The user must create a FORTRAN subroutine called XFORM using the following template:

```
      SUBROUTINE XFORM (OM, OLNGTH, NM, NLNGTH)
C=====================================================
C THIS IS A MEASUREMENT TRANSFORMATION SUBROUTINE
C USED BY THE OLPARS 'MEASXFRM' COMMAND.
C
C (NOTE, WHEREVER THE SYMBOL OF TWO QUESTION MARKS
C  APPEAR, THE DIMENSION OF THE NEW VECTOR MUST BE
C  SUPPLIED BY THE USER.)
C=====================================================
      INTEGER OLNGTH, NLNGTH
      REAL    OM(OLNGTH), NM(??)
C
C       (( TELL MEASXFRM THE SIZE OF THE NEW DATA SET )))
        IF(NLNGTH .GT. 0) GOTO 1000
           NLNGTH = ??
           RETURN
C       ENDIF
1000    CONTINUE
C
C       ** USER SUPPLIED VECTOR TRANSFORMATION
C          SHOULD APPEAR BELOW **
C
                '
                '
                '
        transformation statements
                '
                '
                '
C
C       ** USER SUPPLIED VECTOR TRANSFORMATION
C          SHOULD APPEAR ABOVE **
C
        RETURN
        END
```

==========

where

OM represents the old measurement vector of length OLNGTH, and

NM represents the new measurement vector of length NLNGTH.

NLNGTH is initially an output variable (tells MEASXFRM the size of the new vectors). On all subsequent calls to XFORM; OM, OLNGTH, and NLNGTH are considered input variables, while NM is an output variable of XFORM.

PROGRAM USAGE:

User types in 'MEASXFRM'.

USER INTERACTION:

1. The user must first create a FORTRAN subroutine called XFORM. This is the subroutine that will perform the actual vector transformation.

2. Once the subroutine is compiled, the user must respond to queries asking if the compilation and program linking were successful.

3. If both compilation and linking were successful, the user will be asked to specify a new tree name for the transformed data set.

4. The user will be asked if the transformation should be saved.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Get the current data set (CMGCDS).

Ask user to enter a new treename (UIXFRM).

Create new TI and TV files (CREATR).

Set up new TI header:

Dimensionality of data set.

Zero the new entry table.

```
    Transfer the current node and make it the
    new senior node.

    Set name to ****.

    Set senior node pointer in TI header.

    Set entry table slot number of new senior node to be
    same as before.

    Node level may need adjustment.

IF (current node is a lowest node) THEN

    Transfer the vectors (transforming them using the
    subroutine XFORM).

    Set vector pointer.

ELSE

    DO WHILE (there are more nodes to transfer)

        Get next node (GNXTND).

        Transfer the TI file entry keeping the entry
        table slot number of the node the same.

        IF (node is a lowest node) THEN

            Transform and transfer vectors.

            Compute means and covariances (if the
            dimension of the data set is less than
            or equal to 50).

            Set vector pointer.

        ENDIF

    ENDDO

    Compute means and covariances for all nodes above the
    lowest nodes.

ENDIF

Put up option list (MENU).

END
```

NOTE

Under RSX-11M the MEASXFRM command is implemented via a command file. In Step 1 (of user interaction) above, an editor is initiated for the user to type in his transformation subroutine. (This step can be bypassed if a transformation subroutine already exists.) The command file then initiates the FORTRAN compiler. The command file asks the user if the compilation was successful. If it was, the command file initiates the "task builder" to link together the user's XFORM subroutine with the tree building portion of this measurement transformation function. (The algorithm above is for the tree building portion of the measurement transformation function.) Step 3 above occurs when the newly created (or previously created) task is initiated by the command file processor. Finally, the Step 4 query is issued by the command file.

FILES:

CM - communication
HS - history
TI - tree information (2)
TV - tree vector (2)
instrumentation
OLPARS options

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

ADUPCM, CLOSFX, CLOSTR, CMGCDS, CMGOTH, CREATR, ERASE, GNXTND, INSCHR, INSFLT, INSINI, INSINT, INSRET, MENU, OEXIT, OPENFX, OPENTR, TIGCAP, TIGCOP, TIGET, TIPCAP, TIPCOP, TIPCOV, TIPET, TIPHDR, TIPMN, TIPNAM, TRMPUT, TVGVEC, TVPHDR, TVPVEC, UIXFRM, XFORM

HISTORY:

designed by Steven Haehn September 11, 1978

programmed by Steven Haehn August 14, 1981

PROGRAM NAME: MENU

CATEGORY: Terminal Display Routine (system dependent)

PROGRAM DESCRIPTION:

MENU displays at the user's terminal a set of options available to the OLPARS user. When the SETOPT flag is is set to 'true', MENU uses the program name argument to determine the option number to be placed in the CM file. It then uses the option number and menu number to determine which option list to display at the user terminal. When the SETOPT flag is 'false', MENU determines which option list to display by using the option number found in the CM file and the given menu number. The option number in the CM file is not changed in this case.

PROGRAM USAGE:

```
CALL MENU(FIDCM,PGMNAM,MENUNO,SETOP)
```

INPUT ARGUMENTS:

FIDCM - INTEGER; file descriptor of the CM file

PGMNAM - LOGICAL *1 (10); the program name of the calling program

MENUNO - INTEGER; menu number of the option list (menu) to be displayed

SETOP - LOGICAL; set-the-option-number-in-the-CM-file flag

FILES:

CM - communications
OLPARS option file (OPTION.OLP)

REFERENCED BY:

All OLPARS commands

SUBPROGRAMS REFERENCED:

CMGCDS,CMGOTH,CMGSCN,GETOPT,SETOPT,TEXT

OLPARS Program Specifications
MENU

HISTORY:

designed by Steve Haehn September 21, 1978

programmed by Donna Morris August 31,1979

PROGRAM NAME: MICMAC

CATEGORY: Terminal Display Routine

PROGRAM DESCRIPTION:

MICMAC produces a one-space display. It assumes that the vectors have already been projected onto a projection vector and are stored in the DV file. MICMAC determines whether a macro or micro plot is to be displayed.

PROGRAM USAGE:

```
CALL MICMAC(FDCM,FDDI,FDDV,FDPV,FDS1,CFLAG,PMODE,
            PGMNAM)
```

INPUT ARGUMENTS:

FDCM - INTEGER; the file descriptor of the CM file

FDDI - INTEGER; the file descriptor of the DI file

FDDV - INTEGER; the file descriptor of the DV file

FDPV - INTEGER; the file descriptor of the PV file

FDS1 - INTEGER; the file descriptor of the S1 file

CFLAG - INTEGER; 0 indicates no change in type of plot, 1 indicates change from macro to micro or vise versa

PMODE - INTEGER; OLPARS 'short' or 'long' prompt flag (used only in macro plots containing multiple pages)

PGMNAM - INTEGER array (10); Name to be placed under 'CURRENT OPTION:' on display. (NOTE, when PGMNAM(1) = 0 the current option name found in the CM file is used instead.)

ALGORITHM / NOTES:

```
Erase screen (ERASE).
Get screen coordinate information (CMGSCN).
Obtain display code (DC) from DI header (DIGHDI).
IF (DC = 0, i.e., this display is brand new) THEN
    IF (# of classes in DI header <= 3) THEN
        Create micro plot.
        Set DC = 6.
    ELSE
        Create macro plot.
        Set DC = 5.
    ENDIF
ELSEIF (DC = 6 and CF = 0) THEN
    Create micro plot.
ELSEIF (DC = 6 and CF = 1) THEN
    Create macro plot.
    Set DC = 5.
ELSEIF (DC = 5 and CF = 0) THEN
    Create macro plot.
ELSEIF (DC = 5 and CF = 1) THEN
    Create micro plot.
    Set DC = 6.
ELSE
    Send error message to user.
ENDIF
```

Write DC out to DI header.

Retrieve the projection vector pointer, the scale type flag (indicating counts or probabilities), zoom flag, and the number of bins displayed from the DI header.

Show user:

classes being displayed (DISPCL),
name of plot displayed (micro or macro plot),
range of display values,
mode of scaling (zoom or global),
type of scaling (counts or probabilities),
bin size and number of bins.

IF (multiple pages are to be displayed,
(MACRO plots only)) THEN

Perform 'Show user' for each page and include current option and current data set.

ENDIF

RETURN

FILES:

CM - communications
DI - display information
DV - display value
PV - projection vector
S1 - Scratch 1

REFERENCED BY:

All one-space structure analysis and logic design routines, along with various display utility programs.

SUBPROGRAMS REFERENCED:

CMGSCN, DIGHDI, DIGMAX, DIPHDI, DISPCL, DVPHDR, EQUALA, ERASE, INSINT, INSPGM, INSRET, MACROP, MICROP, PROMPT, PVPHDR, TEXT, TRMGET, TRMPUT, WRTNOW, WRTODS

HISTORY:

designed by David Birnbaum July 20, 1978

programmed by Steven Haehn Oct. 16, 1980

PROGRAM NAME: MICROP

CATEGORY: Terminal Display Routine

PROGRAM DESCRIPTION:

MICROP produces a one-space MICRO plot. MICROP computes screen coordinates, if necessary, and then displays the MICRO plot.

PROGRAM USAGE:

CALL MICROP(FDCM,FDDI,FDDV,FDS1,SCRN)

INPUT ARGUMENTS:

FDCM - INTEGER; file descriptor of the CM file

FDDI - INTEGER; file descriptor of the DI file

FDDV - INTEGER; file descriptor of the DV file

FDS1 - INTEGER; file descriptor of the S1 file

SCRN - INTEGER array (19); the OLPARS screen coordinates

ALGORITHM / NOTES:

Determine which bins the vector of the given data set fall (CNT1SP)

Determine vertical scale on display.

```
        WHILE (there are classes to be displayed)
            Retrieve next class entry from the DI file.
            IF (class should be displayed) THEN
                IF (probabilities are requested) THEN
                    Convert bin counts to probabilities.
                    Display class bin probabilities.
                ELSE
                    Display class bin counts.
                ENDIF
            ENDIF
        ENDWHILE
        IF (probabilities are requested) THEN
            Label probability bin heights.
        ELSE
            Label vector count bin heights.
        ENDIF
        RETURN
```

FILES:

CM - communications
DI - display information
DV - display value
S1 - scratch 1

REFERENCED BY:

MICMAC

OLPARS Program Specifications
MICROP

SUBPROGRAMS REFERENCED:

CMGOTH, CNT1SP, DIGENT, DIGHDI, DIGMAX, DIPHDI,
DVGHDR, DVPHDR, INSINT, INSPGM, INSRET, LINSEG,
MARK, S1GBC, TEXT, TRMPUT

HISTORY:

designed by David Birnbaum July 21, 1978

programmed by Steven Haehn Oct. 17, 1980

PROGRAM NAME: MODDFS

CATEGORY: Utilities Subroutine

PROGRAM DESCRIPTION:

MODDFS modifies the current display flag symbol for the classes entered. If the old display flag symbol was a blank, the class was not used in a previous projection, and so the new display flag symbol will replace the old. Else if the old symbol was not a blank, the class was used in multiple projections, and so the display flag substitute symbol will replace the old symbol.

PROGRAM USAGE:

```
CALL MODDFS (FDDI,FDTI,ET,NCLASS,NOSYMB,CLSMOD,
             DFSYMB,DFSUBS,DSPTYP)
```

INPUT ARGUMENTS:

FDDI - INTEGER; the file descriptor of the PV file

FDTI - INTEGER; the file descriptor of the TI file

ET - INTEGER array (TABSIZ); the TI file entry table

NCLASS - INTEGER; the total number of classes at the node

NOSYMB - INTEGER; the number of classes whose display flag symbol is to be modified

CLSMOD - INTEGER array (NOSYMB); the TI file slot numbers of the classes whose display flag symbol is to be modified

DFSYMB - INTEGER; the new display flag symbol indicating singular use of this class for projections

DFSUBS - INTEGER; the substitute display flag symbol indicating multiple use of this class for projections

DSPTYP - INTEGER; the display type - 1 for one-space
- 2 for two-space

ALGORITHM / NOTES:

```
        DO I=1,NoSymb

           Get the name of the Ith class from the TI file
           (TIGNAM).

           DO J=1,Nclass

              Get the name of the Jth class from the DI file
              (DIGENT).

              IF (the names are the same) THEN

                 Get the old display flag symbol (DIGEFS).

                 IF (the old symbol is a blank) THEN

                    Replace the old display flag symbol with the
                    new display flag symbol (DIPEFS).

                 ELSE

                    Replace the old display flag symbol with the
                    display flag substitute symbol (DIPEFS).

                 ENDIF

                 Break

              ENDIF

           ENDDO

           IF (the class name was not found in the DI file) THEN

              Display an error message.

              Goto BYEBYE.

           ENDIF

        ENDDO

        RETURN
```

FILES:

DI - display information file
TI - tree information file
instrumentation file

REFERENCED BY:

L2FSHP, S2FSHP

SUBPROGRAMS REFERENCED:

DIGEFS, DIGENT, DIPEFS, EQUALA, INSPGM, INSRET, OEXIT, TIGNAM, TRMPUT

HISTORY:

designed by Jill King December 12, 1980

programmed by Jill King December 12, 1980

PROGRAM NAME: MODINI

CATEGORY: Logic Design Routine (system dependent)

PROGRAM DESCRIPTION:

MODINI can be called by logic design modification commands to check the current data set and current logic and to obtain necessary information from the user. If there is more than one logic node of a particular requested type, it asks the user to specify which node is to be used.

PROGRAM USAGE:

```
LOGICAL MODINI
   .
   .
   .
IF (MODINI(FDCM,COMAND,NODTYP,NODNUM,CLIST,NOCLS,ENTAB,
          FDLI,FDLV,FDTI,FDTV,LOGNAM))
```

INPUT ARGUMENTS:

FDCM - INTEGER; the file descriptor of the CM file

COMAND - LOGICAL *1 array(CMDLEN); the name of the calling command

NODTYP - INTEGER; code representing the type of logic that is to be modified

OUTPUT ARGUMENTS:

NODNUM - INTEGER; the logic node number on which to operate

CLIST - INTEGER array(MAXCLS); the list of classes (slot numbers) that lie at logic node 'NODNUM'

NOCLS - INTEGER; number of classes in 'CLIST'

ENTAB - INTEGER array(TABSIZ); the entry table of the current data set

FDLI - INTEGER; logic information file descriptor

FDLV - INTEGER; logic value file descriptor

FDTI - INTEGER; tree information file descriptor

FDTV - INTEGER; tree value file descriptor

LOGNAM - INTEGER array(TRELEN); most recent logic evaluated on current data set

USER INTERACTION:

User is asked for a logic node number (if there is more than one node of the specified type).

ALGORITHM / NOTES:

MODINI = .FALSE.

Get current logic from CM file (CMGLOG).

Get current data set from CM file (CMGCDS).

Open logic and data set files (OPENTR).

IF (current logic or current data set does not exists) THEN

    Print error message to user.

ELSEIF (design set from current logic is not the same as current data set) THEN

    Print error message to user.

ELSEIF (last logic run on the data set was not the current logic (TVGHDR)) THEN

    Print error message - tell user that the last logic applied to the data set was not the current logic. In order to proceed, the user must evaluate the current data set by the current logic.

ELSEIF (there has been a change in the data as denoted in the TV header) THEN

    Print error message - tell user logic can no longer be used on that data set due to a change in the data set.

```
        ELSE

            Display a list of logic nodes (GETLST).

            REPEAT

                Prompt for an  logic node number.

            UNTIL (a correct  logic node number is
            entered or user quits).

            Get a list of classes at the requested logic node
            (GCLIST).

        ENDIF

        RETURN
```

FILES:

```
        CM - communications
        LI - logic information
        LV - logic value
        TI - tree information
        TV - tree vector
```

REFERENCED BY:

```
        NNMOD
```

SUBPROGRAMS REFERENCED:

DIAGNOSTICS:

```
        If MODINI = .FALSE., the calling program quits.
```

HISTORY:

```
        designed by  Steven Haehn    Dec. 2, 1981

        programmed by  Steven Haehn  Dec. 2, 1981
```

PROGRAM NAME: MOVE

CATEGORY: TERMINAL I/O (graphics)

PROGRAM DESCRIPTION:

MOVE moves the cursor to the input x and y coordinates.

PROGRAM USAGE:

CALL MOVE(XCOORD,YCOORD)

INPUT ARGUMENTS:

XCOORD - INTEGER; the x coordinate of the position at which the cursor is to be placed

YCOORD - INTEGER; the y coordinate of the position at which the cursor is to be placed

ALGORITHM / NOTES:

Convert the input x and y coordinates to real plot10 coordinates.

IF (the plot10 coordinates lie within the range of the terminal coordinates) THEN

Convert the real plot10 coordinates into integers.

ENDIF

Move the cursor (TEKTXT).

RETURN

END

REFERENCED BY:

PROJMN

SUBPROGRAMS REFERENCED:

TEKTXT

OLPARS Program Specifications
MOVE

HISTORY:

designed by Jill King January 7, 1981

programmed by Jill King January 7, 1981

PROGRAM NAME: MOVEC

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

MOVEC permits the user to reorient data vectors by moving their projected points on a two-space coordinate vector scatter display, which is currently displayed at the terminal.

This routine is intended for use with trees which contain nearest neighbor reference patterns.

PROGRAM USAGE:

Called by OLPARS user.

USER INTERACTION:

User identifies the vector to be moved by placing the graphics cursor 'over' the corresponding display character and typing in any alphabetic character (except 'Q' or 'R').

After the vector is identified, the user moves the graphics cursor to the position the vector is to be placed, and types in another alphabetic character (except 'Q').

The character 'Q', typed after the first prompt, terminates (quits) the process. The character 'R' redisplays the current display.

The character 'Q', typed after the second prompt, sends the program back to the first prompt to start vector selection over (i.e., the current vector chosen is not the one to be moved). The character 'R', typed after the second prompt, places the vector at its new location and redisplays the current display.

ALGORITHM / NOTES:

All the projected points (except the projected means) found in the display vector file are updated to reflect vector movement.

Since only two measurements are altered with MOVEC, only those statistics incorporating the specified measurements are updated. Consequently, two measurements in the mean vector are modified, and two 'rows' and two 'columns' of the covariance matrix are modified.

To modify the existing mean, we must subtract out the measurements of the user chosen vector and add back in the measurements representing the vector's new position. Note, the total number of vectors remains constant. Therefore,

```
NewMn(X) = OldMn(X) + (NewVec(X)-OldVec(X))/ totVec
```

Where,
- NewMn(X) - is the new mean for measurement X
- OldMn(X) - is the old mean for measurement X
- NewVec(X) - is the new measurement X
- OldVec(X) - is the old measurement X
- totVec - is the total number of vectors in the class

We need to perform a similar operation to modify the existing class covariances. The following equation shows how to obtain a new covarinace element for measurements X and Y.

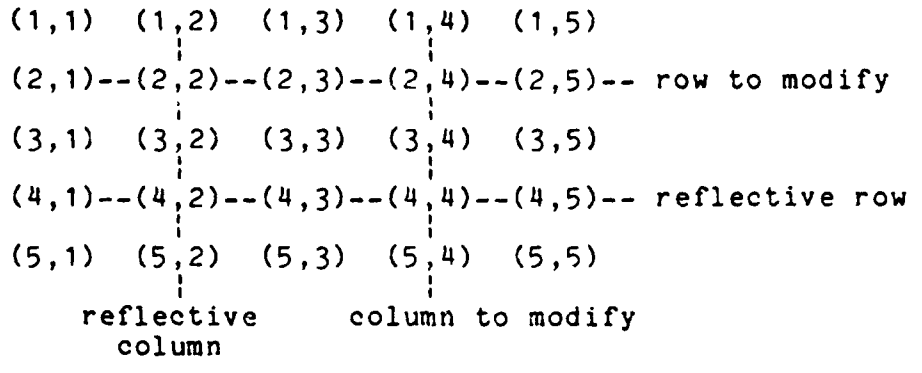
```
NewCov(XY) =

   OldCov(XY) - OldVec(X)*OldVec(Y)/totVec
              + OldMn (X)*OldMn (Y)
              + NewVec(X)*NewVec(Y)/totVec
              - NewMn (X)*NewMn (Y)
```

Remember, the entire covariance matrix is not recomputed. The effect of the vector movement ripples along the column and row of the matrix corresponding to the projected measurements. For instance, suppose the current data set has five measurements per vector and the projected measurements are two and four. The column and row to be modified can be seen in the following figure.

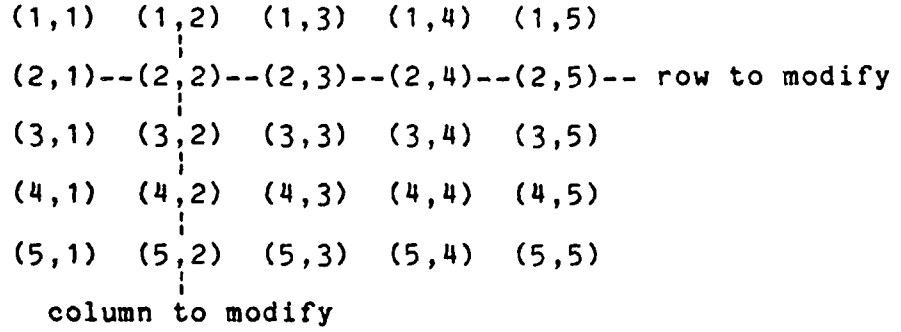
```
(1,1)  (1,2)  (1,3)  (1,4)  (1,5)
                       |
(2,1)--(2,2)--(2,3)--(2,4)--(2,5)-- row to modify
                       |
(3,1)  (3,2)  (3,3)  (3,4)  (3,5)
                       |
(4,1)  (4,2)  (4,3)  (4,4)  (4,5)
                       |
(5,1)  (5,2)  (5,3)  (5,4)  (5,5)
                       |
                 column to modify
```

Since the covariance matrix is a symmetric matrix, the reflective elements of the modified row and column must also be changed (see next figure).

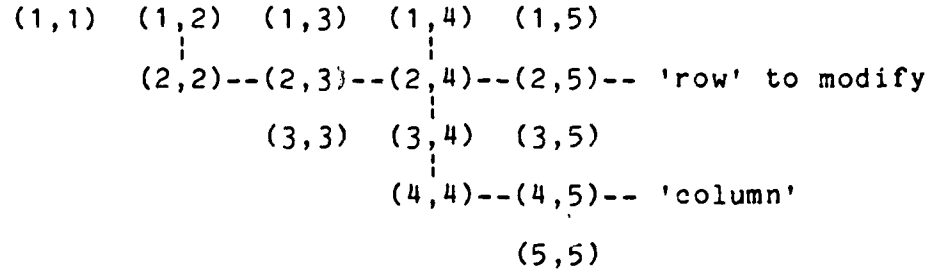


Be aware that the intersecting row-column elements (e.g. (2,2), (2,4)) should be updated only once.

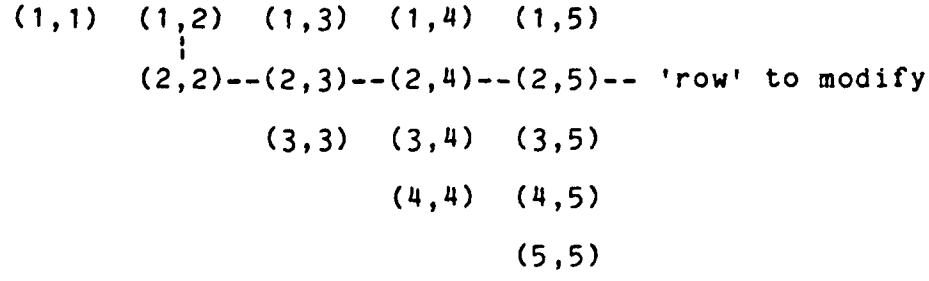
A special case occurs when the non-reflective intersecting row-column element falls on the diagonal of the covariance matrix (i.e., the X and Y projection is determined by a single measurement). In this case, the intersecting row column element is the also the reflective intersecting element. Only one row and column should be modified (see next figure).



In OLPARS, the 'upper triangular' portion covariance matrix is all that is stored because of the symmetric property of the matrix. Therefore, the total number of elements generally modified is equal to twice the dimensionality (ndim) of the matrix minus 1 (for the intersecting row-colum· element). See the next figure.



When the row-column element happens to be the diagonal element of the matrix, only 'ndim' elements are modified. See next figure.



==========

Make sure MOVEC can operate on current display.
(Was display created by coordinate projection command? Is display a scatter plot? Make sure data set is not in excess measurement mode.)

Open current data tree and ...
... get 1st lowest node pointer from senior node
... obtain data class starting vector pointers and slot numbers of lowest nodes (node list of displayed data classes).

```
REPEAT

    Tell user to choose a vector for moving.

    Get user's selected vector coordinates.
    (If given character is 'Q', exit program.
     If character is 'R', put up display again.)

    Get vectors that fall near point, and find closest
    (only look at the classes that are currently being
     displayed, and never look at mean vector found in
     display file).

    IF (vector can not be found at requested position)
    THEN

        Tell user about it and continue to next
        vector position query.

    ENDIF

    Draw rectangle around vector to be moved.

    Tell user to place the chosen vector at new location.

    IF (user did not enter 'Q') THEN

        Compute new data coordinates for screen
        coordinates received and place saved class
        symbol up on the display.

        IF (the newly chosen vector is not from the class
            of the vector chosen immediately before this
            one) THEN

            IF (this is not the very first vector chosen)
            THEN
                Replace old means and covariances in
                lowest node and all ancesteral nodes.

                Read in mean vector and covariance
                matrix of user chosen class.

            ENDIF

        ENDIF

        Read in chosen vector (the vectors in the TV
        file should correspond, in order, to those
        found in the DV file) and copy to new vector.

        Calculate new means for user supplied point.
```

```
            Adjust covariance matrix for use supplied
            coordinates.

            Save altered vector and projection point.

            IF (the user typed 'R') THEN

                Put up a new display.

            ENDIF

        ENDIF

    UNTIL (NEVER)

    IF (any vectors were moved) THEN

       Make sure class statistics reflect modifications.

    ENDIF

    END
```

FILES:

TI - tree information	HS - history
TV - tree vector	CM - communication
DI - display information	OLPARS options
DV - display value	Instrumentation
PV - projection vector	

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

ADUPCM,	BOUND,	CHKEXS,	CLOSFX,	CLOSTR,	CLSCAT,	CMGCDS,
CMGSCN,	DIGDC,	DIGENT,	DIGHDI,	DIGMAX,	DISTEU,	DVGHDR,
DVGVEC,	DVPVEC,	EQUALA,	ERASE,	GIN,	IDX,	INSINI,
INSINT,	INSRET,	MARK,	MENU,	MOVE,	OEXIT,	OPENFX,
OPENTR,	RCTNGL,	SBUPMC,	SETOPT,	TIGCOP,	TIGCOV,	TIGET,
TIGHDR,	TIGMN,	TIPCOV,	TIPMN,	TRMPUT,	TVGVEC,	TVPVEC

SEE ALSO:

NRSTNBR, S2CRDV

HISTORY:

designed by Steven Haehn December 3, 1981

programed by Steven Haehn December 22, 1981

OLPARS Program Specifications
MTRXDE

PROGRAM NAME: MTRXDE

CATEGORY: UTILITY SUBROUTINE

PROGRAM DESCRIPTION:

MTRXDE deletes a saved matrix from the Saved Transformation Matrix (SM) file.

PROGRAM USAGE:

```
INTEGER MTRXDE
     .
     .
     .
N = MTRXDE (FDSM, PMODE)
```

INPUT ARGUMENTS:

FDSM - INTEGER; the file descriptor of the SM file

PMODE - INTEGER; the prompt mode flag from the communications (CM) file

OUTPUT ARGUMENTS:

MTRXDE - INTEGER; indicates the successfulness of the program's completion

USER INTERACTION:

SEE ALGORITHM / NOTES

ALGORITHM / NOTES:

```
Retrieve the SM file header (SMGHDR).

IF (there are no saved matrices in the file) THEN

   Print error message and exit program (TRMPUT).

ENDIF

REPEAT

   Prompt the user for the name of the matrix to be
   deleted (PROMPT, TRMGET).

   IF (only one matrix is to be deleted) THEN
```

```
            DO i = 1,MAXMTR

               Retrieve the ith matrix description (SMGMDS).

               IF (the ith matrix description is the matrix
                   to be deleted) THEN

                  Set the entry number equal to i.

               ELSEIF (this is the last time through the
                   loop) THEN

                  Print message indicating that the specified
                  matrix has not been found in the file.

               ENDIF

            ENDDO

         ENDIF

      UNTIL (reply is OK)

      Adjust the header information (SMPHDR).

      Delete the matrix description (SMPMDS).

      Delete the matrix (SMPLNK).

      RETURN
```

FILES:

SM - saved transformation matrix file
intrumentation file

REFERENCED BY:

MATRIX

SUBPROGRAMS REFERENCED:

CHARFL, EQUALA, INSPGM, INSRET, LENGA, PROMPT,
SMGHDR, SMGLNK, SMGMDS, SMPHDR, SMPLNK, SMPMDS,
TRMGET, TRMPUT

HISTORY:

designed by Jill King March 16, 1981

programmed by Jill King March 16, 1981

PROGRAM NAME: MTRXDP

CATEGORY: UTILITY SUBROUTINE

PROGRAM DESCRIPTION:

Depending on the option chosen by the user, MTRXDP will either 'dump' at the terminal all or just one saved matrix from the saved transformation matrix (SM) file, or MTRXDP will 'print' at the line printer all or just one saved matrix from the SM file.

PROGRAM USAGE:

```
INTEGER MTRXDP
   .
   .
   .
N = MTRXDP (FDCM, FDSM, OPTION, PMODE)
```

INPUT ARGUMENTS:

FDCM - INTEGER; the file descriptor of the communications (CM) file

FDSM - INTEGER; the file descriptor of the SM file

OPTION - INTEGER; the option number chosen by the user; 2 for print, 3 for dump

PMODE - INTEGER; the prompt mode flag from the CM file

OUTPUT ARGUMENTS:

MTRXDP - INTEGER; indicates the successfulness of the program's completion

USER INTERACTION:

SEE ALGORITHM / NOTES

ALGORITHM / NOTES:

Make sure the SM file is not empty (SMGHDR).

Prompt the user for the name of the matrix to be dumped or printed (PROMPT, TRMGET).

```
IF (only one matrix is to be deleted) THEN

   Find the entry number of that matrix (SMGMDS).

   IF (the specified matrix has not been found in the
      file)  Print error message and exit (TRMPUT).

ENDIF

Get the screen coordinates and determine the maximum
number of values per row and the maximum number of
rows per screen allowed (CMGSCN).

DO i = 1,MAXMTR

   IF (the present matrix is to be shown) THEN

      Retrieve the matrix description (SMGMDS).

      IF (the matrix description is empty) THEN

         next

      ELSE

         Erase the screen or initialize the paper.

         Display or dump the matrix description (FILPUT).

         DO j = 1,number of vectors in matrix

            Display or dump the jth row of the matrix
            (FILPUT, SMGENT).

            IF (the cursor now lies at the bottom of the
               screen and the whole matrix has not yet
               been shown) THEN

               Ask the user if she wants to see the next
               page (PROMPT, TRMGET).

            ENDIF

            Fix the pointers for retrieval of the next
            vector.

         ENDDO

      ENDIF
```

```
            IF (that was not the last matrix to be shown) THEN

               Ask the user if she wants to see the next
               matrix (PROMPT, TRMGET).

            ENDIF

         ENDIF

      ENDDO

      IF (the 'print' option has been chosen) THEN

         Print the matrix(s) at the line printer (PRINTR).

      ENDIF

      RETURN
```

FILES:

CM - communication file
SM - saved transformation matrix file
instrumentation file

REFERENCED BY:

MATRIX

SUBPROGRAMS REFERENCED:

CHARFL, CLOSE, CMGSCN, EQUALA, ERASE, FILPUT,
INSPGM, INSRET, LENGA, OPENS, PRINTR, PROMPT,
SMGENT, SMGHDR, SMGMDS, TRMGET, TRMPUT

HISTORY:

designed by Jill King March 19,1981

programmed by Jill King March 19,1981

PROGRAM NAME: MTRXEN

CATEGORY: UTILITY SUBROUTINE

PROGRAM DESCRIPTION:

MTRXEN allows the user to enter a matrix into the Saved Transformation Matrix (SM) file. This can be done either directly from the terminal or from a file already containing the required information.

PROGRAM USAGE:

```
INTEGER MTRXEN
    .
    .
    .
N = MTRXEN (FDSM, PMODE)
```

INPUT ARGUMENTS:

FDSM - INTEGER; the file descriptor of the SM file

PMODE - INTEGER; the prompt mode flag from the communications (CM) file

OUTPUT ARGUMENTS:

MTRXEN - INTEGER; indicates the successfulness of the program's completion

USER INTERACTION:

SEE ALGORITHM / NOTES

ALGORITHM / NOTES:

Retrieve the SM file header (SMGHDR).

Make sure there is at least one saved matrix in the file.

Prompt the user for the name of the matrix to be entered (PROMPT, TRMGET).

```
Make sure the matrix name is a legal OLPARS name
and that it is unique within the SM file (IDCHK,
EQUALA, SMGMDS).

Find the first available empty location for the new
matrix description (SMGMDS).

Prompt the user for the type and dimensionality of
the matrix to be saved (PROMPT, TRMGET).

Ask the user whether the matrix will be entered from
the terminal or from a file (PROMPT, TRMGET).

IF (matrix to be entered from file) THEN

   Ask the user for the file name (PROMPT).

ELSE

   Print instructions for entering the matrix (TRMPUT).

ENDIF

DO  row = 1,MAXMTR

   IF (matrix to be entered from terminal) Ask user
      for the row vector (TRMPUT).

   DO  entry = 1,matrix dimensionality

      Retrieve the entry of the row vector (FILGET).

      IF (reply to prompt is QUIT) THEN

         IF (QUIT is not first entry of row vector) THEN

            Fill the remaining elements of the row vector
            with zeros.

         ELSEIF (QUIT is first entry of first row) THEN

            Print message telling user that no matrix has
            been saved in the file (TRMPUT).

         ELSEIF (QUIT is first entry of any but first row
            vector) THEN

            Break

         ENDIF

      ENDIF
```

```
            ENDDO

            IF (the user is still entering vectors) THEN

               Find the next available location for the new row
               vector (GNSMAE).

               Enter the vector and link pointer (SMPENT).

            ENDIF

         ENDDO

         Put the matrix description into the file (SMPMDS).

         Put the header information into the file (SMPHDR).

         RETURN
```

FILES:

SM - saved transformation matrix file
instrumentation file

REFERENCED BY:

MATRIX

SUBPROGRAMS REFERENCED:

CHARFL, CLOSE, ERASE, EQUALA, FILGET, FLUSHF,
GNSMAE, IDCHK, INSPGM, INSRET, LENGA, OPENS,
PROMPT, SMGHDR, SMGLNK, SMGMDS, SMPENT, SMPHDR,
SMPLNK, SMPMDS, TRMGET, TRMPUT

HISTORY:

designed by Jill King March 16, 1981

programmed by Jill King March 16, 1981

PROGRAM NAME: MTRXLI

CATEGORY: UTILITY SUBROUTINE

PROGRAM DESCRIPTION:

MTRXLI displays at the terminal the saved transformation matrix description information for all matrices in the saved transformation matrix (SM) file.

PROGRAM USAGE:

CALL MTRXLI (FDSM)

INPUT ARGUMENTS:

FDSM - INTEGER; the file descriptor of the SM file

FILES:

SM - saved transformation matrix file
instrumentation file

REFERENCED BY:

MATRIX

SUBPROGRAMS REFERENCED:

INSPGM, INSRET, SMGHDR, SMGMDS, TRMPUT

HISTORY:

designed by Jill King March 13, 1981

programmed by Jill King March 13, 1981

PROGRAM NAME: MXFRM

CATEGORY: TRANSFORMATION ROUTINE

PROGRAM DESCRIPTION:

MXFRM transforms a symetric matrix according to the formula below:

New Matrix A(mn) = (TvecM) * (Old Matrix A) * (TvecN)

where Old Matrix A = the input matrix to be transformed
TvecM = the m-th vector of the transformation matrix
TvecN = the n-th vector of the transformation matrix, and
New Matrix A(mn) = the mn-th element of the newly transformed matrix.

PROGRAM USAGE:

```
REAL MXFRM
  .
  .
  .
N = MXFRM(NDIM,A,TVECM,TVECN)
```

INPUT ARGUMENTS:

NDIM - INTEGER; the dimension of the transformation matrix

A - REAL array (NDIM*(NDIM+1)/2); the symetric matrix to be transformed

TVECM - REAL array (NDIM); the m-th vector of the transformation matrix to be used in the computations

TVECN - REAL array (NDIM); the n-th vector of the transformation matrix to be used in the computations

OUTPUT ARGUMENTS:

MXFRM - REAL; the mn-th element of the newly transformed matrix

OLPARS Program Specifications
MXFRM

FILES:

instrumentation file

REFERENCED BY:

EXFRM

SUBPROGRAMS REFERENCED:

INSPGM, INSRET

HISTORY:

designed by Jill King April 21, 1981

programmed by Jill King April 21, 1981

PROGRAM NAME: NAMELOG

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

NAMELOG creates a new logic tree file and makes it the 'current logic' being used.

PROGRAM USAGE:

User types in 'NAMELOG'.

USER INTERACTION:

User is asked for name of the logic tree that is to be created.

ALGORITHM / NOTES:

Erase the screen (ERASE), get the current data set information (CMGCDS), and get the prompt flag from the CM file (CMGOTH).

Open the data tree files (OPENTR), get the TI file header entry table (TIGET), get the number of low nodes, their entry table slot numbers, and class names (GLONOD).

Get the number of vectors at each lowest node(TIGCOP) and calculate the total number of vectors in the design data set.

REPEAT

Ask user for name of a new logic tree that he wants to make the current logic tree (PROMPT, TRMGET, LGLTRE)

IF (logic name already exists)THEN

Ask user if the existing logic name is to be destroyed.

ENDIF

UNTIL (the logic tree name is legal and unique).

Create the new logic tree files (CREATR) and read the LI and LV file codes from the LL file (LLSRCH).

Initialize data set name in LI header (LIPDSN).
Initialize the counts and pointers in the LI file (LIPCAP).

Put class names of the lowest nodes into the LI header (LIPCNM).

Calculate the a priori probability of a class of vectors residing at a particular data tree's lowest node.

Store probabilities in LI file header (LIPPRB).

Create the classes present bit map (EVALBM).

Create first LI entry (LIPENT).

Initialize LV file header (LVPHDR).

Set all temporary logic elements of the data tree vectors (TV file) to 1 because all vectors are considered to reside at the root logic node of the new logic tree (TIGCOP, TVGVEC, TVPVEC).

Set the latest logic information in the TV file header (TVGHDR, TVPHDR).

Write the current logic information into the CM file (CMPLOG).

END

NOTE

The a priori probability mentioned in the above algorithm is defined by:

$$\text{a priori prob.} = \frac{\text{the number of vectors in a class}}{\text{the number of vectors in the design data set}}$$

FILES:

CM - communications
HS - history (indirectly used)
LI - logic information
LL - logic list
LV - logic value
OP - option (indirectly used)
TI - tree information
TL - tree list (indirectly used)
TV - tree vector

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLOSFX, CLOSTR, CMGCDS, CMGOTH, CMPLOG, CREATR, EQUALA, ERASE, EVALBM, GLONOD, INSCHR, INSFLT, INSINI, INSINT, INSRET, LGLTRE, LIPCOP, LIPCNM, LIPDSN, LIPENT, LIPPRB, LLSRCH, LVPHDR, MENU, OEXIT, OPENFX, OPENTR, PROMPT, SETOPT, TIGCOP, TIGET, TRMGET, TRMPUT, TVGHDR, TVGVEC, TVPHDR, TVPVEC

SEE ALSO:

SETLOG

HISTORY:

designed by Steve Haehn September 14, 1978

programmed by Donna Morris July 1, 1980

OLPARS Program Specifications
NBLANK

PROGRAM NAME: NBLANK

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

NBLANK checks a character string for blanks. If all characteres are blank, the function returns true. If a non-blank character is encountered, the function returns false.

PROGRAM USAGE:

```
LOGICAL NBLANK
        .
        .
        .
IF(NBLANK(LINE,LENGTH,INDEX,NCHAR))
```

INPUT ARGUMENTS:

LINE - LOGICAL *1 ARRAY; character string to check

LENGTH - INTEGER; length of 'LINE'

INDEX - INTEGER; pointer in character string at which to start checking for blanks

NCHAR - INTEGER; number of characters to check

OUTPUT ARGUMENTS:

NBLANK - LOGICAL; 'true' if all characters are blank; 'false' if a non-blank character is encountered

REFERENCED BY:

GEN

HISTORY:

designed by Donna Morris March 16,1979

programmed by Donna Morris March 16,1979

PROGRAM NAME: NMDIST

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

NMDIST applies an NMV logic at a logic node to a vector and determines to which logic node the vector should be assigned. The function is set to the proper index into the array of logic node numbers. NMDIST also returns the set of distances for error information.

PROGRAM USAGE:

```
          INTEGER FUNCTION NMDIST

NODNUM = NMDIST(FIDLV,NDIM,VECTOR,LVNEL,DISTS,REJNOD,
                PROB,LVPTRS,IGNMS,REJCTS,DET)
```

INPUT ARGUMENTS:

FIDLV - INTEGER; the file descriptor of the LV file

NDIM - INTEGER; the dimension of the vector

VECTOR - REAL array (NDIM); the vector

LVNEL - INTEGER; the number of elements in an LV entry

PROB - REAL array (MAXCLS); the a priori probabilities

LVPTRS - INTEGER array (9); the LV file first entry

IGNMS - INTEGER array (MAXDIM); the ignore measurement mask array

REJCTS - REAL array (MAXCLS); the reject distances

DET - REAL array (MAXCLS); the determinants of classes

OUTPUT ARGUMENTS:

NMDIST - INTEGER; the index of the logic node number to which the vector is assigned.

DISTS - REAL array (NCLAS); the set of distances

REJNOD - LOGICAL; 'true' means the vector was assigned to the reject node.

ALGORITHM / NOTES:

```
DO I = 1 to NCLAS

    Call one of DISTEU, DISTVA, DISTMA, DISTQC, to
    compute the distance from the vector to the mean of
    the class (or the quadratic classifier discriminant
    function).

END

IF (the quadratic classifier was not chosen) THEN

    Set NMDIST = logic node number of the class cor-
        responding to the minimum distance.

ELSE

    Set NMDIST = logic node of class corresponding to the
    greatest discriminant function evaluation.

ENDIF

IF (the quadratic classifier was not chosen) THEN

    IF (reject distances are in use) THEN

        IF (the distance is greater than the reject) THEN

            Set REJNOD = .TRUE.

        ENDIF

    ENDIF

ENDIF

RETURN
```

FILES:

LV - logic value

REFERENCED BY:

PENMV, LOGEVAL

SUBPROGRAMS REFERENCED:

DISTEU, DISTMA, DISTQC, DISTVA, GTRGNR, INSPGM, INSRET

SEE ALSO:

NMV

HISTORY:

designed by John W. Tenney March 17, 1981

programmed by John W. Tenney May 25, 1981

PROGRAM NAME: NMEVAL

CATEGORY: Logic Design Command

PROGRAM DESCRIPTION:

NMEVAL does a partial evaluation of NMV logic. The logic must first be created by NMV. NMEVAL can also be used after NMVMOD.

PROGRAM USAGE:

User types in 'NMEVAL'.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Get a list of NMV logic nodes. If there is more than one, ask the user for a node number from the list.

Perform a partial evaluation of the NMV logic and display a confusion matrix (PENMV).

Put up option list at user's terminal (MENU).

END

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CHKEXS, CLOSFX, CMGCDS, CMGLOG, CMGOTH, CMPLOG, ERASE, GETLST, GTRGNI, GTSLOT, INSINI, INSRET, LIGCOP, LIGSTR, LIPCOP, MENU, OEXIT, OPENFX, OPENTR, PENMV, PROMPT, TIGET, TRMGET, TRMPUT

SEE ALSO:

NMV, NMVMOD

HISTORY:

designed by John W. Tenney Sept. 2, 1981

programmed by John W. Tenney Oct. 9, 1981

PROGRAM NAME: NMV

CATEGORY: Logic Design Command

PROGRAM DESCRIPTION:

NMV generates nearest mean vector logic based on various user-specified options.

The first three options of NMV classify an unknown vector from the feature space according to a metric which is computed from the unknown vector to the mean vectors of the classes of the design set. The decision is in favor of the class which produces the minimum value of the metric. The fourth option is the quadratic classifier, which chooses that class that minimizes a set of discriminant functions.

The user is also given the following choices:

a. Set reject distances for all classes or a specific reject distance for each class. This choice is not available for option 4.

b. Base the computations on a subset of the measurements.

c. Base the computations on all vectors in the classes that have vectors at the logic node or just those vectors that lie at the logic node.

OPTIONS:

1. The simple nearest mean vector option causes a vector to be assigned to the class whose mean vector is the closest in Euclidean distance. The metric d(i) (for class i) is

$$d(i) = \sum_{j=1}^{n} (X(j) - Mean(j))^{**}2$$

where

X = (x1,x2,...,xn) is the unknown vector

and n = number of measurements

2. The weighting vector option uses the metric

$$d(i) = \sum_{j=1}^{n} \frac{(x(j) - Mean(j))^{**2}}{Var(j,i)}$$

where

Var(i,j) = the variance of the jth feature of the ith class.

3. The Mahalonobis distance option uses the inverse covariance matrix of class i in the computation of the metric

$$d(i) = (X - Mean(i))\ InCov(i)\ (X - Mean(i))$$

where

InCov(i) = the inverse of the covariance for class i.

and

Mean(i) = the mean for class i.

4. This option is the quadratic classifier. Given the unknown feature vector X, the quadratic classifier is defined as follows. For each class i, compute the discriminant function

$$Gi(X) = -(.5)\,(X)\,(InCov(i))\,(X) + (Mean(i))\,(InCov(i))\,(X) - (.5)\,(Mean(i))\,(InCov(i))\,(Mean(i)) - (.5)\log(Det(i)) + \log(P(i))$$

$$= -(.5)\,[\,(X)\,(InCov(i))\,(X) - 2\,(X)\,(InCov(i))\,(Mean(i)) + (Mean(i))\,(InCov(i))\,(Mean(i))\,] - (.5)\log(Det(i)) + \log(P(i))$$

where

InCov(i) = the inverse covariance of class i

Det(i) = the determinant of class i

Gi(X) = the discriminate result for vector X with respect to class i.

and

P(i) = the a priori probability of class i

The vector X is assigned to that class i such that:

Gi(X) > Gj(X) for all j not equal to i

To evaluate the logic, use NMEVAL.

PROGRAM USAGE:

User types in 'NMV'.

USER INTERACTION:

User is asked to input the various NMV options via calls to subroutines.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Set up for logic design; prompt user for an incomplete node (SETUP).

Set up logic block for NMV. This includes determining which measurements and which vectors to use in the computations, computing, if necessary, the means, variances, the inverses of covariance matrices, and the determinants of the covariance matrices, and storing these quantities in the logic block. All of this is done in NMVBSU.

Prompt user for an option number and set corresponding flag in LV file (NMVOPT).

IF (option 1, 2 or 3 was chosen) THEN

    Prompt user for reject boundaries if desired, and store in logic block (NMVRJT).

ENDIF

Adjust temporary logic elements in TV file (KILLND).

Put up option list at user's terminal (MENU).

END

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CHKEXS, CMGCDS, CMGOTH, ERASE, INSINI, INSRET, KILLND, MENU, NMVBSU, NMVOPT, NMVRJT, OEXIT, OPENFX, SETUP, TRMPUT

SEE ALSO:

NMVMOD, NMEVAL

HISTORY:

designed by John W. Tenney Sept. 10, 1981

programmed by John W. Tenney Oct. 9, 1981

PROGRAM NAME: NMVBSU

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

NMVBSU sets up a NMV logic block. It determines if a user wants to ignore any measurements and fills in the corresponding NMV logic block entry (which is entry 2). It obtains node numbers for the classes and stores them. It obtains LV entries for reject distances and links them with the other entries in the proper order. It obtains entries from the statistical data, computes the data and stores the data in the entries.

PROGRAM USAGE:

```
CALL NMVBSU(FIDLI,FIDLV,FIDTI,FIDTV,NODNUM,ALLVEC,ET,
NDIM,TRUCLS,CLIST,PMODE,QFLAG,LVNEL,ENT1)
```

INPUT ARGUMENTS:

FIDLI - INTEGER; the file descriptor of the LI file

FIDLV - INTEGER; the file descriptor of the LV file

FIDTI - INTEGER; the file descriptor of the TI file

FIDTV - INTEGER; the file descriptor of the TV file

NODNUM - INTEGER; the logic node number

ALLVEC - LOGICAL; TRUE means all vectors from the classes
FALSE means only those vectors at the logic node

ET - INTEGER array (TABSIZ); the TI file slot number array.

NDIM - INTEGER; the dimensionality of the dataset

TRUCLS - INTEGER; the number of classes at the logic node

CLIST - INTEGER array (NCLAS); the list of class slot numbers at the logic node

PMODE - INTEGER; the prompt mode flag

LVNEL - INTEGER; the number of elements in an LV entry

ENT1 - INTEGER; the entry number of the LV first entry

OUTPUT ARGUMENTS:

QF - LOGICAL; 'true' means user wishes to quit

USER INTERACTION:

NMVBSU calls UIGNOR to see if the user wants to ignore measurements and if he wishes to use all vectors from the classes at a node or just those falling at the logic node

ALGORITHM / NOTES:

Get a LV entry for the first entry in the NMV logic block (GNLVAE).

Set number of classes and link element in entry one.

IF (user wishes to ignore measurements) THEN

Ask for which ones to be ignored (UIGNOR).

Store in second entry (PTRGNI).

Set ignore measurement flag in entry 1 to 1 (ignore some).

ELSE

Set ignore measurement flag in entry 1 to 0 (use all)

ENDIF

Get node numbers (LI entries) for the classes (GNLIAE).

Store these node numbers in the LV entries (PTRGNI).

Set up child entries in LI file (LIGCLP, LIGSTR, EVALBM, LIPENT).

Set up parent entry in LI file (LIGCLP, LIGSTR, EVALBM, LIPENT)

```
Allocate reject distance entries (PTRGNI).

Set link word (in entry 1 of the NMV block) to first
entry of this group.

DO WHILE (There are more classes to process).

   IF (user desires to use all vectors from the classes
   at the logic node in his computations, or the logic
   node is 1) THEN

      Retrieve the means of the classes from the TI file
      (TIGMN) and store in the NMV logic block (PTRGNR).

      Retrieve the covariance matrices (TIGCOV).

   ELSE

      Compute mean vector and covariance matrix of set of
      vectors from this class that fall at the logic node
      (LOGMNC) and store means in the LV file (PTRGNR).

   ENDIF

   Set link pointer in LV file.

   IF (measurements were ignored) collapse the covariance
   matrix (COLAPS).

   Invert the covariance matrices (INVERT,DITHER) and
   store inverse in NMV logic block (PTRGNR).

   IF (measurements were ignored) re-expand the matrix
      (EXPAND).

   Set link pointer in LV file.

   Store the determinant of the covariance matrices
   returned from INVERT in a temporary array.

ENDDO

Store the determinants in the logic block (PTRGNR).

RETURN
```

FILES:

TI - tree information
TV - tree vector
LI - logic information
LV - logic value

REFERENCED BY:

NMV

SUBPROGRAMS REFERENCED:

COLAPS, DITHER, EVALBM, EXPAND, GNLIAE, GNLVAE, INSFLT, INSINT, INSPGM, INSRET, INVERT, LIGCLP, LIGCNM, LIGCOP, LIGENT, LIGSTR, LIPENT, LOGMNC, LVPLNK, PTRGNI, PTRGNR, TIGCOV, TIGMN, TRMPUT, UIGNOR

HISTORY:

designed by John W. Tenney March 22, 1981

programmed by John W. Tenney May 25, 1981

PROGRAM NAME: NMVMOD

CATEGORY: Logic Design Command

PROGRAM DESCRIPTION:

NMVMOD is designed to modify an existing NMV logic node. The user is giving a list of nodes with nearest mean vector logic. If there are no nodes, or no logic tree, the program exits. After the user enters the node number the NMV logoc at the node is modified.

NOTE: To evaluate the logic use NMEVAL.

PROGRAM USAGE:

User types in 'NMVMOD'.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Display a list of NMV logic nodes (GETLST,TRMPUT).

Ask user to enter a NMV logic node number (PROMPT).

Save old distance option (GTRGNI).

Prompt user for an option number and set corresponding flag in LV file (NMVOPT).

IF (option 1, 2, or 3 was chosen) THEN

    Prompt user for reject boundaries, if desired, and store in logic block (NMVRJB).

    IF (user wishes to quit) save old option number (PTRGNI)

ENDIF

Fix temporary logic elements in the TV file (FXLTMP).

Put up option list at user's terminal (MENU)

END

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CHKEXS, CMGCDS, CMGLOG, CMGOTH, ERASE, GETLST, GTRGNI, GTSLOT, INSINI, INSRET, KILLND, LIGCOP, LIGSTR, MENU, NMVOPT, NMVRJT, OEXIT, OPENFX, OPENTR, PROMPT, PTRGNI, TIGET, TRMGET, TRMPUT

HISTORY:

designed by John W. Tenney Sept. 2, 1981

programmed by John W. Tenney Oct. 8, 1981

PROGRAM NAME: NMVOPT

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

NMVOPT requests the user to enter an NMV option number. It sets the weight flag in the NMV logic block accordingly. If the user wishes to quit NMVOPT is set equal to TRUE.

PROGRAM USAGE:

```
Q = NMVOPT(FIDLV,ENTNO,LVNEL,OPTNMV,PMODE)

    LOGICAL FUNCTION NMVOPT
```

INPUT ARGUMENTS:

FIDLV - INTEGER; the file descriptor of the LV file

ENTNO - INTEGER; the entry number of the first entry in the NMV logic block

LVNEL - INTEGER; the number of elements in an LV entry

PMODE - INTEGER; the prompt mode flag

OUTPUT ARGUMENTS:

OPTNMV - INTEGER; the option number chosen by the user

USER INTERACTION:

Asks user to enter an NMV option.

OLPARS Program Specifications
NMVOPT

ALGORITHM / NOTES:

NMVOPT = FALSE

Prompt user until a valid NMV option number is entered (PROMPT).

IF (user wishes to quit) THEN

Set NMVOPT = .TRUE.

ELSE

Set weight flag in first entry of NMV logic block (LVPINT).

ENDIF

RETURN

FILES:

LV - logic value

REFERENCED BY:

NMV, NMVMOD

SUBPROGRAMS REFERENCED:

GTRGNI, INSPGM, INSRET, PROMPT, PTRGNI, TRMGET, TRMPUT

DIAGNOSTICS:

If NMVOPT is set to TRUE, the user has chosen to quit.

HISTORY:

designed by David Birnbaum September 22, 1978

programmed by John Tenney Oct. 9, 1981

PROGRAM NAME: NMVRJT

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

NMVRJT asks the user if he desires reject distances for NMV logic. If so, it prompts the user for the distances and stores them in the NMV logic block.

The units the distances are stored in is dependent on the distance option chosen (NMVOPT). In the case of option 1), Euclidean, the distance is stored in the same units as the data. In the case of options 2), and 3), the distance is stored as a number of variances. This is because the distance function for these two returns a multiplier of variances (DISTVA,DISTMA). In actuality, the reject distance is a multiplier of standard deviations, however, since the square of the distances is used rather than the distance itself. This saves computational time. (Variance = Standard Dev. squared).

PROGRAM USAGE:

LOGICAL FUNCTION NMVRJT

```
Q = NMVRJT(FIDLV,ENTNO,PMODE,LVNEL,NDIM,TRUCLS)
```

INPUT ARGUMENTS:

FIDLV - INTEGER; the file descriptor of the LV file

ENTNO - INTEGER; the entry number of the first entry in the NMV logic block

PMODE - INTEGER; the prompt mode flag

LVNEL - INTEGER; the number of elements in an LV entry

NDIM - INTEGER; the dimensionality of the dataset

TRUCLS - INTEGER; the number of classes at the logic node

USER INTERACTION:

Asks user for reject distances.

ALGORITHM / NOTES:

```
NMVRJT = .FALSE.
IF (user wishes reject distances) (TRMGET).
   IF (user wishes to quit) THEN
       Set NMVRJT = .TRUE.
   ELSE
       IF (user wishes a reject distance for all classes)
           Prompt for a multiplier.
       ELSE
           Prompt for a multiplier for each class.
       ENDIF
       IF (Euclidean option was chosen) THEN
           Retrieve variances from LV block (GTRGNR).
           Save reject distance as multiplier * average
                variance.
       ELSE
           Save reject distance as multiplier.
       ENDIF
       Set reject flag in first entry to 1.
       Store reject distances (PTRGNR).
   ENDIF
ELSE
    Set reject flag to 0.
ENDIF
RETURN
```

FILES:

LV - logic value

REFERENCED BY:

NMV, NMVMOD

SUBPROGRAMS REFERENCED:

ERASE, GTRGNI, GTRGNR, INSFLT, INSINT, INSPGM, INSRET, PTRGNI, PTRGNR, TRMGET, TRMPUT

DIAGNOSTICS:

If the user wishes to quit, NMVRJT is set equal to TRUE.

HISTORY:

designed by John W. Tenney Sept. 22, 1981

programmed by John W. Tenney Oct. 9, 1981

PROGRAM NAME: NNBREV

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

NNBREV (Nearest Neighbor Evaluator) evaluates a vector against all the vectors (reference patterns) in a reference pattern tree. It returns the class display symbol associated with the reference pattern(s) closest to the input vector.

PROGRAM USAGE:

```
INTEGER NNBREV
   .
   .
CLSIDX = NNBREV(RFDTV,RPTRS,RCLSCT,NDIM,IGNOR,IGNMES,
                NNBRCT,VECTOR,REJECT,CSYMBL)
```

INPUT ARGUMENTS:

RFDTV - INTEGER; reference pattern TV file descriptor

RPTRS - INTEGER ARRAY(MAXCLS,2); vector pointers and counts of r.p. classes

RCLSCT - INTEGER; r.p. class count - number of classes in reference pattern tree

NDIM - INTEGER; number of measurements in r.p. vector

IGNOR - INTEGER; ignore measurements switch
1 - means ignore some measurements
0 - use all measurements

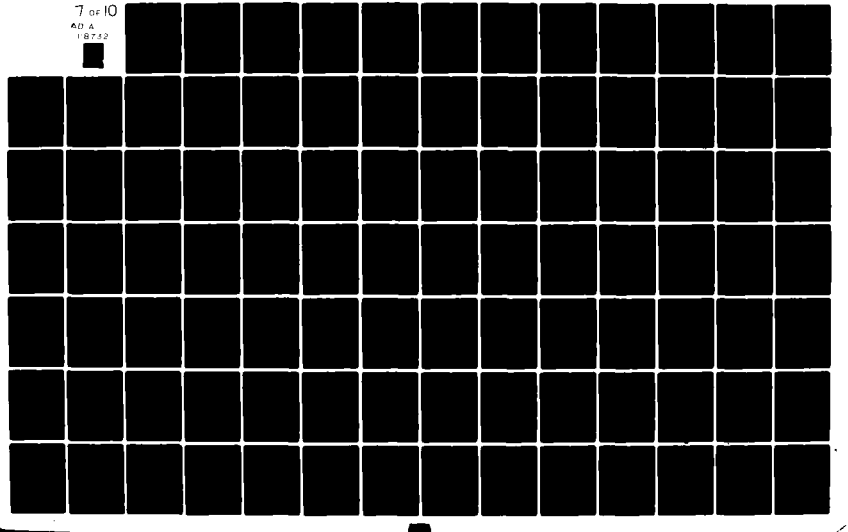
IGNMES - INTEGER ARRAY(MAXDIM); measurements-to-ignore mask vector

NNBRCT - INTEGER; nearest neighbor count - the number of nearest neighbors to choose from when classifying vector

VECTOR - REAL array (NDIM); vector to be evaluated

AD-A118 732 PAR TECHNOLOGY CORP NEW HARTFORD NY F/G 9/2
ON-LINE PATTERN ANALYSIS AND RECOGNITION SYSTEM. OLPARS VI. SOF--ETC(U)
JUN 82 S E HAEHN, D MORRIS
UNCLASSIFIED PAR-82-20 NL

7 OF 10
AD A
118732



OUTPUT ARGUMENTS:

NNBREV - INTEGER; index to the reference pattern class 'closest' to the given vector

REJECT - LOGICAL; vector-was-rejected flag (when true)

CSYMBL - INTEGER; class symbol of reference pattern 'closest' to given vector

ALGORITHM / NOTES:

```
WHILE (there are reference pattern classes)

    WHILE(there is a reference pattern in r.p. class)

        Find the distance between the reference
        pattern and the vector.

        Save the first 'NnbrCt' nearest neighbor
        distances.

        Search for furthest neighbor in saved nearest
        neighbor list.

        IF (the most recent reference pattern is closer
            to the vector than the vector's furthest
            neighbor) THEN

            Replace furthest neighbor with the
            new closer neighbor.

        ENDIF

    ENDWHILE

ENDWHILE

At this point we have the nearest neighbor(s)

IF (we are supposed to look at more than 1 nearest
    neighbor ( K nearest neighbor rule)) THEN

    To classify the vector, the reference pattern
    class which occurs most frequently must be
    found (first tally up votes).

    Now see if there is a tie vote to the maximum vote.
```

```
            IF (there are no tie votes) THEN

                The vector has been classified through the
                above vote count.

            ELSE

                A tie was found, so now break the tie by
                locating the set of reference patterns
                that are closest to the vector.

            ENDIF

        ENDIF

        RETURN
```

FILES:

TV - tree vector (reference patterns)

REFERENCED BY:

EVALU8

SUBPROGRAMS REFERENCED:

DISTEU, INSCHR, INSINT, INSPGM, INSRET, TVGVEC

HISTORY:

designed by Steven Haehn November 9, 1981

programmed by Steven Haehn November 18, 1981

PROGRAM NAME: NNLBSU

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

NNLBSU (nearest neighbor logic block set up) creates the nearest neighbor logic block in the logic value portion of an OLPARS logic tree. It also allocates and links the logic information entries that represent the children of the node at which nearest neighbor logic is being generated.

PROGRAM USAGE:

```
CALL NNLBSU(FDLI,FDLV,LOGNOD,NDIM,NCLS,TREENM,PMODE,
            ASOCND,CONTIN)
```

INPUT ARGUMENTS:

FDLI - INTEGER; descriptor for logic information file

FDLV - INTEGER; descriptor for logic value file

LOGNOD - INTEGER; logic node at which NN logic is being created

NDIM - INTEGER; number of measurements in data set vectors

NCLS - INTEGER; number of classes in data set

TREENM - INTEGER ARRAY(TRELEN); name of data tree containing reference patterns

PMODE - INTEGER; OLPARS prompt mode

OUTPUT ARGUMENTS:

ASOCND - INTEGER ARRAY(MAXCLS+2); logic nodes associated with data set classes

CONTIN - LOGICAL; continuation flag -
when true, keep going
when false, abort program

USER INTERACTION:

This program calls 'UIGNOR' to obtain the measurements to be ignored when evaluating nearest neighbor logic.

ALGORITHM / NOTES:

Ask user for measurements to ignore

Allocate a Logic Information entry (node) for each class present at the logic node, plus 1 for a reject node.

Create nearest neighbor logic entries in Logic Value file

1. (allocate NN block header, 1 entry - fill in header)
2. (create reference pattern tree name entry)
3. (create ignore measurement vector entry)
4. (create associated logic node list entry(ies))
5. (write out NN block header)

Obtain the names of the classes present in the design data set along with the indices of the classes present at the NN logic node (to be used when creating each child node of the NN logic node).

Obtain NN node structure for use in creation of a child node structure template.

Complete and write out each child logic node

Update the NN node so it 'knows' about its children and logic region.

FILES:

LI - logic information
LV - logic value

REFERENCED BY:

NRSTNBR

SUBPROGRAMS REFERENCED:

EVALBM, GNLIAE, GNLVAE, INSCHR, INSINT, INSPGM, INSRET, LIGCLP, LIGCNM, LIGCOP, LIGENT, LIGSTR, LIPENT, PTRGNI, UIGNOR

HISTORY:

designed by Steven Haehn November 9, 1981

programmed by Steven Haehn November 19, 1981

PROGRAM NAME: NNMEAN

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

NNMEAN creates a reference pattern tree using the means of the design data set classes present at the current logic node. Each reference pattern class has a complete node created for it. The class means are placed in tree vector file entries (and the mean vector areas of the tree information file). The individual class covariances are set to zero. The entire tree is created by NNMEAN except for the senior node's means and covariances (these are zeroed out).

PROGRAM USAGE:

```
CALL NNMEAN (NEWTI,NEWTV,OLDTI,OLDTV,CLIST,NCLIST,NDIM,
             LASTND,ENTBLE)
```

INPUT ARGUMENTS:

NEWTI - INTEGER; file descriptor for ref. pat. TI file

NEWTV - INTEGER; file descriptor for ref. pat. TV file

OLDTI - INTEGER; file descriptor of data TI file

OLDTV - INTEGER; file descriptor of data TV file

CLIST - INTEGER ARRAY(MAXCLS); slot numbers of classes present at logic node

NCLIST - INTEGER; number of classes present

NDIM - INTEGER; number of measurements in data vector

ENTBLE - INTEGER ARRAY(TABSIZ); entry table of data tree

OUTPUT ARGUMENTS:

LASTND - INTEGER; entry of last node in ref. pat. tree

ENTBLE - INTEGER ARRAY(TABSIZ); entry table of reference pattern tree

ALGORITHM / NOTES:

```
Create senior node of new tree
(with empty means and covs.).

WHILE (there are low nodes in the existing data tree)

   Write out new node name.
   Obtain and update counts and pointers for new node.
   Write out node's new counts and pointers.
   Write out node's means.
   Write out node's covariances.
   Write out node's vector (the mean vector).

ENDWHILE

Fix last lowest node's lowest-node-link-pointer
(LNLP of this node must indicate there are no
 more lowest nodes in tree), and sibling pointer
(SP must show there are no siblings to last node).

Create new tree's entry table.

Create new tree's header sections (TI and TV)
and make sure senior node knows about its kids.

RETURN
```

FILES:

```
TI - tree information (2)
TV - tree vector (2)
```

REFERENCED BY:

```
NRSTNBR
```

SUBPROGRAMS REFERENCED:

```
INSCHR, INSINT, INSPGM, INSRET, TIGCAP, TIGCOP, TIGHDR,
TIGMN,  TIGNAM, TIPCAP, TIPCOP, TIPCOV, TIPET,  TIPHDR,
TIPMN,  TIPNAM, TVPHDR, TVPVEC
```

HISTORY:

designed by Steven Haehn November 9, 1981

programmed by Steven Haehn November 18, 1981

PROGRAM NAME: NNMNCV

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

NNMNCV computes the means and coavariances for the reference pattern vectors found in the nearest neighbor (NN) rule 'store'. The vectors are transferred to the reference pattern tree vector file currently being created.

PROGRAM USAGE:

```
CALL NNMNCV(STORFD,RPFDTV,NDIM,FRSTVC,NXTVEC,MEAN,COV,
            NVEC)
```

INPUT ARGUMENTS:

```
STORFD - INTEGER; file descriptor of the CNN rule 'store'
RPFDTV - INTEGER; ref. pat. tree vector file descriptor
  NDIM - INTEGER; number of measurements in r.p. vector
FRSTVC - INTEGER; 'store' pointer to first vector of
                  reference pattern class
```

OUTPUT ARGUMENTS:

```
NXTVEC - INTEGER; pointer to next vector in r.p. tree
  MEAN - REAL ARRAY(MAXDIM); class mean vector
   COV - REAL ARRAY(MAXDIM*(MAXDIM+1)/2); class covs.
  NVEC - INTEGER; number of vectors in given r.p. class
```

ALGORITHM / NOTES:

Read each vector from the reference pattern 'store' (following links to obtain each vector), write each vector to the reference pattern tree vector file, and compute the 'running' means and covariances.

FILES:

'store' (see NNPASS)
TV - tree vector

REFERENCED BY:

NNSORT

SUBPROGRAMS REFERENCED:

INSFLT, INSINT, INSPGM, INSRET, TVGVEC, TVPVEC

SEE ALSO:

NNPASS

HISTORY:

designed by Steven Haehn November 9, 1981

programmed by Steven Haehn November 19, 1981

PROGRAM NAME: NNMOD

CATEGORY: Logic Design Command

PROGRAM DESCRIPTION:

NNMOD modifies nearest neighbor logic at a user chosen logic node. The user has the option of changing the 1) number of neighbors being used during evaluation, 2) the reference pattern tree to be used during evaluation, and 3) the set of measurements to be ignored during evaluation.

This command can only be used after the NRSTNBR command has created nearest neighbor logic in the current logic tree.

PROGRAM USAGE:

User types 'NNMOD'.

USER INTERACTION:

The user is asked:

1. which nearest neighbor (NN) logic node should be changed (providing there is more than one NN logic in the current logic tree).

2. for the modification option to be used

   (a) neighbor count modification - (queried for neighbor count)

   (b) use of different reference pattern tree - (queried for tree name)

   (c) measurements to be ignored - (asked if any measurements are to be ignored and for index of measurement to be ignored)

ALGORITHM / NOTES:

```
Ask user to choose which NN logic node is to be modified.
(only if more than one exists in tree).

WHILE (user has not requested the program exit)

    Ask user for modification option.

    IF (the neighbor count is to be changed) THEN

        Ask user for new neighbor count.
        Update NN logic block.

    ELSEIF (R.P. tree name is to be changed) THEN

        Ask user for name of tree to be used in
        subsequent evaluations.

        Update NN logic block.

    ELSEIF (measurements are to be ignored) THEN

        Ask user if any measurements are to be ignored.

        IF (there are any meas. to be ignored) THEN

            Ask user for index of meas. to be ignored.

        ENDIF

        Update NN logic block.

    ENDIF

ENDWHILE
```

FILES:

CM - communications	LI - logic information
HS - history	LV - logic value
TI - tree information (2)	Instrumentation
TV - tree vector (2)	OLPARS options

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLOSTR, CMGCDS, CMGOTH, ERASE, GTRGNI, INSINI, INSRET, LIGCOP, LIGSTR; MENU, MODINI, OEXIT, OPENFX, OPENTR, PROMPT, PTRGNI, TRMGET, TRMPUT, UIGNOR

SEE ALSO:

NRSTNBR

HISTORY:

designed by Steven Haehn November 30, 1981

programmed by Steven Haehn December 7, 1981

PROGRAM NAME: NNPASS

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

NNPASS cycles through all the vectors of the design tree ('grab bag') classifying them using the vectors (reference patterns) in 'store.' Those incorrectly classified are removed from 'grab bag' and added to 'store.' After a complete pass thru the 'grab bag', the user is shown how many vectors have been added to 'store' before the next pass through 'grab bag.' If no vectors have been added, the program is done processing the data tree.

PROGRAM USAGE:

```
CALL NNPASS(DSTI,DSTV,CLIST,NCLIST,NDIM,ENTBLE,ASKOK,
            STORFD,LINKS,CONTIN)
```

INPUT ARGUMENTS:

DSTI - INTEGER; descriptor of data set tree information file

DSTV - INTEGER; descriptor of data set tree vector file

CLIST - INTEGER ARRAY(MAXCLS); slot numbers of the classes present at the NN logic node

NCLIST - INTEGER; number of classes present at the NN logic node

NDIM - INTEGER; number of measurements in a data vector

ENTBLE - INTEGER ARRAY(TABSIZ); entry table of design data tree

ASKOK - LOGICAL; user interaction switch:

true - means user wants to be able to abort creation of nearest neighbor logic

false - means user does not interact with program after each pass through 'grab bag'.

OUTPUT ARGUMENTS:

STORFD - INTEGER; file descriptor of reference pattern 'store'

LINKS - INTEGER ARRAY(MAXCLS); pointers to linked reference pattern vectors

CONTIN - LOGICAL; program continuation switch:

true - means finish creating r.p. tree
false - means do not create r.p. tree

ALGORITHM / NOTES:

Create the 'store' and 'grab bag' vector areas.

Copy the tree vectors to the 'grab bag' region.

Place the last vector read into the 'store' region.

WHILE (there is a vector in the 'grab bag' and the 'store' has been modified)

Look thru the grab bag for a vector that can not be classified by the reference patterns currently found in the 'store'.

IF (the reference pattern closest to the vector does not belong to that vector's class) THEN

Move that vector from the 'grab bag' to the reference pattern 'store'.

ENDIF

Tell user about this pass's results.

Query user if requested to and there has been a change in the reference pattern 'store'.

ENDWHILE

RETURN

FILES:

TI - tree information
TV - tree vector
'store'
'grab bag'

REFERENCED BY:

NRSTNBR

SUBPROGRAMS REFERENCED:

CLOSFX, DISTEU, EQUALA, INSCHR, I SINT, INSPGM, INSRET,
OPNTMP, TIGCOP, TIGHDR, TRMGET, TRMPUT, TVGVEC, TVPVEC

SEE ALSO:

NRSTNBR

HISTORY:

designed by Steven Haehn November 9, 1981

programmed by Steven Haehn November 18, 1981

PROGRAM NAME: NNSORT

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

NNSORT sorts the reference patterns placed in the condensed nearest neighbor 'store' (really NNPASS has already performed an insertion sort; NNSORT simply 'sorts' by following sort ordered links) into class order and stores them in a tree vector file. It also computes the means and covariances of each class and stores them in the corresponding tree information file, along with the proper header entries.

PROGRAM USAGE:

```
CALL NNSORT(RPTI,RPTV,DSTI,STORFD,CLIST,NCLIST,NDIM,
            LINKS,ENTABL,LASTND)
```

INPUT ARGUMENTS:

RPTI - INTEGER; file descriptor of reference pattern tree information file

RPTV - INTEGER; file descriptor of reference pattern tree vector file

DSTI - INTEGER; file descriptor of data set tree information file

STORFD - INTEGER; file descriptor of CNN rule 'store'

CLIST - INTEGER ARRAY(MAXCLS); slot numbers of the classes present at the NN logic node

NCLIST - INTEGER; number of classes present at the NN logic node

NDIM - INTEGER; number of measurements in data vector

LINKS - INTEGER ARRAY(MAXCLS); links to first vector of each reference pattern class

ENTABL - INTEGER ARRAY(TABSIZ); data tree entry table

OUTPUT ARGUMENTS:

ENTABL - INTEGER ARRAY(TABSIZ); reference pattern tree entry table

LASTND - INTEGER; entry (and slot) number of the last node in the reference pattern tree

ALGORITHMS / NOTES:

Create senior node of new tree
(with empty means and covariances).

Pick up the first lowest node in the given data tree.

WHILE (there are low nodes in the existing data tree)

Write out new node name.

Compute the means and covariances of the reference pattern class and move the reference pattern vectors to their tree vector file.

Write out node's means and covariances.

Update counts and pointers for new node.

ENDWHILE

Fix last lowest node's lowest-node-link-pointer
(LNLP of this node must indicate there are no more lowest nodes in tree), and sibling pointer
(SP must show there are no siblings to last node).

Create new tree's entry table.

Create new tree's header sections (TI and TV)
and make sure senior node knows about its kids.

RETURN

FILES:

TI - tree information (reference pattern tree)
TV - tree vector (reference pattern tree)
TI - tree information (design data tree)
'store'

REFERENCED BY:

NRSTNBR

SUBPROGRAMS REFERENCED:

INSCHR, INSINT, INSPGM, INSRET, NNMNCV, TIGCAP, TIGCOP, TIGHDR, TIGNAM, TIPCAP, TIPCOP, TIPCOV, TIPET, TIPHDR, TIPMN, TIPNAM, TVPHDR

HISTORY:

designed by Steven Haehn November 9, 1981

programmed by Steven Haehn November 18, 1981

PROGRAM NAME: NODCOM

CATEGORY: Data Tree File Access Routine

PROGRAM DESCRIPTION:

NODCOM creates the counts and pointers, and the mean vector and covariance matrix for a TI file entry for a set of combined nodes. NODCOM follows the linked list of lowest nodes beneath the parent node and picks out those nodes that are to be combined. The counts and pointers of the new combined node are placed in the TI file entry of the first node to be combined (first in the sense of the lowest node linked list). The reason for this is that any nodes that point to the first combined node do not have to be altered. NODCOM adjusts the counts and pointers of all entries of the TI file affected by the combining. NODCOM returns the slot number of the new node in SLTNEW.

PROGRAM USAGE:

CALL NODCOM (FID, NDIM, ET, TABLE, NUM, SLTPAR, SLTNEW)

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the TI file

NDIM - INTEGER; the dimensionality of the dataset

ET - INTEGER array; the entry table

TABLE - INTEGER array; slot numbers of nodes to be combined

NUM - INTEGER; number of nodes to be combined

SLTPAR - INTEGER; slot number of the parent node

OUTPUT ARGUMENTS;

SLTNEW - INTEGER; slot number of the new node

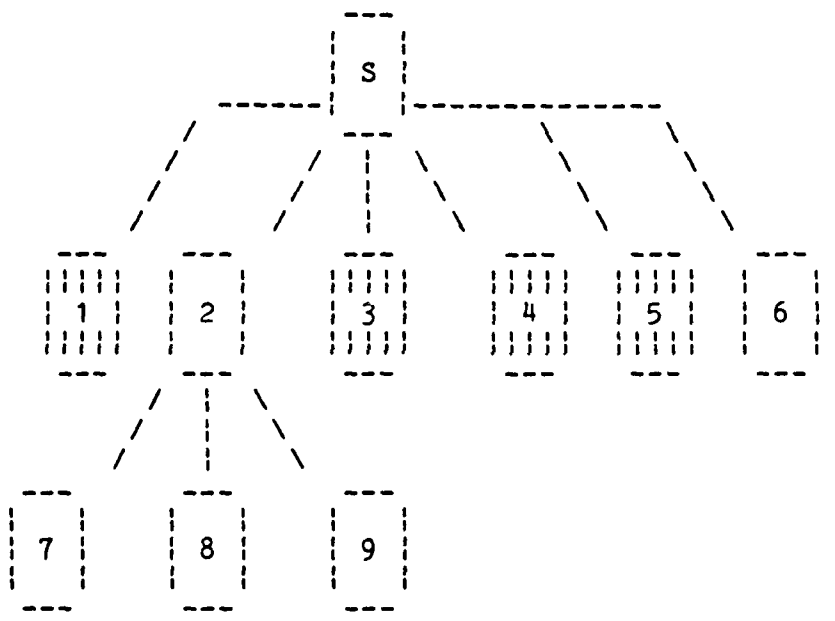
ALGORITHM / NOTES:

In order to understand the algorithm for NODCOM, we need to define two words:

Contiguous - Two siblings in a data tree are continguous if one's sibling pointer points to the other sibling.

Chain - A chain is a list N(1), ..., N(k) of nodes that are siblings and such that N(c) is contiguous with N(c+1) for c = 1, ..., k-1. In general, chains will satisfy some sort of property, e.g., a chain of nodes to be combined. When considering chains that satisfy this property, we will operate on the largest chains possible.

NODCOM works on chains of nodes to be combined. For example, in the diagram below, nodes 1, 3, 4, and 5 are to be combined. NODCOM would first operate on the chain [1] and then on the chain [3,4,5]. All the results would be placed in the entry for node 1.



IF (parent's first child is not the first node to combine) THEN

Check parent's children starting with the sibling of the first child until the first node to combine is found.

ENDIF

Get the sibling of the first node to combine.

```
WHILE (the sibling is a node to combine)

   Add the mean vector and covariance matrix of the
   sibling to the mean vector and covariance matrix
   of the first node to combine.

   IF (we still have more nodes to combine)

      Get the next sibling.

   ENDIF

ENDWHILE

IF (there are at least two nodes in the first chain) THEN

   Update the SP and LNLP of the first node in the chain:

   SP (= SP of last node in chain)
   LNLP (= LNLP of last node in chain)

   IF (last node in chain is not last lowest node) THEN

       Adjust LNBP of the lowest node following last
       node in chain (=entry slot of first node in
       chain)

   ENDIF

ENDIF

WHILE (there are more nodes to be combined)

   Get the next sibling and find the first node of the
   new chain.
   Get the sibling of the first node of the new chain.

   WHILE (the sibling is a node to combine)

      Add the mean vector and covariance matrix of the
      sibling to the mean vector and covariance matrix
      of the first node to combine.
      Get the next sibling.

   ENDWHILE


   For the sibling preceding the chain, adjust the:

      SP (= SP of last node in chain)

   For the lowest node preceding the chain, adjust the:
```

LNLP (= LNLP of last node in chain)

For the lowest node(if any) following the chain, adjust the:

LNBP (=LNBP of first node in chain)

ENDWHILE

Set SLTNEW = BPET of first node to combine

For the first node to combine, adjust the:

NOVECS (= total number of vectors combined)
VP (= 1)

Adjust the number of chidren count of the parent entry and the lowest node count of the parent entry and of the parent's ancestors' entries.

FILES:

TI - tree information

REFERENCED BY:

COMNOD

SUBPROGRAMS REFERENCED:

ADDMCV, FIXKLV, INSINT, INSLOG, INSPGM
INSRET, OEXIT, TIGCAP, TIGCOP, TIGCOV
TIGMN, TIPCAP, TIPCOP, TRMPUT

HISTORY:

designed by David Tipton August 16, 1980

programmed by David J. Tipton October 30, 1980

OLPARS Program Specifications
NORMXFRM

PROGRAM NAME: NORMXFRM

CATEGORY: Transformation Command

PROGRAM DESCRIPTION:

NORMXFRM creates a new data tree which is a normalized version of the current data set. NORMXFRM calculates the standard deviations of the entire current data set (root node) and calls 'NXFRM' to do the transformation. The normalization procedure involves dividing each measurement of each vector by the standard deviation of that measurement for the entire current data set. The variance of each measurement in the new data tree is, therefore, one.

PROGRAM USAGE:

User types in 'NORMXFRM'.

USER INTERACTION:

Prompts for a new treename. At end asks user if he wishes to save transformation matrix.

FILES:

CM - Communications
HS - History (indirectly used)
INSTRU.DAT - Instrumentation (indirectly used)
OPTION.OLP - Option (indirectly used)
SM - Saved Matrix
TI - Tree Information (current tree and new tree)
TL - Tree List
TV - Tree Vector (current tree and new tree)

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CHKEXS, CLOSFX, CLOSTR, CMGCDS, CMGOTH,
ERASE, INSINI, INSINT, INSRET, MENU, NXFRM,
OEXIT, OPENFX, OPENTR, SAVMAT, TIGET,
TIGVAR, TRMPUT, UIXFRM,

HISTORY:

designed by Dave Birnbaum September 7, 1978

programmed by Donna Morris March 1, 1981

PROGRAM NAME: NRSTNBR

CATEGORY: Logic Design Command

PROGRAM DESCRIPTION:

NRSTNBR creates a nearest neighbor logic at the specified node of the current logic tree. The resultant logic consists of a reference tree created by 1) using the entire data tree, 2) the mean vectors of a data tree, or 3) the "condensed nearest neighbor" rule of Hart. (See HART, P. E., "The Condensed Nearest Neighbor Rule," IEEE Trans. Info. Th.,vol. IT 14, pp. 515-516, May 1968.)

No partial evaluation is performed for this logic. Partial evaluations are not needed for the CNN rule or when using the entire data tree because classification is defined to be 100 percent for the design data classes present at the logic node.

For the CNN rule, statistics are presented to the user (e.g., number of passes, number of vectors in classes present at logic node, and number of reference patterns selected).

This program can be expanded or changed to incorporate additional nearest neighbor rules.

PROGRAM USAGE:

User types in 'NRSTNBR'.

USER INTERACTION:

- o Select logic node (if more than one incomplete logic node exists in the logic tree).
- o Select reference pattern tree generation rule.
- o Name reference pattern tree (if rule dictates creation of new tree)
- o If CNN rule is chosen:
  - oo Select interaction at each iteration of rule.
  - oo Abort generation of NN logic

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Ask user to select an NN rule.

Ask user for ref. pat. tree name, if needed.

(Ask if interaction is desired; CNN rule only).

Create logic node in logic tree (NNLBSU).

Create reference pattern tree if:

1. using means of data tree, or
2. using CNN rule.

Put up option list at user's terminal (MENU).

RETURN

NOTE

If the current logic node does not contain the entire set of classes found in the design data set (i.e., NN logic is not being created at the senior logic node), only the classes present at the logic node will be found in the reference pattern tree's set of classes. This is only true when the means of the design data set or the condensed nearest neighbor rule is used to generate the reference pattern tree. If the entire data tree is used, all the classes in the data tree will be there. Thus, there will be more reference pattern classes than classes present at the logic node. The only problem this may cause during evaluation is slower execution time.

FILES:

CM - communication
LI - logic information
LV - logic value
TI - tree information (2)
TV - tree vector (2)
TL - tree list

'grab bag' - temporary
'store' - temporary
Instrumentation
OLPARS option file

OLPARS Program Specifications
NRSTNBR

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

ADUPCM, CHKEXS, CLOSFX, CLOSTR, CMGCDS, CMGLOG, CMGOTH,
CMPLOG, CREATR, DELETR, ERASE, INSINI, INSRET, KILLND,
LIGLOG, LIPCOP, MENU, NNLBSU, NNMEAN, NNPASS, NNSORT,
OEXIT, OPENFX, OPENTR, SETUP, TRMPUT, UINNBR

HISTORY:

designed by Steven Haehn November 9, 1981

programmed by Steven Haehn November 18, 1981

PROGRAM NAME: NXFRM

CATEGORY: TRANSFORMATION ROUTINE

PROGRAM DESCRIPTION:

NXFRM produces a new data tree which is a normalized version of the current data set. The normalization procedure involves dividing each measurement of each vector by the transformation vector and adjusting the means and covariances of all nodes.

PROGRAM USAGE:

```
LOGICAL NXFRM
        .
        .
        .
IF(NXFRM(CDTREE,SLOTNO,NDTREE,NDIM,XVECTR))
```

INPUT ARGUMENTS:

CDTREE - INTEGER array (TRELEN); the current data tree name

SLOTNO - INTEGER; the entry table slot number of the current node

NDTREE - INTEGER array (TRELEN); the new data tree name

NDIM - INTEGER; the vector dimensionality of both the current and the new tree

XVECTR - REAL array (NDIM); the vector that will be used to normalize (transform) the current data set

OUTPUT ARGUMENTS:

NXFRM - LOGICAL; if 'true', indicates the transformation was successful. if 'false' indicates the calling routine should exit.

ALGORITHM / NOTES:

A key feature of the following algorithm is that the entry table slot numbers with which nodes are identified do not change in the new tree. This way a minimum number of pointers in the TI file entries need to be adjusted. (See section in Programmer's Reference Manual on 'Creating New Trees from Old Trees').

Open the current data set tree files and create
the new data tree files (OPENTR, CREATR)

Read in the counts and pointers of the current node
(senior node in the new tree).

Save the total number of vectors in the current tree.

```
INTIALIZE:

    NUMBER OF NODES = 0
    NODE LEVEL = 1
    SAVE CURRENT NODE LEVEL
    VECTOR POINTER IN THE NEW TREE = 1
    SET SENIOR NODE NAME TO BE '*****'
    PARENT POINTER = 0
    SIBLING POINTER = 0

REPEAT

        NUMBER OF NODES = NUMBER OF NODES + 1

        Transfer the TI file entry of the node.

            Adjust the node level if necessary

            entry table slot number of the current
            node = NUMBER OF NODES

        IF (node is a lowest node) THEN

            Adjust vectors and transfer
            DO I = 1,NUMBER OF VECTORS IN THIS NODE
                VECTOR (I) = VECTOR (I) / SD (I)
            ENDDO

            Set vector pointer.

            IF(this is the first lowest node encountered)
                lowest node back pointer = -1
            ENDIF

        ENDIF
```

```
            Adjust the means and covariances
                DO I = 1,NDIM
                    DO J = I,NDIM
                        COV MATRIX(I,J) = COV MATRIX(I,J) /
                        (SD (I) * SD (J))
                    ENDDO

                    MEAN (I) = MEAN (I) / SD (I)

                ENDDO

        UNTIL(THERE ARE NO MORE NODES TO BE TRANSFORMED)

        Set the lowest node link pointer of the last
        lowest node = -1

        Write the new TI and TV file headers.
        Senior node pointer in TI header = slot number
        of the current data set node.

        END
```

FILES:

TI - Tree Information (current tree and new tree)
TV - Tree Vector (current tree and new tree)

REFERENCED BY:

NORMXFRM, MATXFRM

SUBPROGRAMS REFERENCED:

CLOSTR, CREATR, GNXTND, INSINT, INSPGM,
INSRET, OPENTR, TIGCAP, TIGCOV, TIGET, TIGMN
TIPCAP, TIPCOP, TIPCOV, TIPET, TIPHDR, TIPMN,
TIPNAM, TRMPUT, TVGVEC, TVPHDR, TVPVEC

HISTORY:

designed by Donna Morris March 1, 1980

programmed by Donna Morris March 1, 1980

PROGRAM NAME: OBFGEN

CATEGORY: OLPARS Programmer Aids (system dependent)

PROGRAM DESCRIPTION:

OBFGEN (Olpars Binary File GENerator) creates a binary file using a text file and a format file. The text file contains the information to be placed in the binary file. The format file is used to direct the conversion of the contents of the text file into the binary file.

PROGRAM USAGE:

User types 'RUN <optional path name>OBFGEN'

USER INTERACTION:

The user is queried for a file output name (the binary file), text file name, and a format file name.

NOTE - To obtain a new version of an already existing file, make sure to specify ';0' explicitly as the file's version number.

ALGORITHM / NOTES:

The contents of the format file consists of conversion specifications, found one per line. A conversion specification consists of the character '$', an optional assignment suppressing character '/', an optional initial-space-skipping suppressing character '-', a conversion character (upper case only), and an optional numerical MAXIMUM field width specifier (i.e., the field width may actually be smaller than specified, but not larger than specified).

CONVERSION SPECIFICATION

$ [/] [-] conversion-character [MAX field width]

A conversion specification is used to direct the conversion of the next input field; the result is placed within the binary file unless assignment suppresion was indicated by the '/' character. The assignment suppression character '/' directs OBFGEN to skip over the specified type of field in the input stream. An input field is defined as a string of non-space characters. However, an exception can occur when using an 'A' or 'S' conversion specification.

See 'A' description.

The following conversion characters are permissable:

I — indicates that a decimal integer is expected in the input stream; the value in the binary file will be word aligned.

H — indicates that a decimal integer (BYTE size) is expected in the input stream; the value in the binary file is NOT word aligned.

L — indicates that a decimal integer is expected in the input stream; the value in the binary file is word aligned and takes up 2 words.

A — indicates that a character string is expected in the input stream; word aligned, 1 char./word. If an optional field width is not specified, the input record is terminated by either a space character or a newline. If an optional field width is specified, only a newline may terminate the input record before the end of the field is reached. Thus, space characters may be embedded in the output field only if a field width is specified.

S — indicates that a character string is expected; this specification is identical to the 'A' specification except that the string is NOT word aligned in the binary file and there are 2 chars./word.

E or F — indicates that a floating point number is expected in the input stream; word aligned in binary file with a length of two words. The input format for a floating point number is a string of numbers, possibly containing a leading minus sign, followed by an optional exponent field containing an 'E' or 'D', followed by a possible signed integer.

D — indicates that a floating point number is expected in the input stream; word aligned in the binary file with a length of four words. The input format is identical to that of the 'E', 'F' format.

C — indicates that a single character is expected in the input stream; word aligned in binary file, with a length of one word.

Two special escape characters are allowed on the line containing a conversion specification. They are:

:P[n] places the read-cursor to the 'n'th char. position within the current (buffer) line.

:X[n] skips over the next 'n' chars. in the current (buffer) line.

Note, if 'n' is missing in either of the above, 1 is assumed.

When OBFGEN reaches the last conversion specifier (during binary file creation) it starts over again with the first-conversion-specifier remembered.

A '/' character found on a line as the first character effectively tells OBFGEN to forget all the previously used conversion formats. Consequently, the first conversion specification found after the '/' becomes the first-conversion-specifier remembered. If a positive integer is specified immediately after the '/' (eg. /34) the following format strings (to the next '/') are repeated the specified number of times before the next set of format strings are used.

FILES:

Binary output file
Text input file
Format input file

REFERENCED BY:

OLPARS programmer/maintainer

SUBPROGRAMS REFERENCED:

CLOSE, EQUALS, FILGET, FPUT, INDEX, INSSET, LENGA, LENGTH OEXIT, OPENBL, OPENS, TRMGET, TRMPUT

SEE ALSO:

FXDUMP

HISTORY:

designed by Steven Haehn June 23, 1980

programmed by Steven Haehn June 25, 1980

PROGRAM NAME: OBTCLS

CATEGORY: Utility Subroutine

PROGRAM DESCRIPTION:

OBTCLS is used to obtain a list of class symbols from the user. It uses information from the diplay files only. A class select list of valid node names (found in the DI file) is displayed at the users terminal. From these names the user is asked to select valid class symbols.
OBTCLS returns to the calling program the number of valid class symbols typed by the user (NUMSYM), the array containing the valid symbols (SYMBOL), and a flag indicating whether a minus sign was entered as the first character of the input string (NEGATE). A minus sign entered means all the symbols in the class list EXCEPT the ones entered should be selected by the calling program. The 'negate' flag (when set to true) indicates that the user typed a minus sign. If the input obtained from the user is a star '*', then the calling program is returned the entire select list. A minus sign with a star '-*' indicates that no class symbols should be selected by the calling program (the 'negate' flag is true, and the SYMBOL array is empty).

PROGRAM USAGE:

```
INTEGER OBTCLS
  .
  .
N = OBTCLS (FDDI,NCLASS,CLASS,PMODE,ONETWO,CLSFLG,
            NUMSYM,SYMBOL,NEGATE)
```

INPUT ARGUMENTS:

FDDI - INTEGER; file descriptor of the DI file

NCLASS - INTEGER; the number of classes in the DI file

CLASS - INTEGER array dimension(MAXCLS); contains the selected class symbols from the DI file

PMODE - INTEGER; the prompt mode from the CM file

ONETWO - INTEGER; flag indicating whether it is a one or two space display

CLSFLG - LOGICAL; flag indicating wnether the calling program wants all classes or just the ones with the display flag 'on'.

OUTPUT ARGUMENTS:

NUMSYM - INTEGER; number of valid symbols typed by the user

SYMBOL - INTEGER array dimension(MAXCLS); array containing the valid class symbols typed by the user

NEGATE - LOGICAL; flag indicating whether a minus sign was entered as the first character of the input string

ALGORITHM / NOTES:

Obtain the valid class symbols from the DI file

Print out the select list header at users terminal

REPEAT

Obtain class symbols from the user

IF(input .eq. '*' or input .eq. '-*')THEN

Select either all or no class symbols

ELSE

Check for minus sign as first character

If just a minus sign was typed, do prompt

IF(number of symbols is more than one)THEN

Set negate to true

Fill symbol array with symbols typed (not incuding the minus sign)

```
            Check to see if symbols exist

            IF(symbols do not exist) THEN

               If it is the first occurence of the symbol
               print error message to user

               Prompt user again

            ENDIF

         ENDIF

      ENDIF

   UNTIL
```

FILES:

DI - Display Information
Instrumentation file

REFERENCED BY:

INTENSIFY, SELECT

SUBPROGRAMS REFERENCED:

DIGENT, EQUALA, INSINT, INSPGM, INSRET, LENGA, PROMPT, TRMGET, TRMPUT

SEE ALSO:

SELCLS

HISTORY:

designed by Mark Maginn October 15, 1980

programmed by Mark Maginn October 17, 1980

PROGRAM NAME: OCLOSE

CATEGORY: Level I File Manipulation Subroutine
(system dependent)

PROGRAM DESCRIPTION:

After it checks the "put" flag entry in the FACT (for the given FID) to decide whether it needs to write out the current buffered record, OCLOSE frees the portion of the FACT, pointed to by FID, and performs a "system" file close. (Refer to Section 3 of the EASSON Final report.)

PROGRAM USAGE:

CALL OCLOSE(FID)

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the file to be closed

ALGORITHMS / NOTES:

This routine has the following named COMMON block:

FACT - the File Access and Control Table which is described in the EASSON Final report, Section 3-3.

FILES:

FCT - File Code Table
Any OLPARS files

REFERENCED BY:

CLOSTR, CLOSFX

SEE ALSO:

ODELET

HISTORY:

designed by Steven Haehn June 27, 1978

programmed by Donna Morris February 21, 1979

PROGRAM NAME: OCREAT

CATEGORY: Level I File Manipulation Subroutine
(system dependent)

PROGRAM DESCRIPTION:

OCREAT creates a new "system" file for OLPARS program use. NAME specifies the OLPARS name of the file being created. TYPE indicates that the file is one of the following:

1. tree information file
2. tree vector file
3. logic information file
4. logic value file
5. fixed file

All the above types have entries placed in the file portion of the File Code Table (FCT). The entry in the FCT is a "system" file name created using the NAME and TYPE arguments of OCREAT. The file code is returned in FCD. OCREAT fills in the FACT entries the same as OOPEN does and returns a FACT pointer in FID. (Refer to Section 3 of the EASSON Final report.)

PROGRAM USAGE:

```
LOGICAL OCREAT

      .
      .
      .

IF(OCREAT(FID,FCD,NAME,TYPE,HENE,LENOE))
```

INPUT ARGUMENTS:

NAME - LOGICAL *1 ARRAY; the OLPARS name of the file
TYPE - INTEGER; the file type
HENE - INTEGER; the number of elements in the header
LENOE - INTEGER; the number of elements in a logical entry

OUTPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the newly created file

FCD - INTEGER; the file code of the file (simply a pointer into the file code table)

ALGORITHMS / NOTES:

This routine has the following named COMMON block:

FACT - the File Access and Control Table which is described in the EASSON Final report, Section 3-3.

FILES:

FCT - File Code Table
Any OLPARS files

REFERENCED BY:

CREATR, CREAFX

SUBPROGRAMS REFERENCED:

PACKB,CONCAT,FCREAT,SEARCH,FCTOPN,FCGHDR,FCGENT,FCPENT, FCPHDR,OCLOSE,GETFID,TRMPUT

DIAGNOSTICS:

If the file cannot be created, OCREAT will return with a 'false' value.

SEE ALSO:

OOPEN

HISTORY:

designed by Steve Haehn June 27, 1978

programmed by Donna Morris March 12,1979

PROGRAM NAME: ODELET

CATEGORY: Level I File Manipulation Subroutine (system dependent)

PROGRAM DESCRIPTION:

ODELET removes an existing "system" file from the computer operating system's file structure. It also removes an entry in the File Code Table file pointed to by FCD, if the file is not a fixed file.

PROGRAM USAGE:

CALL ODELET(FCD)

INPUT ARGUMENTS:

FCD - INTEGER; the file code of the file to be deleted

FILES:

FCT - File Code Table
Any OLPARS files

REFERENCED BY:

DELETR, DELEFX

SUBPROGRAMS REFERENCED:

DELETE, FCGENT, FCGHDR, FCPENT, FCPHDR, FCTOPN, INSINT, INSPGM, INSRET, OCLOSE, TRMPUT

DIAGNOSTICS:

ODELET will usually be silent when possible errors occur, but instrumentation will show the problem. However, if the system file delete fails, ODELET will scream.

SEE ALSO:

OCLOSE

HISTORY:

designed by Steven Haehn June 27, 1978

programmed by Steven Haehn Aug. 21, 1980

PROGRAM NAME: OEXIT

CATEGORY: Utility subroutine (system dependent)

PROGRAM DESCRIPTION:

OEXIT closes all open files, thus releasing all logical unit numbers. Under the separate program approach (see EASSON Final Report), OEXIT will halt execution of the calling program. (Under the "executive approach", OEXIT should close all the open files and return to its calling routine.

PROGRAM USAGE:

CALL OEXIT

ALGORITHM / NOTES:

NOTE: The block I/O file buffers must be explicitly flushed because the RSX-11M FCS routines will not perform this function. The FCS routines will flush all record I/O buffers, however. Therefore, any record I/O files left open (not including the terminal) will simply be closed.

There was an attempt made here to give programmers a warning message about non-completing program instrumentation (see OLPARS final report). This will help notify the programmer that the number of calls to 'INSPGM' and 'INSRET' are unmatched (not the same).

The subroutine call to 'GETBUF', found in OEXIT, must -NEVER- be executed. It is put in OEXIT only to generate a global symbol that will be resolved at task build (link) time.

REFERENCED BY:

Probably all programs within OLPARS.

SUBPROGRAMS REFERENCED:

CLOSE, EXIT, (GETBUF), TRMPUT, WRITEB

DIAGNOSTICS:

Write errors to any of the block I/O files will be mentioned. As mentioned above, notice of incomplete instrumentation will be given.

HISTORY:

designed by Steven Haehn Dec. 15, 1978

programmed by Steven Haehn Apr. 17, 1979

PROGRAM NAME: OMOVE

CATEGORY: Level I File Manipulation Subroutine (system dependent)

PROGRAM DESCRIPTION:

OMOVE switches the file name portions of two filename records (logical entries) of variable files within the file code table (FCT).

PROGRAM USAGE:

```
LOGICAL OMOVE

IF(OMOVE(FCD1,FCD2))
```

INPUT ARGUMENTS:

FCD1 - INTEGER; FCT filename record pointer (also known as a file's 'file code')

FCD2 - INTEGER; FCT filename record pointer

FILES:

FCT - File Code Table

REFERENCED BY:

RENAMT

DIAGNOSTICS:

OMOVE returns FALSE if a FCD points to a fixed file or to a free list entry.

SEE ALSO:

ORENAM

HISTORY:

designed by Steven E. Haehn June 27, 1978

programmed by David J. Tipton October 31, 1980

PROGRAM NAME: OOPEN

CATEGORY: Level I File Manipulation Subroutine
(system dependent)

PROGRAM DESCRIPTION:

OOPEN locates the first empty entry in the FACT and sets FID so it points to that entry. The file code (FCD) for variable files is obtained from the tree or logic list files before calling OOPEN. The header size (HNE) and logical entry size (LENE) of the file must be given so they can be entered into the FACT. The "put flag" (PF) and the number of the current physical record (CPRN) entries of the FACT will be set to zero. OOPEN obtains a "system" file name from the File Code Table (FCT) via the FCD and performs a "system" file open. The logical unit number entry within the FACT is also set by OOPEN. (Refer to Section 3 of OLPARS Final Report.)

PROGRAM USAGE:

```
LOGICAL OOPEN
        .
        .
        .
IF (OOPEN(FID,FCD,HLEN,LELEN))
```

INPUT ARGUMENTS:

```
  FCD - INTEGER; the file code of the file to be opened
HLEN  - INTEGER; the number of elements in the header
LELEN - INTEGER; the number of elements in a logical
                 entry
```

OUTPUT ARGUMENTS:

```
  FID - INTEGER; the file descriptor of the opened file
```

ALGORITHMS / NOTES:

This routine has the following named COMMON block:

FACT - the File Access and Control Table which is described in the OLPARS Final report, Section 3-3.

OLPARS Program Specifications
OOPEN

FILES:

FCT - File Code Table
Any OLPARS files

REFERENCED BY:

OPENTR, OPENFX

DIAGNOSTICS:

If the file cannot be opened, OOPEN will return a 'false' value.

SEE ALSO:

OCREAT

HISTORY:

designed by Steven Haehn June 27, 1978

programmed by Steven Haehn March 27, 1979

PROGRAM NAME: OPENBL

CATEGORY: File I/O (system dependent)

PROGRAM DESCRIPTION:

OPENBL opens a "system file for "block" access. The program sets up all the necessary information in the File Access and Control Table (FACT) and returns a file descriptor (subscript pointer into the FACT) to the calling program. Note that OPENBL does not use the File Code Table (FCT) like OOPEN and OCREAT.

PROGRAM USAGE:

```
INTEGER OPENBL
      .
      .
      .
IFD = OPENBL(FYLENM,NAMLEN,HLEN,LELEN,ACCESS,FILTYP)
```

INPUT ARGUMENTS:

FYLENM - INTEGER ARRAY(NAMLEN); File name of the system file to be opened. 1 char./integer

NAMLEN - INTEGER; length of the given file name

HLEN - INTEGER; number of real elements in the file header

LELEN - INTEGER; number of real elements in the file's logical entry (record).

ACCESS - INTEGER; 0 - file is to be opened for reading
1 - file is to be opened for writing

FILTYP - INTEGER; 0 - file is an existing (OLD) file
1 - file is a NEW file

OUTPUT ARGUMENTS:

OPENBL - INTEGER; file descriptor of the opened file or a -1 when file fails to open

ALGORITHM / NOTES:

This routine alters the contents of the FACT when the 'file open' succeeds.

REFERENCED BY:

FILEIN

SUBPROGRAMS REFERENCED:

CONCAT, FOPEN, FCREAT, GETFID, PACKB

DIAGNOSTICS:

A -1 is returned when the file fails to open.

SEE ALSO:

CLOSBL

HISTORY:

designed by Steven Haehn June 12, 1979

programmed by Steven Haehn June 19, 1979

PROGRAM NAME: OPENFX

CATEGORY: Level II File Manipulation Subroutine

PROGRAM DESCRIPTION:

OPENFX opens an OLPARS "fixed" file. The fixed file is referenced by its file code (FCD). The file descriptor (FID) of the opened file is returned. For the user's fixed files that depend on the dimensionality of the data vectors (such as DI and DV), NDIM should contain the value of the vector dimensionality and DSPTYP should contain the display type. For the fixed files that do not depend on vector dimensionality, NDIM and DSPTYP are ignored.

PROGRAM USAGE:

```
LOGICAL OPENFX
      .
      .
      .
IF (OPENFX (FID,FCD,NDIM,DSPTYP))
```

INPUT ARGUMENTS:

FCD - INTEGER; the file code of the fixed file to be opened (see CREAFX)

NDIM - INTEGER; the dimensionality of the data set

DSPTYP - INTEGER; the display type code

0: Unknown
1: 2-space cluster
2: 2-space scatter
3: rank order
4: confusion matrix
5: 1-space macro
6: 1-space micro

OUTPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the opened file

FILES:

All the OLPARS "fixed" files.

SUBPROGRAMS REFERENCED:

OOPEN

DIAGNOSTICS:

If the fixed file cannot be opened, OPENFX will return with a 'false' value.

SEE ALSO:

CREAFX, CLOSFX, DELEFX

HISTORY:

designed by Steven Haehn June 26, 1978

programmed by Steven Haehn February 5, 1979

PROGRAM NAME: OPENS

CATEGORY: File I/O (system dependent)

PROGRAM DESCRIPTION:

OPENS gives a program access to a "system" sequential file. The file will be pointed to by a logical unit number.

PROGRAM USAGE:

```
INTEGER OPENS
      .
      .
      .
LUN = OPENS(FILENM,ACCESS,TYPE)
```

INPUT ARGUMENTS:

FILENM - CHARACTER String; the name of the file to be 'opened.'

ACCESS - INTEGER; 0 - represents read access to the file
1 - represents write access to the file
2 - represents append access to the file

TYPE - INTEGER; 0 - represents that the file already exists (an 'old' file type)
1 - represents that the file is to be created (a 'new' file type)

OUTPUT ARGUMENTS:

OPENS - INTEGER; the logical unit number of the sequential file 'opened'

REFERENCED BY:

Programs that will create printer output

SUBPROGRAMS REFERENCED:

VCREAT, VOPEN

DIAGNOSTICS:

A negative one will be returned if the file could not be opened or there were no available logical unit numbers.

SEE ALSO:

CLOSE

HISTORY:

designed by Steven Haehn September 6, 1978

programmed by Steven Haehn April 25, 1979

PROGRAM NAME: OPENTR

CATEGORY: Level II File Manipulation Subroutine

PROGRAM DESCRIPTION:

OPENTR obtains the file descriptor(s) (IFID, VFID) of the tree/logic information file and/or the tree vector/ logic value file (OLPARS data or logic tree files). OPENTR searches the tree list or logic list file depending on which TYPE of tree is being opened. If the NAME of the tree does not appear in the appropriate file, then OPENTR will return with a 'false' value. If the NAME of the tree is found, OPENTR will proceed to open the tree files as specified in the ACCESS parameter. IFID is the file descriptor for the tree/ logic information file and VFID is the file descriptor for the tree vector/logic value file. NDIM, the dimensionality of the tree, is obtained from the tree/logic list file and returned to the calling program.

Value of tree TYPE:
1 - data tree
2 - logic tree

Value of ACCESS:
1 - open information file only
2 - open vector/value file only
3 - open both files of tree

PROGRAM USAGE:

```
LOGICAL OPENTR
      .
      .
      .
IF (OPENTR(IFID,VFID,NDIM,NAME,TYPE,ACCESS))
```

INPUT ARGUMENTS:

NAME - INTEGER; name of the tree

TYPE - INTEGER; see program description

ACCESS - INTEGER; see program description

OUTPUT ARGUMENTS:

NDIM - INTEGER; dimension of the data set

IFID - INTEGER; the file descriptor of the TI or LI file

VFID - INTEGER; the file descriptor of the TV or LV file

FILES:

TI - tree information
TV - tree vector
TL - tree list
LI - logic information
LV - logic value
LL - logic list

SUBPROGRAMS REFERENCED:

TLSRCH, OOPEN, LLSRCH, OPENFX, TRMPUT, CLOSFX, MAXO

DIAGNOSTICS:

When any portion of the tree cannot be opened, OPENTR returns with a 'false' value.

SEE ALSO:

CREATR, CLOSTR, DELETR

HISTORY:

designed by Steve Haehn June 21, 1978

programmed by Donna Morris July 10,1979

PROGRAM NAME: OPNTMP

CATEGORY: FILE I/O (system dependent)

PROGRAM DESCRIPTION:

OPNTMP opens a "system" file for temporary use. The file does not have an entry in the FCT file and it will disappear from the "system" after it is closed. TMPFD will point to the FACT entry of the temporary file. If OLDFD is not zero, then OPNTMP assumes that the temporary file has the format of a previously opened file (pointed to by OLDFD) and copies the header and logical entry sizes of OLDFD to the appropriate places in the FACT entry, pointed to by TMPFD. The name of the temporary file is found in NAME.
(Refer to Section 3 of the EASSON Final Report.)

PROGRAM USAGE:

```
LOGICAL FUNCTION OPNTMP
   .
   .
   .
IF (OPNTMP(TMPFD,OLDFD,NAME,NEWHSZ,NWLESZ))
```

INPUT ARGUMENTS:

OLDFD - INTEGER; the file descriptor of a previously opened file (or zero)

NAME - INTEGER array (FNMLEN); the name of the temporary file

NEWHSZ - INTEGER; the number of elements in the header of the temporary file. Used only when there are no previously opened files.

NWLESZ - INTEGER; the number of elements in a logical entry of the temporary file. Used only when there are no previously opened files

OUTPUT ARGUMENTS:

TMPFD - INTEGER; file descriptor of the temporary file being created

ALGORITHMS / NOTES:

This routine has the following named COMMON blocks:

FACT - the File Access and Control Table which is described in the EASSON Final report, Section 3-3.

SUBPROGRAMS REFERENCED:

FCREAT, GETFID, PACKB

DIAGNOSTICS:

If OPNTMP cannot open up the temporary file, then it will return to the calling program with a 'false' value.

HISTORY:

designed by Steve Haehn June 27, 1978

programmed by Jill King January 15, 1981

PROGRAM NAME: OPTDSP

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

OPTDSP does the modifications for OPTIMLMOD on a pair wise logic node. The vectors are projected onto the optimal discriminate plane (Fisher Direction X Orthogonal Direction) and displayed on the screen. The user is asked to enter a boundary (GIN). The orthogonal components to each segment of the boundary are stored in the LV file (PTRGNR), along with a threshold for each segment. The user is asked for the class symbol of the convex region. This is used to determine which node should be assigned to the convex side of the boundary.

PROGRAM USAGE:

```
CALL OPTDSP(FIDCM,FIDLV,FIDTI,FIDTV,LVNEL,TRUCLS,NDIM,
            PMODE,POINTS,NXTCLS,ET,CLIST,OPTNO,NODNUM)
```

INPUT ARGUMENTS:

FIDCM - INTEGER; file descriptor of the CM file

FIDLV - INTEGER; file descriptor of the LV file

FIDTI - INTEGER; file descriptor of the TI file

FIDTV - INTEGER; file descriptor of the TV file

LVNEL - INTEGER; number of elements in an LV file entry

TRUCLS - INTEGER; number of true classes at logic node

NDIM - INTEGER; dimensionality of dataset

PMODE - INTEGER; the prompt mode flag

POINTS - INTEGER array ((TRUCLS*(TRUCLS-1))/2); the pointers to the class pairs

NXTCLS - INTEGER; the index of this class pair

ET - INTEGER array (TABSIZ); the entry table

CLIST - INTEGER array (TRUCLS); slot numbers of classes at logic node

OPTNO - INTEGER; the option number of OPTIMLMOD

NODNUM - INTEGER; the logic node number being modified

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Put header and rectangle up on screen.

Retrieve the vectors from the TV file (TVGVEC).

Project the vectors for this class pair onto the plane.

Prompt the user for a boundary (GIN).

Ask for a convex point (GIN).

Ask for the class symbol of the class associated with the convex region (TRMGET).

Store the boundary (by segments and thresholds) in the LV file (PTRGNR).

Return to main program.

REFERENCED BY:

OPTIMLMOD

SUBPROGRAMS REFERENCED:

BOUND, CLOSFX, CMGSCN, DIGHDI, DIPEFS, DIPENT, DIPHDI
DIPMAX, DSCVTH, DVPHDR, DVPVEC, EQUALA, ERASE, GIN
GTRGNR, INSPGM, INSRET, LINSEG, LVPLNK, MARK, MOVE
OPENFX, PROJCT, PROMPT, PTRGNR, RCTNGL, SCAT, TEXT
TIGCOP, TIGNAM, TRMGET, TRMPUT, TVGVEC, WRTNOW

SEE ALSO:

THRDSP, OPTIMLMOD

HISTORY:

designed by John W. Tenney Sept. 2, 1981

programmed by John W. Tenney Oct. 9, 1981

PROGRAM NAME: OPTIMLMOD

CATEGORY: Logic Design Command

PROGRAM DESCRIPTION:

OPTIMLMOD is designed to modify an existing FISHER or OPTIML logic node. The user is given a list of logic nodes with FISHER or OPTIML logic. After the node is selected, the user is prompted for a class pair. The projections of the vectors onto the optimal two-space plane are displayed on the screen and the user is prompted for a boundary. Then the user is prompted for the class symbol of the class associated with the convex point. The logic is stored in the LV file. The user is then prompted for another class pair.

To evaluate the modified logic, use PWEVAL.

PROGRAM USAGE:

User types in 'OPTIMLMOD'.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Display a list of PAIRWISE logic nodes (GETLST,TRMPUT).

Ask user to enter a PAIRWISE logic node number (PROMPT).

REPEAT

Prompt user for a class pair (GETPR).

Project and display the vectors on the screen, get the boundary from the user, and store the logic in the LV file (OPTDSP).

UNTIL (user is satisfied)

Fix temporary logic elements in the TV file (KILLND).

Put up option list at user's terminal (MENU)

END

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CHKEXS, CLOSTR, CMGCDS, CMGLOG, CMGOTH, DLREGN, ERASE,
GETLST, GETPR, GTRGNI, GTRGNR, GTSLOT, INSINI, INSRET,
KILLND, LIGCOP, LIGLOG, LIGSTR, LIPLOG, MENU, OEXIT,
OPENFX, OPENTR, OPTDSP, PROMPT, SETOPT, TIGET, TIGNAM,
TRMGET, TRMPUT

SEE ALSO:

FISHER, FISHMOD, THRESHMOD, PWEVAL

HISTORY:

designed by John W. Tenney Sept. 2, 1981

programmed by John W. Tenney Oct. 9, 1981

PROGRAM NAME: ORENAM

CATEGORY: Level I File Manipulation Subroutine
(system dependent)

PROGRAM DESCRIPTION:

ORENAM changes the name of the "system" filename within the FCT file, pointed to by FCD (for variable files). TYPE is the same as in OCREAT. NAME and TYPE are used together to create the new "system" name. (Refer to Section 3 of the EASSON Final Report.)

PROGRAM USAGE:

```
LOGICAL ORENAM

IF (ORENAM(FCD,NAME,TYPE))
```

INPUT ARGUMENTS:

FCD - INTEGER; file code of the file to be renamed

NAME - INTEGER ARRAY; new name of file (with a null character following)

TYPE - INTEGER; the file type

FILES:

FCT - File Code Table

REFERENCED BY:

RENAMT

DIAGNOSTICS:

ORENAM returns FALSE if the FCD points to a fixed file or to a free list pointer of the FCT. ORENAM also returns false if the new name and type already exists in the FCT, or if the system rename fails. In the case of system failure, ORENAM lets the user know via a terminal message.

SEE ALSO:

OMOVE, OCREAT

HISTORY:

designed by Steven E. Haehn June 27, 1978

programed by David J. Tipton October 31, 1980

PROGRAM NAME: PACKB

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

PACKB packs an integer array of characters into a logical *1 array. Packing continues until the number of integers to pack is exhausted, or until the first null character encountered has been packed, depending on whether or not the stop packing flag has been set. The actual number of characters packed is returned.

PROGRAM USAGE:

```
CALL PACKB(WORD,BYTE,NCHAR,NPACKD,STOP)
```

INPUT ARGUMENTS:

WORD - INTEGER; array of characters to pack

BYTE - LOGICAL *1 ARRAY; output buffer for packed characters

NCHAR - INTEGER; number of characters to pack

STOP - LOGICAL; if true, stop packing characters after the first null character has been packed. if false, continue packing characters until 'NCHAR' is exhausted.

OUTPUT ARGUMENTS:

NPACKD - INTEGER; number of characters packed

REFERENCED BY:

Level I Routines

SEE ALSO:

PACKW

HISTORY:

designed by Steve Haehn March 9,1979

programmed by Donna Morris March 12,1979

PROGRAM NAME: PACKW

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

PACKW packs real elements into integer words. Packing continues until the number of real elements to pack is exhausted or until the first null character encountered has been packed, depending on whether or not the stop packing flag has been set. The actual number of real elements packed is returned.

PROGRAM USAGE:

```
CALL PACKW(REALS,INTGRS,NUMBER,NPACKD,STOP)
```

INPUT ARGUMENTS:

REALS - REAL; array of real elements to pack

INTGRS - INTEGER; array; output buffer for packed elements

NUMBER - INTEGER; number of real elements to pack

STOP - LOGICAL; stop packing flag. if true, stop packing real elements after the first null character has been packed. if false, continue packing real elements until 'NUMBER' is exhausted.

OUTPUT ARGUMENTS:

NPACKD - INTEGER; the number of real elements actually packed

REFERENCED BY:

Level I Routines

SEE ALSO:

PACKB

HISTORY:

designed by Steve Haehn March 9,1979

programmed by Donna Morris May 9,1979

PROGRAM NAME: PAIRLG

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

PAIRLG computes the vote count for a vector run through pairwise logic. It goes through the logic block, retrieving necessary data for each class pair, and computing a vote for that class pair.

PROGRAM USAGE:

```
INTEGER FUNCTION PAIRLG

NODNUM=PAIRLG(FIDLI,FIDLV,TRUCLS,VTCNT,PNTR,NDIM,VEC,
              LVNEL,VOTES,REJNOD)
```

INPUT ARGUMENTS:

FIDLI - INTEGER; the file descriptor of the LI file

FIDLV - INTEGER; the file descriptor of the LV file

The next three are the LV first entry

TRUCLS - INTEGER; the number of classes at the logic node

VTCNT - INTEGER; the minimum vote count

PNTR - INTEGER; the link to the first class pair (LV)

NDIM - INTEGER; the dimensionality

VEC - REAL array (NDIM); the vector

LVNEL - INTEGER; the number of elements in an LV entry

OUTPUT ARGUMENTS:

VOTES - INTEGER array (NCLAS); the vote count

NODNUM - INTEGER; the logic node number to which the vector was assigned

REJNOD - LOGICAL; 'true' means that the vector was rejected

ALGORITHM / NOTES:

```
Initialize REJNOD to 'false'.

DO WHILE (there are more class pairs)

    Get first entry for the class pair to determine the
    decision criteria.

    Get proper parameters from LV file.  (GTRGNI,GTRGNR)

    Call proper decision routine:  FISH for Fisher
    and EV2SP for optimal discriminant plane.

    Update vote counts.

ENDDO

IF (maximum class vote count is greater than or equal to
the minimum) THEN

    IF (maximum class vote count ocurred at only one
        class) THEN

        Set NODNUM = the logic node number associated
        with that class.

    ELSE

        Get a priori probabilities for all classes where
        the maximum vote count occurred.

        IF (largest a priori probabilities are not tied)
        THEN

            Set NODNUM = logic node number associated
            with class with largest a priori probability.

        ELSE

            Set NODNUM = logic node number associated
            with reject region. Set REJNOD to 'true'.

        ENDIF

    ENDIF
```

ELSE

Set NODNUM = logic node number associated with the reject region. Set REJNOD to 'true'.

ENDIF

RETURN

FILES:

LI - logic information
LV - logic value

REFERENCED BY:

PEPAIR, LOGEVAL

SUBPROGRAMS REFERENCED:

FISH, GTRGNI, GTRGNR, INSINT, INSPGM, INSRET, LIGPRB, TRMPUT

SEE ALSO:

PWEVAL, PEPAIR, LOGEVAL

HISTORY:

designed by John W. Tenney Sept. 20, 1981

programmed by John W. Tenney Oct. 9, 1981

PROGRAM NAME: PENMV

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

PENMV performs an evaluation of a user-designed NMV logic at a logic node. It sets up the DI file for display of a confusion matrix and then displays the confusion matrix

PROGRAM USAGE:

```
CALL PENMV(FIDCM,FIDDI,FIDLI,FIDLV,FIDTI,FIDTV,ENTNO,
           ET,NODNUM,LVNEL,NASCLS,CLIST,PROGNM,PMODE)
```

INPUT ARGUMENTS:

FIDCM - INTEGER; the file descriptor of the CM file

FIDDI - INTEGER; the file descriptor of the DI file

FIDLI - INTEGER; the file descriptor of the LI file

FIDLV - INTEGER; the file descriptor of the LV file

FIDTI - INTEGER; the file descriptor of the TI file

FIDTV - INTEGER; the file descriptor of the TV file

ENTNO - INTEGER; the number of the LV file entry

ET - INTEGER array (TABSIZ); TI file slot numbr array

NODNUM - INTEGER; the logic node number which equals the LI file entry of the logic node

LVNEL - INTEGER; number of elements in LV file entry

NASCLS - INTEGER; the number of assigned classes

CLIST - INTEGER array (NCLAS); the slots of classes at the logic node

PROGNM - INTEGER array (10); the name of the calling program

PMODE - INTEGER; the prompt mode flag

USER INTERACTION:

Asks user if he wishes a printout of the confusion matrix and error listing.

ALGORITHM / NOTES:

Obtain the weight flag and reject flag from the LV file NMV logic block (GTRGNI).

Get necessary statistical information and reject distances from NMV logic block (GTRGNI,GTRGNR).

Get neccesary tree information from TI file (TIGHDR, TIGET,TIGCOP)

Open an error listing file (OPENS).

DO (while there are more classes to process)

Get vector pointer of next class from TI file (TIGCOP).

Get vector (TIGVEC).

For each vector in that class that is at the logic node, determine which class it should be assigned to or if it should be rejected (NMDIST). If in error, or rejected, enter an entry in error file (FILPUT).

Set temporary logic elements in TV for each vector (TVPVEC).

Update totals for the class.

Insert a DI file entry for this class (DIPCME).

ENDDO

Compute totals for confusion matrix.

Set up DI file header (DIPCMH). Use SETOPT to get the option number and CMGLOG to get the logic name.

Diplay confusion matrix on screen (CMDISP).

```
        IF (user desires a printout of the confusion matrix)

            Print it (CMPRT).

        IF (user desires a printout of the error listings)

            Print error file (PRINTR).

        RETURN
```

FILES:

```
        DI - display information
        LI - logic information
        LV - logic value
        TI - tree information
        TV - tree vector
```

REFERENCED BY:

        NMV, NMVMOD

SUBPROGRAMS REFERENCED:

```
        CLOSE,  CMDISP, CMGCDS, CMGLOG, CMPRT,  DIPCME, DIPCMH,
        EQUALA, FILPUT, GTRGNI, GTRGNR, INSCHR, INSINT, INSPGM,
        INSRET, LIGPRB, NMDIST, OPENS,  PRINTR, PROMPT, SETOPT,
        TIGCOP, TIGHDR, TIGNAM, TRMGET, TRMPUT, TVGVEC, TVPLOG
```

HISTORY:

        designed by John W. Tenney April 9, 1981

        prorammed by John W. Tenney May 24, 1981

PROGRAM NAME: PEPAIR

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

PEPAIR performs a partial pairwise evaluation at a pairwise logic node. The results of the evaluation are presented on the screen in confusion matrix format and the user may request a detailed error listing to be printed.

PROGRAM USAGE:

```
CALL PEPAIR(FIDCM,FIDDI,FIDLI,FIDLV,FIDTI,FIDTV,ENT1,
            ET,NODNUM,LVNEL,LOGCLS,CLIST,PROGNM,PMODE)
```

INPUT ARGUMENTS:

```
 FIDCM - INTEGER; the file descriptor of the CM file
 FIDDI - INTEGER; the file descriptor of the DI file
 FIDLI - INTEGER; the file descriptor of the LI file
 FIDLV - INTEGER; the file descriptor of the LV file
 FIDTI - INTEGER; the file descriptor of the TI file
 FIDTV - INTEGER; the file descriptor of the TV file

  ENT1 - INTEGER; the first LV file entry pointer
NODNUM - INTEGER; the current logic node number
 LVNEL - INTEGER; the number of elements in an LV entry
LOGCLS - INTEGER; the number of classes at the logic node
 CLIST - INTEGER array (NCLAS); the list of slot numbers
PROGNM - INTEGER array (CMDLEN); the calling program name
 PMODE - INTEGER; the prompt mode flag
```

ALGORITHM / NOTES:

```
Create a temporary error file (OPENS).

DO WHILE (there are data classes at the pairwise node)

    Get vector pointer and number of vectors from class
    (TIGCOP).

    DO WHILE (there are vectors in the class that fall
    at the logic node).

        Get next vector.

        Determine which logic node the pairwise logic
        votes for (PAIRLG).

        IF (error has been made) THEN

            Insert information into logic error file.

        ENDIF

        Update total statistics.

    ENDDO

    Update total statistics.

ENDDO

Set up display file for confusion matrix.

Store confusion matrix data in DI file (DIPCMH, DIPCME).

Display confusion matrix (CMDISP).

IF (user desires a printout of the confusion matrix) THEN

    Print it (CMPRT).

ENDIF

IF (user desires a printout of the error listings) THEN

    Print error file (PRINTR).

ENDIF

RETURN
```

FILES:

DI - display information
CONMAT - confusion matrix
MISCLASS - error listing

REFERENCED BY:

PWEVAL

SUBPROGRAMS REFERENCED:

CLOSE, CLOSFX, CMDISP, CMGCDS, CMGLOG, CMPRT, DIPCME, DIPCMH, EQUALA, FILPUT, GTRGNI, INSCHR, INSINT, INSPGM, INSRET, OPENFX, OPENS, PAIRLG, PRINTR, PROMPT, SETOPT, TIGCOP, TIGNAM, TRMGET, TRMPUT, TVGVEC, TVPLOG

HISTORY:

designed by John W. Tenney Sept. 20, 1981

programmed by John W. Tenney Oct. 9, 1981

PROGRAM NAME: PRECPT

CATEGORY: Measurement Evaluation Subroutine

PROGRAM DESCRIPTION:

PRECPT prints exception conditions to 'DSCRMEAS' calculations, that is, it prints information associated with class pairs whose discriminant measurement was set to zero because the denominator in the calculation was zero. A zero value in the denominator implies that either the variances are equal to zero or the number of vectors used in the weighting of the classes is zero.

PROGRAM USAGE:

```
LOGICAL PRECPT
        .
        .
        .
IF(PRECPT(FIDCM, PMODE, ASK, PRINT, NLINES, NCHARS,
          PAGNUM, LINCNT, MEASNO, CLASS1, NVECS1,
          CLASS2, NVECS2,MEANDF))
```

INPUT ARGUMENTS:

FIDCM - INTEGER; the file descriptor of the CM file

PMODE - INTEGER; the prompt mode

ASK - LOGICAL; if 'true', indicates ask the user if exception conditions are to be printed. ask is set to 'false' once the user has been asked

PRINT - LOGICAL; if 'ASK' equals 'true' and the user responds with a 'yes', PRINT is set to equal 'true'; else PRINT is set to 'false'. if 'ASK' equals 'false', the value of PRINT is not changed, and if PRINT equals 'true', a line is printed; else PRECPT just returns.

NLINES - INTEGER; the number of lines that fit on a page, excluding header lines. this variable is calculated the first time PRECPT is called and must be passed on all remaining calls.

NCHARS - INTEGER; the number of characters that fit on a line. this variable is calculated the first time PRECPT is called and must be passed on all remaining calls.

PAGNUM - INTEGER; the current page number. this variable is initialized and updated by PRECPT

LINCNT - INTEGER; the current line number (excludes header lines). this variable must be initialized to '1' the first time PRECPT is called and will be updated before returning from each call to PRECPT. LINCNT should be set to '0' at the end of calculations (last page) so that the user will be asked if he wants to continue.

MEASNO - INTEGER; the measurement number to be output

CLASS1 - INTEGER; the first class symbol to be output

NVECS1 - INTEGER; the number of vectors used in weighting class 1 variances

CLASS2 - INTEGER; the second class symbol to be output

NVECS2 - INTEGER; the number of vectors used in weighting class 2 variances

MEANDF - REAL; the difference between the means of class 1 and class 2

OUTPUT ARGUMENTS:

NLINES, NCHARS, PAGNUM, LINCNT - INTEGER;
ASK, PRINT - LOGICAL;
(see INPUT ARGUMENTS).

PRECPT - LOGICAL; if 'false' indicates that execution of the main command should terminate

USER INTERACTION:

The calling program should initialize the 'ASK' flag to 'true'. If 'ASK' is 'true', the user is asked if he wants a listing of exception conditions, and then 'ASK' is set to 'false'. If the user responds with yes ('Y'), a line of information will be printed for each exception condition that occurs; if the user responds with a no ('N'), he is then asked if he wants to continue execution of the command. If he responds with no, PRECPT returns a 'false' value, and the calling routine should exit. If there is more than one page of output (information), the user is asked at the end of each page if he wants to see the next page. A yes response results in the next page being shown and continued execution of the main command. A no response results in a prompt to the user asking if he wishes to continue execution of the command. Again, if he responds with no, PRECPT returns a 'false' value, and the calling routine should exit. If the user types the 'OLPARS standard character for exiting commands' in response to any of the prompts, PRECPT will return with a 'false' value, and the calling routine should exit.

ALGORITHM / NOTES:

PRECPT prints the following information: the measurement number, the first class (of the class/pair) and the number of vectors associated with that class, the second class (of the class/pair) and the number of vectors associated with that class, and the difference between the means of the 2 classes for the given measurement number. It should be noted that the number of vectors printed is the same as the number of vectors that are in the class only when the user chooses to weight classes by the number of vectors in each class, otherwise, the number of vectors printed is equal to the total number of vectors in the data set divided by the number of classes.

FILES:

CM - Communications File

REFERENCED BY:

DSCRMEAS

SUBPROGRAMS REFERENCED:

CMGSCN, INSPGM, INSRET, PROMPT, TRMGET, TRMPUT

HISTORY:

designed by Donna Morris January 30, 1981

programmed by Donna Morris January 30, 1981

PROGRAM NAME: PRINTR

CATEGORY: File I/O (system dependent)

PROGRAM DESCRIPTION:

PRINTR submits a "system" sequential file to a "system" printing device. Thus, the file's contents are written out to the "printer."

PROGRAM USAGE:

CALL PRINTR(FILENM)

INPUT ARGUMENTS:

FILENM - LOGICAL *1; file name of the "system" file to be printed out

REFERENCED BY:

PRTCM, PRTIDX, PRTOSP, PRTRNK, PRTLOG, PRTDS, EIGPRT

HISTORY:

designed by Steve Haehn September 6, 1978

programmed by Donna Morris July 10,1979

PROGRAM NAME: PROJCT

CATEGORY: Numerical Computation Algorithm

PROGRAM DESCRIPTION:

PROJCT projects a data vector onto a projection vector

PROGRAM USAGE:

```
REAL PROJCT
COORD = PROJCT(DVEC,PVEC,NDIM)
```

INPUT ARGUMENTS:

DVEC - REAL array; the data vector
PVEC - REAL array; the projection vector
NDIM - INTEGER; the vector dimensionality

OUTPUT ARGUMENTS:

COORD - REAL; the result of the projection

ALGORITHM / NOTES:

the length of the projection of vector 'DATA' onto vector 'PROJECTION' is given by:

COORDINATE = the dot product of vector DATA divided by the square root of the dot product of vector PROJECTION

REFERENCED BY:

DSPROJ

SUBPROGRAMS REFERENCED:

SQRT

HISTORY:

designed by Donna Morris June 1,1979

programmed by Donna Morris June 1,1979

OLPARS Program Specifications
PROJMN

PROGRAM NAME: PROJMN

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

After a one-space or two-space display is created, a user can involve PROJMN to have the mean vectors for the classes displayed superimposed on the screen. A rectangle is drawn around the projection of each mean vector on a two-space display.

PROGRAM USAGE:

User types in 'PROJMN'.

ALGORITHM / NOTES:

Get the screen parameters from the CM file (CMGSCN).

Get the display code (DIGDC).

IF (display is MACRO or MICRO) THEN

Display the one-space mean vectors (MEAN1).

ELSE

Display the two-space mean vectors (MEAN2).

ENDIF

Force the cursor to viewable position for the next OLPARS command.

END

FILES:

CM - communications file
DI - display information file
DV - display vector file
instrumentation file

REFERENCED BY:

OLPARS user.

SUBPROGRAMS REFERENCED:

CLOSFX, CMGCDS, CMGSCN, DIGDC, ERASE, INSINI, INSINT, INSRET, MEAN1, MEAN2, MENU, OEXIT, OPENFX, TEXT, TRMPUT

HISTORY:

designed by David Birnbaum July 18, 1978

programmed by Jill King January 1, 1981

PROGRAM NAME: PROMPT

CATEGORY: Terminal I/O (character) (system dependent)

PROGRAM DESCRIPTION:

PROMPT is a terminal output routine used by all programs expecting input from an OLPARS user. The program checks the prompting state of OLPARS, i.e. if it is in long or short prompt mode. If OLPARS is in short prompt mode, SHRTMG is printed at the user's terminal. If OLPARS is in long prompt mode, LNGMSG is printed at the user's terminal.

The help flag (HLPFLG) only affects the message printed out when it is set to the value of -2. (Note: -2 happens to be the function value that TRMGET, the terminal input subroutine, returns when it receives the universal help symbol. '?', from the user.) When OLPARS is in short prompt mode, and the help flag is set to -2, PROMPT will print LNGMSG at the user's terminal. When OLPARS is in long prompt mode, and the help flag is set to -2, PROMPT will print the message, 'USE THE OLPARS HELP COMMAND FOR FURTHER INFORMATION.'

PROGRAM USAGE:

CALL PROMPT (HLPFLG,PMODE,'SHRTMG','LNGMSG')

INPUT ARGUMENTS:

HLPFLG - INTEGER; the help flag

PMODE - INTEGER; the prompt mode of OLPARS (the prompt flag is read from the communications file by the calling routine)

SHRTMG - CHARACTER; the short prompt

LNGMSG - CHARACTER; the long prompt

REFERENCED BY:

All OLPARS commands that interact with user

SUBPROGRAMS REFERENCED:

TRMPUT - used for actual output of prompt message to terminal

SEE ALSO:

TRMGET.TRMPUT

HISTORY:

designed by Steve Haehn May 18, 1978

programmed by Donna Morris May 24,1979

OLPARS Program Specifications
PRTCM

PROGRAM NAME: PRTCM

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

PRTCM produces a printout of a confusion matrix stored in the DI file.

PROGRAM USAGE:

User types in 'PRTCM'.

ALGORITHM / NOTES:

```
IF (display code = 6, i.e., confusion matrix) THEN

    Call CMPRT.

ELSE

    Print error message.

ENDIF

END
```

FILES:

DI - display information
HS - history
Instrumentation
OLPARS option file
CM - communications
confusion matrix listing (CONMAT)

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CMPRT, DIGDC, ERASE, INSINI, INSRET, MENU, OEXIT, OPENFX, TRMPUT

HISTORY:

designed by David Birnbaum September 16, 1978

programmed by Steven Haehn May 14, 1981

PROGRAM NAME: PRTDS

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

PRTDS prints basic statistical information about a data set. There are eight options and two viewing choices that the user can select from to help them look at the data set.

PROGRAM USAGE:

User types in 'PRTDS'.

USER INTERACTION:

The user is first prompted for a tree name for which the statistical information to be printed. This is then followed by a list of eight options that appears at the users terminal, from which the user can select any combinaion. Once the options have been selected and are valid, the user determines the type of output to be used, and whether the entire data set is to be used in the processing of the statistical information.

ALGORITHM / NOTES:

Ask user for a tree name.

Put up options at terminal and have user select the ones to be processed.

Ask user for type of output format to be used. Floating point is used unless specified otherwise.

Ask user if they wish to select a subset of the data set to be processed.

Obtain tree information to be used in processing the desired options.

Perform options until all selected are processed

Print output file

OLPARS Program Specifications
PRTDS

FILES:

CM - Communications file
TL - Tree List file
TI - Tree Information file
TV - Tree Vector file
TREEDUMP - output file
HS - History file
OLPARS Option file
Instrumentation file
Temporary file

SUBPROGRAMS REFERENCED:

ALLVEC, CLOSE. CLOSFX, CMGOTH, CORMAT, COVMAT, EQUALA,
ERASE, GNXTND, INSINI, INSINT, INSLOG, INSRET, LENGA,
MENU, MNSTDV, MNVECD, OEXIT, ONEVEC, OPENFX, OPENS,
OPENTR, PRINTR, PROMPT, RNGLAP, SELCLS, TIGET, TIGHDR,
TLSRCH, TRESTR, TRMGET, TRMPUT

HISTORY:

designed by David Birnbaum September 18, 1978

programmed by Mark Maginn July 16, 1981

PROGRAM NAME: PRTIDX

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

PRTIDX identifies a class symbol found on a terminal two-space or one-space micro plot display. Either vector id's or the number of vectors within a certain area will be displayed to the user.

PROGRAM USAGE:

User types in 'PRTIDX'.

USER INTERACTION:

The user is asked if (s)he wants the output to appear on the display terminal or in a file. If a file is requested, then the user is asked for a file name and whether or not the file is to be printed.

When indexing a cluster plot, the user moves the graphics cursor to the center of the desired display symbol and types in a letter or space. If the letter typed is an 'I,' the user will see the identity of the vector(s) that are in the specified grid of the cluster plot. Any other letter (or space) typed in will tell the program to show the user the "count" of the vectors within the cluster plot grid.

For indexing a scatter plot, the user enters two points (via graphics cursor) that represent the opposite corners of a rectangle. All the display symbols within the interior of the rectangle will have their identities made known to the user.

For a one-space micro plot (histogram), the user enters (via the graphic cursor) the bin from which (s)he wants vector counts and percentages.

ALGORITHM / NOTES:

```
IF (the display file does not contain a proper
    display) THEN

    Tell user that "index" only works with a two-space
    plot and a one-space micro plot in the display file.

    QUIT.

ENDIF

REPEAT
IF (the display file contains a one-space
    micro display) THEN

    IF (output is sent to terminal) THEN

        ReDisplay the micro plot (via MICMAC).

    ENDIF

    Ask the user to specify (via graphics cursor) the
    particular "bin" from which information is to be
    obtained.

    Convert screen coordinates to "bin" number.

    Obtain the number of vectors in the "bin."

    Show information at user's terminal.

    Ask user if micro-plot is to be redisplayed.

ELSEIF (the display file contains a two-space scatter
        plot) THEN

    IF (output is sent to terminal) THEN

        ReDisplay the scatter plot (via CLSCAT).

    ENDIF

    Ask user to "draw" a box around the vector symbol(s)
    that are to be indexed.

    Obtain the vector id's of the vectors within the
    "box" from the DV file and display them at the
    user's terminal (or line printer).

    Ask user if scatter plot is to be redisplayed.
```

```
ELSEIF (the display file contains a two-space cluster
        plot) THEN

    IF (output is sent to terminal) THEN

        ReDisplay the cluster plot (via CLSCAT).

    ENDIF

    Ask user to specify (via the graphics cursor) which
    portion of the cluster plot is to be identified/
    counted.

    IF (user wants id of vectors in cluster plot, i.e.,
        'I' was typed when graphics cursor was
        displayed) THEN

        Obtain the vector id's of the vectors within the
        cluster plot grid from the DV file and display
        them at the user's terminal (or line printer).

    ELSE

        User is only getting the total number of vectors
        for each class present in the grid.

    ENDIF

    Ask user if cluster plot is to be redisplayed.

ENDIF
UNTIL (user says QUIT)

Put up the user's option list.

END
```

NOTE

PRTIDX must know which logical unit belongs to the terminal. Therefore, if the current terminal logical unit (1) of OLPARS changes, PRTIDX must be modified.

FILES:

CM - communications	HS - history
DI - display information	OLPARS option file
DV - display value	
PV - projection vector	Instrumentation
S1 - scratch 1	User output file

CLPARS Program Specifications
PRTIDX

REFERENCED BY:

OLPARS user.

SUBPROGRAMS REFERENCED:

CLOSE, CLOSFX, CLSCAT, CMGCDS, CMGOTH, CMGSCN, DIGDC, DIGENT, DIGHDI, DVGVEC, EQUALA, ERASE, FILPUT, GIN, INSINI, INSINT, INSRET, MENU, MICMAC, MOVE, OEXIT, OPENFX, OPENS, PRINTR, PROMPT, RCTNGL, S1GBC, TRMGET, TRMPUT

HISTORY:

designed by Steven Haehn August 14, 1978

programmed by Steven Haehn Feb. 10, 1981

PROGRAM NAME: PRTLOG

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

PRTLOG will give the user a printer listing of the description of any OLPARS logic tree.

PROGRAM USAGE:

User types in 'PRTLOG'.

ALGORITHM / NOTES:

Give user listing showing:

- Logic tree name.
- Design data set name.
- Dimensionality of the design data set.
- Total number of nodes used in logic tree.
- Number of data class nodes.
- Number of incomplete logic nodes.
- Structure of the logic tree.
- For each logic node:
  - Type of logic.
  - Parent Node.
  - Number of nodes immediately below the node.
  - Names of classes present (residing) at the node.
  - Apriori probabilities of the classes present (residing) at the node, if the node is the senior node.
  - Reassociated class name, if the node is a lowest node and the reassociated class name is different from the original design data set name.

- Reject strategy of the logic node, if one exists.
- Option creating the logic at the node.
- Parameters for the logic at the node.

FILES:

Block I/O	Record I/O
HS - history	terminal
CM - communications	instru.dat
LI - logic information	logdump. - printer listing
LV - logic value	
OPTION.OLP - option	

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLOSE, CMGLOG, CMGOTH, EQUALA, ERASE, EVALBM, FILPUT, GOPTNM, GP1PRT, GP2PRT, INSINI, INSINT, INSRET, LIGCAP, LIGCLP, LIGCNM, LIGDCN, LIGDSN, LIGLOG, LIGPRB, LIGSTR, MENU, NMVPRT, NNBPRT, OEXIT, OPENFX, OPENS, OPENTR, PROMPT, PRTSTR, PWSPRT, TRMGET, TRMPUT

SEE ALSO:

PRTDS

HISTORY:

designed by Steve Haehn October 2, 1978

programmed by Donna Morris October 1, 1981

PROGRAM NAME: PSORT

CATEGORY: Utility Subroutine

PROGRAM DESCRIPTION:

PSORT (permutation sort) creates a permutation vector with elements that point to the elements of the given vector in ascending order. For example, if the elements of the input vector are 4, 6, 2, 0, 9 (in that order) then the elements of the permutation vector are 4, 3, 1, 2, 5. Therefore, to access the elements of the given vector in numerical ascending order, the elements of the vector must be referenced like so,

```
Vec(PrmVec(1)) = Vec(4) = 0
Vec(PrmVec(2)) = Vec(3) = 2
Vec(PrmVec(3)) = Vec(1) = 4
Vec(PrmVec(4)) = Vec(2) = 6
Vec(PrmVec(5)) = Vec(5) = 9
```

PROGRAM USAGE:

```
CALL PSORT(VEC,PRMVEC,NELEM)
```

INPUT ARGUMENTS:

VEC - INTEGER ARRAY(NELEM); the given vector to be sorted

NELEM - INTEGER; the number of elements in the given vector

OUTPUT ARGUMENTS:

PRMVEC - INTEGER ARRAY(NELEM); the permutation vector described above

ALGORITHM / NOTES:

This sorting method is non-destructive to the given vector. It would generally be used where there are many separate arrays with information tied together via the array index, with one of the array elements used as a 'sort key'(i.e., the only way to simulate a multifield record in FORTRAN).

The sorting algorithm is the basically the 'Shell' sort as described in Knuth's 'The Art of Computer Programming' Vol. 3 'SORTING AND SEARCHING' p.84.

OLPARS Program Specifications
PSORT

REFERENCED BY:

MAKOPT

SUBPROGRAMS REFERENCED:

INSPGM, INSRET

HISTORY:

designed by Steven Haehn Aug. 20, 1979

programmed by Steven Haehn Aug. 21, 1979

PROGRAM NAME: PTRGNI

CATEGORY: UTILITY SUBROUTINE

PROGRAM DESCRIPTION:

PTRGNI (put region, convert elements from integer to real) writes a region (a series of entries) to a Logic Value file. The regions written represent specific entities created by the individual calling programs (i.e., group logic discriminant vectors).

PROGRAM USAGE:

```
CALL PTRGNI(FDLV,ELEMCT,LVETSZ,RGNTYP,TRMLNK,REGION,
            RGNPTR,NXTRGN)
```

INPUT ARGUMENTS:

FDLV - INTEGER; File descriptor of the LV file

ELEMCT - INTEGER; the number of elements to be written to the LV region.

LVETSZ - INTEGER; logic value entry size (found in Logic Information file header)

RGNTYP - INTEGER; type of region to be written, NEW or OLD (1 = NEW, 0 = OLD)

TRMLNK - LOGICAL; terminal link flag, when 'true' indicates that the last entry is supposed to contain th LV region termination link (0). Only has meaning when creating a NEW region.

REGION - INTEGER array(ELEMCT); information to be written into LV region

RGNPTR - INTEGER; pointer to the region to be written (Note, the value of this argument is needed for input only when the region being written is OLD)

OUTPUT ARGUMENTS:

RGNPTR - INTEGER; pointer to the region to be written (Note, this argument becomes an output argument only when the region being written is OLD)

NXTRGN - INTEGER; pointer to the next region following the current region (or an end-of-region link)

FILES:

LV - logic value

REFERENCED BY:

CREATLOG

SUBPROGRAMS REFERENCED:

GNLVAE, INSINT, INSLOG, INSPGM, INSRET, LVGLNK, LVPENT, LVPLNK

SEE ALSO:

GTRGNI, GTRGNR, PTRGNR

HISTORY:

designed by Steven Haehn March 5, 1981

programmed by Steven Haehn April 1, 1981

PROGRAM NAME: PTRGNR

CATEGORY: UTILITY SUBROUTINE

PROGRAM DESCRIPTION:

PTRGNR (put region of real elements) writes a region (a series of entries) to a Logic Value file. The regions written represent specific entities created by the individual calling programs (i.e., group logic discriminant vectors).

PROGRAM USAGE:

```
CALL PTRGNR(FDLV,ELEMCT,LVETSZ,RGNTYP,TRMLNK,REGION,
            RGNPTR,NXTRGN)
```

INPUT ARGUMENTS:

FDLV - INTEGER; File descriptor of the LV file

ELEMCT - INTEGER; the number of elements to be written to the LV region.

LVETSZ - INTEGER; logic value entry size (found in Logic Information file header)

RGNTYP - INTEGER; type of region to be written, NEW or OLD (1 = NEW, 0 = OLD)

TRMLNK - LOGICAL; terminal link flag, when 'true' indicates that the last entry is supposed to contain th LV region termination link (0). Only has meaning when creating a NEW region.

REGION - REAL array(ELEMCT); information to be written into LV region

RGNPTR - INTEGER; pointer to the region to be written (Note, the value of this argument is needed for input only when the region being written is OLD)

OUTPUT ARGUMENTS:

RGNPTR - INTEGER; pointer to the region to be written (Note, this argument becomes an output argument only when the region being written is OLD)

NXTRGN - INTEGER; pointer to the next region following the current region (or an end-of-region link)

FILES:

LV - logic value

REFERENCED BY:

CREATLOG

SUBPROGRAMS REFERENCED:

GNLVAE, INSINT, INSLOG, INSPGM, INSRET, LVGLNK, LVPENT, LVPLNK

SEE ALSO:

GTRGNI, GTRGNR, PTRGNI

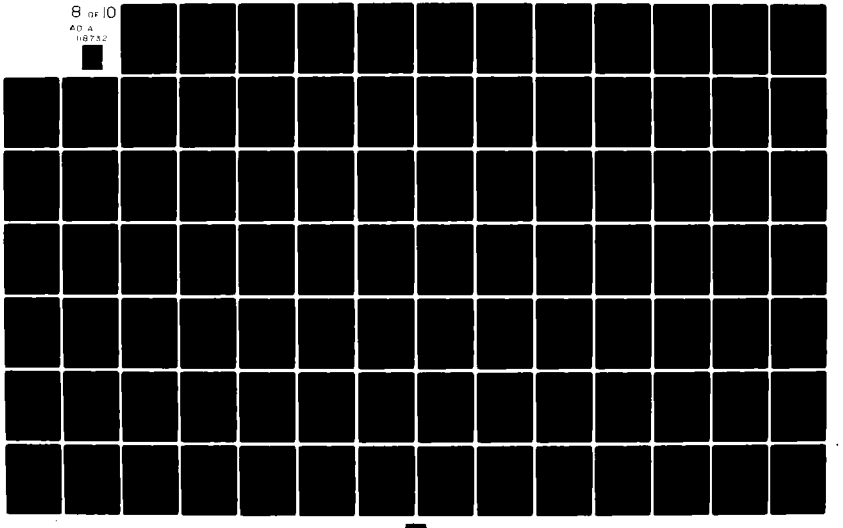
HISTORY:

designed by Steven Haehn March 5, 1981

programmed by Steven Haehn April 1, 1981

AD-A118 732 PAR TECHNOLOGY CORP NEW HARTFORD NY F/G 9/2
ON-LINE PATTERN ANALYSIS AND RECOGNITION SYSTEM. OLPARS VI. SOF--ETC(U)
JUN 82 S E HAEHN, D MORRIS
UNCLASSIFIED PAR-82-20 NL

8 OF 10
AD A
118732



PROGRAM NAME: PUTHST

CATEGORY: Utility subroutine (system dependent)

PROGRAM DESCRIPTION:

PUTHST alters the tally of the current instrumentation level's history file record and writes out that record to the history file.

PROGRAM USAGE:

CALL PUTHST

ALGORITHM / NOTES:

Essentially, the tally of the current history record is decremented by 1. If the tally becomes negative, it is set to zero.

REFERENCED BY:

INSRET

SUBPROGRAMS REFERENCED:

FLUSHB, FPUT, MAX

SEE ALSO:

GETHST

HISTORY:

designed by Steven Haehn Mar. 10, 1979

programmed by Steven Haehn Apr. 4, 1979

OLPARS Program Specifications
PVGENT

PROGRAM NAME: PVGENT

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

PVGENT gets an entry from the PV file for one- or two-space displays. It is passed the file descriptor of the the PV file, the entry number in PV to be read, and the number (NUM) of elements to get from that entry. In general, NUM=NDIM since the length of the projection vectors will always be the dimension of the data set (which is also the length of a PV file entry). However, when retrieving eigenvalues, NUM equals the number of measurements used, which may be smaller than NDIM.

The entry is placed in the real array VEC.

PROGRAM USAGE:

```
CALL PVGENT(FID,ENTNO,NUM,VEC)
```

INPUT ARGUMENTS:

```
  FID - INTEGER; the file descriptor of the PV file
ENTNO - INTEGER; the entry number
  NUM - INTEGER; the number of elements (coordinates)
                 to get
```

OUTPUT ARGUMENTS:

VEC - REAL array (MAXCLS); the projection vector

SUBPROGRAMS REFERENCED:

FGET,TRMPUT

HISTORY:

designed by Dave Birnbaum September 12, 1978

programmed by Donna Morris May 7,1979

PROGRAM NAME: PVGHDR

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

PVGHDR gets elements from a PV file header for one and two space displays. It is passed the file descriptor of the PV file and an information array (INFO). The information array is both an input and an output array. Non-negative values in INFO are replaced with the item referred to by the INFO array subscript (code). The codes are the same as in PVPHDR.

PROGRAM USAGE:

CALL PVGHDR(FID,INFO)

INPUT ARGUMENTS:

FID - INTEGER; file descriptor of the PV file
INFO - INTEGER array (PVNITM); array indicating which items to retrieve. this array is overwritten with the values of the items retrieved

FILES:

PV - projection vector

SUBPROGRAMS REFERENCED:

FGET,TRMPUT

SEE ALSO:

PVPHDR

HISTORY:

designed by Dave Birnbaum June 16, 1978

programmed by Donna Morris May 7,1979

PROGRAM NAME: PVGNAM

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

PVGNAM gets the current data set (treename and nodename) from the PV file header

PROGRAM USAGE:

CALL PVGNAM(FIDPV,NAMES)

INPUT ARGUMENTS:

FIDPV - INTEGER; the file descriptor of the PV file

OUTPUT ARGUMENTS:

NAMES - INTEGER array(TRELEN + NODLEN); treename and nodename of the current data set

FILES:

PV - projection vector file

REFERENCED BY:

Display File Access Routines

SUBPROGRAMS REFERENCED:

FGET, PACKW, TRMPUT, OEXIT

SEE ALSO:

PVPNAM

HISTORY:

designed by Donna Morris May 15, 1979

programmed by Donna Morris May 15, 1979

PROGRAM NAME: PVPENT

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

PVPENT puts an entry into the PV file for one- or two-space displays. It is passed the file descriptor of the PV file, the entry number to be used, the number (NUM) of elements to be put and a vector (VEC) to be placed in entry ENTNO.

PROGRAM USAGE:

CALL PVPENT(FID,ENTNO,NUM,VEC)

INPUT ARGUMENTS:

FID - INTEGER; file descriptor of the PV file

ENTNO - INTEGER; the entry number to be used

NUM - INTEGER; the number of elements to write (i.e. the length of the vector)

VEC - REAL arrau (MAXDIM); the projection vector

FILES:

PV - projection vector

SUBPROGRAMS REFERENCED:

FPUT

SEE ALSO:

PVGENT

HISTORY:

designed by Dave Birnbaum June 19, 1978

programmed by Donna Morris May 7,1979

PROGRAM NAME: PVPHDR

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

PVPHDR puts elements into a PV file header for one- and two-space displays. It is passed the file descriptor of the PV file and an information array (INFO). The INFO array subscript represents the code of an item that can be referenced by PVPHDR. A negative value in the array indicates that the item specified by the code (subscript) is not to be written. The codes are as follows:

CODE	INFORMATION	ELEMENT PV FILE HEADER
1	Display Code	1
2	NDIM	14
3	Number of measurements used	15
4	Number of Proj vectors	16
5	Pointer to First Proj. Vector	17
6	Pointer to Second Proj. Vector	18

PROGRAM USAGE:

CALL PVPHDR(FID,INFO)

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the PV file

INFO - INTEGER array (PVNITM); input integer data

SUBPROGRAMS REFERENCED:

FPUT

HISTORY:

designed by Dave Birnbaum June 16, 1978

programmed by Donna Morris May 7,1979

PROGRAM NAME: PVPNAM

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

PVPNAM writes the current data set (treename and nodename) to the PV file header

PROGRAM USAGE:

CALL PVPNAM(FIDPV,NAMES)

INPUT ARGUMENTS:

FIDPV - INTEGER; the file descriptor of the PV file

NAMES - INTEGER array (TRELEN + NODLEN); treename and nodename of the current data set

FILES:

PV - projection vector file

REFERENCED BY:

Display File Access Routines

SUBPROGRAMS REFERENCED:

FPUT, ITREAL

SEE ALSO:

PVGNAM

HISTORY:

designed by Donna Morris May 15, 1979

programmed by Donna Morris May 15, 1979

PROGRAM NAME: PWEVAL

CATEGORY: Logic Design Command

PROGRAM DESCRIPTION:

PWEVAL does a partial evaluation of PAIRWISE logic. The logic must first be created by FISHER. PWEVAL can also be used after FISHMOD, THRESHMOD, and OPTIMLMOD.

PROGRAM USAGE:

User types in 'PWEVAL'.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Get a list of PAIRWISE logic nodes. If there is more than one, ask the user for a node number from the list.

Perform a partial evaluation of the PAIRWISE logic and display a confusion matrix (PEPAIR).

Put up option list at user's terminal (MENU).

END

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CHKEXS, CLOSFX, CMGCDS, CMGLOG, CMGOTH, ERASE, GETLST, GTRGNI, GTSLOT, INSINI, INSRET, LIGCOP, LIGSTR, MENU, OEXIT, OPENFX, OPENTR, PEPAIR, PROMPT, TIGET, TRMGET, TRMPUT

SEE ALSO:

FISHER, FISHMOD, THRESHMOD, OPTIMLMOD

HISTORY:

designed by John W. Tenney Sept. 2, 1981

programmed by John W. Tenney Oct. 9, 1981

PROGRAM NAME: RANK

CATEGORY: Measurement Evaluation Command (subsidiary)

PROGRAM DESCRIPTION:

RANK checks the validity of the DI file. It asks the user to specify the type of ranking (and necessary parameters). It also sets up an array of values, sorts them, and stores in the DI file. It calls RODISP to cause a display of values.

PROGRAM USAGE:

User types in 'RANK'.

USER INTERACTION:

The user selects the rank type and parameters.

RANK TYPE	PARAMETERS
OVERALL	none
Measurement by class	class
Measurement by class pair	class 1 and class 2
Class by measurement	measurement number
Class pair by measurement	measurement number

FILES:

CM - communication
DI - display information
DV - display values
HS - history (indirectly)
OPTION - option (indirectly)

REFERENCED BY:

OLPARS user.

SUBPROGRAMS REFERENCED:

CDSCHK, CMGCDS, CMGOTH, DIGROH, DIPROE, DIPROH,
DVGROE, ERASE, INSINI, INSINT, INSRET, MENU, OEXIT,
OPENFX, PROMPT, RODCLS, RODISP, RSORTA, RSORTD,
TRMGET, TRMPUT

HISTORY:

designed by Kermit Klingbail April 10, 1978

programmed by Donna Morris January 15, 1981

PROGRAM NAME: RCTNGL

CATEGORY: Terminal I/O (graphic, system dependent)

PROGRAM DESCRIPTION:

RCTNGL displays a rectangle on the display screen according to the coordinates passed to it of the lower left corner and of the upper right corner. If the tic indicator is set to true, then tic marks will appear on the inside of the rectangle. The tic marks break the rectangle up into ten equal units on each border.

PROGRAM USAGE:

```
CALL RCTNGL(IX1,IY1,IX2,IY2,TICIND)
```

INPUT ARGUMENTS:

```
IX1    - INTEGER; the x coordinate of the lower
                  left corner
IY1    - INTEGER; the y coordinate of the lower
                  left corner
IX2    - INTEGER; the x coordinate of the upper
                  right corner
IY2    - INTEGER; the y coordinate of the upper
                  right corner
TICIND - INTEGER;  tic indicator, specifies whether or
                   not tic marks are to appear inside
                   the rectangle to be drawn.
```

ALGORITHM / NOTES:

The coordinates passed to RCTNGL are the actual display screen coordinates of the rectangle to be displayed.

FILES:

Instrumentation file

NC"D BY:

CLSCAT

OG AMS REFERENCED:

INSPGM, INSRET, LINSEG

OR' .

designed by Dave Tipton June 19, 1979

programmed by Dave Tipton June 19, 1979

OLPARS Program Specifications
RDISPLAY

PROGRAM NAME: RDISPLAY

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

RDISPLAY regenerates the most recent one-space, two-space or confusion matrix display.

PROGRAM USAGE:

User types in 'RDISPLAY'.

ALGORITHM / NOTES:

```
Erase screen and home cursor (ERASE).

Get display code from DI file header (DIGDC).

IF (display code = 1 or 2 for cluster or scatter) THEN

    CALL CLSCAT.

ELSEIF (display code = 5 or 6 for macro or micro) THEN

    CALL MICMAC.

ELSEIF (display code = 4 for cluster plot) THEN

    CALL CMDISP.

ELSE

    Print error message.

ENDIF

Put up option list at user's terminal (MENU).

END
```

FILES:

CM - communications	S1 - scratch 1
DI - display information	HS - history
DV - display value	Instrumentation
PV - projection vector	OLPARS option file

ERENCED BY:

OLPARS user

BPROGRAMS REFERENCED:

CLOSFX, CLSCAT, CMDISP, CMGCDS, CMGOTH, DIGDC, ERASE, INSINI, INSRET, MENU, MICMAC, OEXIT, OPENFX, TRMPUT

IS ORY:

designed by David Birnbaum September 20, 1978

programmed by Steven Haehn May 27, 1981

PROGRAM NAME: READB

CATEGORY: Utility subroutine (system dependent)

PROGRAM DESCRIPTION:

READB (read block) performs the actual disk read of all OLPARS fixed length record files. Under RSX-11M, this routine is an assembly routine. The type of reading being done is called a block read. For further information, see an RSX-11M I/O operations manual.

PROGRAM USAGE:

```
CALL READB(LUN,BLKNO,BUFFER,BUFSIZ,IOSB)
```

INPUT ARGUMENTS:

LUN - INTEGER; logical unit number of file being read

BLKNO INTEGER *4; an RSX-11M long integer representing the virtual block to be read

BUFSIZ - INTEGER; size (IN BYTES) of the virtual block to be read

OUTPUT ARGUMENTS:

BUFFER - INTEGER ARRAY(BUFSIZ / 2); the virtual block read is stored here

IOSB - INTEGER ARRAY(2); the I/O status block

IOSB(1) = File Control Services (FCS) error code in lower byte

IOSB(2) = actual number of bytes read (it will be equal to BUFSIZ unless an end-of-file read is attempted; in this case, IOSB(2) contains the 'short' read byte count. The next read will return an end-of-file code (-10) in the low byte of IOSB(1).)

ALGORITHM / NOTES:

A requirement for usage will be that the files to be read must be 'opened' with 'BUFFER COUNT = -1' in a DEC F4P open statement, or that a -1 is placed in the offset F.MBCT in the file descriptor block (FDB) of the file from an assembly 'open' program. (Refer to an RSX-11M FORTRAN IV user's guide.)

REFERENCED BY:

FGET, FPUT

SUBPROGRAMS REFERENCED:

$FCHNL - obtains FORTRAN FDB, when given a logical unit number
.READ - assembly 'block' read program
.WAIT - causes READB to wait until BUFFER is filled

DIAGNOSTICS:

If an invalid logical unit number is encountered, READB will abort.

SEE ALSO:

WRITEB

HISTORY:

designed by Steven Haehn Dec. 15, 1978

programmed by Steven Haehn Jan. 15, 1979

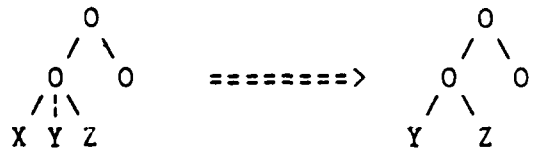
PROGRAM NAME: REARR

CATEGORY: Data Tree File Access Routine

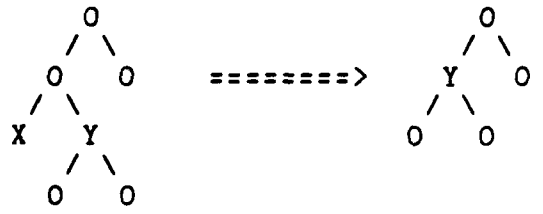
PROGRAM DESCRIPTION:

REARR rearranges the TI file so as to delete a lowest node from a data tree. It makes appropriate changes to the set of counts and pointers and the means and covariances for all affected nodes. When the node to be deleted has at least two siblings, the legality of the tree structure is not affected by the deletion of the node. But, when the deletion of the node leaves only one child at the parent, this is not a legal tree structure. At this point, the node is deleted and the remaining node is 'combined' with the parent node. The parent takes the remaining node's name. For example:

With 2 siblings, where X is the node to be deleted -



With 1 sibling, where X is the node to be deleted -



PROGRAM USAGE:

CALL REARR (FDTI,FDTV,ET,NODSLT,NDIM)

INPUT ARGUMENTS:

FDTI - INTEGER; the file descriptor of the TI file

FDTV - INTEGER; the file descriptor of the TV file

ET - INTEGER array (TABSIZ); the TI file entry table

NODSLT - INTEGER; the slot number of the node to be deleted

NDIM - INTEGER; the dimensionality of the current data set

ALGORITHM / NOTES:

Retrieve the TI file counts and pointers of node to be deleted and its parent node (TIGCAP).

IF (the node being has at least two siblings) THEN

Adjust the parent node's counts and pointers (TIPCAP).

Adjust the deleted node's sibling pointers to point around the deleted node (TIPCOP).

Adjust the first lowest node pointers of all ancestors in the tree which point to the deleted node (TIPCOP).

Adjust the low node link pointers and low node back pointers of all nodes pointing to the deleted node (TIPCOP).

Traverse the tree, adjusting the mean and covariance arrays of all ancestors of the deleted node (SBUPMC).

Traverse the tree, adjusting the number of lowest nodes beneath the present node, the number of vectors beneath the present node, and the number of children at the parent node of the node being deleted (FIXKLV).

Add the deleted node to the TI file free list (GARBND).

ELSE

Adjust the parent node's counts and pointers (TIPCAP).

Adjust the first lowest node pointers of all ancestors in the tree which point to the deleted node (TIPCOP).

Adjust the low node link pointers and low node back pointers of all nodes pointing to the deleted node (TIPCOP).

Change the parent's node name to the name of the remaining node to insure uniqueness within the data tree (TIGNAM, TIPNAM).

Traverse the tree, adjusting the mean and covariance arrays of all ancestors of the deleted node (SBUPMC).

Traverse the tree, adjusting the number of lowest nodes beneath the present node, the number of vectors beneath the present node, and the number of children at the parent node of the node being deleted (FIXKLV).

Add the deleted node and its remaing sibling node to the TI file free list (GARBND).

ENDIF

RETURN

END

SPECIAL NOTE

If there is only one node left below the senior node during the node deletion, that node will in essence become the senior node. This means the name of the node must be changed to '****' and all the vector display symbols in the tree vector file will have to be changed to '*'.

REFERENCED BY:

DDATANOD, DVEC

SUBPROGRAMS REFERENCED:

FIXKLV, GARBND, GNXTND, INSPGM, INSRET, SBUPMC, TIGCAP, TIGCOP, TIGNAM, TIPCAP, TIPCOP, TIPNAM, TVPCLS

HISTORY:

designed by David Birnbaum August 28, 1978

programmed by Jill King May 13, 1981

PROGRAM NAME: REASNAME

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

REASNAME displays a table containing lowest logic node numbers, original design data set class names and reassociated class names. The user may then change any reassociated names. The routine redisplays the table with the modifications and allows corrections. When the user is satisfied, the routine exits.

The reassociated name capability is useful in cases where a test data set is to be evaluated against logic designed on a tree whose class names are different from the test data set.

REASNAME operates only on a completed logic which must be the current logic.

PROGRAM USAGE:

User types in 'REASNAME'.

ALGORITHM / NOTES:

```
Get current logic from CM file (CMGLOG).

IF (logic is not complete) THEN

    Print error message.

ELSE

    Get list of lowest logic node numbers (GETLST).
    Eliminate reject nodes from the list.
    Get the original design data set class names
    and the reassociated class names for the nodes
    (LIGDCN).

    Erase screen and home cursor (ERASE).
    Display list on screen.
```

```
        REPEAT

            REPEAT

                Ask user for a logic node number and a
                new reassociated class name.

                Write new reassociated name into LI file
                (LIPDCN).

            UNTIL (logic node equals zero)

            Redisplay the table with modifications.

        UNTIL (user is satisfied).

    ENDIF

    Set the reassociated class names flag in the LI
    file header to indicate that there are reassociated
    class names (LIPCOP).

    Put up option list at user's terminal (MENU).

    END
```

FILES:

```
    CM - communications
    HS - history (indirectly used)
    INSTRU.DAT - instrumentation file (indirectly used)
    LI - logic information
    LL - logic list (indirectly used)
    OPTION.OLP - option file (indirectly used)
```

REFERENCED BY:

```
    OLPARS user
```

SUBPROGRAMS REFERENCED:

```
    CHARFL, CMGLOG, CMGOTH, CMGSCN, EQUALA, ERASE,
    GETLST, INSINI, INSRET, LENGA, LGLNOD, LIGDCN,
    LIPCOP, LIPDCN, MENU, OEXIT, OPENFX, OPENTR,
    PROMPT, TRMGET, TRMPUT
```

HISTORY:

```
    designed by     Dave Birnbaum    September 18, 1978

    programmed by   Donna Morris     July 8, 1981
```

PROGRAM NAME: REDRAW

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

REDRAW is used to put up an existing decision boundary at an OLPARS user's terminal after a display has been redisplayed as a result from a scale or class change.

PROGRAM USAGE:

User types in 'REDRAW'.

ALGORITHM / NOTES:

Obtain boundary pointers from DI file.

Obtain screen coordinates from CM file

Obtain boundaries from DV file.

Determine wheher both points of a segment are inside the 'window', 1st point in - 2nd out, 2nd point in - 1st out, or both points outside the display window.

IF (the decision boundary lies outside the user's display window) THEN

"Clip" off the portion of the boundary that is outside the window and extend the line segment to the border. Then draw the line segment.

ELSE

Extend the first and last segments of the boundary to the border and draw the decision boundary.

ENDIF

NOTE

"Window" refers to both one-space and two-space display windows.

FILES:

CM - communications file
DI - display information
DV - display value
Instrumentation file

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLOSFX, CMGSCN, DIGHDI, DIGMAX, DVGVEC, EXTLST,
EXT1ST, INSFLT, INSINI, INSINT, INSPGM, INSRET,
LINSEG, OEXIT, OPENFX, RXINT, RYINT, SCATR,
TEXT, TRMPUT

HISTORY:

designed by Steve Haehn September 27, 1978

programmed by Mark Maginn September 9, 1980

PROGRAM NAME: REDVEC

CATEGORY: UTILITY SUBROUTINE

PROGRAM DESCRIPTION:

REDVEC (reduce vector dimensionality) eliminates those measurements specified by the calling routine from a vector.

PROGRAM USAGE:

```
CALL REDVEC(VECTOR,MVEC,NDIM)
```

INPUT ARGUMENTS:

VECTOR - REAL array(150); the vector for which measurements are to be eliminated

MVEC - REAL array(150); the measurement vector - the subscript (index) in 'MVEC' represents the measurement number.
0 = eliminate the given meas. from 'VECTOR'
1 = include the given meas. in 'VECTOR'

OUTPUT ARGUMENTS:

VECTOR - REAL array(150); the vector after specified measurements have been eliminated

REFERENCED BY:

TRANSFORM

SUBPROGRAMS REFERENCED:

INSPGM, INSRET

HISTORY:

designed by Donna Morris March 10, 1981

programmed by Donna Morris March 10, 1981

PROGRAM NAME: RELUN

CATEGORY: Utility Subroutine

PROGRAM DESCRIPTION:

RELUN marks a logical unit number, found in the logical unit table, as being released from active use. RELUN references the 'Luntbl' array found in the CALUN (currently available logical unit numbers) common area.

PROGRAM USAGE:

CALL RELUN(LUN)

INPUT ARGUMENTS:

LUN - INTEGER; logical unit number to be released from use

ALGORITHM / NOTES:

This routine contains the common block

CALUN - a list of currently available logical unit numbers

REFERENCED BY:

CLOSE

SUBPROGRAMS REFERENCED:

TRMPUT for an error message

DIAGNOSTICS:

When an invalid logical unit number is encountered (i.e., it can't be found in the 'Luntbl'), RELUN will print an error message to that effect.

SEE ALSO:

GTLUN

HISTORY:

designed by Steven Haehn Feb. 15, 1979

programmed by Steven Haehn Mar. 9, 1979

PROGRAM NAME: RENAME

CATEGORY: OLPARS utility (system dependent)

PROGRAM DESCRIPTION:

RENAME changes a filename in an RSX11-M directory to another name.

PROGRAM USAGE:

```
INTEGER RENAME

N = RENAME (NAME1, NAME2, ERR)
```

INPUT ARGUMENTS:

NAME1 - LOGICAL *1 ARRAY; NAME TO BE CHANGED

NAME2 - LOGICAL *1 ARRAY; NEW NAME

OUTPUT ARGUMENTS:

RENAME - INTEGER; FUNCTION VALUE: 0 INDICATES SUCCESS, -1 INDICATES ERROR

ERR - INTEGER; RETURNED WHEN RENAME=-1, IT IS THE FILE CONTROL SERVICE (FCS) ERROR CODE

ALGORITHM / NOTES:

RENAME requests two logical units numbers via 'GTLUN' in order to obtain two file descriptor blocks via '$FCHNL'. One FDB stores filename information of the original file, and the other stores information about the new filename.
Note: the file to be renamed should be 'closed' prior to calling 'RENAME'. The reason is that on an unsuccessful rename, the file will automatically be closed by the system.

REFERENCED BY:

ORENAM

SUBPROGRAMS REFERENCED:

GTLUN, RELUN

HISTORY:

designed by David J. Tipton

programmed by David J. Tipton

PROGRAM NAME: RENAMT

CATEGORY: Level II File Manipulation Subroutine

PROGRAM DESCRIPTION:

RENAMT can be used for either of two purposes:

1). To rename an existing OLPARS tree (a tree can be renamed to the name of an existing tree only if privilege to delete the tree with the duplicate name is given to RENAMT (see FORCE argument)).

2). To replace the file code table entry of a tree's TI, LI, TV, or LV file with the file code table entry of another tree's corresponding TI, LI, TV, or LV file. The tree having an FCT entry used as a replacement will be deleted.

Value of tree TYPE:
1 - data tree
2 - logic tree

Value of ACCESS:
1 - replace the TI or LI FCT entry of a tree with another's and delete the other tree

2 - replace the TV or LV FCT entry of a tree with another's and delete the other tree

3 - rename both FCT entries of a tree and rename the tree entry in the TL or LL file; if another tree exists with the new name and same TYPE, see FORCE for course of action

Value of FORCE (used only when ACCESS = 3):

TRUE - delete an existing tree with the new name and same TYPE

FALSE - do not delete an existing tree with the new name and same TYPE

PROGRAM USAGE:

```
LOGICAL RENAMT

IF (RENAMT(NAME1,NAME2,TYPE,ACCESS,FORCE))
```

INPUT ARGUMENTS:

NAME1 - INTEGER ARRAY; name of tree to rename or to have one of its FCT entries replaced

NAME2 - INTEGER ARRAY; new name or name of tree having the replacement FCT entry (this tree will be deleted if ACCESS=1 or 2, or if it exists and ACCESS=3 and FORCE is TRUE)

TYPE - INTEGER; see program description
ACCESS - INTEGER; see program description
FORCE - LOGICAL; see program description

FILES:

TI - tree information
TV - tree vector
TL - tree list
LI - logic information
LV - logic value
LL - logic list
FCT - file code table

SUBPROGRAMS REFERENCED:

DELETR, INSCHR, INSINT, INSLOG, INSPGM, INSRET, LLGENT, LLGHDR, LLPENT, LLPHDR, LLSRCH, OCLOSE, OMOVE, OPENFX, ORENAM, TLGENT, TLGHDR, TLPENT, TLPHDR, TLSRCH, TRMPUT

DIAGNOSTICS:

RENAMT will return with a false value for three reasons:

1). The tree to be renamed (or to have an FCT entry replaced) does not exist.
2). The tree with the replacement FCT entry does not exist (ACCESS=1 or 2).
3). A tree already exists with the same name and TYPE but deletion of it is denied (ACCESS=3; FORCE=FALSE).

SEE ALSO:

DELETR

HISTORY:

designed by Steven E. Haehn August 29, 1980

programmed by David J. Tipton October 31, 1980

PROGRAM NAME: REPROJECT

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

REPROJECT enables a user to project a data set on different eigenvectors after an EIGV or GNDV projection routine has created a one-space or two-space display. A list of eigen values appears and the user is asked to choose one or two values.

PROGRAM USAGE:

User types in 'REPROJECT'.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Check display code (DIGDC) to make sure a one- or two-space display is stored.

Get OLPARS option number (CMGOTH) and make sure it is equal to the option number of one of the EIGV or GNDV commands. The option numbers of these eight routines can be gotten by using SETOPT.

Check current data set; must be equal to data set in DI header (CDSCHK).

Display set of eigenvalues from PV file (EIGPRT).

Ask user to choose one or two (SLTEIG).

Project the data (DSPROJ or LNPROJ).

Create display (CLSCAT or MICMAC).

Put up option list at user's terminal (MENU).

END

OLPARS Program Specifications
REPROJECT

FILES:

CM - communications	LI - logic information
DI - display information	TI - tree information
DV - display value	TV - tree value
PV - projection vector	HS - history
S1 - scratch 1	OLPARS option file
TL - tree list	LL - logic list

OLPARS instrumentation

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CDSCHK, CLOSFX, CLSCAT, CMGCDS, CMGLOG, CMGOTH, DIGDC, DIGHDI, DSPROJ, EIGPRT, EQUALA, ERASE, GCLIST, INSINI, INSRET, LIGCOP, LNPROJ, MENU, MICMAC, OEXIT, OPENFX, OPENTR, PROMPT, SETOPT, SLTEIG, TIGCOP, TIGET, TIGNAM, TRMGET, TRMPUT

HISTORY:

designed by David BirnBaum September 20, 1978

programmed by Steven Haehn February 18, 1981

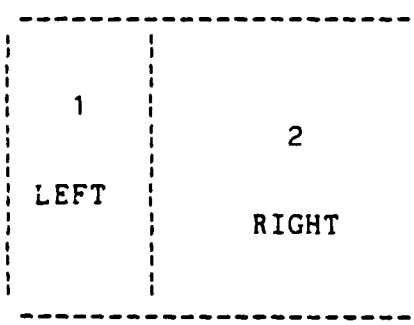
PROGRAM NAME: RESTRUCT

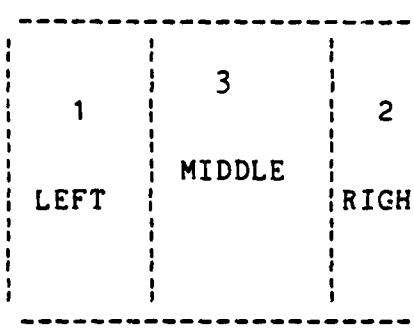
CATEGORY: Structure Analysis Command

PROGRAM DESCRIPTION:

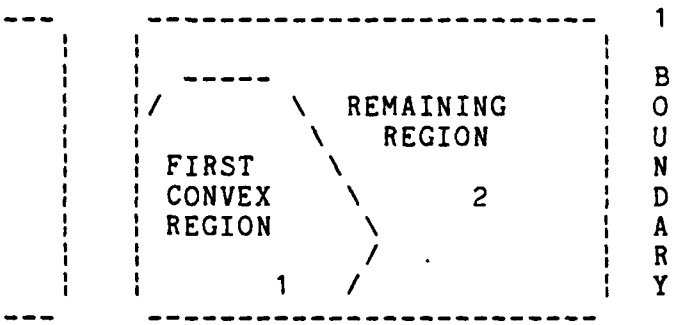
RESTRUCT (restructure data set) relabels a class of the current data set according to the boundary(s) drawn on a one-space or two-space plot. RESTRUCT renames the class using the information found in one of the four following displays:

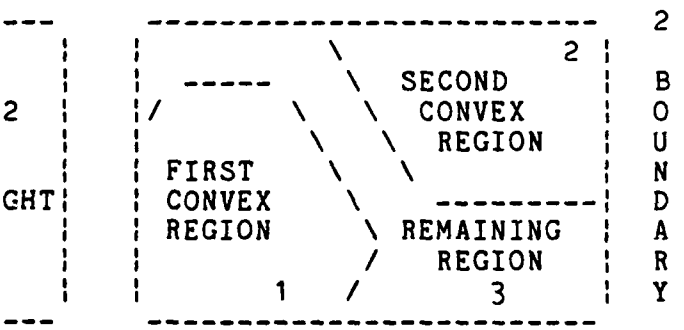
ONE-SPACE DISPLAY





TWO-SPACE DISPLAY





RESTRUCT presents a list of data classes displayed and asks the user to choose one. RESTRUCT then asks the user to input 2 or 3 unique four character node names. Temporary files are then created and the vectors of the data set being divided are sorted according to the drawn boundaries.

If any new data class contains no vectors, RESTRUCT will present a message informing the user of this, and then check to see if enough non-empty nodes are left to be legally added to the OLPARS data tree.

If the display is two-space, RESTRUCT will not divide a class correctly if:

(1) any boundary is not convex
(2) the so-called "convex point" is not on the convex side of a boundary
(3) two boundaries intersect within the range of the vectors being separated.

RESTRUCT checks to be sure that the projection being used has been created by a structure analysis routine.

RESTRUCT creates new TI file entries for the new classes and reinserts the vectors, now sorted by the new classes, into the TV file, using the exact set of entries from which the vectors came.

PROGRAM USAGE:

User types in 'RESTRUCT'.

USER INTERACTION:

See Algorithm

ALGORITHM / NOTES:

Call UIRSTR to

Make sure the projection being used has been created by a structure analysis routine (CMGOPT).

Check to see that a one- or two-space display is stored in the display files and that there are boundaries (DIGDC,DIGHDI).

Display the list of lowest nodes (OBTCLS) and prompt the user for the display symbol of one class to be restructured (PROMPT, TRMGET).

Prompt the user for the new lowest node names (PROMPT, TRMGET).

Check these names for legality (LGLNOD) and to make sure that they are unique within the tree (UNIQND).

Get the boundary line segment endpoints and the convex points for the boundary(s) from DV file (DVGVEC).

Open temporary files to store the vectors of the new classes (OPNTMP).

```
Get the projection vector(s) (PVGVEC).

IF (display is two-space) THEN

   DO i=1,the total number of line segments

      Compute a discriminant vector and a threshold
      value (DSCVTH).

   ENDDO

ENDIF

Get the TI file information for the class being
restructured (TIGCAP, TIGNAM).

DO i=1,number of vectors in the restructured class

   Get the next vector (TVGVEC).

   Determine which region the vector lies in
   (EV1SP, EV2SP).

   Place the vector in the proper temporary file
   (TVPVEC).

   Update a count if vectors in that temporary file.

ENDDO

Check for empty regions and display the proper message.

IF (the present class is not empty) THEN

   Get entry table slots and entry numbers in the TI file
   for the new nodes (GNXTAN).  If there are not enough
   print error message and return.

   Create TI entries for the new nodes (CLNODE).

   IF (this new node is not the first lowest node at this
      parent)  fix the number of children and lowest
      nodes which lie at this parent (FIXKLV).

ENDIF

Store the vectors from the temporary files in the TV
(TVPVEC).  Make sure to change the class symbols of the
vectors and to update the vector pointers of the new
nodes.  The vectors should be stored in the same set of
TV entries that they came from.
```

IF (not in excess measurement mode) THEN

Compute the means and covariances for the new nodes and store in the TI file entries (TIPMN, TIPCOV).

ENDIF

Put up option list at user's terminal (MENU).

END

FILES:

CM - communications file
DI - display information file
DV - display vector file
HS - history file
PV - projection vector file
TI - tree information file
TV - tree vector file
instrumentation file
option file

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLNODE, CLOSFX, CMNCOV, DIGHDI, DSCVTH, DVGVEC, ERASE, EV1SP, EV2SP, FIXKLV, GNXTAN, INSINI, INSINT, INSRET, MENU, OEXIT, OPNTMP, PVGENT, TIGCAP, TIGNAM, TIPCOV, TIPMN, TRMPUT, TVGVEC, TVPVEC, UIRSTR

HISTORY:

designed by David Birnbaum October 10, 1978

programmed by Jill King February 13, 1981

PROGRAM NAME: RODCLS

CATEGORY: Display File Access Routine

PROGRAM DESCRIPTION:

RODCLS looks through the ordered list of class display characters in the Display Information (DI) file and searches for the class symbol stored in 'DSCHAR'. If the class symbol is found, RODCLS returns an index to the display character, residing in the ordered list of class display characters, that matched the character stored in 'DSCHAR', otherwise RODCLS return a zero.

PROGRAM USAGE:

```
INTEGER RODCLS
        .
        .
        .
INDEX = RODCLS(FIDDI,DSCHAR)
```

INPUT ARGUMENTS:

FIDDI - INTEGER; the file descriptor of the DI file
DSCHAR - INTEGER; the class symbol for which to search

OUTPUT ARGUMENTS:

RODCLS - INTEGER; if 'DSCHAR' is found in the DI file header, RODCLS is the index into the ordered list of class display, otherwise RODCLS is zero

FILES:

DI - display information

REFERENCED BY:

RANK

SUBPROGRAMS REFERENCED:

DIGROH, EQUALA, INSPGM, INSRET

OLPARS Program Specifications
RODCLS

HISTORY:

designed by Kermit Klingbail April 10, 1978

programmed by Donna Morris January 29, 1981

PROGRAM NAME: RODISP

CATEGORY: Measurement Evaluation Routine

PROGRAM DESCRIPTION:

RODISP formats a tabular display and presents it to the user. The display contains the results of the various ranking options which are available to the user in conjunction with the measurement evaluation routines.

The discriminant values to be displayed will have been sorted, by RSORTA or RSORTD, and stored in entry 1 of the DI file. The rank option number, in element 17 of the DI file header, determines the title, header and line format (see Algorithm /Notes).

PROGRAM USAGE:

CALL RODISP(FDCM,FDDI,PMODE,PGMNAM)

INPUT ARGUMENTS:

FDCM - INTEGER; the file descriptor of the CM file

FDDI - INTEGER; the file descriptor of the DI file

PMODE - INTEGER; the prompt mode

PGMNAM - INTEGER array; the name of the calling program. this argument is used when there is more than 1 page. if PGMNAM = 0, the current option name in the CM file will be used instead.

USER INTERACTION:

If there are more lines to be displayed than fit on the display, the user is asked if he wishes to see the remaining values.

OLPARS Program Specifications
RODISP

ALGORITHM / NOTES:

RANK OPTION NUMBER	TITLE HEADER (I, J and N are variables stored in the rank option parameters)
1	overall ranking asterisk flag measurement number value best class best classpair
2	a measurement ranking for class I asterisk flag measurement number value
3	a measurement ranking for classpair I/J asterisk flag measurement number value
4	a class ranking for measurement N class symbol value
5	a classpair ranking for measurement N classpair value symbol

FILES:

DI - display information
CM - communications

REFERENCED BY:

RANK, DSCRMEAS, PROBCONF

SUBPROGRAMS REFERENCED:

CMGSCN, DIGROE, DIGROH, DIGROI, ERASE, INSINT, INSPGM, INSRET, PROMPT, TEXT, TRMGET, TRMPUT, WRTNOW, WRTODS

HISTORY:

designed by Kermit Klingbail August 17, 1978

programmed by Donna Morris January 1, 1981

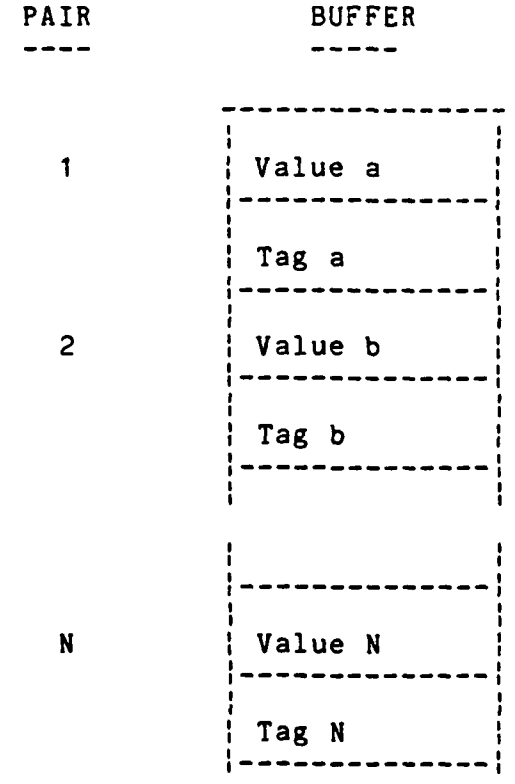
PROGRAM NAME: RSORTA

CATEGORY: Measurement Evaluation Routine

PROGRAM DESCRIPTION:

Upon entry, BUFFER contains NUMBER pairs of words. The first of each pair is a value and the second is a tag. RSORTA sorts these pairs (in ascending order) based on the value.

If FULSRT equals 'false', the minimum value/tag pair is returned as the first entry of BUFFER. If FULSRT equals 'true', the entire list of pairs is sorted using the partition sort ('quicksort') method as described by Booth and Chien in a book entitled "Computing: Fundamentals and Applications" (Hamilton Publishing Co., California, 1974, pp. 253-257), and by Knuth in a book "The Art of Computer Programming" ("Volume 3 / Sorting and Searching") (Addison - Wesley Publishing Co., Inc., 1973, pp. 114-123).



OLPARS Program Specifications
RSORTA

PROGRAM USAGE:

CALL RSORTA(BUFFER,NUMBER,FULSRT)

INPUT ARGUMENTS:

BUFFER - REAL array; 2 * NUMBER words long containing value/tag pairs to be sorted (see picture)

NUMBER - INTEGER; the number of value/tag pairs to be sorted

FULSRT - LOGICAL; if 'true', do a full sort; if 'false', return the minimum value only

OUTPUT ARGUMENTS:

BUFFER - REAL array; contains sorted value/tag pairs

REFERENCED BY:

DSCRMEAS, PROBCONF, RANK, UNION

SEE ALSO:

RSORTD

HISTORY:

designed by Donna Morris April 7, 1978

programmed by Donna Morris January 1, 1981

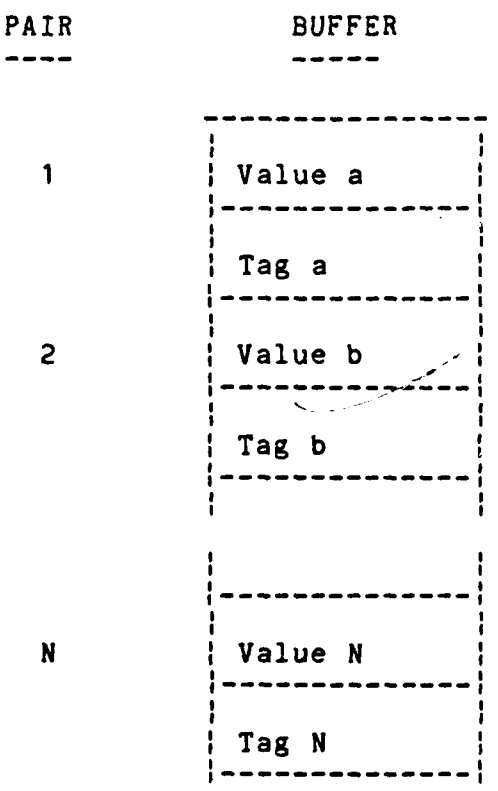
PROGRAM NAME: RSORTD

CATEGORY: Measurement Evaluation Routine

PROGRAM DESCRIPTION:

Upon entry, BUFFER contains NUMBER pairs of words. The first of each pair is a value and the second is a tag. RSORTD sorts these pairs (in descending order) based on the value.

If FULSRT equals 'false', the maximum value/tag pair is returned as the first entry of BUFFER. If FULSRT equals 'true', the entire list of pairs is sorted using the partition sort ('quicksort') method as described by Booth and Chien in a book entitled "Computing: Fundamentals and Applications" (Hamilton Publishing Co., California, 1974, pp. 253-257), and by Knuth in a book "The Art of Computer Programming" ("Volume 3 / Sorting and Searching") (Addison - Wesley Publishing Co., Inc., 1973, pp. 114-123).



OLPARS Program Specifications
RSORTD

PROGRAM USAGE:

CALL RSORTD(BUFFER,NUMBER,FULSRT)

INPUT ARGUMENTS:

BUFFER - REAL array; 2 * NUMBER words long containing value/tag pairs to be sorted (see picture)

NUMBER - INTEGER; the number of value/tag pairs to be sorted

FULSRT - LOGICAL; if 'true', do a full sort; if 'false', return the maximum value only

OUTPUT ARGUMENTS:

BUFFER - REAL array; contains sorted value/tag pairs

REFERENCED BY:

DSCRMEAS, PROBCONF, RANK, UNION

SEE ALSO:

RSORTA

HISTORY:

designed by Donna Morris April 7, 1978

programmed by Donna Morris January 1, 1981

PROGRAM NAME: RXINT

CATEGORY: Numerical Computation Algrithm

PROGRAM DESCRIPTION:

RXINT finds the intersection of any line with a horizontal line.

PROGRAM USAGE:

CALL RXINT(CRNTPT,BNDYY,BNDYX,YCONST,XLOC)

INPUT ARGUMENTS:

CRNTPT - INTEGER; the current point in both point arrays

BNDYY - REAL; the 'y' boundary point array

BNDYX - REAL; the 'x' boundary point array

YCONST - REAL; the 'y' screen coordinate, either the maxium or minimum 'y' value

OUTPUT ARGUMENTS:

XLOC - REAL; the 'x' intersection point

ALGORITHM / NOTES:

Calculate the 'x' intersection point

First we want to fin the slope of the line. The point slope form will be used.

$(Y1 - Y2) = (X1 - X2)M$ where M is the slope

This breaks down to:

$$\frac{Y1 - Y2}{X1 - X2} = M$$

The y - intercept form is now used to find the y intercept.

$Y = MX + B$ where B is the intercept

$$Y2 = MX2 + B$$
$$Y2 - MX2 = B$$

Now put the slope in for M:

$$Y2 - \frac{(Y1 - Y2)}{(X1 - X2)} X2 = B$$

Now find a common denominator:

$$Y2 - \frac{X2Y1 - X2Y2}{X1 - X2} = B$$

Multiply each term by the denominator:

$$\frac{Y2(X1 - X2)}{X1 - X2} - \frac{X2Y1 - X2Y2}{X1 - X2} = B$$

Distribute and add together:

$$\frac{Y2X1 - Y2X2 - X2Y1 + X2Y2}{X1 - X2} = B$$

$$\frac{Y2X1 - X2Y1}{X1 - X2} = B$$

Now fill in the equation Y = MX + B and solve for x by substituting M, B, and Y:

$$X = \frac{Y - B}{M}$$

$$X = \frac{YC - \frac{Y2X1 - Y1X2}{X1 - X2}}{\frac{Y1 - Y2}{X1 - X2}}$$

Now invert the denominatior and multiply:

$$X = \frac{YC(X1 - X2)}{Y1 - Y2} - \frac{Y2X1 - Y1X2}{X1 - X2} * \frac{X1 - X2}{Y1 - Y2}$$

Now cancel and add togther with common denominator:

$$X = \frac{YC(X1 - X2) - Y2X1 + Y1X2}{Y1 - Y2}$$

This is now the dereived equation for the 'x' intercept.

Check the value of the result and set output argument according to the size of the result

FILES:

Instrumentation file

REFERENCED BY:

REDRAW

SUBPROGRAMS REFERENCED:

INSFLT, INSINT, INSPGM, INSRET

HISTORY:

designed by Mark Maginn September 18, 1980

programmed by Mark Maginn September 18, 1980

PROGRAM NAME: RYINT

CATEGORY: Numerical Computation Algrithm

PROGRAM DESCRIPTION:

RYINT finds the intersection of any line with a vertical line.

PROGRAM USAGE:

```
CALL RYINT(CRNTPT,BNDYY,BNDYX,XCONST,YLOC)
```

INPUT ARGUMENTS:

CRNTPT - INTEGER; the current point in both point arrays

BNDYY - REAL; the 'y' boundary point array

BNDYX - REAL; the 'x' boundary point array

XCONST - REAL; the 'x' screen coordinate, either the maxium or minimum 'x' value

OUTPUT ARGUMENTS:

YLOC - REAL; the 'y' intersection point

ALGORITHM / NOTES:

Calculate the 'y' intersection point

First we want to fin the slope of the line. The point slope form will be used.

$(Y1 - Y2) = (X1 - X2)M$ where M is the slope

This breaks down to:

$$\frac{Y1 - Y2}{X1 - X2} = M$$

The y - intercept form is now used to find the y intercept.

$Y = MX + B$ where B is the intercept

$$Y - MX = B$$

Now put the slope in for M:

$$Y1 - \frac{(Y1 - Y2)}{(X1 - X2)} X1 = B$$

Now find a common denominator:

$$Y1 - \frac{X1Y1 - X1Y2}{X1 - X2} = B$$

Multiply each term by the denominator:

$$\frac{Y1(X1 - X2)}{X1 - X2} - \frac{X1Y1 - X1Y2}{X1 - X2} = B$$

Distribute and add together:

$$\frac{Y1X1 - Y1X2 - X1Y1 + X1Y2}{X1 - X2} = B$$

$$\frac{X1Y2 - X2Y1}{X1 - X2} = B$$

Now fill in the equation Y = MX + B by
by substituting M, B, and X:

$$Y = \frac{Y1 - Y2}{X1 - X2}(XC) + \frac{Y2X1 - Y1X2}{X1 - X2}$$

Add together because of common denominator:

$$Y = \frac{(Y1 - Y2)XC + Y2X1 - Y1X2}{X1 - X2}$$

This is now the dereived equation for the 'y'
intercept.

Check the value of the result and set output
argument according to the size of the result

FILES:

Instrumentation file

OLPARS Program Specifications
RYINT

REFERENCED BY:

REDRAW

SUBPROGRAMS REFERENCED:

INSFLT, INSINT, INSPGM, INSRET

HISTORY:

designed by Mark Maginn September 17, 1980

programmed by Mark Maginn September 17, 1980

PROGRAM NAME: S1CRDV

CATEGORY: Structure Analysis Command

PROGRAM DESCRIPTION:

S1CRDV allows a user to project the current data set onto one coordinate. A one-space display is created at the terminal. (NOTE, this program can be used in 'excess measurement mode.)

PROGRAM USAGE:

User types 'S1CRDV'.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Ask user to select one projection coordinate

Enter this coordinate into the DI file.

Open display files.

Initialize DI, DV, PV and S1 file headers (INITD).

For each vector, store the coordinate in DV file and set up DI file entries (CRPROJ).

Create a micro or macro plot (MICMAC).

Put up option list at user's terminal (MENU).

END

FILES:

CM - communications
DI - display information
DV - display value
PV - projection vector
S1 - scratch 1
TL - tree list
HS - history
TI - tree information
TV - tree vector
OLPARS option file
Instrumentation

REFERENCED BY:

OLPARS user

OLPARS Program Specifications
S1CRDV

SUBPROGRAMS REFERENCED:

CLOSFX, CMGCDS, CMGOTH, CRPROJ, DIPHDI, ERASE,
INITD, INSINI, INSRET, MENU, MICMAC, OEXIT,
OPENFX, OPENTR, PROMPT, PVPENT, PVPHDR, TIGCOP,
TIGET, TRMGET, TRMPUT

SEE ALSO:

S2CRDV

HISTORY:

designed by David Birnbaum August 24, 1978

programmed by Steven Haehn Oct. 16, 1980

PROGRAM NAME: S1EIGV

CATEGORY: Structure Analysis Command

PROGRAM DESCRIPTION:

S1EIGV projects the current data set onto a user-chosen eigenvector.

PROGRAM USAGE:

User types in 'S1EIGV'.

USER INTERACTION:

Done in subroutines.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Initialize DI, DV, PV, and S1 file headers (INITD).

Eliminate any measurements (ELIMEA).

Compute eigenvalues and eigenvectors and insert them into PV file (EIGNPV).

Ask user if he requires a printout of the eigenvalues and eigenvectors (EIGPRT).

Ask user to select one eigenvalue for use in the projection (SLTEIG).

Project data set onto corresponding eigenvector and store results in DV file (D3PROJ).

Create a one-dimensional micro or macro plot (MICMAC).

Put up option list at user's terminal (MENU).

FILES:

CM - communications file
DI - display information
DV - display vector
PV - projection vector
S1 - scratch 1
TI - tree information
TV - tree vector

OLPARS Program Specifications
S1EIGV

EIGPRT - eigenvalue listing

REFERENCED BY:

OLPARS user.

SUBPROGRAMS REFERENCED:

DSPROJ, EIGPR'' EIGNPV, ERASE, INITD
MENU, MICMAC

HISTORY:

designed by David A. Birnbaum July 19, 1978

programmed by David J. Tipton December 4, 1980

PROGRAM NAME: S1GBC

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

S1GBC gets the bin counts for a class from the S1 file. (ONE SPACE PROJECTIONS ONLY)

PROGRAM USAGE:

```
CALL S1GBC(FDS1,ENTNO,NBINS,BINCNT)
```

INPUT ARGUMENTS:

FDS1 - INTEGER; the file descriptor of S1
NBINS - INTEGER; the number of bins
ENTNO - INTEGER; the entry number in S1

OUTPUT ARGUMENTS:

BINCNT - REAL array (NBINS); the bin counts for a single class

FILES:

S1 - scratch 1

REFERENCED BY:

MICROP, MACROP

SUBPROGRAMS REFERENCED:

FGET, INSPGM, INSRET, OEXIT, TRMPUT

HISTORY:

designed by David Birnbaum July 7, 1978

programmed by Steven Haehn Oct. 1, 1980

OLPARS Program Specifications
S1GHDR

PROGRAM NAME: S1GHDR

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

S1GHDR retrieves header elements of the S1 file when set up for one-space displays. S1GHDR is passed the file descriptor of the S1 file and an information array (INFO). The information array is both an input and an output array. Non-negative values in INFO are replaced with the item referred to by the INFO array subscript (code). The codes are the same as in S1PHDR.

PROGRAM USAGE:

CALL S1GHDR(FID,INFO)

INPUT ARGUMENTS:

FID - INTEGER: the file descriptor of S1

INFO - INTEGER array (S1NITM); array indicating which items to retrieve. this array is overwritten with the values of the items retrieved

FILES:

S1 - scratch 1

SUBPROGRAMS REFERENCED:

FGET,TRMPUT

HISTORY:

designed by Dave Birnbaum July 21, 1978

programmed by Donna Morris May 17,1979

PROGRAM NAME: S1PBC

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

S1PBC puts the bin counts for a class into the S1 file. (ONE SPACE PROJECTIONS ONLY)

PROGRAM USAGE:

CALL S1PBC(FDS1,ENTNO,NBINS,BINCNT)

INPUT ARGUMENTS:

FDS1 - INTEGER; the file descriptor of S1

NBINS - INTEGER; the number of bins

ENTNO - INTEGER; the entry number in S1

BINCNT - REAL array (NBINS); the bin counts for a single class

FILES:

S1 - scratch 1

REFERENCED BY:

MICROP, MACROP

SUBPROGRAMS REFERENCED:

FPUT, INSPGM, INSRET

HISTORY:

designed by David Birnbaum July 21, 1978

programmed by Steven Haehn Oct. 1, 1980

PROGRAM NAME: S1PHDR

CATEGORY Level II File Access Subroutine

PROGRAM DESCRIPTION:

S1PHDR stores elements in the one-space formatted S1 file header. S1PHDR is passed the file descriptor of the S1 file and an information array (INFO). The INFO array subscript represents the code of an item that can be referenced by S1PHDR. A negative value in the array indicates that the item specified by the code (subscript) is not to be written. The codes are as follows.

CODE	INFO INSERTED INTO
1	Display Code
2	OLPARS Option Number
3	Number of Bins
4	Maximum Bin Count

PROGRAM USAGE:

CALL S1PHDR(FIDS1,INFO)

INPUT ARGUMENTS:

FIDS1 - INTEGER; the file descriptor of S1

INFO - INTEGER array (S1NITM); the information to be stored

FILES:

S1 - scratch 1

SUBPROGRAMS REFERENCED:

FPUT

HISTORY:

designed by Dave Birnbaum July 21, ,1978

programmed by Donna Morris May 7,1979

PROGRAM NAME: S2CRDV

CATEGORY: Structure Analysis Command

PROGRAM DESCRIPTION:

S2CRDV allows a user to project the current data set onto two coordinates. A two-space display is created at the terminal. This command can run in excess measurement mode.

PROGRAM USAGE:

User types in 'S2CRDV'.

USER INTERACTION:

The user is requested to choose two coordinate (measurement) numbers to be used.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Ask user to select two coordinates to project on (these are measurement numbers).

Initialize DI, DV, PV and S1 file headers (INITD).

For each vector in each lowest node, store those coordinates that the user has specified in the DV file and set up the DI file entries (CRPROJ).

Set up the DI and PV file headers.

Create a scatter or cluster plot (CLSCAT).

Put up option list at user's terminal (MENU).

END

OLPARS Program Specifications
S2CRDV

FILES:

CM - Communication File
DI - Display Information File
DV - Display Vector File
HS - History File (Indirectly Used)
OP - Option File (Indirectly Used)
PV - Projection Vector File
S1 - Scratch 1 File
TI - Tree Information File
TL - Tree List File (Indirectly Used)
TV - Tree Vector File

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLOSFX, CLSCAT, CMGCDS, CMGOTH, CRPROJ, DIPHDI, ERASE,
INITD, INSINI, INSRET, MENU, OEXIT, OPENFX, OPENTR,
PROMPT, PVPENT, PVPHDR, TIGCOP, TIGET, TRMGET, TRMPUT

HISTORY:

designed by Dave Birnbaum September 27, 1978

programmed by Donna Morris September 10, 1980

PROGRAM NAME: S2EIGV

CATEGORY: Structure Analysis Command

PROGRAM DESCRIPTION:

S2EIGV projects a data set onto a pair of eigenvectors for use in structure analysis. This program is essentially a sequence of calls to subprograms that carry out the work.

PROGRAM USAGE:

User types in 'S2EIGV'.

USER INTERACTION:

Done in the subprograms.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Set option in communications file and initialize DI, DV, PV and S1 file headers (INITD).

Eliminate any measurements? (ELIMEA)

Compute eigenvalues and eigenvectors and insert them into the PV file (EIGNPV).

Ask user if he desires a printout of the eigenvalues and eigenvectors (EIGPRT).

Ask user to select two eigenvalues for use in the projection (SLTEIG).

Project data set onto eigenvectors corresponding to selected eigenvalues and store results in the DV file (DSPROJ).

Create a scatter or cluster plot (CLSCAT).

Put up option list at user's terminal (MENU).

REFERENCED BY:

OLPARS user.

OLPARS Program Specifications
S2EIGV

SUBPROGRAMS REFERENCED:

CLOSFX,CLSCAT,CMGCDS,DIPHDI,DSPROJ,DVPHDR,EIGNPV,
EIGPRT,ELIMEA,ERASE,INITD,MENU,OEXIT,OPENFX,OPENTR,
PVPHDR,SLTEIG,TIGCOP,TIGET,TRMPUT

HISTORY:

designed by Dave Birnbaum June 26, 1978

programmed by Donna Morris May 7,1979

PRCGRAM NAME: S2FSHP

CATEGORY: Structure Analysis Command

PROGRAM DESCRIPTION:

S2FSHP projects the selected data set on two Fisher directions which correspond to two pairs of data classes within the selected data set.

PROGRAM USAGE:

User types 'S2FSHP'.

USER INTERACTION:

The user is asked for the following information:

1. The first class pair.
2. The second class pair.
3. Scatter or covariance matrix option.
4. Whether or not to eliminate some measurements from the computation, and the numbers of these measurements.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Call UIS2FS to

Initialize DI, DV, PV and S1 file headers (INITD).

Display the list of lowest nodes from the current data set (SELCLS).

Request the first pair of nodes (SELCLS).

Request the second pair of nodes (SELCLS).

Scatter or covariance matrix option? (PROMPT).

Eliminate any measurements? (ELIMEA)

Compute Fisher discriminant for first pair of classes (DCRIM).

Compute Fisher discriminant for second pair of classes (DCRIM).

Orthogonalize the two discriminants.

Store the discriminants in the PV file making sure the pointers to the projection vectors are correct (PVPENT).

Project the data set (DSPROJ).

Moddfy the display flag symbols (MODDFS).

Create the two-space display (CLSCAT).

Put up option list at user's terminal (MENU).

END

FILES:

CM - communications file
DI - display information file
DV - display vector file
HS - history file
PV - projection vector
TI - tree information file
TV - tree vector file
instrumentation file
option file

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLSCAT, CMGCDS, DCRIM, DIPHDI, DSPROJ, ERASE, INSINI, INSINT, INSRET, MENU, MODDFS, OEXIT, OPENFX, PVPENT, TRMPUT, UIS2FS, $SQRT

HISTORY:

designed by David Birnbaum August 17, 1978

programmed by Jill King November 3, 1980

PROGRAM NAME: SAVMAT

CATEGORY: TRANSFORMATION ROUTINE

PROGRAM DESCRIPTION:

SAVMAT may be called by Transformation Commands (NORMXFRM and EIGNXFRM) to save a transformation matrix that was used in a normalization or an eigenvector transformation.

PROGRAM USAGE:

CALL SAVMAT(FDSM,PMODE,TYPE,NDIM,MATRIX)

INPUT ARGUMENTS:

FDSM - INTEGER; the file descriptor of the SM file

PMODE - INTEGER; the prompt mode

TYPE - INTEGER; the type of transformation which was performed to get the resulting matrix. a -1 indicates a normalization transformation; a positive value represents the number of eigenvalues used in an eigen transformation.

NDIM - INTEGER; the dimensionality of the vectors in the matrix to be saved

MATRIX - REAL(50,50); the matrix to be saved

USER INTERACTION:

The user is asked if s(he) wants to save a transformation matrix. If the user responds with 'Y' (for yes), s(he) is then asked to assign a name to the transformation matrix to be saved. The matrix is then stored in the SM file and all the links and header information are updated. If the user indicates that s(he) does not want to save a matrix, SAVMAT returns to the calling program. It is therefore assumed that calling SAVMAT will be the last thing a Transformation Command does before calling the menu.

OLPARS Program Specifications
SAVMAT

FILES:

SM - Saved Matrix File

REFERENCED BY:

EIGNXFRM, NORMXFRM

SUBPROGRAMS REFERENCED:

EQUALA, GNSMAE, IDCHK, INSINT, INSPGM, INSRET, PROMPT
SMGHDR, SMGLNK, SMGMDS, SMPENT, SMPHDR, SMPLNK, SMPMDS
TRMGET, TRMPUT

DIAGNOSTICS:

If there is no available space in the SM file, SAVMAT will give an error message and return to the calling routine. If the user enters an invalid matrix name, or if the name entered already exists in the SM file, SAVMAT will give an error message and prompt the user for another name.

HISTORY:

designed by Donna Morris March 15, 1981

programmed by Donna Morris March 15, 1981

PROGRAM NAME: SBUPMC

CATEGORY: Utilities Subroutine

PROGRAM DESCRIPTION:

SBUPMC traverses a data tree, 'subtracting' the mean and covariance arrays of the deleted node from all of the node's ancestors. Also, the total number of vectors at the ancestors is adjusted. This routine is used when a data tree node is deleted.

PROGRAM USAGE:

CALL SBUPMC(FDTI,ET,NODSLT)

INPUT ARGUMENTS:

FDTI - INTEGER; the file descriptor of the TI file

ET - INTEGER array (TABSIZ); the TI file entry table

NODSLT - INTEGER; the slot number of the node whose mean and covariance arrays must be subtracted from its ancestor nodes

FILES:

TI - tree information file
instrumentation file

REFERENCED BY:

REARR

SUBPROGRAMS REFERENCED:

INSINT, INSPGM, INSRET, SUBMNC, TIGCOP, TIGCOV
TIGHDR, TIGMN

HISTORY:

designed by Jill King May 13, 1981

programmed by Jill King May 13, 1981

OLPARS Program Specifications
SCALRET

PROGRAM NAME: SCALRET

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

SCALRET allows the user, upon completion of one or several zooming operations, to return to the display with the original x- and y- ranges.

PROGRAM USAGE:

User types in 'SCALRET'.

ALGORITHM / NOTES:

Get original min,max values from DI header and store them as current min,max values (DIGMAX, DIPMAX).

Get the display code, zoom flag, and type of scaling flag from the DI header (Make sure code represents a one- or two-space display.)

Set display flag in DV header to 0 (i.e. new screen coordinates must be computed)(DVPHDR).

Set the zoom flag to no - zoom.

Set the type of scaling flag to equal square.

IF (the old display code was for a two-space display)

    Put up two-space display (CLSCAT).

ELSE

    Obtain the prompt flag

    Put up a one-space display (MICMAC).

ENDIF

Put up option list at user's terminal (MENU).

END

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLOSFX, CLSCAT, CMGOTH, DIGHDI, DIGMAX, DIPHDI,
DIPMAX, DVPHDR, ERASE, INSINI, INSINT, INSRET,
MENU, MICMAC, OEXIT, OPENFX, TRMPUT

HISTORY:

designed by David Birnbaum August 24, 1978

programmed by Mark Maginn October 9, 1980

OLPARS Program Specifications
SCALZM

PROGRAM NAME: SCALZM

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

SCALZM determines whether the display is a one or two space display, then calls the corresponding calculation routine to do the zooming if the diplay is a valid one or two space display.

PROGRAM USAGE:

User types in 'SCALZM'.

USER INTERACTION: NONE

ALGORITHM / NOTES:

Obtain the display code, dimensionality, number of bins, scale type, and the status of hte zoom flag from the DI header (DIGHDI).

Obtain screen coordinate information (DIGMAX).

IF (display code indicates that it is not a one - or two space display) THEN

Print message and exit the program.
ELSEIF(display code indicates that it is a two - space display) THEN

call two space calculation routine (ZOM2SP).

ELSE(it was a one space display)

call one space calculation routine (ZOM1SP).

ENDIF

Put up option list at user's terminal (MENU).

END

FILES:

CM - communications file
DI - display information
DV - display value
PV - projection file
S1 - scracth 1 file
Instrumentation file
History file
Option file

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLOSFX, CMGSCN, DIGHDI, DIGMAX, INSFLT,INSINI, INSINT
INSRET, MENU, OEXIT, TRMPUT, ZOM1SP, ZOM2SP

HISTORY:

designed by Steve Haehn October 3, 1978

programmed by Mark Maginn October 3, 1980

PROGRAM NAME: SCAT

CATEGORY: Terminal Display Routine

PROGRAM DESCRIPTION:

After a data set or a set of vectors at a logic node is projected on to a pair of projection vectors, SCAT is called to produce a scatter plot. SCAT computes the scatter screen coordinates of each vector, stores them in the DV file and displays the class symbol of that vector at the terminal screen.

PROGRAM USAGE:

```
CALL SCAT(FIDDI,FIDDV,ISCREN)
```

INPUT ARGUMENTS:

FIDDI - INTEGER; the file descriptor of the DI file

FIDDV - INTEGER; the file descriptor of the DV file

ISCREN - INTEGER array (SC2NUM); the set of two-space screen coordinates

ALGORITHM / NOTES:

Get original min/max values, number of classes, and type of scaling from DI file (DIGMAX,DIGHDI).

Compute the X and Y ranges (depending on whether the type of scaling is SQUARE or RECTANGULAR) to be used in the calculation of screen coordinates.

Rewrite the current min/max values so that they reflect the type of scaling being used.

IF (screen coordinate flag from DV header is off) THEN

Obtain pointers to decision boundary points (DIGHDI)

IF (there are any decision boundaries) THEN

Recompute the screen coordinates for the boundary.

ENDIF

```
DO WHILE (there are more classes)

    Get next class from DI file (DIGENT).

    Compute scatter screen coordinates for the
    mean vector and store in DV.

    IF (class is ON) THEN

        DO WHILE (there are more vectors in
            the class)

            Get next vector.

            Compute and store in DV the scatter
            screen coordinates.

            IF (both screen coordinates are > 0)
            THEN

                Send display symbol to terminal via
                MARK

            ENDIF

        ENDDO

    ELSE

        DO WHILE (there are more vectors in
            the class)

            Get next vector.

            Compute and store in DV the scatter
            screen coordinates.

        END DO

    END IF

ENDDO
```

```
        ELSE

            DO WHILE (there are more classes)

                Get next class from DI file.

                IF (class is ON)  THEN

                    DO WHILE (there are more vectors in)
                        the class)

                        Get next vector from DV.

                        IF (both screen coordinates are > 0)
                        THEN

                            Send display symbol to terminal
                            via MARK

                        ENDIF

                    ENDDO

                ENDIF

            ENDDO

        ENDIF

        Set screen coordinate flag to ON.

        RETURN
```

FILES:

DI - display information
DV - display value

REFERENCED BY:

CLSCAT

SUBPROGRAMS REFERENCED:

DIGENT, DIGMAX, DVGHDR, DVGVEC, DVPHDR, DVPVEC, MARK

SEE ALSO:

CLUS

HISTORY:

designed by David Birnbaum July 17, 1978

programmed by David Tipton MAY 30, 1979

OLPARS Program Specifications
SCATR

PROGRAM NAME: SCATR

CATEGORY: Numerical Computation Algorithm

PROGRAM DESCRIPTION:

SCATR computes a new screen coordinate from a given data value. If the projected value is outside of the 'window', 'scatr' is set to zero, otherwise 'scatr' obtans a valid screen coordinate value.

PROGRAM USAGE:

```
INTEGER SCATR
     .
     .
     .
N = SCATR (PV, MINV, MAXV, RANGE, WIDTH, BCOORD)
```

INPUT ARGUMENTS:

PV . - REAL; projected value

MINV - REAL; minimum projected value that can be dislayed

MAXV - REAL; maximum projected value that can be displayed

RANGE - REAL; the range to use for scaling

WIDTH - REAL; display units of the axis upon which the screen coordinate being computed lies

BCOORD - REAL; data coordinate (x or y) of bottom of display rectangl plus 1

ALGORITHM / NOTES:

Find where the projection value lies

IF(pv .LT. minimum .OR. pv .GT. maximumM) THEN

'scatr' = zero

```
        ELSEIF( range equals zero) THEN
           'scatr' = bottom coord. + width
        ELSE
           compute the scatter plot screen coordinate value
        ENDIF
```

REFERENCED BY:

REDRAW, CLUS, SCAT, CNT1SP

SUBPROGRAMS REFERENCED:

INSFLT, INSINT, INSPGM, INSRET

HISTORY:

designed by David Tipton May 30, 1979

programmed by David Tipton May 30, 1979

PROGRAM NAME: SEARCH

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

SEARCH searches the file code table for a given filename. If the filename is found, the file code (Fcd) is returned. Otherwise, if the filename is not found, Fcd is set to zero.

PROGRAM USAGE:

```
LOGICAL SEARCH
       .
       .
       .
IF (SEARCH(FID,NAME,FCD))
```

INPUT ARGUMENTS:

FID - INTEGER; file descriptor of the file code table

NAME - LOGICAL*1 ARRAY; filename for which to search

OUTPUT ARGUMENTS:

FCD - INTEGER; the file code of the entry 'NAME' (if found) in the file code table

ALGORITHM / NOTES:

SEARCH returns a true value if the given filename exists in the file code table, and a false value if the filename does not exist in the file code table. The file code table must be opened by the calling program.

FILES:

FCT - File Code Table

REFERENCED BY:

OCREAT

SUBPROGRAMS REFERENCED:

FCGHDR ,FCGENT, EQUALS

HISTORY:

designed by Donna Morris March 8,1979

programmed by Donna Morris March 8,1979

OLPARS Program Specifications
SELCLS

PROGRAM NAME: SELCLS

CATEGORY: Utility Subroutine

PROGRAM DESCRIPTION:

SELCLS sends to the terminal a list of node names which it labels 'CLASS SELECT LIST'. It then accepts from the user his choice of classes, entered as a string of class symbols. SELCLS verifies that each symbol entered exists in the 'CLASS SELECT LIST'. It then outputs a list of indices that shows the position within the 'CLASS SELECT LIST' of each node name that corresponds with a chosen class symbol.

PROGRAM USAGE:

```
INTEGER SELCLS

REPLY = SELCLS(NAMLST, NUMNAM, INDLST, NUMIND)
        note: SELCLS returns with the same value
        received from its call to TRMGET, i. e.
        QUIT, HELP,  or '1' if successful.
```

INPUT ARGUMENTS:

NAMLST - INTEGER ARRAY (2 DIMENSIONAL); (i. e. NAMLST(NODLEN, MAXCLS)) the list of node names to be displayed as the 'CLASS SELECT LIST'

NUMNAM - INTEGER; the number of node names in NAMLST

OUTPUT ARGUMENTS:

INDLST - INTEGER ARRAY; (i.e. INDLST(MAXCLS)) the list of indices returned by SELCLS

NUMIND - INTEGER; the number of indices returned in INDLST

USER INTERACTION:

Assume the 'CLASS SELECT LIST' is as follows:
axyz buvw crst dopq elmn fijk gfgh

The user has a choice of three ways of entering class symbols:
1). a string of class symbols (e.g. 'abdfg')
2). a string of class symbols preceded by a minus sign meaning 'I want every node in the 'CLASS SELECT LIST' except these' (e.g. '-ce' equals 'abdfg')
3). a star (a '*' with no other characters), represents all nodes in the 'CLASS SELECT LIST' (e.g. '*' equals 'abcdefg')

ALGORITHM / NOTES:

Print out the 'CLASS SELECT LIST' on the terminal (TRMPUT).

Accept user input (TRMGET).

Return all indices if '*' is the only character.

Check for '-'.

Verify that symbols entered belong to the 'CLASS SELECT LIST'.

IF (a minus sign was NOT entered) THEN

   Match user input with 'CLASS SELECT LIST' and load matched positions in index list (INDLST).

ELSE

   Match symbols of 'CLASS SELECT LIST' with symbols following minus sign. Load index list only when a symbol does NOT match.

ENDIF

NUMIND = number of indices in INDLST.

REFERENCED BY:

COMNOD

OLPARS Program Specifications
SELCLS

SUBPROGRAMS REFERENCED:

EQUALA, FLUSHT, INSCHR, INSPGM, INSRET
TRMGET, TRMPUT

HISTORY:

designed by David J. Tipton October 31, 1980

programmed by David J. Tipton October 31, 1980

PROGRAM NAME: SELECT

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

SELECT gives the user the ability to determine which class symbols are to be displayed at the OLPARS user's terminal, in a one- or two-space display.

PROGRAM USAGE:

User types in 'SELECT'.

USER INTERACTION:

User specifies (via class display symbol) the classes to be displayed.

ALGORITHM / NOTES:

Ask user for classes to be displayed.(OBTCLS)

Display the classes requested.

Put up option list.

NOTE

Classes to be displayed are typed on a single line with no spaces (e.g., 'abc...'). For classes 'not' to be displayed, the user places a minus sign ('-') before the list of class symbols (e.g., '-abc...').

FILES:

DI - display information
DV - display value
CM - communications
PV - projection vector
S1 - scratch 1 (for one-space only)

REFERENCED BY:

OLPARS user

OLPARS Program Specifications
SELECT

SUBPROGRAMS REFERENCED:

CLOSFX, CLSCAT, CMGOTH, DIGENT, DIGHDI, DIPENT,
EQUALA, ERASE, INSINI, INSINT, INSLOG, INSRET,
MENU, MICMAC, OBTCLS, OEXIT, OPENFX, TRMPUT

SEE ALSO:

INTENSIFY

HISTORY:

designed by Steve Haehn October 2, 1978

programmed by Donna Morris August 15, 1979

PROGRAM NAME: SETDS

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

SETDS sets the current data set structure within the communications file with a user-defined value. (i.e, it puts a tree name/node name pair, representing the current data set being worked upon during an OLPARS user's session, along with the data set dimensionality, data set node pointer, and data tree file codes, into the CM file.)

PROGRAM USAGE:

User types in 'SETDS'.

USER INTERACTION:

The user is prompted for a treename and a node name. If either does not exist within OLPARS, the user will be prompted for another value of each. If a colon is typed before the treename, the node name will not be asked for. The senior node will be used.

ALGORITHM / NOTES:

```
Erase the terminal screen.

REPEAT.

    Ask user for new data set name
    (tree name and node name).

    IF (tree does not exist) THEN

        Tell user 'tree not found'.

    ENDIF

UNTIL(tree is found or user quits)

Retrieve Tree header information.
(used to traverse tree structure)
```

```
        REPEAT

            Ask user for node name portion of the
            current data set if the senior node
            was not indicated in the tree prompt.

            Search for the node.

            IF (the node was nonexistent) THEN

               Tell user 'node not in tree'.

            ENDIF

        UNTIL(node name is found or user quits)

        Retrieve file codes from the TL file.

        Place current data set info. into the CM file.

        Show user the option list.

        END
```

FILES:

```
        CM - communications          Instrumentation file
        TI - user tree info. file
        TL - tree list
        HS - history
        OLPARS option file
```

REFERENCED BY:

        OLPARS user.

SUBPROGRAMS REFERENCED:

        ERASE, EQUALA, PROMPT, TRMGET, TLSRCH, OPENFX,
        CMPCDS, TISRCH, MENU, TIGENT

HISTORY:

        designed by   Steven Haehn   May 22, 1978

        programmed by Steven Haehn   Sept. 13, 1979

PROGRAM NAME: SETLOG

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

SETLOG takes an existing logic tree and makes it the current logic tree(restores old logic).

PROGRAM USAGE:

User types in 'SETLOG'.

USER INTERACTION:

The user is asked for the name of the logic tree he wants to be the current logic tree.

ALGORITHM / NOTES:

Ask user for the name of an existing logic tree that he wants to make the current logic tree.

IF (the logic tree does not exist) THEN

Tell the user that the tree does not exist.
Prompt the user again.

ELSE

Get the LI and LV file codes.
Get the number of incomplete logic nodes.
Put the logic name, LI , LV file codes and the number of incomplete logic nodes in the CM file.

ENDIF

Display the options list.

END

FILES:

CM - communications
LI - logic information
LL - logic list

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CLOSFX, CMGOTH, INSINI, INSRET, LGLTRE, MENU, OEXIT
OPENFX, OPENTR, PROMPT, TRMGET, TRMPUT

SEE ALSO:

NAMELOG

HISTORY:

designed by Steven Haehn August 2, 1978

programmed by Mark Maginn July 17, 1980

PROGRAM NAME: SETOPT

CATEGORY: Communication File Access Routine
(system dependent)

PROGRAM DESCRIPTION:

SETOPT obtains the option number of a program from the OLPARS options file. SETOPT opens the options file, using a "system open" subroutine, and searches for the given program name. If the name is found, SETOPT obtains the name's corresponding option number. If the name is not found, SETOPT returns the option number of the program 'ANYTHING'.

When the PUTCM (put into CM file) flag is set to 'true,' SETOPT will place the option number and the corresponding option name in the commmunication file (referenced by FIDCM, the CM file descriptor). Whether or not the PUTCM flag is set to 'true', SETOPT always returns the option number to its calling program.

PROGRAM USAGE:

```
INTEGER SETOPT
      .
      .
      .

OPTNO = SETOPT(FIDCM,PGMNAM,PUTCM)
```

INPUT ARGUMENTS:

```
 FIDCM - INTEGER; file descriptor of the CM file
 PUTCM - LOGICAL; put-option-into-CM-file flag
PGMNAM - LOGICAL *1 ARRAY (10); the program name
```

FILES:

CM - communications
OLPARS option file (OPTION.OLP)

SUBPROGRAMS REFERENCED:

CMPOTH, EQUALS, FGET, OEXIT, OPENBL, PACKW, TRMPUT

OLPARS Program Specifications
SETOPT

SEE ALSO:

MENU

HISTORY:

designed by Steve Haehn June 28, 1978

programmed by Donna Morris August 27, 1979

PROGRAM NAME: SETUP

CATEGORY: Logic Design Routine (system dependent)

PROGRAM DESCRIPTION:

SETUP is called by all L1 and L2 logic design routines to check the current data set and current logic and to obtain necessary information from the user. If there is more than one incomplete logic node, it asks the user to specify which node is to be used. For those logic routines that perform computations on means or covariances, SETUP asks if all the vectors in the data class(es) (classes that lie at the incomplete logic node) are to be used in the computation or just those vectors falling at the logic node.

PROGRAM USAGE:

```
LOGICAL SETUP
   .
   .
   .
IF (SETUP(FDCM,COMAND,ALLVEC,NODNUM,CLIST,NOCLS,ENTAB,
          FDLI,FDLV,FDTI,FDTV,LOGNAM))
```

INPUT ARGUMENTS:

FDCM - INTEGER; the file descriptor of the CM file

COMAND - LOGICAL *1 array(CMDLEN); the name of the calling command

ALLVEC - LOGICAL; TRUE means to ask the question concerning all vectors, FALSE means do not ask the question

OUTPUT ARGUMENTS:

ALLVEC - LOGICAL; TRUE means all vectors, FALSE means only vectors at logic node

NODNUM - INTEGER; the logic node number on which to operate

CLIST - INTEGER array(MAXCLS); the list of classes (slot numbers) that lie at logic node 'NODNUM'

OLPARS Program Specifications
SETUP

NOCLS - INTEGER; number of classes in 'CLIST'

ENTAB - INTEGER array(TABSIZ); the entry table of the current data set

FDLI - INTEGER; logic information file descriptor

FDLV - INTEGER; logic value file descriptor

FDTI - INTEGER; tree information file descriptor

FDTV - INTEGER; tree value file descriptor

LOGNAM - INTEGER array(TRELEN); most recent logic evaluated on current data set

USER INTERACTION:

User is asked for:

- An incomplete logic node number (if there is more than one).
- Whether to use all vectors of a class or just those that fall at the logic node.

ALGORITHM / NOTES:

SETUP = .FALSE.

Get current logic from CM file (CMGLOG).

Get current data set from CM file (CMGCDS).

Open logic and data set files (OPENTR).

IF (current logic or current data set does not exists) THEN

    Print error message to user.

ELSEIF (design set from current logic is not the same as current data set) THEN

    Print error message to user.

```
ELSEIF (last logic run on the data set was not the
current logic (TVGHDR)) THEN

    Print error message - tell user that the last
    logic applied to the data set was not the current
    logic. In order to proceed, the user must evaluate
    the current data set by the current logic.

ELSEIF (there has been a change in the data as denoted
in the TV header) THEN

    Print error message - tell user logic can no
    longer be used on that data set due to a change
    in the data set.

ELSE

    Display a list of incomplete logic nodes (GETLST).

    REPEAT

        Prompt for an incomplete logic node number.

    UNTIL (a correct incomplete logic node number is
    entered or user quits).

    Insert logic node number as current logic node
    in LI file header (LIPCOP).

    Insert option number in LI file entry (SETOPT,
                                           LIPLOG).

    Get a list of classes at the incomplete logic node
    (GCLIST).

    IF (ALLVEC = TRUE) THEN

        Ask user to specify all vectors of the classes
        or just those that fall at the logic node.

        Set ALLVEC accordingly.

    ENDIF

ENDIF

RETURN
```

OLPARS Program Specifications
SETUP

FILES:

CM - communications
LI - logic information
LV - logic value
TI - tree information
TV - tree vector

REFERENCED BY:

All L1, L2 routines

SUBPROGRAMS REFERENCED:

CMGCDS, CMGLOG, CMGOTH, EQUALA, GCLIST,
GETLST, GLONOD, INSPGM, INSRET, LIGDSN, LIPCOP,
LIPLOG, OPENTR, PROMPT, SETOPT, TIGET, TIGNAM,
TRMGET, TRMPUT, TVGHDR

DIAGNOSTICS:

If SETUP = .FALSE., the calling program quits.

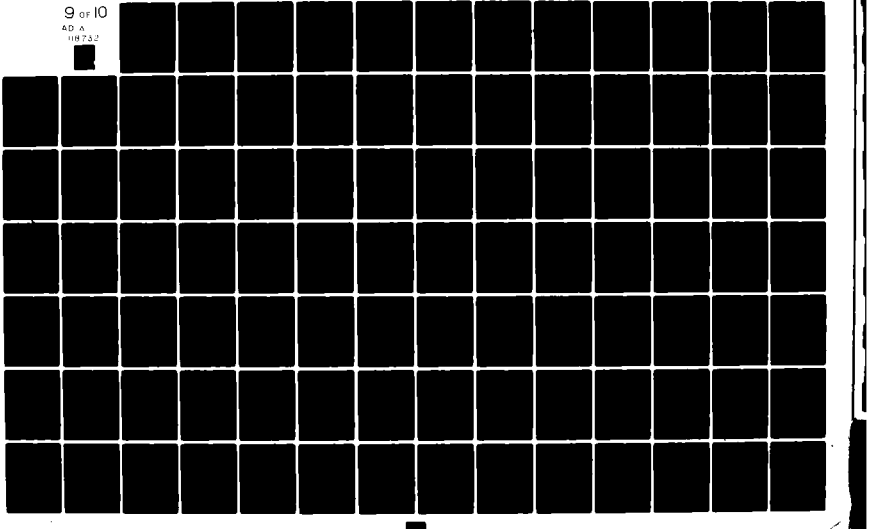
HISTORY:

designed by David BirnBaum September 11, 1978

programmed by Steven Haehn Nov. 23, 1980

AD-A118 732 PAR TECHNOLOGY CORP NEW HARTFORD NY F/G 9/2
ON-LINE PATTERN ANALYSIS AND RECOGNITION SYSTEM. OLPARS VI. SOF--ETC(U)
JUN 82 S E HAEHN, D MORRIS
UNCLASSIFIED PAR-82-20 NL

9 OF 10
AD A
118732



PROGRAM NAME: SLCTMEAS

CATEGORY: Measurement Evaluation Command (subsidiary)

PROGRAM DESCRIPTION:

SLCTMEAS checks to see that the DI file is in rank order format and that the current data set name in the DI and CM files are the same. The user is then asked if he wishes to enter a threshold value or specific measurements.

If a threshold is entered, it is compared to all the ranked measurement values; those measurements whose value is greater (if DSCRMEAS is the command creating the display) or less than (if PROBCONF is the command creating the display) the threshold are "selected" or "asterisked."

If individual measurements are specified, then the asterisk flag for these measurements is set if currently unset, or unset if currently set.

Finally, RODISP is called to display the ranked values according to the *rank* options specified in the DI file.

PROGRAM USAGE:

User types in 'SLCTMEAS'.

USER INTERACTION:

The user is asked if he wants to select a threshold value or specific measurements. Depending upon which he selects, he is prompted for those values.

FILES:

BLOCK I/O	RECORD I/O
HS - history	terminal
CM - communication	instru.dat
DI - display information	
OPTION.OLP - option file	

REFERENCED BY:

OLPARS user.

SUBPROGRAMS REFERENCED:

CDSCHK, CMGCDS, CMGOTH, DIGROE, DIGROH, DIGROI, DIPROI, ERASE, INSFLT, INSINI, INSINT, INSRET, MENU, OEXIT, OPENFX, PROMPT, RODISP, TRMGET, TRMPUT

DIAGNOSTICS:

If the user wishes to enter a threshold, the ranked values in the DI file must be measurement numbers, that is, the type of ranking in the DI file must be either (1) overall, (2) measurement by class, or (3) measurement by class pair. SLCTMEAS will exit if the type of ranking is not 1, 2, or 3. If the user wishes to enter specific measurements, the type of ranking is unimportant, but if the type of ranking is not 1, 2, or 3, the user will receive a message saying that the current display will not indicate which measurements have been selected.

HISTORY:

designed by	Kermit Klingbail	April 10, 1978
programmed by	Donna Morris	December 29, 1981

PROGRAM NAME: SLTEIG

CATEGORY: Display, Projection Vector File Access Routine

PROGRAM DESCRIPTION:

SLTEIG is passed the file descriptor's of the DI and PV file and a flag (ONETWO) indicating one- or two-space. It asks the user to select one or two eigenvalues depending on whether a one-space or two-space display is going to be presented. SLTEIG sets the necessary pointers to the selected vectors in the DI and PV files. If the value of SLTEIG is TRUE, the calling routine quits.

PROGRAM USAGE:

```
LOGICAL SLTEIG
      .
      .
      .
IF(SLTEIG(FIDCM,FIDDI,FIDPV,ONETWO))
```

INPUT ARGUMENTS:

FIDCM - INTEGER; the file descriptor of the CM file

FIDDI - INTEGER; the file descriptor of the DI file

FIDPV - INTEGER; the file descriptor of the PV file

ONETWO - INTEGER; 1 stands for one-space, 2 stands for two-space

ALGORITHM / NOTES:

Ask user to select one or two eigenvalues (PROMPT).

IF (user gives correct response) THEN

Set the number of projection vectors in the PV file header (PVPHDR).

Set pointers in elements 17 and 18 of PV file header (PVPHDR).

Set pointers to these vectors in the DI file header (DIPHDI).

ELSE

Again, ask user to select one or two eigenvalues.

ENDIF

RETURN

FILES:

CM - communications file
DI - display information
PV - projection vector

REFERENCED BY:

S2EIGV, S2EIGV, L1EIGV, L2EIGV

SUBPROGRAMS REFERENCED:

DIPHDI, PROMPT, TRMGET, PVPHDR, PVGHDR

HISTORY:

designed by Dave Birnbaum September 6, 1978

programmed by Donna Morris May 24,1979

PROGRAM NAME: SMGENT

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

SMGENT gets an entry from the Saved Transformation Matrix (SM) file.

PROGRAM USAGE:

CALL SMGENT(FDSM,ENTNO,NDIM,LNKPTR,VECTOR)

INPUT ARGUMENTS:

FDSM - INTEGER; the file descriptor of the SM file

ENTNO - INTEGER; the entry number of the vector being retrieved from the SM file

NDIM - INTEGER; the dimensionality of the matrix to be retrieved from the SM file

OUTPUT ARGUMENTS:

LNKPTR - INTEGER; the link pointer to the next vector of the matrix

VECTOR - REAL array (NDIM); the saved vector to be retrieved from the SM file

FILES:

SM - the saved transformation matrix file
instrumentation file

SUBPROGRAMS REFERENCED:

FGET, INSPGM, INSRET, OEXIT, TRMPUT

HISTORY:

designed by David Birnbaum September 30, 1978

programmed by Jill King March 3, 1981

OLPARS Program Specifications
SMGHDR

PROGRAM NAME: SMGHDR

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

SMGHDHR retrieves the first portion of the header from the Saved Transformation Matrix (SM) file.

PROGRAM USAGE:

CALL SMGHDR(FDSM,NSMTRX,PTRHED,PTREND,NVECS)

INPUT ARGUMENTS:

FDSM - INTEGER; the file descriptor of the SM file

OUTPUT ARGUMENTS:

NSMTRX - INTEGER; the number of saved matrices

PTRHED - INTEGER; the pointer to the head of the free list

PTREND - INTEGER; the pointer to the end of the free list

NVECS - INTEGER; the number of vectors in the SM file

FILES:

SM - saved transformation matrix instrumentation file

SUBPROGRAMS REFERENCED:

FGET, INSPGM, INSRET, OEXIT, TRMPUT

HISTORY:

designed by David Birnbaum September 30, 1978

programmed by Jill King March 3, 1981

PROGRAM NAME: SMGLNK

CATEGORY: LEVEL II FILE ACCESS SUBROUTINE

PROGRAM DESCRIPTION:

SMGLNK reads the link pointer from a Saved Transformation Matrix (SM) file entry.

PROGRAM USAGE:

```
INTEGER SMGLNK
    .
    .
    .
N = SMGLNK (FDSM, ENTNO, LNKPTR)
```

INPUT ARGUMENTS:

FDSM - INTEGER; the file descriptor of the SM file

ENTNO - INTEGER; the entry number of the vector whose link pointer is to be read

OUTPUT ARGUMENTS:

LNKPTR - INTEGER; the link pointer value to be read

SMGLNK - INTEGER; indicates the successfulness of the program's completion

FILES:

SM - saved transformation matrix file
instrumentation file

SUBPROGRAMS REFERENCED:

FGET, INSPGM, INSRET, OEXIT, TRMPUT

HISTORY:

designed by Jill King March 16,1981

programmed by Jill King March 16,1981

PROGRAM NAME: SMGMDS

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

SMGMDS retrieves the matrix description information from the Saved Transformation Matrix (SM) file.

PROGRAM USAGE:

```
CALL SMGMDS(FDSM,ENTNO,MATNAM,TYPE,NDIM,MATPTR)
```

INPUT ARGUMENTS:

FDSM - INTEGER; the file descriptor of the SM file

ENTNO - INTEGER; the entry number of the matrix description information

OUTPUT ARGUMENTS:

MATNAM - INTEGER array (TRELEN); the name of the saved transformation matrix

TYPE - INTEGER; the type of transformation which was performed to get the resulting matrix. A -1 indicates a normal transformation; a positive value indicates an eigenvector transformation where the value represents the number of eigenvalues used.

NDIM - INTEGER; the dimensionality of the saved matrix

MATPTR - INTEGER; pointer to first vector of saved matrix

FILES:

SM - saved transformation matrix instrumentation file

SUBPROGRAMS REFERENCED:

FGET, INSPGM, INSRET, OEXIT, PACKW, TRMPUT

HISTORY:

designed by David Birnbaum September 30, 1978

programmed by Jill King March 3, 1981

PROGRAM NAME: SMPENT

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

SMPENT writes an entry into the saved transformation matrix (SM) file.

PROGRAM USAGE:

CALL SMPENT(FDSM,ENTNO,NDIM,LNKPTR,VECTOR)

INPUT ARGUMENTS:

FDSM - INTEGER; the file descriptor of the SM file

ENTNO - INTEGER; the entry number of the vector to be saved in the SM file

NDIM - INTEGER; the dimensionality of the matrix to be saved in the SM file

LNKPTR - INTEGER; the link pointer to the next vector of the matrix

VECTOR - REAL array (NDIM); the vector to be saved in the SM file

FILES:

SM - saved transformation matrix file
instrumentation file

SUBPROGRAMS REFERENCED:

FPUT, INSPGM, INSRET

HISTORY:

designed by David Birnbaum September 30, 1978

programmed by Jill King March 3, 1981

OLPARS Program Specifications
SMPHDR

PROGRAM NAME: SMPHDR

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

SMPHDR writes the first portion of the header into the Saved Transformation Matrix (SM) file.

PROGRAM USAGE:

```
CALL SMPHDR(FDSM,NSMTRX,PTRHED,PTREND,NVECS)
```

INPUT ARGUMENTS:

FDSM - INTEGER; the file descriptor of the SM file

NSMTRX - INTEGER; the number of saved matrices in the SM file

PTRHED - INTEGER; the pointer to the head of the free list

PTREND - INTEGER; the pointer to the end of the free list

NVECS - INTEGER; the number of vectors in the SM file

FILES:

SM - saved transformation matrix file
instrumentation file

SUBPROGRAMS REFERENCED:

FPUT, INSPGM, INSRET

HISTORY:

designed by David Birnbaum September 30, 1978

programmed by Jill King March 3, 1981

PROGRAM NAME: SMPLNK

CATEGORY: LEVEL II FILE ACCESS SUBROUTINE

PROGRAM DESCRIPTION:

SMPLNK writes the link pointer to the Saved Transformation Matrix (SM) file entry.

PROGRAM USAGE:

CALL SMPLNK (FDSM, ENTNO, LNKPTR)

INPUT ARGUMENTS:

FDSM - INTEGER; the file descriptor of the SM file

ENTNO - INTEGER; the entry number of the vector whose link pointer is being written

LNKPTR - INTEGER; the link pointer value to be written

FILES:

SM - saved transformation matrix file
instrumentation file

SUBPROGRAMS REFERENCED:

FPUT, INSPGM, INSRET

HISTORY:

designed by Jill King March 13,1981

programmed by Jill King March 13,1981

OLPARS Program Specifications
SMPMDS

PROGRAM NAME: SMPMDS

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

SMPMDS writes the matrix description information into the saved transformation matrix (SM) file header.

PROGRAM USAGE:

CALL SMPMDS(FDSM,ENTNO,MATNAM,TYPE,NDIM,MATPTR)

INPUT ARGUMENTS:

FDSM - INTEGER; the file descriptor of the SM file

ENTNO - INTEGER; the entry number of the matrix description information

MATNAM - INTEGER array (TRELEN); the name of the saved transformation matrix

TYPE - INTEGER; the type of transformation which was performed to get the resulting matrix. A -1 indicates a normal transformation; a positive value indicates an eigenvector transformation where the value represents the number of eigenvalues used.

NDIM - INTEGER; the dimensionality of the saved matrix

MATPTR - INTEGER; the pointer to the first vector of the saved matrix

FILES:

SM - saved transformation matrix file
instrumentation file

SUBPROGRAMS REFERENCED:

FPUT, INSPGM, INSRET, ITREAL

HISTORY:

designed by David Birnbaum September 30, 1978

programmed by Jill King March 3, 1981

PROGRAM NAME: SORT

CATEGORY: Support Routine

PROGRAM DESCRIPTION:

SORT returns the descending order pointers (array positions) of the elements of a real array.

PROGRAM USAGE:

CALL SORT(NSORTD,SIZE,ORDER)

INPUT ARGUMENTS:

NSORTD - REAL array; the array for which pointers are returned

SIZE - INTEGER; the size of the array

OUTPUT ARGUMENTS:

ORDER - INTEGER array; the order array - contains the array positions of the elements of the array in descending order

REFERENCED BY:

EIGPRT

HISTORY:

designed by Donna Morris June 18,1979

programmed by Donna Morris June 18,1979

OLPARS Program Specifications
SU1SP

PROGRAM NAME: SU1SP

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

SU1SP (set up one-space logic block) retrieves one space projection logic from the LV file and stores it in a common area for use by logic routines.

PROGRAM USAGE:

CALL SU1SP(FDLV,LVPTR,NDIM,LVETSZ)

INPUT ARGUMENTS:

FDLV - INTEGER; the file descriptor of the LV file

LVPTR - INTEGER; entry number in LV file of first block of the one-space logic

NDIM - INTEGER; data set dimensionality

LVETSZ - INTEGER; logic value file's entry size

OUTPUT ARGUMENTS:

Common block, named GP1BLK, containing number of boundaries, logic nodes associated with regions, a discriminant vector and threshold(s).

FILES:

LV - logic value

REFERENCED BY:

EVALU8, GP1PRT

SUBPROGRAMS REFERENCED:

GTRGNR, INSFLT, INSINT, INSPGM, INSRET

HISTORY:

designed by	Kermit Klingbail	October 3, 1978
programmed by	Donna Morris	June 1, 1981

PROGRAM NAME: SU2SP

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

SU2SP (set up two-space logic block) retrieves two-space projection logic from the LV file and stores it in a COMMON area for use by logic routines.

PROGRAM USAGE:

CALL SU2SP(FDLV,LVPTR,NDIM,LVETSZ)

INPUT ARGUMENTS:

FDLV - INTEGER; the file descriptor of the LV file

LVPTR - INTEGER; entry number in LV file of first block of the two-space logic

NDIM - INTEGER; data set dimensionality

LVETSZ - INTEGER; logic value file's entry size

OUTPUT ARGUMENTS:

Common block, named GP2BLK, containing number of boundaries, logic nodes associated with regions, discriminant vector(s), and threshold(s).

FILES:

LV - logic value

REFERENCED BY:

EVALU8, GP2PRT, PWSPRT

SUBPROGRAMS REFERENCED:

GTRGNI, GTRGNR, INSFLT, INSINT, INSPGM, INSRET

HISTORY:

designed by	Kermit Klingbail	October 3, 1978
programmed by	Donna Morris	June 1, 1981

PROGRAM NAME: SUBMNC

CATEGORY: Utilities Subroutine

PROGRAM DESCRIPTION:

SUBMNC 'subtracts' a covariance or scatter array and its corresponding mean vector from the parent node mean and covariance or scatter arrays. Also, the total number of vectors at the parent node is adjusted. This routine is used when a data tree node is deleted. The formulas are given below:

LET

- MeanA = the mean vector of the node to be deleted
- MeanAB = the mean vector of the parent node of the node to be deleted
- MeanB = the mean vector of the remaining nodes at the parent node
- A = the covariance or scatter array of the node to be deleted
- AB = the covariance or scatter array of the parent node
- B = the covariance or scatter array of the remaining nodes at the parent node
- NvecA = the number of vectors used to compute the arrays at the node to be deleted
- NvecAB = the number of vectors used to compute the arrays at the parent node
- NvecB = the number of vectors used to compute the arrays of the remaining nodes at the parent node

COMBINED MEAN VECTOR:

```
MeanB = ((NvecAB*MeanAB) - (NvecA*MeanA)) / NvecB
```

COMBINED COVARIANCE OR SCATTER ARRAY:

```
B = [NvecAB * (AB(i,j) + MeanAB(i) * MeanAB(j)) +
     NvecA * (A(i,j) + MeanA(i) * MeanA(j))] / NvecB
     - [MeanB(i) * MeanB(j)]
```

PROGRAM USAGE:

CALL SUBMNC(FDTI,DESNOD,NDIM,NVECA,MEANA,COVA)

INPUT ARGUMENTS:

FDTI - INTEGER; the file descriptor of the TI file

DESNOD - INTEGER; the entry number of the node whose mean and covariance arrays are being adjusted

NDIM - INTEGER; the dimension of the vectors

NVECA - INTEGER; the number of vectors being deleted

MEANA - REAL array (NDIM); the mean vector of the vectors deleted

COVA - REAL array (NDIM*(NDIM+1)/2); the covariance matrix of vectors deleted

FILES:

TI - tree information file
instrumentation file

REFERENCED BY:

SBUPMC

SUBPROGRAMS REFERENCED:

INSFLT, INSINT, INSPGM, INSRET, TIGCOP, TIGCOV
TIGMN, TIPCOP, TIPCOV, TIPMN

HISTORY:

designed by Jill King May 13, 1981

programmed by Jill King May 13, 1981

OLPARS Program Specifications
SUFISH

PROGRAM NAME: SUFISH

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

SUFISH (set up pairwise logic block) retrieves pairwise logic from the LV file and stores it in a COMMON area for use by logic routines.

PROGRAM USAGE:

CALL SUFISH(FDLV,LVETSZ,LVPTR1)

INPUT ARGUMENTS:

FDLV - INTEGER; the file descriptor of the LV file

LVETSZ - INTEGER; logic value file's entry size

LVPTR1 - INTEGER; entry number in LV file of first block of the pairwise logic

OUTPUT ARGUMENTS:

Common block, named FSHBLK, containing the LV file first entry for the pairwise logic (includes the number of classes, minimum vote count, and link element to the first entry containing the class pair links), and the the logic node numbers associated with data classes at the node.

FILES:

LV - logic value

REFERENCED BY:

EVALU8,

SUBPROGRAMS REFERENCED:

GTRGNI, INSPGM, INSRET

HISTORY:

designed by Donna Morris October 3, 1978

programmed by Donna Morris June 1, 1981

OLPARS Program Specifications
SUMMCM

PROGRAM NAME: SUMMCM

CATEGORY: Utility Command

PROGRAM DESCRIPTION:

SUMMCM causes a summary of a confusion matrix, stored in the DI file, to be displayed at the terminal.

PROGRAM USAGE:

User types in 'SUMMCM'.

ALGORITHM / NOTES:

```
IF (display code = 4, i.e., confusion matrix) THEN
    Print summary.
ELSE
    Print error message.
ENDIF
Put up option list at user's terminal (MENU).
END
```

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

DIGCMH, DIGDC, ERASE, INSINI, INSRET, MENU, OEXIT, OPENFX, TRMPUT

HISTORY:

designed by David BirnBaum September 25, 1978

programmed by Steven Haehn May 29, 1981

PROGRAM NAME: SUNMV

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

SUNMV (set up nearest mean vector logic block) retrieves nearest mean vector logic from the LV file and stores it in a COMMON area for use by logic routines.

PROGRAM USAGE:

CALL SUNMV(FDLV,NDIM,LVETSZ,LVPTR1)

INPUT ARGUMENTS:

FDLV - INTEGER; the file descriptor of the LV file

NDIM - INTEGER; data set dimensionality

LVETSZ - INTEGER; logic value file's entry size

LVPTR1 - INTEGER; entry number in LV file of first block of the nearest mean vector logic

OUTPUT ARGUMENTS:

Common block, named NMVBLK, containing the LV file first entry for the NMV logic, the logic nodes associated with data classes, the ignore measurements mask array, the reject boundary distance values, and the determinants of the covariance matrix.

FILES:

LV - logic value

REFERENCED BY:

EVALU8, NMVPRT

SUBPROGRAMS REFERENCED:

GTRGNI, GTRGNR, INSPGM, INSRET

OLPARS Program Specifications
SUNMV

HISTORY:

designed by Donna Morris October 3, 1978

programmed by Donna Morris June 1, 1981

PROGRAM NAME: TEXT

CATEGORY: Terminal I/O (graphic, system dependent)

PROGRAM DESCRIPTION:

This routine places a string of text on the graphics display, starting at screen coordinates (X,Y).

The screen coordinates are relative screen coordinates, used to redefine the origin.

STRING must be dimensioned in the calling program to at least the number of characters in STRING. In other words, each character in the string takes up one integer in the array. (HGT,WID,ANGLE,INDIC have no use and are only included to be compatible with the PLTMAP FORTRAN subroutine).

The last character in the text string must be a null character (i.e. a zero).

PROGRAM USAGE:

CALL TEXT(X,Y,STRING,HGT,WID,ANGLE,INDIC)

INPUT ARGUMENTS:

```
     X - INTEGER; the starting x screen coordinate
     Y - INTEGER; the starting y screen coordinate
STRING - INTEGER; the character string
   HGT - INTEGER; ignored
   WID - INTEGER; ignored
 ANGLE - INTEGER; ignored
 INDIC - INTEGER; ignored
```

REFERENCED BY:

MENU

HISTORY:

designed by Steve Haehn April 14, 1978

programmed by Steve Haehn April 1, 1981

OLPARS Program Specifications
THRDSP

PROGRAM NAME: THRDSP

CATEGORY: Logic Design Routine

PROGRAM DESCRIPTION:

THRDSP does the modifications for THRESHMOD on a pair-wise logic node. The vectors are projected onto the Fisher direction and displayed on the screen. The user is prompted for a threshold number to modify, and is asked to enter a new threshold (GIN). The new threshold is stored in the LV file (PTRGNR) when the user is satisfied with the changes.

PROGRAM USAGE:

```
      CALL THRDSP(FIDCM,FIDLV,FIDTI,FIDTV,LVNEL,TRUCLS,NDIM,
     +            PMODE,POINTS,NXTCLS,ET,CLIST,OPTNO)
```

INPUT ARGUMENTS:

FIDCM - INTEGER; file descriptor of the CM file

FIDLV - INTEGER; file descriptor of the LV file

FIDTI - INTEGER; file descriptor of the TI file

FIDTV - INTEGER; file descriptor of the TV file

LVNEL - INTEGER; number of elements in an LV file entry

TRUCLS - INTEGER; number of true classes at logic node

NDIM - INTEGER; dimensionality of dataset

PMODE - INTEGER; the prompt mode flag

POINTS - INTEGER array ((TRUCLS*(TRUCLS-1))/2); the pointers to the class pairs

NXTCLS - INTEGER; the index of this class pair

ET - INTEGER array (TABSIZ); the entry table

CLIST - INTEGER array (TRUCLS); slot numbers of classes at logic node

OPTNO - INTEGER; option number of THRESHMOD

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Put header and base line up on screen.

Retrieve the vectors from the TV file (TVGVEC).

Project the vectors for this class pair onto the line.

Project the means for this class pair onto the base line.

Prompt the user for a threshold number.

Prompt the user for a new threshold position (GIN).

Store the thresholds in the LV file (PTRGNR).

Return to main program.

REFERENCED BY:

THRESHMOD

SUBPROGRAMS REFERENCED:

CMGSCN, DLREGN, ERASE, GIN, GTRGNR, INSPGM, INSRET
LINSEG, MOVE, PROJCT, PROMPT, PTRGNR, TIGCOP, TIGHDR
TIGMN, TRMGET, TRMPUT, TVGVEC, WRTNOW

SEE ALSO:

OPTDSP, THRESHMOD

HISTORY:

designed by John W. Tenney Sept. 2, 1981

programmed by John W. Tenney Oct. 9, 1981

OLPARS Program Specifications
THRESHMOD

PROGRAM NAME: THRESHMOD

CATEGORY: Logic Design Command

PROGRAM DESCRIPTION:

THRESHMOD is designed to modify an existing FISHER or OPTIML logic node. The user is given a list of logic nodes with FISHER or OPTIML logic. After the node is selected, the user is prompted for a class pair. The projections of the vectors onto the Fisher direction are displayed on the screen and the user is prompted for a threshold number. Then the user is prompted for the threshold's new position (GIN). The logic is stored in the LV file. The user is then prompted for another class pair.

To evaluate the modified logic, use PWEVAL

PROGRAM USAGE:

User types in 'THRESHMOD'.

ALGORITHM / NOTES:

Erase screen and home cursor (ERASE).

Display a list of PAIRWISE logic nodes (GETLST,TRMPUT).

Ask user to enter a PAIRWISE logic node number (PROMPT).

REPEAT

Prompt user for a class pair (GETPR).

Project and display the vectors on the screen, get the threshold number and location from the user, and store the logic in the LV file (THRDSP).

UNTIL (user is satisfied)

Fix temporary logic elements in the TV file (KILLND).

Put up option list at user's terminal (MENU)

END

REFERENCED BY:

OLPARS user

SUBPROGRAMS REFERENCED:

CHKEXS, CLOSTR, CMGCDS, CMGLOG, CMGOTH, ERASE, GETLST, GETPR, GTRGNI, GTSLOT, INSINI, INSRET, KILLND, LIGCOP, LIGSTR, MENU, OEXIT, OPENFX, OPENTR, PROMPT, SETOPT, THRDSP, TIGET, TIGNAM, TRMGET, TRMPUT

SEE ALSO:

FISHER, FISHMOD, OPTIMLMOD, PWEVAL

HISTORY:

designed by John W. Tenney Sept. 2, 1981

programmed by John W. Tenney Oct. 9, 1981

OLPARS Program Specifications
TIGCAP

PROGRAM NAME: TIGCAP

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TIGCAP gets the counts and pointers from a node entry in TI. It is passed the file descriptor (FID) of the TI file and an entry number in that file (ENTNO). It returns (in ICAP) the set of counts and pointers (elements 5 through 16 of a TI file (entry)) from that entry number.

PROGRAM USAGE:

```
CALL TIGCAP(FID,ENTNO,ICAP)
```

INPUT ARGUMENTS:

```
  FID - INTEGER; the file descriptor of the TI file
ENTNO - INTEGER; the entry number of the node
```

OUTPUT ARGUMENTS:

```
 ICAP - INTEGER array(12); the set of counts and
                           pointers
```

REFERENCED BY:

TISRCH, GNXTND

SUBPROGRAMS REFERENCED:

FGET

HISTORY:

designed by David A. Birnbaum August 29, 1978

programmed by Dave Tipton June 20, 1979

PROGRAM NAME: TIGCOP

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TIGCOP gets a count or pointer from a TI file entry. It is passed the file descriptor (FID), entry number (ENTNO) and a code (ICODE) specifying what to get from the TI file. TIGCOP is set equal to the value requested. The code is given by the following table:

CODE	INFORMATION REQUESTED	ELEMENT NO.
1	No. of vectors at a node	5
2	No. of lowest nodes beneath a node	6
3	No. of children	7
4	Node level (NL)	8
5	Parent pointer (PP)	9
6	Sibling pointer (SP)	10
7	First child point (FCP)	11
8	First lowest node pointer (FLNP)	12
9	Lowest node link pointer (LNLP)	13
10	Lowest node back pointer (LNBP)	14
11	Vector pointer to TV file (VP)	15
12	Back pointer to entry table (BPET)	16
13	Free list pointer (FRELST)	1

PROGRAM USAGE:

```
INTEGER TIGCOP
PTR = TIGCOP(FID,ENTNO,ICODE)
```

OLPARS Program Specifications
TIGCOP

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the TI file
ENTNO - INTEGER; the entry number of a node
ICODE - INTEGER; a code

OUTPUT ARGUMENTS:

TIGCOP - INTEGER; the count or pointer from a TI file entry

FILES:

TI - tree information

REFERENCED BY:

TISRCH

SUBPROGRAMS REFERENCED:

FGET

HISTORY:

designed by Dave Birnbaum May 18, 1978

programmed by David Tipton June 7,1979

PROGRAM NAME: TIGCOV

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TIGCOV gets the entire lower triangular portion of the covariance matrix whose size is (NDIM*(NDIM+1))/2. The matrix is placed in VEC.

PROGRAM USAGE:

```
CALL TIGCOV(FID,ENTNO,NDIM,VEC)
```

INPUT ARGUMENTS:

```
  FID - INTEGER; the file descriptor of the TI file
ENTNO - INTEGER; the entry number of the node
 NDIM - INTEGER; the vector dimensionality
```

OUTPUT ARGUMENTS:

```
  VEC - REAL array ((MAXDIM*(MAXDIM+1))/2);
                   the covariances
```

FILES:

TI - tree information
Instrumentation files

REFERENCED BY:

APPEND, COPYND, DCRIM, EIGNPV, NODCOM

SUBPROGRAMS REFERENCED:

FGET,TRMPUT

SEE ALSO:

TIGMN

HISTORY:

designed by Dave Birnbaum May 31, 1978

programmed by Donna Morris May 17,1979

OLPARS Program Specifications
TIGET

PROGRAM NAME: TIGET

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TIGET gets one or all entries of the entry table from a TI file. If FIRST=0, then all entries will be returned to ENTTBL; otherwise, only that entry specified by FIRST will be returned to ENTTBL. The file descriptor of the TI file is passed to TIGET in FID.

PROGRAM USAGE:

CALL TIGET(FID,FIRST,ENTTBL)

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the TI file

FIRST - INTEGER; first element of entry table to be extracted

OUTPUT ARGUMENTS:

ENTTBL - INTEGER array (TABSIZ); the entry table

FILES:

TI - tree information

REFERENCED BY:

SETDS

SUBPROGRAMS REFERENCED:

FGET

HISTORY:

designed by Dave Birnbaum May 24, 1978

programmed by David Tipton June 7, 1979

PROGRAM NAME: TIGHDR

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TIGHDR gets the first five elements of the header of the TI file. It places the following quantities in the NUM(5) array:

- Number of nodes in tree.
- Dimension.
- Next available node pointer.
- Next open entry at end of file.
- Senior node pointer.

PROGRAM USAGE:

CALL TIGHDR(FID,NUM)

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the TI file

OUTPUT ARGUMENTS:

NUM - INTEGER array (5); the output information

FILES:

TI - tree information

REFERENCED BY:

COPYND

SUBPROGRAMS REFERENCED:

FGET

HISTORY:

designed by David A. Birnbaum August 30, 1978

programmed by Dave Tipton June 20, 1979

PROGRAM NAME: TIGMN

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TIGMN gets that portion of the mean vector from coordinate IBEG to coordinate IEND. TIGMN places the means in VEC. The number 50 is chosen as a compromise between many disk accesses and a large "in-core" storage requirement.

PROGRAM USAGE:

```
CALL TIGMN(FID,ENTNO,IBEG,IEND,VEC)
```

INPUT ARGUMENTS:

```
  FID - INTEGER; the file descriptor of the TI file
ENTNO - INTEGER; the entry number of the node
 IBEG - INTEGER; the beginning coordinate
 IEND - INTEGER; the end coordinate
```

OUTPUT ARGUMENTS:

```
  VEC - Real array (MAXDIM); the coordinates of the mean
                             vector
```

FILES:

TI - tree information
Instrumentation files

REFERENCED BY:

COPYND, NODCOM

SUBPROGRAMS REFERENCED:

FGET

SEE ALSO:

TIPMN

HISTORY:

designed by	Dave Birnbaum	May 25, 1978
programmed by	David Tipton	June 7, 1979

PROGRAM NAME: TIGNAM

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TIGNAM gets the name of a node in a data set. It is passed the file descriptor (FID) of a TI file and the number of an entry (ENTNO) in that file. It retrieves the nodename of that entry and places it in NAME.

PROGRAM USAGE:

```
CALL TIGNAM(FID,ENTNO,NAME)
```

INPUT ARGUMENTS:

```
  FID - INTEGER; the file descriptor of the TI file
ENTNO - INTEGER; the entry number of the node
```

OUTPUT ARGUMENTS:

```
 NAME - INTEGER array (NODLEN); nodename
```

FILES:

TI - tree information
Instrumentation files

REFERENCED BY:

TISRCH

SUBPROGRAMS REFERENCED:

FGET

HISTORY:

designed by Dave Birnbaum May 19, 1978

programmed by David Tipton July 10, 1979

PROGRAM NAME: TIGVAR

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TIGVAR uses NDIM calls to FGET to build up variance array from covariance matrix.

PROGRAM USAGE:

CALL TIGVAR(FIDTI,ENTNO,NDIM,VARRAY)

INPUT ARGUMENTS:

FIDTI - INTEGER; the file descriptor of the TI file
ENTNO - INTEGER; the entry number of the node
NDIM - INTEGER; the dimension of the data set

OUTPUT ARGUMENTS:

VARRAY - REAL array(NDIM); the variances

FILES:

TI - tree information

REFERENCED BY:

DSCRMEAS

SUBPROGRAMS REFERENCED:

FGET, INSPGM, INSRET, OEXIT, TRMPUT

HISTORY:

designed by Kermit Klingbail April 17, 1978

programmed by Donna Morris December 1, 1980

OLPARS Program Specifications
TIPCAP

PROGRAM NAME: TIPCAP

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TIPCAP inserts elements 5 through 16 into a TI file entry.

PROGRAM USAGE:

```
CALL TIPCAP(FID,ENTNO,ICAP)
```

INPUT ARGUMENTS:

```
  FID - INTEGER; the file descriptor of the TI file
ENTNO - INTEGER; the entry number of the node
 ICAP - INTEGER array (12); the counts and pointers
```

FILES:

TI - tree information

SUBPROGRAMS REFERENCED:

FPUT

HISTORY:

designed by David A. Birnbaum August 30, 1978

programmed by Dave Tipton June 21, 1979

PROGRAM NAME: TIPCOP

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TIPCOP puts a count or pointer into a TI file entry. It is passed the file descriptor, the entry number, a code specifying what to put into the TI file, and the count or pointer. The codes are the same as in TIGCOP.

PROGRAM USAGE:

```
CALL TIPCOP(FID,ENTNO,ICODE,COUNT)
```

INPUT ARGUMENTS:

```
  FID - INTEGER; the file descriptor of the TI file
ENTNO - INTEGER; the entry number of the node
ICODE - INTEGER; a code
COUNT - INTEGER; the count or pointer
```

FILES:

TI - tree information

SUBPROGRAMS REFERENCED:

FPUT

SEE ALSO:

TIGCOP

HISTORY:

designed by	David A. Birnbaum	May 25, 1978
programmed by	Dave Tipton	June 21, 1979

OLPARS Program Specifications
TIPCOV

PROGRAM NAME: TIPCOV

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TIPCOV puts the entire lower triangular portion of the covarialnce matrix, whose size is (NDIM*(NDIM+1))/2, into the tree information file.

PROGRAM USAGE:

CALL TIPCOV(FID,ENTNO,NDIM,VEC)

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the TI file

ENTNO - INTEGER; the entry number of the node

NDIM - INTEGER; the vector dimensionality.

VEC - REAL array (MAXDIM*(MAXDIM+1)/2); the covariances

FILES:

TI - tree information

REFERENCED BY:

COPYND

SUBPROGRAMS REFERENCED:

FPUT

SEE ALSO:

TIGCOV

HISTORY:

designed by Dave Birnbaum May 31, 1978

programmed by David Tipton July 10,1979

PROGRAM NAME: TIPET

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TIPET writes one or all entries of the entry table to a TI file. If FIRST is zero, then all 100 entries will be written; otherwise, only that entry specified by FIRST will be written. The file descriptor of the TI file is passed to TIPET in FID.

PROGRAM USAGE:

```
CALL TIPET(FID,FIRST,ENTTBL)
```

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the TI file

FIRST - INTEGER; element number to be written to the entry table of the TI header; if 0, then write all elements of the entry table to the TI header

ENTTBL - INTEGER array (TABSIZ); the entry table

FILES:

TI - tree information

SUBPROGRAMS REFERENCED:

FPUT

HISTORY:

designed by David A. Birnbaum May 24, 1978

programmed by Dave Tipton June 21, 1979

OLPARS Program Specifications
TIPHDR

PROGRAM NAME: TIPHDR

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TIPHDR writes the first five elements of the header of the TI file. The following quantities are put in the TI header.

- Number of nodes in tree.
- Dimension.
- Next available node pointer.
- Next open entry at end of file.
- Senior node pointer.

PROGRAM USAGE:

```
CALL TIPHDR(FID,NODCNT,NDIM,NXTAVL,ENDPTR,SENIOR)
```

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the TI file

NODCNT - INTEGER; the number of nodes in the data tree

NDIM - INTEGER; the data set dimensionality

NXTAVL - INTEGER; the next available node position

ENDPTR - INTEGER; the next open entry at the end of the file

SENIOR - INTEGER; the senior node pointer

FILES:

TI - tree information

SUBPROGRAMS REFERENCED:

FPUT

HISTORY:

designed by David A. Birnbaum August 30, 1978

programmed by David J. Tipton June 22, 1979

OLPARS Program Specifications
TIPMN

PROGRAM NAME: TIPMN

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TIPMN places into entry number ENTNO the mean vector from measurements IBEG to IEND.

PROGRAM USAGE:

```
CALL TIPMN(FID,ENTNO,IBEG,IEND,MEAS)
```

INPUT ARGUMENTS:

```
  FID - INTEGER; the file descriptor of the TI file
ENTNO - INTEGER; the entry number of the node
 IBEG - INTEGER; the beginning measurement
 IEND - INTEGER; the ending measurement
 MEAS - REAL array (MAXDIM); the mean vector
```

FILES:

TI - tree information

REFERENCED BY:

COPYND

SUBPROGRAMS REFERENCED:

FPUT

SEE ALSO:

TIGMN

HISTORY:

designed by David A. BIrnbaum May 26, 1978

programmed by David J. Tipton June 22, 1979

PROGRAM NAME: TIPNAM

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TIPNAM inserts a four character node name into a TI file entry.

PROGRAM USAGE:

```
CALL TIPNAM(FID,ENTNO,NAME)
```

INPUT ARGUMENTS:

```
  FID - INTEGER; the file descriptor of the TI file
ENTNO - INTEGER; the entry number of the node
 NAME - INTEGER array (NODLEN); the name of the node
```

FILES:

TI - tree information

SUBPROGRAMS REFERENCED:

FPUT

HISTORY:

designed by	David A. Birnbaum	August 30, 1978
programmed by	David J. Tipton	June 22, 1979

PROGRAM NAME: TISRCH

CATEGORY: Data Tree File Access Routine

PROGRAM DESCRIPTION:

TISRCH searches a TI file for a nodename. It is passed the file descriptor (TIFD) of the TI file, the node name (NAME), the entry table of the TI file (ENTAB), and the entry table slot number (ISTART) of the node where the search begins. TISRCH searches for a nodename match only for nodes underneath the beginning one. TISRCH returns 0 if no match is found, or it returns the entry table slot number of the matching node. The starting node does not have to be the senior node of a data tree.

PROGRAM USAGE:

```
INTEGER TISRCH
      .
      .
      .
ISLOT = TISRCH(TIFD,NAME,ENTAB,ISTART)
```

INPUT ARGUMENTS:

TIFD - INTEGER; the file descriptor of the TI file

NAME - CHARACTER STRING(4); the name of a node to find

ENTAB - INTEGER array; the entry table

ISTART - INTEGER; the starting point of the search

OUTPUT ARGUMENTS:

TISRCH - INTEGER; the entry table slot number of the matching node

ALGORITHM / NOTES:

```
TISRCH = 0
MATCH = .FALSE.

Make starting node the present node.

WHILE (MATCH = .FALSE. and there is a present node to be
compared)

    IF (present nodename = the correct nodename) THEN

        MATCH = .TRUE.

        TISRCH = entry slot number of present node.

    ELSEIF (there is no next node) THEN

        Indicate that there is no present node to be
        compared.

    ENDIF

ENDWHILE
```

FILES:

```
TI - tree information
Instrumentation files
```

REFERENCED BY:

SETDS

SUBPROGRAMS REFERENCED:

TIGNAM, TIGCAP, EQUALA, GNXTND (get next node)

HISTORY:

designed by David Birnbaum May 19, 1978

programmed by Steven Haehn Sept. 10, 1979

PROGRAM NAME: TLGENT

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TLGENT is passed the file descriptor (FIDTL) of the TL file and the entry number (ENTNO) in that file to be read. TLGENT returns the file code of the TI file (FCDTI), the file code of the TV file (FCDTV), the data set dimensionality (NDIM), the link pointer (LNKPTR), and the name of the data set (NAME).

PROGRAM USAGE:

```
CALL TLGENT(FIDTL,ENTNO,FCDTI,FCDTV,NDIM,LNKPTR,NAME)
```

INPUT ARGUMENTS:

```
FIDTL - INTEGER; the file descriptor of the TL file
ENTNO - INTEGER; entry number in the TL file
```

OUTPUT ARGUMENTS:

```
FCDTI - INTEGER; the file code of the TI file
FCDTV - INTEGER; the file code of the TV file
 NDIM - INTEGER; the dimension of the data set
LNKPTR - INTEGER; the link pointer
 NAME - INTEGER array (TRELEN); the name of the
                                data set
```

FILES:

TL - tree list
Instrumetation files

REFERENCED BY:

TLSRCH, DELETR

SUBPROGRAMS REFERENCED:

FGET,PACKW,TRMPUT

HISTORY:

designed by Dave Birnbaum August 17, 1978

programmed by Donna Morris July 26,1979

PROGRAM NAME: TLGHDR

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TLGHDR gets the number of entries element (NOE) and the alphabetic link pointer element (LNKPTR) from the TL file header.

PROGRAM USAGE:

```
CALL TLGHDR(FIDTL,NOE,LNKPTR)
```

INPUT ARGUMENTS:

FIDTL - INTEGER; the file descriptor of the TL file

OUTPUT ARGUMENTS:

NOE - INTEGER; the number of entries

LNKPTR - INTEGER; the link pointer which points to the first entry in the alphabetically linked list

FILES:

TL - tree list
Instrumentation files

REFERENCED BY:

TLSRCH, DDATATREE, DELETR

SUBPROGRAMS REFERENCED:

FGET, TRMPUT

HISTORY:

designed by Dave Birnbaum May 19, 1978

programmed by Donna Morris July 26,1979

PROGRAM NAME: TLPENT

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TLPENT writes a logical entry to the TL file. It is passed the file descriptor of the TL file (FIDTL), the logical entry number (ENTNO), and the information on the tree to be placed in the TL file.

PROGRAM USAGE:

```
CALL TLPENT(FIDTL,ENTNO,FCDTI,FCDTV,NDIM,LNKPTR,NAME)
```

INPUT ARGUMENTS:

```
 FIDTL - INTEGER; the file descriptor of the TL file
 ENTNO - INTEGER; the logical entry number
 FCDTI - INTEGER; the file code of the TI file
 FCDTV - INTEGER; the file code of the TV file
  NDIM - INTEGER; the dimension of the data set
LNKPTR - INTEGER; the link pointer
  NAME - INTEGER array (TRELEN); the name of the
                                 data set
```

REFERENCED BY:

COPY, CRANDTS, CREATR, DELETR

SUBPROGRAMS REFERENCED:

FPUT, ITREAL

HISTORY:

designed by Dave Birnbaum May 24, 1978

programmed by Donna Morris July 26,1979

PROGRAM NAME: TLPHDR

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TLPHDR writes the number of entries element (NOE) and the alphabetic link pointer element (LNKPTR) to the TL file header.

PROGRAM USAGE:

```
CALL TLPHDR(FIDTL,NOE,LNKPTR)
```

INPUT ARGUMENTS:

FIDTL - INTEGER; the file descriptor of the TL file

NOE - INTEGER; the number of entries

LNKPTR - INTEGER; the link pointer which points to the first entry in the alphabetically linked list

FILES:

TL - tree list
Instrumetation files

REFERENCED BY:

TLADD, CREATR, DELETR

SUBPROGRAMS REFERENCED:

FPUT

HISTORY:

designed by Dave Birnbaum July 5, 1978

programmed by Donna Morris July 26,1979

OLPARS Program Specifications
TLSRCH

PROGRAM NAME: TLSRCH

CATEGORY: Tree List File Access Routine

PROGRAM DESCRIPTION:

TLSRCH is passed the file descriptor (in FIDTL) of the tree list file and the name of a tree (in NAME). TLSRCH then searches the tree list for a tree with the same name. If a match is found, TLSRCH is set equal to true and the file codes and dimension of the tree are placed in FCDTI, FDCTV and NDIM, respectively. If no match is found, NDIM and ENTNO are set to zero and FCDTI and FCDTV are set to equal the entry number and link pointer, respectively, needed to update the tree list file alphabetic linked list when a new entry with the given name (NAME) is added to the tree list.

PROGRAM USAGE:

```
LOGICAL TLSRCH
        .
        .
        .
IF (TLSRCH(FIDTL,ENTNO,NAME,FCDTI,FCDTV,NDIM))
```

INPUT ARGUMENTS:

FIDTL - INTEGER; the file descriptor of the TL file

NAME - INTEGER array (TRELEN); name of the data tree

OUTPUT ARGUMENTS:

ENTNO - INTEGER; if tlsrch is true, ENTNO contains the entry number of the TL entry found.

FCDTI - INTEGER; if tlsrch is true, FCDTI is the file code of the TI file. if tlsrch is false, FCDTI is the entry number of the treename that alphabetically precedes the position where NAME belongs in the linked list and is used by the calling routine to update the list if the entry NAME is added to the tree list.

FCDTV - INTEGER; if tlsrch is true, FCDTV is the file code of the TV file. if tlsrch is false, FCDTV is the link pointer that should be written to the entry NAME in the event that the calling routine adds the entry NAME to the tree list.

NDIM - INTEGER; if tlsrch is true, NDIM is the dimension of the data set. if tlsrch is false, NDIM is set to zero.

ALGORITHM / NOTES:

Tlsrch searches the tree alphabetically by following the linked list pointers.

Read in the TL file header to get the first link pointer

I = link pointer

while (I is not equal to the last link pointer (-1))

get entry I (we will use its name and link pointer)

if (treenames match) then

tlsrch =.true.
break

elseif (NAME > the TL entry name) then

I = link pointer
stay in the loop to get the next entry

else

tlsrch = .false.
return the link information in FCDTI and FCDTV
set NDIM to zero
break

endif

endwhile

```
        if (link pointer = -1)

             tlsrch = .false
             return the link information in FCDTI and FCDTV
             set NDIM to zero

        endif
```

FILES:

TL - time list

REFERENCED BY:

Any routine that must check to see that a data set exists or must use a data set.

SUBPROGRAMS REFERENCED:

TLGHDR, TLGENT

HISTORY:

designed by Dave Birnbaum May 19, 1978

programmed by Donna Morris July 10, 1979

PROGRAM NAME: TMPXRD

CATEGORY: Utility Subroutine

PROGRAM DESCRIPTION:

TMPXRD reads a transposed matrix from a temporary file.

PROGRAM USAGE:

CALL TMPXRD (FDTMP, VECSIZ, VECTOR, COLSTR)

INPUT ARGUMENTS:

FDTMP - INTEGER; file descriptor of the temporary containing the transposed matrix

VECSIZ - INTEGER; size of a row in the transposed matrix

ROW - INTEGER; transposed matrix row to be accessed

COLSTR - INTEGER; column starting position of accessed matrix

OUTPUT ARGUMENTS:

VECTOR - REAL ARRAY (VECSIZ); row of transposed matrix to be filled by this routine

ALGORITHM / NOTES:

Obtain transposed row (FGET)

FILES:

Temporary file

REFERENCED BY:

RNGLAP

SUBPROGRAMS REFERENCED:

FGET, INSINT, INSPGM, INSRET, OEXIT, TRMPUT

SEE ALSO:

TMPXWT

HISTORY:

designed by Steve Haehn July 6, 1981

programmed by Steve Haehn July 6, 1981

PROGRAM NAME: TMPXWT

CATEGORY: Utility Subroutine

PROGRAM DESCRIPTION:

TMPXWT writes a transposed matrix to a temporary file.

PROGRAM USAGE:

CALL TMPXWT (FDTMP, VECSIZ, VECTOR, COLUMN)

INPUT ARGUMENTS:

FDTMP - INTEGER; file descriptor of the temporary containing the transposed matrix

VECSIZ - INTEGER; size of a row in the transposed matrix

VECTOR - REAL ARRAY (VECSIZ); row of matrix being transposed

COLUMN - INTEGER; transposed matrix column to be accessed

ALGORITHM / NOTES:

DO vector index = 1 to VECSIZ

Write vector to temporary file (FPUT)

ENDDO

FILES:

Temporary file

REFERENCED BY:

RNGLAP

OLPARS Program Specifications
TMPXWT

SUBPROGRAMS REFERENCED:

FPUT, INSINT, INSPGM, INSRET

SEE ALSO:

TMPXRD

HISTORY:

designed by Steve Haehn July 6, 1981

programmed by Steve Haehn July 6, 1981

PROGRAM NAME: TRANSFRM

CATEGORY: MEASUREMENT EVALUATION COMMAND (subsidiary)

PROGRAM DESCRIPTION:

TRANSFRM creates a new tree whose structure is identical to the Current Data Set using those measurements (for each vector) which have been "selected" during measurement evaluation (those measurements which have their asterisk flag set).

PROGRAM USAGE:

User types in 'TRANSFRM'.

USER INTERACTION:

The user names a new data tree.

FILES:

CM - communication
DI - display information
TL - tree list
TI - tree information (current data set)
TV - tree vector (current data set)
TI - tree information (new data set)
TV - tree vector (new data set)
HS - history (indirectly used)
OPTION.OLP - option file (indirectly used)

REFERENCED BY:

OLPARS user.

SUBPROGRAMS REFERENCED:

ADUPCM, CDSCHK, CLOSFX, CLOSTR, CMGCDS, CMGOTH, CMNCOV, COLAPS, CREATR, DIGROH, DIGROI, ERASE, GNXTND, INSCHR, INSFLT, INSINI, INSINT, INSRET, MENU, OEXIT, OPENFX, OPENTR, REDVEC, TIGCAP, TIGCOP, TIGCOV, TIGET, TIGMN, TIPCAP, TIPCOP, TIPCOV, TIPET, TIPHDR, TIPMN, TIPNAM, TRMPUT, TVGVEC, TVPHDR, TVPVEC, UIXFRM

DIAGNOSTICS:

An error message is printed if no measurements are "set" and TRANSFRM exits.

OLPARS Program Specifications
TRANSFRM

HISTORY:

designed	by	Kermit Klingbail	April 13, 1978
programmed	by	Donna Morris	March 1, 1981

PROGRAM NAME: TRMGET

CATEGORY: Terminal I/O (character, system dependent)

PROGRAM DESCRIPTION:

TRMGET is a terminal input function subprogram. It reads characters from the terminal, interprets them according to a format, and stores the result in its arguments. The format string usually contains specifications which are used to direct interpretation of input sequences. Refer to Section 6 of OLPARS VI Programmer's Reference Manual.

PROGRAM USAGE:

```
INTEGER TRMGET
      .
      .
      .
N = TRMGET('FORMAT STRING',ARG1,ARG2, ... )
```

INPUT ARGUMENTS:

FORMAT STRING - HOLLERITH string containing conversion specifications for variables to be read

ARG1,ARG2 ... - addresses to place the internal form of the arguments being read

OUTPUT ARGUMENTS:

TRMGET - INTEGER; the number of succesfully matched and assigned input items,

-1 ; program termination symbol was read

-2 ; universal help symbol was read

ALGORITHM/NOTES:

A format string may contain:

- The special string characters representing blanks, tabs, newlines, backspaces, linefeeds, carriage returns, or formfeeds, which are ignored (these will be known as 'space' characters).
- Ordinary characters (not '$') which are expected to match the next non-space character of the input stream.

- Conversion specifications, consisting of the character '$', an optional assignment suppressing character '/', an optional space-skipping-suppression character '-', an optional numerical field repetition factor, a conversion character, and an optional numerical 'MAXIMUM' field width specifier (i.e., the field to be read may actually be smaller than specified, but not larger than specified.)
  Note, the maximum field width specifier may be replaced by the character '*', which indicates that the argument following the one currently being processed contains the value of the maximum field width.

A conversion specification is used to direct the conversion of the next input field; the result is placed in the variable pointed to by the corresponding argument, unless assignment suppresion was indicated by the '/' character. The assignment suppression character '/' directs TRMGET to skip over the specified type of field in the input stream. An input field is defined as a string of non-space characters. However, an exception can occur when using an 'A' or 'S' conversion specification. See 'A' description.

The following conversion characters are permissable:

$ indicates that a single '$' character is expected in the input stream at this point; no assignment is done.

I indicates that a decimal integer is expected in the input stream; the corresponding argument should be of type integer.

H indicates that a decimal integer is expected in the input stream; the corresponding argument should be of type half integer (i.e., in DEC FORTRAN the type is LOGICAL*1 or BYTE).

L indicates that a decimal integer is expected in the input stream; the corresponding argument should be of type long integer (i.e., in DEC FORTRAN the type is INTEGER*4).

A indicates that a character string is expected in the input stream; the corresponding argument should be an integer array large enough to accept the string and an end-of-string (EOS = 0, the null character) symbol, which will be added. If an optional field width is not specified, the input record is terminated by either a space character or a newline.

If an optional field width is specified, only a newline may terminate the input record before the end of the field is reached. Thus, space characters may be embedded in the output field only if a field width is specified.

S indicates that a character string is expected; this specification is identical to the 'A' specification except that the corresponding argument should be a HOLLERITH field (i.e., in DEC FORTRAN this is the variable LOGICAL*1).

E
F indicates that a floating point number is expected in the input stream; the corresponding argument should be a single precision 'REAL' variable. The input format for a floating point number is a string of numbers, possibly containing a leading minus sign, followed by an optional exponent field containing an 'E' or 'D', followed by a possible signed integer.

D indicates that a floating point number is expected in the input stream; the corresponding argument should be a double precision 'REAL' variable. The input format is identical to that of the 'E', 'F' format.

C indicates that a single character is expected in the input stream; the corresponding argument should be an integer variable. Note, no EOS symbol is tacked on the end of the character in the output field.

TRMGET returns, as its value, the number of successfully matched and assigned input items. This can be used to decide how many input items were found.

If a program termination symbol (a null line or carriage return) is encountered by TRMGET, a -1 is returned.

If the OLPARS universal help symbol (?) is found as the first non-space character in the input record, a -2 is returned.

REFERENCED BY:

All OLPARS commands

SUBPROGRAMS REFERENCED:

ARGCHK, ERROR, GETNUM, TRMINT
ICI$, RCI$, $FCHNL (F4P-SYSTEM LIBRARY),
.GETSQ (SYSTEM LIBRARY)

DIAGNOSTICS:

ERROR MESSAGES ARE TAKEN CARE OF BY THE 'ERROR' MODULE.

SEE ALSO:

TRMPUT, FILPUT, FILGET, FLUSHT, FLUSHF

HISTORY:

designed by Steven Haehn July 4, 1978

programmed by Curry Bartlett Feb. 22, 1979

modified by Dave Tipton Mar. 11, 1979

(I added the assignment suppression and the repeat factor options.)

modified by Steven Haehn Sept. 4 1979

(I added the FLUSHT and FLUSHF entry points)

PROGRAM NAME: TRMOPN

CATEGORY: OLPARS utility (system dependent)

PROGRAM DESCRIPTION:

TRMOPN will use RSX-11M V3.1 file control service routines to open the terminal TT0:, which is a variable record I/O file, for read, write(update), or append.

PROGRAM USAGE:

```
INTEGER TRMOPN
        .
        .
        .
N = TRMOPN (FILNAM,ACCESS)
```

INPUT ARGUMENTS:

FILNAM - LOGICAL*1 array; file name represented by ASCII characters

ACCESS - INTEGER; 0 = file open for read only
1 = file open for write (update under RSX-11M)
2 = file open for append

ALGORITHM / NOTES:

The logical unit number is the function value returned.

If an error is found, the value returned is -1.

REFERENCED BY:

Level I subroutines and utility programs

HISTORY:

designed by Steve Haehn March 12, 1979

programmed by Dave Tipton March 12, 1979

PROGRAM NAME: TRMPUT

CATEGORY: Terminal I/O (character, system dependent)

PROGRAM DESCRIPTION:

TRMPUT is a terminal output subprogram. It formats, converts, and prints its arguments to a user's terminal, under control of a format string. The format string will contain three types of objects: plain characters, which are simply copied to the terminal; special characters, which are used for cursor control; and conversion specifications, which are used for converting and printing arguments which follow the format string. Refer to Section 6 of OLPARS Programmer's Reference Manual.

PROGRAM USAGE:

```
CALL TRMPUT('FORMAT STRING',ARG1,ARG2, ... )
```

INPUT ARGUMENTS:

FORMAT STRING - HOLLERITH string containing characters to be printed out and conversion specifications for variables to be printed out

ARG1,ARG2 ... - arguments to be printed out according to a converson specification found in the format string

ALGORITHM/NOTES:

Each conversion specification found in the format string begins with the character '$'. Following the '$' there may be:

- an optional plus or minus sign which specifies right or left adjustment of the converted argument in the indicated field.
- an optional digit string representing a field repetition factor. If it is not present, it is assumed to be 1.
- a character which indicates the type of conversion to be applied.

- an optional digit string specifying a 'MINIMUM' field width (i.e., the field to be printed may actually be larger than specified, but it will not be smaller than specified); if the converted argument has fewer characters than the field width, it will be padded (on left or right, depending on the field conversion type and the field adjustment indicator) with blanks to make up the field width. If the character '*' is placed in this position, the next argument in the list (the one that follows the argument to be printed) is used to obtain the minimum field width.

- an optional period ('.') which serves to separate the field width from the next digit string.

- an optional digit string (the precision) which specifies the number of digits to be printed to the right of the decimal point of a single or double precision number, or the maximum number of characters to be printed from a character string. If the character '*' is placed in this position, the next argument in the list of arguments is used to obtain the maximum field width.

The conversion characters and their meanings are as follows:

I - The argument (1 word long) is converted to a decimal integer. If an adjustment indicator is not present in the conversion specification, the resulting output character string will be right adjusted within its field.

H - The argument is considered a half integer (1 byte long) Default field adjustment is 'right'.

L - The argument is considered a long integer (2 words long) Default field adjustment is 'right'.

O - The argument (1 word long) is printed out as an octal number. The resulting output string wil be right adjusted in its output field, when a justification indicator is not present.

A - The argument is taken to be a string of characters. The characters are stored in an integer array, one character per integer. Characters from the string are printed until an end of string character (0) is reached, or until the number of characters, indicated by the precision, is exhausted. If an adjustment indicator is not present in the conversion specification, the resulting output character string will be left justified in its output field.

S - The argument is taken to be a character string. The only difference between the 'S' and 'A' conversion specifiers is that the characters to be printed are stored in the FORTRAN hollerith data type, instead of the FORTRAN integer data type.

F - The argument is taken to be a single precision floating point number and is converted to the decimal notation of the form [-]mmm.nnnnnnn, where the length of the string of n's is defined by the 'precision' specification. The default 'precision' is 7. Default field adjustment is to the right.

E - The argument is taken to be a single precision floating point number and is converted to the decimal notation of the form [-]m.nnnnnnnE(+ or -)ee, where the length of the string of n's is defined by the 'precision' specification. The default precision is 7. The default field adjustment is to the right.

D - The argument is taken to be a double precision floating point number and is converted to the decimal notation of the form [-]m.nnnnnnnnnnnnnnnnD(+, -)ee, where the length of the string of n's is defined by the 'precision' specification. The default precision is 16. Default field adjustment is to the right.

If no recognizable character appears after the '$', that character is printed; thus '$' may be printed by the usage of the string '$$'.

REFERENCED BY:

All OLPARS commands, PROMPT

SUBPROGRAMS REFERENCED:

ARGCHK,GETNUM,TRMINT

DIAGNOSTICS:

TRMPUT signals its failure to write to the user's terminal by an appropriate message.

SEE ALSO:

TRMGET

HISTORY:

designed by Steven Haehn July 4, 1978

programmed by Curry Bartlett Jan. 12, 1979

modified by Steven Haehn Feb. 21, 1979

(I added the capability of printing out fortran hollerith fields (module FLDS) and redocumented program. The variable field width capability (usage of a variable field indicator (VFI) in the format string) was also added (i.e., modifications were made to the 'GETNUM' and 'FLDOPR' subroutines).)

modified by Dave Tipton April 11, 1979

(I added the half integer field conversion capability and combined the half, single, and double integer field conversion routines into one module. I also made GETNUM a global subroutine, which resides in the module TRMGLB, along with other buffers and variables I globalized for compatibility with TRMGET.)

OLPARS Program Specifications
TVGCLS

PROGRAM NAME: TVGCLS

CATEGORY: Level II file Access Routine

PROGRAM DESCRIPTION:

TVGCLS retrieves the class symbol of a given data vector.

PROGRAM USAGE:

```
CALL TVGCLS(FDTV,VECNO,CLSSYM)
```

INPUT ARGUMENTS:

FDTV - INTEGER; file descriptor of the Tree vector file
VECNO - INTEGER; the entry no. (vector) being accessed

OUTPUT ARGUMENTS:

CLSSYM - INTEGER; the class symbol

FILES:

TV - tree vector

SUBPROGRAMS REFERENCED:

FGET, INSINT, INSPGM, INSRET, OEXIT, TRMPUT

SEE ALSO:

TVPCLS

HISTORY:

designed by Donna Morris October 4, 1981

programmed by Donna Morris October 4, 1981

PROGRAM NAME: TVGHDR

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TVGHDR is passed the file descriptor of the TV file. It returns the name of the last logic applied to the tree and the file code numbers of that logic.

PROGRAM USAGE:

```
CALL TVGHDR(FID,LOGNAM,FCDLI,FCDLV,NOE)
```

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the TV file

OUTPUT ARGUMENTS:

LOGNAM - INTEGER array (TRELEN); the name of the logic
FCDLI - INTEGER; the file code of the LI file
FCDLV - INTEGER; the file code of the LV file
NOE - INTEGER; next open entry at end of file

FILES:

TV - tree vector

REFERENCED BY:

COPYND

SUBPROGRAMS REFERENCED:

FGET

SEE ALSO:

TVPHDR

HISTORY:

designed by David Birnbaum May 26, 1978

programmed by Steven Haehn July 27, 1979

PROGRAM NAME: TVGLOG

CATEGORY: Level II file Access Routine

PROGRAM DESCRIPTION:

TVGLOG retrieves the logic indicator of a given data vector. The logic indicator (or temporary logic element) shows at which logic node the vector lies.

PROGRAM USAGE:

```
CALL TVGLOG(FDTV,VECNO,LOGIND)
```

INPUT ARGUMENTS:

FDTV - INTEGER; file descriptor of the Tree vector file
VECNO - INTEGER; the entry no. (vector) being accessed

OUTPUT ARGUMENTS:

LOGIND - INTEGER; logic indicator

FILES:

TV - tree vector

REFERENCED BY:

CREVAL

SUBPROGRAMS REFERENCED:

FGET, INSINT, INSPGM, INSRET, OEXIT, TRMPUT

SEE ALSO:

TVPLOG

HISTORY:

designed by Steven Haehn Mar. 5, 1981

programmed by Steven Haehn Mar. 11, 1981

PROGRAM NAME: TVGVEC

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TVGVEC is passed the file descriptor of the TV file and the entry number of the vector desired. TVGVEC returns the following information:

- The one-character class symbol.
- The identification number of the vector.
- The temporary logic element corresponding to the vector.
- The measurements of the vector.

PROGRAM USAGE:

```
CALL TVGVEC(FID,ENTNO,NDIM,CLASS,ID,LOGIK,MEAS)
```

INPUT ARGUMENTS:

```
  FID - INTEGER; the file descriptor of the TV file
ENTNO - INTEGER; the entry number of the vector
 NDIM - INTEGER; the dimension of the vector
```

OUTPUT ARGUMENTS:

```
CLASS - INTEGER; the class symbol
   ID - INTEGER; the ID of the vector
LOGIK - INTEGER; the logic element
MEAS  - REAL array (NDIM); the measurements
```

FILES:

```
TV - tree vector
Instrumentation files
```

REFERENCED BY:

COPYVC,DSPROG

SUBPROGRAMS REFERENCED:

FGET

OLPARS Program Specifications
TVGVEC

HISTORY:

designed by Dave Birnbaum May 20, 1978

programmed by David Tipton June 13, 1979

PROGRAM NAME: TVPCLS

CATEGORY: Level II file Access Routine

PROGRAM DESCRIPTION:

TVPCLS writes the class symbol of a given data vector.

PROGRAM USAGE:

CALL TVPCLS(FDTV,VECNO,CLSSYM)

INPUT ARGUMENTS:

FDTV - INTEGER; file descriptor of the Tree vector file
VECNO - INTEGER; the entry no. (vector) being accessed
CLSSYM - INTEGER; the class symbol

FILES:

TV - tree vector

REFERENCED BY:

REARR

SUBPROGRAMS REFERENCED:

FPUT, INSINT, INSPGM, INSRET

SEE ALSO:

TVGCLS

HISTORY:

designed by Donna Morris October 4, 1981

programmed by Donna Morris October 4, 1981

PROGRAM NAME: TVPHDR

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TVPHDR writes the header information into the TV file.

PROGRAM USAGE:

```
CALL TVPHDR(FID,LOGNAM,FCDLI,FCDLV,NOE)
```

INPUT ARGUMENTS:

```
   FID - INTEGER; the file descriptor of the TV file
LOGNAM - INTEGER array (TRELEN); the logic name
 FCDLI - INTEGER; the file code of the LI file
 FCDLV - INTEGER; the file code of the LV file
   NOE - INTEGER; next open entry at end of file
```

FILES:

TV - tree vector

REFERENCED BY:

COPYND

SUBPROGRAMS REFERENCED:

FPUT

SEE ALSO:

TVGHDR

HISTORY:

designed by David Birnbaum May 26, 1978

programmed by Steven Haehn July 27, 1979

PROGRAM NAME: TVPLOG

CATEGORY: Level II file Access Routine

PROGRAM DESCRIPTION:

TVPLOG writes the logic indicator of a given data vector. The logic indicator (or temporary logic element) shows at which logic node the vector lies.

PROGRAM USAGE:

```
CALL TVPLOG(FDTV,VECNO,LOGIND)
```

INPUT ARGUMENTS:

```
  FDTV - INTEGER; file descriptor of the Tree vector file
 VECNO - INTEGER; the entry no. (vector) being accessed
LOGIND - INTEGER; logic indicator
```

FILES:

TV - tree vector

REFERENCED BY:

CREVAL

SUBPROGRAMS REFERENCED:

FPUT, INSINT, INSPGM, INSRET

SEE ALSO:

TVGLOG

HISTORY:

designed by Steven Haehn Mar. 5, 1981

programmed by Steven Haehn Mar. 11, 1981

PROGRAM NAME: TVPVEC

CATEGORY: Level II File Access Subroutine

PROGRAM DESCRIPTION:

TVPVEC is passed the file descriptor of the TV file and the entry number of the vector to write. TVPVEC writes the following information:

- The one-character class symbol.
- The identification number of the vector.
- The temporary logic element corresponding to the vector.
- The measurements of the vector.

PROGRAM USAGE:

```
CALL TVPVEC(FID,ENTNO,NDIM,CLASS,ID,LOGIC,MEAS)
```

INPUT ARGUMENTS:

FID - INTEGER; the file descriptor of the TV file being accessed

ENTNO - INTEGER; entry number of the vector desired

NDIM - INTEGER; the vector dimensionality

CLASS - INTEGER; the class symbol

ID - INTEGER; the vector ID

LOGIC - INTEGER; the logic element

MEAS - REAL array (NDIM); the coordinates

FILES:

TV - tree vector

REFERENCED BY:

COPYVC

SUBPROGRAMS REFERENCED:

FPUT

HISTORY:

designed by Dave Birnbaum May 26, 1978

programmed by David Tipton July 10,1979

OLPARS Program Specifications
TYPE

PROGRAM NAME: TYPE

CATEGORY: Utility Routine (system dependent)

PROGRAM DESCRIPTION:

The function TYPE determines whether a particular character is a letter, digit, printable ASCII that is not a letter or digit, or something else.

The function returns:
1 - letter
2 - digit
3 - printable ASCII
4 - something else

PROGRAM USAGE:

```
INTEGER TYPE
   .
   .
IF(TYPE(CHAR).EQ.LETTER))
```

INPUT ARGUMENTS:

CHAR - INTEGER; the character to be tested

OUTPUT ARGUMENTS:

TYPE - INTEGER; character type of variable CHAR

REFERENCED BY:

IDCHK

HISTORY:

designed by Steven Haehn June 2, 1978

programmed by Steven Haehn April 8, 1980

PROGRAM NAME: UIGNOR

CATEGORY: Utility Subroutine

PROGRAM DESCRIPTION:

UIGNOR prompts the user for measurements to ignore, and returns the ignore measurement mask vector to the calling program.

PROGRAM USAGE:

CALL UIGNOR (PMODE,NDIM,CONTIN,IGNOR,IGNMES)

INPUT ARGUMENTS:

PMODE - INTEGER; the prompt mode flag

NDIM - INTEGER; the vector dimensionality

OUTPUT ARGUMENTS:

CONTIN - LOGICAL; continuation-flag,
false if the user wants to quit

IGNOR - INTEGER array (MAXDIM+1); the mask vector
the last element stores the number of
measurements ignored.

IGNMES - INTEGER; ignore measurement flag
1 - ignore measurements
0 - don't ignore measurements

ALGORITHM / NOTES:

See if user wants to ignore measurements (PROMPT).

IF (the user wants to ignore measurements) THEN

Set ignore measurement flag on.

```
        REPEAT

            Get the measurement to be ignored.

            IF (all measurements have been ignored) THEN

                Tell user he has ignored all measurements,
                and go back and start over.

            ELSEIF (measurement has been previosly ignored) THEN

                Tell user he has ignored this one already and
                get next measurement.

            ELSEIF (user entered a zero) THEN

                Break out of loop.

            ELSE

                Place this measurement number in working array.

            ENDIF

        UNTIL (User is done)

        DO WHILE (There are more measurements to process)

            Use entered array as index into mask vector,
            and set these elements to zero.

        ENDDO

        Set last entry to number of measurements ignored.

        RETURN
```

FILES:

Instrumentation

REFERENCED BY:

NMVBSU

SUBPROGRAMS REFERENCED:

EQUALA, INSPGM, INSRET, PROMPT, TRMGET, TRMPUT

SEE ALSO:

NMVBSU

HISTORY:

designed by John W. Tenney May 14, 1981

programmed by John W. Tenney May 25, 1981

OLPARS Program Specifications
UIXFRM

PROGRAM NAME: UIXFRM

CATEGORY: TRANSFORMATION ROUTINE

PROGRAM DESCRIPTION:

UIXFRM prompts the user for the name of the new data tree to be created as the result of a transformation.

PROGRAM USAGE:

```
INTEGER UIXFRM
        .
        .
        .
RESPONSE = UIXFRM(PMODE,TLFD,ODTREE,NDTREE)
```

INPUT ARGUMENTS:

PMODE - INTEGER; the prompt mode

TLFD - INTEGER; the file descriptor of TL

ODTREE - INTEGER (8); treename of current data set

OUTPUT ARGUMENTS:

NDTREE - INTEGER (8); user-selected name for the new tree

UIXFRM - INTEGER; 1 = the user response was ok
-1 = the user wants to quit

USER INTERACTION:

If the name entered by the user is the same as the current tree name, UIXFRM gives an error message and continues to prompt the user. If the treename entered already exists, the user is asked if the existing tree should be destroyed. If the user responds with yes, the name entered by the user is returned in 'NDTREE', otherwise the user is asked again to enter a treename.

FILES:

TL - Tree List File

REFERENCED BY:

FIGNXFRM, NORMXFRM, TRANSFRM

SUBPROGRAMS REFERENCED:

EQUALA, INSCHR, INSPGM, INSRET, LGLTRE,
PROMPT, TLSRCH, TRMGET, TRMPUT

HISTORY:

designed by Donna Morris February 15, 1981

programmed by Donna Morris February 15, 1981

PROGRAM NAME: UNION

CATEGORY: Measurement Evaluation Command (subsidiary)

PROGRAM DESCRIPTION:

For each class/class pair entry in the DV file, UNION determines which measurement best discriminates that class/class pair (that is which measurement number corresponds to the smallest or largest discriminant value), and sets the asterisk flag for the given measurement. A union by class reads the class entries in the DV file and a union by class pair reads the class pair entries in the DV file. UNION displays an overall rank order display.

PROGRAM USAGE:

User types in 'UNION'.

USER INTERACTION:

The user is asked if he wants to perform a union by class or a union by class pair.

FILES:

CM - communication
DI - display information
DV - display values

REFERENCED BY:

OLPARS user.

SUBPROGRAMS REFERENCED:

CDSCHK, CMGCDS, CMGOTH, DIGROH, DIGROI, DIPROH, DIPROI, DVGROE, ERASE, INSINI, INSINT, INSRET, MENU, OEXIT, OPENFX, PROMPT, RODISP, RSORTD, TRMGET, TRMPUT

HISTORY:

designed by Kermit Klingbail April 13, 1978

programmed by Donna Morris January 21, 1981

PROGRAM NAME: UNIQND

CATEGORY: Data Tree File Access Routine

PROGRAM DESCRIPTION:

UNIQND is passed the file descriptor (in FID) of the TI file, its entry table (ET), the entry slot number (SENIOR) of the senior node, a table of entry slots (TABLE) corresponding to the lowest nodes to be disregarded in the search, the number of such nodes (NUM) and the name to be checked (NAME).

UNIQND is used by routines such as COMNOD to check for uniqueness of a nodename and uniqueness of the display symbol. Certain nodes may be disregarded in the search, e.g., nodes to be combined in COMNOD. UNIQND assumes that nodenames and display symbols (the first character of the nodename) are unique.

UNIQND returns the following:

0 - if the nodename is unique and its display character is unique.

1 - the nodename is not unique.

2 - the display symbol is not unique.

UNIQND will also be used by structure analysis routines in checking uniqueness of user input names. In these routines, the node upon which the structure analysis has taken place should be designated as excluded. New nodenames may not equal the name of this node, but may have the same display character (because the excluded node will no longer be a lowest node). When used by structure analysis routines, NUM should be set equal to 1. NUM should be >= 2 when UNIQND is used by other routines.

PROGRAM USAGE:

```
INTEGER UNIQND
        .
        .
        .
N = UNIQND(FDTI,ENTBLE,SENIOR,TABLE,NUM,NAME)
```

OLPARS Program Specifications
UNIQND

INPUT ARGUMENTS:

FDTI - INTEGER; the file descriptor of the TI file

ENTBLE - INTEGER array (TABSIZ); the entry table

SENIOR - INTEGER; the entry table slot number of the senior node

TABLE - INTEGER array (NUM); a table of entry slots to be disregarded

NUM - INTEGER; the number in TABLE

NAME - INTEGER array (NODLEN); the name to be checked

OUTPUT ARGUMENTS:

UNIQND - INTEGER; see PROGRAM DESCRIPTION

ALGORITHM / NOTES:

```
UNIQND = UNIQUE (0)

Get counts and pointers from senior node.

Present node = senior node.

FINISH = FALSE

 WHILE (UNIQND = UNIQUE and FINISH = FALSE)

    IF (present node is not an excluded lowest node) THEN

        Check name of present node with Name.

        IF (these are not equal) THEN

            Check equality of display symbol of present
            node with display symbol of NAME.

            IF (these are equal) THEN

                UNIQND = SAMSYM (2)
            ENDIF

        ELSE

            UNIQND = SAMNAM (1)
        ENDIF
```

```
            ELSE
                IF (NUM <= 1) THEN
                    Check name of present node with NAME.
                    IF (these are equal) THEN
                        UNIQND = SAMANM (1)
                    ENDIF
                ENDIF
            ENDIF
            IF (UNIQND = UNIQUE (0)) THEN
                Get next node.
                IF ('gnxtnd' returns a false value) THEN
                    FINISH = TRUE
                ENDIF
            ENDIF
        ENDWHILE
```

FILES:

TI - tree information

REFERENCED BY:

APPEND, COMNOD

SUBPROGRAMS REFERENCED:

EQUALA, GNXTND, INSINT, INSLOG, INSPGM, INSRET
TIGCAP, TIGNAM

HISTORY:

designed by David Birnbaum June 8, 1980

programmed by Mark Maginn August 8, 1980

PROGRAM NAME: VALUEA

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

'VALUEA' conditionally returns to the calling program the numeric equivalent of the given character argument. When a unique numeric value is desired for the given character, the 'EXACT' switch should be set to 'true'. When 'exact' is set to 'false', all lower-case character values will be equivalent to their upper-case values.

PROGRAM USAGE:

```
INTEGER VALUEA
        .
        .
        .
N=VALUEA(CHAR,EXACT)
```

INPUT ARGUMENTS:

CHAR - INTEGER; character to be translated into a numeric value

EXACT - LOGICAL; flag indicating exact translation

OUTPUT ARGUMENTS:

VALUEA - INTEGER; numeric equivalent of 'CHAR'

REFERENCED BY:

EQUALA

HISTORY:

designed by Steve Haehn February 26,1979

programmed by Steve Haehn February 26,1979

PROGRAM NAME: VALUES

CATEGORY: Utility Subroutine (system dependent)

PROGRAM DESCRIPTION:

'VALUES' conditionally returns to the calling program the numeric equivalent of the given character argument. When a unique numeric value is desired for the given character, the 'EXACT' switch should be set to 'true'. When 'exact' is set to 'false', all lower-case character values will be equivalent to their upper-case values.

PROGRAM USAGE:

```
INTEGER VALUES
      .
      .
      .
N=VALUES(CHAR,EXACT)
```

INPUT ARGUMENTS:

CHAR - LOGICAL *1; character to be translated into a numeric value

EXACT - LOGICAL; flag indicating exact translation

OUTPUT ARGUMENTS:

VALUES - INTEGER; numeric equivalent of 'CHAR'

REFERENCED BY:

EQUALS

HISTORY:

designed by Steve Haehn February 26,1979

programmed by Steve Haehn February 26,1979

PROGRAM NAME: VCREAT

CATEGORY: OLPARS utility (system dependent)

PROGRAM DESCRIPTION:

VCREAT will create a variable length record I/O file using file control service routines under RSX-11M V3.1.

PROGRAM USAGE:

```
INTEGER VCREAT
        .
        .
        .
N = VCREAT (FILNAM,SAVFLG)
```

INPUT ARGUMENTS:

FILNAM - LOGICAL*1 array; file name represented by ASCII characters

SAVFLG - INTEGER; 0 = file to be saved after closing
1 = file to be deleted upon closing

ALGORITHM / NOTES:

The logical unit number is the function value returned.

If an error is found, the value returned is -1.

REFERENCED BY:

Level I subroutines and utility programs

HISTORY:

designed by Steve Haehn January 19, 1979

programmed by Dave Tipton March 12, 1979

PROGRAM NAME: VOPEN

CATEGORY: OLPARS utility (system dependent)

PROGRAM DESCRIPTION:

VOPEN will use RSX-11M V3.1 file control service routines to open a variable length record I/O file for read, write (update), or append.

PROGRAM USAGE:

```
INTEGER VOPEN
        .
        .
        .
N = VOPEN (FILNAM,ACCESS)
```

INPUT ARGUMENTS:

FILNAM - LOGICAL*1 array; file name represented by ASCII characters

ACCESS - INTEGER; 0 = file open for read only
1 = file open for write (update under RSX-11M)
2 = file open for append

ALGORITHM / NOTES:

The logical unit number is the function value returned.

If an error is found, the value returned is -1.

REFERENCED BY:

Level I subroutines and utility programs

HISTORY:

designed by Steve Haehn January 19, 1979

programmed by Dave Tipton March 12, 1979

OLPARS Program Specifications
WRITEB

PROGRAM NAME: WRITEB

CATEGORY: Utility subroutine (system dependent)

PROGRAM DESCRIPTION:

WRITEB (write block) performs the actual disk write of all OLPARS fixed length record files. Under RSX-11M, this routine is an assembly routine. The type of writing being done is called a block write. For further information, see an RSX-11M I/O operations manual.

PROGRAM USAGE:

CALL WRITEB(LUN,BLKNO,BUFFER,BUFSIZ,IOSB)

INPUT ARGUMENTS:

LUN - INTEGER; logical unit number of file being written

BLKNO - INTEGER *4; an RSX-11M long integer representing the virtual block to be written

BUFSIZ - INTEGER; size (IN BYTES) of the virtual block to be written

BUFFER - INTEGER ARRAY(BUFSIZ/2); the virtual block written is stored here

OUTPUT ARGUMENTS:

IOSB - INTEGER ARRAY(2); the I/O status block

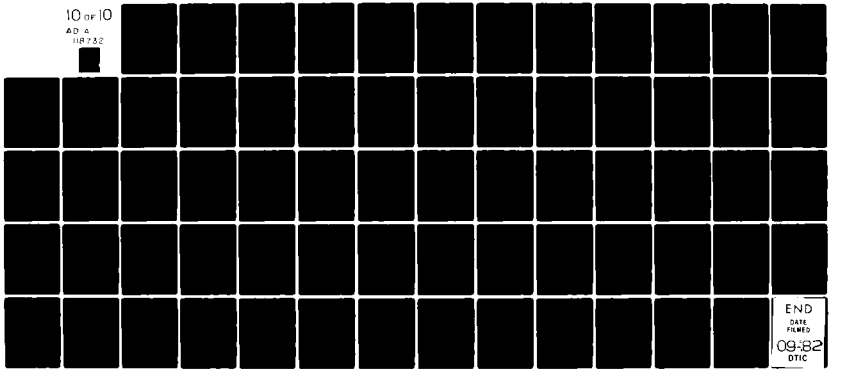
IOSB(1) = File Control Services (FCS) error code in lower byte

IOSB(2) = actual number of bytes written (it will be equal to BUFSIZ)

ALGORITHM / NOTES:

A requirement for usage will be that the files to be written must be 'opened' with 'BUFFER COUNT = -1' in a DEC F4P open statement, or that a -1 is placed in the offset F.MBCT in the file descriptor block (FDB) of the file from an assembly 'open' program. (Refer to an RSX-11M FORTRAN user's guide.)

AD-A118 732 PAR TECHNOLOGY CORP NEW HARTFORD NY F/G 9/2
ON-LINE PATTERN ANALYSIS AND RECOGNITION SYSTEM. OLPARS VI. SOF--ETC(U)
JUN 82 S E HAEHN, D MORRIS
UNCLASSIFIED PAR-82-20 NL



REFERENCED BY:

FGET, FPUT

SUBPROGRAMS REFERENCED:

$FCHNL - obtains FORTRAN FDB, when given a logical unit number
.WRITE - assembly 'block' write program
.WAIT - causes WRITEB to wait until BUFFER is written

DIAGNOSTICS:

If an invalid logical unit number is encountered, WRITEB will abort.

SEE ALSO:

READB

HISTORY:

designed by	Steven Haehn	Dec. 15, 1978
programmed by	Steven Haehn	Jan. 15, 1979

PROGRAM NAME: WRTNOW

CATEGORY: Terminal I/O (system dependent)

PROGRAM DESCRIPTION:

WRTNOW writes the present date and time on the terminal screen in the following format (without quotes):

'DATE: dd-mmm-yy hh:mm:ss'

PROGRAM USAGE:

CALL WRTNOW

REFERENCED BY:

CLSCAT

SUBPROGRAMS REFERENCED:

DATE, INSPGM, INSRET, TIME, TRMPUT

HISTORY:

designed by David J. Tipton 12-Jul-79

programmed by David J. Tipton 12-Jul-79

PROGRAM NAME: WRTODS

CATEGORY: Terminal Display Routine

PROGRAM DESCRIPTION:

WRTODS can be used to put up the 'CURRENT OPTION' and 'CURRENT DATA SET' portion of an OLPARS display when the current command is in 'page' mode (i.e., the command has multiple output pages for the user to view).

PROGRAM USAGE:

CALL WRTODS(FDCM,PGMNAM,SCRN)

INPUT ARGUMENTS:

FDCM - INTEGER; file descriptor for the CM file

PGMNAM - INTEGER array (10); Name to be placed under 'CURRENT OPTION:' (NOTE, when PGMNAM(1) = 0 the current option name found in the CM file is used instead.)

SCRN - INTEGER array (19); OLPARS screen coordinates

ALGORITHM / NOTES:

Note, the MENU program normally puts up the current option and current data set names. WRTODS is used when these names are desired on the display without the program menu (normally used when a command must put up several pages of display information).

FILES:

CM - communications

REFERENCED BY:

MICMAC

SUBPROGRAMS REFERENCED:

CMGCDS, CMGOPT, INSCHR, INSPGM, INSRET, TEXT, TRMPUT

OLPARS Program Specifications
WRTODS

HISTORY:

designed by Steven Haehn Oct. 9, 1980

programmed by Steven Haehn Oct. 11, 1980

PROGRAM NAME: XINT

CATEGORY: Numerical Computation Algorithm

PROGRAM DESCRIPTION:

XINT finds the intersection of any line with a horizontal line.

PROGRAM USAGE:

CALL XINT(CRNTPT,BNDYY,BNDYX,YCONST,XLOC)

INPUT ARGUMENTS:

CRNTPT - INTEGER; the current point in both point arrays

BNDYY - INTEGER; the 'y' boundary point array

BNDYX - INTEGER; the 'x' boundary point array

YCONST - INTEGER; the 'y' screen coordinate, either the maxium or minimum 'y' value

OUTPUT ARGUMENTS:

XLOC - INTEGER; the 'x' intersection point

ALGORITHM / NOTES:

Calculate the 'x' intersection point

First we want to fin the slope of the line. The point slope form will be used.

(Y1 -Y2) = (X1 - X2)M where M is the slope

This breaks down to:

$$\frac{Y1 - Y2}{X1 - X2} = M$$

The y - intercept form is now used to find the y intercept.

Y = MX + B where B is the intercept
Y2 = MX2 + B
Y2 - MX2 = B

Now put the slope in for M:

$$Y2 - \frac{(Y1 - Y2)}{(X1 - X2)} X2 = B$$

Now find a common denominator:

$$Y2 - \frac{X2Y1 - X2Y2}{X1 - X2} = B$$

Multiply each term by the denominator:

$$\frac{Y2(X1 - X2)}{X1 - X2} - \frac{X2Y1 - X2Y2}{X1 - X2} = B$$

Distribute and add together:

$$\frac{Y2X1 - Y2X2 - X2Y1 + X2Y2}{X1 - X2} = B$$

$$\frac{Y2X1 - X2Y1}{X1 - X2} = B$$

Now fill in the equation Y = MX + B and solve for x by substituting M, B, and Y:

$$X = \frac{Y - B}{M}$$

$$X = \frac{YC - \frac{Y2X1 - Y1X2}{X1 - X2}}{\frac{Y1 - Y2}{X1 - X2}}$$

Now invert the denominatior and multiply:

$$X = \frac{YC(X1 - X2)}{Y1 - Y2} - \frac{Y2X1 - Y1X2}{X1 - X2} * \frac{X1 - X2}{Y1 - Y2}$$

Now cancel and add togther with common denominator:

$$X = \frac{YC(X1 - X2) - Y2X1 + Y1X2}{Y1 - Y2}$$

This is now the dereived equation for the 'x' intercept.

Check the value of the result and set output argument according to the size of the result

FILES:

Instrumentation file

REFERENCED BY:

REDRAW

SUBPROGRAMS REFERENCED:

INSFLT, INSINT, INSPGM, INSRET

HISTORY:

designed by Mark Maginn September 18, 1980

programmed by Mark Maginn September 18, 1980

OLPARS Program Specifications
YINT

PROGRAM NAME: YINT

CATEGORY: Numerical Computation Algorithm

PROGRAM DESCRIPTION:

YINT finds the intersection of any line with a vertical line.

PROGRAM USAGE:

CALL YINT(CRNTPT,BNDYY,BNDYX,XCONST,YLOC)

INPUT ARGUMENTS:

CRNTPT - INTEGER; the current point in both point arrays

BNDYY - INTEGER; the 'y' boundary point array

BNDYX - INTEGER; the 'x' boundary point array

XCONST - INTEGER; the 'x' screen coordinate, either the maxium or minimum 'x' value

OUTPUT ARGUMENTS:

YLOC - INTEGER; the 'y' intersection point

ALGORITHM / NOTES:

Calculate the 'y' intersection point

First we want to fin the slope of the line. The point slope form will be used.

(Y1 -Y2) = (X1 - X2)M where M is the slope

This breaks down to:

$$\frac{Y1 - Y2}{X1 - X2} = M$$

The y - intercept form is now used to find the y intercept.

Y = MX + B where B is the intercept
Y - MX = B

Now put the slope in for M:

$$Y1 - \frac{(Y1 - Y2)}{(X1 - X2)} X1 = B$$

Now find a common denominator:

$$Y1 - \frac{X1Y1 - X1Y2}{X1 - X2} = B$$

Multiply each term by the denominator:

$$\frac{Y1(X1 - X2)}{X1 - X2} - \frac{X1Y1 - X1Y2}{X1 - X2} = B$$

Distribute and add together:

$$\frac{Y1X1 - Y1X2 - X1Y1 + X1Y2}{X1 - X2} = B$$

$$\frac{X1Y2 - X2Y1}{X1 - X2} = B$$

Now fill in the equation Y = MX + B by
by substituting M, B, and X:

$$Y = \frac{Y1 - Y2}{X1 - X2}(XC) + \frac{Y2X1 - Y1X2}{X1 - X2}$$

Add together because of common denominator:

$$Y = \frac{(Y1 - Y2)X2 + Y2X1 - Y1X2}{X1 - X2}$$

This is now the dereived equation for the 'y'
intercept.

Check the value of the result and set output
argument according to the size of the result

FILES:

Instrumentation file

OLPARS Program Specifications
YINT

REFERENCED BY:

REDRAW

SUBPROGRAMS REFERENCED:

INSFLT, INSINT, INSPGM, INSRET

HISTORY:

designed by Mark Maginn September 17, 1980

programmed by Mark Maginn September 17, 1980

PROGRAM NAME: ZOM1SP

CATEGORY: UTILITY SUBROUTINE

PROGRAM DESCRIPTION:

ZOM1SP allows the user to select a subarea of the current one-space display and have a closer, more detailed examination of that area. In the manner described below the user first selects the left-most point on the baseline, then the right- most point on the baseline to be maginefied. The new display is this area.

PROGRAM USAGE:

```
CALL ZOM1SP (FDCM,FDDI,FDDV,FDPV,FDS1,MINMAX,SCREEN,
             DIHDR,DISCOD,NDIM,ZOMFLG)
```

INPUT ARGUMENTS:

FDCM - INTEGER; file descriptor of the CM file

FDDI - INTEGER; file descriptor of the DI file

FDDV - INTEGER; file descriptor of the DV file

FDPV - INTEGER; file descriptor of the PV file

FDS1 - INTEGER; file descriptor of the S1 file

MINMAX - REAL; array of current min - max values

SCREEN - INTEGER; array of screen coordinate values

DIHDR - INTEGER; array of info. from DI header

DISCOD - INTEGER; the display code

ZOMFLG - INTEGER; the zoom flag from the DI header

USER INTERACTION:

Using the graphics cursor, the user must select the portion of the current display to be "zoomed" upon. When the graphics cursor is on for the first time, the user positions it at the left-most point on the baseline for the new xmin and enters any character. The second time, the user positions it at the right most point on the baseline for the new xmax and enters any character.

ALGORITHM / NOTES:

Turn on the graphics cursor

Get lower left-hand position input by user (GIN).

Get lower right-hand position input by user

Check zoom flag to determine new value of zoom flag

Find the range of the old 'x' points

Compute new current x(min), x(max), and store them in DI file header (DIPMAX).

Write out new values to DI header

Turn the screen coordinate flag in the DV file header 'off' so that the display subroutine "knows" that it must recalculate a new set of screen coordinates for the one-space display.

Put up new two-space display (MICMAC).

NOTE

For one-space projections, computation of the new max,min values from the screen coordinates given by the user, is done by solving for x and y in the equations for scatter screen coordinates (OLPARS Programmer and System Maintenance Manual). If x(L),x(R), are the screen coordinates of the two points entered by the user, this yields:

$$\text{new } x_{min} = \frac{\text{range } (x(L) - E)}{G - E} + \text{xmin}$$

$$\text{new } x_{max} = \frac{\text{range } (x(R) - E)}{G - E} + \text{xmin}$$

The new min,max values are interpreted as real numbers.

FILES:

CM - communications file
DI - display information file
DV - display vector file
PV - projection vector file
S1 - scratch 1 file
Instrumentation file

REFERENCED BY:

SCALZM

SUBPROGRAMS REFERENCED:

CMGOTH	DIPHDI	DIPMAX	DVPHDR	EQUALA	ERASE	GIN
INSFLT	INSPGM	INSRET	LINSEG	MICMAC	PROMPT	TEXT
TRMGET	TRMPUT					

HISTORY:

designed by Mark Maginn November 19, 1980

programmed by Mark Maginn November 19, 1980

PROGRAM NAME: ZOM2SP

CATEGORY: UTILITY SUBROUTINE

PROGRAM DESCRIPTION:

ZOM2SP allows the user to select a subarea of the current two-space display and have a closer, more detailed examination of that area. In the manner described below, the user first selects the lower left-hand corner, then the upper right-hand corner of the area to be magnified. The new display is that subarea.

PROGRAM USAGE:

```
CALL ZOM2SP (FDCM,FDDI,FDDV,FDPV,MINMAX,SCREEN,DIHDR,
             DISCOD,NDIM,SCALTY,ZOMFLG)
```

INPUT ARGUMENTS:

FDCM - INTEGER; file descriptor of the CM file
FDDI - INTEGER; file descriptor of the DI file
FDDV - INTEGER; file descriptor of the DV file
FDPV - INTEGER; file descriptor of the PV file

MINMAX - REAL; array of current min - max values

SCREEN - INTEGER; array of screen coordinate values

DIHDR - INTEGER; array of info. from DI header

DISCOD - INTEGER; the display code

SCALTY - INTEGER; the type of scaling flag from the DI header

ZOMFLG - INTEGER; the zoom flag from the DI header

USER INTERACTION:

Using the graphics cursor, the user must select the portion of the current display upon which to "zoom". When the graphics cursor is on for the first time, the user positions it at the lower left-hand corner of the new display area and enters a any character on the keyboard. The second time, the user positions it at the upper right-hand corner of the new display area, and again enters any character.

ALGORITHM / NOTES:

Turn on the graphics cursor

Get lower left-hand position input by user (GIN).

Get upper right-hand position input by user

Check zoom flag to determine new value of zoom flag and type of scaling flag

Set maximum value used in computing new current points according to type of scaling flag.

Compute new current x(min), x(max), y(min), and y(max) and store them in DI file header (DIPMAX).

Write out new values to DI header

Turn the screen coordinate flag in the DV file header 'off' so that the display subroutine "knows" that it must recalculate a new set of screen coordinates for the one-space display.

Put up new two-space display (CLSCAT).

NOTE

For two-space projections, computation of the new max,min values from the screen coordinates given by the user, is done by solving for x and y in the equations for scatter screen coordinates (OLPARS Programmer and System Maintenance Manual). If (x(L),y(L)),(x(R),y(R)) are the screen coordinates of the two points entered by the user, this yields:

$$\text{new } x_{min} = \frac{\text{maxX}(x(L) - A - WC)}{WD - 2WC} + \text{xmin}$$

$$\text{new } x_{max} = \frac{\text{maxX}(x(R) - A - WC)}{WD - 2WC} + \text{xmin}$$

$$\text{new } y_{min} = \frac{\text{maxY}(y(L) - B - HC)}{HD - 2HC} + \text{ymin}$$

$$\text{new } y_{max} = \frac{\text{maxY}(y(R) - B - HC)}{HD - 2HC} + \text{ymin}$$

The new max,min values are interpreted as real numbers.

FILES:

CM - communications file
DI - display information file
DV - display vector file
PV - projection vector file
Instrumentation file

REFERENCED BY:

SCALZM

SUBPROGRAMS REFERENCED:

CLSCAT, CMGOTH, DIPHDI, DIPMAX, DVPHDR, EQUALA, ERASE, GIN, INSFLT, INSPGM, INSRET, PROMPT, RCTNGL, TEXT, TRMGET, TRMPUT

HISTORY:

designed by Mark Maginn October 7, 1980

programmed by Mark Maginn October 7, 1980

APPENDIX A

OLPARS SYSTEM DEPENDENT PROGRAMS

This section contains a list of the system dependent commands and subprograms.

EXECUTIVE PROGRAM

CIP

COMMANDS

ANYTHING
BYEOLP
FILEIN
FILEOUT
HELOLP
HELP
MEASXFRM

PROGRAMMER AIDS

GEN	ODTDMP
INSHSP	OLTDMP
MAKOPT	

MISCELLANEOUS TASKS

CMINIT (part of HELOLP process)

INSTRUMENTATION PACKAGE

GETHST	INSFLT
HSGHDR	INSINI
HSGREC	INSINT
HSPHDR	INSLOG
HSPLNK	INSPGM
HSPREC	INSRET
INSCHR	INSSET
INSDBL	PUTHST

GRAPHICS PACKAGE

ERASE	MOVE
GIN	RCTNGL
LINSEG	TEXT
MARK	

SUPPORT SUBPROGRAMS

ASCDEC	GETOPT	ORENAM
CHKEXS	GETSTR	PACKB
CLOSBL	GETVEC	PACKW
CLOSE	GOPTNM	PRINTR
CONCAT	GTLUN	PROMPT
DELETE	HASH	READB
EQUALA	INDEX	RENAME
EQUALS	INITD	SEARCH
EVALBM	ITREAL	SETOPT
FCGENT	LENGTH	SETUP
FCGHDR	LESS	TRMGET
FCPENT	MENU	TRMGLB
FCPHDR	MODINI	TRMOPN
FCREAT	NBLANK	TRMPUT
FCTOPN	OCLOSE	TYPE
FGET	OCREAT	USRIO1
FILGET	OEXIT	VALUEA
FILPUT	ODELET	VALUES
FLUSHB	OMOVE	VCREAT
FMNCOV	OOPEN	VOPEN
FOPEN	OPENBL	WRITEB
FPUT	OPENS	WRTNOW
GETFID	OPNTMP	

APPENDIX B

SYSTEM DEPENDENT PARAMETERS

1. Physical Record Length

   On the PDP-11/70 under RSX-11M, a physical record is 512 bytes or 256 words in length. This is the block size that FGET and FPUT work on.

2. Two-Space Screen Coordinates

   Ws = number of display units in screen width
   Hs = number of display units in screen height
   Wd = number of display units in display width
   Hd = number of display units in display height
   Wc = number of display units in character width
   Hc = number of display units in character height
   (a,b) = lowest left coordinates of display
   (c,d) = upper right coordinates of display
   M = number of rows in cluster plot
   N = number of columns in cluster plot
   L = number of characters per line

3. One-Space Screen Coordinates

   (e,f) = coordinates of the beginning of the first display line

   (g,f) = coordinates of the end of the first display line

   D = the length between successive display lines

   NC = the number of classes that can be displayed on the screen at once

==========
An example of a file containing the one-space and two-space coordinates can be seen in the OLPARS VI Programmers Reference Manual, Appendix B.
==========

APPENDIX C

NAMED COMMON BLOCKS

This appendix contains a list of all the named common blocks found in OLPARS. The name of the source file (include file) in which the common block is defined is found in parenthesis following the common block name. References to the OLPARS VI Programmers Reference Manual are abbreviated to PRM.

BLKBUF (fact.dcl)

Contains the physical record buffer space which is part of the File Access and Control Table (see PRM).

BUFCNT (fact.dcl)

Contains actual number of block buffers allocated at task build time.

CALUN (calun.dcl)

A list of currently available logical unit numbers.

COMSTK

Contains stack pointer used in the 'AFT' FORTRAN preprocessor's implementation of internal procedures.

FACT (fact.dcl)

The File Access and Control Table (see PRM)

FSHBLK (fshblk.dcl)

Contains Fisher logic information used during overall logic evaluation.

GP1BLK (gp1blk.dcl)

Contains one-space group logic information to be used during overall logic evaluation.

GP2BLK (gp2blk.dcl)

Contains two-space group logic information to be used during overall logic evaluation.

HISTRY (histry.dcl)

Instrumentation package common area.

LOGNOD (lognod.dcl)

Contains overall logic evaluation information.

NMVBLK (nmvblk.dcl)

Contains Nearest Mean Vector logic information to be used during an overall logic evaluation.

NNBLK (nnblk.dcl)

Contains Nearest Neighbor logic information to be used during overall logic evaluation.

ONEDT (dscrmthr.dcl)

Contains projection discriminant and threshold information for one-space logic creation.

TWODT (twodt.dcl)

Contains projection discriminant and threshold information for twospace logic creation.

APPENDIX D

OLPARS PROGRAM
CROSS REFERENCE TABLE

This appendix contains a cross reference table of all the currently implemented OLPARS programs found in this manual. The cross reference table is in alphabetical order with program names (subject program) on the left and "subprograms referenced" and "programs called by" lines on the right (a '+' symbol, preceding a particular line, indicates that the names found on that line are programs which reference the subject program).

OLPARS PROGRAM CROSS REFERENCE LISTING — D
(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

$ALOG +DISTOC

$DSQRT +CORMAT DCRIM2 NORMXFRM

$SQRT +L2FSHP MNSTDV MNVECD MNVRJT PROJCT S2FSHP

ADDMCV INSDBL INSFLT INSINT INSPGM INSRET TIGCOP TIGCOV TIGMN TIPCOP TIPCOV TIPMN
+ADUPCM MAKMRG NODCOM

ADUPCM ADDMCV INSINT INSPGM INSRET TIGCOP TIGCOV TIGHDR TIGMN
+APPEND CRANDTS DVEC MAKETREE MEASXFRM MOVEC NRSTNBR TRANSFRM

ALLVEC FILPUT INSINT INSPGM INSRET PACKB TRMPUT TVGVEC
+PRTDS

ALNCPN INSCHR INSINT INSPGM INSRET TIGCOP TIPCAP TIPCOP TIPNAM
+APPEND CLNODE

ANYTHING CMGDIR CONCAT ERASE FILGET INSINI INSRET MENU OEXIT OPENFX OPENS TRMPUT

APPEND ADUPCM ALNCPN CMGOTH COPYCV COPYMN COPYVC ERASE FIXXLV GNXTAN INSINI INSINT INSRET MENU
OEXIT OPENFX TIGCOP TRMPUT TVGHDR TVPHDR UIAPND

ASCDEC +GEN OBFGEN

BINDET +CNT1SP

BINWIDTH CLOSFX CMGOTH DIGHDI DIGMAX DIPHDI DIPMAX DVPHDR EQUALA ERASE INSFLT INSINI INSINT INSRET
MENU MICMAC OEXIT OPENFX PROMPT TRMGET TRMPUT

BNDEXP CLSCAT EQUALA ERASE FLUSHT INSPGM INSRET MICMAC TRMGET TRMPUT
+DRAWBNDY

BOUND +DRAWBNDY MOVEC OPTDSP

CDEFAULT CMGOTH CMPOTH ERASE INSINI INSINT INSRET MENU OEXIT OPENFX PROMPT TRMGET TRMPUT

CDISPLAY CLOSFX CLSCAT CMGCDS CMGOTH DIGDC DVPHDR INSINI INSRET MENU MICMAC OEXIT OPENFX TRMPUT

CDSCHK CMGCDS DIGNAM EQUALA INSPGM INSRET
+RANK REPROJECT SLCTMEAS TRANSFRM UNION

CHARFL INSPGM INSRET
+DBSUBSTR DRAWTREE MATXFRM MTRXDE MTRXDP MTRXEN REASNAME SAVMAT UICMND UIRSTR

CHKEXS INSPGM INSRET MENU OEXIT TRMPUT
+DRAWBNDY DVEC ESETUP FISHER FISHMOD L1ASDG L1EIGV L2ASDG L2EIGV L2FSHP MATXFRM MOVEC
+MMEVAL MMV MMVMOD NORMXFRM NRSTNBR OPTIMLMOD PMEVAL S2FSHP THRESHMOD UIAPND
+UICMND

CIP CLOSE CONCAT EQUALS EXIT FILGET FLUSHF GETMCR INDEX LENGTH OPENS TRMGET TRMPUT VALUES

CLFYMD CLOSE EQUALA ERASE EVALU8 FILPUT GETLST INSPGM INSRET LIGCLP LIGCOP LIGDCN LIGDSN OPENS

OLPARS PROGRAM CROSS REFERENCE LISTING — D
(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

PRINTR PROMPT TIGCOP TIGET TRMGET TRMPUT TVGHDR TVGVEC TVPHDR TVPLOG UILGE4 WRTNOW
+LOGEVAL

CLNODE ALNCPN INSPGM INSRET TIGCAP TIGCOP TIPCAP TIPCOP TIPNAM
+RESTRUCT

CLOS$ +FILEOUT HELP

CLOSBL CLOSE INSPGM INSRET OEXIT TRMPUT WRITEB
+FILEIN FNNCOV GETOPT GETVEC GOPTNM SETOPT

CLOSE +CIP CLFYMD CLOSBL CMINIT CMPMOD CMPRT CREVAL EIGPRT FILEIN GEN MAKMRG MTRXDP MTRXEN
+OBFGEN OCLOSE OEXIT PENMV PEPAIR PRINTR PRTDS PRTIDX PRTLOG

CLOSFX OCLOSE
+BINWIDTH CDISPLAY CMINIT CMPMOD CRANDTS CREATLOG CREATR CSCALE DBNDY
+DDATATREE DELETR DLOGTREE DRAWBNDY DTRENAME EIGNXFRM ESETUP FILEIN
+FISHSU GEN INSINI INTENSIFY L1ASDG L1CRDV L1EIGV L2ASDG L2CRDV L2EIGV LSETUP
+LTRENAME MAKETREE MATXFRM MEASXFRM MOVEC NAMELOG NMEVAL NMPASS NORMXFRM
+NRSTNBR OPENTR OPTDSP PEPAIR PROJMN PRTDS PRTIDX PWEVAL RDISPLAY REDRAW REPROJECT
+RESTRUCT S1CRDV S1EIGV S2CRDV S2EIGV SCALRET SCALZN SELECT SETLOG TRANSFRM UIAPND
+UICMND UICRLG UIL2FS UINMBR UIRSTR UIS2FS

CLOSTR INSPGM INSRET OCLOSE
+CMPMOD COMMOD CRANDTS DDSUBSTR DTRENAME EIGNXFRM EVALMN EXFRM FILEIN FILEOUT
+L1CRDV L1EIGV L2CRDV L2EIGV LISTLOGS LOGEVAL LTRENAME MAKCMB MAKMRG MEASXFRM
+MOVEC NAMELOG NMV NNMOD NORMXFRM NRSTNBR NXFRM OPTIMLMOD REPROJECT S1EIGV
+THRESHMOD TRANSFRM UICRLG UIL2FS

CLSCAT CLUS CMGCDS CMGOTH CMGSCN DIGHDI DIGMAX DIGNAM DIPHDI DISPCL DVPHDR EQUALA ERASE INSCHR
INSFLT INSINT INSPGM INSRET PVGHDR PVPHDR RCTNGL SCAT TEXT TRMPUT WRTNOW
+BNDEXP CDISPLAY CSCALE DBNDY L2ASDG L2CRDV L2EIGV L2FSHP MOVEC PRTIDX RDISPLAY
+REPROJECT S2CRDV S2EIGV S2FSHP SCALRET SELECT ZOM2SP

CLSTRX +CLUS

CLSTRY +CLUS

CLUS CLSTRX CLSTRY DIGENT DIGHDI DIGMAX DIPMAX DVGHDR DVGVEC DVPHDR DVPVEC GRIDIS INSFLT INSINT
INSPGM INSRET SCATR
+CLSCAT

CMDISP CMGSCN DIGCME DIGCMH ERASE INSINT INSPGM INSRET TRMPUT
+CMPMOD PENMV PEPAIR RDISPLAY

CMGCDS FGET INSCHR INSINT INSPGM INSRET OEXIT PACKW TRMPUT
+CDISPLAY CDSCHK CLSCAT CRANDTS CREATLOG CREVAL DDATAMOD DDATATREE
+DRAWBNDY DRAWTREE DSCRMEAS DVEC ESETUP FILEOUT FISHER FISHMOD INITD L1ASDG
+L1CRDV L1EIGV L2ASDG L2CRDV L2EIGV L2FSHP LSETUP MATXFRM MEASXFRM MENU MODINI MOVEC
+NAMELOG NMEVAL NMV NMVMOD NNMOD NORMXFRM NRSTNBR OPTIMLMOD PENMV PEPAIR PROJMN
+PRTCM PRTIDX PWEVAL RANK RDISPLAY REPROJECT S1CRDV S1EIGV S2CRDV S2EIGV S2FSHP
+SETUP SLCTMEAS THRESHMOD TRANSFRM UICRLG UIRSTR UNION WRTODS

CMGDIR FGET INSPGM INSRET PACKW

(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

```
                +ANYTHING       GETOPT  GOPTNM  HELP    SETOPT

CMGLOG           FGET    INSCHR  INSINT  INSPGM  INSRET  OEXIT   PACKW   TRMPUT
                +CREVAL  DLOGTREE        DLSUBSTR        DRAWLOG FISHER  FISHMOD LSETUP  MODINI  NMEVAL  NMV     NMVMOD
                +NRSTNBR OPTIMLMOD       PENMV   PEPAIR  PRTLOG  PWEVAL  REASNAME        REPROJECT       SETUP
                +THRESHMOD       UICRLG

CMGOPT           FGET    INSPGM  INSRET  OEXIT   PACKW   TRMPUT
                +MENU    UICRLG  UIRSTR  WRTODS

CMGOTH           FGET    OEXIT   TRMPUT
                +APPEND  BINWIDTH        CDEFAULT        CDISPLAY        CLSCAT  COMMOD  CRANDTS CREVAL  CSCALE  DBNDY
                +BDATAMOD        DDATATREE       DDSUBSTR        DLOGTREE        DLSUBSTR        DRAWBNDY        DRAWLOG
                +DRAWTREE        DSCRMEAS        DTRENAME        DVEC    EIGPRT  ELIMEA  ESETUP  FILEIN  FILEOUT FISHER
                +FISHMOD HELP    INSINI  INTENSIFY       L1CRDV  L1EIGV  L2CRDV  L2EIGV  LSETUP  LTRENAME        MACROP
                +MAKETREE        MATRIX  MATXFRM MEASXFRM        MICROP  MODINI  NAMELOG NMEVAL  NMV     NMVMOD  NNMOD
                +NORMXFRM        NRSTNBR OPTIMLMOD       PRTDS   PRTIDX  PRTLOG  PWEVAL  RANK    RDISPLAY
                +REASNAME        REPROJECT       S1CRDV  S1EIGV  S2CRDV  SCALRET SELECT  SETDS   SETLOG  SETUP
                +SLCTMEAS        SLTEIG  THRESHMOD       TRANSFRM        UICRLG  UIL2FS  UILASD  UIRSTR  UIS2FS  UNION
                +ZOM1SP  ZOM2SP

CMGSCN           FGET    INSPGM  INSRET  OEXIT   PACKW   TRMPUT
                +CLSCAT  CMDISP  CREVAL  DRAWBNDY        DRAWLOG DRAWTREE        LSETUP  MENU    MICMAC  MOVEC   MTRXDP
                +OPTDSP  PRECPT  PROJMN  PRTIDX  REASNAME        REDRAW  RODISP  SCALZM  THRDSP

CMINIT           CLOSE   CLOSFX  CMPDIR  CMPSCN  CONCAT  FILGET  FLUSHF  GETHCR  INSSET  LENGTH  OEXIT   OPENFX  OPENS
                 PROMPT  TRMGET  TRMPUT

CMNCOV           INSFLT  INSPGM  INSRET  TVGVEC  TVPVEC
                +RESTRUCT        TRANSFRM

CMPCDS           FPUT    INSPGM  INSRET  ITREAL
                +CRANDTS DDATATREE       SETDS

CMPDIR           FPUT    INSPGM  INSRET  ITREAL
                +CMINIT

CMPLOG           EQUALA  FPUT    INSPGM  INSRET  ITREAL
                +CREVAL  DLOGTREE        FISHER  NAMELOG NMV     NRSTNBR SETLOG

CMPMOD           CLOSE   CLOSFX  CLOSTR  CMDISP  CMPRT   DIPCME  DIPCMH  EQUALA  EVALUB  FILPUT  GNXTLN  INSPGM  INSRET
                 LIGCLP  LIGCNM  LIGCOP  LIGDCN  LIGDSN  OPENFX  OPENS   PRINTR  TIGCOP  TIGET   TIGWAM  TRMPUT  TVGHDR
                 TVGVEC  TVPHDR  TVPLOG  UILGE4  UILGE5  UILGE6
                +LOGEVAL

CMPOPT           FPUT    INSPGM  INSRET  ITREAL
                +SETOPT

CMPOTH           FPUT    INSPGM  INSRET
                +CDEFAULT

CMPRT            CLOSE   DIGCME  DIGCMH  FILPUT  INSPGM  INSRET  OPENS   PRINTR  TRMPUT
                +CMPMOD  PENMV   PEPAIR  PRTCN
```

OLPARS PROGRAM CROSS REFERENCE LISTING — D
(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

CMPSCN FPUT INSPGM INSRET ITREAL
+CMINIT

CNT1SP BINDET DIGENT DIPENT DVGVEC DVPVEC INSCHR INSINT INSPGM INSRET OEXIT S1GHDR S1PBC S1PHDR
SCATR TRMPUT
+MACROP MICROP

COLAPS INSPGM INSRET
+DCRIM1 EIGNPV NMVBSU TRANSFRM

COMBIN INSPGM INSRET
+DCRIM1

COMMOD CLOSTR CMGOTH COPYVC CREATR ERASE GARBND INSCHR INSINI INSINT INSLOG INSRET MENU NODCOM
OEXIT OPENFX RENAMT TIGCAP TIGCOP TIPCOP TIPNAM TRMPUT TVPHDR UICMND

CONCAT +ANYTHING CIP CMINIT GETHST GETOPT HELP MAKOPT OCREAT OOPEN OPENBL ORENAM

COPYCV INSPGM INSRET TIGCOV TIPCOV
+APPEND

COPYMN INSINT INSPGM INSRET TIGMN TIPMN
+APPEND

COPYVC INSINT INSLOG INSPGM INSRET TVGHDR TVGVEC TVPHDR TVPVEC
+APPEND COMMOD DDSUBSTR

CORMAT $DSQRT FILPUT INSINT INSPGM INSRET PACKB TIGCOV TRMPUT
+PRTDS

COVMAT FILPUT INSINT INSPGM INSRET PACKB TIGCOV TRMPUT
+PRTDS

CRANDTS ADUPCM CLOSFX CLOSTR CMGCDS CMGOTH CMPCDS CREATR EQUALA ERASE GMXTMD INSFLT INSINI INSINT
INSLOG INSRET LGLTRE MENU OEXIT OPENFX OPENTR PROMPT RAN RENAMT SUBMMC TIGCAP TIGCOP
TIGET TIGHDR TIPCAP TIPCOV TIPET TIPHDR TIPMN TIPNAM TLSRCH TRMGET TRMPUT TVGVEC TVPHDR
TVPVEC

CREAFX OCREAT
+GEN

CREATLOG CLOSFX CMGCDS CREVAL CRLOG1 CRLOG2 ERASE GTRGNI GTRGNR INSINI INSINT INSRET LIGCNM LIGCOP
OEXIT OPENS OPENTR TRMPUT UICRLG

CREATR CLOSFX INSPGM INSRET LLGENT LLGHDR LLPENT LLPHDR LLSRCH OCREAT OPENFX TLGENT TLGHDR TLPENT
TLPHDR TLSRCH TRMPUT
+COMMOD CRANDTS DDSUBSTR EXFRM FILEIN MAKETREE MEASXFRM NAMELOG NRSTNDR NXFRM
+TRANSFRM

CREVAL CLOSE CMGCDS CMGLOG CMGOTH CMGSCN CMPLOG EQUALA ERASE EV1SP EV2SP FILPUT INSCHR INSINT
INSPGM INSRET LIPCOP PRINTR PROMPT TIGCOP TIGNAM TRMGET TRMPUT TVGVEC TVPLOG WRTNOW
+CREATLOG

CRLOG1 CRLV1 EVALBM GNLIAE INSPGM INSRET LIGCNM LIGCOP LIGENT LIGSTR LIPENT

OLPARS PROGRAM CROSS REFERENCE LISTING -- D
(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

```
                +CREATLOG

CRLOG2           CRLV2   EVALBM  GNLIAE  INSPGM  INSRET  LIGCNM  LIGCOP  LIGENT  LIGSTR  LIPENT
                +CREATLOG

CRLV1            DIGHDI  DVGVEC  INSPGM  INSRET  PTRGNR  PVGENT
                +CRLOG1

CRLV2            DIGHDI  DSCVTH  DVGVEC  GNLVAE  INSPGM  INSRET  PTRGMI  PTRGNR  PVGENT
                +CRLOG2

CRPROJ           GLONOD  INSINT  INSPGM  INSRET  LCPROJ
                +S1CRDV  S2CRDV

CSCALE           CLOSFX  CLSCAT  CMGOTH  DIGHDI  DIGMAX  DIPHDI  DIPMAX  DVPHDR  EQUALA  ERASE   INSINI  INSRET  MENU
                 MICMAC  OEXIT   OPENFX  TRMGET  TRMPUT

DATE            +WRTNOW

DRNDY            CLOSFX  CLSCAT  CMGOTH  DIGHDI  DIPHDI  INSINI  INSINT  INSRET  MENU    MICMAC  OEXIT   OPENFX  TRMPUT

DCRIM            DCRIM1  DCRIM2  DITHER  INSFLT  INSINT  INSPGM  INSRET  INVERT  PVGENT
                +FISHSU  L1ASDG  L2ASDG  L2FSHP  S2FSHP

DCRIM1           COLAPS  COMBIN  INSINT  INSPGM  INSRET  LOGMNC  PVGENT  PVGHDR  TIGCOP  TIGCOV  TIGMN   TRMPUT
                +DCRIM

DCRIM2           $DSORT  IDX     INSFLT  INSPGM  INSRET  TRMPUT
                +DCRIM

DDATAMOD         CMGCDS  CMGOTH  ERASE   GLONOD  INSINI  INSRET  MENU    OEXIT   OPENFX  OPENTR  REARR   SELCLS  TIGET
                 TRMPUT  TVGHDR  TVPHDR

DDATATREE        CLOSFX  CMGCDS  CMGOTH  CMPCDS  DELETR  EQUALA  ERASE   INSINI  INSRET  MENU    OEXIT   OPENFX  PROMPT
                 TLGHDR  TRMGET  TRMPUT

DDSUBSTR         CHARFL  CLOSTR  CMGOTH  COPYVC  CREATR  EQUALA  ERASE   FIXXLV  GARBND  GLONOD  GNXTND  INSCHR  INSINI
                 INSINT  INSRET  LENGA   MENU    OEXIT   OPENFX  OPENTR  PROMPT  RENAMT  TIGCAP  TIGCOP  TIGET   TIGHDR
                 TIPCAP  TIPCOP  TIPNAM  TISRCH  TRMGET  TRMPUT  TVPHDR  UNIQND

DELETE          +ODELET

DELETR           CLOSFX  INSINT  INSLOG  INSPGM  INSRET  LLGENT  LLGHDR  LLPENT  LLPHDR  LLSRCH  ODELET  OPENFX  TLGENT
                 TLGHDR  TLPENT  TLPHDR  TLSRCH  TRMPUT
                +DDATATREE       DLOGTREE        NRSTNBR RENAMT

DIGCME           FGET    INSPGM  INSRET  OEXIT   TRMPUT
                +CMDISP  CMPRT

DIGCMH           FGET    INSPGM  INSRET  OEXIT   TRMPUT
                +CMDISP  CMPRT   SUMMCM

DIGDC            FGET    INSINT  INSPGM  INSRET  OEXIT   TRMPUT
                +CDISPLAY        DRAWBNDY        MOVEC   PROJMN  PRTCM   PRTIDX  RDISPLAY        REPROJECT       SUMMCM
```

OLPARS PROGRAM CROSS REFERENCE LISTING -- D
(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

```
                +UICRLG  UIRSTR

DIGEFS           FGET    INSCHR  INSINT  INSPGM  INSRET  OEXIT   TRMPUT
                +MODDFS

DIGENT           FGET    INSPGM  INSRET  OEXIT   PACKW   TRMPUT
                +CLUS    CNT1SP  DISPCL  MACROP  MEAN1   MEAN2   MICROP  MODDFS  MOVEC   OBTCLS  PRTIDX  SCAT

DIGHDI           FGET    INSPGM  INSRET  OEXIT   TRMPUT
                +BINWIDTH        CLSCAT  CLUS    CRLV1   CRLV2   CSCALE  DBNDY   DISPCL  INTENSIFY       MACROP  MEAN1
                +MEAN2   MICMAC  MICROP  MOVEC   OPTDSP  PRTIDX  REDRAW  RESTRUCT        SCALRET SCALZM  SCAT    SELECT
                +UICRLG  UIRSTR

DIGMAX           FGET    INSPGM  INSRET  OEXIT   TRMPUT
                +BINWIDTH        CLSCAT  CLUS    CSCALE  DRAWBNDY        MACROP  MICMAC  MICROP  MOVEC   OPTDSP  REDRAW
                +SCALRET SCALZM  SCAT

DIGNAM           FGET    INSPGM  INSRET  OEXIT   TRMPUT
                +CDSCHK  CLSCAT

DIGROE           FGET    INSPGM  INSRET  OEXIT   TRMPUT
                +RODISP  SLCTMEAS

DIGROH           FGET    INSPGM  INSRET  PACKW
                +RANK    RODCLS  RODISP  SLCTMEAS        TRANSFRM        UNION

DIGROI           FGET    INSPGM  INSRET  PACKW
                +RODISP  SLCTMEAS        TRANSFRM        UNION

DIPCME           FPUT    INSPGM  INSRET
                +CMPMOD  PENMV   PEPAIR

DIPCMH           FPUT    INSPGM  INSRET
                +CMPMOD  PENMV   PEPAIR

DIPEFS           FPUT    INSCHR  INSINT  INSPGM  INSRET
                +LNPROJ  MODDFS  OPTDSP

DIPENT           FPUT    INSPGM  INSRET  ITREAL
                +CNT1SP  DSPROJ  INTENSIFY       LCPROJ  LNPROJ  OPTDSP  SELECT

DIPHDI           FPUT    INSPGM  INSRET
                +BINWIDTH        CLSCAT  CSCALE  DBNDY   DRAWBNDY        DSPROJ  ESETUP  INITD   L1ASDG  L1CRDV  L1EIGV
                +L2ASDG  L2CRDV  L2EIGV  L2FSHP  LCPROJ  LNPROJ  MACROP  MICMAC  MICROP  OPTDSP  S1CRDV  S1EIGV  S2CRDV
                +S2EIGV  S2FSHP  SCALRET SLTEIG  ZOM1SP  ZOM2SP

DIPMAX           FPUT    INSPGM  INSRET  TRMPUT
                +BINWIDTH        CLUS    CSCALE  DSPROJ  LCPROJ  LNPROJ  OPTDSP  SCALRET SCAT    ZOM1SP  ZOM2SP

DIPNAM           FPUT    INSPGM  INSRET  OEXIT   TRMPUT
                +DSCRMEAS        INITD   L1ASDG  L1CRDV  L2ASDG  L2CRDV  L2EIGV  L2FSHP

DIPROE           FPUT    INSPGM  INSRET
                +DSCRMEAS        RANK    UNION
```

OLPARS PROGRAM CROSS REFERENCE LISTING — D
(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

DIPROH	FPUT	INSPGM	INSRET	ITREAL									
	+DSCRMEAS		RANK	UNION									
DIPROI	FPUT	INSPGM	INSRET	ITREAL									
	+DSCRMEAS		SLCTMEAS		UNION								
DISPCL	DIGENT	DIGHDI	INSPGM	INSRET	OEXIT	TRMPUT							
	+CLSCAT	MICMAC											
DISTEU	INSFLT	INSPGM	INSRET										
	+MOVEC	NMDIST	NNREV	NNPASS	UILASD								
DISTMA	INSFLT	INSPGM	INSRET										
	+NMDIST												
DISTOC	%ALOG	INSFLT	INSPGM	INSRET									
	+NMDIST												
DISTVA	INSFLT	INSPGM	INSRET										
	+NMDIST												
DITHER	INSFLT	INSINT	INSPGM	INSRET									
	+DCRIM	NMVDSU											
DLOGTREE	CLOSFX	CMGLOG	CMGOTH	CMPLOG	DELETR	EQUALA	ERASE	INSINI	INSRET	LLGHDR	MENU	OEXIT	OPENFX
	PROMPT	TRMGET	TRMPUT										
DLREGN	INSINT	INSPGM	INSRET	LVGHDR	LVGLNK	LVPHDR	LVPLNK	OEXIT	TRMPUT				
	+FISHTH	KILLND	OPTIMLMOD		THRDSP								
DLSUBSTR	CMGLOG	CMGOTH	EQUALA	ERASE	INSINI	INSRET	KILLND	LIGCOP	LIGDSN	MENU	OEXIT	OPENFX	OPENTR
	PROMPT	TRMGET	TRMPUT	TVGHDR									
DRAWBNDY	BNDEXP	BOUND	CHKEXS	CLOSFX	CMGCDS	CMGOTH	CMGSCN	DIGDC	DIGMAX	DIPHDI	DVGHDR	DVPVEC	EQUALA
	ERASE	GIN	INSINI	INSINT	INSLOG	INSRET	LINSEG	MARK	MENU	OEXIT	OPENFX	RCTNGL	TEXT
	TRMPUT												
DRAWLOG	CMGLOG	CMGOTH	CMGSCN	EQUALA	ERASE	GCLIST	GLSTRC	GOPTNM	INSFLT	INSINI	INSINT	INSRET	LIGCNM
	LIGCOP	LIGDCN	LIGDSN	LIGLOG	LIGSTR	LINSEG	MENU	MOVE	OEXIT	OPENFX	OPENTR	PROMPT	TRMGET
	TRMPUT	WRTNOW											
DRAWTREE	CHARFL	CMGCDS	CMGOTH	CMGSCN	EQUALA	ERASE	GDSTRC	INSFLT	INSINI	INSINT	INSRET	LENGA	LINSEG
	MENU	MOVE	OEXIT	OPENFX	OPENTR	PROMPT	TIGET	TIGHDR	TISRCH	TRMGET	TRMPUT	WRTNOW	
DSCRMEAS	CMGCDS	CMGOTH	DIPNAM	DIPROE	DIPROH	DIPROI	DVGROE	DVPROE	DVPROH	ERASE	GLONOD	INSCHR	INSFLT
	INSINI	INSINT	INSRET	MENU	OEXIT	OPENFY	OPENTR	PRECPT	RODISP	RSORTD	SETOPT	TIGCOP	TIGET
	TIGMN	TIGVAR	TRMPUT	TVGVEC	UIDCMS								
DSCVTH	INSFLT	INSPGM	INSRET										
	+CRLV2	OPTDSP	RESTRUCT										
DSPROJ	DIPENT	DIPHDI	DIPMAX	DVPHDR	DVPVEC	GLONOD	INSPGM	INSRET	PROJCT	PVGENT	PVGHDR	TIGCOP	TIGMN
	TVGVEC												

OLPARS PROGRAM CROSS REFERENCE LISTING -- D
(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

	+REPROJECT	S1EIGV	S2EIGV	S2FSHP									
DTRENAME	CLOSFX	CLOSTR	CNGOTH	EQUALA	ERASE	INSINI	INSRET	LGLTRE	MENU	OEXIT	OPENFX	OPENTR	PROMPT
	RENAMT	TLSRCH	TRMGET	TRMPUT									
DVEC	ADUPCM	CHKEXS	CNGCBS	CNGOTH	EQUALA	ERASE	GLOMOD	INSFLT	INSINI	INSINT	INSRET	MENU	OEXIT
	OPENFX	OPENTR	PROMPT	REARR	SBUPMC	SELCLS	TIGCOP	TIGCOV	TIGET	TIGMN	TIPCOP	TIPCOV	TIPMN
	TRMGET	TRMPUT	TVGHDR	TVGVEC	TVPHDR	TVPVEC							
DVGHDR	FGET	INSPGM	INSRET	OEXIT	TRMPUT								
	+CLUS	DRAWBNDY		MACROP	MICROP	MOVEC	SCAT						
DVGROE	FGET	INSPGM	INSRET	OEXIT	TRMPUT								
	+DSCRMEAS		RANK	UNION									
DVGVEC	FGET	INSPGM	INSRET	OEXIT	TRMPUT								
	+CLUS	CNT1SP	CRLV1	CRLV2	MEAN1	MEAN2	MOVEC	PRTIDX	REDRAW	RESTRUCT		SCAT	
DVPHDR	FPUT	INSPGM	INSRET										
	+BINWIDTH		CDISPLAY		CLSCAT	CLUS	CSCALE	DSPROJ	INITD	LCPROJ	LNPROJ	MACROP	MICMAC
	+MICROP	OPTDSP	SCALRET	SCAT	ZOM1SP	ZOM2SP							
DVPROE	FPUT	INSPGM	INSRET										
	+DSCRMEAS												
DVPROM	FPUT	INSPGM	INSRET										
	+DSCRMEAS												
DVPVEC	FPUT	INSPGM	INSRET	OEXIT	TRMPUT								
	+CLUS	CNT1SP	DRAWBNDY		DSPROJ	LCPROJ	LNPROJ	MOVEC	OPTDSP	SCAT			
EIGENJ	INSPGM	INSRET											
	+EIGMPV												
EIGLPV	INSPGM	INSRET	PVGENT	PVGHDR									
	+L1EIGV	L2EIGV											
EIGMPV	COLAPS	EIGENJ	INSPGM	INSRET	PVGENT	PVGHDR	PVPENT	SORT	TIGCOV				
	+EIGNXFRM		S1EIGV	S2EIGV									
EIGNXFRM	CLOSFX	CLOSTR	EIGMPV	EIGPRT	ERASE	ESETUP	EXFRM	GTHRSH	INSINI	INSRET	MENU	OEXIT	OPENFX
	PVGENT	SAVMAT	TRMPUT										
EIGPRT	CLOSE	CNGOTH	EQUALA	FILPUT	INSPGM	INSRET	OPENS	PRINTR	PROMPT	PVGENT	PVGHDR	TRMGET	TRMPUT
	+EIGNXFRM		L1EIGV	L2EIGV	REPROJECT		S1EIGV	S2EIGV					
ELIFRE	INSINT	INSPGM	INSRET	LIGCOP	LIPCOP	LIPLOG	OEXIT	TRMPUT					
	+KILLND												
ELIMEA	CNGOTH	EQUALA	INSPGM	INSRET	PROMPT	PVPENT	PVPHDR	TRMGET	TRMPUT				
	+FISHSU	L1EIGV	L2EIGV	S1EIGV	S2EIGV	UIL2FS	UILASD	UIS2FS					
EQUALA	INSCHR	INSPGM	INSRET	VALUEA									
	+BINWIDTH		BNDEXP	CDSCHK	CLFYMD	CLSCAT	CMPLOG	CMPMOD	CRAMDTS	CREVAL	CSCALE	DDATATREE	

OLPARS PROGRAM CROSS REFERENCE LISTING — D
(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED.

PROGRAM NAME

	+DDSUBSTR		DLOGTREE		DLSUBSTR		DRAWBNDY		DRAWLOG	DRAWTREE		DTRENAME	
	+DVEC	EIGPRT	ELIMEA	EVALNN	FILEOUT	FISHMOD	FXLTMP	GETPR	GETVEC	GTSLOT	INTENSIFY		
	+LTRENAME		MAKCMD	MAKETREE		MAKMRG	MATRIX	MATXFRM	MICMAC	MODDFS	MODINI	MOVEC	MTRXDE
	+MTRXDP	MTRXEN	NAMELOG	NMEVAL	NMVMOD	NMPASS	OBTCLS	ONEVEC	OPTDSP	OPTIMLMOD		PENMV	PEPAIR
	+PRECPT	PRTDS	PRTIDX	PRTLOG	PWEVAL	REASNAME		REPROJECT		RODCLS	RODISP	RSTORE	SAVMAT
	+SELCLS	SELECT	SETUP	THRDSP	THRESHMOD		TISRCH	UIAPND	UICMND	UICRLG	UIGNOR	UILASD	UILGE3
	+UILGE4	UILGE5	UILGE6	UINMBR	UIRSTR	UIXFRM	UNIGND	USRI01	ZOM1SP	ZOM2SP			
EQUALS	VALUES												
	+CIP	GEN	GETHST	HELP	INSHSP	MAKOPT	OBFGEN	SEARCH	SETOPT				
ERASE	+ANYTHING		APPEND	BINWIDTH		BNDEXP	CDEFAULT		CLFYMD	CLSCAT	CMDISP	COMMOD	CRANDTS
	+CREATLOG		CREVAL	CSCALE	DDATANOD		DDATATREE		DDSUBSTR		DLOGTREE		
	+DLSUBSTR		DRAWBNDY		DRAWLOG	DRAWTREE		DSCRMEAS		DTRENAME		DVEC	
	+EIGNXFRM		FILEIN	FILEOUT	FISHER	FISHMOD	GETPR	HELP	INTENSIFY		L1ASDG	L1CRDV	L1EIGV
	+L2ASDG	L2CRDV	L2EIGV	L2FSHP	LISTLOGS		LISTREES		LOGEVAL	LTRENAME		MAKETREE	
	+MATRIX	MATXFRM	MEASXFRM		MICMAC	MOVEC	MTRXDP	MTRXEN	NAMELOG	NMEVAL	NMV	NMVMOD	NMVRJT
	+NNMOD	NNPASS	NORMXFRM		NRSTNBR	OPTDSP	OPTIMLMOD		PRECPT	PROJMN	PRTCM	PRTDS	PRTIDX
	+PRTLOG	PWEVAL	RANK	RDISPLAY		REASNAME		REPROJECT		RESTRUCT		RODISP	S1CRDV
	+S1EIGV	S2CRDV	S2EIGV	S2FSHP	SELECT	SETDS	SETLOG	SLCTMEAS		SUMMCM	THRDSP	THRESHMOD	
	+TRANSFRM		UILASD	UILGE1	UILGE3	UIRSTR	UNION	ZOM1SP	ZOM2SP				
ERRSET	+FILEOUT	HELP	INSINI										
ERRTST	+FILEOUT												
ESETUP	CHKEXS	CLOSFX	CMGCDS	CMGOTH	DIPHDI	INSPGM	INSRET	OPENFX	OPENTR	PVPHDR	PVPNAM	TIGET	TRMPUT
	UIXFRM												
	+EIGNXFRM												
EV1SP	INSINT	INSPGM	INSRET										
	+CREVAL	EVALUB	RESTRUCT										
EV2SP	INSINT	INSPGM	INSRET										
	+CREVAL	EVALUB	PAIRLG	RESTRUCT									
EVALBM	INSPGM	INSRET											
	+CRLOG1	CRLOG2	FISHSL	GCLIST	NAMELOG	NMVBSU	NMLBSU	OLTDMP	PRTLOG	PRTSTR	UICRLG		
EVALNN	CLOSTR	EQUALA	GTRGNI	INSPGM	INSRET	LIGDCN	NNBREV	OPENTR	TIGCOP	TIGET	TIGHDR	TIGNAM	TRMPUT
	+EVALUB												
EVALUB	EV1SP	EV2SP	EVALNN	INSPGM	INSRET	LIGENT	NMDIST	PAIRLG	SU1SP	SU2SP	SUFISH	SUMMV	
	+CLFYMD	CMFMOD	RSTORE	SINGLE									
EXFRM	CLOSTR	CREATR	GNXTMD	INSFLT	INSINT	INSPGM	INSRET	NXFRM	OPENTR	TIGCAP	TIGCOV	TIGET	TIGMN
	TIPCAP	TIPCOP	TIPCOV	TIPET	TIPHDR	TIPMN	TIPNAM	TRMPUT	TVGVEC	TVPHDR	TVPVEC		
	+EIGNXFRM		MATXFRM										
EXIT	+CIP	OEXIT											
EXPAND	INSPGM	INSRET											
	+NMVBSU												

(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM_NAME

EXTIST INSFLT INSPGM INSRET RXINT RYINT
+REDRAW

EXTLST INSFLT INSPGM INSRET RXINT RYINT
+REDRAW

FCGENT FCGHDR FGET OEXIT TRMPUT
+GEN OCREAT ODELET OMOVE OOPEN ORENAM SEARCH

FCGHDR FGET OEXIT TRMPUT
+FCGENT GEN OCREAT ODELET SEARCH

FCPENT FPUT
+GEN OCREAT ODELET OMOVE ORENAM

FCPHDR FPUT
+GEN OCREAT ODELET

FCREAT +GEN OCREAT OPENBL OPNTMP

FCTOPN FOPEN GETFID OEXIT TRMPUT
+GEN OCREAT ODELET OMOVE OOPEN ORENAM

FGET OEXIT READB TRMPUT WRITEB
+CMGCDS CMGDIR CMGLOG CMGOPT CMGOTH CMGSCN DIGCME DIGCMH DIGDC DIGEFS DIGENT DIGHDI DIGMAX
+DIGNAM DIGROE DIGROH DIGROI DVGHDR DVGROE DVGVEC FCGENT FCGHDR GETOPT GOPTNM HSGHDR HSGREC
+LIGCAP LIGCLP LIGCNM LIGCOP LIGDCN LIGDSN LIGENT LIGLOG LIGPRB LIGSTR LLGENT LLGHDR LVGENT
+LVGHDR LVGLNK PVGENT PVGHDR PVGNAM S1GBC S1GHDR SETOPT SMGENT SMGHDR SMGLNK SMGNDS TIGCAP
+TIGCJP TIGCOV TIGET TIGHDR TIGNN TIGNAM TIGVAR TLGENT TLGHDR TMPXRD TVGHDR TVGLOG TVGVEC

FILEIN CLOSBL CLOSE CLOSFX CLOSTR CMGOTH CREATR ERASE FNNCOV GETVEC INSINI INSINT INSRET MENU
OEXIT OPENBL OPENFX TIPCAP TIPET TIPHDR TIPNAM TRMPUT TVGVEC TVPHDR TVPVEC USRIO1

FILEOUT CLOS$ CLOSTR CMGCDS CMGOTH EQUALA ERASE ERRSET ERRTST GNXTND GTLUN INSINI INSINT INSRET
MENU OEXIT OPEN$ OPENFX OPENTR PACKB PROMPT TIGET TIGHDR TRMGET TRMPUT TVGVEC

FILGET +ANYTHING CIP CMINIT GETVEC HELP INSHSP MAKOPT MTRXEN OBFGEN

FILPUT +ALLVEC CLFYMD CMPMOD CMPRT CORMAT COVMAT CREVAL EIGPRT GP1PRT GP2PRT INSCHR INSDBL INSFLT
+INSINT INSLOG INSPGM INSRET MAKETREE MAKMRG MNSTDV MNVECD MTRXDP MNVPRT MNBPRT ONEVEC
+PENMV PEPAIR PRTIDX PRTLOG PRTSTR PWSPRT RNGLAP TRESTR

FISH INSPGM INSRET PROJCT
+PAIRLG

FISHER CHKEXS CMGCDS CMGLOG CMGOTH CMPLOG ERASE FISHSL FISHSU FISHTH FISHVC INSINI INSRET KILLND
LIPCOP MENU OEXIT OPENFX SETUP TRMPUT

FISHMOD CHKEXS CMGCDS CMGLOG CMGOTH EQUALA ERASE FISHTH FISHVC GETLST GTRGNI INSINI INSRET KILLND
LIGCOP LIGDSN LIGSTR MENU OEXIT OPENFX OPENTR PROMPT PTRGNI SETOPT TRMGET TRMPUT TVGHDR

FISHSL EVALBM GNLIAE GTRGNI INSPGM INSRET LIGCLP LIGCNM LIGENT LIGSTR LIPENT PTRGNI
+FISHER

OLPARS PROGRAM CROSS REFERENCE LISTING -- D
(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

FISHSU
CLOSFX DCRIN ELIMEA GNLVAE INSPGM INSRET LIGCOP LOGMN LVPLNK OPENFX PROJCT PTRGNI PTRGNR
SETOPT TIGMN TRMPUT
+FISHER

FISHTH
DLREGN GTRGNI GTRGNR INSPGM INSRET PROMPT PTRGNR TRMGET TRMPUT
+FISHER FISHMOD

FISHVC
GTRGNI INSPGM INSRET PROMPT PTRGNI TRMGET TRMPUT
+FISHER FISHMOD

FIXXLV
INSINT INSPGM INSRET TIGCOP TIPCOP
+APPEND DDSUBSTR MODCOM REARR RESTRUCT

FLUSHB
OEXIT TRMPUT WRITEB
+GETHST PUTHST

FLUSHF
+CIP CMINIT HELP MAKOPT MTRXEN

FLUSHT
+BNDEXP GETPR INSHSP ODTDMP OLTDMP SETDS USRI01

FMNCOV
CLOSBL INSPGM INSRET OPENBL TIPCOV TIPMN TRMPUT TVGVEC TVPVEC
+FILEIN

FOPEN
+FCTOPN OOPEN OPENBL

FPUT
OEXIT READB TRMPUT WRITEB
+CMPCDS CMPDIR CMPLOG CMPOPT CMPOTH CMPSCN DIPCME DIPCMH DIPEFS DIPENT DIPHDI DIPMAX DIPNAM
+DIPROE DIPROH DIPROI DVPHDR DVPROE DVPROH DVPVEC FCPENT FCPHDR GEN HSPHDR HSPLNK HSPREC
+INSHSP LIPCAP LIPCLP LIPCMH LIPCOP LIPDCM LIPDSM LIPENT LIPLOG LIPPRB LIPSTR LLPENT LLPHDR
+LVPENT LVPHDR LVPLNK MAKOPT OBFGEN PUTHST PVPENT PVPHDR PVPNAM SIPBC SIPHDR SMPENT SMPHDR
+SMPLNK SMPMDS TIPCAP TIPCOP TIPCOV TIPET TIPHDR TIPMN TIPNAM TLPENT TLPHDR TMPXWT TVPHDR
+TVPLOG TVPVEC

FXLTMP
EQUALA GCLIST GNXTLN INSPGM INSRET TIGCOP TIGET TIGHDR TIGNAM TVGLOG TVPLOG
+KILLND

GARBKD
INSINT INSPGM INSRET TIGHDR TIPCOP TIPET TIPHDR
+COMMOD DDSUBSTR REARR

GCLIST
EVALBM INSINT INSPGM INSRET LIGCLP LIGCMM
+DRAWLOG FXLTMP GTSLOT MODINI REPROJECT SETUP UICRLG

GDSTRC
GNXTND INSINT INSPGM INSRET TIGCAP TIGNAM
+DRAWTREE

GEN
ASCDEC CLOSE CLOSFX CREAFX EQUALS FCGENT FCGHDR FCPENT FCPHDR FCREAT FCTOPN FPUT GETMCR
GETSTR HSPHDR HSPREC INDEX INSSET LENGTH NBLANK OCLOSE OEXIT SMPHDR TRMPUT

GETBUF
+OEXIT

GETFID
OEXIT TRMPUT
+FCTOPN OCREAT OOPEN OPENBL OPNTMP

GETHST
CONCAT EQUALS FLUSHB HASH HSGHDR HSGREC HSPHDR HSPLNK HSPREC OEXIT TRMPUT

(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

	+INSHSP	INSPGM											
GETLST	GNXTLN	INSINT	INSPGM	INSRET	LIGLOG	LIGSTR							
	+CLFYMD	FISHMOD	MODINI	NMEVAL	NMVMOD	OPTIMLMOD		PWEVAL	REASNAME		SETUP	THRESHMOD	
GETMCR	+CIP	CMINIT	GEN										
GETOPT	CLOSBL	CNGDIR	CONCAT	FGET	INSINT	INSPGM	INSRET	OEXIT	OPENBL	TRMPUT			
	+MENU												
GETPR	EQUALA	ERASE	FLUSHT	INSPGM	INSRET	TRMGET	TRMPUT						
	+OPTIMLMOD		THRESHMOD										
GETSTR	TRMGET												
	+GEN												
GETVEC	CLOSBL	EQUALA	FILGET	INSCHR	INSINT	INSPGM	INSRET	LGLMOD	OPENBL	TRMPUT	TVPVEC		
	+FILEIN												
GIN	+DRAWBNDY		MOVEC	OPTDSP	PRTIDX	THRDSP	ZOM2SP						
GLOMOD	INSPGM	INSRET	TIGCOP	TIGNAM									
	+CRPROJ	DDATAMOD		DDSUBSTR		DSCRMEAS		DSPROJ	DVEC	GTSLOT	MODINI	NAMELOG	RSTORE
	+SETUP	SINGLE	SLPAIR	UILASD	UIS2FS								
GLSTRC	GNXTLN	INSINT	INSPGM	INSRET	LIGLOG	LIGSTR	TRMPUT						
	+DRAWLOG												
GNLIAE	INSINT	INSPGM	INSRET	LIGCOP	LIGLOG	LIPCOP							
	+CRLOG1	CRLOG2	FISHSL	NMVBSU	NNLBSU								
GNLVAE	INSINT	INSPGM	INSRET	LVGHDR	LVGLNK	LVPHDR	LVPLNK						
	+CRLV2	FISHSU	NMVBSU	NNLBSU	PTRGNI	PTRGNR							
GNSMAE	INSINT	INSPGM	INSRET	SMGHDR	SMGLNK	SMPHDR	SMPLNK						
	+MTRXEN	SAVMAT											
GNXTAN	INSINT	INSPGM	INSRET	TIGCOP	TIGHDR	TIPET	TIPHDR						
	+APPEND	RESTRUCT											
GNXTLN	INSINT	INSLOG	INSPGM	INSRET	LIGSTR								
	+CMPMOD	FXLTMP	GETLST	GLSTRC	OLTDMP	PRTSTR							
GNXTND	INSCHR	INSINT	INSLOG	INSPGM	INSRET	TIGCAP	TIGNAM						
	+CRANDTS	DDSUBSTR		EXFRM	FILEOUT	GDSTRC	MEASXFRM		NXFRM	ODTDMP	PRTDS	REARR	TISRCH
	+TRANSFRM		UNIQND										
GOPTNM	CLOSBL	CNGDIR	FGET	INSINT	INSPGM	INSRET	OEXIT	OPENBL	TRMPUT				
	+DRAWLOG	OLTDMP	PRTLOG	PRTSTR	PWSPRT								
GP1PRT	FILPUT	INSFLT	INSINT	INSPGM	INSRET	PACKB	SU1SP						
	+PRTLOG												
GP2PRT	FILPUT	INSINT	INSPGM	INSRET	PACKB	SU2SP							

OLPARS PROGRAM CROSS REFERENCE LISTING — D
(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM_NAME

```
                +PRTLOG

GRIDIS           INSPGM  INSRET  MARK
                +CLUS

GTHRSH           INSFLT  INSPGM  INSRET  PROMPT  PVGENT  TRMGET  TRMPUT
                +EIGMXFRM

GTLUN           +FILEOUT HELP

GTRGNI           INSINT  INSPGM  INSRET  LVGENT
                +CREATLOG        EVALNM  FISHMOD FISHSL  FISHTH  FISHVC  KILLND  NMDIST  NMEVAL  NMVMOD  NMVOPT  NMVRJT
                +NMBPRT  NMMOD   OPTIMLMOD       PAIRLG  PENMV   PEPAIR  PWEVAL  PWSPRT  SU2SP   SUFISH  SUNMV
                +THRESHMOD

GTRGNR           INSINT  INSPGM  INSRET  LVGENT
                +CREATLOG        FISHTH  NMDIST  NMVPRT  NMVRJT  OPTDSP  OPTIMLMOD       PAIRLG  PENMV   PWSPRT  SU1SP
                +SU2SP   SUNMV   THRDSP

GTSLOT           EQUALA  GCLIST  GLOMOD  INSPGM  INSRET  TIGNAM
                +NMEVAL  NMVMOD  OPTIMLMOD       PWEVAL  THRESHMOD

HASH            +GETHST

HELP             CLOS$   CMGDIR  CMGOTH  CONCAT  EQUALS  ERASE   ERRSET  FILGET  FLUSHF  GTLUN   INDEX   INSSET  LENGTH
                 MENU    OEXIT   OPEN$   OPENFX  PROMPT  RELUN   TRMGET  TRMPUT  VALUES

HSGHDR           FGET    OEXIT   TRMPUT
                +GETHST

HSGREC           FGET    OEXIT   TRMPUT
                +GETHST

HSPHDR           FPUT
                +GEN     GETHST

HSPLNK           FPUT
                +GETHST

HSPREC           FPUT
                +GEN     GETHST

IDCHK            INSCHR  INSINT  INSLOG  INSPGM  INSRET  LENGA   TYPE
                +LGLNOD  LGLTRE  MTRXEN  SAVMAT

IDX              TRMPUT
                +DCRIM2  MOVEC

INDEX           +CIP     GEN     HELP    OBFGEN

INITD            CMGCDS  DIPHDI  DIPNAM  DVPHDR  INSPGM  INSRET  PVPHDR  PVPNAM  S1PHDR  SETOPT  TRMPUT
                +L1ASDG  L1CRDV  L1EIGV  L2ASDG  L2CRDV  L2EIGV  S1CRDV  S1EIGV  S2CRDV  S2EIGV  UIL2FS  UIS2FS

INSCHR           FILPUT  OEXIT   TRMPUT
```

QLPARS PROGRAM CROSS REFERENCE LISTING -- D
(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

	+ALNCPN	CLSCAT	CMGCDS	CMGLOG	CNT1SP	COMMOD	CREVAL	DDSUBSTR		DIGEFS	DIPEFS	DSCRMEAS	
	+EQUALA	GETVEC	GNXTND	IDCHK	LISTLOGS		LLSRCH	MAKCMB	MAKMRG	MAKOPT	MEASXFRM		MENU
	+NAMELOG	NNBREV	NNLBSU	NNMEAN	NNPASS	NNSORT	OPENBL	ORENAM	RENAMT	SELCLS	SETOPT	TISRCH	TLSRCH
	+TRANSFRM		TYPE	UICMND	UIXFRM	USRI01	WRTODS						
INSDBL	FILPUT	OEXIT	TRMPUT										
	+ADDMCV												
INSFLT	FILPUT	OEXIT	TRMPUT										
	+ADDMCV	BINWIDTH		CLSCAT	CLUS	CMMCOV	CRANDTS	DCRIM	DCRIM2	DISTEU	DISTMA	DISTQC	DISTVA
	+DITHER	DRAWLOG	DRAWTREE		DSCRMEAS		DSCVTH	DVEC	EXFRM	EXTIST	EXTLST	GP1PRT	GTHRSH
	+LOGMN	MATXFRM	MEASXFRM		NAMELOG	NNMNCV	OPTDSP	REDRAW	RNGLAP	RSORTA	RSORTD	RXINT	RYINT
	+SCALZM	SCAT	SCATR	SLCTMEAS		SU1SP	SUBMNC	TRANSFRM		UILASD	UILGE3	ZOM1SP	ZOM2SP
INSHSP	EQUALS	FILGET	FLUSHT	FPUT	GETHST	INSSET	LENGTH	OEXIT	OPENS	TRMGET	TRMPUT		
INSINI	CLOSFX	CMGOTH	ERRSET	INSPGM	OEXIT	OPENFX	OPENS	TRMPUT					
	+ANYTHING		APPEND	BINWIDTH		CDEFAULT		CDISPLAY		COMMOD	CRANDTS	CREATLOG	
	+CSCALE	DBNDY	DDATAMOD		DDATATREE		DDSUBSTR		DLOGTREE		DLSUBSTR		
	+DRAWBNDY		DRAWLOG	DRAWTREE		DSCRMEAS		DTRENAME		DVEC	EIGNXFRM		FILEIN
	+FILEOUT	FISHER	FISHMOD	INTENSIFY		L1ASDG	L1CRDV	L1EIGV	L2ASDG	L2CRDV	L2EIGV	L2FSHP	
	+LISTLOGS		LISTREES		LOGEVAL	LTRENAME		MAKETREE		MAKOPT	MATRIX	MATXFRM	
	+MEASXFRM		MOVEC	NAMELOG	NMEVAL	NMV	NMVMOD	NNMOD	NORMXFRM		NRSTNBR	OPTIMLMOD	
	+PROJMN	PRTCM	PRTDS	PRTIDX	PRTLOG	PWEVAL	RANK	RDISPLAY		REASNAME		REDRAW	
	+REPROJECT		RESTRUCT		S1CRDV	S1EIGV	S2CRDV	S2EIGV	S2FSHP	SCALRET	SCALZM	SELECT	SETDS
	+SETLOG	SLCTMEAS		SUMMCM	THRESHMOD		TRANSFRM		UNION				
INSINT	FILPUT	OEXIT	TRMPUT										
	+ADDMCV	ADUPCM	ALLVEC	ALNCPN	APPEND	BINWIDTH		CDEFAULT		CLSCAT	CLUS	CMDISP	CMGCDS
	+CMGLOG	CNT1SP	COMMOD	COPYMN	COPYVC	CORMAT	COVMAT	CRANDTS	CREATLOG		CREVAL	CRPROJ	DBNDY
	+DCRIM	DCRIM1	DDSUBSTR		DELETR	DIGDC	DIGEFS	DIPEFS	DITHER	DLREGN	DRAWBNDY		DRAWLOG
	+DRAWTREE		DSCRMEAS		DVEC	ELIFRE	EV1SP	EV2SP	EXFRM	FILEIN	FILEOUT	FIXXLV	GARBND
	+GCLIST	GDSTRC	GETLST	GETOPT	GETVEC	GLSTRC	GMLIAE	GNLVAE	GNSMAE	GNXTAN	GNXTLN	GNXTND	GOPTMM
	+GP1PRT	GP2PRT	GTRGNI	GTRGNR	IDCHK	INTENSIFY		INVERT	KILLWD	L1ASDG	L2ASDG	L2FSHP	LCPROJ
	+LIGCLP	LIGCOP	LIGLOG	LIPCLP	LIPLOG	LLSRCH	LNPROJ	LOGMN	LOGMNC	LSETUP	LVGENT	LVGLNK	LVPENT
	+LVPLNK	MACROP	MAKCMB	MAKMRG	MAKOPT	MATXFRM	MEAN1	MEAN2	MEASXFRM		MENU	MICMAC	MICROP
	+MNSTDV	MNVECD	MOVEC	NAMELOG	NMVPRT	NNBREV	NNLBSU	NNMEAN	NNMNCV	NNPASS	NNSORT	NODCOM	
	+NORMXFRM		NXFRM	OBTCLS	ODELET	OMOVE	ONEVEC	OPENBL	OPTDSP	ORENAM	PROJMN	PRTDS	PRTIDX
	+PRTLOG	PTRGNI	PTRGNR	PVGHDR	PWSPRT	RANK	REDRAW	RENAMT	REPROJECT		RESTRUCT		RNGLAP
	+RODISP	RXINT	RYINT	S2FSHP	SAVMAT	SBUPMC	SCALRET	SCALZM	SCAT	SCATR	SELECT	SETLOG	SETOPT
	+SLCTMEAS		SLPAIR	SU1SP	SU2SP	SUBMNC	TISRCH	TLSRCH	TMPXRD	TMPXWT	TRANSFRM		TVGLOG
	+TVPLOG	UIAPND	UICMND	UICRLG	UIL2FS	UILASD	UILGE3	UIRSTR	UIS2FS	UNION	UNIQND	USRI01	ZOM1SP
INSLOG	FILPUT	OEXIT	TRMPUT										
	+COMMOD	COPYVC	CRANDTS	DELETR	DRAWBNDY		GNXTLN	GNXTND	IDCHK	INTENSIFY		L2FSHP	LESS
	+LGLMOD	LGLTRE	LLSRCH	MODCOM	OMOVE	PRTDS	PTRGNI	PTRGNR	RENAMT	SELCLS	SELECT	TISRCH	TLSRCH
	+UICMND	UIL2FS	UNIQND	USRI01									
INSPGM	FILPUT	GETHST	LENGTH	OEXIT	TRMPUT								
	+ADDMCV	ADUPCM	ALLVEC	ALNCPN	BNDEXP	CDSCHK	CHARFL	CHKEXS	CLFYMD	CLNODE	CLOSBL	CLOSTR	CLSCAT
	+CLUS	CMDISP	CMGCDS	CMGDIR	CMGLOG	CMGOPT	CMGSCN	CMMCOV	CMPCDS	CMPDIR	CMPLOG	CMPMOD	CMPOPT
	+CMPOTH	CMPRT	CMPSCN	CNT1SP	COLAPS	COMBIN	COPYCV	COPYMN	COPYVC	CORMAT	COVMAT	CREATR	CREVAL
	+CRLOG1	CRLOG2	CRLV1	CRLV2	CRPROJ	DCRIM	DCRIM1	DCRIM2	DELETR	DIGCME	DIGCMN	DIGDC	DIGEFS
	+DIGENT	DIGHDI	DIGMAX	DIGNAM	DIGROE	DIGROH	DIGROI	DIPCME	DIPCMN	DIPEFS	DIPENT	DIPHDI	DIPMAX

OLPARS PROGRAM CROSS REFERENCE LISTING -- D
(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

+DIPNAM DIPROE DIPROH DIPROI DISPCL DISTEU DISTMA DISTQC DISTVA DITHER DLREGN DSCVTH DSPROJ
+DVGHDR DVGROE DVGVEC DVPHDR DVPROE DVPROH DVPVEC EIGENJ EIGLPV EIGNPV EIGPRT ELIFRE ELIMEA
+EQUALA ESETUP EV1SP EV2SP EVALBM EVALNM EVALU8 EXFRM EXPAND EXT1ST EXTLST FISH FISHSL
+FISHSU FISHTH FISHVC FIXKLV FMNCOV FXLTMP GARBND GCLIST GDSTRC GETLST GETOPT GETPR GETVEC
+GLONOD GLSTRC GNLIAE GNLVAE GNSMAE GNXTAN GNXTLN GNXTND GOPTNM GP1PRT GP2PRT GRIDIS GTHRSH
+GTRGNI GTRGNR GTSLOT IDCHK INITD INSINI INVERT KILLND LCPROJ LESS LGLNOD LGLTRE LIGCAP
+LIGCLP LIGCNM LIGCOP LIGDCN LIGDSN LIGENT LIGLOG LIGPRB LIGSTR LIPCAP LIPCLP LIPCNM LIPCOP
+LIPDCN LIPDSN LIPENT LIPLOG LIPPRB LIPSTR LLGENT LLGHDR LLPENT LLPHDR LLSRCH LNPROJ LOGMN
+LOGMNC LSETUP LVGENT LVGHDR LVGLNK LVPENT LVPHDR LVPLNK MACROP MAKCMB MAKMRG MEAN1 MEAN2
+MENU MICMAC MICROP MNSTDV MNVECD MODDFS MODINI MTRXDE MTRXDP MTRXEN MTRXLI MXFRM NMDIST
+NMVBSU NMVOPT NMVPRT NMVRJT NNBPRT NNBREV NNLBSU NNMEAN NNMNCV NNPASS NNSORT NODCOM NXFRM
+OBTCLS ODELET OMOVE ONEVEC OPENBL OPENTR OPTDSP ORENAM PAIRLG PENMV PEPAIR PRECPT PRINTR
+PROJCT PROMPT PRTSTR PSORT PTRGNI PTRGNR PVGENT PVGHDR PVGNAM PVPENT PVPHDR PVPNAM PWSPRT
+RCTNGL REARR REDRAW REDVEC RENAMT RNGLAP RODCLS RODISP RSORTA RSORTD RSTORE RXINT RYINT
+S1GBC S1GHDR S1PBC S1PHDR SAVMAT SBUPMC SCAT SCATR SELCLS SETOPT SETUP SINGLE SLPAIR
+SLTEIG SMGENT SMGHDR SMGLNK SMGMDS SMPENT SMPHDR SMPLNK SMPMDS SORT SU1SP SU2SP SUBMNC
+SUFISH SUNMV THRDSP TIGCAP TIGCOP TIGCOV TIGET TIGHDR TIGMN TIGNAM TIGVAR TIPCAP TIPCOP
+TIPCOV TIPET TIPHDR TIPMN TIPNAM TISRCH TLGENT TLGHDR TLPENT TLPHDR TLSRCH TMPXRD TMPXWT
+TRESTR TVGHDR TVGLOG TVGVEC TVPHDR TVPLOG TVPVEC TYPE UIAPND UICMND UICRLG UIDCMS UIGNOR
+UILIFS UILASD UILGE1 UILGE2 UILGE3 UILGE4 UILGE5 UILGE6 UINNBR UIRSTR UIS2FS UIXFRM UNIQND
+USRI01 WRTNOW WRTODS ZOM1SP ZOM2SP

INSRET FILPUT OEXIT PUTHST TRMPUT
+ADDMCV ADUPCM ALLVEC ALMCPM ANYTHING APPEND BINWIDTH BNDEXP CDEFAULT
+CDISPLAY CDSCHK CHARFL CHKEXS CLFYMD CLNODE CLOSBL CLOSTR CLSCAT CLUS CMDISP CMGCDS
+CMGDIR CMGLOG CMGOPT CMGSCN CMNCOV CMPCDS CMPDIR CMPLOG CMPMOD CMPOPT CMPOTH CMPRT CMPSCN
+CNT1SP COLAPS COMBIN COMNOD COPYCV COPYMM COPYVC CORMAT COVMAT CRANDTS CREATLOG CREATR
+CREVAL CRLOG1 CRLOG2 CRLV1 CRLV2 CRPROJ CSCALE DBNDY DCRIM DCRIM1 DCRIM2 DDATANOD
+DDATATREE DDSUBSTR DELETR DIGCME DIGCMH DIGDC DIGEFS DIGENT DIGHDI DIGMAX DIGNAM
+DIGROE DIGROH DIGROI DIPCME DIPCMH DIPEFS DIPENT DIPHDI DIPMAX DIPNAM DIPROE DIPROH DIPROI
+DISPCL DISTEU DISTMA DISTQC DISTVA DITHER DLOGTREE DLREGN DLSUBSTR DRAWBNDY
+DRAWLOG DRAWTREE DSCRMEAS DSCVTH DSPROJ DTRENAME DVEC DVGHDR DVGROE DVGVEC
+DVPHDR DVPROE DVPROH DVPVEC EIGENJ EIGLPV EIGNPV EIGNXFRM EIGPRT ELIFRE ELIMEA EQUALA
+ESETUP EV1SP EV2SP EVALBM EVALNM EVALU8 EXFRM EXPAND EXT1ST EXTLST FILEIN FILEOUT FISH
+FISHER FISHMOD FISHSL FISHSU FISHTH FISHVC FIXKLV FMNCOV FXLTMP GARBND GCLIST GDSTRC GETLST
+GETOPT GETPR GETVEC GLONOD GLSTRC GNLIAE GNLVAE GNSMAE GNXTAN GNXTLN GNXTND GOPTNM GP1PRT
+GP2PRT GRIDIS GTHRSH GTRGNI GTRGNR GTSLOT IDCHK INITD INTENSIFY INVERT KILLND L1ASDG
+L1CRDV L1EIGV L2ASDG L2CRDV L2EIGV L2FSHP LCPROJ LESS LGLNOD LGLTRE LIGCAP LIGCLP LIGCNM
+LIGCOP LIGDCN LIGDSN LIGENT LIGLOG LIGPRB LIGSTR LIPCAP LIPCLP LIPCNM LIPCOP LIPDCN LIPDSN
+LIPENT LIPLOG LIPPRB LIPSTR LISTLOGS LISTREES LLGENT LLGHDR LLPENT LLPHDR LLSRCH
+LNPROJ LOGEVAL LOGMN LOGMNC LSETUP LTRENAME LVGENT LVGHDR LVGLNK LVPENT LVPHDR LVPLNK
+MACROP MAKCMB MAKETREE MAKMRG MAKOPT MATRIX MATXFRM MEAN1 MEAN2 MEASXFRM MENU
+MICMAC MICROP MNSTDV MNVECD MODDFS MODINI MOVEC MTRXDE MTRXDP MTRXEN MTRXLI MXFRM NAMELOG
+NMDIST NMEVAL NMV NMVBSU NMVMOD NMVOPT NMVPRT NMVRJT NNBPRT NNBREV NNLBSU NNMEAN NNMNCV
+NNMOD NNPASS NNSORT NODCOM NORMXFRM NRSTNBR NXFRM OBTCLS ODELET OMOVE ONEVEC OPENBL
+OPENTR OPTDSP OPTIMLMOD ORENAM PAIRLG PENMV PEPAIR PRECPT PRINTR PROJCT PROJMN PROMPT
+PRTCM PRTDS PRTIDX PRTLOG PRTSTR PSORT PTRGNI PTRGNR PVGENT PVGHDR PVGNAM PVPENT PVPHDR
+PVPNAM PWEVAL PWSPRT RANK RCTNGL RDISPLAY REARR RELSNAME REDRAW REDVEC RENAMT
+REPROJECT RESTRUCT RNGLAP RODCLS RODISP RSORTA RSORTD RSTORE RXINT RYINT S1CRDV
+S1EIGV S1GBC S1GHDR S1PBC S1PHDR S2CRDV S2EIGV S2FSHP SAVMAT SBUPMC SCALRET SCALZM SCAT
+SCATR SELCLS SELECT SETDS SETLOG SETOPT SETUP SINGLE SLCTMEAS SLPAIR SLTEIG SMGENT
+SMGHDR SMGLNK SMGMDS SMPENT SMPHDR SMPLNK SMPMDS SORT SU1SP SU2SP SUBMNC SUFISH SUNMCM
+SUNMV THRDSP THRESHMOD TIGCAP TIGCOP TIGCOV TIGET TIGHDR TIGMN TIGNAM TIGVAR TIPCAP
+TIPCOP TIPCOV TIPET TIPHDR TIPMN TIPNAM TISRCH TLGENT TLGHDR TLPENT TLPHDR TLSRCH TMPXRD

(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME													
	+TMPXWT	TRANSFRM		TRESTR	TVGHDR	TVGLOG	TVGVEC	TVPHDR	TVPLOG	TVPVEC	TYPE	UIAPND	UICMND
	+UICRLG	UIDCMS	UIGNOR	UIL2FS	UILASD	UILGE1	UILGE2	UILGE3	UILGE4	UILGE5	UILGE6	UINMBR	UIRSTR
	+UIS2FS	UIXFRM	UNION	UNIQND	USRI01	WRTMOW	WRTODS	ZOM1SP	ZOM2SP				
INSSET	OEXIT	OPENFX	TRMPUT										
	+CMINIT	GEN	HELP	INSHSP	OBFGEN	ODTDMP	OLTDMP						
INTENSIFY	CLOSFX	CMGOTH	DIGHDI	DIPENT	EQUALA	ERASE	INSINI	INSINT	INSLOG	INSRET	MENU	MICMAC	OBTCLS
	OEXIT	OPENFX	TRMPUT										
INVERT	INSINT	INSPGM	INSRET										
	+DCRIM	NMVBSU											
ITREAL	+CMPCDS	CMPDIR	CMPLOG	CMPOPT	CMPSCN	DIPENT	DIPROH	DIPROI	LIPDCN	LIPENT	LIPSTR	LLPENT	PVPNAM
	+SMPMDS	TLPENT											
KILLND	DLREGN	ELIFRE	FXLTMP	GTRGNI	INSINT	INSPGM	INSRET	LIGCOP	LIGLOG	LIGSTR	LIPLOG	LIPSTR	
	+DLSUBSTR		FISHER	FISHMOD	NMV	NMVMOD	NRSTNBR	OPTIMLMOD		THRESHMOD			
L1ASDG	CHKEXS	CLOSFX	CMGCDS	DCRIM	DIPHDI	DIPNAM	ERASE	INITD	INSINI	INSINT	INSRET	LNPROJ	MENU
	MICMAC	MODDFS	OEXIT	OPENFX	PVPENT	PVPHDR	SETUP	TRMPUT	UILASD				
L1CRDV	CLOSFX	CLOSTR	CMGCDS	CMGOTH	DIPHDI	DIPNAM	ERASE	INITD	INSINI	INSRET	LCPROJ	MENU	MICMAC
	OEXIT	OPENFX	PROMPT	PVPENT	PVPHDR	SETUP	TRMGET	TRMPUT					
L1EIGV	CHKEXS	CLOSFX	CLOSTR	CMGCDS	CMGOTH	DIPHDI	EIGLPV	EIGPRT	ELIMEA	ERASE	INITD	INSINI	INSRET
	LNPROJ	MENU	MICMAC	OEXIT	OPENFX	OPENTR	SETUP	SLTEIG	TRMPUT				
L2ASDG	CHKEXS	CLOSFX	CLSCAT	CMGCDS	DCRIM	DIPHDI	DIPNAM	ERASE	INITD	INSINI	INSINT	INSRET	LNPROJ
	MENU	MODDFS	OEXIT	OPENFX	PVPENT	PVPHDR	SETUP	TRMPUT	UILASD				
L2CRDV	CLOSFX	CLOSTR	CLSCAT	CMGCDS	CMGOTH	DIPHDI	DIPNAM	ERASE	INITD	INSINI	INSRET	LCPROJ	MENU
	OEXIT	OPENFX	OPENTR	PROMPT	PVPENT	PVPHDR	SETUP	TIGCOP	TRMGET	TRMPUT			
L2EIGV	CHKEXS	CLOSFX	CLOSTR	CLSCAT	CMGCDS	CMGOTH	DIPHDI	DIPNAM	EIGLPV	EIGPRT	ELIMEA	ERASE	INITD
	INSINI	INSRET	LNPROJ	MENU	OEXIT	OPENFX	OPENTR	SETUP	SLTEIG	TIGCOP	TRMPUT		
L2FSHP	$SQRT	CHKEXS	CLSCAT	CMGCDS	DCRIM	DIPHDI	DIPNAM	ERASE	INSINI	INSINT	INSLOG	INSRET	LNPROJ
	MENU	MODDFS	OEXIT	OPENFX	PVPENT	PVPHDR	TRMPUT	UIL2FS					
LCPROJ	DIPENT	DIPHDI	DIPMAX	DVPHDR	DVPVEC	INSINT	INSPGM	INSRET	TIGCOP	TIGHDR	TIGMW	TIGNAM	TRMPUT
	TVGVEC												
	+CRPROJ	L1CRDV	L2CRDV										
LENGA	+DDSUBSTR		DRAWTREE		IDCHK	MATXFRM	MTRXDE	MTRXDP	MTRXEN	OBFGEN	OBTCLS	ONEVEC	PRTDS
	+REASNAME		SAVMAT	SETDS	UIAPND	UICMND	UIRSTR						
LENGTH	+CIP	CMINIT	GEN	HELP	INSHSP	INSPGM	LESS	OBFGEN					
LESS	INSLOG	INSPGM	INSRET	LENGTH	VALUES								
	+MAKOPT												
LGLMOD	IDCHK	INSLOG	INSPGM	INSRET									
	+GETVEC	REASNAME		UIAPND	UICMND	UIRSTR							

OLPARS PROGRAM CROSS REFERENCE LISTING — D
(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM_NAME

LGLTRE
IDCHK INSLOG INSPGM INSRET
+CRANDTS DTRENAME LTRENAME MAKETREE NAMELOG SETLOG UINMBR UIXFRM USRIO1

LIGCAP
FGET INSPGM INSRET OEXIT TRMPUT
+OLTDMP PRTLOG

LIGCLP
FGET INSINT INSPGM INSRET OEXIT TRMPUT
+CLFYMD CMPMOD FISHSL GCLIST NMVBSU NMLBSU PRTLOG PRTSTR UICRLG

LIGCNM
FGET INSPGM INSRET OEXIT TRMPUT
+CMPMOD CREATLOG CRLOG1 CRLOG2 DRAWLOG FISHSL GCLIST NMVBSU NMLBSU OLTDMP PRTLOG

LIGCOP
FGET INSINT INSPGM INSRET OEXIT TRMPUT
+CLFYMD CMPMOD CREATLOG CRLOG1 CRLOG2 DLSUBSTR DRAWLOG ELIFRE FISHMOD FISHSU GNLIAE
+KILLND LISTLOGS LSETUP NMEVAL NMVBSU NMVMOD NMLBSU NNMOD OPTIMLMOD PWEVAL
+REPROJECT SETLOG SINGLE THRESHMOD UICRLG

LIGDCN
FGET INSPGM INSRET OEXIT PACKW TRMPUT
+CLFYMD CMPMOD DRAWLOG EVALNM PRTLOG REASNAME SINGLE

LIGDSN
FGET INSPGM INSRET OEXIT TRMPUT
+CLFYMD CMPMOD DLSUBSTR DRAWLOG FISHMOD MODINI NMEVAL NMVMOD OLTDMP OPTIMLMOD PRTLOG
+PWEVAL RSTORE SETUP THRESHMOD

LIGENT
FGET INSPGM INSRET OEXIT PACKW TRMPUT
+CRLOG1 CRLOG2 EVALU8 FISHSL NMVBSU NMLBSU OLTDMP

LIGLOG
FGET INSINT INSPGM INSRET OEXIT TRMPUT
+DRAWLOG GETLST GLSTRC GNLIAE KILLND LSETUP NRSTNBR OLTDMP OPTIMLMOD PENMV PEPAIR PRTLOG
+PRTSTR

LIGPRB
FGET INSPGM INSRET OEXIT TRMPUT
+LSETUP OLTDMP PAIRLG PENMV PRTLOG

LIGSTR
FGET INSPGM INSRET OEXIT PACKW TRMPUT
+CRLOG1 CRLOG2 DRAWLOG FISHMOD FISHSL GETLST GLSTRC GNXTLN KILLND NMEVAL NMVBSU NMVMOD NMLBSU
+NNMOD OPTIMLMOD PRTLOG PRTSTR PWEVAL THRESHMOD

LINSEG
+DRAWBNDY DRAWLOG DRAWTREE MACROP MICROP OPTDSP RCTNGL REDRAW THRDSP

LIPCAP
FPUT INSPGM INSRET
+NAMELOG

LIPCLP
FPUT INSINT INSPGM INSRET

LIPCNM
FPUT INSPGM INSRET
+NAMELOG

LIPCOP
FPUT INSPGM INSRET
+CREVAL ELIFRE FISHER GNLIAE NMV NRSTNBR REASNAME SETUP

LIPDCN
FPUT INSPGM INSRET ITREAL
+REASNAME

DLPARS PROGRAM CROSS REFERENCE LISTING — D
(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM_NAME

```
LIPDSN          FPUT    INSPGM  INSRET
                +NAMELOG

LIPENT          FPUT    INSPGM  INSRET  ITREAL
                +CRLOG1 CRLOG2  FISHSL  NAMELOG NMVBSU  NMLBSU

LIPLOG          FPUT    INSINT  INSPGM  INSRET
                +ELIFRE KILLND  OPTIMLMOD       SETUP

LIPPRB          FPUT    INSPGM  INSRET
                +NAMELOG

LIPSTR          FPUT    INSPGM  INSRET  ITREAL
                +KILLND

LISTLOGS        CLOSTR  ERASE   INSCHR  INSINI  INSRET  LIGCOP  LLGENT  LLGHDR  MENU    OEXIT   OOPEN   OPENFX  TRMPUT
                WRTNOW

LISTREES        ERASE   INSINI  INSRET  MENU    OEXIT   OPENFX  TLGENT  TLGHDR  TRMPUT  WRTNOW

LLGENT          FGET    INSPGM  INSRET  OEXIT   PACKW   TRMPUT
                +CREATR DELETR  LISTLOGS        LLSRCH  RENAMT

LLGHDR          FGET    INSPGM  INSRET  OEXIT   TRMPUT
                +CREATR DELETR  DLOGTREE        LISTLOGS        LLSRCH

LLPENT          FPUT    INSPGM  INSRET  ITREAL
                +CREATR DELETR  RENAMT

LLPHDR          FPUT    INSPGM  INSRET
                +CREATR DELETR

LLSRCH          INSCHR  INSINT  INSLOG  INSPGM  INSRET  LLGENT  LLGHDR  VALUEA
                +CREATR DELETR  LTRENAME        NAMELOG OPENTR  RENAMT  SETLOG

LNDCHK          +OLTDMP

LNPROJ          DIPEFS  DIPENT  DIPHDI  DIPMAX  DVPHDR  DVPVEC  INSINT  INSPGM  INSRET  PROJCT  PVGENT  PVGHDR  TIGCOP
                TIGMN   TIGNAM  TVGVEC
                +L1ASDG L1EIGV  L2ASDG  L2EIGV  L2FSHP  REPROJECT

LOGEVAL         CLFYMD  CLOSTR  CMPMOD  ERASE   INSINI  INSRET  LSETUP  MENU    OEXIT   OPENFX  RSTORE  SINGLE  TRMPUT

LOGMN           INSFLT  INSINT  INSPGM  INSRET  TIGCOP  TVGVEC
                +FISHSU UILASD

LOGMNC          INSINT  INSPGM  INSRET  TIGCOP  TVGVEC
                +DCRIM1 NMVBSU

LSETUP          CLOSFX  CMGCDS  CMGLOG  CMGOTH  CMGSCN  INSINT  INSPGM  INSRET  LIGCOP  LIGLOG  LIGPRB  OPENFX  OPENTR
                SETOPT  TRMPUT  UILGE1  UILGE2
                +LOGEVAL
```

OLPARS PROGRAM CROSS REFERENCE LISTING -- D
(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

LTRENAME
CLOSFX CLOSTR CMGOTH EQUALA ERASE INSINI INSRET LGLTRE LLSRCH MENU OEXIT OPENFX OPENTR
PROMPT RENAMT TRMGET TRMPUT

LVGENT
FGET INSINT INSPGM INSRET OEXIT TRMPUT
+GTRGNI GTRGNR

LVGHDR
FGET INSPGM INSRET OEXIT TRMPUT
+DLREGN GNLVAE

LVGLNK
FGET INSINT INSPGM INSRET OEXIT TRMPUT
+DLREGN GNLVAE PTRGNI PTRGNR

LVPENT
FPUT INSINT INSPGM INSRET
+PTRGNI PTRGNR

LVPHDR
FPUT INSPGM INSRET
+DLREGN GNLVAE MAKELOG

LVPLNK
FPUT INSINT INSPGM INSRET
+DLREGN FISHSU GNLVAE NMVBSU PTRGNI PTRGNR

MACROP
CMGOTH CNT1SP DIGENT DIGHDI DIGMAX DIPHDI DVGHDR DVPHDR INSINT INSPGM INSRET LINSEG S1GBC
TEXT TRMPUT
+MICMAC

MAKCMB
CLOSTR EQUALA INSCHR INSINT INSPGM INSRET OPENTR TIGCAP TIGCOP TIGCOV TIGET TIGHDR TIGMN
TIGNAM TIPCAP TIPCOP TIPCOV TIPET TIPHDR TIPMN TIPNAM TRMPUT TVGVEC TVPHDR TVPVEC
+MAKETREE

MAKETREE
ADUPCM CLOSFX CMGOTH CREATR EQUALA ERASE FILPUT INSINI INSRET LGLTRE MAKCMB MAKMRG MENU
OEXIT OPENFX OPENS PROMPT TLSRCH TRMGET TRMPUT

MAKMRG
ADDMCV CLOSE CLOSTR EQUALA FILPUT INSCHR INSINT INSPGM INSRET OPENTR PRINTR TIGCAP TIGCOP
TIGCOV TIGET TIGHDR TIGMN TIGNAM TIPCAP TIPCOP TIPCOV TIPET TIPHDR TIPMN TIPNAM TRMPUT
TVGVEC TVPHDR TVPVEC
+MAKETREE

MAKOPT
CONCAT EQUALS FILGET FLUSHF FPUT INSCHR INSINI INSINT INSRET LESS OEXIT OPENBL OPENS
PSORT TRMPUT

MARK
+DRAWBNDY GRIDIS MEAN2 MICROP MOVEC OPTDSP REDRAW SCAT

MATRIX
CMGOTH EQUALA ERASE INSINI INSRET MENU MTRXDE MTRXDP MTRXEN MTRXLI OEXIT OPENFX PROMPT
TRMGET TRMPUT

MATXFRM
CHARFL CHKEXS CLOSFX CMGCDS CMGOTH EQUALA ERASE EXFRM INSFLT INSINI INSINT INSRET LENGA
MENU NXFRM OEXIT OPENFX PROMPT SMGENT SMGMDS TRMGET TRMPUT UIXFRM

MEAN1
DIGENT DIGHDI DVGVEC INSINT INSPGM INSRET MOVE TRMPUT
+PROJMN

MEAN2
DIGENT DIGHDI DVGVEC INSINT INSPGM INSRET MARK RCTNGL
+PROJMN

(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME	
MEASXFRM	ADUPCM CLOSFX CLOSTR CMGCDS CMGOTH CREATR ERASE GNXTND INSCHR INSFLT INSINI INSINT INSRET MENU OEXIT OPENFX OPENTR TIGCAP TIGCOP TIGET TIPCAP TIPCOP TIPCOV TIPET TIPHDR TIPHN TIPNAM TRMPUT TVGVEC TVPHDR TVPVEC UIXFRM XFORM
MENU	CMGCDS CMGOPT CMGSCN GETOPT INSCHR INSINT INSPGM INSRET OEXIT SETOPT TEXT TRMPUT +ANYTHING APPEND BINWIDTH CDEFAULT CDISPLAY CHKEXS COMMOD CRANDTS CSCALE +DBNDY DDATAMOD DDATATREE DDSUBSTR DLOGTREE DLSUBSTR DRAWBNDY +DRAWLOG DRAWTREE DSCRMEAS DTRENAME DVEC EIGNXFRM FILEIN FILEOUT FISHER +FISHMOD HELP INTENSIFY L1ASDG L1CRDV L1EIGV L2ASDG L2CRDV L2EIGV L2FSHP LISTLOGS +LISTREES LOGEVAL LTRENAME MAKETREE MATRIX MATXFRM MEASXFRM MOVEC NAMELOG +NMEVAL NMV NMVMOD NNMOD NORMXFRM NRSTNBR OPTIMLMOD PROJNM PRTCM PRTDS PRTIDX +PRTLOG PWEVAL RANK RDISPLAY REASNAME REPROJECT RESTRUCT S1CRDV S1EIGV +S2CRDV S2EIGV S2FSHP SCALRET SCALZM SELECT SETDS SETLOG SLCTMEAS SUMMCM THRESHMOD +TRANSFRM UNION
MICMAC	CMGSCN DIGHDI DIGMAX DIPHDI DISPCL DVPHDR EQUALA ERASE INSINT INSPGM INSRET MACROP MICROP PROMPT PVPHDR TEXT TRMGET TRMPUT WRTNOW WRTODS +BINWIDTH BNDEXP CDISPLAY CSCALE DBNDY INTENSIFY L1ASDG L1CRDV L1EIGV PRTIDX +RDISPLAY REPROJECT S1CRDV S1EIGV SCALRET SELECT ZOM1SP
MICROP	CMGOTH CNT1SP DIGENT DIGHDI DIGMAX DIPHDI DVGHDR DVPHDR INSINT INSPGM INSRET LINSEG MARK S1GBC TEXT TRMPUT +MICMAC
MNSTDV	$SQRT FILPUT INSINT INSPGM INSRET PACKB TIGCOP TIGMN TIGVAR TRMPUT +PRTDS
MNVECD	$SQRT FILPUT INSINT INSPGM INSRET PACKB TIGMN TRMPUT +PRTDS
MODDFS	DIGEFS DIGENT DIPEFS EQUALA INSPGM INSRET OEXIT TIGNAM TRMPUT +L1ASDG L2ASDG L2FSHP S2FSHP
MODINI	CMGCDS CMGLOG CMGOTH EQUALA GCLIST GETLST GLONOD INSPGM INSRET LIGBSN OPENTR PROMPT TIGET TIGNAM TRMGET TRMPUT TVGHDR +NMMOD
MOVE	TEKTXT +DRAWLOG DRAWTREE MEAN1 MOVEC OPTDSP PRTIDX THRDSP
MOVEC	ADUPCM BOUND CHKEXS CLOSFX CLOSTR CLSCAT CMGCDS CMGSCN DIGDC DIGENT DIGHDI DIGMAX DISTEU DVGHDR DVGVEC DVPVEC EQUALA ERASE GIN IDX INSINI INSINT INSRET MARK MENU MOVE OEXIT OPENFX OPENTR RCTNGL SBUPMC SETOPT TIGCOP TIGCOV TIGET TIGHDR TIGMN TIPCOV TIPHN TRMPUT TVGVEC TVPVEC
MTRXDE	CHARFL EQUALA INSPGM INSRET LENGA PROMPT SMGHDR SMGLNK SMGNDS SMPHDR SMPLNK SMPNDS TRMGET TRMPUT +MATRIX
MTRXDP	CHARFL CLOSE CMGSCN EQUALA ERASE FILPUT INSPGM INSRET LENGA OPENS PRINTR PROMPT SMGENT SMGHDR SMGNDS TRMGET TRMPUT +MATRIX
MTRXEN	CHARFL CLOSE EQUALA ERASE FILGET FLUSHF GNSMAE IDCHK INSPGM INSRET LENGA OPENS PROMPT

OLPARS PROGRAM CROSS REFERENCE LISTING -- D
(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

SMGHDR SMGLNK SMGMDS SMPENT SMPHDR SMPLNK SMPMDS TRMGET TRMPUT
+MATRIX

MTRXLI INSPGM INSRET SMGHDR SMGMDS TRMPUT
+MATRIX

MXFRM INSPGM INSRET
+EXFRM

NAMELOG CLOSFX CLOSTR CMGCDS CMGOTH CMPLOG CREATR EQUALA ERASE EVALBM GLONOD INSCHR INSFLT INSINI
INSINT INSRET LGLTRE LIPCAP LIPCNM LIPDSN LIPENT LIPPRB LLSRCH LVPHDR MENU OEXIT OPENFX
OPENTR PROMPT SETOPT TIGCOP TIGET TRMGET TRMPUT TVGHDR TVGVEC TVPHDR TVPVEC

NBLANK +GEN

NMDIST DISTEU DISTMA DISTQC DISTVA GTRGNI GTRGNR INSPGM INSRET
+EVALUB PENMV

NMEVAL CHKEXS CLOSFX CMGCDS CMGLOG CMGOTH EQUALA ERASE GETLST GTRGNI GTSLOT INSINI INSRET LIGCOP
LIGDSN LIGSTR MENU OEXIT OPENFX OPENTR PENMV PROMPT TIGET TRMGET TRMPUT TVGHDR

NMV CHKEXS CLOSTR CMGCDS CMGLOG CMGOTH CMPLOG ERASE INSINI INSRET KILLND LIPCOP MENU NMVBSU
NMVOPT NMVRJT OEXIT OPENFX SETUP TRMPUT

NMVBSU COLAPS DITHER EVALBM EXPAND GNLIAE GNLVAE INSPGM INSRET INVERT LIGCLP LIGCNM LIGCOP LIGENT
LIGSTR LIPENT LOGMNC LVPLNK PTRGNI PTRGNR TIGCOV TIGMN TRMPUT UIGNOR
+NMV

NMVMOD CHKEXS CMGCDS CMGLOG CMGOTH EQUALA ERASE GETLST GTRGNI GTSLOT INSINI INSRET KILLND LIGCOP
LIGDSN LIGSTR MENU NMVOPT NMVRJT OEXIT OPENFX OPENTR PROMPT PTRGNI TIGET TRMGET TRMPUT
TVGHDR

NMVOPT GTRGNI INSPGM INSRET PROMPT PTRGNI TRMGET TRMPUT
+NMV NMVMOD

NMVPRT FILPUT GTRGNR INSINT INSPGM INSRET PACKB SUNMV
+PRTLOG

NMVRJT $SQRT ERASE GTRGNI GTRGNR INSPGM INSRET PTRGNI PTRGNR TRMGET TRMPUT
+NMV NMVMOD

NNBPRT FILPUT GTRGNI INSPGM INSRET
+PRTLOG

NNBREV DISTEU INSCHR INSINT INSPGM INSRET TVGVEC
+EVALNN

NNLBSU EVALBM GNLIAE GNLVAE INSCHR INSINT INSPGM INSRET LIGCLP LIGCNM LIGCOP LIGENT LIGSTR LIPENT
PTRGNI UIGNOR
+NRSTNBR

NNMEAN INSCHR INSINT INSPGM INSRET TIGCAP TIGMN TIGNAM TIPCAP TIPCOP TIPCOV TIPET TIPHDR TIPMN
TIPNAM TVPHDR TVPVEC
+NRSTNBR

(MMU CALLS MMUM; '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

MMMNCV	INSFLT INSINT INSPGM INSRET TVGVEC TVPVEC +MMSORT
MMMOD	CLOSTR CMGCDS CMGOTH ERASE GTRGNI INSINI INSRET LIGCOP LIGSTR MENU MODINI OEXIT OPENFX OPENTR PROMPT PTRGNI TRMGET TRMPUT UIGNOR
MMPASS	CLOSFX DISTEU EQUALA ERASE INSCHR INSINT INSPGM INSRET OPNTMP TIGCOP TRMGET TRMPUT TVGVEC TVPVEC +MRSTNBR
MMSORT	INSCHR INSINT INSPGM INSRET MMMNCV TIGCAP TIGNAM TIPCAP TIPCOP TIPCOV TIPET TIPHDR TIPMN TIPNAM TVPHDR +MRSTNBR
MODCHK	+ODTDMP
MODCOM	ADDMCV FIXKLV INSINT INSLOG INSPGM INSRET OEXIT TIGCAP TIGCOP TIGCOV TIGMN TIPCAP TIPCOP TRMPUT +COMMOD
MRMXFRM	$DSQRT CHKEXS CLOSFX CLOSTR CMGCDS CMGOTH ERASE INSINI INSINT INSRET MENU MXFRM OEXIT OPENFX OPENTR SAVMAT TIGET TIGVAR TRMPUT UIXFRM
MRSTNBR	ADUPCM CHKEXS CLOSFX CLOSTR CMGCDS CMGLOG CMGOTH CMPLOG CREATR DELETR ERASE INSINI INSRET KILLND LIGLOG LIPCOP MENU MMLBSU MMMEAN MMPASS MMSORT OEXIT OPENFX OPENTR SETUP TRMPUT UIMMBR
MXFRM	CLOSTR CREATR GNXTND INSINT INSPGM INSRET OPENTR TIGCAP TIGCOV TIGET TIGMN TIPCAP TIPCOP TIPCOV TIPET TIPHDR TIPMN TIPNAM TRMPUT TVGVEC TVPHDR TVPVEC +MATXFRM MORMXFRM
OBFGEN	ASCIEC CLOSE EQUALS FILGET FPUT INDEX INSSET LENGA LENGTH OEXIT OPENBL OPENS TRMGET TRMPUT
OBTCLS	DIGENT EQUALA INSINT INSPGM INSRET LENGA PROMPT TRMGET TRMPUT +INTENSIFY SELECT UIRSTR
OCLOSE	CLOSE OEXIT TRMPUT WRITEB +CLOSFX CLOSTR GEN OCREAT ODELET OMOVE OOPEN ORENAM RENAMT
OCREAT	CONCAT FCGENT FCGHDR FCPENT FCPHDR FCREAT FCTOPN GETFID OCLOSE PACKB SEARCH TRMPUT +CREAFX CREATR
ODELET	DELETE FCGENT FCGHDR FCPENT FCPHDR FCTOPN INSINT INSPGM INSRET OCLOSE TRMPUT +DELETR
ODTDMP	FLUSHT GNXTND INSSET MODCHK OEXIT OPENTR TIGCAP TIGCOP TIGET TIGHDR TIGNAM TRMGET TRMPUT VALID
OEXIT	CLOSE EXIT GETBUF TRMPUT WRITEB +ANYTHING APPEND BINWIDTH CDEFAULT CDISPLAY CHKEXS CLOSBL CMGCDS CMGLOG +CMGOPT CMGOTH CMGSCN CMINIT CNT1SP COMMOD CRANDTS CREATLOG CSCALE DBMDY DDATAMOD +DDATATREE DDSUBSTR DIGCME DIGCMH DIGDC DIGEFS DIGENT DIGHDI DIGMAX DIGNAM DIGROE

OLPARS PROGRAM CROSS REFERENCE LISTING -- D
(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM_NAME

+DIPNAM DISPCL DLOGTREE DLREGN DLSUBSTR DRAWBNDY DRAWLOG DRAWTREE
+DSCRMEAS DTRENAME DVEC DVGHDR DVGRDE DVGVEC DVPVEC EIGNXFRM ELIFRE FCGENT
+FCGHDR FCTOPN FGET FILEIN FILEOUT FISHER FISHMOD FLUSHB FPUT GEN GETFID GETHST GETOPT
+GOPTNM HELP HSGHDR HSGREC INSCHR INSDBL INSFLT INSHSP INSINI INSINT INSLOG INSPGM INSRET
+INSSET INTENSIFY L1ASDG L1CRDV L1EIGV L2ASDG L2CRDV L2EIGV L2FSHP LIGCAP LIGCLP LIGCNM
+LIGCOP LIGDCN LIGDSN LIGENT LIGLOG LIGPRB LIGSTR LISTLOGS LISTREES LLGENT LLGHDR
+LOGEVAL LTRENAME LVGENT LVGHDR LVGLNK MAKETREE MAKOPT MATRIX MATXFRM MEASXFRM
+MENU MODDFS MOVEC NAMELOG NMEVAL NMV NMVMOD NNMOD NODCOM NORMXFRM NRSTNBR OBFGEN
+OCLOSE ODTDMP OLTDMP OPTIMLMOD PROJNM PRTCM PRTDS PRTIDX PRTLOG PVGENT PVGHDR PVGNAM
+PWEVAL RANK RDISPLAY REASNAME REDRAW REPROJECT RESTRUCT S1CRDV S1EIGV
+S1GBC S1GHDR S2CRDV S2EIGV S2FSHP SCALRET SCALZM SELECT SETDS SETLOG SETOPT SLCTMEAS
+SMGENT SMGHDR SMGLNK SMGNDS SUMMCM THRESHMOD TIGCAP TIGCOP TIGCOV TIGET TIGHDR TIGNM
+TIGNAM TIGVAR TLGENT TLGHDR TMPXRD TRANSFRM TVGLOG TVGVEC UNION

OLTDMP EVALBM FLUSHT GNXTLN GOPTNM INSSET LIGCAP LIGCNM LIGDSN LIGENT LIGLOG LIGPRB LNDCHK OEXIT
OPENFX OPENTR TRMGET TRMPUT

OMOVE FCGENT FCPENT FCTOPN INSINT INSLOG INSPGM INSRET OCLOSE TRMPUT
+RENAMT

ONEVEC EQUALA FILPUT INSINT INSPGM INSRET LENGA PACKB PROMPT TRMGET TRMPUT TVGVEC
+PRTDS

OOPEN CONCAT FCGENT FCTOPN FOPEN GETFID OCLOSE
+LISTLOGS OPENFX OPENTR

OPENM +CIP FILEOUT HELP

OPENBL CONCAT FCREAT FOPEN GETFID INSCHR INSINT INSPGM INSRET PACKB
+FILEIN FNMCOV GETOPT GETVEC GOPTNM MAKOPT OBFGEN SETOPT

OPENFX OOPEN TRMPUT
+ANYTHING APPEND BINWIDTH CDEFAULT CDISPLAY CMINIT CMPMOD COMMOD CRANDTS
+CREATR CSCALE DBNDY DDATAMOD DDATATREE DDSUBSTR DELETR DLOGTREE
+DLSUBSTR DRAWBNDY DRAWLOG DRAWTREE DSCRMEAS DTRENAME DVEC
+EIGNXFRM ESETUP FILEIN FILEOUT FISHER FISHMOD FISHSU HELP INSINI INSSET INTENSIFY
+L1ASDG L1CRDV L1EIGV L2ASDG L2CRDV L2EIGV L2FSHP LISTLOGS LISTREES LOGEVAL LSETUP
+LTRENAME MAKETREE MATRIX MATXFRM MEASXFRM MOVEC NAMELOG NMEVAL NMV NMVMOD
+NNMOD NORMXFRM NRSTNBR OLTDMP OPENTR OPTDSP OPTIMLMOD PEPAIR PROJNM PRTCM PRTDS
+PRTIDX PRTLOG PWEVAL RANK RDISPLAY REASNAME REDRAW RENAMT REPROJECT S1CRDV
+S1EIGV S2CRDV S2EIGV S2FSHP SCALRET SCALZM SELECT SETDS SETLOG SLCTMEAS SUMMCM
+THRESHMOD TRANSFRM UIAPMD UICMND UICRLG UIL2FS UINMBR UIRSTR UIS2FS UNION

OPENS VCREAT VOPEN
+ANYTHING CLFYMD CMINIT CMPMOD CMPRT CREATLOG EIGPRT INSHSP INSINI MAKETREE
+MAKOPT MTRXDP MTRXEN OBFGEN PENMV PEPAIR PRINTR PRTDS PRTIDX PRTLOG USRI01

OPENTR CLOSFX INSPGM INSRET LLSRCH OOPEN OPENFX TLSRCH TRMPUT
+CRANDTS CREATLOG DDATAMOD DDSUBSTR DLSUBSTR DRAWLOG DRAWTREE
+DSCRMEAS DTRENAME DVEC ESETUP EVALNM EXFRM FILEOUT FISHMOD L1EIGV L2CRDV L2EIGV
+LSETUP LTRENAME MAKCMB MAKMRG MEASXFRM MODINI MOVEC NAMELOG NMEVAL NMVMOD NNMOD
+NORMXFRM NRSTNBR NXFRM ODTDMP OLTDMP OPTIMLMOD PRTDS PRTLOG PWEVAL REASNAME
+REPROJECT S1CRDV S1EIGV S2CRDV S2EIGV SETDS SETLOG SETUP THRESHMOD TRANSFRM
+UIAPMD UICMND UICRLG UIRSTR UIS2FS

(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

OPNTMP	FCREAT	GETFID	PACKB										
	+NMPASS	RESTRUCT		RNGLAP									
OPTDSP	BOUND	CLOSFX	CMGSCN	DIGHDI	DIGMAX	DIPEFS	DIPENT	DIPMDI	DIPMAX	DSCVTH	DVPHDR	DVPVEC	EQUALA
	ERASE	GIN	GTRGNR	INSFLT	INSINT	INSPGM	INSRET	LINSEG	MARK	MOVE	OPENFX	PROJCT	PROMPT
	PTRGNR	RCTNGL	SCAT	TEXT	TIGCOP	TIGNAM	TRMGET	TRMPUT	TVGVEC	WRTNOW			
	+OPTIMLMOD												
OPTIMLMOD	CHKEXS	CLOSTR	CMGCDS	CMGLOG	CMGOTH	DLREGN	EQUALA	ERASE	GETLST	GETPR	GTRGNI	GTRGNR	GTSLOT
	INSINI	INSRET	KILLND	LIGCOP	LIGDSN	LIGLOG	LIGSTR	LIPLOG	MENU	OEXIT	OPENFX	OPENTR	OPTDSP
	PROMPT	SETOPT	TIGET	TIGNAM	TRMGET	TRMPUT	TVGHDR						
ORENAM	CONCAT	FCGENT	FCPENT	FCTOPN	INSCHR	INSINT	INSPGM	INSRET	OCLOSE	PACKB	RENAME	SEARCH	TRMPUT
	+RENAMT												
PACKB	+ALLVEC	CORMAT	COVMAT	FILEOUT	GP1PRT	GP2PRT	MNSTDV	MNVECD	MNVPRT	OCREAT	ONEVEC	OPENBL	OPNTMP
	+ORENAM	PWSPRT	RNGLAP										
PACKW	+CMGCDS	CMGDIR	CMGLOG	CMGOPT	CMGSCN	DIGENT	DIGROH	DIGROI	LIGDCN	LIGENT	LIGSTR	LLGENT	PVGNAM
	+SMGMDS	TLGENT											
PAIRLG	EV2SP	FISH	GTRGNI	GTRGNR	INSPGM	INSRET	LIGPRB	TRMPUT					
	+EVALU8	PEPAIR											
PENMV	CLOSE	CMDISP	CMGCDS	CMGLOG	CMPRT	DIPCME	DIPCMH	EQUALA	FILPUT	GTRGNI	GTRGNR	INSPGM	INSRET
	LIGLOG	LIGPRB	MNDIST	OPENS	PRINTR	PROMPT	TIGCOP	TIGNAM	TRMGET	TRMPUT	TVGVEC	TVPLOG	
	+NMEVAL												
PEPAIR	CLOSE	CLOSFX	CMDISP	CMGCDS	CMGLOG	CMPRT	DIPCME	DIPCMH	EQUALA	FILPUT	GTRGNI	INSPGM	INSRET
	LIGLOG	OPENFX	OPENS	PAIRLG	PRINTR	PROMPT	TIGCOP	TIGNAM	TRMGET	TRMPUT	TVGVEC	TVPLOG	
	+PWEVAL												
PRECPT	CMGSCN	EQUALA	ERASE	INSPGM	INSRET	PROMPT	TRMGET	TRMPUT					
	+DSCRMEAS												
PRINTR	CLOSE	INSPGM	INSRET	OPENS	TRMPUT								
	+CLFYMD	CMPMOD	CMPRT	CREVAL	EIGPRT	MAKMRG	MTRXDP	PENMV	PEPAIR	PRTDS	PRTIDX	PRTLOG	
PROJCT	$SQRT	INSPGM	INSRET										
	+DSPROJ	FISH	FISHSU	LNPROJ	OPTDSP	THRDSP							
PROJMN	CLOSFX	CMGCDS	CMGSCN	DIGDC	ERASE	INSINI	INSINT	INSRET	MEAN1	MEAN2	MENU	OEXIT	OPENFX
	TEXT	TRMPUT											
PROMPT	INSPGM	INSRET	TRMPUT										
	+BINWIDTH		CDEFAULT		CLFYMD	CMINIT	CRANDTS	CREVAL	DDATATREE		DDSUBSTR		
	+DLOGTREE		DLSUBSTR		DRAWLOG	DRAWTREE		DTRENAME		DVEC	EIGPRT	ELIMEA	FILEOUT
	+FISHMOD	FISHTH	FISHVC	GTHRSH	HELP	L1CRDV	L2CRDV	LTRENAME		MAKETREE		MATRIX	MATXFRM
	+MICMAC	MODINI	MTRXDE	MTRXDP	MTRXEN	NAMELOG	NMEVAL	NMVMOD	NMVOPT	NNMOD	OBTCLS	ONEVEC	OPTDSP
	+OPTIMLMOD		PENMV	PEPAIR	PRECPT	PRTDS	PRTIDX	PRTLOG	PWEVAL	RANK	REASNAME		
	+REPROJECT		RODISP	S1CRDV	S2CRDV	SAVMAT	SETDS	SETLOG	SETUP	SLCTMEAS		SLTEIG	THRDSP
	+THRESHMOD		UIAPNB	UICMMD	UICRLG	UIDCMS	UIGMOR	UIL2FS	UILASD	UILGE1	UILGE2	UILGE3	UILGE4
	+UILGE5	UILGE6	UINMBR	UIRSTR	UIS2FS	UIXFRM	UNION	USRI01	ZOM1SP	ZOM2SP			

OLPARS PROGRAM CROSS REFERENCE LISTING -- D
(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

PRTCM CMGCDS CMPRT DIGDC ERASE INSINI INSRET MENU OEXIT OPENFX TRMPUT

PRTDS ALLVEC CLOSE CLOSFX CMGOTH CORMAT COVMAT EQUALA ERASE GNXTND INSINI INSINT INSLOG INSRET
LENGA MENU MNSTDV MNVECD OEXIT ONEVEC OPENFX OPENS OPENTR PRINTR PROMPT RNGLAP SELCLS
TIGCAP TIGET TIGHDR TLSRCH TRESTR TRMGET TRMPUT

PRTIDX CLOSE CLOSFX CLSCAT CMGCDS CMGOTH CMGSCN DIGDC DIGENT DIGHDI DVGVEC EQUALA ERASE FILPUT
GIN INSINI INSINT INSRET MENU MICMAC MOVE OEXIT OPENFX OPENS PRINTR PROMPT RCTNGL
SIGBC TRMGET TRMPUT

PRTLOG CLOSE CMGLOG CMGOTH EQUALA ERASE EVALBM FILPUT GOPTNM GP1PRT GP2PRT INSINI INSINT INSRET
LIGCAP LIGCLP LIGCNM LIGDCN LIGDSN LIGLOG LIGPRB LIGSTR MENU MMVPRT MMBPRT OEXIT OPENFX
OPENS OPENTR PRINTR PROMPT PRTSTR PWSPRT TRMGET TRMPUT

PRTSTR EVALBM FILPUT GNXTLN GOPTNM INSPGM INSRET LIGCLP LIGLOG LIGSTR
+PRTLOG

PSORT INSPGM INSRET
+MAKOPT

PTRGNI GNLVAE INSINT INSLOG INSPGM INSRET LVGLNK LVPENT LVPLNK
+CRLV2 FISHMOD FISHSL FISHSU FISHVC MMVBSU MMVMOD MMVOPT MMVRJT MMLBSU MMMOD

PTRGNR GNLVAE INSINT INSLOG INSPGM INSRET LVGLNK LVPENT LVPLNK
+CRLV1 CRLV2 FISHSU FISHTH MMVBSU MMVRJT OPTDSP THRDSP

PUTHST FLUSHB FPUT
+INSRET

PVGENT FGET INSPGM INSRET OEXIT TRMPUT
+CRLV1 CRLV2 DCRIM DCRIM1 DSPROJ EIGLPV EIGNPV EIGNXFRM EIGPRT GTHRSH LNPROJ
+RESTRUCT

PVGHDR FGET INSINT INSPGM INSRET OEXIT TRMPUT
+CLSCAT DCRIM1 DSPROJ EIGLPV EIGNPV EIGPRT LNPROJ SLTEIG

PVGNAM FGET INSPGM INSRET OEXIT PACKW TRMPUT

PVPENT FPUT INSPGM INSRET
+EIGNPV ELIMEA L1ASDG L1CRDV L2ASDG L2CRDV L2FSHP S1CRDV S2CRDV S2FSHP

PVPHDR FPUT INSPGM INSRET
+CLSCAT ELIMEA ESETUP INITD L1ASDG L1CRDV L2ASDG L2CRDV L2FSHP MICMAC S1CRDV S2CRDV S2FSHP
+SLTEIG

PVPNAM FPUT INSPGM INSRET ITREAL
+ESETUP INITB

PWEVAL CHKEXS CLOSFX CMGCDS CMGLOG CMGOTH EQUALA ERASE GETLST GTRGNI GTSLOT INSINI INSRET LIGCOP
LIGDSN LIGSTR MENU OEXIT OPENFX OPENTR PEPAIR PROMPT TIGET TRMGET TRMPUT TVGHDR

PWSPRT FILPUT GOPTNM GTRGNI GTRGNR INSINT INSPGM INSRET PACKB SU2SP
+PRTLOG

(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

RAN
+CRANDTS

RANK
CDSCHK CMGCDS CMGOTH DIGROH DIPROE DIPROH DVGROE ERASE INSINI INSINT INSRET MENU OEXIT
OPENFX PROMPT RODCLS RODISP RSORTA RSORTD TRMGET TRMPUT

RCTNGL
INSPGM INSRET LINSEG
+CLSCAT DRAWBNDY MEAN2 MOVEC OPTDSP PRTIDX ZOM2SP

RDISPLAY
CLOSFX CLSCAT CMDISP CMGCDS CMGOTH DIGDC ERASE INSINI INSRET MENU MICMAC OEXIT OPENFX
TRMPUT

READB
+FGET FPUT

REARR
FIXKLV GARBND GNXTND INSPGM INSRET SBUPMC TIGCAP TIGCOP TIGNAM TIPCAP TIPCOP TIPNAM
+DDATAMOD DVEC

REASNAME
CHARFL CMGLOG CMGOTH CMGSCN EQUALA ERASE GETLST INSINI INSRET LENGA LGLNOD LIGDCN LIPCOP
LIPDCN MENU OEXIT OPENFX OPENTR PROMPT TRMGET TRMPUT

REDRAW
CLOSFX CMGSCN DIGHDI DIGMAX DVGVEC EXT1ST EXTLST INSFLT INSINI INSINT INSPGM INSRET LINSEG
MARK OEXIT OPENFX RXINT RYINT SCATR TEXT TRMPUT

REDVEC
INSPGM INSRET
+TRANSFRM

RELUN
TRMPUT
+HELP

RENAME
+ORENAM

RENAMT
DELETR INSCHR INSINT INSLOG INSPGM INSRET LLGENT LLPENT LLSRCH OCLOSE OMOVE OPENFX ORENAM
TLGENT TLPENT TLSRCH TRMPUT
+COMNOD CRANDTS DDSUBSTR DTRENAME LTRENAME

REPROJECT
CDSCHK CLOSFX CLOSTR CLSCAT CMGCDS CMGLOG CMGOTH DIGDC DSPROJ EIGPRT EQUALA ERASE GCLIST
INSINI INSINT INSRET LIGCOP LMPROJ MENU MICMAC OEXIT OPENFX OPENTR PROMPT SETOPT SLTEIG
TIGCOP TIGET TIGNAM TRMGET TRMPUT

RESTRUCT
CLNODE CLOSFX CMNCOV DIGHDI DSCVTH DVGVEC ERASE EV1SP EV2SP FIXKLV GNXTAN INSINI INSINT
INSRET MENU OEXIT OPNTMP PVGENT TIGCAP TIGNAM TIPCOV TIPMN TRMPUT TVGVEC TVPVEC UIRSTR

RNGLAP
FILPUT INSFLT INSINT INSPGM INSRET OPNTMP PACKB TMPXRD TMPXWT TRMPUT TVGVEC
+PRTDS

RODCLS
DIGROH EQUALA INSPGM INSRET
+RANK

RODISP
CMGSCN DIGROE DIGROH DIGROI EQUALA ERASE INSINT INSPGM INSRET PROMPT TEXT TRMGET TRMPUT
WRTNOW WRTOBS
+DSCRMEAS RANK SLCTMEAS UNION

RSORTA
INSFLT INSPGM INSRET TRMPUT
+RANK UNION

OLPARS PROGRAM CROSS REFERENCE LISTING -- D
(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

RSORTD
INSFLT INSPGM INSRET TRMPUT
+DSCRMEAS RANK UNION

RSTORE
EQUALA EVALU8 GLONOD INSPGM INSRET LIGDSN TIGCOP TIGET TRMPUT TVGHDR TVGVEC TVPHDR TVPLOG
+LOGEVAL

RXINT
INSFLT INSINT INSPGM INSRET
+EXT1ST EXTLST REDRAW

RYINT
INSFLT INSINT INSPGM INSRET
+EXT1ST EXTLST REDRAW

S1CRDV
CLOSFX CNGCDS CNGOTH CRPROJ DIPHDI ERASE INITD INSINI INSRET MENU MICMAC OEXIT OPENFX
OPENTR PROMPT PVPENT PVPHDR TIGCOP TIGET TRMGET TRMPUT

S1EIGV
CLOSFX CLOSTR CNGCDS CNGOTH DIPHDI DSPROJ EIGMPV EIGPRT ELIMEA ERASE INITD INSINI INSRET
MENU MICMAC OEXIT OPENFX OPENTR SLTEIG TIGCOP TIGET TRMPUT

S1GBC
FGET INSPGM INSRET OEXIT TRMPUT
+MACROP MICROP PRTIDX

S1GHDR
FGET INSPGM INSRET OEXIT TRMPUT
+CNT1SP

S1PBC
FPUT INSPGM INSRET
+CNT1SP

S1PHDR
FPUT INSPGM INSRET
+CNT1SP INITD

S2CRDV
CLOSFX CLSCAT CNGCDS CNGOTH CRPROJ DIPHDI ERASE INITD INSINI INSRET MENU OEXIT OPENFX
OPENTR PROMPT PVPENT PVPHDR TIGCOP TIGET TRMGET TRMPUT

S2EIGV
CLOSFX CLSCAT CNGCDS DIPHDI DSPROJ EIGMPV EIGPRT ELIMEA ERASE INITD INSINI INSRET MENU
OEXIT OPENFX OPENTR SLTEIG TIGCOP TIGET TRMPUT

S2FSHP
$SORT CHKEXS CLSCAT CNGCDS DCRIM DIPHDI DSPROJ ERASE INSINI INSINT INSRET MENU MODDFS
OEXIT OPENFX PVPENT PVPHDR TRMPUT UIS2FS

SAVMAT
CHARFL EQUALA GNSMAE IDCHK INSINT INSPGM INSRET LENGA PROMPT SMGHDR SMGLNK SMGNDS SMPENT
SMPHDR SMPLNK SMPNDS TRMGET TRMPUT
+EIGNXFRM NORMXFRM

SBUPMC
INSINT INSPGM INSRET SUBMNC TIGCOP TIGCOV TIGHDR TIGMN
+DVEC MOVEC REARR

SCALRET
CLOSFX CLSCAT CNGOTH DIGHBI DIGMAX DIPHDI DIPMAX DVPHDR INSINI INSINT INSRET MENU MICMAC
OEXIT OPENFX TRMPUT

SCALZM
CLOSFX CNGSCN DIGHDI DIGMAX INSFLT INSINI INSINT INSRET MENU OEXIT OPENFX TRMPUT ZOM1SP
ZOM2SP

SCAT
DIGENT DIGHDI DIGMAX DIPMAX DVGHDR DVGVEC DVPHDR DVPVEC INSFLT INSINT INSPGM INSRET MARK

(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

SCATR
+CLSCAT OPTDSP

SCATR INSFLT INSINT INSPGM INSRET
+CLUS CNT1SP REDRAW SCAT

SEARCH EQUALS FCGENT FCGHDR
+OCREAT ORENAM

SELCLS EQUALA INSCHR INSLOG INSPGM INSRET TRMGET TRMPUT
+DDATAMOD DVEC PRTDS SLPAIR UICMND UICRLG UILASD UIS2FS

SELECT CLOSFX CLSCAT CMGOTH DIGHDI DIPENT EQUALA ERASE INSINI INSINT INSLOG INSRET MENU MICMAC
OBTCLS OEXIT OPENFX TRMPUT

SETDS CMGOTH CMPCDS ERASE FLUSHT INSINI INSRET LENGA MENU OEXIT OPENFX OPENTR PROMPT TIGET
TIGHDR TISRCH TLSRCH TRMGET TRMPUT

SETLOG CLOSFX CMGOTH CMPLOG ERASE INSINI INSINT INSRET LGLTRE LIGCOP LLSRCH MENU OEXIT OPENFX
OPENTR PROMPT TRMGET TRMPUT

SETOPT CLOSBL CMGDIR CMPOPT EQUALS FGET INSCHR INSINT INSPGM INSRET OEXIT OPENBL TRMPUT
+DSCRMEAS FISHMOD FISHSU INITD LSETUP MENU MOVEC NAMELOG OPTIMLMOD REPROJECT
+SETUP THRESHMOD

SETUP CMGCDS CMGLOG CMGOTH EQUALA GCLIST GETLST GLONOD INSPGM INSRET LIGDSN LIPCOP LIPLOG OPENTR
PROMPT SETOPT TIGET TIGNAM TRMGET TRMPUT TVGHDR
+FISHER L1ASDG L1CRDV L1EIGV L2ASDG L2CRDV L2EIGV MMV NRSTNBR UIL2FS

SINGLE EVALU8 GLONOD INSPGM INSRET LIGCOP LIGDCN TIGET TRMPUT UILGE3 UILGE4
+LOGEVAL

SLCTMEAS CDSCHK CMGCDS CMGOTH DIGROE DIGROH DIGROI DIPROI ERASE INSFLT INSINI INSINT INSRET MENU
OEXIT OPENFX PROMPT RODISP TRMGET TRMPUT

SLPAIR GLONOD INSINT INSPGM INSRET SELCLS TRMPUT
+UIL2FS

SLTEIG CMGOTH DIPHDI INSPGM INSRET PROMPT PVGHDR PVPHDR TRMGET TRMPUT
+L1EIGV L2EIGV REPROJECT S1EIGV S2EIGV

SMGENT FGET INSPGM INSRET OEXIT TRMPUT
+MATXFRM MTRXDP

SMGHDR FGET INSPGM INSRET OEXIT TRMPUT
+GMSMAE MTRXDE MTRXDP MTRXEN MTRXLI SAVMAT

SMGLNK FGET INSPGM INSRET OEXIT TRMPUT
+GMSMAE MTRXDE MTRXEN SAVMAT

SMGMDS FGET INSPGM INSRET OEXIT PACKW TRMPUT
+MATXFRM MTRXDE MTRXDP MTRXEN MTRXLI SAVMAT

SMPENT FPUT INSPGM INSRET

OLPARS PROGRAM CROSS REFERENCE LISTING — D
(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

	+MTRXEN SAVMAT
SMPHDR	FPUT INSPGM INSRET
	+GEN GNSMAE MTRXDE MTRXEN SAVMAT
SMPLNK	FPUT INSPGM INSRET
	+GNSMAE MTRXDE MTRXEN SAVMAT
SMPMDS	FPUT INSPGM INSRET ITREAL
	+MTRXDE MTRXEN SAVMAT
SQRT	INSPGM INSRET
	+EIGMPV
SU1SP	GTRGNR INSFLT INSINT INSPGM INSRET
	+EVALU8 GP1PRT
SU2SP	GTRGNI GTRGNR INSINT INSPGM INSRET
	+EVALU8 GP2PRT PWSPRT
SUBMNC	INSFLT INSINT INSPGM INSRET TIGCOP TIGCOV TIGMN TIPCOP TIPCOV TIPMN
	+CRANDTS SBUPMC
SUFISH	GTRGNI INSPGM INSRET
	+EVALU8
SUMMCH	DIGCMH DIGDC ERASE INSINI INSRET MENU OEXIT OPENFX TRMPUT
SUMMV	GTRGNI GTRGNR INSPGM INSRET
	+EVALU8 MMVPRT
TEXTXT	+MOVE
TEXT	+CLSCAT DRAWBNDY MACROP MENU MICMAC MICROP OPTDSP PROJMN REDRAW RODISP WRTODS ZOM2SP
THRDSP	CMGSCN DLREGN EQUALA ERASE GIN GTRGNR INSPGM INSRET LINSEG MOVE PROJCT PROMPT PTRGNR
	TIGCOP TIGHDR TIGMN TRMGET TRMPUT TVGVEC WRTNOW
	+THRESHMOD
THRESHMOD	CHKEXS CLOSTR CMGCDS CMGLOG CMGOTH EQUALA ERASE GETLST GETPR GTRGNI GTSLOT INSINI INSRET
	KILLND LIGCOP LIGDSN LIGSTR MENU OEXIT OPENFX OPENTR PROMPT SETOPT THRDSP TIGET TIGNAM
	TRMGET TRMPUT TVGHDR
TIGCAP	FGET INSPGM INSRET OEXIT TRMPUT
	+CLNODE COMMOD CRANDTS DDSUBSTR EXFRM GDSTRC GNXTMD MAKCMB MAKMRG MEASXFRM MMMEAN
	+MMSORT MODCOM MXFRM ODTDMP PRTDS REARR RESTRUCT TISRCH TRANSFRM UICMND UNIQND
TIGCOP	FGET INSPGM INSRET OEXIT TRMPUT
	+ADDMCV ADUPCM ALMCPM APPEND CLFYMD CLNODE CMPMOD COMMOD CRANDTS CREVAL DCRIM1 DDSUBSTR
	+DSCRMEAS DSPROJ DVEC EVALMM FIXXLV FXLTMP GLONOD GNXTAN L2CRDV L2EIGV LCPROJ LNPROJ
	+LOGMM LOGMNC MAKCMB MAKMRG MEASXFRM MMSTDV MOVEC NAMELOG NMPASS MODCOM ODTDMP OPTDSP
	+PENMV PEPAIR REARR REPROJECT RSTORE S1CRDV S1EIGV S2CRDV S2EIGV SBUPMC SUBMNC THRDSP
	+TRANSFRM TRESTR UIAPMD UICRLG UILGE3 UIRSTR

(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

TIGCOV	FGET	INSPGM	INSRET	OEXIT	TRMPUT							
	+ADDMCV	ADUPCM	COPYCV	CORMAT	COVMAT	DCRIM1	DVEC	EIGNPV	EXFRM	MAKCMB	MAKMRG	MOVEC	MMVBSU
	+NODCOM	NXFRM	SBUPMC	SUBMNC	TRANSFRM								
TIGET	FGET	INSPGM	INSRET	OEXIT	TRMPUT							
	+CLFYMD	CMPMOD	CRANDTS	DDATAMOD		DDSUBSTR		DRAWTREE		DSCRMEAS		DVEC	ESETUP
	+EVALMM	EXFRM	FILEOUT	FXLTMP	MAKCMB	MAKMRG	MEASXFRM		MODINI	MOVEC	NAMELOG	MMEVAL	MMVMOD
	+NORMXFRM		NXFRM	OBTDMP	OPTIMLMOD		PRTDS	PWEVAL	REPROJECT		RSTORE	S1CRDV	S1EIGV
	+S2CRDV	S2EIGV	SETDS	SETUP	SINGLE	THRESHMOD		TRANSFRM		UIAPMD	UICMND	UICRLG	UIRSTR
	+UIS2FS												
TIGHDR	FGET	INSPGM	INSRET	OEXIT	TRMPUT								
	+ADUPCM	CRANDTS	DDSUBSTR		DRAWTREE		EVALMM	FILEOUT	FXLTMP	GARBMD	GMXTAN	LCPROJ	MAKCMB
	+MAKMRG	MOVEC	ODTDMP	PRTDS	SBUPMC	SETDS	THRDSP	UIAPMD	UICMND	UIRSTR			
TIGMM	FGET	INSPGM	INSRET	OEXIT	TRMPUT								
	+ADDMCV	ADUPCM	COPYMM	DCRIM1	DSCRMEAS		DSPROJ	DVEC	EXFRM	FISHSU	LCPROJ	LMPROJ	MAKCMB
	+MAKMRG	MMSTDV	MMVECD	MOVEC	MMVBSU	MMMEAN	NODCOM	NXFRM	SBUPMC	SUBMNC	THRDSP	TRANSFRM	
	+UILASD												
TIGMAM	FGET	INSPGM	INSRET	OEXIT	TRMPUT								
	+CMPMOD	CREVAL	EVALMM	FXLTMP	GDSTRC	GLOMOD	GMXTMD	GTSLOT	LCPROJ	LMPROJ	MAKCMB	MAKMRG	MODDFS
	+MODINI	MMMEAN	MMSORT	ODTDMP	OPTDSP	OPTIMLMOD		PEMMV	PEPAIR	REARR	REPROJECT		
	+RESTRUCT		SETUP	THRESHMOD		TISRCH	UICMND	UICRLG	UIRSTR	UNIQMD			
TIGVAR	FGET	INSPGM	INSRET	OEXIT	TRMPUT								
	+DSCRMEAS		MMSTDV	NORMXFRM									
TIME	+WRTNOW												
TIPCAP	FPUT	INSPGM	INSRET										
	+ALNCPM	CLNODE	CRANDTS	DDSUBSTR		EXFRM	FILEIN	MAKCMB	MAKMRG	MEASXFRM		MMMEAN	MMSORT
	+NODCOM	NXFRM	REARR	TRANSFRM									
TIPCOP	FPUT	INSPGM	INSRET										
	+ADDMCV	ALNCPM	CLNODE	COMMOD	DDSUBSTR		DVEC	EXFRM	FIXKLV	GARBMD	MAKCMB	MAKMRG	
	+MEASXFRM		MMMEAN	MMSORT	NODCOM	NXFRM	REARR	SUBMNC	TRANSFRM				
TIPCOV	FPUT	INSPGM	INSRET										
	+ADDMCV	COPYCV	CRANDTS	DVEC	EXFRM	FMMCOV	MAKCMB	MAKMRG	MEASXFRM		MOVEC	MMMEAN	MMSORT
	+NXFRM	RESTRUCT		SUBMNC	TRANSFRM								
TIPET	FPUT	INSPGM	INSRET										
	+CRANDTS	EXFRM	FILEIN	GARBMD	GMXTAN	MAKCMB	MAKMRG	MEASXFRM		MMMEAN	MMSORT	NXFRM	
	+TRANSFRM												
TIPMDR	FPUT	INSPGM	INSRET										
	+CRANDTS	EXFRM	FILEIN	GARBMD	GMXTAN	MAKCMB	MAKMRG	MEASXFRM		MMMEAN	MMSORT	NXFRM	
	+TRANSFRM												
TIPMM	FPUT	INSPGM	INSRET										
	+ADDMCV	COPYMM	CRANDTS	DVEC	EXFRM	FMMCOV	MAKCMB	MAKMRG	MEASXFRM		MOVEC	MMMEAN	MMSORT
	+NXFRM	RESTRUCT		SUBMNC	TRANSFRM								

CLPARS PROGRAM CROSS REFERENCE LISTING — D
(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

TIPNAM
FPUT INSPGM INSRET
+ALMCPN CLMODE COMMOD CRANDTS DDSUBSTR EXFRM FILEIN MAKCMD MAKMRG MEASXFRM MMMEAN
+NMSORT NXFRM REARR TRANSFRM

TISRCH
EQUALA GNXTND INSCHR INSINT INSLOG INSPGM INSRET TIGCAP TIGNAM
+DDSUBSTR DRAWTREE SETDS UIAPND UICMND

TLGENT
FGET INSPGM INSRET OEXIT PACKW TRMPUT
+CREATR DELETR LISTREES RENAMT TLSRCH

TLGHDR
FGET INSPGM INSRET OEXIT TRMPUT
+CREATR DDATATREE DELETR LISTREES TLSRCH

TLPENT
FPUT INSPGM INSRET ITREAL
+CREATR DELETR RENAMT

TLPHDR
FPUT INSPGM INSRET
+CREATR DELETR

TLSRCH
INSCHR INSINT INSLOG INSPGM INSRET TLGENT TLGHDR VALUEA
+CRANDTS CREATR DELETR DTRENAME MAKETREE OPENTR PRTDS RENAMT SETDS UIAPND UICMND
+UINMBR UIXFRM USRI01

TMPXRD
FGET INSINT INSPGM INSRET OEXIT TRMPUT
+RNGLAP

TMPXWT
FPUT INSINT INSPGM INSRET
+RNGLAP

TRANSFRM
ADUPCM CDSCHK CLOSFX CLOSTR CMGCDS CMGOTH CMMCOV COLAPS CREATR DIGROM DIGROI ERASE GNXTND
INSCHR INSFLT INSINI INSINT INSRET MENU OEXIT OPENFX OPENTR REDVEC TIGCAP TIGCOP TIGCOV
TIGET TIGMM TIPCAP TIPCOP TIPCOV TIPET TIPHDR TIPMM TIPNAM TRMPUT TVGVEC TVPHDR TVPVEC
UIXFRM

TRESTR
FILPUT INSPGM INSRET TIGCOP
+PRTDS

TRMGET
+BINWIDTH BNDEXP CDEFAULT CIP CLFYMD CMINIT CRANDTS CREVAL CSCALE DDATATREE
+DDSUBSTR DLOGTREE DLSUBSTR DRAWLOG DRAWTREE DTRENAME DVEC EIGPRT
+ELIMEA FILEOUT FISHMOD FISHTH FISHVC GETPR GETSTR GTHRSH HELP INSHSP L1CRDV L2CRDV
+LTRENAME MAKETREE MATRIX MATXFRM MICMAC MODINI MTRXDE MTRXDP MTRXEN NAMELOG NMEVAL
+NMVMOD NMVOPT NMVRJT NNMOD NMPASS OBFGEN OBTCLS ODTDMP OLTDMP ONEVEC OPTDSP OPTINLMOD
+PENMV PEPAIR PRECPT PRTDS PRTIDX PRTLOG PWEVAL RANK REASNAME REPROJECT RODISP
+S1CRDV S2CRDV SAVMAT SELCLS SETDS SETLOG SETUP SLCTMEAS SLTEIG THRDSP THRESHMOD
+UIAPND UICMND UIDCMS UIGNOR UIL2FS UILASD UILGE1 UILGE2 UILGE3 UILGE4 UILGE5 UILGE6 UINMBR
+UIRSTR UIS2FS UIXFRM UNION USRI01 ZOM1SP ZOM2SP

TRMPUT
+ALLVEC ANYTHING APPEND BINWIDTH BNDEXP CDEFAULT CDISPLAY CHKEXS CIP
+CLFYMD CLOSBL CLSCAT CMDISP CMGCDS CMGLOG CMGOPT CMGOTH CMGSCN CMINIT CMPMOD CMPRT CNT1SP
+COMMOD CORMAT COVMAT CRANDTS CREATLOG CREATR CREVAL CSCALE DBNDY DCRIM1 DCRIM2
+DDATAMOD DDATATREE DDSUBSTR DELETR DIGCME DIGCMM DIGDC DIGEFS DIGENT DIGHDI
+DIGMAX DIGNAM DIGROE DIPMAX DIPNAM DISPCL DLOGTREE DLREGN DLSUBSTR DRAWBNDY
+DRAWLOG DRAWTREE DSCRMEAS DTRENAME DVEC DVGHDR DVGROE DVGVEC DVPVEC
+EIGNXFRM EIGPRT ELIFRE ELIMEA ESETUP EVALMM EXFRM FCGENT FCGHDR FCTOPN FGET FILEIN

PROGRAM NAME													
	+FILEOUT	FISHER	FISHMOD	FISHSU	FISHTH	FISHVC	FLUSHB	FMNCOV	FPUT	GEN	GETFID	GETHST	GETOPT
	+GETPR	GETVEC	GLSTRC	GOPTNM	GTHRSH	HELP	HSGHDR	HSGREC	IDX	INITD	INSCHR	INSDBL	INSFLT
	+INSHSP	INSINI	INSINT	INSLOC	INSPGM	INSRET	INSSET	INTENSIFY		L1ASDG	L1CRDV	L1EIGV	L2ASDG
	+L2CRDV	L2EIGV	L2FSHP	LISTED	LIGCAP	LIGCLP	LIGCNM	LIGCOP	LIGDCM	LIGDSM	LIGENT	LIGLOG	LIGPRB
	+LIGSTR	LISTLOGS		LISTREES		LLGENT	LLGHDR	LOGEVAL	LSETUP	LTRENAME		LVGENT	LVGHDR
	+LVGLNK	MACROP	MAKCMB	MAKETREE		MAKMRG	MAPT	MATRIX	MATXFRM	MEAN1	MEASXFRM		MENU
	+MICMAC	MICROP	MNSTDV	MNVECD	MODDFS	MODINI	MOVEC	MTRXDE	MTRXDP	MTRXEN	MTRXLI	NAMELOG	NMEVAL
	+NMV	NMVBSU	NMVMOD	NMVOPT	NMVRJT	NNMOD	NNPASS	NODCOM	NORMXFRM		NRSTNBR	NXFRM	OBFGEN
	+OBTCLS	OCLOSE	OCREAT	ODELET	ODTDMP	OEXIT	OLTDMP	OMOVE	ONEVEC	OPENFX	OPENTR	OPTDSP	
	+OPTIMLMOD		ORENAM	PAIRLG	PENMV	PEPAIR	PRECPT	PRINTR	PROJMN	PROMPT	PRTCM	PRTDS	PRTIDX
	+PRTLOG	PVGENT	PVGHDR	PVGNAM	PWEVAL	RANK	RDISPLAY		REASNAME		REDRAW	RELUN	RENAMT
	+REPROJECT		RESTRUCT		RNGLAP	RODISP	RSORTA	RSORTD	RSTORE	S1CRDV	S1EIGV	S1GBC	S1GHDR
	+S2CRDV	S2EIGV	S2FSHP	SAVMAT	SCALRET	SCALZM	SELCLS	SELECT	SETDS	SETLOG	SETOPT	SETUP	SINGLE
	+SLCTMEAS		SLPAIR	SLTEIG	SMGENT	SMGHDR	SMGLNK	SMGMDS	SUMMCM	THRDSP	THRESHMOD		TIGCAP
	+TIGCOP	TIGCOV	TIGET	TIGHDR	TIGNM	TIGNAM	TIGVAR	TLGENT	TLGHDR	TMPXRD	TRANSFRM		TVGLOG
	+TVGVEC	UIAPND	UICMND	UICRLG	UIGNOR	UIL2FS	UILASD	UILGE1	UILGE3	UINMBR	UIRSTR	UIS2FS	UIXFRM
	+UNION	USRIO1	WRTNOW	WRTODS	ZOM1SP	ZOM2SP							
TVGHDR	FGET	INSPGM	INSRET										
	+APPEND	CLFYMD	CMPMOD	COPYVC	DDATANOD		DLSUBSTR		DVEC	FISHMOD	MODINI	NAMELOG	NMEVAL
	+NMVMOD	OPTIMLMOD		PWEVAL	RSTORE	SETUP	THRESHMOD		UIRSTR				
TVGLOG	FGET	INSINT	INSPGM	INSRET	OEXIT	TRMPUT							
	+FXLTMP												
TVGVEC	FGET	INSPGM	INSRET	OEXIT	TRMPUT								
	+ALLVEC	CLFYMD	CMNCOV	CMPMOD	COPYVC	CRANDTS	CREVAL	DSCRMEAS		DSPROJ	DVEC	EXFRM	FILEIN
	+FILEOUT	FMNCOV	LCPROJ	LNPROJ	LOGNM	LOGNNC	MAKCMB	MAKMRG	MEASXFRM		MOVEC	NAMELOG	NNBREV
	+NNMNCV	NNPASS	NXFRM	ONEVEC	OPTDSP	PENMV	PEPAIR	RESTRUCT		RNGLAP	RSTORE	THRDSP	
	+TRANSFRM		UILGE3										
TVPHDR	FPUT	INSPGM	INSRET										
	+APPEND	CLFYMD	CMPMOD	COMMOD	COPYVC	CRANDTS	DDATANOD		DDSUBSTR		DVEC	EXFRM	FILEIN
	+MAKCMB	MAKMRG	MEASXFRM		NAMELOG	NNMEAN	NNSORT	NXFRM	RSTORE	TRANSFRM		UIRSTR	
TVPLOG	FPUT	INSINT	INSPGM	INSRET									
	+CLFYMD	CMPMOD	CREVAL	FXLTMP	PENMV	PEPAIR	RSTORE						
TVPVEC	FPUT	INSPGM	INSRET										
	+CMNCOV	COPYVC	CRANDTS	DVEC	EXFRM	FILEIN	FMNCOV	GETVEC	MAKCMB	MAKMRG	MEASXFRM		MOVEC
	+NAMELOG	NNMEAN	NNMNCV	NNPASS	NXFRM	RESTRUCT		TRANSFRM					
TYPE	INSCHR	INSPGM	INSRET	VALUEA									
	+IDCHK												
UIAPND	CHKEXS	CLOSFX	EQUALA	INSINT	INSPGM	INSRET	LENGA	LGLNOD	OPENFX	OPENTR	PROMPT	TIGCOP	TIGET
	TIGHDR	TISRCH	TLSRCH	TRNGET	TRMPUT	UNIQND							
	+APPEND												
UICMND	CHARFL	CHKEXS	CLOSFX	EQUALA	INSCHR	INSINT	INSLOG	INSPGM	INSRET	LENGA	LGLNOD	OPENFX	OPENTR
	PROMPT	SELCLS	TIGCAP	TIGET	TIGHDR	TIGNAM	TISRCH	TLSRCH	TRNGET	TRMPUT	UNIQND		
	+COMMOD												
UICRLG	CLOSFX	CLOSTR	CMGCDS	CMGLOG	CMGOPT	CMGOTH	DIGDC	DIGMDI	EQUALA	EVALBM	GCLIST	INSINT	INSPGM

OLPARS PROGRAM CROSS REFERENCE LISTING -- D
(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM_NAME

INSRET LIGCLP LIGCOP OPENFX OPENTR PROMPT SELCLS TIGCOP TIGET TIGNAM TRMPUT
+CREATLOG

UIDCMS INSPGM INSRET PROMPT TRMGET
+DSCRMEAS

UIGNOR EQUALA INSPGM INSRET PROMPT TRMGET TRMPUT
+NMVBSU NMLBSU NMMOD

UIL2FS CLOSFX CLOSTR CMGOTH ELIMEA INITD INSINT INSLOG INSPGM INSRET OPENFX PROMPT SETUP SLPAIR
TRMGET TRMPUT
+L2FSHP

UILASD CMGOTH DISTEU ELIMEA EQUALA ERASE GLOMOD INSFLT INSINT INSPGM INSRET LOGMN PROMPT SELCLS
TIGMN TRMGET TRMPUT
+L1ASDG L2ASDG

UILGE1 ERASE INSPGM INSRET PROMPT TRMGET TRMPUT
+LSETUP

UILGE2 INSPGM INSRET PROMPT TRMGET
+LSETUP

UILGE3 EQUALA ERASE INSFLT INSINT INSPGM INSRET PROMPT TIGCOP TRMGET TRMPUT TVGVEC
+SINGLE

UILGE4 EQUALA INSPGM INSRET PROMPT TRMGET
+CLFYMD CMPMOD SINGLE

UILGE5 EQUALA INSPGM INSRET PROMPT TRMGET
+CMPMOD

UILGE6 EQUALA INSPGM INSRET PROMPT TRMGET
+CMPMOD

UIMNBR CLOSFX EQUALA INSPGM INSRET LGLTRE OPENFX PROMPT TLSRCH TRMGET TRMPUT
+NRSTNBR

UIRSTR CHARFL CLOSFX CMGCDS CMGOPT CMGOTH DIGDC DIGHDI EQUALA ERASE INSINT INSPGM INSRET LENGA
LGLMOD OBTCLS OPENFX OPENTR PROMPT TIGCOP TIGET TIGHDR TIGNAM TRMGET TRMPUT TVGHDR TVPHDR
UNIQMD
+RESTRUCT

UIS2FS CLOSFX CMGOTH ELIMEA GLOMOD INITD INSINT INSPGM INSRET OPENFX OPENTR PROMPT SELCLS TIGET
TRMGET TRMPUT
+S2FSHP

UIXFRM EQUALA INSCHR INSPGM INSRET LGLTRE PROMPT TLSRCH TRMGET TRMPUT
+ESETUP MATXFRM MEASXFRM NORMXFRM TRANSFRM

UNION CDSCHK CMGCDS CMGOTH DIGROM DIGROI DIPROE DIPROM DIPROI DVGROE ERASE INSINI INSINT INSRET
MENU OEXIT OPENFX PROMPT RODISP RSORTA RSORTB TRMGET TRMPUT

UNIQMD EQUALA GMXTMD INSINT INSLOG INSPGM INSRET TIGCAP TIGNAM

(WHO CALLS WHOM, '+' MEANS 'CALLED BY' LINE INDICATED)

PROGRAM NAME

	+DDSUBSTR UIAPND UICMND UIRSTR
USRI01	EQUALA FLUSHT INSCHR INSINT INSLOG INSPGM INSRET LGLTRE OPENS PROMPT TLSRCH TRMGET TRMPUT +FILEIN
VALID	+ODTDMP
VALUEA	+EQUALA LLSRCH TLSRCH TYPE
VALUES	+CIP EQUALS HELP LESS
VCREAT	+OPENS
VOPEN	+OPENS
WRITEB	+CLOSBL FGET FLUSHB FPUT OCLOSE OEXIT
WRTNOW	DATE INSPGM INSRET TIME TRMPUT +CLFYMD CLSCAT CREVAL DRAWLOG DRAWTREE LISTLOGS LISTREES MICMAC OPTDSP RODISP +THRDSP
WRTODS	CMGCDS CMGOPT INSCHR INSPGM INSRET TEXT TRMPUT +MICMAC RODISP
XFORM	+MEASXFRM
ZOM1SP	CMGOTH DIPHDI DIPMAX DVPHDR EQUALA ERASE INSFLT INSINT INSPGM INSRET MICMAC PROMPT TRMGET TRMPUT +SCALZM
ZOM2SP	CLSCAT CMGOTH DIPHDI DIPMAX DVPHDR EQUALA ERASE GIN INSFLT INSPGM INSRET PROMPT RCTNGL TEXT TRMGET TRMPUT +SCALZM

INDEX

Addmcv 5
Adupcm 8
Alncpn 10
Anything 13
Append 14
Ascdec 16

Binwidth 18
Bound 21
Byeolp 23

Cdefault 24
Cdisplay 27
Cdschk 29
Charfl 31
Chkexs 32
Cip 34
Clnode 36
Closbl 38
Close 40
Closfx 41
Clostr 42
Clscat 43
Clus 46
Cmdisp 50
Cmgcds 52
Cmgdir 53
Cmglog 54
Cmgopt 55
Cmgoth 56
Cmgscn 58
Cminit 59
Cmncov 61
Cmpcds 63
Cmpdir 64
Cmplog 65
Cmpopt 66
Cmpoth 67
Cmprt 69
Cmpscn 70
Cntlsp 71
Colaps 74
Combin 76
Comnod 78
Concat 81
Copycv 83
Copymn 85
Copyvc 87
Crandts 89
Creafx 94

Creatlog 96
Creatr 98
Creval 100
Crlog1 102
Crlog2 104
Crlv1 106
Crlv2 108
Crproj 110
Cscale 112

Dbndy 116
Dcrim 118
Dcrim1 121
Dcrim2 124
Ddatanod 126
Ddatatree 128
Ddsubstr 130
Delete 134
Deletr 135
Digcme 137
Digcmh 139
Digdc 141
Digefs 142
Digent 143
Dighdi 145
Digmax 148
Dignam 149
Digroe 150
Digroh 151
Digroi 153
Dipcme 154
Dipcmh 156
Dipefs 158
Dipent 159
Diphdi 161
Dipmax 162
Dipnam 163
Diproe 164
Diproh 165
Diproi 167
Dispcl 168
Disteu 169
Distma 170
Distqc 172
Distva 173
Dither 174
Dlogtree 176
Dlregn 178
Dlsubstr 180
Drawbndy 184
Drawlog 189
Drawtree 195
Dscrmeas 197

Dscvth 201
Dsproj 204
Dtrename 207
Dvec 208
Dvghdr 211
Dvgroe 213
Dvgroh 214
Dvgvec 215
Dvphdr 217
Dvproe 218
Dvproh 219
Dvpvec 220

Eigenj 222
Eiglpv 223
Eignlg 225
Eignpv 228
Eignxfrm 230
Eigprt 232
Elifre 234
Elimea 235
Equala 237
Equals 239
Erase 241
Ev1sp 242
Ev2sp 245
Evalbm 248
Evalu8 250
Exfrm 256
Expand 259
Ext1st 261
Extlst 263

Fcgent 265
Fcghdr 266
Fcpent 267
Fcphdr 268
Fcreat 269
Fctopn 270
Fget 271
Filein 273
Fileout 277
Filget 279
Filput 281
Fish 282
Fisher 283
Fishmod 290
Fishsl 292, 295
Fishth 298
Fishvc 300
Fixklv 301
Flushb 303
Flushf 304

Flusht 304
Fopen 305
Fput 306
Fxltmp 308

Garbnd 310
Gclist 311
Gdstrc 312
Gen 316
Getfid 318
Gethst 319
Getlst 321
Getopt 323
Getpr 325
Getstr 326
Gin 327
Glonod 328
Glstrc 330
Gnliae 335
Gnlvae 337
Gnsmae 339
Gnxtan 341
Gnxtln 343
Gnxtnd 346
Goptnm 350
Gridis 351
Gtlun 352
Gtrgni 353
Gtrgnr 355

Hash 357
Helolp 359
Help 360
Hsghdr 363
Hsgrec 364
Hsphdr 366
Hsplnk 367
Hsprec 368

Idchk 370
Idx 372
Index 374
Initd 375
Inschr 377
Insdbl 378
Insflt 379
Insini 380
Insint 382
Inslog 383
Inspgm 384
Insret 386
Insset 387
Intensify 389

Invert 391
Itreal 393

Killnd 394

L1asdg 396
L1crdv 398
L1eigv 400
L2asdg 402
L2crdv 404
L2eigv 406
L2fshp 408
Lcproj 411
Lenga 414
Length 415
Less 416
Lglnod 418
Lgltre 420
Ligcap 422
Ligclp 424
Ligcnm 426
Ligcop 427
Ligdcn 429
Ligdsn 430
Ligent 431
Liglog 433
Ligprb 435
Ligstr 436
Linseg 438
Lipcap 439
Lipclp 441
Lipcnm 442
Lipcop 443
Lipdcn 445
Lipdsn 446
Lipent 447
Liplog 449
Lipprb 451
Lipstr 452
Listlogs 454
Listrees 456
Llgent 457
Llghdr 459
Llpent 460
Llphdr 462
Llsrch 463
Lnproj 466
Logeval 469
Logmn 478
Logmnc 480
Ltrename 482
Lvgent 483
Lvghdr 484

Lvglnk 485
Lvpent 486
Lvphdr 487
Lvplnk 488

Macrop 489
Maketree 492
Makopt 498
Mark 500
Matrix 501
Matxfrm 503
Mean1 505
Mean2 507
Measxfrm 509
Menu 513
Micmac 515
Microp 518
Moddfs 521
Modini 524
Move 527
Movec 529
Mtrxde 536
Mtrxdp 538
Mtrxen 541
Mtrxli 544
Mxfrm 545

Namelog 547
Nblank 550
Nmdist 551
Nmeval 554
Nmv 555
Nmvbsu 560
Nmvmod 564
Nmvopt 567
Nmvrjt 569
Nnbrev 572
Nnlbsu 575
Nnmean 577
Nnmncv 579
Nnmod 581
Nnpass 584
Nnsort 587
Nodcom 590
Normxfrm 594
Nrstnbr 596
Nxfrm 599

Obfgen 602
Obtcls 605
Oclose 608
Ocreat 609
Odelet 611

Oexit 612
Omove 614
Oopen 615
Openbl 617
Openfx 619
Opens 621
Opentr 623
Opntmp 625
Optdsp 627
Optimlmod 629
Orenam 631

Packb 633
Packw 634
Pairlg 635
Penmv 638
Pepair 641
Precpt 644
Printr 648
Projct 649
Projmn 650
Prompt 652
Prtcm 654
Prtds 655
Prtidx 657
Prtlog 661
Psort 663
Ptrgni 665
Ptrgnr 667
Puthst 669
Pvgent 670
Pvghdr 671
Pvgnam 672
Pvpent 673
Pvphdr 674
Pvpnam 676
Pweval 677

Rank 678
Rctngl 680
Rdisplay 682
Readb 684
Rearr 686
Reasname 690
Redraw 692
Redvec 694
Relun 695
Rename 696
Renamt 698
Reproject 701
Restruct 703
Rodcls 707
Rodisp 709

Rsorta 711
Rsortd 713
Rxint 715
Ryint 718

S1crdv 721
S1eigv 723
S1gbc 725
S1ghdr 726
S1pbc 727
S1phdr 728
S2crdv 729
S2eigv 731
S2fshp 733
Savmat 735
Sbupmc 737
Scalret 738
Scalzm 740
Scat 742
Scatr 746
Search 748
Selcls 750
Select 753
Setds 755
Setlog 757
Setopt 759
Setup 761
Slctmeas 765
Slteig 767
Smgent 769
Smghdr 770
Smglnk 771
Smgmds 772
Smpent 773
Smphdr 774
Smplnk 775
Smpmds 776
Sort 777
Su1sp 778
Su2sp 780
Submnc 782
Sufish 784
Summcm 786
Sunmv 787

Text 789
Thrdsp 790
Threshmod 792
Tigcap 794
Tigcop 795
Tigcov 797
Tiget 798
Tighdr 799

Tigmn 800
Tignam 802
Tigvar 803
Tipcap 804
Tipcop 805
Tipcov 806
Tipet 807
Tiphdr 808
Tipmn 810
Tipnam 811
Tisrch 812
Tlgent 814
Tlghdr 815
Tlpent 816
Tlphdr 817
Tlsrch 818
Tmpxrd 821
Tmpxwt 823
Transfrm 825
Trmget 827
Trmopn 831
Trmput 832
Tvgcls 836
Tvghdr 837
Tvglog 838
Tvgvec 839
Tvpcls 841
Tvphdr 842
Tvplog 843
Tvpvec 844
Type 846

Uignor 847
Uixfrm 850
Union 852
Uniqnd 853

Valuea 856
Values 857
Vcreat 858
Vopen 859

Writeb 860
Wrtnow 862
Wrtods 863

Xint 865

Yint 868

Zom1sp 871
Zom2sp 874