

AD-A099 125

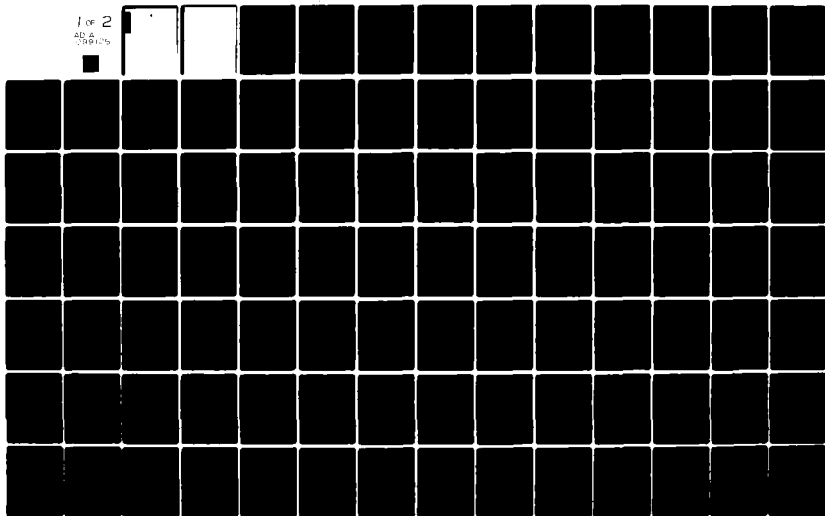
DAVID W TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CE--ETC F/0 5/2
PAGED GIRS (GRAPH INFORMATION RETRIEVAL SYSTEM) USERS MANUAL.(U)
MAY 81 I S ZARITSKY

UNCLASSIFIED

DTNSROC-81/024

NL

1 of 2
AD-A
009125



AD A099125

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM										
1. REPORT NUMBER DTNSRDC-81/024	2. GOVT ACCESSION NO. AD-A099125	3. RECIPIENT'S CATALOG NUMBER										
4. TITLE (and Subtitle) PAGED GIRS (GRAPH INFORMATION RETRIEVAL SYSTEM) USERS MANUAL		5. TYPE OF REPORT & PERIOD COVERED Final										
7. AUTHOR(s) Irving S. Zaritsky		6. PERFORMING ORG. REPORT NUMBER										
9. PERFORMING ORGANIZATION NAME AND ADDRESS David W. Taylor Naval Ship Research and Development Center Bethesda, Maryland 20084		8. CONTRACT OR GRANT NUMBER(s) F43411										
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Sea Systems Command Washington, D.C. 20362		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS (See reverse side)										
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 12F-1152		12. REPORT DATE May 1981										
		13. NUMBER OF PAGES 156										
		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) <table border="0"> <tr> <td>Associative Memory</td> <td>Hashed Addressing</td> </tr> <tr> <td>Data Base</td> <td>Information Retrieval System</td> </tr> <tr> <td>Data Definition Language</td> <td>Data Base Management System</td> </tr> <tr> <td>Data Management</td> <td>Paging Schemes</td> </tr> <tr> <td>Graph</td> <td></td> </tr> </table>			Associative Memory	Hashed Addressing	Data Base	Information Retrieval System	Data Definition Language	Data Base Management System	Data Management	Paging Schemes	Graph	
Associative Memory	Hashed Addressing											
Data Base	Information Retrieval System											
Data Definition Language	Data Base Management System											
Data Management	Paging Schemes											
Graph												
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The report describes the use of the paged version of an associative (content addressable) computer memory simulation called GIRS (Graph Information Retrieval System). GIRS provides a convenient and efficient technique for the dynamic insertion, retrieval, modification, and deletion of data in a data base. Pointer manipulation is convenient and paged GIRS is well adapted for concurrent operation on more than one graph and therefore will (Continued on reverse side)												

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

381062

J-4

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(Block 10)

Program Element 62543N
 Project F43411
 Task Area ZF 43411001
 Work Unit 1808-009

(Block 20 continued)

handle shared and distributed data bases. Users of a large data base could have their own unique description of common data which might be stored elsewhere. The paged version of GIRS allows for a wide range of flexibilities, in which, at a minimum, a user may leave many parameters to default. For maximum flexibility, however, a user may include a user-embedded strategy and hence may satisfy queries of various degrees of imprecision depending on the inferential search technique used.

The implementation described here is in FORTRAN for the PDP 11/45 computer system and is described in sufficient detail to allow conversion to another computer or alteration of the existing overlay structure.

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

TABLE OF CONTENTS

	Page
LIST OF TABLES.	v
ABSTRACT.	1
ADMINISTRATIVE INFORMATION.	1
INTRODUCTION.	1
BRIEF DESCRIPTION.	2
MOTIVATION	3
MEMORY SCHEME	4
THE GIRS BUFFER.	4
THE CONTINUANT	4
THE DIRECTORY.	6
THE BUFFER COMPOSITION	7
INITIALIZATION OF THE GIRS BUFFER.	7
COMMON LVBUFR.	8
REPRESENTATION OF NODES AND LINKS.	9
PAGE AND CONTINUANT DETERMINATION FOR THE TRIPLE	10
Page Determination.	10
Continuant Determination.	10
THE FLAG FIELD	11
DISK FORMAT	13
PAGING SCHEME	15
GENERAL DISCUSSION	15
I/O FOR THE DIRECTORIES OF DISK LOCATIONS OF CONTINUANTS	15
I/O FOR THE CONTINUANTS.	16
PHILOSOPHY	17
USER-CALLABLE GIRS SUBROUTINES.	18
INITIALIZATION	18
Getting Started	18
Subroutine LVSETP	22
Subroutine LVGRN.	24
RETRIEVAL OF VALUES.	25
Discussion.	25
Subroutine LVFDEX	26

	Page
RETRIEVAL OF MVL INDEX OF GIVEN VALUE OF A FUNCTION (INCLUSION)	30
Subroutine LVINCL	30
INSERTION.	31
Discussion	31
Subroutine LVINEX	32
Subroutine LVREOR	38
DELETION	39
Discussion.	39
Subroutine LVDLEX	40
DISK STORAGE AND RETRIEVAL OF A GRAPH.	43
Discussion.	43
Subroutine LVDUMP	44
Subroutine LVFECH	46
EXECUTING A GIRS PROGRAM.	47
GENERAL DISCUSSION	47
INDIRECT USE OF A GIRS SUBROUTINE VIA GIRL	47
Preprocessing and Compiling of GIRL/FORTRAN Program	49
DIRECT USE OF GIRS SUBROUTINES	50
Linking and Executing a GIRL/FORTRAN Program.	50
OVERLAY STRUCTURE	52
LIMITATIONS AND MEMORY REQUIREMENTS	54
ADDING A USER-EMBEDDED STRATEGY	56
INTRODUCTION	56
USE.	57
PROPOSED EXTENSIONS	57
ACKNOWLEDGMENTS	57
APPENDIX A - VARIABLES IN LABELED COMMON.	59
APPENDIX B - SUBROUTINE CALLING STRUCTURE	61
APPENDIX C - SUBROUTINE LISTINGS.	65
REFERENCES.	149

LIST OF TABLES

	Page
1 - The Continuant Header	5
2 - The Flag Field.	12
3 - The Disk Format	14
4 - Overlay Region Sizes.	52
5 - The Overlay Structure	53

ABSTRACT

This report describes the use of the paged version of an associative (content addressable) computer memory simulation called GIRS (Graph Information Retrieval System). GIRS provides a convenient and efficient technique for the dynamic insertion, retrieval, modification, and deletion of data in a data base. Pointer manipulation is convenient and paged GIRS is well adapted for concurrent operation on more than one graph and therefore will handle shared and distributed data bases. Users of a large data base could have their own unique description of common data which might be stored elsewhere. The paged version of GIRS allows for a wide range of flexibilities, in which, at a minimum, a user may leave many parameters to default. For maximum flexibility, however, a user may include a user-embedded strategy and hence may satisfy queries of various degrees of imprecision depending on the inferential search technique used.

The implementation described here is in FORTRAN for the PDP 11/45 computer system and is described in sufficient detail to allow conversion to another computer or alteration of the existing overlay structure.

ADMINISTRATIVE INFORMATION

This work was completed in the Computer Science Division of the Computation, Mathematics, and Logistics Department under the sponsorship of NAVSEA 03F, Task Area ZF 43411001, Work Unit 1808-009.

INTRODUCTION

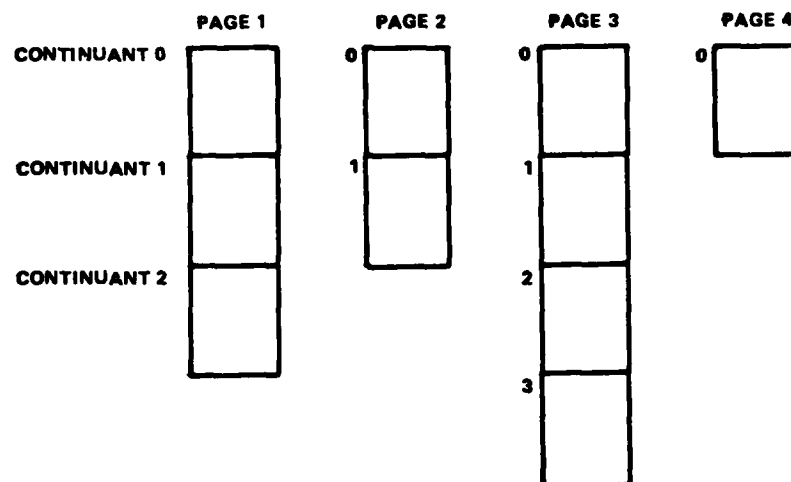
This report describes the use, and to a lesser degree, the implementation of the paged version of an associative (content addressable) computer memory simulation called GIRS (Graph Information Retrieval System). GIRS provides a convenient and efficient technique for the dynamic insertion, retrieval, modification, and deletion of data in a data base. Pointer manipulation is convenient and paged GIRS is well adapted to handle shared and distributed data bases. The flexibility of the paged version of GIRS allows the user the option of leaving many parameters to default. For maximum flexibility, however, a user may include a user-embedded strategy and hence may satisfy queries of various degrees of imprecision depending on the inferential search technique used.

The implementation described here is in FORTRAN for the PDP 11/45 computer system and is in sufficient detail to allow conversion to another computer or alteration of the existing overlay structure.

Paged or Out-Core GIRS is an extension of an existing version of GIRS as described by Zaritsky.^{1*} The original version will be referred to in this report as "In-Core" GIRS. Out-Core GIRS is an implementation based on a report by Berkowitz,² "Design Trade-Offs For A Software Associative Memory." Some familiarity with In-Core GIRS is assumed and it is also assumed that the reader has access to copies of both prior reports.

BRIEF DESCRIPTION

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



*A complete listing of references is given on page 149.

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

If a user chooses a buffer size which holds only one continuant and requests just one page, then the system is similar to in-core GIRS except for the capability of automatic overflow to a new continuant.

MOTIVATION

The paged version of GIRS has several advantages over in-core GIRS and other data manipulation facilities:

1. Its large data storage capability (see the section entitled "Limitations and Memory Requirements").

2. Concurrent operation on more than one graph. Paged GIRS is ideal for shared and distributed data bases. Each user of a large data base might be assigned his or her own page to uniquely describe common data which might be stored elsewhere. An example of this type of application is described by Zaritsky.³

3. Its capability for a user-embedded strategy, which allows for the inclusion of operations such as an inferential search to handle imprecise queries. This capability is described in the section on "Adding a User-Embedded Strategy."

In the near future, a paged hardware associative memory⁴ will be merged with out-core GIRS. The result will be an enhanced system with high speed relational processing.

MEMORY SCHEME

THE GIRS BUFFER

The GIRS buffer consists of four fields, represented by the four arrays, NODSPC, LSTSPC, LNKSPC, and FLGSPC, from commons LVVTR1, LVVTR2, LVVTR3, and LVVTR4, respectively, as was the case with the in-core version of GIRS. The buffer contains both continuants and a directory for locating the continuants residing in the buffer. The number of continuants that may reside in the buffer is unrestricted. It is fixed by the user in variable LVNCOR in labeled common LVBUFR. The buffer location immediately preceding the beginning of each continuant is called the control point (CP) and the directory is located at CP0.

THE CONTINUANT

Although each continuant requires an equal length of NODSPC, LSTSPC, LNKSPC, and FLGSPC,* each field is composed of three "spaces": a "working space" and an "available space," as was the case with in-core GIRS, and a space for the header to describe the state of the continuant. The header takes up eight cells,** two in each of the four fields. The header is described in Table 1; the variables (from labeled common LVHDVL) in brackets indicate the relative location from the beginning of the continuant.

*It is convenient to refer to the length of any one of these arrays, instead of all four, as being the continuant size. The continuant size requested by the user must be a multiple of 64 (up to 960 on default), but the actual usable continuant size is always two less than $n*64$ to account for the header.

**On the PDP 11, a cell takes up the space of one word.

TABLE 1 - THE CONTINUANT HEADER

NODSPC a) The relative Mass Storage Address (MSA) of this continuant [THSMSA] b) The REGISTER of Available SPace (REGASP) for this continuant [REGAS]
LSTSPC a) This continuant's page number [PAGENO] b) This continuant's relative position within the page [CONTNO]
LNKSPC a) The size of this continuant's "working space" (number of insertions less deletions) [INSDEL] b) The number of times this continuant has been accessed since it was last brought into main memory [USECT]
FLGSPC a) [HDRFLG] The continuant descriptor flags, if on, indicate the following: 1) LVWRIT - The continuant has been modified since it was brought into the buffer and therefore must be written out to disk when either another continuant is brought into the same segment (control point) of the buffer or when the file is closed. 2) LVNUSE - The continuant has not yet been used. Either it has just been created or it has been brought into the buffer and not yet accessed. This flag is turned off when the continuant is accessed. b) An indicator of how recently this continuant was brought into the buffer [READVL]

If a user wishes to access information described in Table 1 and if the desired continuant is the current continuant of the current page, then the user may do so by adding the bracketed label to LVCTRL and using that quantity as an index to the appropriate buffer array. For example, "Transfer to statement 10 if the continuant read into the buffer most recently (making it the current continuant of the current page) has not yet been accessed:"

IF((FLGSPC(LVCTRL + HDRFLG) .AND. LVNUSE) .NE. 0) GO TO 10.

THE DIRECTORY

The directory is a continuant without a header. It is never taken out of the buffer and is located at the beginning of the buffer. Its size (within each of the four fields of the buffer) is determined by the number of continuants residing in the buffer (LVNCOR) and is calculated as follows:

$$LVDRSZ = 64 * ((LVNCOR/64) + 1)$$

Each control point is stored as the sink node (integer value) of a triple where the source node is the "continuant number + 1" and the link is the page number. The directory address for the triple is determined as follows:

$$LOC = (Page\ No. + Cont.\ No + 1) \text{ Mod } (LVDRSZ)$$

THE BUFFER COMPOSITION

A typical buffer may be illustrated as follows:

Location	Length	Contents			
		NODSPC	LSTSPC	LNKSPC	FLGSPC
1	LVDRSZ	Directory			
CP1	LVHDRS	Continuant Header			
CP2	LVVSZE	Body of Continuant			
		Continuant Header			
		Body of Continuant			
		.			
		.			
CPn		.			
	LVPGHD	Continuant Header			
		Body of Continuant			

where: CP1 = LVDRSZ
 CP2 = LVDRSZ + LVHDRS + LVVSZE
 CPn = LVDRSZ + (n-1) * LVPGHD
 LVHDRS = 2
 LVPGHD = LVHDRS + LVVSZE
 LVVSZE = continuant size as defined by the user

INITIALIZATION OF THE GIRS BUFFER

As for In-Core GIRS, the buffer is initialized by calling subroutine LVSETP. The available space (AS) ring is identical to that in the In-Core version except that as many copies of it will be placed in the buffer as there are continuants residing in the buffer.

The values in the continuant header as described in Table 1 are initialized as follows:

NODSPC

- a) The relative block address of the continuant on the disk file as computed by LVSETP
- b) REGASP = 1

LSTSPC

- a) The page number of the continuant
- b) Continuant numbers as assigned in order of their creation, beginning with zero

LNKSPC

- a) The number of spaces in the continuant which have been removed from AS = 0
- b) Access count = 0

FLGSPC

- a) Continuant descriptor flags = 0
- b) Continuant I/O history set to the current value of LVRcnt from common LVREGS

Subroutine LVSETP initializes all the continuants requested through array LVSTAK and upon completion brings page one, continuant zero, back into the buffer.

COMMON LVBUFR

Common LVBUFR contains all the variables for determining of the memory buffer size and the continuant locations on disk. The order in which the variables are listed here does not necessarily match the actual order as shown in Appendix A. The internal names for each variable are noted in brackets.

LVVSZE - Single array length[†] of the continuant

$$[PAGSZE] = (n*64) - 2 \quad \text{where } 1 \leq n \leq 8$$

LVHDRS - Single array length of the header

$$[HDRSZE] = 2$$

LVPGHD - Single array length of the combined continuant and header

$$[PAGHDR] = PAGSZE + HDRSZE = n*64 \text{ where } 1 \leq n \leq 8$$

[†]The four arrays--NODSPC, LSTSPC, LNKSPC, and FLGSPC--are of equal length.

LVBKSZ - Number of blocks (256 words each) required to hold one continuant on disk
 $[BLKSZE] = 4 * PAGHDR / 256$
 LVPGH4 - Total length, in words, of one continuant
 $[PAGHD4] = 4 * PAGHDR$
 LVNCOR - Number of continuants which reside in the in-core buffer [INCORE]
 LVDRSZ - Single array length of the in-core directory. It must be a multiple of 64
 $[DIRSIZE] = 64 * ((INCORE / 64) + 1)$
 LVBFSZ - Total single array length of the in-core buffer. NODSPC, LSTSPC, LNKSPC, and FLGSPC are all dimensioned to this value
 $[BUFSZE] = DIRSIZE + (INCORE * PAGHDR)$
 LVDRBK - Number of blocks required to hold the in-core directory on disk
 $[DIRBLK] = 4 * DIRSIZE / 256$
 LVMSAD - Location on disk (relative block number) of the in-core directory
 $[MSADIR] = 2$

REPRESENTATION OF NODES AND LINKS

Before nodes and links may be used in a graph, they must be assigned to a page and given a random number which is unique to that page. This is accomplished by calling subroutine LVGRN. Page numbers can be either specifically requested by the user ($1 \leq LVREQP(1) \leq 63$) or assigned by LVGRN ($LVREQP(1) = -1$) to a new page. The random number returned is in the range of 1 to LVVSZE (the continuant size) and the same sequence of random numbers is repeated for each page. The total number of nodes and links which the user may define for any one page may not exceed LVVSZE or the program will terminate. Unless the default values are modified, nodes and links have the following form:

15	10	9	0
Page No.		Random Number	

PAGE AND CONTINUANT DETERMINATION FOR THE TRIPLE

Page Determination

Subroutine LVINEX determines the page (and continuant) on which a triple is placed. The information needed for page placement is extracted from the source node at the time of insertion. If the source node is fully defined (from subroutine LVGRN), the prefix determines page placement. If the source node is not fully defined, it is expected to have one of the following values:

- 1, Place the triple on a new page
- 0, Place the triple on the current page
- $1 \leq n \leq 63$, Place the triple on page n

In all these cases, LVINEX will call LVGRN to fully define the source node.

A request to place the triple on a new page is a special case. Two variables are used to compute a page number. LVHAPG, from common LVREGS, is an internal counter which keeps track of the highest page number in which there has been an insertion or for which a random number has been generated. LVHREQ, also from common LVREGS, is set by the user at the beginning of the program. This variable defines the number of pages created prior to execution of the program. During the course of execution, if LVHAPG(1) exceeds LVHREQ, continuant zero of a new page is created and LVHAPG(1) is incremented by one.

Continuant Determination

Before an insertion, deletion, or retrieval may take place, the particular continuant must be determined. If the user does not specify a continuant, all the continuants of the requested page will be examined in sequential order until either the requested function is found or the set of continuants for that page is exhausted. If the function does not exist, the triple is placed on the (sequentially) first continuant of the requested page which has available space. The continuant request is made with variable LVREQP(2) from labeled common LVREGS, which may take the following values:

- $0 \leq n \leq 63$, Continuant n is requested
- 1, New continuant is requested
- 2, Continuant is unspecified (default)
- 3, Current continuant if requested page is current page

If a value is to be added to a list that has been specifically placed on a particular continuant, but a different continuant is specifically requested, subroutine LVREOR reports an error. However, the insertion proceeds with the entire list moved onto the newly requested continuant.

With judicious use of subroutine LVREOR, two continuants may be MERGED² and also a list may be SEPARATED² from one continuant and placed on another.

THE FLAG FIELD

The flag field, contained in FLGSPC in Common LVVTR4, consists of eleven one-bit flags and two two-bit flags:

FLGSPC

14	13	12	11	10	9-8	0	1	2	3	4	5	6-7
----	----	----	----	----	-----	---	---	---	---	---	---	-----

Each flag describes a different aspect of the contents of the associated location in the buffer (Table 2). The first seven flags are the same as those for in-core GIRS.

TABLE 2 - THE FLAG FIELD

FLAG	FLAG VALUE	CONTENTS OF ASSOCIATED LOCATION
Flag 0	2^7	Head of a multivalued list.
Flag 1	2^6	Location already occupied.
Flag 2	2^5	A value on a multivalued list.
Flag 3	2^4	A node or link value. Does not refer to the actual contents of the location. Rather, the location value itself is used as a random number to define either a node or a link.
Flag 4	2^3	Head of a multivalued list which has been modified either by an insertion or by an indexed deletion, thus bypassing the "saved index" upon retrieval feature. (See the description of Subroutine LVFNV for further details.)
Flag 5	2^2	Head of a conflict list.
Flag 6-7	$2^1 + 2^0$	Type of value contained in the location: 00 Random number 01 Numeric data 10 Continuing string of Hollerith data 11 The only, or final, cell in a Hollerith data string
Flag 8-9	$2^8 + 2^9$	Type of triple contained in the location: 00 NODE LINK value 01 NODE value NODE 10 value LINK NODE
Flag 10	2^{10}	MVL backward continuation flag. This continuant does not contain the beginning of the list. A portion of this function resides on a lower-numbered continuant.
Flag 11	2^{11}	List forward continuation flag. This continuant does not contain the end of the list. A portion of this function resides on a higher-numbered continuant.
Flag 12	2^{12}	Inhibit reorganization of this list onto another continuant.
Flag 13	2^{13}	Head of a list which is a non-movable continuation of a list on some other continuant.
Flag 14	2^{14}	Pointer to sequence space.

DISK FORMAT

A saved GIRS file contains (in sequence) the following information: System values from the labeled commons, up to 228 user identifiers from labeled common LVUSER, variables for generating or continuing the random number sequence for up to 64 different pages, a copy of the directory of the continuants residing in the buffer when the program terminated, a directory containing the disk locations (relative to the beginning of the file) of all the continuants in the system, and copies of all of the continuants in the system.

The continuants are sequentially placed onto the file in the order of their creation. At the beginning of a "creation" type program, empty copies of all requested continuants are placed onto the file in sequence of increasing pages and continuants. After that, continuants are placed onto the file as they are created.

Also, at the beginning of a creation type program, sixteen blocks are allocated for the "out-core" directory. Each block holds the relative locations for the continuants of four pages. Otherwise, the block contains zeros.

Table 3 describes the disk format for a GIRS file which has been saved.

TABLE 3 - THE DISK FORMAT

Relative Location (in blocks) [†]	Size (in blocks) [†]	Contents
0	1	GIRS system variables from labeled commons LVREGS, LVRAND, LVBUFR, and LVVSEQ. Also, up to 228 user identifiers from labeled common LVUSER.
1	1	LVNTBL (256) from labeled common LVRAND.
2	LVDRBK	Directory of continuants residing in the buffer. LVDRBK = ((LVNCOR/64)+1)/64
LVDRBK+2	1	Directory containing the locations (relative to the beginning of the file) of all continuants from pages 1-4.
.		
.		
.		
LVDRBK+2+n	1	Out-core directory for all continuants from pages n*4+1 to n*4+4 where 0≤n≤15.
LVDRBK+18	LVBKSZ	Page 1, Continuant 0 LVBKSZ = LVVSZE/64 (the continuant size, LVVSZE, must be a multiple of 64)
LVDRBK+18 +LVBKSZ	LVDKSZ	Page 1, Continuant 1 or Page 2, Continuant 0; continuants are placed sequentially as they are created.
LVDRBK+18 +n*LVBKSZ	LVDKSZ	nth continuant to be placed onto the file.
[†] Each block contains 256 words.		

PAGING SCHEME

GENERAL DISCUSSION

All the general I/O for out-core GIRS is handled on the PDP-11 computer by two RT-11 System Subroutine Library routines: IREADW and IWRITW. These two routines operate in a block-oriented, random access, unformatted mode. They are called by four GIRS routines: LVPAGR and LVPAGW, to read in and write out the continuants; and LVDRRD and LVDRWR, to read in and write out any of the sixteen directories of continuant locations on disk. The I/O channels are initialized when GIRS subroutine LVSETP calls RT-11 System Subroutine Library functions: ICSI, IGETC, IFETCH, IENTER, and LOOKUP. The new channel is closed when GIRS subroutine LVDUMP calls subroutine LVCLOS which in turn calls System Subroutine Library routine CLOSEC. Since only six GIRS subroutines interact with the RT-11 System Subroutine Library, the I/O functions are relatively isolated. This leaves an otherwise portable all FORTRAN package.

I/O FOR THE DIRECTORIES OF DISK LOCATIONS OF CONTINUANTS

There are sixteen out-core directories with enough space to locate up to 64 continuants for each of 64 pages[†] on disk. Each directory has 256 words (one block) to locate the continuants for four consecutive pages: 1-4, 5-8, . . . etc. The directories are located on relative locations LVDRBK+2 through LVDRBK+17 of the disk file. Only one directory at a time is stored in main memory in array LVOTDR(256) in labeled common LVREGS. To find the desired location within the directory three variables also from labeled common LVREGS are needed: LVDRPG, LVDIRC, and LVOTLC. LVDRPG contains the current directory number as determined by the last requested page.

$$\text{LVDRPG} = (\text{Page No.} - 1) / 4 + 1$$

$$(\text{value range} = 1-16)$$

LVDIRC determines the quadrant number within the directory for the requested page.

$$\text{LVDIRC} = \text{Page No.} - 4 * (\text{LVDRPG} - 1)$$

$$(\text{value range} = 1-4)$$

LVOTLC is the position within the directory of the disk location for the requested page and continuant number.

$$\text{LVOTLC} = 1 + 64 * (\text{LVDIRC} - 1) + \text{Cont. No.}$$

$$(\text{value range} = 1-256)$$

[†] Practical considerations limit the number of pages to a maximum of 63, not 64.

I/O FOR THE CONTINUANTS

I/O for the continuants is controlled by an I/O executive routine, LVEXCH. LVEXCH takes as input a requested page and continuant number, LVREQP(1) and LVREQP(2) from labeled common LVREGS, and either confirms its current residency in the GIRS buffer or brings it into the buffer. In either case, the current page (LVCUPG(1)) and LVCUPG(2) from labeled common LVREGS) is updated to the requested page.

The general flow of LVEXCH is as follows:

- 1) Call LVDRCT to search the in-core directory and determine whether the requested page and continuant (REQ(P,C)) are in the buffer. If so, update the "current page" register and return.
- 2) Call LVMSA to bring into main memory the correct "Directory of Continuant Locations on Disk" if necessary and then determine whether the continuant exists and if so, its location on disk.
- 3) Call LVOPEN to make a control point ("continuant block") available in the GIRS buffer. If the buffer contains more than one continuant block, call LVALUE to determine (using the continuant header values) which in-core continuant is of least value. If the current continuant has been modified since it was brought into the buffer, write it out to disk. (The algorithm used for this determination is discussed in the next section.)
- 4) Call LVPAGR to bring the requested page into the GIRS buffer.
- 5) Update the "current-page" register (LVCUPG()).
- 6) Call LVRPLC to update the In-Core directory.
- 7) Call LVSUM to update the new continuant header and then return.

PHILOSOPHY

The philosophy used by Out-Core GIRS for bringing in continuants is generally known as "demand paging," that is, a continuant is brought into the buffer only when it is specifically requested. However, any continuant presently residing in the buffer must be saved before it is written over if it has been modified by an insertion or deletion. Furthermore, if the buffer holds more than one "continuant block," a specific continuant must be selected for removal.

Subroutine LVALUE contains the formula[†] used for this purpose. It is a modification of an optimization formula designed for the Control Data Corporation (CDC) Interactive Graphics Data Handler.⁶

Each continuant has a desirability value computed from values stored in the header of that continuant. The continuant with the lowest desirability value is either written over or written out to disk, of course. The formula is:

$$\text{value} = A * \text{order} + B * \text{usage} + C * \text{space} + D * \text{write}$$

where the weighting factors A, B, C, and D sum to 100. The weighting factors are set as follows: A = 15.0, B = 20.0, C = 15.0, and D = 50.0. Order is a measure of how long the continuant has been in core. Continuants most recently read in are weighted more heavily. Usage is the ratio of the use count for an individual continuant to the total usage for all the continuants in the buffer at the time of the computation. Usage is defined as the sum of all calls to subroutines LVINEX, LVFDEX, and LVDLEX which reference a particular continuant from the time that continuant was read into the buffer. Space refers to the fill ratio of the continuant. The emphasis of the fill ratio varies with the type of computer run. For a creation type run, a half-filled continuant is emphasized and for a production type run, a 5/8 to 7/8 filled continuant is emphasized and an empty continuant is deemphasized. The write parameter greatly emphasizes a continuant which has been modified because of the immediate 50 percent savings in disk I/O if the present continuant does not have to be written out to disk prior to reading in the requested continuant.

[†]This formula was devised by Mr. M. Haas, formerly with CDC and with DTNSRDC.

USER-CALLABLE GIRS SUBROUTINES

INITIALIZATION

Getting Started

To execute a GIRS program, the following labeled commons and declarations should be included in the driving program:

REAL*4 DEFEXT,LVCORE

LOGICAL*1 LVSTP,LVSNGL,LVNXTR,LVIN1,LVIN2,LVFD1,
LVFD2,LVDL1,LVDL2,LVIN3,LVFD3,LVDL3,LVDMP,
LVFD4,LVDL4,LVIN4,LVCRNT

COMMON /LVARGS/ LVFUNC,LVVARG,LVPOS,LVVTYP,LVVAL,LVVNVL,LVSKIP,

1 LVVTR,LVVINC,LVNDXN,LVVALS(10),LVTYPE(10)

2 ,LVSRSF,LVLNSF,LVSNSF,LVNTYP

COMMON /LVVSEQ/ LVSIZE,LVSEQ1,LVSEQ2,SEQSPC(1)

COMMON /LVRAND/ LVKPRM,LVKS,LVKY,LVKDY,LVKDX,LVTEMP,LVLIST,LVNTBL(256)

COMMON /LVVTR1/ NODSPC(buffer size)

1 /LVVTR2/ LSTSPC(buffer size)

2 /LVVTR3/ LNKSPC(buffer size)

3 /LVVTR4/ FLGSPC(buffer size)

COMMON /LVCRNT/ LVVGSP,LVCTRL,LVCTR1,LVLSTV,LVNFRE,LVFREE,

1 LVDREG,LVVMISA,LVPGLC,LVCRNT

COMMON /LVBUFR/ LVVSZE,LVNWCH,LVOLCH,LVCMPR,LVPGHD,LVBFSZ

1 LVDRSZ,LVNCOR,LVHDSR,LVMSAD,

2 LVSFSZ,LVBKSZ,LVDRBK,LVPGH4

COMMON /LVREGS/ LVCUPG(4),LVREQP(4),LVLVPG(4),LVMSAR,

1 LVHRPG,LVNMSA,LVHAPG(2),LVRcnt,LVUCNT,LVDRPG,

2 LVDIRC,LVOTLC,LVOTDR(256),

3 LVRWBF(4*continuant size)

COMMON /LVPRAM/ LVBFLC,LVLNTH,LVVERR,LVERNO,LVBNRY,LVBCD,LVMODE,LVPGS,LVLUN

COMMON /LVRUN/ LVRNTP,LVCORE

COMMON /LVSTAK/ LVLEVL,LVNVAR,LVSTAK(140)

```

COMMON /LVMASK/ LVWRIT,LVNUSE,LVNWCN,LVMSK3,LVMSSF,LVMSPF
COMMON /LVSWIT/ LVSTP,LVSNGL,LVNXTR,LVIN1,LVIN2,LVFD1,
1          LVFD2,LVDL1,LVDL2,LVIN3,LVFD3,LVDL3,LVDMP,
2          LVFD4,LVDL4,LVIN4
COMMON /LVUSER/ USER(228)
COMMON /LVUTIL/ FILSPC(39),DEFEXT(2)

```

Note:

1) A user may place up to 228 identifiers in Common LVUSER. These identifiers will automatically be placed on disk if a file is created.

2) If the "swap USR" function* of the RT-11 operating system for the PDP-11 computer is on (default), then COMMON /LVUTIL/ should be placed at the end of the set of labeled commons to prevent its being swapped out of main memory. If this labeled common is swapped out of main memory, the operating system as well as the program will go down as soon as the input and output file names are read in. If this common block is placed at the end and the system still goes down, either "SET USR NOSWAP" or try placing a dummy array in front of the common.

The following declaration and labeled commons should be included in all sub-routines in which there are GIRS operations:

```

LOGICAL*1 LVSTP,LVSNGL,LVNXTR,LVIN1,LVIN2,LVFD1,
1          LVFD2,LVDL1,LVDL2,LVIN3,LVFD3,LVDL3,LVDMP,
2          LVFD4,LVDL4,LVIN4,LVCRNT
COMMON /LVARGS/ LVFUNC,LVVARG,LVVP0S,LVVTYP,LVVAL,LVVNVL,LVSKIP,
1          LVVTR,LVVINC,LVNDXN,LVVALS(10),LVTYPE(10),
2          LVSRSF,LVLNSF,LVSNSF,LVNTYP
COMMON /LVREGS/ LVCUPG(4),LVREQP(4),LVLVPG(4),LVMSAR,
1          LVHRPG,LVNMSA,LVHAPG(2),LVRcnt,LVUCNT,LVDRPG,
2          LVDIRC,LVOTLC,LVOTDR(256),LVRWBF(512)

```

*The "swap USR" function will swap out of main memory the first 2000 words of a user's program in order to bring in RT-11 system routines.

```
COMMON /LVSWIT/ LVSTP,LVSNGL,LVNXTR,LVIN1,LVIN2,LVFD1,
1           LVFD2,LVDL1,LVDL2,LVIN3,LVFD3,LVDL3,LVDMP,
2           LVFD4,LVDL4,LVIN4
```

If Subroutine LVDUMP is to be called from a subroutine, the following labeled COMMON is needed:

```
COMMON /LVPRAM/ LVBFLC,LVLNTH,LVVERR,LVERNO,LVBNRY,LVBCD,
1           LVMODE,LVPGS,LVLUN
```

In order to initialize the GIRS buffer and the random number generator, LVSETP must be the first GIRS subroutine called. The following variables must also be defined prior to the call to LVSETP and any calls to LVGRN:

LVSTAK()	LVSIZE	LVKPRM
LVVSZE	LVNCOR	LVRNTP
LVHRPG	LVMSPF [†]	LVMSSE [†]
LVSFSZ [†]		

These variables are described in subsequent sections on LVSETP and LVGRN.

The letters "LV" must not be used to begin subroutine and variable names. These initial letters are reserved for GIRS.

The user must first decide on a continuant size (LVVSZE), which determines the maximum number of nodes and links that may be defined for a given page. Its value must be $(n*64)-2$, $n > 0$.^{††} Next, the user must decide how many continuants may be present in core simultaneously (LVNCOR). This value will determine the in-core directory size (LVDRSZ) as computed by LVSETP to be $64*((LVNCOR/64)+1)$. Consequently, the space needed for each of the four fields (NODSPC, LSTSPC, LNKSPC, and FLGSPC) of the GIRS buffer is

$$64*((LVNCOR/64)+1) + LVNCOR*(LVVSZE+2)$$

The user must then decide whether "Sequence Space" will be used. If so, LVSIZE is set to that value; otherwise, LVSIZE is set to 1. Also, the user must dimension array LVRWBF from Common LVREGS to:

$$4*(\text{continuant size}+2)$$

Note that this dimension must be a multiple of 256.

[†]These variables have default values and need be defined only if the node suffix and prefix sizes are modified.

^{††}When used with GIRL, it must be a multiple of 64.

GIRS expects a program to do one of the following things:

- 1) create a new graph
- 2) update an old graph
- 3) query an old graph

LVRNTP must be set to one, two, or three to indicate the type of program to be executed.

It is more efficient for pages (and continuants of those pages) to be initialized at the beginning of execution of a program than to be created on demand. Set LVHRPG to the highest page number desired. There is a limit of 63 pages unless LVSFSZ is modified. LVSFSZ is the node suffix size and has a default size of ten bits, which allows for a maximum continuant size of $2^{10}-1$ or 1024. The prefix size is therefore six bits which allows for 2^6-1 or 63 pages. Changing the suffix size will modify these upper limits accordingly. If LVSFSZ is modified, the prefix and suffix masks, named LVMSPF and LVMSSF, must be updated accordingly. For example, if LVSFSZ is set to 12, set

LVMSPF to 170000

and LVMSSF to 7777

Continuants for each page may be initialized as follows:

Set the I^{th} location in array LVSTAK() to the number of continuants desired (beyond the zeroth) for page I. There is a limit of 63 extra continuants per page. Set the rest of the 140 locations in LVSTAK() to zero.

To initialize the random number generator (LVGRN), set LVKPRM to the first prime number $\geq (\sqrt{\text{LVVSZE}})/2$.

Finally, before calling LVSETP, an output file to contain error statements should be assigned a logical unit number (LVLUN). The following statements, for example, will work on the RT-11 operating system for the PDP-11 computer:

LVLUN = 17

CALL ASSIGN(17, 'SY:ERROR.ERR', 12)

When an identifier is defined by the random number generator (LVGRN), it is given a prefix (a page definition) and a suffix (a random number, unique to that page). The user must assign the prefix via LVREQP(1). The value range for LVREQP(1) is 1 to 63.

Note that if LVDUMP is called, up to 228 variables may be automatically saved at the end of a program if they are placed into COMMON /LVUSER/.

Subroutine LVSETP

Function:

Initializes the I/O channels for the files containing the old and new graphs. Initializes those variables needed for Subroutine LVGRN. Initializes the in-core and out-core directories. Initializes all requested continuants and places them onto disk.

Calling Format:

CALL LVSETP

Input Parameters:

(In COMMON /LVREGS/)

LVHRPG Highest initially requested page. No default.

(In COMMON /LVRAND/)

LVKPRM First prime number $\geq (\sqrt{\text{LVVSZE}})/2$.

(In COMMON /LVBUFR/)

LVVSZE Continuant size; similar to MEMSZE from in-core GIRS. No default.

LVNCOR Number of continuant slots in the in-core GIRS buffer. No default.

LVSFSZ Node suffix size, default is ten bits.

(In COMMON /LVRUN/)

LVRNTP Type of run:

- = 1 Create a new graph (default)
- = 2 Update an old graph
- = 3 Query an old graph

Comments:

LVSETP must be the first GIRS subroutine called by the driving program since it is the main initialization routine. Before initializing the GIRS buffer and other tables, the user is prompted at the teletype for file names for the old and new graphs. The user response must include both names, in command string format, even if only one is needed. The default extension for both file names is .GRF.

LVSETP is in overlay region 1, segment "SETPOP".

Program Length:

2615₈ 1421₁₀

Subroutines Called:

LVFECH	LVDRWR	ICSI
LVGRN	LVMSA	IGETC
VPAGW	LVFIND	IFETCH
LVNSRT	VPAGR	IENTER
		LOOKUP

Called by the Following Subroutines:

LVNPAG
LVNCON

Subroutine LVGRN

Function:

Assigns a page and "random number," which is unique to that page, to a given GIRS identifier.

Calling Format:

CALL LVGRN(NODE)

Input Parameters:

(In COMMON /LVREGS/)

LVREQP(1) Requested page number
 = 0, define an identifier on a new page.
 $1 \leq n \leq 63$, define an identifier on page n.

(In COMMON /LVRAND/)

LVLIST Number of identifiers to be assigned random
 numbers. Default is one.

Output Parameters:

(Format Argument)

Node Contains generated random number. It must be
 dimensioned to "LVLIST" if LVLIST > 1.

Comments:

For each page, a repeatable sequence of unique random numbers is generated in the range of 1 to LVVSZE. LVLIST numbers are generated per call. An attempt to define more than LVVSZE number of identifiers for any one page will terminate the program unless a random number has been "undefined" by Subroutine LVRTRN. Identifiers must be integers. The generated sequence has been previously described by Berkowitz² and Zaritsky.¹

Equivalent GIRS Code:

Identifiers may be defined in GIRL in at least two ways. At the beginning of each routine, a list of identifiers may be defined for page n in the following manner:

G DEFINEn NODE1,..., NODEk

Identifiers may be given random numbers at any time with the following code:

LVREQP(1) = "page number"
G \$'NODE1

Identifiers may also be defined during the execution of an insertion, as discussed further under "Insertion."

Program Length:

750₈ 488₁₀

Subroutines Called:

LVLFSH

LVEXCF

LVERR

Called by the Following Subroutines:

LVSETP

LVINEX

RETRIEVAL OF VALUES

Discussion

Value retrieval is overseen by the find executive routine LVFDEX. This routine brings in the proper continuant so that the lower level routines LVFIND and LVFNV may search for the desired function and value. If the continuant is not specified (default), all the continuants of the requested page will be searched in sequential order until either the function is found or all the continuants have been examined. If the continuant has been specified and the search is to be from "top-to-bottom," the search will proceed to the next higher numbered continuant only if FLAG-11 has been set for that list. If the continuant has been specified and the search is to be from "bottom-to-top," the search will proceed to the previous (lower-numbered) continuant only if FLAG-10 has been set for that list.

LVFDEX expects the user to provide two find strategy routines: USRFD1 and USRFD2. USRFD1 precedes the actual retrieval, but it is skipped if LVFD1 is .FALSE. (default). The retrieval may be skipped if LVFD4 is set within USRFD1 to .FALSE. (default is .TRUE.). USRFD2 follows the retrieval but it is skipped if LVFD2 is .FALSE. (default). If USRFD1 is called, LVFD2 may be modified by LVFD3. USRFD1 and USRFD2 cannot be used recursively.

Subroutine LVFDEX

Function:

- a) Calls user find strategies USRFD1 and USRFD2, skipping the retrieval if LVFD4 is .FALSE.
- b) Brings in the proper continuant or sequence of continuants (if there is no specific request) in preparation for the retrieval.
- c) Breaks up LVFUNC and LVARG into their prefix (page) and suffix (random number) components.
- d) Oversees the following operations:
 - 1) function address computation.
 - 2) determination of function existence. If the function does exist, then
 - 3) location of function within the continuant (since it may not be first on the conflict list, and may therefore reside anywhere in the continuant).
 - 4) determination of whether the function is an SVL or MVL.
 - 5) location in continuant of preceding function on the conflict list.
 - 6) retrieval of the IPOSth value (and its location) of the type indicated, from the top or bottom (depending on the sign of IPOS) of a list of values of a specified function.

Calling Format:

Call LVFDEX(INDEX,INDXAD,KFUNC,KARG,SAVCON)

Input Parameters:

(In COMMON /LVARGS/)

- LVFUNC Link of the triple, also known as the function. The value in IFUNC must contain both a prefix (page number) and a suffix (random number) as defined by a call to LVGRN.
- LVVARG Source node of the triple, also known as the argument of the function. The value in IARG must contain both a prefix (page number) and a suffix (random number) as defined by

LVGRN. The source node prefix determines the page placement of the function and hence the page on which to search.

LVVPOS Position in the multivalue list, IPOS locations from the top (if IPOS is positive) or from the bottom (if IPOS is negative). If ITYP is specified, only that type of value is considered in determining the position.

LVVTYP Type of value to be retrieved:

- = 0 Random number plus page
- = 1 Integer data
- = 2 Hollerith data
- = 3 No specified type (default value)

LVSKIP Saved-index defeat switch. If LVSKIP = 1, the saved-index operation is skipped; otherwise the saved-index feature is in effect. LVSKIP can be set either at the start of the program or just before a call to LVFDEX (after which it may be reset). The saved-index feature is described by Zaritsky.¹

(In COMMON /LVREGS/)

LVREQP(2) Requested continuant:

- = -2 continuant unspecified
- $0 \leq n \leq 63$, continuant n is requested

(In COMMON /LVSWIT/)

- LVFD1 = .TRUE. call user's first retrieval strategy routine.
- = .FALSE. skip user's first retrieval strategy routine (default)
- LVFD2 = .TRUE. call user's second retrieval strategy routine.
- = .FALSE. skip user's second retrieval strategy routine (default)

[Input from USRFD1]

LVFD3 may be used to modify LVFD2

- LVFD4 = .TRUE. proceed with the retrieval (default)
- = .FALSE. skip the retrieval

(Formal Parameter Set)

The formal parameter set is needed by LVFDEX when the saved-index option is to be used. The parameter set consists of five variables, each of which must be unique for each new call to LVFDEX involving a saved index.

KARG	Source node associated with a particular call to LVFDEX
KFUNC	Link associated with a particular call to LVFDEX
INDEX	Position in the list of the value retrieved from the most recent call to LVFDEX. If INDEX is negative, it is the position from the bottom of the list.
INDXAD	Location in continuant SAVCON of the value retrieved from the most recent call to LVFDEX
SAVCON	Continuant on which list resides

Output Parameters:

(In COMMON /LVARGS/)

LVVPOS	Set to 1 (default value)
LVVTYP	Set to 3 (default value)
LVVAL	Retrieved value (LVVPOS th value of the type LVVTYP). LVVAL is set to LVVARG if the value cannot be found.
LVVTR	If the LVVPOS th value of the LVVTYP exists, LVVTR = 1; otherwise LVVTR = -1.

(In COMMON /LVREGS/)

LVCUPG(1)	Current page. If LVVTR=1, LVCUPG(1) is set to the page containing the requested function.
LVCUPG(2)	Current continuant. If LVVTR=1, LVCUPG(2) is set to the particular continuant containing the requested function.

Equivalent GIRL Code:

C SET LVREQP(2) TO THE REQUESTED CONTINUANT, IF DESIRED.

G NODE+LINK.tsJ

where t is the type of value to be retrieved:

= "	Identifier (node defined by LVGRN)
= .	Integer value
= /	Hollerith value
= "blank"	Any type value

s is the indicating direction of search:

= + or

"blank"

Search from top of list

= -

Search from bottom of list

J is the same as IPOS (=LVVPOS)

Program Length:

1254₈

686₁₀

Subroutines Called:

LVSTAC

LVFIND

LVRTSH

LVBOTM

LVPOP

LVFNV

LVEXCH

Called by the Following Subroutines:

LVDLEX

LVINEX

LVINCL

RETRIEVAL OF MVL INDEX OF GIVEN VALUE OF A FUNCTION (INCLUSION)

Subroutine LVINCL

Function:

Determine the first MVL position of a given value.

Calling Format:

CALL LVINCL

Input Parameters:

(In COMMON /LVARGS/)

LVVINC Value on which the list position is to be determined

Output Parameters:

(In COMMON /LVARGS/)

LVVPOS First position in the MVL in which the indicated value
 is found

LVVINC = 1 Desired value has been found on the MVL
 = -1 Desired value has not been found on the MVL

LVVTR Same as LVVINC

Equivalent GIRL Code:

Use of the GIRL inclusion operator can best be explained with three examples.
Further discussions and examples are given in Berkowitz.⁵

Assume for all examples that the source node is NODE and the link is LINK:

Example 1. Delete value3 on the MVL

G NODE+LINK-.:value3

Example 2. Determine the position of value1 (if such
a value exists) on the MVL and name it
INDEX; otherwise transfer to fail.

G NODE+LINK value1/fail':INDEX

Example 3. Replace value1 on the MVL with value2.

G NODE LINK: value1 value2

Program Length:

230₈ 152₁₀

Subroutine Called:

LVFDEX

INSERTION

Discussion

The insertion operation is overseen by the Insert Executive Routine LVINEX. This routine ensures that the triple is completely defined and then determines placement of that triple. If the triple is already fully defined, the requested page is determined by the prefix of the source node (LVVARG). It is in the domain of insertion that a new page or continuant can be requested. If a particular continuant is not requested and the function did not previously exist, the triple is placed on the (sequentially) first continuant with available space. If a value is to be added to a list that has been specifically placed on a particular continuant, but a different continuant is specifically requested, subroutine LVREOR reports an error. The insertion proceeds, however, with the entire list moved onto the newly requested continuant.

LVINEX expects the user to provide two insertion strategy routines: USRIN1 and USRIN2. USRIN1 precedes the actual insertion, but it is skipped if LVIN1 is .FALSE. (default). The insertion may be skipped if LVIN4 is set within USRIN1 to .FALSE. (default is .TRUE.). USRIN2 follows the insertion, but it is skipped if LVIN2 is .FALSE. (default). If USRIN1 is called, LVIN2 may be modified by LVIN3. USRIN1 and USRIN2 cannot be used recursively.

Subroutine LVINEX

Function:

- a) Calls user insertion strategies USRIN1 and USRIN2, skipping the insertion if LVIN4 is .FALSE.
- b) Ensures that the source node, link and, if a random number, the sink node are all completely defined (contain both a prefix and suffix).
- c) Determines on which page and continuant to place the triple and brings that continuant into the buffer, if necessary.
- d) Oversees the actual insertion by Subroutine LVNSRT.

Calling Format:

CALL LVINEX

Input Parameters:

(In COMMON /LVARGS/)

LVFUNC	Link of the triple, also known as the function. A fully defined link contains a prefix* (page number) and a suffix (random number) as given by LVGRN. = 0 Define the link with a prefix set to the current page = $1 \leq n \leq 63$ Define the link with a prefix set to n
LVVARG	Source node of the triple, also known as the argument of the function. The prefix of LVVARG determines on which page to place the triple. A fully defined node contains a prefix and

*Unless modified by the user, the prefix consists of the leftmost six bits of the node or link.

a suffix as given by LVGRN. Otherwise,

- = -1 Place triple on a new page and define the node
- = 0 Place triple on the current page and define the node
- $1 \leq n \leq 63$ Place triple on page n and define the node

LVVNVL Number of values (up to ten) to be inserted (default is 1)

LVTYP(10) Type of each value in LVVALS(i) to be inserted:

- = 0 Random number (default value)
- = 1 Integer data
- = 2 Continuing Hollerith data
- = 3 The only or final cell or a Hollerith data string

LVVALS(10) Array containing the values or sink nodes to be inserted. LVVALS(i) may contain any of the following types of values:

- . Random number, as defined by LVGRN
- . Integer data; see Berkowitz² for limitations on Integer data
- . Hollerith data; see Berkowitz² for limitations on Hollerith data

If LVTYP(i) = 0 (random number), LVVALS(i) may also take on the following forms:

- = -1 Define the sink node with a prefix = "current page + 1"
- = 0 Define the sink node with a prefix set to the current page
- $1 \leq n \leq 63$ Define the sink node with a prefix set to n

LVNTYP Orientation of insertion

- = 0 Insert sink node (default)
- = 1 Insert source node
- = 2 Insert link

LVNDXN

Type of insertion to be made:

- = 0 Normal insertion; the triple is always placed at the end of the (null) list. This is the default value.
- = 1 Destructive insertion; the contents of the LVVPOSth member of the LVVTYP type (counting from the top or bottom of the list, depending on the sign of LVVPOS) are replaced by the contents of LVVALS(1).
- = 2 Nondestructive insertion; the contents of LVVALS(1) are wedged into the list, making the new value the LVVPOSth member of the LVVTYP type from the top or bottom of the list (depending on the sign of LVVPOS).

The following two variables are needed only if LVNDXN = 1 or 2:

LVVPOS

LVNSRT will place the value to be inserted LVVPOS locations (as modified by LVVTYP) from the beginning or, if negative, from the end of the list.

LVVTYP

Type of value to be counted when attempting to insert a value at LVVPOS locations from the beginning or end of a list.

(In COMMON /LVREGS/)

LVREQP(2)

Requested continuant. Note that LVREQP(1) contains the requested page which is extracted from LVVARG.

- = -1 Request new continuant
- = -2 Continuant unspecified (default)
- = -3 Current continuant (if current page = requested page)

$0 \leq n \leq 63$ Continuant n is requested

(In COMMON /LVSWIT/)

LVINI

- = .TRUE. Call user's first insertion strategy routine
- = .FALSE. Skip user's first insertion strategy routine (default)

LVIN2 = .TRUE. Call user's second insertion strategy routine
 = .FALSE. Skip user's second insertion strategy routine
 (default)

(input from USRIN1)

LVIN3 May be used to modify LVIN2
 LVIN4 = .TRUE. Proceed with the insertion (default)
 = .FALSE. Skip the insertion

Output Variables:

(In COMMON /LVARGS/)

LVVPOS Set internally to 1 (default value)
 LVVTYP Set internally to 3 (default value)
 LVVAL Set internally to LVVALS(1)
 LVVNVL Set internally to 1 (default value)
 LVVTR = -1 Function did not exist prior to this
 insertion
 = 1 Function did exist prior to this
 insertion
 LVNDXN Set internally to 0 (default value)

(In COMMON /LVREGS/)

LVCUPG(1) Current page (as a result of this insertion)
 LVCUPG(2) Current continuant of current page. (Contains inserted
 triple.)

Equivalent GIRL Code:

Assume that NODE1 is the source node and LINK1 is the LINK and (in the first four examples) both NODE1 and LINK1 have been initialized in a DEFINEN statement:

- 1) Add random number value1 to the (null) list:

G NODE1 LINK1 value1

- 2) Add integer I to the end of the list

G NODE1 LINK1 "I"

- 3) Place value1 in the third location from the bottom of the list.

G NODE1 LINK1 .-3 value1

- 4) Replace the second integer value from the top of the list with the integer 10.

G NODE1 LINK1-..2 "10"

5) Assign a random number to value1 (for the current page) and place the triple on page 5, continuant 0.

```

      NODE1 = 5
      LVREQP(2) = 0
      VALUE1 = 0
G     NODE1   LINK1   VALUE1
      PRINT NODE1, VALUE1

```

6) Place each of the following ten triples on new pages, assign random numbers to the source nodes, links, and sink nodes, and define each of the links and sink nodes to the page which is current at the time of definition of the source node. The triples will automatically be placed on the zeroth continuants of each new page.

```

      DO 5 I = 1, 10
      NODE = -1
      LINK = 0
      SINK = 0
G     NODE LINK SINK
      PRINT NODE, LINK, SINK
5     CONTINUE

```

7) Define NODE1, LINK1, SINK to page 3, place this triple on continuant 2, and call the first insert strategy routine.

```

G     DEFINE3 NODE1, LINK1, SINK
      .
      .
      .
      LVIN1 = .TRUE.
      LVREQP(2) = 2
G     NODE1 LINK1 SINK

```

Program Length:

1622₈

914₁₀

Subroutines Called:

LVSTAC	LVLFSH
LVPOP	LVRTSH
LVNPAG	LVERR
LVGRN	LVEXCH
LVNCON	LVFDEX
LVREOR	LVFIND
LVOVER	

Subroutine LVREOR

Function:

To move a list from its present location to a new continuant as specified by REQCON.

Calling Format:

CALL LVREOR(REQCON)

Input Parameters:

(Formal Parameter Set)

REQCON	The list is to be moved from continuant "LVREQP(2)" to continuant "REQCON" of page "LVREQP(1)".
--------	---

Comments:

Subroutine LVFDEX must be called immediately prior to a call to this routine. If the original list was specifically placed on LVREQP(2), an error message is written out. Two continuants may be MERGED together by calling LVFDEX and this routine once for each function in the original continuant (LVREQP(2)). A particular list may be SEPARATED from one continuant and placed on another (REQCON) in the same fashion. The present version of LVREOR expects a new triple to be added to continuant REQCON each time it is called. Also, this triple must be fully defined.

DELETION

Discussion

The delete operation is overseen by the delete executive Routine LVDLEX. If no continuant is requested, LVDLEX brings in (sequentially) all continuants of the requested page (as defined by the prefix of the source node) until either the function is located or there are no more continuants of the requested page.

LVDLEX expects the user to provide two deletion strategy routines: USRDL1 and USRDL2. USRDL1 precedes the actual insertion, but it is skipped if LVDL1 is .FALSE. (default). The deletion may be skipped if LVDL4 is set within USRDL1 to .FALSE. (default is .TRUE.). USRDL2 follows the deletion but it is skipped if LVDL2 is .FALSE. (default). If USRDL1 is called, LVDL2 may be modified by LVDL3. USRDL1 and USRDL2 cannot be used recursively.

Subroutine LVDLEX

Function:

- a) Calls user deletion strategies USRDL1 and USRDL2, skipping the deletion if LVDL4 is .FALSE.
- b) Searches in sequential order (unless the continuant is specified) the continuants of the requested page for the requested function.
- c) Oversees the actual deletion by Subroutine LVDLET.

Calling Format:

CALL LVDLEX

Input Parameters:

(In COMMON /LVARGS/)

LVFUNC	Link of the triple; must be a random number as defined by LVGRN.
LVVARG	Source node of the triple; must be a random number as defined by LVGRN.
LVNDXN	= 0 Delete entire function (default) = 1 Delete specific value as described by LVVPOS and LVVTYP

The following two variables are needed only if LVNDXN = 1:

LVVPOS	Position in the MVL of the value to be deleted (number of locations from the top, if positive, and from the bottom, if negative). If LVVTYP is specified, only that type of value is counted in determining the position in the list. LVVPOS is used only for indexed deletion.
LVVTYP	Type of value to be deleted from a multivalued list (used only for indexed deletion) = 0 Random number = 1 Integer data = 2 Hollerith data = 3 No specified type (default value)

(In COMMON /LVREGS/)

LVREQP(2)	Requested continuant = -2 Continuant unspecified (default) 0 ≤ n ≤ 63 Continuant n is requested
-----------	---

(In COMMON /LVSWIT/)

LVDL2 = .TRUE. Call user's first deletion strategy routine
 = .FALSE. Skip user's first deletion strategy routine
 (default)
LVDL2 = .TRUE. Call user's second deletion strategy routine
 = .FALSE. Skip user's second deletion strategy routine
 (default)

(input from USRDL1)

LVDL3 May be used to modify LVDL2
LVDL4 = .TRUE. Proceed with the deletion (default)
 = .FALSE. Skip the deletion

Output Parameters:

(In COMMON /LVARGS/)

LVVAL Deleted value. If the entire list is deleted, LVVAL
 returns the first value of the list.
LVVTR Function indicator. If the function or specified value
 of that function does not exist, the attempted deletion
 is considered to have failed. LVVTR is actually set in
 LVFIND and LVFNV.
 = 1 Function exists
 = -1 Function does not exist
LVVPOS Set internally to 1 (default value)
LVVTYP Set internally to 3 (default value)
LVNDXN Set internally to 0 (default value)

(In COMMON /LVREGS/)

LVCUPG(1) Current page (contained deleted triple)
LVCUPG(2) Current continuant of current page

Equivalent GIRL Code:

Assume NODE1 is the source node and LINK1 is the link.

Example 1.

Delete entire function which begins on continuant 2.

LVREQP(2) = 2

G NODE1-LINK1

Example 2.

Delete the I^{th} value on an MVL, continuant is not known.

G NODE1+LINK1-.I

Program Length:

762₈ 498₁₀

Subroutines Called:

LVSTAC	LVDLET
LVPOP	USRDL1
LVERR	USRDL2
LVFDEX	

DISK STORAGE AND RETRIEVAL OF A GRAPH

Discussion

After a graph has been created, it may be conveniently stored in binary format on disk and later retrieved from disk via the Subroutines LVDUMP and LVFECH. Although this task can be performed without these routines, their use ensures that all pertinent variables will be properly defined. LVDUMP also enables the user to have the entire graph, a single page of that graph, or the contents of the buffer generated in ASCII format for debugging purposes. The dump is placed on logical unit LVLUN which must be defined in a call to SYSLIB function ASSIGN.

Another advantage of this arrangement is that it makes it easy for the user to restart a program using new data. The original graph will be retrieved whenever a new call to LVFECH is made. If LVDUMP is called, up to 228 identifiers may be automatically saved at the end of a program if they are placed into COMMON /LVUSER/.

The names for the files containing the old and new graphs are declared at the beginning of execution of the program. A prompt character is sent to the teletype and the user response must include names, in command string format, for both an old and new graphs, even if only one is needed. The default extension for both file names is .GRF.

Subroutine LVDUMP

Function:

LVDUMP will either:

a) Store the entire graph, pertinent GIRS system variables, and 128 identifiers from COMMON /LVUSER/ onto the output file in a format suitable for later recovery by Subroutine LVFECH, or

b) For debugging purposes, create an ASCII file on logical unit LVLUN consisting of GIRS system variables plus one of the following:

- 1) The entire graph
- 2) A single page of the graph
- 3) Those continuants residing in the buffer at the time of the call to LVDUMP

Calling Format:

CALL LVDUMP(DUMP)

Input Parameters:

(In COMMON /LVFRAM/)

LVMODE Determines whether to invoke function "a" or "b"
 = LVBNRY function "a"
 = LVBCD function "b"

If function "b" is invoked, the following three parameters are needed:

LVPGS = -1 Output those continuants residing in
 the buffer at the time of the call to
 LVDUMP

 = 0 Output all continuants of all pages
 1≤n≤63 Output all continuants of page n

LVLUN Logical unit number of the ASCII file which will contain
 the output from LVDUMP. It must be defined in a CALL ASSIGN
 statement.

(Formal Parameter)

DUMP = 0 Output to LVLUN some of the pertinent GIRS
 variables found in the labeled commons
 = 1 Output to LVLUN all the pertinent GIRS
 variables found in the labeled commons

Program Length:

553₈ 363₁₀

Subroutines Called:

LVERR LVPAGW

LVWRIT LVCLOS

LVEXEC

Subroutine LVFECH

Function:

Reads in (in binary format) pertinent GIRS system variables, up to 228 user identifiers from labeled COMMON /LVUSER/, and a previously created graph from disk into the GIRS buffer. Then it copies the graph onto a new disk file.

Calling Format:

CALL LVFECH

Comments:

LVFECH expects the disk file to have been created by LVDUMP. At the beginning of the program LVFECH is called by LVSETP if LVRNTP = 1 or 2. LVFECH may be called by the user directly if there is a need to reinitialize the graph.

Program Length:

1036₈ 542₁₀

Subroutines Called:

LVERR	LVDRRD
LVPAGR	LVDWR
LVMSA	LVPAGW

Called by the Following Subroutine:

LVSETP

EXECUTING A GIRS PROGRAM

GENERAL DISCUSSION

GIRS may be used directly via user calls to the GIRS subroutines or indirectly with the GIRL⁵ language. In either case, the object code for the driving program must precede the object code for the GIRS routines in any LINK-LOAD.

It is generally more advantageous for the user to use GIRS indirectly via GIRL, since GIRL not only includes all the capabilities of GIRS but also spares the user from concern over setting up all the labeled commons and initializing pertinent variables. The command sequences and FORTRAN statements needed to preprocess, compile, link, and execute GIRL/GIRS programs on the PDP-11 follow.

INDIRECT USE OF A GIRS SUBROUTINE VIA GIRL

A GIRL program must include the following statements:

Options card

Continuant specification card 1

Continuant specification card 2 (if >25 pages specified)

Continuant specification card 3 (if >50 pages specified)

First user program card

or

\$ SUBROUTINE name

non-DATA specification statements

G DEFINE1 var1,var2,..., varn (optional)

G DEFINE2 var1, varj,..., vark (optional)

.

.

.

G DEFINE63 varx,vary,..., varz (optional)

DATA string (optional)

G EXECUTE

GIRL/FORTRAN executable code (no END statement)

G COMPLETE

Other GIRL/FORTRAN routines

(Purely FORTRAN routines may be included here but it is faster to add them later when the object files are linked together.)

Notes:

- 1) In the GIRL/FORTRAN routines, GIRL statements are declared by placing a G in Column 1. Continuation cards are handled as in FORTRAN.
- 2) The option card has the following entries: (the first three items must be entered in a 3I4 format)

Continuant size - Must be set to a multiple of 64, with a maximum value of 960.* This value determines the size for all continuants of all pages. It also determines the maximum number of nodes which may be defined for each page. No default.

Number of continuants to reside in the buffer - The in-core directory, the continuant size, and this item determine the size of the buffer. If the buffer, which consists of four arrays of equal size, will contain less than 64 continuants, it will have a length of:

$$4 * (64 + (\text{cont. size} * \text{no. of conts. in buffer}))$$

No default.

Highest requested page number - It is more efficient to initialize pages at the beginning of execution of a program than to create them "on demand." Value range is 1-63.

*This assumes a default prefix size of six bits and suffix size of ten bits.

The following options are in free format and must be separated by at least one blank or comma:

OUTCOR	Self explanatory. Default is the non-paged "In-Core" version of GIRS.
CREATE	Create a new graph (current default value). Note that CREATE, UPDATE, and QUERY are mutually exclusive.
UPDATE	Modify an existing graph.
QUERY	Query an existing graph.
SUFFIXnn	Allot nn bits for the identifier suffix. Default is ten bits.
\$IIIIII	Declare the size of SEQ. (An integer of at most six digits preceded by a dollar sign (\$).) Default size is one location.
PRINT	Print GIRL program on output file. Default is no-print.
COMMENTS	Place GIRL code with a G in Column 1 into pre-processed FORTRAN code. Default is no-comment.
LXX	Declare the maximum allowable levels of parenthesization. (An integer of at most two digits preceded by a letter L.)
NOSAVE	Eliminates the saved-index facility, and is therefore appropriate for short multivalued lists. (See the discussion of "saved index" by Zaritsky. ¹)

3) Continuant specification card(s): Continuants for each page may be initialized at the beginning of execution of a program. The value range is 0-63. The format is 25I3 for all three continuant specification cards. If the "highest requested page" (see discussion on options card) has value n, then n continuant specifications are expected to be read in.

Preprocessing and Compiling a GIRL/FORTRAN Program

Assume that all the files are to reside on the system disk* and that the GIRL program "USER.GRL" is to be preprocessed and executed. The preprocessor accepts the GIRL, FORTRAN, and list file names in Command String Interpreter format with default file extension names GRL, FOR, and LST, respectively. The preprocessor will create

*The graph used by the preprocessor 'PRPGRF.BIN' must reside on the system disk drive ('SY:').

a FORTRAN file and (as an option) a GIRL listing. These files are to be named "USER.FOR" and "USER.LST," respectively. A copy of the GIRL listing will also be sent to the terminal if the PRINT option has been requested. The periods and asterisks at the beginning of lines are system prompt characters. The terminal dialog involved in preprocessing and compiling the GIRL program "USER.GRL" is as follows (linking and executing the program are described in the next section):

	<u>Line No.</u>
.R PREP	(1)
ALL REAL VARIABLES MUST BE DECLARED	
ERRORS ARE FLAGGED BY ****ERROR	
PLEASE ENTER FILE NAMES IN COMMAND STRING FORM	
*USER=USER	(2)
or, if a list file is also desired:	
*USER,USER=USER	(2a)
.R FORTRAN	(3)
.USER=USER/W	(4)
*^C (control C)	(5)

DIRECT USE OF GIRS SUBROUTINES

Calls to GIRS routines may be placed directly into FORTRAN programs. Programs are compiled as with any ordinary FORTRAN program.

Linking and Executing a GIRL/FORTRAN Program

Linking a compiled GIRL program is best accomplished indirectly by executing a BATCH program which contains the link statements. If both the user's program and the GIRS routines reside on the system disk, the BATCH file "USER.BAT" appears as follows:

\$JOB
\$RUN LINK
\$DATA
USER=USER,SYSLIB,RK1:FINDOP,DLETOP/F/C
NSRTOP/0:1/C
DLEXOP/0:1/C
DRCTOP/0:2/C
OPENOP/0:2/C
MSAOP/0:2/C
SETPOP/0:3/C
SAVEOP/0:3/C
SPECOP/0:3/C
PAGIOP/0:4/C
DIRIOP/0:4/C
UTILOP/0:5/C
VALUOP/0:5/C
EROP/0:5
\$EOD
\$EOJ

The following statements are needed to execute USER.BAT

.LOAD TT,BA
.ASS TT,LOG,LST
.R BATCH
*USER

The following steps should be taken if the program does not fit into main memory:

.R BATCH
*/U
.UNLOAD TT,BA

The program is now ready for execution.

.R USER

GIRS will immediately respond by printing out

PLEASE ENTER FILE NAMES OF OLD AND NEW GRAPHS

IN COMMAND STRING FORMAT (NEW.EXT = OLD.EXT)

.GRF IS ASSUMED EXTENSION

*NEWFIL=OLDFIL

Although the program may not need both "NEWFIL" and an "OLDFIL," dummy file names must be given.

OVERLAY STRUCTURE

An overlay structure has been created to reduce the effective size of out-core GIRS from 13653₁₀ to 8332₁₀ words. In general, the effective size cannot be reduced further due to the complex interrelationships among the subroutines as shown in Appendix B. However, under some circumstances, GIRS subroutines which perform special operations such as the creation of a new page on demand or the dumping of all GIRS system variables may be left out, reducing the size even further. These and other special operations which may be removed are discussed in note 3 in the section on limitations and memory requirements. Of course, if a user has subroutines which do not use GIRS, further space may be saved by linking them into the three overlay regions.

The sizes (in words) of the overlay regions are listed in Table 4 and the overlay structure is given in Table 5.

TABLE 4 - OVERLAY REGION SIZES

OVERLAY REGION	OCTAL	DECIMAL
Root Segment	4531	2393
1	5434	2844
2	345	229
3	4123	2131
4	412	266
5	725	469
Total	20214	8332

TABLE 5 - THE OVERLAY STRUCTURE

Overlay Region	Subroutines Listed by Segment (Size in Decimal Words)		
Root Segment	LVFDEX, LVFIND, LVFNV, LVBOTM, LVEXCH, LVDLET (2393)		
1	LVINEX LVNSRT LVUPDT (2844)	LVDLEX (481)	
2	LVMSA (229)	LVDRCT (196)	LVOPEN LVRPLC (157)
3	LVSETP LVGRN LVNPAG LVNCON (2131)	LVFECH LVDUMP LVWRIT LVCLOS (1578)	LVREOR LVOVER LVINCL (926)
4	LVPAGR LVPAGW (266)	LVDDRRD LVDRWR (99)	
5	LVALUE LVSUM (469)	LVSTAC LVPOP LVRTRN LVLFSS LVRTSH (348)	LVERR (21) [skeleton version]

LIMITATIONS AND MEMORY REQUIREMENTS

The following limitations are based on the 16-bit word size and 32K memory of a PDP-11 computer. A default suffix size of ten bits is assumed.

maximum number of pages = 63

(numbered 1 to 63)

maximum number of continuants/page = 64

(numbered 0 to 63)

prefix size = 6 bits = $2^6 - 1 = 63$

suffix size = 10 bits = $2^{10} - 1 = 1023$

maximum continuant size = 960

maximum range of node values/page = 1-958

Maximum size for user program and GIRS buffer:

approximately 9900 words

Notes:

1) The GIRS buffer consists of the four arrays NODSPC, LSTSPC, LNKSPC, and FLGSPC from labeled commons LVVTR1, LVVTR2, LVVTR2, and LVVTR4, respectively.

2) The size of each array is determined as follows:

$$\text{length} = 64 * ((\text{LVNCOR}/64) + 1) + \text{LVNCOR} * (\text{LVHDRS} + \text{LVVSZE})$$

where LVHDRS is internally defined to two and LVVSZE (the continuant size) must be two less than a multiple of 64.

3) Special functions may be eliminated from the GIRS package if not needed. Of course, this will result in a linkage error message: UNDEF GLOBALS. The following subroutines may be considered:*

*All subroutine lengths are in decimal words.

Overlay Region 1

Segment 1

LVINEX

LVNSRT List insertion package (2844)

LVUPDT

Segment 2

LVDEX List deletion executive (481)

Overlay Region 3

Segment 1

LVGRN Generate a random number (481)

LVNPAG Create a new page on demand (139)

LVNCON Create a new continuant on demand (227)

Segment 2

LVFECH Read-in a previously created graph (512)

LVDDUMP } { Create either an ASCII dump of GIRS continuants
LVWRIT } { or a binary file which contains the graph and
LVCLOS } { close that channel (1066)

Segment 3

LVREOR List reorganization. Required only if lists are to be placed on specifically requested continuants (639)

LVINCL Inclusion operation (152)

Overlay Region 5

Segment 3

LVERR GIRS system variable dump (1098)

Note that the LVERR routine listed in Table 5 is only a skeleton version of this routine

4) User subroutines which have no calls to GIRS routines may be added to the present overlay structure. The maximum sizes (in words) of the overlay regions are as follows:

Overlay region 1	2844
Overlay region 2	229
Overlay region 3	2131
Overlay region 4	266
Overlay region 5	469

If NODSPC, LSTSPC, LNKSPC, and FLGSPC all have lengths of 128 words and LVRWBF has a minimum length of 256 words, the minimum space required for GIRS labeled commons is 1865 words.

ADDING A USER-EMBEDDED STRATEGY

INTRODUCTION

One of the major goals of any information retrieval system is to allow efficient access to the data base, most likely by more than one user and possibly by users who do not have a sophisticated knowledge of a computing environment. The Data Base Administrator (DBA) may control this situation at the time that the information is organized and placed into the data base and also when an attempt is made to retrieve information from the data base.

If the DBA is to control the placement of information into the graph by and for several users, the DBA may wish to create a graph partition strategy which is universal to that particular set of users. As described by Berkowitz:²

"A typical STRATEGY might be: "if the link is A, place the sink node on page 3; if the link is B, place the sink node on page 4; otherwise default."

An efficient graph partition would reduce disk I/O for retrievals considerably.

It is reasonable to expect users, particularly unsophisticated users, to make queries which cannot be directly answered by the data base. At the expense of some computer space and time, these queries may be handled with inferential search strategies. For example, if a particular retrieval should fail, call a retrieval strategy to determine whether the link exists at some level below the source node. This technique is described further by Zaritsky,³ pages 46-51, and Berkowitz,² pages 28-44.

It is also possible for the data base to adapt to the needs of the users. For example, a monitoring strategy could be created to keep a "scorecard" of imprecise queries. Direct relationships might be placed into the graph for those queries made most often.

USE

Paged GIRS allows for the inclusion of user strategies both before and after insertion, retrieval, and deletion operations. The appropriate GIRS subroutines expect the strategies to be named as follows:

Insertion

USRIN1 (before)

USRIN2 (after)

Retrieval

USRFD1 (before)

USRFD2 (after)

Deletion

USRDL1 (before)

USRDL2 (after)

The following switches (all from labeled common LVSWIT) are needed to use the strategy. Variables with "IN" in the name are used for insertion, those with "FD" for retrieval, and those with "DL" for deletion.

The "before" strategies may be skipped if LVIN1, LVFD1, or LVDL1 are .FALSE. (default). The "after" strategies may be skipped if LVIN2, LVFD2, or LVDL2 are .FALSE. (default). The insertion, retrieval, or deletion operations may be skipped entirely if LVIN4, LVFD4, or LVDL4 are set in USRIN2, USRFD1, or USRDL1, respectively, to .FALSE. (default is .TRUE.). If the "before" strategies are called, LVIN2, LVFD2, or LVDL2, may be modified by LVIN3, LVFD3, or LVDL3, respectively. The user strategies may not be used recursively.

PROPOSED EXTENSIONS

In the near future, we hope to merge the paged version of GIRS with a paged hardware associative memory facility. The result will be an enhanced system with high speed relational processing.

ACKNOWLEDGMENTS

The overall scheme for the software described was designed by Dr. S. Berkowitz. It was written under his supervision with his advice and encouragement.

APPENDIX A
VARIABLES IN LABELED COMMON

In the following list of all the labeled commons required by out-core GIRS, the external names, as created by the GIRL preprocessor, are given for those commons which must be included in the user's main program. Otherwise, internal names are used.

EXTERNAL:

```
COMMON /LVARGS/ LVFUNC,LVVARG,LVVPOS,LVVTYP,LVVAL,LVVNVL,LVSKIP,
1          LVVTR,LVVINC,LVNDXN,LVVALS(10),LVTYPE(10),
2          LVSRSF,LVLNSF,LVSNSF,LVNTYP
COMMON /LVVSEQ/ LVSIZE,LVSEQ1,LVSEQ2,SEQSPC(1)
COMMON /LVRAND/ LVKPRM,LVKS,LVKY,LVKDY,LVKDX,LVTEMP,LVLIST,
1          LVNTBL(256)
COMMON /LVVTR1/ NODSPC(buffer size)
1          /LVVTR2/ LSTSPC(buffer size)
2          /LVVTR3/ LNKSPC(buffer size)
3          /LVVTR4/ FLGSPC(buffer size)
COMMON /LVCRTN/ LVVGSP,LVCTRL,LVCTR1,LVLSTV,LVNFRE,LVFREE,LVDREG,
2          LVVMSA,LVPGLC,LVCRTN
COMMON /LVBUFR/ LVVSZE,LVNWCH,LVOLCH,LVCMPR,LVPGHD,LVBFSZ,
1          LVDRSZ,LVNCOR,LVHDRS,LVMSAD,
2          LVSFSZ,LVBKSZ,LVDRBK,LVPGH4
COMMON /LVREGS/ LVCUPG(4),LVREQP(4),LVLVPG(4),LVMSAR
1          LVHRPG,LVNMSA,LVHAPG(2),LVRCNT,LVUCNT,LVDRPG,
2          LVDIRC,LVOTLC,LVOTDR(256),
3          LVRWBF(4*continuant size)
COMMON /LVPRAM/ LVBFLC,LVLNTH,LVVERR,LVERNO,LVBNRY,LVBCD,
1          LVMODE,LVPRS,LVLUN
COMMON /LVRUN/  LVRNTP,LVCORE
COMMON /LVSTAK/ LVLEVL,LVNVAR,LVSTAK(140)
COMMON /LVMASK/ LVWRIT,LVNUSE,LVNWCN,LVMSK3,LVMSSF,LVMSPF
```

```

COMMON /LVSWIT/ LVSTP,LVSNGL,LVNXTR,LVIN1,LVIN2,LVFD1,
1             LVFD2,LVDL1,LVDL2,LVIN3,LVFD3,LVDL3,LVDMP,
2             LVFD4,LVDL4,LVIN4
COMMON /LVUSER/ USER(228)
COMMON /LVUTIL/ FILSPC(39),DEFEXT(2)

```

INTERNAL:

```

COMMON /LVFLAG/ FLOMSK,FL1MSK,FL2MSK,FL3MSK,FL4MSK,FL5MSK,
1             FLG67,FLAG8,FLAG9,FLAG10,FLAG11,FLAG12,FLAG13,
2             FLAG14,FLAG15
COMMON /LVHDL/ THMSA,REGAS,PAGENO,CONTNO,INSDEL,
1             USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
COMMON /LVFND/ IADD,THIS,LSTHED,LOC,LAST,LASTLC
COMMON /LVFND/ COUNT,ABSPOS,LSTCON
COMMON /LVINS1/ REORG,FULL,RPLACE
COMMON /LVDEL1/ NUMRET

```

Note that all variables from labeled common LVSWIT must be set to LOGICAL*1.

APPENDIX B
SUBROUTINE CALLING STRUCTURE

This appendix lists all the subroutines in Out-Core GIRS and the GIRS subroutines called by them:

```
SUBROUTINE LVSETP
    LVFECH
    LVGRN()
    LVPAGW
    LVDRWR
    LVMSA()
    LVPAGR()
    LVFIND
    LVNSRT

SUBROUTINE LVMSA(CONNUM)
    LVERR()
    LVDRRD()

SUBROUTINE LVCLDS
    LVERR()
    LVPAGW
    LVDRWR

SUBROUTINE LVDRRD(CHAN)
    LVERR()

SUBROUTINE LVDRWR
    LVERR()

SUBROUTINE LVPAGR(CHAN)
    LVERR()

SUBROUTINE LVGRN(NODE)
    LVLFSH(,)
    LVEXCH
    LVERR()

SUBROUTINE LVEXCH
    LVDRCT
    LVMSA()
    LVOPEN
    LVPAGR()
    LVRPLC
    LVSUM

SUBROUTINE LVSTAC
    LVERR()
```

```

SUBROUTINE LVPOP
    LVERR()

SUBROUTINE LVDRCT
    LVSTAC
    LVFIND
    LVPOP

FUNCTION LVLFSH(WORD,BITS)

FUNCTION LVRTSH(WORD,BITS)

SUBROUTINE LVDLEX
    LVSTAC
    LVPOP
    LVFDEX(,,,)
    LVDLET
    LVERR()

SUBROUTINE LVDLET

SUBROUTINE LVRTRN

SUBROUTINE LVFDEX(INDEX,INDXAD,KFUNC,KARG,SAVCON)
    LVSTAC
    LVPOP
    LVRTSH(,)
    LVEXCH
    LVFIND
    LVFNV(,,,)
    LVEXCH
    LVBOTM

SUBROUTINE LVFIND
    LVERR()

SUBROUTINE LVFNV(INDEX,INDXAD,KFUNC,KARG,SAVCON)

SUBROUTINE LVBOTM
    LVEXCH
    LVERR()
    LVFIND

SUBROUTINE LVINCL
    LVFDEX(,,,)

```

SUBROUTINE LVINEX

LVSTAC
LVPOP
LVNPAG
LVGRN()
LVLFSH(,)
LVRTSH(,)
LVERR()
LVEXCH
LVNCON
LVFDEX
LVREOR()
LVNSRT
LVFIND
LVOVER

SUBROUTINE LVNSRT

LVUPDT
LVFIND
LVFNV(,,,,)

SUBROUTINE LVUPDT

SUBROUTINE LVREOR(REQCON)

LVERR()
LVEXCH
LVSTAC
LVFIND
LVNSRT
LVDLET
LVPOP

SUBROUTINE LVOVER

LVSTAC
LVDLET
LVPOP
LVNSRT

SUBROUTINE LVNPAG

LVMSA()
LVEXCH
LVOPEN
LVSETP
LVPAGW
LVRPLC
LVSUM

SUBROUTINE LVNCON

LVMSA()
LVOPEN
LVSETP
LVPAGW
LVPAGR()
LVRPLC
LVSUM

SUBROUTINE LVERR(DUMP)

SUBROUTINE LVOPEN

LVALUE
LVPAGW

SUBROUTINE LVRPLC

LVSTAC
LVFIND
LVDLET
LVNSRT
LVPOP

SUBROUTINE LVSUM

SUBROUTINE LVALUE

LVDUMP()

SUBROUTINE LVFECH

LVERR()
LVPAGR()
LVDRRD()
LVDRWR
LVMSA()
LVPAGW

SUBROUTINE LVDUMP(DUMP)

LVERR()
LVWRIT(,)
LVEXCH
LVPAGW
LVCLOS

SUBROUTINE LVWRIT(NBIAS,NUMBLK)

APPENDIX C SUBROUTINE LISTINGS

```

C
C
C
0001 SUBROUTINE LVFDEX (INDEX, INXDAD, KFUNC, KARG, SAVCON)
0002 IMPLICIT INTEGER (A-Z)
0003 LOGICAL*1 SNGIBK, SETUP, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR, DL1STR,
1 DL2STR, DUMPF1, CURENT, FINDFI, DLETF1, NSRTFI, FD1TMP,
2 DL2TMP, IN2TMP, FD2TMP, REORG, FULL, LSTCON, RPLACE
0004 COMMON /I VARG/ IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, ITESTR,
1 INCLUD, INDXON, IVALS(10), ITYP1(10), SRCSUF,
2 LNKSUF, SNKSUF, INSTYP
0005 COMMON /I VREGS/ CURPAG(4), REQPG(4), LSTVPG(4), MSARET,
1 HREQPG, NXTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0006 COMMON /I VMASK/ MWRITE, NOTUSD, NEWCON, FI GMSK, MASKSF, MASKPF
0007 COMMON /I VFI AG/ FI 0MSK, FI 1MSK, FI 2MSK, FI 3MSK, FI 4MSK, FI 5MSK, FI 67,
1 FI AG8, FI AG9, FI AG10, FI AG11, FI AG12, FI AG13, FI AG14,
2 FI AG15
0008 COMMON /I VCRNT/ REGASP, CTRI PT, CTRI 1, LEASTV, NTFREE, FREE, DREGSP,
1 MSA, PAGLOC, CURENT
0009 COMMON /I VBUFR/ PAGESZ, NWCHAN, OI CHAN, CMPAND, PAGHDR, BUFSZE, DIRSZF,
1 INCORE, HDRSZF, MSADIR, SUFSZE, BIKSZF, DIRBI K, PAGHD4
0010 COMMON /I VHDVI/ THMSA, REGAS, PAGENO, CONTNO, INSDPL,
1 USECT, HDRFIG, READVI, OI DNPH, DNODEH, NROWH, DROWH
0011 COMMON /I VSWIT/ SETUP, SNGIBK, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
1 DL1STR, DL2STR, IN2TMP, FD2TMP, DL2TMP, DUMPF1,
2 FINDFI, DLETF1, NSRTFI
0012 COMMON /I VPRAM/ BUFI OC, LENGTH, IERR, ERRNUM, BINARY, BCD, MODE, PAGES,
1 LUN
0013 COMMON /I VFNP/ COUNT, ABSPOS, LSTCON
0014 COMMON /I VINS1/ REORG, FULL, RPLACE
0015 COMMON /I VVTR1/ NOPSPC(1)
1 /I VVTR2/ LSTSPC(1)
2 /I VVTR3/ LNKSPC(1)
3 /I VVTR4/ FIGSPC(1)
C
D PAUSE 'IN LVFDEX'
C THE PURPOSE OF THE FIND EXECUTIVE ROUTINE IS TO BRING THE PROPER
C CONTINUANT INTO THE BUFFER. IF THE PROGRAMMER DOES NOT SPECIFY THE
C CONTINUANT, ALL OF THE CONTINUANTS OF THAT PAGE WILL BE SEARCHED UNTIL
C EITHER A VALUE IS FOUND OR ALL OF THE CONTINUANTS HAVE BEEN LOOKED AT.
C IF THE CONTINUANT HAS BEEN SPECIFIED, THE SEARCH WILL PROCEED
C TO THE NEXT CONTINUANT ONLY IF FLAG 11 HAS BEEN SET FOR THAT LIST.
C IF THE CONTINUANT HAS BEEN SPECIFIED, THE SEARCH WILL PROCEED
C TO THE PREVIOUS CONTINUANT ONLY IF FLAG 10 HAS BEEN SET FOR THAT LIST.
C IF FD1STR IS .TRUE, USER STRATEGY ROUTINE USRFD1 PRECEEDS
C RETRIEVAL ACTION AND CONTINUES IF FINDFI IS .TRUE.
C USRFD2 IS CALLED AFTER THE RETRIEVAL IF FD2STR IS SET TO .TRUE.
C
0016 XXX=1000
0017 IF((FIGSPC(CTRI 1+REGASP).OR.FI 3MSK).NE.FI 3MSK)XXX=XXX*XXX
0019 REG=NOPSPC(DIRSZF+REGAS)
0020 USE=LNKSPC(DIRSZF+USECT)
C IF(USE.GT.440) XXX=XXX*XXX
C IF((FIGSPC(67).EQ.0).AND.(FIGSPC(144).EQ.0)) XXX=XXX*XXX
0021 COUNT = 0
0022 ABSPOS = IABS(IPOS)

```

```

C
0023 C CALL USER'S FIRST RETRIEVAL STRATEGY ROUTINE ?
      IF(FD1STR .EQ. .FALSE.) GO TO 100
C
C TO PREVENT RECURSION, INHIBIT FURTHER CALLS TO USER STRATEGY ROUTINES
0025     FD1TMP = FD1STR
0026     FD2TMP = FD2STR
0027     FD1STR = .FALSE.
0028     FD2STR = .FALSE.
0029     DL1TMP = DL1STR
0030     DL2TMP = DL2STR
0031     DL1STR = .FALSE.
0032     DL2STR = .FALSE.
0033     IN1TMP = IN1STR
0034     IN2TMP = IN2STR
0035     IN1STR = .FALSE.
0036     IN2STR = .FALSE.
C
C SET UP FOR FIRST USER ROUTINE
0037     CALL LVSTAC
0038     CALL USRFD1
0039     CALL LVPOP
0040     FD1STR = FD1TMP
0041     FD2STR = FD2TMP
0042     DL1STR = DL1TMP
0043     DL2STR = DL2TMP
0044     IN1STR = IN1TMP
0045     IN2STR = IN2TMP
C
C PROCEED WITH RETRIEVAL ?
0046     IF(FINDFI .EQ. .FALSE.) GO TO 600
C
C REQAG(2) IS SET IN CALLING PROGRAM. DEFAULT IS -2 ("ANY" CONTINUANT)
C SEPARATE PREFIX AND SUFFIX FROM SOURCE NODE (IARG) AND LINK (IFUNC)
0048     100 IF(IARG .IT. 2**SUFSZ) RETURN
0050         REQAG(1) = LVRTSH(IARG .AND. MASKPF, SUFSZ)
0051         SRCSUF = IARG .AND. MASKSF
0052         IF(IFUNC .IT. 2**SUFSZ) RETURN
0054         REQAG(3) = LVRTSH(IFUNC .AND. MASKPF, SUFSZ)
0055         LNKSUF = IFUNC .AND. MASKSF
C
C IS SAVED INDEX OPTION ON ?
0056     IF(NSKIP .EQ. 1) GO TO 150
0058     REQAG(2) = SAVCON
C
C REQAG(2) IS SET AT THE END OF LVFIND TO -2, IF IT IS NOT RESET BY THE
C PROGRAMMER FOR A RETRIEVAL, THEN THE REQUESTED CONTINUANT IS SET TO
C ZERO AND A SEARCH OF ALL CONTINUANTS IS ALLOWED.
C
0059     150 REQCON = REQAG(2)
0060     IF(REQAG(2) .EQ. -2) REQAG(2) = 0
C
C*** BRING THE REQUESTED PAGE, CONTINUANT INTO CORE.
C MAKE IT THE CURRENT PAGE, CONTINUANT.
C
0062     ITESTR = -1
0063     200 CALL LVEXCH

```

```

C
0064 C HAVE ALL CONTINUANTS OF REQ(PAGE) BEEN EXAMINED
      IF(MSARET .IE. 0) GO TO 600
C
C DESIRED PAGE, CONTINUANT IS NOW IN PLACE.
C ASSUME LIST DOES NOT CONTINUE BEYOND PRESENT CONTINUANT
0066 LSTCON = .FALSE.
C
C SEARCH FOR FUNCTION HEAD.
0067 CALL LVFIND
C
C FLAG CONTINUANT AS USED
0068 FIGSPC(CTRIPT+HDRFIG) = FIGSPC(CTRIPT+HDRFIG) .AND. .NOT. NOTUST
0069 LNKSPC(CTRIPT+USECT) = LNKSPC(CTRIPT+USECT) + 1
C
C HAS THE FUNCTION HEAD BEEN FOUND ?
0070 IF(ITESTR .GT. 0) GO TO 300
C
C IF THE CONTINUANT IS NOT SPECIFIED, EXAMINE NEXT CONTINUANT.
0072 IF(REQCON .NE. -2) GO TO 600
0074 REQAG(2) = REQAG(2) + 1
0075 GO TO 200
C
C FUNCTION HEAD FOUND
C SEARCH FROM TOP OR BOTTOM OF LIST ?
0076 300 IF(IPOS) 500,600,410
C
0077 400 CALL LVFIND
C
C DOES A PORTION OF THE CORRECT LIST RESIDE ON THIS CONTINUANT ?
0078 IF(ITESTR .LT. 0) GO TO 450
C
C BEGIN SEARCH DOWN THE LIST
0080 410 CALL LVFNV(INDEX,INDXAD,KFUNC,KARG,SAVCON)
C
C FLAG CONTINUANT AS USED
0081 FIGSPC(CTRIPT+HDRFIG) = FIGSPC(CTRIPT+HDRFIG) .AND. .NOT. NOTUST
0082 LNKSPC(CTRIPT+USECT) = LNKSPC(CTRIPT+USECT) + 1
C
C SUCCESSFUL RETRIEVAL ?
0083 IF(ITESTR .GT. 0) GO TO 600
C
C DOES THE LIST EXTEND TO ANOTHER CONTINUANT ?
0085 IF(LSTCON .EQ. .FALSE.) GO TO 600
C
C UPDATE REQUESTED CONTINUANT AND BRING INTO THE BUFFER
0087 450 REQAG(2) = REQAG(2) + 1
0088 CALL LVEXCH
C
C HAVE ALL CONTINUANTS OF REQ(PAGE) BEEN EXAMINED ?
0089 IF(MSARET .IE. 0) GO TO 600
0091 GO TO 400
C
C SEARCH FROM THE BOTTOM OF THE LIST
C BRING IN CONTINUANT CONTAINING LAST PORTION OF MVL
0092 500 CALL LVBOTM
0093 GO TO 530

```

```

0094 C 520 CALL LVFIND
C
0095 C DOES A PORTION OF THE CORRECT LIST RESIDE ON THIS CONTINUANT ?
      IF(ITESTR .IT. 0) GO TO 550
C
0097 C BEGIN SEARCH UP THE LIST
      530 CALL LVFNV(INDEX, INDXAD, KFUNC, KARG, SAVCON)
C
0098 C FLAG CONTINUANT AS USED
      FIGSPC(CTRIPT+HDRFLG) = FIGSPC(CTRIPT+HDRFLG) .AND. .NOT. NOTUST
0099 LNKSPC(CTRIPT+USECT) = LNKSPC(CTRIPT+USECT) + 1
C
0100 C SUCCESSFUL RETRIEVAL ?
      IF(ITESTR .GT. 0) GO TO 600
C
0102 C DOES THE LIST EXTEND TO ANOTHER CONTINUANT ?
      IF(LSTCON .EQ. .FALSE.) GO TO 600
C
0104 C UPDATE REQUESTED CONTINUANT
      550 REQPA(2) = REQPA(2) - 1
C
0105 C HAVE ALL CONTINUANTS OF REQ(PAGE) BEEN EXAMINED ?
      IF(REQPA(2) .IT. 0) GO TO 600
C
0107 C BRING REQ(P,C) INTO THE BUFFER
      CALL LVEXCH
0108 GO TO 520
C
0109 C CALL SECOND USER RETRIEVAL STRATEGY ROUTINE ?
      600 IF(FD2STR .EQ. .FALSE.) GO TO 700
0111 FD1TMP = FD1STR
0112 FD2TMP = FD2STR
0113 FD1STR = .FALSE.
0114 FD2STR = .FALSE.
0115 DL1TMP = DL1STR
0116 DL2TMP = DL2STR
0117 DL1STR = .FALSE.
0118 DL2STR = .FALSE.
0119 IN1TMP = IN1STR
0120 IN2TMP = IN2STR
0121 IN1STR = .FALSE.
0122 IN2STR = .FALSE.
0123 CALL LVSTAC
0124 CALL USRFD2
0125 CALL LVPOP
0126 FD1STR = FD1TMP
0127 FD2STR = FD2TMP
0128 DL1STR = DL1TMP
0129 DL2STR = DL2TMP
0130 IN1STR = IN1TMP
0131 IN2STR = IN2TMP
C
0132 C RESET 'REQUESTED CONTINUANT' DEFAULT TO 'ANY'
      700 REQPA(2) = -2
0133 C RESET TO DEFAULT VALUES
      IPOS = 1
0134 ITYP = 3
0135 RETURN
0136 END

```

```

C
C
C
0001 SUBROUTINE LVFIND
0002 IMPLICIT INTEGER(A-Z)
0003 LOGICAL*1 SNGIBK, SETUP, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR, DL1STR,
1 DL2STR, DUMPF1, CURENT, FINDFI, DLETFI, NSRTFI,
2 DL2TMP, IN2TMP, FD2TMP, REORG, FULL, LSTCON, RPLACE
0004 COMMON /I VARG/ IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, ITESTR,
1 INCIUD, INPXON, IVALS(10), ITYP1(10), SRCSTF,
2 LNKSUF, SNKSUF, INSTYP
0005 COMMON /I VREGS/ CURPAG(4), REQPA(4), LSTVPG(4), MSARET,
1 HREQPG, NYTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0006 COMMON /I VMASK/ MWRITE, NOTUSD, NEWCON, FI GMSK, MASKSF, MASKPF
0007 COMMON /I VFLAG/ FI 0MSK, FI 1MSK, FI 2MSK, FI 3MSK, FI 4MSK, FI 5MSK, FI 67,
1 FI AG8, FI AG9, FI AG10, FI AG11, FI AG12, FI AG13, FI AG14,
2 FI AG15
0008 COMMON /I VCRNT/ REGASP, CTRI PT, CTRI 1, LEASTV, NTFREE, FREE, DREGSP,
1 MSA, PAGLOC, CURENT
0009 COMMON /I VBUIFR/ PAGSZE, NWCHAN, OI CHAN, CMPAND, PAGHDR, BUFSZE, DIRSZE,
1 INCORE, HDRSZE, MSADIR, SUFSZE, BIKSZE, DIRBI K, PAGBD4
0010 COMMON /I VBDVI/ THMSA, REGAS, PAGENO, CONTNO, INSDI,
1 USECT, HDRFIG, READVI, OI DNMH, DNOPEH, NROWH, DROWH
0011 COMMON /I VSWIT/ SETUP, SNGIBK, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
1 DL1STR, DL2STR, IN2TMP, FD2TMP, DL2TMP, DUMPF1,
2 FINDFI, DLETFI, NSRTFI
0012 COMMON /I VPRAM/ BUFI OC, LENGTH, IERR, ERRNUM, BINARY, BCD, MODE, PAGES,
1 LUN
0013 COMMON /I VADDR/ IADD, THIS, LSTHED, LOC, LAST, LASTLC
0014 COMMON /I VFND/ COUNT, ABSPOS, LSTCON
0015 COMMON /I VINS1/ REORG, FULL, RPLACE
0016 COMMON /I VVTR1/ NODSPC(1)
1 /I VVTR2/ LSTSPC(1)
2 /I VVTR3/ LNKSPC(1)
3 /I VVTR4/ FIGSPC(1)
C
C IADD = (RELATIVE) COMPUTED FUNCTION ADDRESS
C THIS = (RELATIVE) LOCATION OF FUNCTION ON CONFLICT LIST
C LOC = (RELATIVE) LOCATION OF RETRIEVED VALUE
C LSTHED = -1, SINGLE VALUED LIST
C = 0, NO LIST IS FOUND
C > 0, (RELATIVE) ADDRESS OF FIRST VALUE
C ITESTR = 1, RETRIEVAL IS SUCCESSFUL (IVAL = RETURNED VALUE)
C = -1, RETRIEVAL IS FAILURE (IVAL = SOURCE NODE)
C
C
D PAUSE 'IN LVFIND'
0017 ITESTR = 1
0018 IADD = SRCSTF + LNKSUF
0019 IF(IADD .GT. PAGSZE) IADD = IADD-PAGSZE
0021 IF(IADD .IE. PAGSZE) GO TO 2
C
C IFUNC OR IARG ARE INCORRECT, STOP
0023 TYPE 3, IFUNC, IARG
0024 3 FORMAT(//, ' ****ERROR**** LINK ',15, ' OR SOURCE NODE ',15, ' ARE
1 UNDEFINED',/)
0025 ERRNUM = 40

```

```

0026      DUMP = 0
0027      CALL LVERR(DUMP)
0028      STOP
C
0029      2      LSTHED = 0
0030      THIS = IADD
0031      IF((FIGSPC(CTRI1 + THIS) .AND. FLSMSK) .EQ. 0) GO TO 99
C
C SEARCH CONFLICT LIST FOR KEY (IFUNC OR LINK)
0033      1      IF(NOPSPC(CTRI1 + THIS) .EQ. IFUNC) GO TO 4
0035      LAST = THIS
0036      THIS = LNKSPC(CTRI1 + THIS)
0037      IF((FIGSPC(CTRI1 + THIS) .AND. FLSMSK) .NE. 0) GO TO 99
0039      GO TO 1
C
C THE FUNCTION HAS BEEN FOUND.
C TEST FOR SINGLE VALUE LIST (SVL) OR MULTIVALUED LIST (MVL).
0040      4      IF((FIGSPC(CTRI1 + THIS) .AND. FLMSK) .NE. 0) GO TO 14
C
C SINGLE VALUED LIST.
0042      LSTHED = -1
0043      LOC = THIS
0044      IVAL = LSTSPC(CTRI1 + LOC)
0045      RETURN
C
C MULTIVALUED LIST. OBTAIN FIRST VALUE.
0046      14     LSTHED = LSTSPC(CTRI1 + THIS)
0047      LOC = LSTHED
0048      IVAL = NOPSPC(CTRI1 + LOC)
0049      LASTLC = LNKSPC(CTRI1 + LSTHED)
0050      RETURN
C
C FUNCTION IS NOT ON THIS CONTINUANT
0051      99     ITESTR = -1
0052      IVAL = IARG
0053      RETURN
0054      END

```

```

C
C
0001 SUBROUTINE LVFNV (INDEX, INDXAD, KFUNC, KARG, SAVCON)
0002 IMPLICIT INTEGER (A-Z)
0003 LOGICAL*1 SNGIBK, SETUP, NYTRAN, IN1STR, IN2STR, FD1STR, FD2STR, DL1STR,
1 DL2STR, DUMPF1, CURENT, FINDFI, DLETFI, NSRTFI,
2 DL2TMP, IN2TMP, FD2TMP, REORG, FULL, LSTCON, RPLACE
0004 COMMON /IVARGS/ IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, ITESTR,
1 INCIUD, INDXON, IVALS(10), ITYP1(10), SRCSUF,
2 LNKSUF, SNKSUF, INSTYP
0005 COMMON /IVREGS/ CURPAG(4), RFOPAG(4), LSTVPG(4), MSARET,
1 HREOPG, NYTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0006 COMMON /IVMASK/ MWRITE, NOTUSD, NEWCON, FIGMSK, MASKSF, MASKPF
0007 COMMON /IVFIAG/ FI0MSK, FI1MSK, FI2MSK, FI3MSK, FI4MSK, FI5MSK, FI667,
1 FIAG8, FIAG9, FIAG10, FIAG11, FIAG12, FIAG13, FIAG14,
2 FIAG15
0008 COMMON /IVCRNT/ REGASP, CTRIPT, CTRI1, LEASTV, NTFREE, FREE, DREGSP,
1 MSA, PAGLOC, CURENT
0009 COMMON /IVBUFR/ PAGSIZE, NWCHAN, OLCHAN, CMPAND, PAGHDR, BUFSZE, DIRSZE,
1 INCORE, HDRSZE, MSADIR, SUFSZE, BIKSZE, DIRBIK, PAGHD4
0010 COMMON /IVHDI/ THMSA, REGAS, PAGENO, CONTNO, INSDI,
1 USECT, HDRFIG, RFADVI, OINDPH, DNOPEH, NROWH, DROWH
0011 COMMON /IVSWIT/ SETUP, SNGIBK, NYTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
1 DL1STR, DL2STR, IN2TMP, FD2TMP, DL2TMP, DUMPF1,
2 FINDFI, DLETFI, NSRTFI
0012 COMMON /IVPRAM/ BUFILOC, LENGTH, IERR, ERRNUM, BINARY, BCD, MODE, PAGES,
1 LUN
0013 COMMON /IVADDR/ IADD, THIS, LSTHED, LOC, LAST, LASTLC
0014 COMMON /IVFND/ COUNT, ABSPOS, LSTCON
0015 COMMON /IVINS1/ REORG, FULL, RPLACE
0016 COMMON /IVVTR1/ NODSPC(1)
1 /IVVTR2/ LSTSPC(1)
2 /IVVTR3/ LNKSPC(1)
3 /IVVTR4/ FIGSPC(1)
0017 DATA NFIAG4/ 177767/

C
C LVFIND MUST BE CALLED IMMEDIATELY PRIOR TO THE CALL TO THIS ROUTINE
C INPUT IS EXPECTED THRU COMMONS IVARGS, LVFND, AND LVADDR. THIS ROUTINE
C SEARCHES THE MULTIVALUE LIST FOR THE IPOS'TH VALUE OF THE REQUESTED
C TYPE. IF SVI, TYPE MUST BE EITHER UNSPECIFIED OR CORRECT.
C
C DOES THE FUNCTION EXIST ?
D PAUSE 'IN LVFNV'
0018 IF (ITESTR .LT. 0) GO TO 700
0020 IF (LSTHED .GT. 0) GO TO 100
C SVI - DOES FUNCTION QUALIFY ?
0022 IF (ABSPOS .NE. 1) GO TO 699
0024 IF (ITYP .EQ. 3) GO TO 700
0026 ISTYP = (FIGSPC(CTRI1 + LOC) .AND. FI667)
0027 IF (ISTYP .EQ. 3) ISTYP = 2
0029 IF (ISTYP .NE. ITYP) GO TO 699
0031 GO TO 700

C
C MVI - FIRST VALUE HAS ALREADY BEEN FOUND BY LVFIND
0032 100 IF (IPOS .EQ. 1 .AND. ITYP .EQ. 3) GO TO 500

```

```

C *** BEGIN SEARCH
C IF THE SAVED INDEX FACILITY IS NOT TO BE USED. GO TO 200
0034 120 IF(NSKIP .EQ. 1) GO TO 200
0036 IF(INDEX .EQ. 0) GO TO 200
C
C SAVED INDEX CAN'T BE USED IF IMMEDIATE PAST HISTORY = 1 EXED
C INSERTION OR DELETION.
0038 IF((FIGSPC(CTRI1 + THIS) .AND. FI4MSK) .NE. 0) GO TO 200
C
C SAVED INDEX CAN'T BE USED IF SOURCE NODE OR LINK HAVE BEEN CHANGED
0040 IF((KFUNC .NE. IFUNC) .OR. (KARG .NE. IARG)) GO TO 200
C
C SAVED INDEX CAN'T BE USED IF DIRECTION OF SEARCH HAS SWITCHED
0042 IF((IPOS*INDEX) .IE. 0) GO TO 200
0044 NDX = FIGSPC(CTRI1 + INDXAD)
C
C SAVED INDEX CAN'T BE USED IF VALUE AT SAVED INDEX HAS BEEN MOVED
0045 IF((NDX .AND. FI5MSK) .NE. 0) GO TO 200
C
C SAVED INDEX CAN'T BE USED IF VALUE AT SAVED INDEX HAS BEEN REMOVED
0047 IF((NDX .AND. FI1MSK) .EQ. 0) GO TO 200
C
C IS SEARCH FROM BEGINNING FASTER THAN FROM SAVED INDEX ?
0049 KNDEX = IABS(INDEX)
0050 IF(ABSPOS .IT. 2) GO TO 200
0052 IF(ABSPOS*ABSPOS) .IE. KNDEX) GO TO 200
C
C SAVED INDEX CAN BE USED. BEGIN SEARCH AT INDXAD.
0054 LOC = INDXAD
C FIND RELATIVE DISTANCE FROM SAVED INDEX AND DETERMINE WHETHER TO
C COUNT UP OR DOWN. IF REQUESTED POSITION IS CLOSER TO THE BEGINNING
C OF THE LIST THAN THE SAVED INDEX, COUNT UP, OTHERWISE, COUNT DOWN.
C
0055 LENGTH = INDEX-IPOS
0056 ABSPOS = IABS(LENGTH)
0057 IF(LENGTH) 300,450,170
C
C COUNT UP FROM INDXADD
C
0058 170 ITOP = 0
0059 GO TO 420
C
C DO NOT USE SAVED INDEX. START FROM THE BEGINNING OR END OF LIST
C
0060 200 FIGSPC(CTRI1 + THIS) = FIGSPC(CTRI1 + THIS) .AND. NFLAG4
0061 IF(IPOS) 400,699,320
C
C COUNT DOWN
C
0062 300 LASTLC = LOC
0063 LOC = LSTSPC(CTRI1 + LOC)
0064 IF((FIGSPC(CTRI1 + LOC) .AND. FI0MSK) .NE. 0) GO TO 600
0066 320 IF(ITYP .EQ. 3) GO TO 330
0068 ISTYP = (FIGSPC(CTRI1 + LOC) .AND. FI067)
0069 IF(ISTYP .EQ. 3) ISTYP = 2
0071 IF(ISTYP .NE. ITP) GO TO 300

```

```

0073 330 COUNT = COUNT+1
0074 IF(COUNT .NE. ABSPOS) GO TO 300
0076 GO TO 450
C
COUNT UP FROM THE BOTTOM OF THE LIST
C
0077 400 ITOP = 1
0078 420 LOC = LNKSPC(CTRL1 + LOC)
0079 IF(ITOP .EQ. 1) GO TO 430
0081 IF((FIGSPC(CTRL1 + LSTSPC(CTRL1 + LOC)) .AND. FIOMSK) .NE. 0)
1 GO TO 650
0083 430 ITOP = 0
0084 IF(ITYP .EQ. 3) GO TO 440
0086 ISTYP = (FIGSPC(CTRL1 + LOC) .AND. FIG67)
0087 IF(ISTYP .EQ. 3) ISTYP = 2
0089 IF(ISTYP .NE. ITYP) GO TO 420
0091 440 COUNT = COUNT+1
0092 IF(COUNT .NE. ABSPOS) GO TO 420
0094 450 IVAL = NONSPC(CTRL1 + LOC)
C
C SAVE INDEX PARAMETERS AFTER SUCCESSFUL RETRIEVAL
C
0095 500 IF(NSKIP .EQ. 1) GO TO 700
0097 KARG = IARG
0098 KFUNC = IFUNC
0099 INDXAD = LOC
0100 INDEX = IPOS
0101 SAVCON = CURPAG(2)
0102 GO TO 700
C
C POSSIBLE FAILURE. DOES MVI EXTEND FORWARD TO ANOTHER CONTINUANT
0103 600 IF((FIGSPC(CTRL1 + LASTLC) .AND. FLAG11) .EQ. 0) GO TO 699
0105 LSTCON = .TRUE.
0106 GO TO 699
C
C POSSIBLE FAILURE. DOES MVI EXTEND BACKWARD TO ANOTHER CONTINUANT
0107 650 IF((FIGSPC(THIS) .AND. FLAG10) .EQ. 0) GO TO 699
0109 LSTCON = .TRUE.
C
C FAILURE EXIT
0110 699 ITESTR = -1
0111 IF(NSKIP .EQ. 0) INDEX = 0
0113 IVAL = IARG
C
C SUCCESS EXIT, SET DEFAULTS.
0114 700 ITYP = 3
0115 RETURN
0116 END

```

```

C
C
0001 SUBROUTINE LVBOTM
0002 IMPLICIT INTEGER(A-Z)
0003 LOGICAL*1 SNGIBK, SETUP, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR, DL1STR,
1 DL2STR, DUMPF1, CURENT, FINDFI, DLETFI, NSRTFI, BAKCON,
2 DL2TMP, IN2TMP, FD2TMP, REORG, FULL, LSTCON, RPLAC
0004 COMMON /I VARGS/ IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, ITESTR,
1 INCIUD, INDXON, IVALS(10), ITYP1(10), SRCSUF,
2 LNKSUF, SNKSUF, INSTYP
0005 COMMON /I VREGS/ CURPAG(4), REQPG(4), LSTVPG(4), MSARET,
1 HREQPG, NXTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0006 COMMON /I VMASK/ MWRITE, NOTUSD, NEWCON, FLGMSK, MASKSF, MASKPF
0007 COMMON /I VFIAG/ FI0MSK, FI1MSK, FI2MSK, FI3MSK, FI4MSK, FI5MSK, FI667,
1 FIAG8, FIAG9, FIAG10, FIAG11, FIAG12, FIAG13, FIAG14,
2 FIAG15
0008 COMMON /I VCRNT/ REGASP, CTRIPT, CTRI1, LEASTV, NTFREE, FREE, DREGSP,
1 MSA, PAGLOC, CURENT
0009 COMMON /I VBUPR/ PAGSZE, NWCHAN, OICHAN, CMPAND, PAGHDR, BUFSZE, DIRSZE,
1 INCORE, HDRSZE, MSADIR, SUFSZE, BIKSZE, DIRBIK, PAGHD4
0010 COMMON /I VHDVI/ THSMSA, REGAS, PAGENO, CONTNO, INSDLE,
1 USECT, HDRFIG, READVI, OI DNDH, DNOPEH, NROWH, DROWH
0011 COMMON /I VSWIT/ SETUP, SNGIBK, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
1 DL1STR, DL2STR, IN2TMP, FD2TMP, DL2TMP, DUMPF1,
2 FINDFI, DLETFI, NSRTFI
0012 COMMON /I VPRAM/ BUFILOC, LENGTH, IERR, ERRNUM, BINARY, BCD, MODE, PAGES,
1 LUN
0013 COMMON /I VADDR/ IADD, THIS, LSTHED, LOC, LAST, LASTLC
0014 COMMON /I VFND/ COUNT, ABSPOS, LSTCON
0015 COMMON /I VINS1/ REORG, FULL, RPLAC
0016 COMMON /I VDEL1/ NUMRET, BAKCON
0017 COMMON /I VVTR1/ NORSPC(1)
1 /I VVTR2/ LSTSPC(1)
2 /I VVTR3/ LNKSPC(1)
3 /I VVTR4/ FLGSPC(1)
C
C THIS ROUTINE BRINGS INTO THE BUFFER THE LAST CONTINANT OF A PAGE WHICH
C CONTAINS A PORTION OF THE REQUESTED LIST.
C ASSUME THAT THE BUFFER CONTAINS THE CONTINANT WHICH HOLDS THE FIRST
C PORTION OF THE MVL.
C
C D PAUSE 'IN LVBOTM'
0018 TMPREQ = CURPAG(2)
C SVI ?
0019 100 IF(LSTHED.GT. 0) GO TO 120
0021 LASTLC = THIS
0022 GO TO 140
C
C GET FIRST "VALUE" ON MVL
0023 120 ISTLOC = LSTSPC(CTRI1 + THIS)
C
C GET LAST "VALUE" ON MVL
0024 LASTLC = LNKSPC(CTRI1 + ISTLOC)
C
C DOES THE LIST END ON THIS CONTINANT ?

```

```

0025 140 IF((FLGSPC(CTRI1 + LASTLC) .AND. FLAG11) .EQ. 0) RETURN
0027 LSTCON = .TRUE.
C
C EXAMINE NEXT (SEQUENTIAL) CONTINUANT FOR A PORTION OF THE MVI
0028 200 REQPA(2) = REQPA(2) + 1
0029 CALL LVEXCH
C
C ERROR IF SET OF CONTINUANTS IS EXHAUSTED
0030 IF(MSARET .GT. 0) GO TO 250
C NO ERROR IF SEARCH ORIGINATED FROM LVINLEX
0032 IF(BAKCON .EQ. .FALSE.) GO TO 220
0034 REQPA(2) = TMPREQ
0035 RETURN
C
0036 220 ERRNUM = 42
0037 DUMP = 0
0038 CALL LVERR(DUMP)
0039 STOP
C
C DOES THIS CONTINUANT CONTAIN A PORTION OF THE MVI ?
0040 250 CALL LVFIND
0041 IF(ITESTR .LT. 0) GO TO 200
0043 TMPREQ = REQPA(2)
0044 GO TO 100
0045 END

```

```

C
C
0001 SUBROUTINE LVINEX
0002 IMPLICIT INTEGER(A-Z)
0003 LOGICAL*1 SNGIBK, SETUP, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR, DL1STR,
1 DL2STR, DUMPF1, CURENT, FINDF1, DLETF1, NSRTF1, FD1TMP,
2 DL2TMP, IN2TMP, FD2TMP, INSIDE, FULL, REORG, LSTCON, NXTCON,
3 RPLACE
0004 COMMON /IVARGS/ IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, ITESTR,
1 INCLUD, INPXON, IVALS(10), ITYP1(10), SRCSUF,
2 LNKSUF, SNKSUF, INSTYP
0005 COMMON /IVREGS/ CURPAG(4), REQPG(4), LSTVPG(4), MSARET,
1 HREQPG, NXTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0006 COMMON /VMASK/ MWRITE, NOTUSD, NEWCON, FIGMSK, MASKSF, MASKPF
0007 COMMON /VFLAG/ F10MSK, F11MSK, F12MSK, F13MSK, F14MSK, F15MSK, F16G7,
1 FLAG8, FLAG9, FLAG10, FLAG11, FLAG12, FLAG13, FLAG14,
2 FLAG15
0008 COMMON /VCRNT/ REGASP, CTR1PT, CTR11, LEASTV, NTFREE, FREE, DREGSP,
1 MSA, PAGLOC, CURENT
0009 COMMON /VBUFR/ PAGSZE, NWCHAN, OICHAN, CMPAND, PAGHDR, BUFSZE, DIRSZE,
1 INCORE, HDRSZE, MSADIR, SUFSZE, BIKSZE, DIRBIK, PAGHD4
0010 COMMON /VHDVI/ THMSA, REGAS, PAGENO, CONTNO, INSDI,
1 USECT, HDRFIG, READVI, OINDPH, DNODEH, NROWH, DROWH
0011 COMMON /VSWIT/ SETUP, SNGIBK, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
1 DL1STR, DL2STR, IN2TMP, FD2TMP, DL2TMP, DUMPF1,
2 FINDF1, DLETF1, NSRTF1
0012 COMMON /VPRAM/ BUFOC, LENGTH, IERR, ERRNUM, BINARY, BCD, MODE, PAGES,
1 LUN
0013 COMMON /VADDR/ IADD, THIS, LSTHED, LOC, LAST, LASTLC
0014 COMMON /VFNDF/ COUNT, ABSPOS, LSTCON
0015 COMMON /VINS1/ REORG, FULL, RPLACE
0016 COMMON /VVTR1/ NOPSPC(1)
1 /VVTR2/ LSTSPC(1)
2 /VVTR3/ LNKSPC(1)
3 /VVTR4/ FIGSPC(1)
C
0017 DATA INSIDE /.FALSE./
C
C THE INSERT EXECUTIVE ROUTINE COMPLETES THE TRIPLE IF NECESSARY AND
C OBTAINS THE CORRECT P,C FOR SUBROUTINE LVNSRT TO OPERATE ON.
C
C IS LVNSRT BEING CALLED FROM AN INSERT STRATEGY ROUTINE ?
D PAUSE 'IN LVINEX'
0018 IF(INSIDE .EQ. .TRUE.) GO TO 100
C
C TO PREVENT RECURSION, SAVE THE FIND STRATEGY FLAGS AND TURN THEM OFF
0020 FD1TMP = FD1STR
0021 FD2TMP = FD2STR
0022 FD1STR = .FALSE.
0023 FD2STR = .FALSE.
C
C CALL USER'S FIRST INSERT STRATEGY ROUTINE ?
0024 IF(IN1STR .EQ. .FALSE.) GO TO 100
C
C TO PREVENT RECURSION, INHIBIT CALLS TO ALL USER STRATEGY ROUTINES

```

```

0026      INITMP = INISTR
0027      IN2TMP = IN2STR
0028      INISTR = .FALSE.
0029      IN2STR = .FALSE.
0030      DL1TMP = DL1STR
0031      DL2TMP = DL2STR
0032      DL1STR = .FALSE.
0033      DL2STR = .FALSE.
C
C SET UP FOR FIRST USER ROUTINE
0034      CALL LVSTAC
0035      INSIDE = .TRUE.
0036      CALL USRINI
0037      INSIDE = .FALSE.
0038      CALL LVPOP
0039      INISTR = INITMP
0040      IN2STR = IN2TMP
0041      DL1STR = DL1TMP
0042      DL2STR = DL2TMP
C
C PROCEED WITH INSERTION ?
0043      IF(NSRTFI .EQ. .FALSE.) GO TO 1000
C
C *** ENSURE THAT THE TRIPLE IS COMPLETELY DEFINED ***
C *** BRING IN REQ(P,C), DEFINE AS CURRENT(P,C)
C TEST SOURCE NODE
C      IARG = -1      PLACE ON NEW PAGE AND DEFINE
C      = 0           PLACE ON CURRENT PAGE AND DEFINE
C      >= 2**SUFSZ, ALREADY DEFINED, SEPARATE PREFIX AND SUFFIX
C      =N<2**SUFSZ, PLACE ON PAGE N AND DEFINE
0045      100  NXTCON = .FALSE.
0046      REQCON = REQPA(2)
0047      IF(IARG .EQ. -1) REQCON = -1
0049      IF(IARG) 110,120,130
C
C PLACE ON NEW PAGE (CONT = 0)
0050      110  CALL LVNPAG
C
C PLACE ON CURRENT PAGE (AND CONT) AND DEFINE SUFFIX
0051      120  CALL LVGRN(SRCSUF)
C
C RECONSTRUCT IARG
0052      IARG = LVIFSH(CURPAG(1),SUFSZ) .OR. SRCSUF
0053      GO TO 200
C
C A SPECIFIC PAGE IS REQUESTED
C IS ONLY THE SUFFIX DEFINED ? (IF SO, IT IS A PAGE REQUEST W/O SUF)
0054      130  IF(IARG .GE. 2**SUFSZ) GO TO 140
0056      REQPA(1) = IARG
0057      GO TO 145
0058      140  REQPA(1) = LVRTSH(IARG,SUFSZ)
C
C IMPROPER PAGE REQUEST ?
0059      IF(REQPA(1) .IE. HACTPG(1)) GO TO 145
0061      ERRNUM = 60
0062      DUMP = 0
0063      CALL LVERR(DUMP)

```

```

0064      STOP
C
C TEST "REQUESTED CONTINUANT"
C   REQ PAG(2) >= 0, CONTINUANT SPECIFIED
C   -1, NEW CONTINUANT
C   -2, ANY CONTINUANT
C   -3, CURRENT CONTINUANT IF CURRENT PAGE = REQUESTED PAGE
0065 145 IF(REQ PAG(2) .GE. 0) GO TO 155
0067      IF(REQ PAG(2) + 2) 150,160,170
C
C CURRENT CONTINUANT OF REQUESTED PAGE IF ALSO CURRENT PAGE
0068 150 REQ PAG(2) = CUR PAG(2)
0069      IF(REQ PAG(1) .NE. CUR PAG(1)) REQ PAG(2) = 0
C
C BRING IN PROPER CONTINUANT
0071 155 CALL LVEXCH
0072 157 IF(IARG .GE. 2**SUF SIZE) GO TO 200
0074      GO TO 120
C
C CONTINUANT NOT SPECIFIED, SET TO ZERO
0075 160 REQ PAG(2) = 0
0076      GO TO 155
C
C CREATE NEW CONTINUANT
0077 170 CALL LVNCON
0078      GO TO 157
C
C TEST IFUNC AND RECONSTRUCT IF NECESSARY
0079 200 IF(IFUNC .GE. 2**SUF SIZE) GO TO 300
0081      TEMPAG = REQ PAG(1)
0082      REQ PAG(1) = IFUNC
0083      IF(IFUNC .EQ. 0) REQ PAG(1) = CUR PAG(1)
0085      REQ PAG(3) = REQ PAG(1)
0086      CALL LVGRN(LNKSUF)
0087      IFUNC = LVIFSH(REQ PAG(3),SUF SIZE) .OR. LNKSUF
0088      REQ PAG(1) = TEMPAG
C
C TEST THE SINK NODE AND RECONSTRUCT IF NECESSARY
C   RANDOM NUMBER ?
0089 300 IF(ITYP1(1) .NE. 0) GO TO 400
0091      IF(IVAL S(1) .GE. 2**SUF SIZE) GO TO 400
0093      TEMPAG = REQ PAG(1)
0094      IF(IVAL S(1)) 310,320,330
C
C SINK NODE POINTS TO NEW PAGE
0095 310 REQ PAG(4) = HACTPG(1) + 1
0096      GO TO 340
C
C SINK NODE POINTS TO CURRENT PAGE
0097 320 REQ PAG(4) = CUR PAG(1)
0098      GO TO 340
C
C SINK NODE POINTS TO DIFFERENT PAGE
0099 330 REQ PAG(4) = IVAL S(1)
0100 340 REQ PAG(1) = REQ PAG(4)
0101      CALL LVGRN(SNKSUF)
0102      REQ PAG(1) = TEMPAG

```

```

0103      IVALS(1) = LVIFSH(REQPAG(4),SUFSSZE) .OR. SNKSUF
      C
      C BEGIN SEARCH FOR EXISTING LIST
0104 400 FULL = .FALSE.
0105      IF(INDXON .NE. 0) GO TO 420
      C
      C "NORMAL" INSERTION, FIND BOTTOM OF LIST
0107 410 IPOS = -1
0108 420 TEMPOS = IPOS
0109      TMPREQ = REQPAG(2)
0110      CALL LVFDEX
0111      REQPAG(2) = TMPREQ
0112      IPOS = TEMPOS
      C
      C LIST FOUND ON REQUESTED CONTINUANT ?
0113      IF(ITESTR .GT. 0) GO TO 500
      C
      C REQUEST FOR NEW CONTINUANT (OR PAGE) ?
0115      IF(REQCON .EQ. -1) GO TO 500
      C
      C PLACE NEW LIST ACCORDING TO REQPAG(2) BUT FIRST SEARCH ELSEWHERE
0117      IPOS = TEMPOS
0118      TMPREQ = REQPAG(2)
0119      REQPAG(2) = -2
0120      CALL LVFDEX
0121      IPOS = TEMPOS
0122      REQPAG(2) = TMPREQ
      C
      C FOUND LIST ON DIFFERENT CONTINUANT ?
0123      IF(ITESTR .GT. 0) GO TO 450
      C
      C LIST DOES NOT EXIST ON ANY CONTINUANT
0125      IPOS = TEMPOS
0126      CALL LVFDEX
0127      IPOS = TEMPOS
0128      REQPAG(2) = TMPREQ
0129      GO TO 500
      C
      C IF CONT WAS NOT SPECIFIED, LIST FOUND ON NON-ZERO' TH CONTINUANT
      C OTHERWISE, LIST WAS FOUND ON THE "WRONG" CONTINUANT AND MVI MUST
      C BE REORGANIZED AND PLACED ON REQCON
0130 450 IF(REQCON .EQ. -2) GO TO 500
0132      REORG = .TRUE.
0133      CALL LVREOR(REQCON)
0134      REORG = .FALSE.
0135      GO TO 800
      C
      C PERFORM INSERTION
0136 500 CALL LVNSRT
0137      IF(.NOT. FULL) GO TO 800
      C
      C CONTINUANT IS FULL, PLACE ON NEXT CONTINUANT IF SPACE IS AVAILABE
0139      FULL = .FALSE.
      C
      C SPECIAL HANDLING FOR OVERFLOW ON INDEXED INSERTION
0140      IF(INDXON .NE. 0) GO TO 600
      C

```

```

0142 C DOES A PORTION OF THE MVI RESIDE ON THE CURRENT CONTINUANT ?
505 IF(ITESTR .IE. 0) GO TO 520
C
C SET MVI CONTINUATION FLAG
0144 FIGSPC(CTRL1+LASTLC)=FIGSPC(CTRL1+LASTLC) .OR. FLAG11
0145 NXTCON = .TRUE.
0146 520 REOPAG(2) = REOPAG(2) + 1
0147 FIGSPC(CTRIPT+HDRFIG)=FIGSPC(CTRIPT+HDRFIG) .AND. .NOT. NOTUSD
1 .OR. MWRITE
0148 CALL LVEXCH
0149 IF(WSARET .IE. 0) CALL LVNCON
0151 CALL LVFIND
0152 GO TO 505
C
C OVERFLOW ON INDEXED INSERTION
0153 600 IF(IPOS .EQ. -1) GO TO 505
0155 CALL LVOVER
0156 GO TO 505
C
C RESET CONTINUANT USAGE RATIO
0157 800 IF(RPLACE .EQ. .TRUE.) GO TO 820
0159 LNKSPC(CTRIPT+INSEL) = LNKSPC(CTRIPT+INSEL) + NVAL
C
C CONTINUANT HAS BEEN MODIFIED
0160 820 FIGSPC(CTRIPT+HDRFIG)=FIGSPC(CTRIPT+HDRFIG) .AND. .NOT. NOTUSD
1 .OR. MWRITE
0161 NODSPC(CTRIPT+REGAS) = REGASP
C
C IF LIST IS CONTAINED ON MORE THAN ONE CONTINUANT,
C SET BACK POINTING FLAG
0162 IF(NXTCON .EQ. .TRUE.)
1 FIGSPC(CTRL1 + THIS) = FIGSPC(CTRL1 + THIS) .OR. FLAG10
C
C IF A SPECIFIC CONTINUANT WAS REQUESTED, SET REORG INHIBIT FLAG
0164 IF(REQCON .NE. -2)
1 FIGSPC(CTRL1 + THIS) = FIGSPC(CTRL1 + THIS) .OR. FLAG12
0166 REOPAG(2) = -2
C
C CALL SECOND USER INSERTION STRATEGY ROUTINE ?
0167 1000 IF(IN2STR .EQ. .FALSE.) GO TO 1100
0169 IN1TMP = IN1STR
0170 IN2TMP = IN2STR
0171 IN1STR = .FALSE.
0172 IN2STR = .FALSE.
0173 DL1TMP = DL1STR
0174 DL2TMP = DL2STR
0175 DL1STR = .FALSE.
0176 DL2STR = .FALSE.
0177 CALL LVSTAC
0178 INSIDE = .TRUE.
0179 CALL USRIN2
0180 INSIDE = .FALSE.
0181 CALL LVPOP
0182 IN1STR = IN1TMP
0183 IN2STR = IN2TMP
0184 DL1STR = DL1TMP
0185 DL2STR = DL2TMP
C
C RESTORE FIND STRATEGY FLAGS
0186 1100 IF(INSIDE .EQ. .TRUE.) RETURN
0188 FD1STR = FD1TMP
0189 FD2STR = FD2TMP
0190 RETURN
0191 END

```

```

C
C
0001 SUBROUTINE LVNSRT
0002 IMPLICIT INTEGER(A-Z)
0003 LOGICAL*1 SNGIBK, SETUP, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR, DL1STR,
1 DL2STR, DUMPFI, CURENT, FINDFI, DLETFI, NSRTFI, IN1TMP,
2 DL2TMP, IN2TMP, FD2TMP, FULL, REORG, LSTCON, RPLCE
0004 COMMON /IVARGS/ IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, ITESTR,
1 INCIUD, INDXON, IVALS(10), ITYP1(10), SRCSUF,
2 LNKSUF, SNKSUF, INSTYP
0005 COMMON /IVREGS/ CURPAG(4), REOPAG(4), LSTVPG(4), MSARET,
1 HREQPG, NXTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0006 COMMON /VWASK/ NWRITE, NOTUSD, NEWCON, FI1MSK, MASKSF, MASKPF
0007 COMMON /VFIAG/ FI0MSK, FI1MSK, FI2MSK, FI3MSK, FI4MSK, FI5MSK, FI67,
1 FIAG8, FIAG9, FIAG10, FIAG11, FIAG12, FIAG13, FIAG14,
2 FIAG15
0008 COMMON /VCRNT/ REGASP, CTRIPT, CTRI1, LEASTV, NTFREE, FREE, DREGSP,
1 MSA, PAGLOC, CURENT
0009 COMMON /VBUFR/ PAGSIZE, NWCHAN, OICHAN, CMPAND, PAGHDR, BUFSIZE, DIRSZ,
1 INCORE, HDRSZ, MSADIR, SUFSZ, BIKSZ, DIRBIK, PAGBD4
0010 COMMON /VHVDI/ THMSA, REGAS, PAGENO, CONTNO, INSPCL,
1 USECT, HDRFIG, READVI, OI DNDH, DNOPEH, NROWH, DROWH
0011 COMMON /VSWIT/ SETUP, SNGIBK, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
1 DL1STR, DL2STR, IN2TMP, FD2TMP, DL2TMP, DUMPFI,
2 FINDFI, DLETFI, NSRTFI
0012 COMMON /VPRAM/ BUFILOC, LENGTH, IERR, ERRNUM, BINARY, BCD, MODE, PAGES,
1 LUN
0013 COMMON /VADDR/ IADD, THIS, LSTHED, LOC, LAST, LASTLC
0014 COMMON /VEND/ COUNT, ABSPOS, LSTCON
0015 COMMON /VINSI/ REORG, FULL, RPLCE
0016 COMMON /VVTRI/ NOPSPC(1)
1 /VVTR2/ LSTSPC(1)
2 /VVTR3/ LNKSPC(1)
3 /VVTR4/ FIGSPC(1)
0017 DATA TWO/'2', THREE/'3', NF1G67/'177774', SV1RPL/'0'
C
C CALLS TO LVFINI OR LVFNV MUST PRECEDE A CALL TO THIS ROUTINE.
C
C INSERTION TYPE ?
C
D PAUSE 'IN LVNSRT'
0018 RPLCE = .FALSE.
0019 IF(INDXON-1) 125, 126, 127
C
C IS THE GIRS BUFFER FULL ?
0020 125 IF(REGASP .EQ. LSTSPC(CTRI1 + REGASP)) GO TO 98
C
C FORM FIRST WORD OF SINGLE OR MULTIVALUED FUNCTION
0022 FIGTMP = FI1MSK .OR. ITYP1(1)
0023 IF(NVAL .EQ. 1) GO TO 20
0025 LSTTMP = REGASP
0026 FIGTMP = FIGTMP .OR. FI0MSK .OR. FI2MSK
0027 GO TO 21
0028 20 LSTTMP = IVALS(1)
C

```

```

C IF THIS FUNCTION ALREADY EXISTS, GO TO 43
0029 21 IF (ITESTR .GT. 0) GO TO 43
C
C IF THAT ADDRESS IS ALREADY IN WORKING SPACE, GO TO 25
0031 IF ((FI1MSK .AND. FIGSPC(CTRI1 + IADD)) .NE. 0) GO TO 25
C
C UPDATE REGASP (IF NECESSARY)
0033 IF (IADD .EQ. REGASP) REGASP = LSTSPC(CTRI1 + IADD)
C
C UPDATE AVAILABLE SPACE
0035 LSTSPC(CTRI1 + NOPSPC(CTRI1 + IADD)) = LSTSPC(CTRI1 + IADD)
0036 NOPSPC(CTRI1 + LSTSPC(CTRI1 + IADD)) = NOPSPC(CTRI1 + IADD)
C
C INSERT FUNCTION
0037 NOPSPC(CTRI1 + IADD) = IFUNC
0038 LSTSPC(CTRI1 + IADD) = LSTTMP
0039 LNKSPC(CTRI1 + IADD) = IADD
0040 FIGSPC(CTRI1 + IADD) = FIGSPC(CTRI1 + IADD) .OR. FI4TMP .OR. FI5MSK
C
C INSERT ANY ADDITIONAL FUNCTION VALUES
0041 IF (NVAL .EQ. 1) GO TO 100
0043 HEAD = IADD
0044 OLDLOC = IADD
0045 GO TO 50
C
C IF THAT ADDRESS CONTAINS THE HEAD OF A CONFLICT LIST, GO TO 60
0046 25 IF ((FI5MSK .AND. FIGSPC(CTRI1 + IADD)) .GT. 0) GO TO 60
C
C IF THAT ADDRESS CONTAINS A VALUE ON A MULTIVALUE LIST, GO TO 35
0048 IF ((FI2MSK .AND. FIGSPC(CTRI1 + IADD)) .GT. 0) .AND.
1 (FI0MSK .AND. FIGSPC(CTRI1 + IADD)) .EQ. 0) GO TO 35
C
-----
C- THE ADDRESS CONTAINS A FUNCTION ON A CONFLICT LIST, BUT NOT THE HEAD OF LIST
0050 THIS = IADD
C
C FIND THE PRECEDING FUNCTION ON THE CONFLICT LIST
0051 26 IF (LNKSPC(CTRI1 + LNKSPC(CTRI1 + THIS)) .EQ. IADD) GO TO 27
0053 THIS = LNKSPC(CTRI1 + THIS)
0054 GO TO 26
0055 27 LAST = LNKSPC(CTRI1 + THIS)
0056 NEWLOC = REGASP
0057 IF (REGASP .EQ. LSTSPC(CTRI1 + REGASP)) GO TO 98
C
C UPDATE AVAILABLE SPACE AND REGASP
0059 CALL LVUPDT
C MOVE THE FUNCTION ON A CONFLICT LIST TO THE FIRST CELL OF AVAILABLE
C SPACE
0060 NOPSPC(CTRI1 + NEWLOC) = NOPSPC(CTRI1 + IADD)
0061 LSTSPC(CTRI1 + NEWLOC) = LSTSPC(CTRI1 + IADD)
0062 LNKSPC(CTRI1 + NEWLOC) = LNKSPC(CTRI1 + IADD)
0063 FIGSPC(CTRI1 + NEWLOC) = FIGSPC(CTRI1 + IADD) .OR. FI4MSK
0064 FIGSPC(CTRI1 + IADD) = 0
0065 LNKSPC(CTRI1 + LAST) = NEWLOC
C
C INSERT THIS FUNCTION AS THE HEAD OF A CONFLICT LIST
0066 NOPSPC(CTRI1 + IADD) = IFUNC

```

```

0067     LNKSPC(CTRI1 + IADD) = IADD
0068     LSTSPC(CTRI1 + IADD) = LSTTMP
0069     FIGSPC(CTRI1 + IADD) =
0070     1 FIGSPC(CTRI1 + IADD) .OR. FIGTMP .OR. FI4MSK .OR. FI5MSK
      IF((FIGSPC(CTRI1 + NEWIOC) .AND. FI0MSK) .EQ. 0) GO TO 34
C
C IF THE FUNCTION THAT WAS MOVED IS THE HEAD OF A MULTIVALUE LIST, FIX POINTERS
0072     NEXT = LSTSPC(CTRI1 + NEWIOC)
0073 30    NEXT = LSTSPC(CTRI1 + NEXT)
0074     IF(LSTSPC(CTRI1 + NEXT) .NE. IADD) GO TO 30
0076     LSTSPC(CTRI1 + NEXT) = NEWIOC
C
C INSERT ANY ADDITIONAL FUNCTION VALUES
0077 34    HEAD = IADD
0078     OLDLOC = IADD
0079     IF(NVAL .GT. 1) GO TO 50
0081     GO TO 100
C
C-----
C-THE ADDRESS CONTAINS A VALUE ON A MULTIVALUE LIST
0082 35    NEWIOC = REGASP
0083     IF(REGASP .EQ. LSTSPC(CTRI1 + REGASP)) GO TO 98
C
C UPDATE AVAILABLE SPACE AND REGASP
0085     CALL LVUPDT
C
C MOVE THE VALUE ON A MULTIVALUE LIST TO THE FIRST CELL OF
C AVAILABLE SPACE
0086     NODSPC(CTRI1 + NEWIOC) = NODSPC(CTRI1 + IADD)
0087     LSTSPC(CTRI1 + NEWIOC) = LSTSPC(CTRI1 + IADD)
0088     LNKSPC(CTRI1 + NEWIOC) = LNKSPC(CTRI1 + IADD)
0089     FIGSPC(CTRI1 + NEWIOC) = FIGSPC(CTRI1 + IADD)
0090     FIGSPC(CTRI1 + IADD) = 0
C
C RESET POINTERS
C
0091     L1 = LSTSPC(CTRI1 + NEWIOC)
0092     IF((FI0MSK .AND. FIGSPC(CTRI1 + L1)) .EQ. 0) GO TO 200
0094     LNKSPC(CTRI1 + LSTSPC(CTRI1 + L1)) = NEWIOC
0095     GO TO 201
0096 200   LNKSPC(CTRI1 + L1) = NEWIOC
0097 201   KZVAL = LSTSPC(CTRI1 + LNKSPC(CTRI1 + NEWIOC))
0098     IF((FIGSPC(CTRI1 + KZVAL) .AND. FI0MSK) .NE. 0) GO TO 38
0100     LSTSPC(CTRI1 + LNKSPC(CTRI1 + NEWIOC)) = NEWIOC
0101     GO TO 39
0102 38    LSTSPC(CTRI1 + KZVAL) = NEWIOC
0103 39    NODSPC(CTRI1 + IADD) = IFUNC
C INSERT THIS FUNCTION AS THE HEAD OF A CONFLICT LIST
0104     LNKSPC(CTRI1 + IADD) = IADD
0105     LSTSPC(CTRI1 + IADD) = LSTTMP
0106     FIGSPC(CTRI1 + IADD) =
0107     1 FIGSPC(CTRI1 + IADD) .OR. FIGTMP .OR. FI4MSK .OR. FI5MSK
      GO TO 100
C
C-----
C-THE FUNCTION TO BE INSERTED IS ON THE CONFLICT LIST
0108 43    HEAD = THIS

```

```

C
C IS THIS A SINGLE VALUE LIST OR MULTIVALUE LIST?
0109 IF(LSTHED .IT. 0) GO TO 51
C
C OLDLOC IS THE LOCATION OF THE LAST VALUE ON THE MULTIVALUE LIST
C
0111 OLDLOC = LNKSPC(CTRI1 + LSTHED)
C
C-----
C-INSERT ADDITIONAL FUNCTION VALUES
0112 50 LSTASP = NODSPC(CTRI1 + REGASP)
0113     IN = 0
0114     GO TO 56
C
C-----
C-FORM MULTIVALUE LIST TO ADD VALUE(S) TO SINGLE-VALUED FUNCTION
0115 51 IN = 0
0116     IF(REGASP .EQ. LSTSPC(CTRI1 + REGASP)) GO TO 98
0118     LSTASP = NODSPC(CTRI1 + REGASP)
0119     NEWLOC = REGASP
0120     REGASP = LSTSPC(CTRI1 + REGASP)
0121     NODSPC(CTRI1 + NEWLOC) = LSTSPC(CTRI1 + THIS)
0122     TEMP = (FIGSPC(CTRI1 + THIS) .AND. FIG67)
0123     FIGSPC(CTRI1 + NEWLOC) = (TEMP .OR. FIGSPC(CTRI1 + NEWLOC))
0124     FIGSPC(CTRI1 + THIS) = (FIGSPC(CTRI1 + THIS) .AND. NFIG67)
0125     FIGSPC(CTRI1 + THIS) = (FIG2MSK .OR. FIGSPC(CTRI1 + THIS))
0126     FIGSPC(CTRI1 + THIS) = (FIG0MSK .OR. FIGSPC(CTRI1 + THIS))
0127     LNKSPC(CTRIPT+INSPCL) = LNKSPC(CTRIPT+INSPCL) + 1
0128     OLDLOC = THIS
C
C-----
C INSERT ANOTHER VALUE ON MULTIVALUE LIST
0129 52 FIGSPC(CTRI1 + NEWLOC) = (FIG2MSK .OR. FIGSPC(CTRI1 + NEWLOC))
0130     FIGSPC(CTRI1 + NEWLOC) = (FIG1MSK .OR. FIGSPC(CTRI1 + NEWLOC))
0131     LSTSPC(CTRI1 + OLDLOC) = NEWLOC
0132     LNKSPC(CTRI1 + NEWLOC) = OLDLOC
0133     OLDLOC = NEWLOC
0134     56 NEWLOC = REGASP
0135     IF(IN .GT. 0) GO TO 57
C
C NO VALUES HAVE BEEN INSERTED YET
0137     IN = 1
0138     GO TO 58
C
C SOME VALUES HAVE BEEN INSERTED
0139 57 IF(IN .EQ. NVAL) GO TO 67
0141     IN = IN+1
C
0142 58 IF(REGASP .EQ. LSTSPC(CTRI1 + REGASP)) GO TO 98
0144     REGASP = LSTSPC(CTRI1 + REGASP)
0145     NODSPC(CTRI1 + NEWLOC) = IVALS(IN)
0146     FIGSPC(CTRI1 + NEWLOC) = (ITYP1(IN) .OR. FIGSPC(CTRI1 + NEWLOC))
0147     ITYP1(IN) = 0
0148     GO TO 52
C
C END MULTIVALUE LIST AND UPDATE AVAILABLE SPACE
0149 67 LSTSPC(CTRI1 + OLDLOC) = HEAD

```

```

0150      NOPSPC(CTRI1 + REGASP) = LSTASP
0151      LSTSPC(CTRI1 + LSTASP) = REGASP
0152      LNKSPC(CTRI1 + LSTSPC(CTRI1 + HEAD)) = OLDLOC
0153      GO TO 100
C
C-----
C-THE FUNCTION TO BE INSERTED IS NOT ON THE CONFLICT LIST
0154 60      ASPREG = REGASP
0155      LSTASP = NOPSPC(CTRI1 + REGASP)
0156      IF(REGASP .EQ. LSTSPC(CTRI1 + REGASP)) GO TO 98
C
C UPDATE AVAILABLE SPACE AND REGASP
0158      CALL LVUPDT
C
C INSERT FUNCTION IN FIRST CELL OF AVAILABLE SPACE
0159      NOPSPC(CTRI1 + ASPREG) = IFUNC
0160      IF(NVAL .EQ. 1) GO TO 611
0162      LSTSPC(CTRI1 + ASPREG) = REGASP
0163      FIGSPC(CTRI1 + ASPREG) = (FI2MSK .OR. FIGSPC(CTRI1 + ASPREG))
0164      FIGSPC(CTRI1 + ASPREG) = (FI0MSK .OR. FIGSPC(CTRI1 + ASPREG))
0165      GO TO 612
0166 611      LSTSPC(CTRI1 + ASPREG) = IVALS(1)
0167 612      FIGSPC(CTRI1 + ASPREG) =
1 FIGSPC(CTRI1 + ASPREG) .OR. ITYP1(1) .OR. FI1MSK .OR. FI4MSK
0168      LNKSPC(CTRI1 + ASPREG) = IADD
0169      LNKSPC(CTRI1 + LAST) = ASPREG
0170      IF(NVAL .EQ. 1) GO TO 100
C
C INSERT ADDITIONAL VALUES
0172      LSTASP = NOPSPC(CTRI1 + REGASP)
0173      OLDLOC = ASPREG
0174      HEAD = ASPREG
0175      IN = 0
0176      GO TO 56
C
C DESTRUCTIVE INSERTION
C
C A CALL TO LVFIND MUST PRECEDE A CALL TO EITHER 126 OR 127.
C GIVEN N VALUES OF TYPE K ON A LIST WHERE N.GE.0, INDEXED
C INSERTIONS SHALL SUCCEED FOR IPOS.GE.1 .AND. IPOS .LE. N+1
C
C DEFEAT SAVED INDEX UNTIL NEXT RETRIEVAL.
0177 126      FIGSPC(CTRI1 + THIS) = FIGSPC(CTRI1 + THIS) .OR. FI4MSK
0178      ABSPOS = IABS(IPOS)
0179      KPOS = IPOS
0180      INDEX = 0
C DOES THE IPOS'TH VALUE OF THE PROPER TYPE EXIST?
0181      IF(ITESTR .IT. 0) GO TO 90
C REPLACE VALUE AT LOCATION 'LOC'. SVI OR MVI?
0183      RPLACF = .TRUE.
0184      IF(LSTHED .GT. 0) GO TO 356
C SVI
0186      LSTSPC(CTRI1 + LOC) = IVALS(1)
0187      SVIRPL = 1
0188      GO TO 365
C MVI
0189 356      NOPSPC(CTRI1 + LOC) = IVALS(1)

```

```

C REPLACE TYPE.
0190 365 FIGSPC(CTRI1 + LOC) =
      1 ((FIGSPC(CTRI1 + LOC) .AND. NFIG67) .OR. ITYP1(1))
0191 GO TO 100
C
C IPOS'TH VALUE WAS NOT FOUND, INDEXED INSERTION CAN STILL SUCCEED
C IF (IPOS-1) VALUE IS FOUND. THIS THEN BECOMES A NORMAL INSERTION
C IF ABSPOS = 1 OR THE VALUE WILL BE THE LAST IN THE LIST. OTHERWISE,
C THIS BECOMES A NONDESTRUCTIVE INSERTION TO THE FIRST POSITION IN
C THE LIST
0192 90 IF(ABSPOS .EQ. 1) GO TO 125
0194 IF(KPOS) 91,97,92
0195 91 KPOS = KPOS+1
0196 GO TO 93
0197 92 KPOS = KPOS-1
0198 93 CALL LVFIND
0199 IPOS = KPOS
0200 CALL LVENV(INDEX,INDEX,INDEX,INDEX)
C FAILURE IF NO VALUE IS FOUND.
0201 IF(ITESTR .IT. 0) GO TO 97
C NORMAL INSERTION IF REQUEST WAS IPOS'TH FROM THE TOP.
0203 IF(KPOS .GT. 0) GO TO 125
C NONDESTRUCTIVE INSERTION AT THE BEGINNING OF THE LIST.
0205 NEWLOC = REGASP
0206 IF(REGASP .EQ. LSTSPC(CTRI1 + REGASP)) GO TO 98
0208 CALL LVUPDT
C SVI OR MVI?
0209 IF(LSTHED .GT. 0) GO TO 377
0211 GO TO 344
C
C NONDESTRUCTIVE INSERTION
C
C IF IPOS = -1, PLACE AT THE END OF THE LIST (NORMAL INSERTION).
0212 127 IF(IPOS .EQ. -1) GO TO 125
C
C DEFFAT SAVED INDEX UNTIL NEXT RETRIEVAL.
0214 FIGSPC(CTRI1 + THIS) = FIGSPC(CTRI1 + THIS) .OR. FI4MSK
0215 ABSPOS = IABS(IPOS)
0216 KPOS = IPOS
0217 INDEX = 0
0218 NEWLOC = REGASP
C DOES THE IPOS'TH VALUE OF THE PROPER TYPE EXIST?
0219 IF(ITESTR .IT. 0) GO TO 90
0221 IF(REGASP .EQ. LSTSPC(CTRI1 + REGASP)) GO TO 98
0223 CALL LVUPDT
C SVI OR MVI?
0224 IF(LSTHED .IT. 0) GO TO 344
C MVI
0226 IF(KPOS .IT. 0) GO TO 347
C
C PLACE VALUE AT THE IPOS'TH POSITION (WRT ITYP) FROM THE TOP OF LIST
0228 377 ISTLOC = INKSPC(CTRI1 + LOC)
0229 NODSPC(CTRI1 + NEWLOC) = IVALS(1)
0230 LSTSPC(CTRI1 + NEWLOC) = LOC
0231 INKSPC(CTRI1 + NEWLOC) = ISTLOC
0232 FIGSPC(CTRI1 + NEWLOC) = FI1MSK .OR. FI2MSK .OR. ITYP1(1)

```

```

0233      IF(LOC .NE. LSTHED) GO TO 321
0235      LSTSPC(CTRI1 + LSTSPC(CTRI1 + ISTLOC)) = NEWIOC
0236      GO TO 322
0237 321   LSTSPC(CTRI1 + ISTLOC) = NEWIOC
0238 322   LNKSPC(CTRI1 + LOC) = NEWIOC
0239      GO TO 100

C
C PLACE VALUE AT THE IPOS'TH POSITION (WRT ITYP) FROM THE BOTTOM OF
C THE LIST
0240 347   NOPSPC(CTRI1 + NEWIOC) = IVALS(1)
0241      LSTSPC(CTRI1 + NEWIOC) = LSTSPC(CTRI1 + LOC)
0242      LNKSPC(CTRI1 + NEWIOC) = LOC
0243      FIGSPC(CTRI1 + NEWIOC) = FI1MSK .OR. FI2MSK .OR. ITYP1(1)
0244      IF((FIGSPC(CTRI1 + LSTSPC(CTRI1 + LOC)) .AND. FI0MSK) .EQ. 0)
1         GO TO 323
0246      KZVAL = LSTSPC(CTRI1 + LOC)
0247      LNKSPC(CTRI1 + LSTSPC(CTRI1 + KZVAL)) = NEWIOC
0248      GO TO 324
0249 323   LNKSPC(CTRI1 + LSTSPC(CTRI1 + LOC)) = NEWIOC
0250 324   LSTSPC(CTRI1 + LOC) = NEWIOC
0251      GO TO 100

C
C CREATE MVI WITH NEW VALUE AT THE TOP OF THE LIST.
0252 344   IF(REGASP .EQ. LSTSPC(CTRI1 + REGASP)) GO TO 98
0254      NWIOC2 = REGASP
0255      CALL LVUPDT
0256      NOPSPC(CTRI1 + NEWIOC) = IVALS(1)
0257      LSTSPC(CTRI1 + NEWIOC) = NWIOC2
0258      LNKSPC(CTRI1 + NEWIOC) = NWIOC2
0259      FIGSPC(CTRI1 + NEWIOC) = FI1MSK .OR. FI2MSK .OR. ITYP1(1)
0260      NOPSPC(CTRI1 + NWIOC2) = LSTSPC(CTRI1 + THIS)
0261      LSTSPC(CTRI1 + NWIOC2) = THIS
0262      LNKSPC(CTRI1 + NWIOC2) = NEWIOC
0263      KIGTEP = FIGSPC(CTRI1 + THIS) .AND. FIG67
0264      FIGSPC(CTRI1 + NWIOC2) = (FI1MSK .OR. FI2MSK) .OR. KIGTEP
0265      LSTSPC(CTRI1 + THIS) = NEWIOC
0266      FIGSPC(CTRI1 + THIS) =
1         (FIGSPC(CTRI1 + THIS) .OR. FI0MSK) .OR. FI2MSK
C FLAG 4 IS SET BECAUSE THIS INSERTION MIGHT BE A RECREATION OF AN
C OLD LIST
0267 100   FIGSPC(CTRI1 + THIS) = FIGSPC(CTRI1 + THIS) .OR. FI4MSK
0268      IVAL = IVALS(1)
C "FAILURE" IF IFUNC+IARG DID NOT PREVIOUSLY EXIST
C IF((FIGSPC(CTRI1 + THIS) .AND. FI0MSK) .NE. 0) .OR.
C 1 SVIRPL .EQ. 1) ITESTR = 1
0269 97    IPOS = 1
0270      SVIRPL = 0
0271      ITYP = 3
0272      INDXON = 0
0273      NVAL = 1
0274      ITYP1(1) = 0
0275      RETURN

C
C CONTINUANT IS FULL, TRY AGAIN
0276 98    FULL = .TRUE.
0277      RETURN
0278      END

```

```

C
C
C
0001 SUBROUTINE LVUPDT
0002 IMPLICIT INTEGER(A-Z)
0003 LOGICAL*1 CURENT
0004 COMMON /VCRNT/ REGASP,CTRI PT,CTRI 1,LEASTV,NTFREE,FREE,DREGSP,
1 MSA,PAGLOC,CURENT
0005 COMMON /VVTR1/ NODSPC(1)
1 /VVTR2/ LSTSPC(1)
2 /VVTR3/ LNKSPC(1)
3 /VVTR4/ FIGSPC(1)
0006 COMMON /VFIAG/ FI0MSK,FI1MSK,FI2MSK,FI3MSK,FI4MSK,FI5MSK,FI67,
1 FIAG8,FIAG9,FIAG10,FIAG11,FIAG12,FIAG13,FIAG14,
2 FIAG15
C
C THIS ROUTINE UPDATES AVAILABLE SPACE AND THE REGISTER OF AVAILABLE
C SPACE - REGASP
C
D PAUSE 'IN LVUPDT'
0007 LSTSPC(CTRI 1 + NODSPC(CTRI 1 + REGASP)) = LSTSPC(CTRI 1 + REGASP)
0008 NODSPC(CTRI 1 + LSTSPC(CTRI 1 + REGASP)) = NODSPC(CTRI 1 + REGASP)
0009 REGASP = LSTSPC(CTRI 1 + REGASP)
0010 XXX=1000
0011 IF((FIGSPC(CTRI 1+REGASP).OR.FI3MSK).NE.FI3MSK)XXX=XXX*XXX
0012 RETURN
0013 END
0014

```

```

C
C
0001 SUBROUTINE LVDLEX
0002 IMPLICIT INTEGER(A-Z)
0003 LOGICAL*1 SNGIBK, SETUP, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR, DL1STR,
1 DL2STR, DUMPF1, CURENT, FINDFI, DLETF1, NSRTFI, FD1TMP,
2 DL2TMP, IN2TMP, FD2TMP, INSIDE, REORG, FULL, LSTCON, RPLACE,
3 BAKCON
0004 COMMON /IVARGS/ IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, ITESTR,
1 INCIUD, INDXON, IVALS(10), ITYP1(10), SRCSUF,
2 LNKSUF, SNKSUF, INSTYP
0005 COMMON /IVREGS/ CURPAG(4), REQPA(4), LSTVPG(4), MSARET,
1 HREQPG, NXTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0006 COMMON /IVMASK/ MWRITE, NOTUSD, NEWCON, FIGMSK, MASKSF, MASKPF
0007 COMMON /IVFLAG/ FI0MSK, FI1MSK, FI2MSK, FI3MSK, FI4MSK, FI5MSK, FI67,
1 FIAG8, FIAG9, FIAG10, FIAG11, FIAG12, FIAG13, FIAG14,
2 FIAG15
0008 COMMON /IVCRNT/ REGASP, CTRIPT, CTRI1, LEASTV, NTFREE, FREE, DREGSP,
1 MSA, PAGLOC, CURENT
0009 COMMON /IVBUFR/ PAGSZE, NWCHAN, OICHAN, CMPAND, PAGHDR, BUFSZE, DIRSZE,
1 INCORE, HDRSZE, MSADIR, SUFSZE, BIKSZE, DIRBIK, PAGHD4
0010 COMMON /IVBDVI/ THSMS, REGAS, PAGENO, CONTNO, INSDLE,
1 USECT, HDRFIG, READVI, OI DNDH, DNODEH, NROWH, DROWH
0011 COMMON /IVSWIT/ SETUP, SNGIBK, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
1 DL1STR, DL2STR, IN2TMP, FD2TMP, DL2TMP, DUMPF1,
2 FINDFI, DLETF1, NSRTFI
0012 COMMON /IVPRAM/ BUFILOC, LENGTH, IERR, ERRNUM, BINARY, BCD, MODE, PAGES,
1 LUN
0013 COMMON /IVFND/ COUNT, ABSPOS, LSTCON
0014 COMMON /IVINS1/ REORG, FULL, RPLACE
0015 COMMON /IVDEL1/ NUMRET, BAKCON
0016 COMMON /IVVTR1/ NOISPC(1)
1 /IVVTR2/ LSTSPC(1)
2 /IVVTR3/ LNKSPC(1)
3 /IVVTR4/ FIGSPC(1)
C
0017 DATA INSIDE /.FALSE./
C
C THE DELETE EXECUTIVE ROUTINE OBTAINS THE CORRECT P,C FOR SUBROUTINE
C LVDLET TO OPERATE ON.
C
C D PAUSE 'IN LVDLEX'
C IS LVDLET BEING CALLED FROM A DELETE STRATEGY ROUTINE ?
0018 IF(INSIDE .EQ. .TRUE.) GO TO 100
C
C TO PREVENT RECURSION, SAVE THE FIND STRATEGY FLAGS AND TURN THEM OFF
0020 FD1TMP = FD1STR
0021 FD2TMP = FD2STR
0022 FD1STR = .FALSE.
0023 FD2STR = .FALSE.
C
C CALL USER'S FIRST DELETE STRATEGY ROUTINE ?
0024 IF(DL1STR .EQ. .FALSE.) GO TO 100
C
C TO PREVENT RECURSION, INHIBIT CALLS TO ALL USER STRATEGY ROUTINES

```

```

0026      DL1TMP = DL1STR
0027      DL2TMP = DL2STR
0028      DL1STR = .FALSE.
0029      DL2STR = .FALSE.
0030      IN1TMP = IN1STR
0031      IN2TMP = IN2STR
0032      IN1STR = .FALSE.
0033      IN2STR = .FALSE.
      C
      C SET UP FOR FIRST USER ROUTINE
0034      CALL LVSTAC
0035      INSIDE = .TRUE.
0036      CALL USRDL1
0037      INSIDE = .FALSE.
0038      CALL LVPOP
0039      DL1STR = DL1TMP
0040      DL2STR = DL2TMP
0041      IN1STR = IN1TMP
0042      IN2STR = IN2TMP
      C
      C PROCEED WITH DELETION ?
0043      IF(DLETFI .EQ. .FALSE.) GO TO 600
      C
      C BRING IN PROPER CONTINUANT
0045      100 J = 0
0046      CALL LVFDEX(J,J,J,J,J)
      C
      C NO LIST TO BE DELETED ?
0047      IF(ITESTR .IT. 0) GO TO 600
      C
      C ASSUME LIST DOES NOT PROCEED TO ANOTHER CONTINUANT
0049      200 LSTCON = .FALSE.
0050      BAKCON = .FALSE.
      C
      C NUMRET COUNTS THE NUMBER OF LOCATIONS RETURNED TO AVAILABLE SPACE
0051      NUMRET = 0
0052      CALL LVDLET
      C
      C UPDATE CONTINUANT FILL QUANTITY
0053      LNKSPC(CTRIPT+INSDLE) = LNKSPC(CTRIPT+INSDLE) - NUMRET
      C
      C CONTINUANT HAS BEEN MODIFIED
0054      FIGSPC(CTRIPT+HDRFIG) = FIGSPC(CTRIPT+HDRFIG) .OR. MWRITE
      C
      C INDEXED DELETE ?
0055      IF(INPXON .EQ. 1) GO TO 400
      C
      C FINISHED ?
0057      IF(LSTCON .EQ. .FALSE.) GO TO 600
      C
      C EXAMINE NEXT CONTINUANT
0059      300 REQPA(2) = CURPA(2) + 1
0060      J = 0
0061      CALL LVFDEX(J,J,J,J,J)
      C
      C NO MORE CONTINUANTS ?
0062      IF(MSARET .IE. 0) GO TO 500

```

AD-A099 125

DAVID W TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CE--ETC F/6 5/2
PAGED GIRS (GRAPH INFORMATION RETRIEVAL SYSTEM) USERS MANUAL.(U)
MAY 81 I S ZARITSKY

UNCLASSIFIED

DTNSRDC-81/024

NL

2 of 2
AD-A099 125

END
DATE
FILMED
6-81
DTIC

```

C
C DOES A PORTION OF THE LIST RESIDE ON CURRENT ?
0064 IF(ITESTR .EQ. 0) GO TO 300
0065 GO TO 200
C
C RESET DEFAULT TO "DELETE ENTIRE LIST"
0067 400 INPXON = 0
0068 IF(BAKCON .EQ. .FALSE.) GO TO 450
C
C LIST NO LONGER POINTS FORWARD TO A FOLLOWING CONTINUANT, REMOVE FLAG
0070 IPOS = -1
0071 CALL LVFDEX
0072 FIGSPC(CTRL1+LOC) = FIGSPC(CTRL1+LOC) .AND. .NOT. FLAG11
0073 450 IF(LSTCON .EQ. .FALSE.) GO TO 600
C
C LIST NO LONGER POINTS BACKWARD TO A PREVIOUS CONTINUANT, REMOVE FLAG
0075 IPOS = 1
0076 CALL LVFDEX
0077 FIGSPC(CTRL1+THIS) = FIGSPC(CTRL1+THIS) .AND. .NOT. FLAG10
0078 GO TO 600
C
C ERROR, LIST CONTINUATION FLAG BUT NO MORE CONTINUANTS !
0079 500 ERRNUM = 50
0080 MODE = BCD
0081 PAGES = -1
0082 DUMP = 0
0083 CALL LVDUMP(DUMP)
0084 X = 1000
0085 X = X * X
0086 STOP
C
C CALL SECOND USER DELETION STRATEGY ROUTINE ?
0087 600 IF(DL2STR .EQ. .FALSE.) GO TO 700
0088 DL1TMP = DL1STR
0089 DL2TMP = DL2STR
0090 DL1STR = .FALSE.
0091 DL2STR = .FALSE.
0092 IN1TMP = IN1STR
0093 IN2TMP = IN2STR
0094 IN1STR = .FALSE.
0095 IN2STR = .FALSE.
0096 CALL LVSTAC
0097 INSIDE = .TRUE.
0098 CALL USRDL2
0099 INSIDE = .FALSE.
0100 CALL LVPOP
0101 DL1STR = DL1TMP
0102 DL2STR = DL2TMP
0103 IN1STR = IN1TMP
0104 IN2STR = IN2TMP
C
C RESTORE FIND STRATEGY FLAGS
0106 700 IF(INSIDE .EQ. .TRUE.) RETURN
0108 FD1STR = FD1TMP
0109 FD2STR = FD2TMP
0110 RETURN
0111 END

```

```

C
C
0001 SUBROUTINE LVPLET
0002 IMPLICIT INTEGER(A-Z)
0003 LOGICAL*1 SNGIBK, SETUP, NYTRAN, IN1STR, IN2STR, FD1STR, FD2STR, DL1STR,
1 DL2STR, DUMPF1, CURENT, FINDFI, DLETFI, NSRTFI, DL1TMP,
2 DL2TMP, IN2TMP, FD2TMP, REORG, FULL, LSTCON, RPLACE, BAKCON
0004 COMMON /I VARG/ IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, ITESTR,
1 INCIUD, INDXON, IVALS(10), ITYP1(10), SRCSUF,
2 LNKSUF, SNKSUF, INSTYP
0005 COMMON /I VREG/ CURPAG(4), REQPG(4), LSTPG(4), MSARPT,
1 HREQPG, NXTMSA, HACTPG(2), RMACT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0006 COMMON /I VMASK/ MWRITE, NOTUSD, NEWCON, FIGMSK, MASKSF, MASKPF
0007 COMMON /I VFIAG/ FI0MSK, FI1MSK, FI2MSK, FI3MSK, FI4MSK, FI5MSK, FI67,
1 FIAG9, FIAG9, FIAG10, FIAG11, FIAG12, FIAG13, FIAG14,
2 FIAG15
0008 COMMON /I VCRNT/ REGASP, CTRI PT, CTRI 1, LEASTV, NTFREE, FREE, DREGSP,
1 MSA, PAGLOC, CURENT
0009 COMMON /I VBUFR/ PAGSIZE, NWCHAN, OI CHAN, CMPAND, PAGHDR, BUFSZE, DIRSZE,
1 INCORE, HDRSZE, MSADIR, SUFSZE, BLKSZE, DIRBIK, PAGHD4
0010 COMMON /I VHDVI/ THSMSA, REGAS, PAGENO, CONTNO, INSDEL,
1 USECT, HDRFI G, READVI, OI DNPH, DNOPER, NROWH, DROWH
0011 COMMON /I VSWIT/ SETUP, SNGIBK, NYTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
1 DL1STR, DL2STR, IN2TMP, FD2TMP, DL2TMP, DUMPF1,
2 FINDFI, DLETFI, NSRTFI
0012 COMMON /I VPRAM/ BUFI OC, LENGTH, IERR, ERRNUM, BINARY, BCD, MODE, PAGES,
1 LUN
0013 COMMON /I VADDR/ IADD, THIS, LSTHED, LOC, LAST, LASTLC
0014 COMMON /I VSTAK/ CURI EV, NUMVAR, STACK(1)
0015 COMMON /I VFND/ COUNT, ABSPOS, LSTCON
0016 COMMON /I VINS1/ REORG, FULL, RPLACE
0017 COMMON /I VDEL1/ NUMRET, BAKCON
0018 COMMON /I VVTR1/ NODSPC(1)
1 /I VVTR2/ LSTSPC(1)
2 /I VVTR3/ LNKSPC(1)
3 /I VVTR4/ FIGSPC(1)
0019 DATA NF1023/'177517/
C
C
C DOES THE LIST EXIST ?
D PAUSE 'IN LVPLET'
0020 IF(ITESTR .IT. 0) RETURN
C
C SVI. OR MVI ?
0022 IF(LSTHED .IT. 0) GO TO 200
C
C INDEXED DELETE ?
0024 IF(INDXON .EQ. 1) GO TO 500
C
C DELETE ENTIRE MULTIVALUE LIST
0026 ISADD = LSTHED
0027 LOC = THIS
0028 100 NXTADD = LSTSPC(CTRI 1 + ISADD)
0029 IF((FIGSPC(CTRI 1 + NXTADD) .AND. FIAG11) .NE. 0) LSTCON = .TRUE.

```

```

0031     NODSPC(CTRI1 + ISADD) = NODSPC(CTRI1 + REGASP)
0032     LSTSPC(CTRI1 + ISADD) = REGASP
0033     LNKSPC(CTRI1 + ISADD) = 0
0034     FIGSPC(CTRI1 + ISADD) = 0
0035     LSTSPC(CTRI1 + NODSPC(CTRI1 + REGASP)) = ISADD
0036     NODSPC(CTRI1 + REGASP) = ISADD
0037     NUMRET = NUMRET + 1
0038     IF((FIGSPC(CTRI1 + NXTADD) .AND. FI0MSK) .NE. 0) GO TO 200
0040     ISADD = NXTADD
0041     GO TO 100
C
C DELETE SINGLE VALUED FUNCTION
C FORWARD OR BACK MVI CONTINUANT POINTER FLAGS MAY HAVE TO BE REMOVED
0042 200 IF((FIGSPC(CTRI1 + THIS) .AND. FLAG11) .NE. 0) LSTCON = .TRUE.
0044 IF((FIGSPC(CTRI1 + THIS) .AND. FLAG10) .NE. 0) BAKCON = .TRUE.
C IF THE LIST EXTENDED TO BOTH A PREVIOUS AND FOLLOWING CONTINUANT,
C DO NOT REMOVE POINTER FLAGS
0046 IF(LSTCON .EQ. .FALSE. .OR. BAKCON .EQ. .FALSE.) GO TO 220
0048 LSTCON = .FALSE.
0049 BAKCON = .FALSE.
C IS THE FUNCTION HEAD OF A CONFLICT LIST
0050 220 IF(THIS .NE. IADD) GO TO 400
0052 NXFUNC = LNKSPC(CTRI1 + IADD)
C IF THIS FUNCTION IS THE ONLY ONE ON THE CONFLICT LIST, GO TO 300.
C OTHERWISE, PLACE NEXT FUNCTION ON CONFLICT LIST IN 'HEAD OF
C CONFLICT LIST' LOCATION (IADD)
0053 IF(NXFUNC .EQ. IADD) GO TO 300
0055 NODSPC(CTRI1 + IADD) = NODSPC(CTRI1 + NXFUNC)
0056 LSTSPC(CTRI1 + IADD) = LSTSPC(CTRI1 + NXFUNC)
0057 LNKSPC(CTRI1 + IADD) = LNKSPC(CTRI1 + NXFUNC)
0058 FIGSPC(CTRI1 + IADD) = FIGSPC(CTRI1 + NXFUNC)
0059 FIGSPC(CTRI1 + IADD) = FIGSPC(CTRI1 + IADD) .OR. FI5MSK
0060 IF((FIGSPC(CTRI1 + IADD) .AND. FI6MSK) .EQ. 0) GO TO 270
C IF THE MOVED FUNCTION IS A MVI, THE POINTER FROM THE LAST VALUE OF
C THE LIST TO THE HEAD MUST BE UPDATED.
0062 KVAL = LSTSPC(CTRI1 + IADD)
0063 250 KVAL = LSTSPC(CTRI1 + KVAL)
0064 IF((FIGSPC(CTRI1 + LSTSPC(CTRI1 + KVAL)) .AND. FI0MSK) .EQ. 0)
GO TO 250
0066 LSTSPC(CTRI1 + KVAL) = IADD
0067 270 LOC = NXFUNC
C RETURN LOCATION TO AVAILABLE SPACE
0068 300 NODSPC(CTRI1 + LOC) = NODSPC(CTRI1 + REGASP)
0069 LSTSPC(CTRI1 + LOC) = REGASP
0070 LNKSPC(CTRI1 + LOC) = 0
0071 FIGSPC(CTRI1 + LOC) = 0
0072 NODSPC(CTRI1 + LSTSPC(CTRI1 + LOC)) = LOC
0073 LSTSPC(CTRI1 + NODSPC(CTRI1 + LOC)) = LOC
0074 NUMRET = NUMRET + 1
0075 RETURN
C
C FUNCTION TO BE DELETED IS NOT THE HEAD OF A CONFLICT LIST.
C THE FUNCTION PRECEDING THIS (FUNCTION BEING DELETED) MUST POINT TO
C THE FUNCTION FOLLOWING THIS
0076 400 LNKSPC(CTRI1 + LAST) = LNKSPC(CTRI1 + THIS)
0077 GO TO 300
C

```

```

C *** INDEXED DELETE
C
C FUNCTION MUST BE A MVI OR, IF SVI, ABS(IPOS) = 1 WITH PROPER TYPE.
C DELETE VALUE AT LOC. DEFEAT SAVED INDEX FOR THIS LIST UNTIL AFTER
C NEXT RETRIEVAL.
0078 500 FIGSPC(CTRL1 + THIS) = FIGSPC(CTRL1 + THIS) .OR. FI4WSK
C
C INDEXED DELETE CAN BE REDUCED TO FOUR CASES. DELETE VALUE IN
C FIRST, MIDDLE, OR LAST POSITION ON LIST, OR REDUCE TO SVI.
C
0079     NEXT = LSTSPC(CTRL1 + LOC)
0080     NPRIOR = LNKSPC(CTRL1 + LOC)
C
C IS LOC THE LAST POSITION IN THE LIST ?
0081     IF(NEXT .EQ. THIS) GO TO 600
C
C IS LOC THE FIRST POSITION IN THE LIST ?
0083     IF(LSTSPC(CTRL1 + NPRIOR) .EQ. THIS) GO TO 700
C
C VALUE IS IN A MIDDLE POSITION IN THE LIST. RECONNECT VALUES
C PRECEEDING AND FOLLOWING LOC.
C
0085     LSTSPC(CTRL1 + NPRIOR) = NEXT
0086     LNKSPC(CTRL1 + NEXT) = NPRIOR
0087     GO TO 300
C
C DELETE VALUE IN LAST POSITION IN LIST
0088 600 LSTSPC(CTRL1 + NPRIOR) = NEXT
0089     NEXT1 = LSTSPC(CTRL1 + NEXT)
0090     LNKSPC(CTRL1 + NEXT1) = NPRIOR
0091     IF((FIGSPC(CTRL1 + LOC) .AND. FLAG11) .NE. 0)
0093         1 FIGSPC(CTRL1 + NPRIOR) = FIGSPC(CTRL1 + NPRIOR) .OR. FLAG11
        GO TO 800
C
C DELETE VALUE IN FIRST POSITION IN LIST
0094 700 LNKSPC(CTRL1 + NEXT) = NPRIOR
0095     LSTSPC(CTRL1 + THIS) = NEXT
C
C CONVERT TO A SINGLE VALUE LIST ?
C
0096 800 IF(LNKSPC(CTRL1 + NPRIOR) .NE. NPRIOR) GO TO 300
C IF DELETING LAST VALUE, RESET NEXT TO FIRST VALUE
        IF(NEXT .EQ. THIS) NEXT = NPRIOR
0098     LSTSPC(CTRL1 + THIS) = NOPSPC(CTRL1 + NEXT)
0100     FIGSPC(CTRL1 + THIS) =
0101         1 (FIGSPC(CTRL1 + THIS) .OR. FIGSPC(CTRL1 + NEXT)) .AND. NF1023
0102     FIGSPC(CTRL1 + NEXT) = 0
0103     LNKSPC(CTRL1 + NEXT) = 0
0104     NOPSPC(CTRL1 + NEXT) = NOPSPC(CTRL1 + REGASP)
0105     LSTSPC(CTRL1 + NEXT) = REGASP
0106     NOPSPC(CTRL1 + LSTSPC(CTRL1 + NEXT)) = NEXT
0107     LSTSPC(CTRL1 + NOPSPC(CTRL1 + NEXT)) = NEXT
0108     NUMRET = NUMRET + 1
0109     GO TO 300
0110     END

```

```

C
0001 SUBROUTINE LVSETP
0002 IMPLICIT INTEGER(A-Z)
0003 REAL*4 DEFEXT,CORE, TOP, BOTTOM
0004 LOGICAL*1 SNGLBK, SETUP, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR, DL1STR,
1 DL2STR, DUMPF1, CURENT, IN2TMP, FD2TMP, DL2TMP, FINDFI,
2 DLETFI, NSRTFI, REORG, FULL, RPLACE
0005 COMMON /IVARGS/ IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, ITESTR,
1 INCIUD, INDXON, IVALS(10), ITYP1(10), SRCSUF,
2 LNKSUF, SNKSUF, INSTYP
0006 COMMON /IVREGS/ CURPAG(4), REQPG(4), LSTVPG(4), MSARET,
1 HREQPG, NYTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0007 COMMON /IVMASK/ MWRITE, NOTUSD, NEWCON, FIGWSK, MASKSF, MASKPF
0008 COMMON /IVFLAG/ FI0MSK, FI1MSK, FI2MSK, FI3MSK, FI4MSK, FI5MSK, FI067,
1 FLAG8, FLAG9, FLAG10, FLAG11, FLAG12, FLAG13, FLAG14,
2 FLAG15
0009 COMMON /IVRAND/ PRIME, SEED, NROW, DNODE, DROW, OLDNOP, LISTSZ,
1 GRNTBI(256)
0010 COMMON /IVCRNT/ REGASP, CTRLPT, CTRL1, LEASTV, NTFREE, FREE, DREGSP,
1 NSA, PAGLOC, CURENT
0011 COMMON /IVBUFR/ PAGSZE, NWCHAN, OLCHAN, CMPAND, PAGHDR, BUFSZE, DIRSZE,
1 INCORE, HDRSZE, MSADIR, SUFSZE, BIKSZE, DIRBLK, PAGHD4
0012 COMMON /IVHDVI/ THMSA, REGAS, PAGENO, CONTNO, INSDI,
1 USECT, HDRFIG, READVI, OLDNDP, DNODEH, NROWH, DROWH
0013 COMMON /IVSWIT/ SETUP, SNGLBK, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
1 DL1STR, DL2STR, IN2TMP, FD2TMP, DL2TMP, DUMPF1,
2 FINDFI, DLETFI, NSRTFI
0014 COMMON /IVVSEQ/ ISEQSZ, ISOPOS, LASTSO, SEQSPC(1)
0015 COMMON /IVPRAM/ BUFILOC, LENGTH, IERR, ERRNUM, BINARY, NCT, NODE, PAGES,
1 LUN
0016 COMMON /IVSTAK/ CURLEV, NUMVAR, STACK(1)
0017 COMMON /IVUTIL/ FILSPC(39), DEFEXT(2)
0018 COMMON /IVINS1/ REORG, FULL, RPLACE
0019 COMMON /IVRUN/ RUNTYP, CORE
0020 COMMON /IVVTR1/ NODSPC(1)
1 /IVVTR2/ LSTSPC(1)
2 /IVVTR3/ LNKSPC(1)
3 /IVVTR4/ FIGSPC(1)
C
D
0021 PAUSE 'IN LVSETP'
0022 IF(SNGLBK) GO TO 120
0023 PAGHDR = PAGSZE + HDRSZE
0024 BIKSZE = PAGHDR*64
0025 PAGHD4 = 4*PAGHDR
0026 DIRSZE = 64*(INCORE/64) + 1)
0027 DIRBLK = DIRSZE/64
0028 BUFSZE = DIRSZE + (INCORE*PAGHDR)
C
0029 TYPE 1
0030 ! FORMAT(' PLEASE ENTER FILE NAMES OF OLD AND NEW GRAPHS'//
1 ' IN COMMAND STRING FORMAT (NEW.EXT = OLD.EXT)')//
2 ' .GRF IS ASSUMED EXTENSION')//
0031 IF(ICS1(FILSPC, DEFEXT, , 0) .NE. 0) STOP 'INVALID COMMAND STRING'
C RUN TYPE 1 - CREATE NEW GRAPH

```

```

C      RUN TYPE 2 - UPDATE OLD GRAPH
C      RUN TYPE 3 - QUERY OLD GRAPH
0033  IF(RUNTYP .EQ. 3) GO TO 100
C      ASSIGN CHANNEL TO OUTPUT (NEW) GRAPH
0035  NWCHAN = IGETC()
0036  IF(NWCHAN .IT. 0) STOP 'NO OUTPUT CHANNEL AVAILABLE'
0038  IF(IFETCH(FILSPC(1)) .NE. 0) STOP 'OUTPUT DEVICE HANDLER FETCH
1 FAILURE'
0040  IF(ENTER(NWCHAN,FILSPC(1),0) .IT. 0) STOP 'ENTRY FAILURE'
0042  IF(RUNTYP .EQ. 1) GO TO 110
C      ASSIGN CHANNEL TO INPUT (OLD) GRAPH
0044  100 OICHAN = IGETC()
0045  IF(OICHAN .IT. 0) STOP 'NO INPUT CHANNEL AVAILABLE'
0047  IF(IFETCH(FILSPC(16)) .NE. 0) STOP 'INPUT DEVICE HANDLER FETCH
1 FAILURE'
0049  IF(LOOKUP(OICHAN,FILSPC(16)) .IT. 0) STOP 'INPUT FILE LOOKUP
1 FAILURE'
C      READ OLD GRAPH INTO BUFFER AS DEFINED BY STORED IN-CORE DIRECTORY
0051  CALL LVFECH
D      PAUSE ' LEAVING LVSETP 1'
0052  RETURN
C
C CREATION RUN
0053  110 READCT = 1
0054  USECNT = 1
C
0055  120 SFED = PRIME/2
0056  NROW = SEED
0057  OLDNOP = SEED - PRIME
0058  DROW = PRIME
0059  DNODE = PRIME
0060  LISTSZ = 1
0061  REGASP = 1
C SET UP SINGLE BLOCK?
0062  IF(SNGIBK) GO TO 160
0064  140 DO 145 I = 1,64
0065  J = 4*(I-1)
0066  GRNTBI(J + OLDNOP) = OLDNOP
0067  GRNTBI(J + DNODEH) = DNODE
0068  GRNTBI(J + NROWH) = NROW
0069  145 GRNTBI(J + DROWH) = DROW
0070  TOP = INCORE-1
0071  BOTTOM = INCORE
0072  IF(BOTTOM .EQ. 0) BOTTOM = 1
0074  CORE = TOP/BOTTOM
C SET UP DIRECTORY AVAILABLE SPACE
0075  DRECSF = 1
0076  DO 150 I = 2,DIRSZ
0077  NOISPC(I) = 1-1
0078  LSTSPC(I-1) = 1
0079  LNKSPC(I) = 0
0080  150 FIGSPC(I) = FI3MSK
0081  NODSPC(I) = DIRSZ
0082  LNKSPC(I) = 0
0083  FIGSPC(I) = FI3MSK
0084  LSTSPC(DIRSZ) = 1
C

```

```

C SET UP WRKSPC OR SINGLE C. PT.
0085   CTRIPT = DIRSZ
0086   ISTLOC = DIRSZ + HDRSZ
0087 160   ENLOC = CTRIPT + PAGHDR
0088       CTRL1 = CTRIPT + HDRSZ
0089       CNTRL1 = CTRL1 + 1
0090       DO 170 I = CNTRL1, ENLOC
0091       LNKSPC(I) = 0
0092 170   FIGSPC(I) = FI3MSK

C
C INITIALIZE AVAILABLE SPACE RING STRUCTURE OF THE REQUESTED CONTROL
C POINT OF WRKSPC. IF THIS IS AT THE BEGINNING OF A CREATION RUN,
C IT IS C. PT. #1, THEN COPY TO THE OTHER C. PTS.
0093   SETUP = .TRUE.
0094   CALL LVGRN(REGASP)
0095   OLD=REGASP
0096   DO 180 I=2, PAGSZ
0097   CALL LVGRN(NEW)
0098   NODSPC(NEW + CTRL1) = OLD
0099   LSTSPC(OLD + CTRL1) = NEW
0100 180   OLD=NEW
0101       NODSPC(REGASP + CTRL1) = OLD
0102       LSTSPC(OLD + CTRL1) = REGASP
0103       NROW=SEED
0104       OI DNO = SEED - PRIME
0105       DROW = PRIME
0106       DNODE=PRIME
0107       SETUP = .FALSE.
0108       LIST = INCORE
0109       IF(SNGIBK) LIST = 1
0110       DO 200 I=1, LIST
C
C SET HEADER WORDS FOR THE CONTINUANTS
0112   NODSPC(CTRIPT+REGAS)=REGASP
0113   LNKSPC(CTRIPT+INDEL)=0
0114   LNKSPC(CTRIPT+USECT)=0
0115   FIGSPC(CTRIPT+HDRFIG) = 0
0116   FIGSPC(CTRIPT+READVI)=READCT
0117   IF(SNGIBK) GO TO 200
0118   IF(I.EQ. 1) GO TO 195
C COPY AS RING STRUCTURE TO REMAINING C. PTS.
0121   DO 190 K=1, PAGSZ
0122   NODSPC(CTRL1 + K) = NODSPC(ISTLOC + K)
0123   LSTSPC(CTRL1 + K) = LSTSPC(ISTLOC + K)
0124   LNKSPC(CTRL1 + K) = 0
0125 190   FIGSPC(CTRL1 + K) = FI3MSK
0126 195   CTRIPT=CTRIPT+PAGHDR
0127       CTRL1=CTRIPT+HDRSZ
0128 200   CONTINUE
0129   IF(.NOT. SNGIBK) GO TO 210

C
C COMPLETE HEADER WORDS FOR SINGLE CONTINUANT
C ASSUME CORRECT OUTCORE DIRECTORY BLOCK IS IN CORE
0131   NODSPC(CTRIPT + THMSA) = NXTMSA
0132   LSTSPC(CTRIPT + PAGENO) = REQAG(1)
0133   LSTSPC(CTRIPT + CONTNO) = REQAG(2)
0134   NXTMSA = NXTMSA + BKSZ

```

```

0135 D PAUSE ' LEAVING LVSETP 2'
      RETURN
C
C CREATE CONTINUANTS BY PAGE ORDER
0136 210 PAGNUM = 0
0137      CTRIPT = DIRSZ
0138      CTRL1 = CTRIPT + HDRSZ
C
C CREATE PAGE PAGNUM, CONTINUANT CONTIN.
C COMPLETE HEADER WORDS FOR CONTINUANTS
0139      LIST = HREOPG
0140      DIRCNT = 0
0141      DIRPAG = 1
C INITIALIZE NXTMSA TO LOCATION OF PAGE 1, CONT 0
0142      NXTMSA = DIRBIK + 18
0143      DO 300 I=1,LIST
0144          PAGNUM = PAGNUM+1
0145          NUMCON = STACK(PAGNUM) + 1
0146          DIRCNT = DIRCNT + 1
0147          IF(DIRCNT .NE. 1) GO TO 215
0149      DO 214 J=1,256
0150 214 OUTDIR(J) = 0
C
0151 215 DO 220 K=1,NUMCON
0152      CONTIN = K-1
0153      NODSPC(CTRLPT + THMSA) = NXTMSA
0154      LSTSPC(CTRLPT + PAGENO) = PAGNUM
0155      LSTSPC(CTRLPT + CONTNO) = CONTIN
C OUTPUT UNUSED CONTINUANT TO DISK
0156      LENGTH = PAGHDR
0157      BUFIOC = CTRLPT + 1
0158      ERRNUM = 1
0159      MSA = NODSPC(CTRLPT + THMSA)
0160      CALL LVPAGW
C ENTER CONTINUANT LOCATION INTO OUTCORE DIRECTORY
0161      OUTLOC = 1 + 64*(DIRCNT-1) + CONTIN
0162      OUTDIR(OUTLOC) = MSA
0163      NXTMSA = NXTMSA + BIKSZ
C UPDATE CONTROL POINTER
0164      CTRIPT=CTRLPT+PAGHDR
0165      IF(CTRLPT.GE.BUFSZ) CTRIPT=DIRSZ
0167 220 CONTINUE
C SAVE THIS BLOCK OF THE OUTCORE DIRECTORY IF ALL 4 SEGMENTS ARE FILLED
0168      IF(DIRCNT .IT. 4) GO TO 300
0170      CALL LVDRWR
0171      DIRCNT = 0
0172      DIRPAG = DIRPAG + 1
0173 300 CONTINUE
C
C SAVE MOST RECENT OUTCORE DIRECTORY BLOCK IF NECESSARY
0174      IF(DIRCNT .EQ. 0) GO TO 310
0176      CALL LVDRWR
C
C ZERO OUT REMAINING UNUSED OUTCORE DIRECTORY BLOCKS
0177 310 DO 312 I=1,256
0178 312 OUTDIR(I) = 0
C

```

```

0179 315 IF(DIRPAG.EQ.16) GO TO 320
0181 DIRPAG = DIRPAG + 1
0182 CALL LVDRWR
0183 GO TO 315

C
C BRING FIRST OUTCORE DIRECTORY BLOCK INTO MAIN MEMORY
0184 320 REOPAG(1) = 1
0185 REOPAG(2) = 0
0186 CALL LVMSA(DUMMY)

C
C IF IT IS NOT THERE, BRING PAGE 1, CONTINUANT 0 BACK INTO CORE
0187 CTRLPT = DIRSZ
0188 CTRL1 = CTRLPT + HDRSZ
0189 IF((LSTSPC(CTRLPT + PAGENO).EQ.1).AND.
1 (LSTSPC(CTRLPT + CONTNO).EQ.0)) GO TO 340
0191 MSA = OUTDIR(1)
0192 BUFLOC = CTRLPT + 1
0193 LENGTH=PAGHDR
0194 ERRNUM = 2
0195 CHAN = NWCHAN
0196 CALL LVPAGR(CHAN)

C
C INSERT INCORE CONTINUANTS INTO DIRECTORY PAGE
0197 340 CTRLPT = - HDRSZ
0198 CTRL1 = 0
0199 TMPSZ = PAGSZ
0200 PAGSZ = DIRSZ
0201 CNTRL1 = DIRSZ - PAGHDR
0202 INDXON = 0
0203 NVAL=1
0204 INSTYP=1
0205 DO 400 K=1,INCORE
0206 CNTRL1 = CNTRL1 + PAGHDR
0207 PAGE = LSTSPC(CNTRL1 + PAGENO)
0208 CONT = LSTSPC(CNTRL1 + CONTNO)
0209 IARG = CONT + 1
0210 IFUNC = PAGE
0211 SRCSUF = IARG
0212 LNKSUF = IFUNC

D PAUSE 'B FIND'
0213 CALL LVFIND

D PAUSE 'AFT FIND'
0214 IVALS(1)= CNTRL1
0215 ITYP1(1)=1
0216 SNKSUF = IVALS(1)
0217 CALL LVNSRT
0218 400 CONTINUE
0219 DREGSP=REGASP
0220 REGASP=1
0221 PAGSZ = TMPSZ

C
C ESTABLISH REGISTERS
C PAGE 1, CONT 0 IS:
C HIGHEST ACTIVE PAGE
C CURRENT PAGE - CONT
C PREVIOUS CURRENT PAGE - CONT
C REQUESTED PAGE - CONT

```

```

0222      C      HACTPG(1) = 0
0223      HACTPG(2) = OUTDIR(1)
0224      CTRLPT = DIRSZ
0225      CTRL1 = CTRLPT + HDRSZ

0226      C      CURPAG(1) = 1
0227      CURPAG(2) = 0
0228      CURPAG(3) = OUTDIR(1)
0229      CURPAG(4) = CTRLPT

0230      C      REOPAG(1) = CURPAG(1)
0231      REOPAG(2) = -2
0232      REOPAG(3) = CURPAG(3)
0233      REOPAG(4) = CURPAG(4)
0234      CURENT = .TRUE.

0235      C      DECLARE THE LAST CONTROL POINT AS AVAILABLE.
0236      LEASTV = FREE
0237      CNTRL1 = BUFSIZE - PAGEHDR
0238      LSTVPG(1) = LSTSPC(CNTRL1 + PAGENO)
0239      LSTVPG(2) = LSTSPC(CNTRL1 + CONTNO)
0240      LSTVPG(3) = NODSPC(CNTRL1 + THSMSA)
0241      LSTVPG(4) = CNTRL1
0242      D      PAUSE ' LEAVING LVSETP 3'
          RETURN
          END

```

```

C
C
0001 BLOCK DATA
0002 IMPLICIT INTEGER(A-Z)
0003 REAL*4 DEFEXT
0004 LOGICAL*1 SNGIBK, SETUP, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR, DL1STR,
1 DL2STR, DUMPF1, CURENT, IN2TMP, FD2TMP, DL2TMP, FINDFI,
2 DLETFI, NSRTFI, REORG, FULL, RPLACE
0005 COMMON /IVARGS/ IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, ITESTR,
1 INCLUD, INDXON, IVALS(10), ITYP1(10), SRCSUF,
2 LNKSUF, SNKSUF, INSTYP
0006 COMMON /IVFLAG/ FI0MSK, FI1MSK, FI2MSK, FI3MSK, FI4MSK, FI5MSK, FI67,
1 FIAG8, FIAG9, FIAG10, FIAG11, FIAG12, FIAG13, FIAG14,
2 FIAG15
0007 COMMON /IVCRNT/ REGASP, CTRIPT, CTRI1, LEASTV, NTFREE, FREE, DREGSP,
1 MSA, PAGLOC, CURENT
0008 COMMON /IVBUFR/ PAGSZ, NWCHAN, OICHAN, COMPAND, PAGHDR, BUFSZ, DIRSZ,
1 INCORE, HDRSZ, MSADIR, SUFSZ, BKSZ, DIRBK, PAGHD4
0009 COMMON /IVHDI/ THMSA, REGAS, PAGENO, CONTNO, INSDI,
1 USECT, HDRFIG, READVI, OI DNDH, DNODEH, NROWH, DROWH
0010 COMMON /IVSWIT/ SETUP, SNGIBK, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
1 DL1STR, DL2STR, IN2TMP, FD2TMP, DL2TMP, DUMPF1,
2 FINDFI, DLETFI, NSRTFI
0011 COMMON /IVPRAM/ BUFI OC, LENGTH, IERR, ERRNUM, BINARY, BCD, MODE, PAGES,
1 LUN
0012 COMMON /IVMASK/ MWRITE, NOTUSD, NEWCON, FI GMSK, MASKSF, MASKPF
0013 COMMON /IVUTIL/ FILSPC(39), DEFEXT(2)
0014 COMMON /IVINS1/ REORG, FULL, RPLACE
C
0015 DATA FI0MSK, FI1MSK, FI2MSK, FI3MSK, FI4MSK, FI5MSK, FI67, FIAG8,
1 FIAG9, FIAG10, FIAG11, FIAG12, FIAG13, FIAG14, FIAG15
2 /"200", "100", "40", "20", "10", "4", "3",
3 "400", "1000", "2000", "4000", "10000", "20000", "40000", "100000"/
0016 DATA THMSA, REGAS, PAGENO, CONTNO, INSDI, USECT,
1 HDRFIG, READVI, OI DNDH, DNODEH, NROWH, DROWH
2 /1,2,1,2,1,2,1,2,1,2,3,4/
0017 DATA MWRITE, NOTUSD, NEWCON, FI GMSK, MASKSF, MASKPF
1 /"4", "2", "1", "7", "1777", "176000"/
0018 DATA SUFSZ, NTFREE, FREE, BINARY, BCD, INCLUD, INSTYP, MSADIR,
1 IPOS, ITYP, NVAL, ITESTR, INDXON, HDRSZ, PAGLOC, OI CHAN
2 /10,0,1,0,1,0,0,2,
3 1,3,1,-1,0,2,-1,-1/
0019 DATA ITYP1/10*0/
0020 DATA IVALS/10*0/
0021 DATA SETUP, SNGIBK, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
1 DL1STR, DL2STR, IN2TMP, FD2TMP, DL2TMP, DUMPF1,
2 FINDFI, DLETFI, NSRTFI, REORG, FULL
3 /.FALSE...FALSE...FALSE...FALSE...FALSE...FALSE...FALSE..
4 .FALSE...FALSE...FALSE...FALSE...FALSE...FALSE...FALSE..
5 .TRUE...TRUE...TRUE...FALSE...FALSE..
0022 DATA DEFEXT /6GRGFGRF,6GRGFGRF/
C
C IF THE FOLLOWING FLAGS ARE ON. THEY REPRESENT THE FOLLOWING:
C
C FI0MSK- HEAD OF A MULTIVALUED LIST
C FI1MSK- THE CELL IS IN WORKING SPACE, NOT AVAILABLE SPACE

```

C FI2MSK- VALUE ON A MULTIVALUE LIST
 C FI3MSK- A NODE HAS BEEN DEFINED WITH THIS RELATIVE ADDRESS AS ITS VALUE
 C FI4MSK- THE SAVED INDEX OPERATION IS NOT IN EFFECT FOR THIS LIST
 C FI5MSK- HEAD OF A CONFLICT LIST
 C FI667 - 00- A RANDOM NUMBER
 C 01- NUMERIC DATA (INTEGER)
 C 10- A CONTINUING STRING OF HOLLERITH DATA
 C 11- THE ONLY, OR FINAL, CELL IN A HOLLERITH DATA STRING
 C FLAG8 - THE CELL CONTAINS A POINTER TO SEQUENCE SPACE
 C FLAG9 - UNUSED
 C FLAG10- MULTIVALUE LIST CONTINUATION FLAG (FUNCTION CONTINUES ON
 C PREVIOUS CONTINUANT. THIS CONTINUANT DOES NOT CONTAIN
 C THE BEGINNING OF THE LIST
 C FLAG11- MULTIVALUE LIST CONTINUATION FLAG (FUNCTION CONTINUES ON
 C NEXT CONTINUANT)
 C FLAG12- REORG INHIBIT FLAG
 C FLAG13- THE CELL IS THE HEAD OF A MULTIVALUE LIST WHICH IS A
 C NON-MOVABLE CONTINUATION OF A LIST ON SOME OTHER CONTINUANT
 C

0023

END

```

C
C
0001 SUBROUTINE LVFECH
0002 IMPLICIT INTEGER(A-Z)
0003 REAL*4 CORE,TOP,BOTTOM
0004 LOGICAL*1 SNGLBK, SETUP, NYTRAN, IN1STR, IN2STR, FD1STR, FD2STR, DL1STR,
1 DL2STR, DUMPF1, CURENT, IN2TMP, FD2TMP, DL2TMP, FINDFI,
2 DLETF1, NSRTFI
0005 COMMON /LVARGS/ IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, ITESTR,
1 INCLUD, INDXON, IVALS(10), ITYP1(10), SRCSUP,
2 LNKSUF, SNKSUF, INSTYP
0006 COMMON /LVREGS/ CURPAG(4), REOPAG(4), LSTVPG(4), MSARET,
1 HREOPG, NYTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0007 COMMON /VMSK/ MWRITE, NOTUSP, NEWCON, FLGMSK, MASKSF, MASKPF
0008 COMMON /VFLAG/ FL0MSK, FL1MSK, FL2MSK, FL3MSK, FL4MSK, FL5MSK, FL67,
1 FLAG8, FLAG9, FLAG10, FLAG11, FLAG12, FLAG13, FLAG14,
2 FLAG15
0009 COMMON /VVRAND/ PRIME, SEED, NROW, DNODE, DROW, OLDNOP, LISTSZ,
1 GRNTBI(256)
0010 COMMON /VCRNT/ REGASP, CTRLPT, CTRL1, LEASTV, NTFREE, FREE, DREGSP,
1 MSA, PAGLOC, CURENT
0011 COMMON /VBUFR/ PAGSIZE, NWCHAN, OLCHAN, CMPANT, PAGHDR, BUFSZE, DIRSZE,
1 INCORE, HDRSZE, MSADIR, SUFSZE, BLKSZE, DIRBIK, PAGBD4
0012 COMMON /VBDVI/ THMSA, REGAS, PAGENO, CONTNO, INSDI,
1 USECT, HDRFIG, READVI, OLDNIH, DNOPEH, NROWB, DROWB
0013 COMMON /VSWIT/ SETUP, SNGLBK, NYTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
1 DL1STR, DL2STR, IN2TMP, FD2TMP, DL2TMP, DUMPF1,
2 FINDFI, DLETF1, NSRTFI
0014 COMMON /VVSEQ/ ISEQSZ, ISOPOS, LASTSO, SEQSPC(1)
0015 COMMON /VPRAM/ BUFLOC, LENGTH, IERR, ERRNUM, BINARY, BCD, MODE, PAGES,
1 LUN
0016 COMMON /VSTAK/ CURLEV, NUMVAR, STACK(1)
0017 COMMON /VVRUN/ RUNTYP, CORE
0018 COMMON /VUSER/ USER(228)
0019 COMMON /VVTR1/ NODSPC(1)
1 /VVTR2/ LSTSPC(1)
2 /VVTR3/ LNKSPC(1)
3 /VVTR4/ FLGSPC(1)
C
C THIS ROUTINE READS A PREVIOUSLY CREATED GRAPH FROM DISK INTO THE GIRS
C BUFFER AND COPIES IT ONTO A NEW DISK FILE.
C
C READ IN SYSTEM VARIABLES
D PAUSE 'IN LVFECH'
0020 MSA = 0
0021 LENGTH = 256
0022 ERRNUM = 29
0023 IERR = IREADW(LENGTH, RWBUF(1), MSA, OLCHAN)
0024 DUMP = 1
0025 IF(IERR.I.T.0) CALL LVERR(DUMP)
C
0027 REGASP = RWBUF( 1)
0028 NYTMSA = RWBUF( 2)
0029 PAGSIZE = RWBUF( 3)
0030 PAGHDR = RWBUF( 4)

```

```

0031      BUFSIZE      = RWBUF( 5)
0032      DIRSIZE      = RWBUF( 6)
0033      DREGSP        = RWBUF( 7)
0034      INCORE         = RWBUF( 8)
0035      HDRSZ         = RWBUF( 9)
0036      HREQPG        = RWBUF(10)
0037      HACTPG(1)     = RWBUF(11)
0038      HACTPG(2)     = RWBUF(12)
0039      READCT        = RWBUF(13)
0040      BLKSZ         = RWBUF(14)
0041      SUFSZ         = RWBUF(15)
0042      DIRBLK        = RWBUF(16)
0043      PRIME         = RWBUF(17)
0044      SEED          = RWBUF(18)
0045      LISTSZ        = RWBUF(19)
0046      ISEQSZ        = RWBUF(20)
0047      DO 5 I = 1, 4
0048      CURPAG(I)     = RWBUF(20+I)
0049      LSTVPG(I)     = RWBUF(24+I)
0050      S              CONTINUE
0051      C              READ IN SAVED USER VARIABLES
0052      DO 7 I = 1, 228
0053      7              J = I + 28
0054      C              USER(I) = RWBUF(J)
0055      USECNT = 1
0056      LEASTV = NTFREE
0057      TOP = INCORE-1
0058      BOTTOM = INCORE
0059      CORE = TOP/BOTTOM
0060      C              READ IN GRN TABLE
0061      LENGTH = 256
0062      MSA = 1
0063      ERNUM = 30
0064      IERR = IREADW(LENGTH, GRNTBI(1), MSA, OLCHAN)
0065      DUMP = 1
0066      IF(IERR .LT. 0) CALL LVERR(DUMP)
0067      C              READ IN IN-CORE DIRECTORY
0068      MSA = 2
0069      LENGTH = DIRSIZE
0070      ERNUM = 31
0071      BUFLOC = 1
0072      CHAN = OLCHAN
0073      CALL LVPAGR(CHAN)
0074      C              COPY OUT-CORE DIRECTORY TO NEW DISK FILE
0075      DIRPAG = 17
0076      10      DIRPAG = DIRPAG - 1
0077      CALL LVDRRP(OLCHAN)
0078      CALL LVDWR
0079      IF(DIRPAG .GT. 1) GO TO 10
0080      C              COPY OLD GRAPH TO NEW DISK FILE
0081      BUFLOC = DIRSIZE + 1

```

```

0079     ERRNUM = 32
0080     HIPAGE = HACTPG(1)
0081     IF(HREQPG.GT. HIPAGE) HIPAGE = HREQPG
C SEQUENCE ON PAGES
0083     DO 30 PAGE = 1, HIPAGE
C SEQUENCE ON CONTINUANTS
0084     DO 20 I = 1, 64
0085     CONT = I-1
0086     REQPA(1) = PAGE
0087     REQPA(2) = CONT
0088     CALL LVMSA(CONNUM)
0089     IF(MSARET.IE. 0) GO TO 30
0091     MSA = MSARET
0092     CHAN = OLCHAN
0093     LENGTH = PAGHDR
0094     CALL LVPAGR(CHAN)
0095     CALL LVPAGW
0096     20 CONTINUE
0097     30 CONTINUE
C
C EXAMINE IN-CORE DIRECTORY AND BRING IN LISTED CONTINUANTS INTO CORE
0098     DO 50 I = 1, DIRSZ
0099     IF((FLGSPC(I).AND.F1MSK).EQ.0) GO TO 50
0101     REQPA(1) = NODSPC(I)
0102     REQPA(2) = I - REQPA(1) - 1
0103     IF(REQPA(2).LT. 0) REQPA(2) = REQPA(2) + DIRSZ
0105     CTRIPT = LSTSPC(I)
0106     CTRI1 = CTRIPT + HDRSZ
0107     CALL LVMSA(CONNUM)
0108     MSA = MSARET
0109     BUFI0C = CTRIPT + 1
0110     LENGTH = PAGHDR
0111     CALL LVPAGR(CHAN)
0112     50 CONTINUE
C
C ESTABLISH REGISTERS
0113     CTRIPT = CURPA(4)
0114     CTRI1 = CTRIPT + HDRSZ
0115     CURENT = .TRUE.
0116     REQPA(2) = -2
C READ IN SEQUENCE SPACE (LATER VERSION)
0117     TYPE 60
0118     60 FORMAT(/, ' GRAPH HAS BEEN PLACED INTO MEMORY',/)
0119     RETURN
0120     END

```

```

C
C
0001 SUBROUTINE LVGRN(NODE)
0002 IMPLICIT INTEGER(A-Z)
0003 LOGICAL*1 SNGLBK, SETUP, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR, DL1STR,
1 DL2STR, DUMPF1, CURENT, IN2TMP, FD2TMP, DL2TMP, FINDFI,
2 DLETF1, NSRTFI
0004 DIMENSION NODE(1)
0005 COMMON /LVREGS/ CURPAG(4), REQ PAG(4), LSTVPG(4), MSARET,
1 HREQPG, NXTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0006 COMMON /LVFLAG/ FL0MSK, FL1MSK, FL2MSK, FL3MSK, FL4MSK, FL5MSK, FL67,
1 FLAG8, FLAG9, FLAG10, FLAG11, FLAG12, FLAG13, FLAG14,
2 FLAG15
0007 COMMON /LVRAND/ PRIME, SEED, NROW, DNODE, DROW, OLDNOD, LISTSZ,
1 GRNTBI(256)
0008 COMMON /LVBUFR/ PAGSIZE, NWCHAN, OLCHAN, CMPAND, PAGHDR, BUFSZ, DIRSZ,
1 INCORE, HDRSZ, MSADIR, SUFSZ, BLKSZ, DIRBIK, PAGHD4
0009 COMMON /LVHDL/ THSWA, REGAS, PAGENO, CONTNO, INSDI,
1 USECT, HDRFIG, READVI, OLDNDH, DNODEH, NROWH, DROWH
0010 COMMON /LVSWIT/ SETUP, SNGLBK, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
1 DL1STR, DL2STR, IN2TMP, FD2TMP, DL2TMP, DUMPF1,
2 FINDFI, DLETF1, NSRTFI
0011 COMMON /LVVTR1/ NODSPC(1)
1 /VVTR2/ LSTSPC(1)
2 /VVTR3/ LNKSPC(1)
3 /VVTR4/ FIGSPC(1)

C
C THE PURPOSE OF THIS ROUTINE IS TO PROVIDE A SEQUENCE OF 'RANDOM'
C NUMBERS OF LENGTH LISTSZ TO THE REQUESTED PAGE (CONTINUANT = 0)
C
C PROGRAM INITIALIZATION ?
D PAUSE 'IN LVGRN'
0012 PAGE = 0
0013 IF(SETUP) GO TO 100

C
C IS THE PAGE DEFINED ?
0015 IF(REQ PAG(1) .GT. HACTPG(1)) HACTPG(1) = REQ PAG(1)
0017 IF(REQ PAG(1) .GT. 0) GO TO 50
0019 HACTPG(1) = HACTPG(1) + 1
0020 REQ PAG(1) = HACTPG(1)

C
C OBTAIN CURRENT VALUES FOR GRN PARAMETERS FOR REQUESTED PAGE
50 PAGE = REQ PAG(1)
0021 GRNDEX = 4*(PAGE - 1)
0022 OLDNOD = GRNTBI(GRNDEX + OLDNDH)
0023 DNODE = GRNTBI(GRNDEX + DNODEH)
0024 NROW = GRNTBI(GRNDEX + NROWH)
0025 DROW = GRNTBI(GRNDEX + DROWH)

C
0027 100 DO 200 J = 1, LISTSZ
0028 I = J
0029 NODE(I) = OLDNOD + DNODE
0030 OLDNOD = NODE(I)
0031 DNODE = DNODE + 1
0032 IF(NODE(I) .IE. PAGSIZE) GO TO 199

```

```

0034 C RESIDUE GENERATION ?
      IF(NROW.GT.PRIME) GO TO 150
0036 C ROW UPDATE
      NROW = NROW+SEED
0037      IF(NROW.GT.PRIME) NROW = NROW-PRIME
0039      NODE(1) = NROW
0040      OI DNOP = NODE(1)
0041      DNODE = PRIME+1
0042 C RESIDUE GENERATION ?
      IF(NODE(1).NE.SEED) GO TO 199
0044      NROW = 0
0045      DROW = PRIME
0046 C RESIDUE GENERATION
150      DROW = DROW+1
0047      NROW = NROW +DROW
0048      NODE(1) = NROW
0049      OI DNOP = NODE(1)
0050      DNODE = DROW
0051      IF(NODE(1).GT.PAGSIZE) GO TO 300
0053 C OUTPUT NODE
199      IF(SETUP .EQ. .TRUE.) RETURN
0055      NODE(1) = NODE(1).OR.IVIFSB(PAGE,SUFSIZE)
      C
0056 C UPDATE HEADER
250      GRNTBI (GRNDEX + OI DNOP) = OI DNOP
0057      GRNTBI (GRNDEX + DNODE) = DNODE
0058      GRNTBI (GRNDEX + NROWB) = NROW
0059      GRNTBI (GRNDEX + DROWB) = DROW
0060      LISTSZ=1
0061      RETURN
      C
0062 C ORIGINAL CREATION SEQUENCE IS EXHAUSTED, RECOVER IN DEFINED NOPS
      C BRING IN CONTINUANT ZERO OF THE REQUESTED PAGE IF NECESSARY
300      REQCON = REQAG(2)
0063      REQAG(2) = 0
0064      CALL LVEXCH
0065      DO 430 L = 1,LISTSZ
0066      DO 400 K=1,PAGSIZE
0067      LOC = CTRIP T + HDRSZ + K
0068      IF((FIGSPC(LOC).AND.FI3MSK).NE.0) GO TO 390
0070      NODE(L)=K
0071      FIGSPC(LOC)=FIGSPC(LOC).AND.FI3MSK
0072      GO TO 430
0073      390 IF(K .EQ. PAGSIZE) GO TO 440
0075      400 CONTINUE
0076      430 CONTINUE
0077 C BRING IN ORIGINAL REQUESTED (PAGE,CONT)
      REQAG(2) = REQCON
0078      CALL LVEXCH
0079      LISTSZ = 1
0080      RETURN
      C
0081      440 TYPE 450
0082      450 FORMAT(1H,'ERROR...NUMBER OF NOPS EXCEEDS REQUESTED MEMORY,.'/
      1 PROGRAM IS TERMINATED.')
0083      ERRNUM = 10
0084      DUMP = 0
0085      CALL LVERR(DUMP)
0086      STOP
0087      END

```

```

C
C
0001 SUBROUTINE LVNPAG
0002 IMPLICIT INTEGER(A-Z)
0003 LOGICAL*1 SNGI BK, SETUP, NXTRAN, IN1STR, IN2STR, PD1STR, PD2STR, DL1STR,
1 DL2STR, DUMPFI, CURENT, IN2TMP, PD2TMP, DL2TMP, FINDFI,
2 DLETFI, NSRTFI
0004 COMMON /IVREGS/ CURPAG(4), REQPG(4), LSTVPG(4), NSARET,
1 HREQPG, NYTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0005 COMMON /IVMASK/ NWRITE, NOTUSD, NEWCON, FIGMSK, MASKSF, MASKPF
0006 COMMON /IVCRNT/ REGASP, CTRI PT, CTRI 1, LEASTV, NTFREE, FREE, DRGSP,
1 NSA, PAGLOC, CURENT
0007 COMMON /IVBIFFR/ PAGSIZE, NWCHAN, OI CHAN, CNPAND, PAGEDE, BUFSZ, DIRSZ,
1 INCORE, HDRSZ, MSADIR, SUPSZ, BIKSZ, DIRBK, PAGEID4
0008 COMMON /IVHDI/ THMSA, REGAS, PAGENO, CONTNO, INSDI,
1 USECT, HDRFIG, READVI, OI DNTB, DNOPEH, NROWB, DROWB
0009 COMMON /IVSWIT/ SETUP, SNGI BK, NXTRAN, IN1STR, IN2STR, PD1STR, PD2STR,
1 DL1STR, DL2STR, IN2TMP, PD2TMP, DL2TMP, DUMPFI,
2 FINDFI, DLETFI, NSRTFI
0010 COMMON /IVPRAM/ BUFILOC, LENGTH, IERR, ERRNUM, BINARY, BCD, MODE, PAGES,
1 LUN
0011 COMMON /IVVTR1/ NONSPC(1)
1 /IVVTR2/ LSTSPC(1)
2 /IVVTR3/ LNKSPC(1)
3 /IVVTR4/ FIGSPC(1)
C
C THIS ROUTINE WILL PLACE A NEW PAGE (ZEROth CONTINUANT) INTO THE BUFFER.
C IF THE NEW PAGE EXCEEDS THE NUMBER OF PAGES ORIGINALLY REQUESTED BY THE
C USER: PUT IT ON DISK.
C REGISTERS AND IN-CORE AND OUT-CORE DIRECTORIES ARE UPDATED.
C
C DEFINE PAGE NO. AND UPDATE HIGHEST ACTIVE PAGE
D PAUSE IN LVNPAG
0012 HACTPG(1) = HACTPG(1)+1
0013 REQPG(1) = HACTPG(1)
C
C BRING IN OUT-CORE DIRECTORY PAGE AND DEFINE OUTLOC (LOC IN O-C D P)
0014 REQPG(2) = -1
0015 CALL LVMSA(CONNUM)
0016 REQPG(2) = 0
C
C ARE ANY PREALLOCATED PAGES (THAT HAVE ALREADY BEEN OPENED ON DISK)
C STILL AVAILABLE ?
0017 IF (HACTPG(1) .GT. HREQPG) GO TO 10
C
C REQ(P,0) WAS CREATED AT THE BEGINNING OF THE PROGRAM
0019 CALL LVEXCH
0020 HACTPG(2) = NSARET
0021 RETURN
C
C OPEN A PAGE-BLOCK IN THE BUFFER
0022 10 IF (LEASTV .EQ. NTFREE) CALL LVOPEN
C
C*** NEW PAGE MUST BE ADDED TO THE DISK IMMEDIATELY FOLLOWING THE LAST
C CREATED CONTINUANT.

```

```

0024      HACTPG(2) = NXTMSA
      C
      C PLACE LOCATION OF NEW PAGE INTO OUT-CORE DIRECTORY
0025      OUTDIR(OUTLOC) = HACTPG(2)
      C
      C SET UP AVAILABLE SPACE AND HEADER
0026      SNGIBK = .TRUE.
0027      CTRIPT = LSTVPG(4)
0028      CTRI1 = CTRIPT + HDRSZ
0029      CALL LVSETP
      C
      C PLACE EMPTY PAGE ON DISK
0030      LENGTH = PAGHDR
0031      BUFLOC = CTRIPT + 1
0032      ERRNUM = 23
0033      MSA = HACTPG(2)
0034      CALL LVPAGN
      C
      C UPDATE REGISTERS
0035      40 CURPAG(1) = HACTPG(1)
0036      CURPAG(2) = 0
0037      CURPAG(3) = HACTPG(2)
0038      CURPAG(4) = LSTVPG(4)
      C
      C PAGE HAS BEEN PLACED IN "LEAST VALUED BLOCK"
      C
      C UPDATE IN-CORE DIRECTORY
0039      CALL LVRPLC
      C
      C PROTECT THIS PAGE FROM BEING TAKEN OUT OF CORE BEFORE IT IS USED
0040      FIGSPC(CTRIPT+HDRFIG) = NOTUST .OR. NEWCON
0041      RETURN
0042      END

```

```

C
C
0001 SUBROUTINE LVNCON
0002 IMPLICIT INTEGER(A-Z)
0003 LOGICAL*1 SNGIBK, SETUP, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR, DL1STR,
1 DL2STR, DUMPFI, CURENT, IN2TMP, FT2TMP, DL2TMP, FINDFI,
2 DLETFI, NSRTFI
0004 COMMON /LVREGS/ CURPAG(4), REOPAG(4), LSTVPG(4), MSARET,
1 HREOPG, NYTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0005 COMMON /VWASK/ MWRITE, NOTUSD, NEWCON, FLGWSK, MASKSF, MASKPF
0006 COMMON /VCRNT/ REGASP, CTRLPT, CTRL1, LEASTV, NTFREE, FREE, DREGSP,
1 MSA, PAGLOC, CURENT
0007 COMMON /VBUF/ PAGSIZE, NWCHAN, OLCHAN, CMPAND, PAGHDR, BUFSIZE, DIRSZF,
1 INCORE, HDRSZF, MSADIR, SUFSZF, BIKSZF, DIRBIK, PAGHD4
0008 COMMON /VBDVI/ THSWA, REGAS, PAGENO, CONTNO, INSPDL,
1 USECT, HDRFIG, READVI, OI DNDH, DNODEH, NROWB, DROWB
0009 COMMON /VSWIT/ SETUP, SNGIBK, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
1 DL1STR, DL2STR, IN2TMP, FT2TMP, DL2TMP, DUMPFI,
2 FINDFI, DLETFI, NSRTFI
0010 COMMON /VPRAM/ BUFILOC, LENGTH, IERR, ERRNUM, BINARY, BCD, MODE, PAGES,
1 LUN
0011 COMMON /VVTRI/ NONSPC(1)
1 /VVTR2/ LSTSPC(1)
2 /VVTR3/ LNKSPC(1)
3 /VVTR4/ FLGSPC(1)
0012 DIMENSION CONIST(64)
0013 DATA CONIST /64*0/
C
C THIS ROUTINE PLACES AN UNUSED CONTINUANT OF AN ESTABLISHED PAGE
C INTO THE BUFFER. IF REQ(P,C) WAS NOT INITIALIZED AT THE BEGINNING
C OF THE PROGRAM, A CONTINUANT IS CREATED AND PLACED ON DISK.
C REGISTERS, IN-CORE, AND OUT-CORE DIRECTORIES ARE UPDATED.
C
C CONIST() IS A LIST OF HIGHEST ACTIVE CONTINUANTS FOR EACH PAGE
D PAUSE 'IN LVNCON'
0014 PAGE = REOPAG(1)
0015 HICONT = CONIST(PAGE)
0016 REOPAG(2) = HICONT + 1
0017 CONIST(PAGE) = REOPAG(2)
C
C BRING IN OUTCORE DIRECTORY
0018 CALL LVMSA(CONNUM)
C
C OPEN A PAGE-BLOCK IN THE BUFFER
0019 IF(LEASTV .EQ. NTFREE) CALL LVOPEN
C
C ARE ANY PREINITIALIZED CONTINUANTS STILL AVAILABLE ?
0021 IF(MSARET .GT. 0) GO TO Y0
C
C*** NEW CONTINUANT MUST BE ADDED TO THE DISK IMMEDIATELY FOLLOWING THE
C LAST CREATED CONTINUANT.
0023 MSA = NYTMSA
C
C PLACE LOCATION OF NEW PAGE INTO OUT-CORE DIRECTORY
0024 OUTDIR/OUTLOC + 1) = MSA

```

```

C
0025 C SET UP AVAILABLE SPACE AND HEADER
0026     SNGIBK = .TRUE.
0027     CTRIPT = LSTVPG(4)
0028     CTRI1 = CTRIPT + HDRSZ
    CALL LVSETP
C
0029 C PLACE EMPTY PAGE ON DISK
0030     LENGTH = PAGHDR
0031     BUFLOC = CTRIPT + 1
0032     ERRNUM = 25
0033     CALL LVPAGW
    GO TO 40
C
    C REQ(P,0) WAS CREATED AT THE BEGINNING OF THE PROGRAM
    C READ REQ(P,0) INTO CORE
0034 10   MSA = MSARET
0035     LENGTH = PAGHDR
0036     BUFLOC = LSTVPG(4) + 1
0037     ERRNUM = 26
0038     CHAN = NWCHAN
0039     CALL LVPAGR(CHAN)
C
    C UPDATE REGISTERS
0040 40   CURPAG(1) = REQPG(1)
0041     CURPAG(2) = REQPG(2)
0042     CURPAG(3) = MSA
0043     CURPAG(4) = LSTVPG(4)
C
    C PAGE HAS BEEN PLACED IN "LEAST VALUED BLOCK"
    C UPDATE IN-CORE DIRECTORY
0044     CALL LVRPLC
C
    C PROTECT THIS PAGE FROM BEING TAKEN OUT OF CORE BEFORE IT IS USED
0045     FIGSPC(CTRIPT+HDRFIG) = NOTUST .OR. NEWCON
0046     RETURN
0047     END

```

```

C
C
C
0001 SUBROUTINE LVEXCB
0002 IMPLICIT INTEGER(A-Z)
0003 LOGICAL*1 CURENT
0004 COMMON /I.VARGS/ IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, ITESTR,
1 INCLUD, INDXON, IVALS(10), ITYP1(10), SRCSTF,
2 LNKSUF, SNKSUF, INSTYP
0005 COMMON /I.VREGS/ CURPAG(4), REOPAG(4), LSTVPG(4), MSARET,
1 HREOPG, NYTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0006 COMMON /I.VCRNT/ REGASP, CTRIPT, CTRI1, LEASTV, NTFREE, FREE, DREGSP,
1 MSA, PAGLOC, CURENT
0007 COMMON /I.VBUFR/ PAGSIZE, NWCHAN, OI CHAN, CMPAND, PAGHDR, BUFSSZ, DIRSZ,
1 INCORE, HDRSZ, MSADIR, SUPSZ, BLKSZ, DIRBLK, PAGHD4
0008 COMMON /I.VPRAM/ BUFLOC, LENGTH, IERR, ERRNUM, BINARY, BCD, MODE, PAGES,
1 LUN
0009 COMMON /I.VBDVI/ THMSA, REGAS, PAGENO, CONTNO, INSDI,
1 USECT, HDRFLG, READVI, OI.DNIB, DNOPEH, NROWB, DROWB
0010 COMMON /I.VVTR1/ NOPSPC(1)
1 /I.VVTR2/ LSTSPC(1)
2 /I.VVTR3/ LNKSPC(1)
3 /I.VVTR4/ FLGSPC(1)

C
C THIS ROUTINE BRINGS THE REQUESTED (PAGE,CONT) INTO CORE IF NECESSARY
C AND UPDATES THE IN-CORE DIRECTORY AND "CURRENT" REGISTER
C TO REQUESTED (P,C)
C FAILURE RETURN IF MSARET LT 0
C
C IS REQ(P,C) IN CORE ?
D PAUSE 'IN LVEXCB'
D TYPE 1
D1 FORMAT(' REQ(1) REQ(2)')
D TYPE 2, REOPAG(1), REOPAG(2)
D2 FORMAT(2(2X,14))
0011 MSARET = 10000
0012 CALL LVPRCT
0013 IF(PAGLOC .GT. 0) RETURN

C
C BRING REQ(P,C) INTO CORE IF IT EXISTS
C LOCATE REQ(P,C) ON DISK
0015 CALL LVMSA(CONT)
C DOES REQ(P,C) EXIST ?
0016 IF(MSARET .LT. 0) GO TO 100
C MAKE A PAGE-BLOCK AVAILABLE IN THE GIRS BUFFER
0018 CALL LVOPEN
C BRING REQ(P,C) INTO "LEAST-VALUED" PAGE-BLOCK
0019 ERRNUM = 10
0020 LENGTH = PAGHDR
0021 MSA = MSARET
0022 BUFLOC = LSTVPG(4) + 1
0023 CHAN = NWCHAN
0024 CALL LVPAGR(CHAN)
C UPDATE "CURRENT PAGE" REGISTERS
0025 CTRIPT = BUFLOC - 1
0026 CTRI1 = CTRIPT + HDRSZ

```

```

0027     PAGLOC = CTRIPT
0028     CURPAG(1) = LSTSPC(CTRIPT + PAGENO)
0029     CURPAG(2) = LSTSPC(CTRIPT + CONTNO)
0030     CURPAG(3) = MSA
0031     CURPAG(4) = CTRIPT
0032     CURENT = .TRUE.
0033     REGASP = NOISPC(CTRIPT + REGAS)
      C UPDATE IN-CORE DIRECTORY
0034     CALL LVRPLC
      C UPDATE (P,C) HEADER
0035     READCT = READCT + 1
0036     FLGSPC(CTRIPT+READVL) = READCT
0037     LNKSPC(CTRIPT+USECT) = 0
0038     LEASTV = NTFREE
0039     RETURN
0040     END

```

```

C
C
0001 SUBROUTINE LVDRCT
0002 IMPLICIT INTEGER(A-Z)
0003 LOGICAL*1 CURENT
0004 COMMON /IVARGS/ IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, ITESTR,
1 INCLUD, INDXON, IVALS(10), ITYP1(10), SRCSUF,
2 LNKSUF, SNKSUF, INSTYP
0005 COMMON /IVREGS/ CURPAG(4), REQ PAG(4), LSTVPG(4), MSARET,
1 HREQPG, NHTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0006 COMMON /IVCRNT/ REGASP, CTRIPT, CTRI1, LEASTV, NTFREE, FREE, DREGSP,
1 MSA, PAGLOC, CURENT
0007 COMMON /IVBUFR/ PAGSIZE, NWCHAN, OICHAN, CMPAND, PAGHDR, BUFSZ, DIRSZ,
1 INCORE, HDRSZ, MSADIR, SUFSZ, BIKSZ, DIRBIK, PAGHD4
0008 COMMON /IVBDVI/ THMSA, REGAS, PAGENO, CONTNO, INSDI,
1 USECT, HDRFIG, READVI, OLDNDH, DNOPEH, NROWH, DROWH
0009 COMMON /IVPRAM/ BUFLOC, LENGTH, IERR, ERRNUM, BINARY, BCT, MODE, PAGES,
1 LUN
0010 COMMON /IVSTAK/ CURLEV, NUMVAR, STACK(1)
0011 COMMON /IVVTR1/ NODSPC(1)
1 /IVVTR2/ LSTSPC(1)
2 /IVVTR3/ LNKSPC(1)
3 /IVVTR4/ FLGSPC(1)
0012 DIMENSION TEMP(4)
C
C THIS ROUTINE SEARCHES THE DIRECTORY TO SEE IF THE REQUESTED PAGE-
C CONTINUANT (REQPAG) IS IN CORE. THE DIRECTORY IS A PAGE-BLOCK WHICH
C STAYS IN CORE. IT IS AT THE FIRST CONTROL POINT (CTRIPT = 0). DIRSZ
C IS THE DIRECTORY PAGE SIZE. FOR EACH PAGE-CONTINUANT THAT IS IN
C CORE, A TRIPLE IS STORED. THE SOURCE NODE IS THE CONTINUANT+1, THE
C LINK (OR KEY) IS THE PAGE NUMBER, AND THE VALUE IS THE LOCATION
C PRECEDING THE FIRST WORD OF THAT PAGE IN CORE (= CTRIPT).
C
C SUCCESS -- UPDATE CURPAG AND CTRIPT, CURENT = .TRUE.
C FAILURE -- PAGLOC = -1
C
C*** DOES THE REQUESTED PAGE-CONTINUANT = THE CURRENT PAGE-CONTINUANT ?
C
D PAUSE 'IN LVDRCT'
0013 IF((REQPAG(1) .EQ. CURPAG(1)) .AND. (REQPAG(2) .EQ. CURPAG(2)))
1 GO TO 40
0015 IF(INCORE .EQ. 1) GO TO 98
0017 GO TO 50
C
40 CTRIPT = CURPAG(4)
0018 CTRI1 = CTRIPT + HDRSZ
0019 PAGLOC = CTRIPT
0020 MSA = CURPAG(3)
0021 REGASP = NOTSPC(CTRIPT + REGAS)
0022 CURENT = .TRUE.
0023 RETURN
0024
C
C*** TEMPORARILY STORE SYSTEM VARIABLES FOR THE SEARCH
C

```

```

0025 S0    CALL LVSTAC
0026      OLD CPT = CTRI PT
      C
      C RESET SYSTEM VARIABLES FOR THE DIRECTORY
      C
0027      CTRI PT = -HDSZ
0028      CTRI 1 = 0
0029      PAGSZ = DIRSZ
0030      IARG  = REQ PAG(2)+1
0031      IFUNC  = REQ PAG(1)
0032      SRCSUF = IARG
0033      LNKSUF = IFUNC
      C
0034      CALL LVFIND
0035      PAGLOC = ITESTR*IVAL
      C
      C RESTORE SYSTEM VARIABLES
0036      CALL LVPOP
0037      IF(PAGLOC .LT. 0) GO TO 99
      C
      C PAGE-CONTINUANT HAS BEEN FOUND IN THE DIRECTORY.
0039      CTRI PT = PAGLOC
0040      CTRI 1 = CTRI PT + HDSZ
0041      MSA = NOISPC(CTRI PT + THMSA)
      C
      C UPDATE CURPAG
0042      CURPAG(1) = REQ PAG(1)
0043      CURPAG(2) = REQ PAG(2)
0044      CURPAG(3) = MSA
0045      CURPAG(4) = CTRI PT
0046      CURENT = .TRUE.
0047      REGASP = NOISPC(CTRI PT + REGAS)
0048      RETURN
      C
      C FAILURE
0049 98     PAGLOC = -1
0050 99     CURENT = .FALSE.
0051      CTRI PT = OLD CPT
0052      CTRI 1 = CTRI PT + HDSZ
0053      RETURN
0054      END

```

```

C
C
C
0001 SUBROUTINE LVMSA(CONNUM)
0002 IMPLICIT INTEGER(A-Z)
0003 COMMON /1VREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
1 HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
2 DIRCNT,OUTLOC,OUTDIR(256),RWBUFF(1)
0004 COMMON /1VBUFR/ PAGESZ,NWCHAN,OICHAN,CMPAND,PAGHDR,BUFSZ,DIRSZ,
1 INCORE,HDRSZ,MSADIR,SUFSZ,BKSIZE,DIRBLK,PAGHD4
C
C THIS ROUTINE BRINGS INTO OUTDIR() THE CORRECT OUTCORE DIRECTORY BLOCK
C IF NECESSARY. UPDATES DIRPAG AND DIRCNT.
C
C SUCCESS:
C RETURNS MSA OF (REQPAG(1),REQPAG(2)) IN MSARET.
C SETS CONNUM = REQPAG(2).
C FAILURE OR NEW CONTINUANT:
C MSARET = -1
C CONNUM = HIGHEST EXISTING CONTINUANT NUMBER OF REQPAG(1)
C UNDEFINED PAGE:
C CONNUM = -1
C
D PAUSE 'IN LVMSA'
0005 PAGE = REQPAG(1)
0006 CONT = REQPAG(2)
0007 ERRNUM = 11
0008 DUMP = 0
0009 IF((PAGE.GT. 64).OR.(CONT.GT. 63)) CALL LVERP(DUMP)
C COMPUTE OUTCORE DIRECTORY BLOCK
0011 NEWDIR = (PAGE - 1)/4 + 1
0012 DIRCNT = PAGE - 4*(NEWDIR - 1)
C BRING IN DIRECTORY BLOCK IF NECESSARY
0013 IF(NEWDIR.EQ. DIRPAG) GO TO 100
0015 DIRPAG = NEWDIR
0016 CHAN = NWCHAN
0017 CALL LVDRRD(CHAN)
C
C DETERMINE IF "ANY", "SPECIFIC", OR "NEW" CONTINUANT IS REQUESTED
0018 100 IF(CONT + 1) 200, 300, 210
C
C "ANY" -- SET TO ZERO
0019 200 CONT = 0
C
C "SPECIFIC"
0020 210 OUTLOC = 1 + 64*(DIRCNT-1) + CONT
0021 MSARET = OUTDIR(OUTLOC)
D
D9 TYPE 9
D9 FORMAT(' REQPAG(1),REQPAG(2),DIRPAG,DIRCNT,CONT,OUTLOC,MSARET')
D TYPE 10, REQPAG(1),REQPAG(2),DIRPAG,DIRCNT,CONT,OUTLOC,MSARET
D10 FORMAT(1X,8(2X,15))
0022 CONNUM = CONT
0023 IF(CONT.LT. 0) GO TO 220
0025 IF(MSARET.GT. 0) GO TO 220
0027 CONT = CONT - 1
0028 GO TO 210
C FAILURE ?

```

```

0029 220 IF (CONT .NE. REQ PAG(2)) MSARET = -1
0031 RETURN
C
C "NEW CONTINUANT" (PAGE MUST BE DEFINED, CONT NOT YET INITIALIZED)
0032 300 MSARET = -1
0033 CONNUM = -1
0034 310 CONT = CONT + 1
0035 IF (CONT .GE. 64) STOP 'REQUEST EXCEEDS ALLOWABLE NUMBER OF
1 CONTINUANTS'
0037 OUTLOC = 1 + 64*(DIRCNT-1) + CONT
0038 MSARET = OUTDIR(OUTLOC)
0039 IF (MSARET .I.E. 0) RETURN
0041 CONNUM = CONT
0042 GO TO 310
0043 END

```

```

C
C
0001 SUBROUTINE LVOPEN
0002 IMPLICIT INTEGER(A-Z)
0003 REAL*4 CORE
0004 COMMON /IVREGS/ CURPAG(4), REOPAG(4), LSTVPG(4), MSARET,
1 HREQPG, NXTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0005 COMMON /IVMASK/ MWRITE, NOTUSD, NEWCON, FIGMSK, MASKSF, MASKPF
0006 COMMON /IVCRNT/ REGASP, CTRIPT, CTRI1, LEASTV, NTFREE, FREE, DREGSP,
1 MSA, PAGLOC, CURENT
0007 COMMON /IVBUFR/ PAGSIZE, NWCHAN, OI CHAN, CMPAND, PAGHDR, BUFSIZE, DIRSZF,
1 INCORE, HDRSZF, MSADIR, SUFSZF, BIKSZF, DIRBIK, PAGWD4
0008 COMMON /IVBDVI/ THMSA, REGAS, PAGENO, CONTNO, INSPDL,
1 USECT, HDRFIG, READVI, OI DNDH, DNODEH, NROWH, DROWH
0009 COMMON /IVPRAM/ BUFI OC, LENGTH, IERR, ERRNUM, BINARY, BCD, MODE, PAGES,
1 LUN
0010 COMMON /IVRUN/ RUNTYP, CORE
0011 COMMON /IVVTR1/ NODSPC(1)
1 /IVVTR2/ LSTSPC(1)
2 /IVVTR3/ LNKSPC(1)
3 /IVVTR4/ FIGSPC(1)
C
C THE PURPOSE OF THIS ROUTINE IS TO MAKE A PAGE-BLOCK AVAILABIE
C IN WRKSPC.
C
C IF LEASTV = FREE, THEN LSTVPG CONTAINS THE CONTROL POINT FOR AN
C AVAILABIE PAGE-BLOCK
C
D PAUSE 'IN LVOPEN'
0012 IF(LEASTV .EQ. FREE) RETURN
C
C PREPARE TOTAL USAGE COUNT FOR LVALUE
0014 CALL LVSUM
C
C ROUTINE LVALUE WILL RETURN THE DISK AND IN-CORE LOCATIONS IN LSTVPG()
C OF A PAGE, CONT WHICH IT HAS DETERMINED TO BE OF LEAST IMMEDIATE USE
C TO THE SYSTEM.
C
0015 CALL LVALUE
C
C TEST WRITE-BIT OF LEAST VALUED CONTINUANT. THE WRITE-BIT INDICATES
C WHETHER OR NOT A PAGE HAS BEEN CHANGED SINCE IT WAS READ INTO
C MEMORY
C
0016 FIGLOC = LSTVPG(4) + HDRFIG
0017 IF((MWRITE .AND. FIGSPC(FIGLOC)) .EQ. 0) RETURN
C
C WRITE LEAST VALUED PAGE TO DISK
C
C ZERO OUT FLAGS
0019 FIGSPC(FIGLOC) = 0
0020 MSA = LSTVPG(3)
0021 LENGTH = PAGHDR
0022 ERRNUM = 27
0023 BUFI OC = LSTVPG(4) + 1
0024 CALL LVPAGW
0025 RETURN
0026 END

```

```

C
C
0001 SUBROUTINE LVSUM
0002 IMPLICIT INTEGER(A-Z)
0003 COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
1      HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
2      DIRCNT,OUTLOC,OUTDIR(256),RWBUP(1)
0004 COMMON /LVWASK/ MWRITE,NOTUST,NEWCON,FLGMSK,MASKSF,MASKPF
0005 COMMON /LVCRNT/ REGASP,CTRI PT,CTRI 1,LEASTV,NTFREE,FREE,DREGSP,
1      MSA,PAGLOC,CURENT
0006 COMMON /LVBUFR/ PAGSIZE,NWCHAN,OI CHAN,CWPANT,PAGHDR,BUFSZ,DIRSZ,
1      INCORE,HDRSZ,MSADIR,SUPSZ,BLK SZ,DIRBLK,PAGBD4
0007 COMMON /LVBDVI/ THMSA,REGAS,PAGENO,CONTNO,INSPCL,
1      USECT,HDRFLG,READVI,OLDNTH,DNODEH,NROW,DROWB
0008 COMMON /LVVTR1/ NOISPC(1)
1      /LVVTR2/ LSTSPC(1)
2      /LVVTR3/ LNKSPC(1)
3      /LVVTR4/ FIGSPC(1)
C
C THIS ROUTINE IS CALLED WHENEVER A CONTINUANT IS CREATED OR READ
C INTO MEMORY. ITS RESULTS ARE USED BY THE LVALUE ROUTINE FOR THE
C 'USAGE' PARAMETER.
D PAUSE 'IN LVSUM'
0009 USECNT = 0
0010 MBIAS = DIRSZ
C RECOMPUTE USE COUNT OF ALL INCORE CONTINUANTS
0011 DO 10 I = 1,INCORE
0012 USECNT = USECNT + LNKSPC(MBIAS + USECT)
0013 MBIAS = MBIAS + PAGHDR
0014 10 CONTINUE
0015 IF(USECNT.EQ. 0) USECNT = 1
0017 RETURN
0018 END

```

```

C
C
0001 SUBROUTINE LVALUE
0002 IMPLICIT INTEGER(A-Z)
0003 LOGICAL*1 CURENT
0004 REAL*4 A,B,C,D,BOTOM1,BOTTOM,CORE,ORDER,TOP,TOP1,TTLUSE,USAGE,
1 USAGE1,VALUE,VALUE1,WRITE,SPACE,CAPACY
0005 COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
1 HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
2 DIRCNT,OUTLOC,OUTDIR(256),RWBUFF(1)
0006 COMMON /LVSWIT/ SETUP,SGLBK,NXTRAN,IN1STR,IN2STR,FP1STR,FP2STR,
1 DL1STR,DL2STR,IN2TMP,FP2TMP,DL2TMP,DUMPF1,
2 FINDFI,DLETFI,NSRTFI
0007 COMMON /VIMASK/ MWRITE,NOTUSD,NEWCON,FIGMSK,MASKSF,MASKPF
0008 COMMON /VCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
1 MSA,PAGLOC,CURENT
0009 COMMON /VBUFR/ PAGSIZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSIZE,DIRSZ,
1 INCORE,HDRSZ,MSADIR,SUFSZ,BLKSIZE,DIRBLK,PAGBD4
0010 COMMON /VHVDVI/ THMSA,REGAS,PAGENO,CONTNO,INSDEL,
1 USECT,HDRFIG,READVI,OLDNDH,DNODEH,NROWH,DROWH
0011 COMMON /VPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
1 LUN
0012 COMMON /VVRUN/ RUNTYP,CORE
0013 COMMON /VVTR1/ NOTSPC(1)
1 /VVTR2/ LSTSPC(1)
2 /VVTR3/ LNKSPC(1)
3 /VVTR4/ FLGSPC(1)
0014 DATA A,B,C,D/15.0,20.0,15.0,50.0/
C
C THIS ROUTINE WILL DETERMINE WHICH CONTINUANT IS LEAST NEEDED IN
C CORE. THE ALGORITHM USED IS A MODIFICATION OF THE INTERACTIVE DATA
C MANAGER OPTIMIZATION ALGORITHM WRITTEN BY MEL HAAS, CODE 1833.
C THE VALUES RANGE FROM 0 (LEAST NEEDED CONTINUANT) TO 100 (MOST
C USEFUL CONTINUANT).
D PAUSE 'IN LVALUE'
0015 JBIAS = DIRSZ
0016 MBIAS = JBIAS
0017 IF(INCORE.EQ.1) GO TO 20
0018 LEASTV = NTFREE
0019 VALUE = 100000.0
0020 DO 10 I = 1,INCORE
0021 IF((FIGSPC(JBIAS + HDRFIG) .AND. NOTUSD) .NE. 0) GO TO 9
0022 PAUSE 'IN LVALUE LOOP'
D
C
C CALCULATE ORDER VALUE
0024 INPOS = FIGSPC(JBIAS + READVI)
0025 TOP = READCT-INPOS
0026 BOTTOM = READCT
0027 ORDER = 1.0 - (TOP/BOTTOM)
C
C CALCULATE WRITE VALUE
0028 WRITE = 0.0
0029 IF((FIGSPC(JBIAS + HDRFIG) .AND. MWRITE) .NE. 0) WRITE = CORE
C
C CALCULATE USAGE VALUE
0031 TTLUSE = USECNT

```

```

0032          USAGE1 = LNKSPC(JBIAS + USECT)
0033          USAGE = USAGE1/TTLUSE
C
C CALCULATE SPACE VALUE
0034          TOP1 = PAGSIZE - LNKSPC(JBIAS + INSPCL)
0035          BOTOM1 = PAGSIZE
0036          CAPACY = TOP1/BOTOM1
0037          GO TO (1,2,3), RUNTYP
C
C CREATION TYPE RUN
0038          1 SPACE = -4.0*CAPACY*(CAPACY-1.0)
0039          GO TO 5
C
C PRODUCTION TYPE RUN
0040          2 SPACE = 1.0
0041          IF (CAPACY .LT. .125) SPACE = 8.0*CAPACY
0043          IF (CAPACY .GE. .750) SPACE = 0.0
0045          IF ((CAPACY .GT. .375) .AND. (CAPACY .LT. .75))
0047          1 SPACE = 2.0-(8.0/3.0)*CAPACY
          GO TO 5
C
C QUERY TYPE RUN
0048          3 SPACE = 1.0-CAPACY
C
C *** CALCULATE THE IN-CORE VALUE OF THIS CONTINUANT
C
0049          5 VALUE1 = A*ORDER+B*USAGE+C*SPACE+D*WRITE
D          TYPE 111,ORDER,USAGE,SPACE,WRITE,VALUE1,VALUE
D111      FORMAT(1X,6(F10.3,2X))
0050          IF (VALUE1 .GE. VALUE) GO TO 9
0052          LEASTV = FREE
0053          VALUE = VALUE1
0054          MBIAS = JBIAS
0055          9 JBIAS = JBIAS+PAGHDR
0056          10 CONTINUE
C
C UPDATE 'LEAST VALUED' REGISTER LSTVPG UNLESS NONE WAS FOUND
0057          IF (LEASTV .EQ. FREE) GO TO 20
0059          TYPE 15
0060          15 FORMAT(/, ' *** ERROR IN LVALUE *** ',/)
C
C PREVENT POSSIBLE RECURSION
0061          IF (DUMPF1 .EQ. TRUE) STOP
0063          IERR = 0
0064          ERRNUM = 28
0065          MODE = BCD
0066          PAGES = -1
0067          DUMP = 0
0068          CALL LVDUMP(DUMP)
0069          STOP
C
0070          20 LSTVPG(1) = LSTSPC(MBIAS + PAGENO)
0071          LSTVPG(2) = LSTSPC(MBIAS + CONTNO)
0072          LSTVPG(3) = NODSPC(MBIAS + THSMSA)
0073          LSTVPG(4) = MBIAS
C SET LEASTV FOR INCORF = 1
0074          LEASTV = FREE
0075          RETURN
0076          END

```

```

C
C
0001 SUBROUTINE LVRPLC
0002 IMPLICIT INTEGER(A-Z)
0003 LOGICAL*1 CURENT, REORG, FULL, RPLACE
0004 COMMON /IARGS/ IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, ITESTR,
1      INCLUD, INDXON, IVALS(10), ITYP1(10), SRCSUF,
2      LNKSUF, SNKSUF, INSTYP
0005 COMMON /IVREGS/ CURPAG(4), REOPAG(4), LSTVPG(4), MSARET,
1      HREOPG, NXTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2      DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0006 COMMON /IVCRNT/ REGASP, CTRIPT, CTRL1, LEASTV, NTFREE, FREE, DREGSP,
1      MSA, PAGLOC, CURENT
0007 COMMON /IVBUF/ PAGSIZE, NWCBAN, OLCBAN, CMPAND, PAGHDR, BUFSIZE, DIRSZ,
1      INCORE, HDRSZ, MSADIR, SUPSZ, BIKSZ, DIRBIK, PAGHD4
0008 COMMON /IVPEL1/ NUMRET
0009 COMMON /IVINS1/ REORG, FULL, RPLACE
0010 COMMON /IVVTR1/ NONSPC(1)
1      /IVVTR2/ LSTSPC(1)
2      /IVVTR3/ LNKSPC(1)
3      /IVVTR4/ FIGSPC(1)
C
C THIS ROUTINE UPDATES THE DIRECTORY BY DELETING LSTVPG (THE LEAST
C VALUED BLOCK IN CORE) FROM IT, AND THEN INSERTING CURPAG (THE NEW
C CURRENT PAGE) INTO IT.
C
C*** SAVE SYSTEM VARIABLES
C
D   PAUSE 'IN LVRPLC'
0011 CALL LVSTAC
C
C*** DELETE OLD PAGE, CONTINUANT, LSTVPG, FROM DIRECTORY
C
0012 CTRIPT = -HDRSZ
0013 CTRL1 = 0
0014 PAGSIZE = DIRSZ
0015 IF(LSTVPG(1) .EQ. 0) GO TO 5
0017 IARG = LSTVPG(2) + 1
0018 IFUNC = LSTVPG(1)
0019 SRCSUF = IARG
0020 LNKSUF = IFUNC
0021 CALL LVFIND
0022 ITYP = 1
0023 IPOS = 1
0024 INDXON = 0
0025 REORG = .FALSE.
0026 NUMRET = 0
0027 TEMP = REGASP
0028 REGASP = DREGSP
0029 CALL LVPLET
C
C*** PLACE NEW PAGE-CONTINUANT INTO DIRECTORY
C
5   IARG = CURPAG(2) + 1
0031 IFUNC = CURPAG(1)
0032 SRCSUF = IARG

```

```

0033     LNKSUF = IFUNC
0034     CALL LVPIND
0035     INSTYP = 1
0036     NVAL = 1
0037     INDXON = 0
0038     ITYP1(1) = 1
0039     IVALS(1) = CURPAG(4)
0040     SNKSUF = CURPAG(4)
0041     CALL LVNSRT
      C
C*** RESTORE SYSTEM VARIABLES
      C
0042     DREGSP = REGASP
0043     REGASP = TEMP
0044     CALL LVPOP
0045     CTRIPT = CURPAG(4)
0046     CTRI1 = CTRIPT + HDRSZ
      C
C PAGE-BLOCK HAS BEEN FILLED AND IS NO LONGER AVAILABLE
0047     LEASTV = NTFREE
0048     RETURN
0049     END

```

```

C
C
0001 SUBROUTINE LVSTAC
0002 IMPLICIT INTEGER(A-Z)
0003 DIMENSION STORE(1),STOR(6)
0004 COMMON /IVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,
1 INCLUD,INDXON,IVAL5(10),ITYP1(10),SRCSUF,
2 LNKSUF,SNKSUF,INSTYP
0005 COMMON /IVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
1 HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
2 DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
0006 COMMON /IVBUFR/ PAGSIZE,NCHAN,OCHAN,CMPAND,PAGHDR,BUFSIZE,DIRSZF,
1 INCORE,HDRSZF,MSADIR,SUFSZE,BKKSZE,DIRBK,PAGHD4
0007 COMMON /IVPRAM/ BUFILOC,LENGTH,IERR,ERRNUM,BINARY,BCT,MODE,PAGES,
1 LUN
0008 COMMON /IVADDR/ IADD,THIS,LSTHED,LOC,LAST,LASTLC
0009 COMMON /IVSTAK/ CURLEV,NUMVAR,STACK(140)
0010 EQUIVALENCE (IFUNC,STORE(1))
0011 EQUIVALENCE (IADD,STOR(1))
0012 DATA NUMVAR,MAXLEV,CURLEV/34,3,0/

C
C THIS ROUTINE SAVES UP TO 3 SETS OF /IVARGS/ VARIABLES AND REQUEST REGISTERS
D
0013 PAUSE 'IN LVSTAC'
0014 CURLEV = CURLEV + 1
0015 IF(CURLEV.GT. MAXLEV) GO TO 99
0016 ISTLOC = (CURLEV-1)* (NUMVAR + 11) + 1
0017 DO 10 I = 1,NUMVAR
0018 STACK(I+ISTLOC) = STORE(I)
0019 10 CONTINUE
0020 DO 20 I = 1,4
0021 STACK(NUMVAR+ISTLOC+I) = REQPAG(I)
0022 20 CONTINUE
0023 DO 30 I = 1,6
0024 STACK(NUMVAR+ISTLOC+I+4) = STOR(I)
0025 30 CONTINUE
0026 STACK(NUMVAR+ISTLOC+11) = PAGSIZE
0027 RETURN

C
C FAILURE - ATTEMPT TO STACK TOO MANY SETS OF VARIABLES
99 ERRNUM = 21
0029 DUMP = 1
0030 CALL LVERR(DUMP)
0031 RETURN
0032 END

```

```

C
C
0001 SUBROUTINE LVPOP
0002 IMPLICIT INTEGER(A-Z)
0003 DIMENSION STOR(1),STOR(6)
0004 COMMON /IVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,
1 INCIUD,INDXON,IVAL5(10),ITYP1(10),SRCSUF,
2 LNKSUF,SNKSUF,INSTYP
0005 COMMON /VREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
1 HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
2 DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
0006 COMMON /VBUF/ PAGSZE,NWCHAN,OICHAN,CNPAND,PAGHDR,BUFSZE,DIRSZE,
1 INCORE,HDRSZE,MSADIR,SUFSZE,BKSZE,DIRBK,PAGHD4
0007 COMMON /VPRAM/ BUFLC,LENGTH,IERR,ERRNUM,BINARY,BCD,NODE,PAGES,
1 LUN
0008 COMMON /VADDR/ IADD,THIS,LSTHED,LOC,LAST,LASTLC
0009 COMMON /VSTAK/ CURLEV,NUNVAR,STACK(1)
0010 EQUIVALENCE (IFUNC,STOR(1))
0011 EQUIVALENCE (IADD,STOR(1))

C
C THIS ROUTINE RETURNS UP TO 3 SETS OF /IVARGS/ VARIABLES
D
0012 CURLEV = CURLEV - 1
0013 IF(CURLEV .LT. 0) GO TO 99
0015 ISTLOC = (CURLEV)*(NUNVAR+1)+1
0016 DO 10 I = 1,NUNVAR
0017 STORE(I) = STACK(I+ISTLOC)
0018 CONTINUE
0019 DO 20 I = 1,4
0020 REQPG(I) = STACK(I+ISTLOC+NUNVAR)
0021 CONTINUE
0022 DO 30 I = 1,6
0023 STOR(I) = STACK(I+ISTLOC+NUNVAR+4)
0024 CONTINUE
0025 PAGSZE = STACK(NUNVAR+ISTLOC+11)
0026 RETURN

C
C FAILURE - ATTEMPT TO RETURN TOO MANY SETS OF VARIABLES
99
0027 ERRNUM = 22
0028 DUMP = 1
0029 CALL LVERR(DUMP)
0030 RETURN
0031 END

```

```

C
C
C
0001 SUBROUTINE LVREOR (REQCON)
0002 IMPLICIT INTEGER (A-Z)
0003 LOGICAL*1 SNGIBK, SETUP, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR, DL1STR,
1 DL2STR, DUMPF1, CURENT, FINDF1, DLETF1, NSRTF1, FD1TMP,
2 DL2TMP, IN2TMP, FD2TMP, INSIDE, FULL, REORG, LSTCON, RPLACE,
3 FINISH
0004 COMMON /IVARGS/ IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, ITESSTR,
1 INCLUD, INDXON, IVALS(10), ITYP1(10), SRCSTF,
2 LNKSUF, SNKSUF, INSTYP
0005 COMMON /IVRFGS/ CURPAG(4), REQPG(4), LSTVPG(4), MSARET,
1 HRFQPG, NXTMSA, HACTPG(2), RFADCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0006 COMMON /IVMASK/ MWRITE, NOTUSD, NEWCON, FIGMSK, MASKSF, MASKPF
0007 COMMON /IVFLAG/ FI0MSK, FI1MSK, FI2MSK, FI3MSK, FI4MSK, FI5MSK, FI667,
1 FLAG8, FLAG9, FLAG10, FLAG11, FLAG12, FLAG13, FLAG14,
2 FLAG15
0008 COMMON /IVCRNT/ REGASP, CTRIPT, CTRI1, LEASTV, NTFREE, FREE, DREGSP,
1 MSA, PAGLOC, CURENT
0009 COMMON /IVBUFR/ PAGSZF, NWCHAN, OICHAN, CMPAND, PAGHDR, BUFSZF, DIRSZF,
1 INCORE, HDRSZF, MSADIR, SUFSZF, BIKSZF, DIRBIK, PAGHD4
0010 COMMON /IVHDLV/ THSMSA, REGAS, PAGENO, CONTNO, INSDFL,
1 USEPCT, HDRFIG, READVI, OINDNH, DNOEH, NROWH, DROWH
0011 COMMON /IVSWIT/ SETUP, SNGIBK, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
1 DL1STR, DL2STR, IN2TMP, FD2TMP, DL2TMP, DUMPF1,
2 FINDF1, DLETF1, NSRTF1
0012 COMMON /IVPRAM/ BUFILOC, LENGTH, IERR, ERRNUM, BINARY, BCD, MODE, PAGES,
1 LUN
0013 COMMON /IVADDR/ IADD, THIS, LSTHED, LOC, LAST, LASTLC
0014 COMMON /IVEND/ COUNT, ABSPOS, LSTCON
0015 COMMON /IVINS1/ REORG, FULL, RPLACE
0016 COMMON /IVVTR1/ NODSPC(1)
1 /IVVTR2/ LSTSPC(1)
2 /IVVTR3/ LNKSPC(1)
3 /IVVTR4/ FIGSPC(1)
C
D PAUSE 'IN LVREOR'
C
C REORG MOVES THE LIST FROM ITS PRESENT LOCATION (CONTINuant = REQPG(2))
C TO CONTINuant REQCON.
0017 ERRNUM = 62
0018 DUMP = 0
0019 IF (REQCON .EQ. CURPAG(2)) CALL LVERR (DUMP)
C
C WAS THE ORIGINAL LIST PLACED ON CURPAG(2) AS A RESULT OF A SPECIFIC
C REQUEST ? IF SO, ERROR.
0021 IF (FIGSPC(CTRI1+THIS) .AND. FLAG12) .EQ. 0 GO TO 20
C
C TYPE OUT ERROR, BUT PROCEED ANYWAY
0023 TYPE 10, SRCSTF, LNKSUF, IVALS(1), REQPG(1), REQCON, CURPAG(2)
0024 10 FORMAT('... ** ERROR **... THE FOLLOWING TRIPLE ', 3(06.2X),
1 ' OF PAGE ', 13, ' WILL BE PLACED ON CONTINuant ', 13, '/',
2 ' ORIGINAL LIST WAS FOUND ON CONTINuant ', 13)
C
0025 20 OLDCON = CURPAG(2)

```

```

0026      OI DCPT = CTRIPT
0027      OI DCP1 = CTRI1
      C
0028      C SPECIAL HANDLING IF BUFFER HOLDS ONLY ONE CONTINUANT
      IF (INCORE .EQ. 1) GO TO 30
      C
0030      C KEEP CONTINUANT WITH OLD LIST IN CORE
      FIGSPC(CTRIPT+HDRFIG) = FIGSPC(CTRIPT+HDRFIG) .OR. NOTUSD
      C
0031      C LOCATE NEW CONTINUANT
      REOPAG(2) = REQCON
0032      CALL LVEXCH
0033      NEWCPT = CTRIPT
0034      NEWCP1 = CTRI1
      C
0035      C SAVE THE VALUE TO BE INSERTED
      CALL LVSTAC
      C
0036      IF (LSTHED .GT. 0) GO TO 50
      C SVI
0038      KVAL = 1
0039      IVALS(1) = LSTSPC(OI DCP1+THIS)
0040      ITYP1(1) = FIGSPC(OI DCP1+THIS) .AND. FIG67
0041      IF (INCORE .GT. 1) GO TO 45
0043      REOPAG(2) = REQCON
0044      CALL LVEXCH
0045      45 CALL LVFIND
0046      CALL LVNSRT
0047      NOPSPC(CTRIPT+REGAS) = REGASP
0048      FIGSPC(CTRIPT+HDRFIG) = FIGSPC(CTRIPT+HDRFIG) .OR. MWRITE
0049      GO TO 70
      C MVI
0050      50 OI DLOC = LSTHED
0051      KVAL = 0
0052      MVAL = 0
0053      IF (INCORE .EQ. 1) GO TO 63
0055      60 KVAL = KVAL + 1
0056      IVALS(1) = NOPSPC(OI DCP1+OI DLOC)
0057      ITYP1(1) = FIGSPC(OI DCP1+OI DLOC) .AND. FIG67
0058      CALL LVFIND
0059      CALL LVNSRT
0060      OI DLOC = LSTSPC(OI DCP1+OI DLOC)
0061      IF (FIGSPC(OI DCP1+OI DLOC) .AND. FIOMSK) .EQ. 0) GO TO 60
0063      GO TO 70
      C
0064      C FOR MVI (INCORE = 1) SWAP OLD AND NEW CONTINUANTS IN AND OUT WHILE
0065      C RE-INSERTING UP TO TEN VALUES AT A TIME
      63 FINISH = .FALSE.
0066      65 REOPAG(2) = OI DCON
0067      CALL LVEXCH
0068      66 MVAL = MVAL + 1
0069      KVAL = KVAL + 1
0070      IVALS(KVAL) = NOPSPC(OI DCP1+OI DLOC)
0071      ITYP1(KVAL) = FIGSPC(OI DCP1+OI DLOC) .AND. FIG67
0072      OI DLOC = LSTSPC(OI DCP1+OI DLOC)
0073      IF (FIGSPC(OI DCP1+OI DLOC) .AND. FIOMSK) NE. 0) FINISH = .TRUE.
0074      IF KVAL GE 10) GO TO 67

```

```

0076      IF(.NOT. FINISH) GO TO 66
0078 67      REOPAG(2) = REQCON
0079      CALL LVEXCH
0080      CALL LVFIND
0081      NVAL = KVAL
0082      KVAL = 0
0083      CALL LVNSRT
0084      NODSPC(CTRIPT+REGAS) = REGASP
0085      FIGSPC(CTRIPT+HDRFIG)=FIGSPC(CTRIPT+HDRFIG) .OR. MWRITE
0086      IF(.NOT. FINISH) GO TO 65
0088      KVAL = MVAL
0089      REOPAG(2) = OI DCON
0090      CALL LVEXCH
      C
      C DELETE OLD LIST
0091 70      INDXON = 0
0092      CTRIPT = OI DCPT
0093      CTRI 1 = OI DCP1
0094      IF(INCORF .GT. 1) GO TO 75
0096      REOPAG(2) = OI DCON
0097      CALL LVEXCH
0098 75      CALL LVFIND
0099      CALL LVPLET
      C
      C UPDATE HEADER
      C OLD CONTINUANT HAS BEEN MODIFIED
0100      FIGSPC(CTRIPT+HDRFIG)=FIGSPC(CTRIPT+HDRFIG) .AND. .NOT. NOTUSD
      1 .OR. MWRITE
      C
      C RESET CONTINUANT USAGE RATIO
0101      LNKSPC(OI DCPT+INSDCL) = LNKSPC(OI DCPT+INSDCL) - KVAL
      C
      C INSERT NEW VALUE
0102      CALL LVPOP
0103      REOPAG(2) = REQCON
0104      IF(INCORF .EQ. 1) CALL LVEXCH
0106      CTRIPT = NEWCPT
0107      CTRI 1 = NEWCP1
0108      CALL LVFIND
0109      CALL LVNSRT
      C
      C UPDATE HEADER
      C RESET CONTINUANT USAGE RATIO
0110      LNKSPC(NEWCPT+INSDCL) = LNKSPC(NEWCPT+INSDCL) + KVAL
0111      RETURN
0112      END

```

```

C
C
C
0001 SUBROUTINE LVINCI
0002 IMPLICIT INTEGER(A-Z)
0003 LOGICAL*1 SNGIBK, SETUP, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR, DL1STR,
1 DL2STR, DUMPF1, CURENT, FINDFI, DLETF1, NSRTFI,
2 DL2TMP, IN2TMP, FD2TMP, REORG, FULL, RPLACE
0004 COMMON /IVARGS/ IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, ITESTR,
1 INCIUD, INDXON, IVALS(10), ITYP1(10), SRCSTF,
2 LNKSUF, SNKSUF, INSTYP
0005 COMMON /IVRFGS/ CURPAG(4), RFOPAG(4), LSTVPG(4), MSARET,
1 HREQPG, NXTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0006 COMMON /IVMASK/ MWRITE, NOTUSD, NEWCON, FIGMSK, MASKSF, MASKPF
0007 COMMON /IVFIAG/ FI0MSK, FI1MSK, FI2MSK, FI3MSK, FI4MSK, FI5MSK, FI67,
1 FIAG8, FIAG9, FIAG10, FIAG11, FIAG12, FIAG13, FIAG14,
2 FIAG15
0008 COMMON /IVCRNT/ REGASP, CTRIPT, CTRI1, LEASTV, NTFRFE, FREE, DREGSP,
1 MSA, PAGLOC, CURENT
0009 COMMON /IVBUFR/ PAGSZE, NWCHAN, OICHAN, CMPAND, PAGHDR, BUFSZE, DIRSZE,
1 INCORE, HDRSZE, MSADIR, SUFSZE, BIKSZE, DIRBIK, PAGHD4
0010 COMMON /IVHDVI/ THSMSA, REGAS, PAGENO, CONTNO, INSPEL,
1 USECT, HDRFIG, READVI, OI DNDH, DNODEH, NROWH, DROWH
0011 COMMON /IVSWIT/ SETUP, SNGIBK, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
1 DL1STR, DL2STR, IN2TMP, FD2TMP, DL2TMP, DUMPF1,
2 FINDFI, DLETF1, NSRTFI
0012 COMMON /IVPRAM/ BUFILOC, LENGTH, IERR, ERRNUM, BINARY, BCD, MODE, PAGES,
1 LUN
0013 COMMON /IVADDR/ IADD, THIS, LSTHED, LOC, LAST, LASTLC
0014 COMMON /IVINS1/ REORG, FULL, RPLACE
0015 COMMON /IVVTR1/ NOPSPC(1)
1 /IVVTR2/ LSTSPC(1)
2 /IVVTR3/ LNKSPC(1)
3 /IVVTR4/ FIGSPC(1)
C
C THIS ROUTINE SEARCHES THE LIST TO FIND THE VALUE IN INCIUD. IF IT
C IS FOUND, ITS POSITION WRT THE TOP OF THE LIST IS RETURNED
C
C DOES THE LIST EXIST ?
D PAUSE 'IN LVINCI'
IPOS = 1
RFOPAG(2) = -2
J = 0
CALL LVFDEX(J, J, J, J, J)
0020 IF (ITESTR .IT. 0) GO TO 31
0022 IF (IVAL EQ. INCIUD) GO TO 25
0024 IF (LSTHED .IT. 0) GO TO 31
C MVI FOUND
JVAL = IVAL
KSKIP = NSKIP
0028 NSKIP = 0
0029 INDEX = 0
0030 INDXAD = 0
0031 KFUNC = 0
0032 KARG = 0
0033 SAVCON = 0

```

```

0034      KPOS = 0
0035 10    KPOS = KPOS+1
0036      IPOS = KPOS
0037      CALL LVFDEX(INDEX, INDXAD, KFUNC, KARG, SAVCON)
0038      IF(ITESTR .IT. 0) GO TO 30
0040      IF(IVAL .NE. INCIUD) GO TO 10
      C
      C EXIT FROM LOOP, INCIUD = 1, SUCCESS
      C      INCIUD = -1, FAILURE
      C EXCEPT FOR IPOS, OUTPUT MUST APPEAR AS IF IT WAS FROM LVFIND
0042 30    IVAL = JVAL
0043      NSKIP = KSKIP
0044      LOC = LSTHED
0045 25    INCIUD = ITESTR
0046      ITESTR = 1
0047      RETURN
      C
      C SVI FAILURE RETURN
0048 31    INCIUD = -1
0049      RETURN
0050      END

```

```

C
C
0001 SUBROUTINE LVOVER
0002 IMPLICIT INTEGER(A-Z)
0003 LOGICAL*1 SNGIBK, SETUP, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR, DL1STR,
1 DL2STR, DUMPF1, CURENT, FINDFI, DLETFI, NSRTFI, FD1TMP,
2 DL2TMP, IN2TMP, FD2TMP, INSIDE, FULL, REORG, LSTCON, RPLACE,
3 FINISH
0004 COMMON /I VARGS/ IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, ITESTR,
1 INCIUD, INDXON, IVALS(10), ITYP1(10), SRCSTUF,
2 LNKSUF, SNKCUF, INSTYP
0005 COMMON /I VREGS/ CURPAG(4), REQPG(4), LSTVPG(4), MSARET,
1 HREQPG, NXTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0006 COMMON /I VMASK/ MWRITE, NOTUSD, NEWCON, FIGMSK, MASKSF, MASKPF
0007 COMMON /I VFIAG/ FI0MSK, FI1MSK, FI2MSK, FI3MSK, FI4MSK, FI5MSK, FI067,
1 FIAG8, FIAG9, FIAG10, FIAG11, FIAG12, FIAG13, FIAG14,
2 FIAG15
0008 COMMON /I VCRNT/ REGASP, CTRIP1, CTR1, LEASTV, NTFREE, FREE, DREGSP,
1 MSA, PAGLOC, CURENT
0009 COMMON /I VBUFR/ PAGSIZE, NWCHAN, OICHAN, CMPAND, PAGHDR, BUFSZ, DIRSZ,
1 INCORE, HDRSZ, MSADIR, SUFSZ, BIKSZ, DIRBIK, PAGHD4
0010 COMMON /I VHDVI/ THSMSA, REGAS, PAGENO, CONTNO, INSDI,
1 USECT, HDRFIG, READVI, OLDNDH, DNODEH, NROWH, DROWH
0011 COMMON /I VSWIT/ SETUP, SNGIBK, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
1 DL1STR, DL2STR, IN2TMP, FD2TMP, DL2TMP, DUMPF1,
2 FINDFI, DLETFI, NSRTFI
0012 COMMON /I VPRAM/ BUFILOC, LENGTH, IERR, ERRNUM, BINARY, BCT, MODE, PAGES,
1 LUN
0013 COMMON /I VADDR/ IADD, THIS, LSTHED, LOC, LAST, LASTLC
0014 COMMON /I VFND/ COUNT, ABSPOS, LSTCON
0015 COMMON /I VINS1/ REORG, FULL, RPLACE
0016 COMMON /I VVTR1/ NODSPC(1)
1 /I VVTR2/ LSTSPC(1)
2 /I VVTR3/ LNKSPC(1)
3 /I VVTR4/ FIGSPC(1)
C
C D
C D
C THIS ROUTINE HANDLES CONTINUANT OVERFLOW ON INDEXED INSERTION
C
C IF FUNCTION DOES NOT EXIST, RETURN AND PLACE ON NEXT CONTINUANT
0017 IF(ITESTR .IT. 0) RETURN
C
C PICK UP LAST VALUE ON LIST,
0019 IF(LSTHED .GT. 0) GO TO 50
C SVI
0021 NVAL = 2
0022 IVALS(2) = IVALS(1)
0023 ITYP1(2) = ITYP1(1)
0024 TMPVAL = LSTSPC(CTRI1+THIS)
0025 TMPTYP = FIGSPC(CTRI1+THIS) .AND. FI067
0026 GO TO 100
C MVI
C SAVE THE VALUE TO BE INSERTED
0027 50 CALL LVSTAC

```

```

0028      TMPVAL = NORSPC(CTRI1+1ASTLC)
0029      TMPTYP = FIGSPC(CTRI1+1ASTLC) .AND. FIG67
      C
      C DELETE LAST VALUE,
0030 100      INDXON = 1
0031      IPOS = -1
0032      CALL LVDLET
      C
      C RESET CONTINUANT USAGE RATIO
0033      LNKSPC(CTRIPT+INDEL) = LNKSPC(CTRIPT+INDEL) -1
0034      IF(LSTHED .LT. 0) GO TO 150
      C
      C INSERT ORIGINAL VALUE,
0036      CALL LVPOP
0037      CALL LVNSRT
      C
      C INSERT LAST VALUE ON NEXT CONTINUANT
0038 150      IVALS(1) = TMPVAL
0039      ITYP1(1) = TMPTYP
0040      INDXON = 0
0041      RETURN
0042      END

```

```

C
C
C
0001 SUBROUTINE LVDUMP(DUMP)
0002 IMPLICIT INTEGER(A-Z)
0003 LOGICAL*1 CURENT, SETUP, SNGIBK, NYTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
1 DL1STR, DL2STR, IN2TMP, FD2TMP, DL2TMP, DUMPF1,
2 FINDFI, DLETFI, NSRTFI
0004 COMMON /LVREGS/ CURPAG(4), REQPA(4), LSTVPG(4), MSARET,
1 HREQPG, NYTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0005 COMMON /VMASK/ MWRITE, NOTUSD, NEWCON, FIGMSK, MASKSF, MASKPF
0006 COMMON /VCRNT/ REGASP, CTRLPT, CTRL1, LEASTV, NTFREE, FREE, DREGSP,
1 MSA, PAGLOC, CURENT
0007 COMMON /VBUFR/ PAGSZE, NWCHAN, OLCNAN, CMPAND, PAGHDR, BUFSZE, DIRSZE,
1 INCORE, HDRSZF, MSADIR, SUFSZE, BLKSZE, DIRBLK, PAGBD4
0008 COMMON /VHDVI/ THSMSA, REGAS, PAGENO, CONTNO, INSDPL,
1 USECT, HDRFIG, READVI, OI DNDH, DNODEH, NROWH, DROWH
0009 COMMON /VSWIT/ SETUP, SNGIBK, NYTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
1 DL1STR, DL2STR, IN2TMP, FD2TMP, DL2TMP, DUMPF1,
2 FINDFI, DLETFI, NSRTFI
0010 COMMON /VPRAM/ BUFILOC, LENGTH, IERR, ERRNUM, BINARY, BCD, MODE, PAGES,
1 LUN
0011 COMMON /IVSEQ/ ISEOSZ, ISOPOS, LASTSO, SEQSPC(1)
0012 COMMON /IVVTR1/ NOTSPC(1)
1 /IVVTR2/ LSTSPC(1)
2 /IVVTR3/ LNKSPC(1)
3 /IVVTR4/ FIGSPC(1)
0013 EQUIVALENCE(LUN, CHAN)
C
C THIS ROUTINE HAS TWO MODES OF OPERATION:
C A) BCD DUMP - OUTPUTS SYSTEM VARIABLES AND INTERNAL STRUCTURE OF GIRS
C PAGES AS A DEBUGGING AID
C B) BINARY DUMP
C COPIES ALL (MODIFIED) IN-CORE CONTINUANTS TO THE NEW DISK FILE.
C A CALL TO LVCIOS SAVES THE SYSTEM PARAMETERS.
C
C PAGES = -1, DUMP IN-CORE PAGE-BLOCKS
C = 0, DUMP ALL PAGES W/ ALL CONTS
C = N, DUMP PAGE N
C
C IS THIS A BINARY OR BCD DUMP ?
D PAUSE 'IN LVDUMP()'
0014 IF(MODE .EQ. BINARY) GO TO 100
C
C*** BCD
C LOGICAL UNIT NUMBER SHOULD BE DEFINED IN A CALL TO ASSIGN
C WRITE HEADERS
0016 WRITE(LUN, 10)
0017 10 FORMAT(/, ' GIRS MEMORY DUMP (IN OCTAL)', '/')
0018 ERRNUM = -1
0019 DUMPF1 = .TRUE.
0020 CALL LVERR(DUMP)
0021 WRITE(LUN, 21)
0022 21 FORMAT(///)
0023 WRITE(LUN, 21)
C WRITE IN-CORE DIRECTORY

```

```

0024 WRITE(LUN,35)
0025 35 FORMAT(' IN-CORE DIRECTORY',/)
0026 WRITE(LUN,45)
0027 45 FORMAT(1X,' COUNTERS KEY(PAGE) CONTINUANT "IOC-1" LNKSPC
      1 FIGSPC',/)
0028 NBIAS = -HDRSZ
0029 NUMBK = 1
0030 CALL LVWRIT(NBIAS,NUMBK)
      C
      C TYPE OF DUMP ?
0031 IF(PAGES .GE. 0) GO TO 50
      C WRITE OUT ONLY THOSE CONTINUANTS THAT ARE IN CORE
0033 NBIAS = DIRSZ
0034 NUMBK = INCORE
0035 CALL LVWRIT(NBIAS,NUMBK)
0036 DUMPF1 = .FALSE.
0037 RETURN
      C
      C TEST TO WRITE OUT EITHER A SINGLE PAGE AND ALL OF ITS CONTINUANTS
      C OR ALL PAGES
0038 50 NPAGE = PAGES
0039 HIPAGE = HACTPG(1)
0040 IF(HREQPG .GT. HIPAGE) HIPAGE = HREQPG
0042 IF(PAGES.NF.0) GO TO 60
0044 55 PAGES = PAGES+1
0045 60 REQPG(1) = PAGES
0046 REQPG(2) = -1
0047 65 REQPG(2) = REQPG(2)+1
      C ATTEMPT TO LOCATE OR BRING IN PAGE, CONT
0048 CALL LVEXCH
      C IS REQ(P,C) IN CORE ?
0049 IF(PAGLOC .GT. 0) GO TO 68
      C ANY MORE CONTINUANTS TO THIS PAGE
0051 IF(MSARET .IE. 0) GO TO 70
      C WRITE OUT PAGE
0053 68 NBIAS = CTRIPT
0054 NUMBK = 1
0055 CALL LVWRIT(NBIAS,NUMBK)
0056 GO TO 65
      C WRITE SINGLE PAGE ONLY ?
0057 70 IF(NPAGE .NE. 0) GO TO 80
      C IS THIS THE HIGHEST ACTIVE PAGE ?
0059 IF(PAGES .IT. HIPAGE) GO TO 55
0061 80 DUMPF1 = .FALSE.
0062 RETURN
      C
      C ***BINARY WRITE
      C
0063 100 CTRIPT = DIRSZ
0064 CTRI1 = CTRIPT + HDRSZ
0065 DO 150 I = 1, INCORE
      C COPY TO DISK IF CONTINUANT HAS BEEN MODIFIED IN CORE
0066 IF((FIGSPC(CTRIPT + HDRFIG) .AND. MWRITE) .EQ. 0) GO TO 145
0068 FIGSPC(CTRIPT + HDRFIG) = 0
0069 LENGTH = PAGHDR
0070 BUFLOC = CTRIPT + 1
0071 MSA = NOPSPC(CTRIPT + THMSA)
0072 CALL LVPAGW
0073 145 CTRIPT = CTRIPT + PAGHDR
0074 CTRI1 = CTRIPT + HDRSZ
0075 150 CONTINUE
0076 CALL LVCIOS
0077 RETURN
0078 END

```

```

C
C
0001 SUBROUTINE LVWRIT(NBIAS,NUMBK)
0002 IMPLICIT INTEGER(A-Z)
0003 COMMON /1VCRNT/ REGASP,CTRIPT,CTRI1,LEASTV,NTFREE,FREE,DREGSP,
1 MSA,PAGLOC,CURRENT
0004 COMMON /1VBUFR/ PAGSIZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
1 INCORE,HDRSZE,MSADIR,SUFSZE,BIKSZE,DIRBIK,PAGHD4
0005 COMMON /1VFLAG/ FL0MSK,FL1MSK,FL2MSK,FL3MSK,FL4MSK,FL5MSK,FL667,
1 FLAG8,FLAG9,FLAG10,FLAG11,FLAG12,FLAG13,FLAG14,
2 FLAG15
0006 COMMON /1VBDVI/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
1 USECT,HDRFIG,READVI,OI DNDH,DNOPEH,NROWH,DROWH
0007 COMMON /1VPRAM/ BUFI OC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
1 LUN
0008 COMMON /1VMASK/ MWRITE,NOTUSD,NEWCON,FIGMSK,MASKSF,MASKPF
0009 COMMON /1VVTR1/ NODSPC(1)
1 /1VVTR2/ LSTSPC(1)
2 /1VVTR3/ LNKSPC(1)
3 /1VVTR4/ FIGSPC(1)

C
C THIS ROUTINE PERFORMS A BCD WRITE OF NUMBK PAGE-BLOCKS BEGINNING AT
C GIRS BUFFER LOCATION NBIAS
C
C FLAG CONTINUANT AS USED
0010 FIGSPC(CTRIPT+HDRFIG) = FIGSPC(CTRIPT+HDRFIG) .AND. .NOT. NOTUSD
D PAUSE 'IN LVWRIT'
0011 DO 100 M = 1,NUMBK
0012 ISTLOC = NBIAS+HDRSZE+1
0013 LAST = ISTLOC+PAGSZE-1
0014 IF(NBIAS .GT. 0) GO TO 10
0015 LAST = DIRSZE
0016 GO TO 30
C EXTRACT HEADER VALUES
0018 10 MSA = NODSPC(NBIAS + THSMSA)
0019 REGASP = NODSPC(NBIAS + REGAS)
0020 PAGE = LSTSPC(NBIAS + PAGENO)
0021 CONT = LSTSPC(NBIAS + CONTNO)
0022 INSPLT = LNKSPC(NBIAS + INSDEL)
0023 USEAGE = LNKSPC(NBIAS + USECT)
0024 FLAGS = FIGSPC(NBIAS + HDRFIG)
0025 RDVAL = FIGSPC(NBIAS + READVI)
0026 WRITE(LUN,1)
0027 WRITE(LUN,2) PAGE,CONT
C WRITE HEADER
0028 WRITE(LUN,3)
0029 WRITE(LUN,4) MSA,REGASP,INSPLT,USEAGE,FLAGS,RDVAL
0030 WRITE(LUN,5)
0031 30 COUNT = 0
0032 DO 50 BUFCNT = ISTLOC, LAST
0033 COUNT = COUNT + 1
0034 M1 = NODSPC(BUFCNT)
0035 M2 = LSTSPC(BUFCNT)
0036 M3 = LNKSPC(BUFCNT)
0037 M4 = FIGSPC(BUFCNT)
0038 IF(NBIAS .GT. 0) GO TO 45

```

```

C IN-CORE DIRECTORY
0040      CONTIN = -1
0041      IF((FLGSPC(BUFCNT) .AND. FLIMSK) .EQ. 0) GO TO 40
0043      CONTIN = BUFCNT - M1 - 1
0044      IF(CONTIN .LT. 0) CONTIN = CONTIN + DIRSZ
0046      GO TO 47
0047 40     M1 = 0
0048      M2 = 0
0049 47     WRITE(LUN,7) BUFCNT,COUNT,M1,CONTIN,M2,M3,M4,COUNT
0050      GO TO 50
0051 45     WRITE(LUN,6) BUFCNT,COUNT,M1,M2,M3,M4,COUNT
0052 50     CONTINUE
0053      NBIAS = NBIAS+PAGEHDR
0054 100    CONTINUE
0055 1      FORMAT(///,10X,'PAGE',3X,'CONTINUANT',/)
0056 2      FORMAT(8X,16,3X,16,/)
0057 3      FORMAT(1X,'MSA' REGASP (INSERTIONS-DELETIONS) USAGE
1        'FLAGS' READ COUNT')
0058 4      FORMAT(1X,15,2X,15,10X,15,14X,15,3X,03,3X,15,/)
0059 5      FORMAT(1X,'WRKSPC' NOPSPC LSTSPC LNKSPC FLGSPC
1        'OCTAL COUNTER',/)
0060 6      FORMAT(1X,16,2X,16,2X,06,2X,06,2X,06,2X,06,2X,06)
0061 7      FORMAT(1X,16,2X,16,2X,06,6X,13,5X,06,2X,06,2X,06,2X,06)
0062      RETURN
0063      END

```

```

C
C
0001 SUBROUTINE LVCIOS
0002 IMPLICIT INTEGER(A-Z)
0003 LOGICAL*1 CURENT
0004 COMMON /IVREGS/ CURPAG(4), REOPAG(4), LSTVPG(4), MSARET,
1 HREQPG, NXTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0005 COMMON /IVMASK/ MWRITE, NOTUSD, NEWCON, FLGMSK, MASKSF, MASKPF
0006 COMMON /IVRAND/ PRIME, SEED, NROW, DNODE, DROW, OLDNOD, LISTSZ,
1 GRNTBI(256)
0007 COMMON /IVCRNT/ REGASP, CTRIPT, CTRI1, LEASTV, NTFREE, FREE, DREGSP,
1 MSA, PAGLOC, CURENT
0008 COMMON /IVBUFR/ PAGSZ, NWCHAN, OLCHAN, CMPAND, PAGHDR, BUFSZ, DIRSZ,
1 INCORE, HDRSZ, MSADIR, SUFSZ, BLKSZ, DIRBIK, PAGHD4
0009 COMMON /IVHDI/ THMSA, REGAS, PAGENO, CONTNO, INSEL,
1 USECT, HDRFIG, READVI, OLDNDH, DNODEH, NROWH, DROWH
0010 COMMON /IVSEQ/ ISEQSZ, ISOPOS, LASTSO, SEQSPC(1)
0011 COMMON /IVPRAM/ BUFILOC, LENGTH, IERR, ERRNUM, BINARY, BCD, MODE, PAGES,
1 LUN
0012 COMMON /IVUSER/ USFR(224)
D PAUSE 'IN LVCIOS'
C
C SAVE SYSTEM VARIABLES ON FIRST BLOCK OF DISK
0013 RWBUF(1) = REGASP
0014 RWBUF(2) = NXTMSA
0015 RWBUF(3) = PAGSZ
0016 RWBUF(4) = PAGHDR
0017 RWBUF(5) = BUFSZ
0018 RWBUF(6) = DIRSZ
0019 RWBUF(7) = DREGSP
0020 RWBUF(8) = INCORE
0021 RWBUF(9) = HDRSZ
0022 RWBUF(10) = HREQPG
0023 RWBUF(11) = HACTPG(1)
0024 RWBUF(12) = HACTPG(2)
0025 RWBUF(13) = READCT
0026 RWBUF(14) = BLKSZ
0027 RWBUF(15) = SUFSZ
0028 RWBUF(16) = DIRBIK
0029 RWBUF(17) = PRIME
0030 RWBUF(18) = SEED
0031 RWBUF(19) = LISTSZ
0032 RWBUF(20) = ISEQSZ
0033 DO 10 I = 1, 4
0034 RWBUF(20+I) = CURPAG(I)
0035 RWBUF(24+I) = LSTVPG(I)
0036 CONTINUE
10
C
C USER WILL HAVE ACCESS TO WORDS 29 THRU 256 OF THE FIRST BLOCK TO STORE
C VARIABLES IF A PERMANENT FILE IS TO BE CREATED.
0037 DO 15 I = 29, 256
0038 J = I - 28
0039 RWBUF(I) = USFR(J)
0040 LENGTH = 256
0041 MSA = 0

```

```

0042      ERRNUM = 3
0043      IERR = IWRITW(LENGTH,RWBUF(1),MSA,NWCHAN)
0044      DUMP = 0
0045      IF(IERR.IT.0) CALL LVERR(DUMP)
      C
      C SAVE GRN VARIABLES
      C
0047      LENGTH = 256
0048      MSA = 1
0049      ERRNUM = 4
0050      IERR = IWRITW(LENGTH,GRNTBL(1),MSA,NWCHAN)
0051      DUMP = 0
0052      IF(IERR.IT.0) CALL LVERR(DUMP)
      C
      C SAVE INCORE DIRECTORY BEGINNING AT MSA = 2
      C
0054      MSA = 2
0055      BUFLOC = 1
0056      LENGTH=DIRSZ
0057      ERRNUM = 5
0058      CALL LVPAGW
      C
      C SAVE CURRENT OUTCORE DIRECTORY
0059      CALL LVDRWR
      C
      C CLOSE CHANNEL
0060      CALL CIOSEC(NWCHAN)
0061      RETURN
0062      END

```

```

C
C
0001 SUBROUTINE LVPAGR(CHAN)
0002 IMPLICIT INTEGER(A-Z)
0003 LOGICAL*1 CURENT
0004 COMMON /1VREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
1 HREQPG,NXTMSA,RACTPG(2),READCT,USDCNT,DIRPAG,
2 DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
0005 COMMON /1VCRNT/ REGASP,CTRIPT,CTRI1,LEASTV,NTFREE,FREE,DRFGSP,
1 MSA,PAGLOC,CURENT
0006 COMMON /1VBUFFR/ PAGSIZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
1 INCORE,HDRSZE,MSADIR,SUFSZE,BIKSZE,DIRBIK,PAGHD4
0007 COMMON /1VPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
1 LUN
0008 COMMON /1VMASK/ MWRITE,NOTUSD,NEWCON,FIGMSK,MASKSF,MASKPF
0009 COMMON /1VHDL/ THMSA,REGAS,PAGENO,CONTNO,INDEL,
1 USECT,HDRFIG,READVI,OIDNPH,DNODEH,NROWH,DROWH
0010 COMMON /1VVTR1/ NOTSPC(1)
1 /1VVTR2/ LSTSPC(1)
2 /1VVTR3/ LNKSPC(1)
3 /1VVTR4/ FIGSPC(1)
C
C THIS ROUTINE READS DATA FROM DISK INTO RWBUF AND PLACES IT INTO WRKSPC
D
0011 PAUSE 'IN LVPAGR()'
0012 NEWLEN = 4*LENGTH
0013 IERR = IREADW(NEWLEN,RWBUF(1),MSA,CHAN)
0014 ERRNUM = 8
0015 DUMP = 0
0016 IF(IERR.I.T.0) CALL LVERR(DUMP)
0017 ISTLOC = BUFLOC - 1
0018 DO 10 I = 1,LENGTH
0019 NOTSPC(ISTLOC + 1) = RWBUF(1)
0020 LSTSPC(ISTLOC + 1) = RWBUF(LENGTH + 1)
0021 LNKSPC(ISTLOC + 1) = RWBUF(2*LENGTH + 1)
0022 10 FIGSPC(ISTLOC + 1) = RWBUF(3*LENGTH + 1)
0023 IF(ISTLOC .I.E. 0) RETURN
C
C IF NOT DIRECTORY, FLAG CONTINUANT AS NOT USED
0025 FIGSPC(ISTLOC+HDRFIG) = FIGSPC(ISTLOC+HDRFIG) .OR. NOTUSD
0026 RETURN
0027 END

```

```

C
C
0001 SUBROUTINE LVPAGW
0002 IMPLICIT INTEGER(A-Z)
0003 LOGICAL*1 CURENT
0004 COMMON /1VREGS/ CURPAG(4), RPOPAG(4), LSTVPG(4), MSARFT,
1 HREFPG, NYTMSA, HACTPG(2), RFADCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0005 COMMON /1VCRNT/ REGASP, CTRIPT, CTRI1, LEASTV, NTPREE, FREE, DRFGSP,
1 MSA, PAGLOC, CURENT
0006 COMMON /1VBUFF/ PAGSZE, NWCHAN, OLCUAN, CMPAND, PAGHDR, BUFSZE, DIRSZF,
1 INCORE, HDRSZF, MSADIR, SUFSZE, BIKSZE, DIRBIK, PAGHD4
0007 COMMON /1VPRAM/ BUFILOC, LENGTH, IERR, ERRNUM, BINARY, BCD, MODE, PAGES,
1 LUN
0008 COMMON /1VHVDI/ THSMAS, REGAS, PAGFNO, CONTNO, INSPDL,
1 USECT, HDRFIG, READVI, OI DNDH, DNODEH, NROWB, DROWB
0009 COMMON /1VMASK/ MWRITE, NOTUSD, NEWCON, FIGMSK, MASKSF, MASKPF
0010 COMMON /1VVTR1/ NOPSPC(1)
1 /1VVTR2/ LSTSPC(1)
2 /1VVTR3/ LNKSPC(1)
3 /1VVTR4/ FIGSPC(1)
C
C THIS ROUTINE TRANSFERS THE CONTENTS OF WRKSPC TO RWBUF TO BE WRITTEN
C OUT TO DISK
D PAUSE 'IN LVPAGW'
0011 NEWLEN = 4*LENGTH
0012 ISTLOC = BUFILOC - 1
C
C IF NOT DIRECTORY, TURN FLAGS OFF
0013 IF(ISTLOC .IE. 0) GO TO 5
0015 FIGSPC(ISTLOC+HDRFIG) = FIGSPC(ISTLOC+HDRFIG) .AND. .NOT. FIGMSK
0016 5 DO 10 I = 1, LENGTH
0017 RWBUF(I) = NOPSPC(ISTLOC + 1)
0018 RWBUF(LENGTH + 1) = LSTSPC(ISTLOC + 1)
0019 RWBUF(2*LENGTH + 1) = LNKSPC(ISTLOC + 1)
0020 10 RWBUF(3*LENGTH + 1) = FIGSPC(ISTLOC + 1)
0021 IERR = IWRITE(NEWLEN, RWBUF(1), MSA, NWCHAN)
0022 ERRNUM = 9
0023 DUMP = 0
0024 IF(IERR .IT. 0) CALL LVERR(DUMP)
0026 RETURN
0027 END

```

```

C
C
C
0001 SUBROUTINE LVDRRD(CHAN)
0002 IMPLICIT INTEGER(A-Z)
0003 COMMON /LVREGS/ CURPAG(4), ROPAG(4), LSTVPG(4), MSARET,
1 HRFPG, NYTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0004 COMMON /LVBUF/ PAGSZF, NWCHAN, OICHAN, CPAND, PAGHDR, BUFSZF, DIRSZF,
1 INCORE, HDRSZF, MSADIR, SUFSZF, BLSZF, DIRBIK, PAGHDR,
0005 COMMON /LVPRAM/ BUFOC, LENGTH, IERR, ERRNUM, BINARY, BCD, MODE, PAGES,
1 LUN
C
C THIS ROUTINE READS A SELECTED OUTCORE DIRECTORY BLOCK INTO OUTDIR
C
C
D PAUSE 'IN LVDRRD'
0006 LENGTH = 256
0007 MSA = MSADIR + DIRBIK + DIRPAG
0008 ERRNUM = 6
0009 IERR = IREADW(LENGTH, OUTDIR(1), MSA, CHAN)
0010 DUMP = 0
0011 IF (IERR.NE.0) CALL LVERR(DUMP)
0012 RETURN
0013
0014 END
C
C
C
0001 SUBROUTINE LVDRWR
0002 IMPLICIT INTEGER(A-Z)
0003 COMMON /LVREGS/ CURPAG(4), ROPAG(4), LSTVPG(4), MSARET,
1 HRFPG, NYTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0004 COMMON /LVBUF/ PAGSZF, NWCHAN, OICHAN, CPAND, PAGHDR, BUFSZF, DIRSZF,
1 INCORE, HDRSZF, MSADIR, SUFSZF, BLSZF, DIRBIK, PAGHDR,
0005 COMMON /LVPRAM/ BUFOC, LENGTH, IERR, ERRNUM, BINARY, BCD, MODE, PAGES,
1 LUN
C
C THIS ROUTINE WRITES THE OUTCORE DIRECTORY BLOCK FROM OUTDIR TO DISK
C
C
C
D PAUSE 'IN LVDRWR'
0006 LENGTH = 256
0007 MSA = MSADIR + DIRBIK + DIRPAG - 1
0008 ERRNUM = 7
0009 IERR = IWRITE(LENGTH, OUTDIR(1), MSA, NWCHAN)
0010 DUMP = 0
0011 IF (IERR.NE.0) CALL LVERR(DUMP)
0012 RETURN
0013
0014 END

```

```

00001 SUBROUTINE LVERR DUMP
00002 IMPLICIT INTEGER A-Z
00003 REAL*4 CORE
00004 LOGICAL*4 SNGI,BA,SETUP,NYTRAS,IN1STR,IN2STR,ED1STR,ED2STR,D1STR,
00005 D2STR,DUMPEI,CURRENT,IN2TMP,ED2TMP,D1TMP,D2TMP,FINDEI
00006 D1TFI,NSRTEI,FULL,REORG,RPLACF
00007 COMMON /VARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSATE,ITESTR
00008 INCLD,INDXON,IVALS(10),ITYP(10),SRCSUF
00009 LNKSTF,SNKSTF,INSTYF
00010 COMMON /VREGS/ CURPAG,4,REOPAG,4,ESTVPG,4,MSARIT,
00011 HREFPG,NYMSA,HAUTPG,2,READDT,USEFNT,DIRPAG
00012 DIRST,OUTD1,OUTD2,256,0,RMBUF,256,0
00013 COMMON /VMASK/ WRITE,NOTED,NEWCON,FLGMSK,MASKF,MASKF
00014 IVALAG,FLGMSK,FLGMSK,FLGMSK,FLGMSK,FLGMSK,FLGMSK,FLGMSK,FLGMSK,FLGMSK
00015 FLAG0,FLAG0,FLAG0,FLAG0,FLAG0,FLAG0,FLAG0,FLAG0,FLAG0,FLAG0
00016 FLAG,5
00017 COMMON /VRND/ PRIME,SEED,NROW,DNODE,DROW,OLDNODE,LISTSZ
00018 GRNTBI,256,0
00019 COMMON /VCRNT/ REGASP,CURPT,CTR1,LEASTV,STEREF,REF,DREGSP,
00020 MSA,PAGE,0,CURRENT
00021 COMMON /VBUF/ PAGSZ,NCHAN,ICHAN,CHFAND,PAGHDR,BFESZ,DIRSZ,
00022 INCORE,HDRSZ,MSADR,STESZ,BIASZ,DIRBK,PAGHDR
00023 COMMON /VHVDI/ THMSA,REGAS,LAGNO,CONTG,INSDI
00024 USECT,HDRFIC,READVI,OLDINH,DNODEH,NROWH,DROWH
00025 COMMON /VNSWT/ SETUP,SNGI,BA,NYTRAS,IN1STR,IN2STR,ED1STR,ED2STR,
00026 D1STR,D2STR,IN2TMP,ED2TMP,D1TMP,D2TMP,DUMPEI,
00027 FINDEI,D1TFI,NSRTEI
00028 COMMON /VVSFO/ ISFOSZ,ISOPDS,LASTSQ,SEOSP,0,0
00029 COMMON /VPRAB/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BUD,MODE,PAGES
00030 IUN
00031 COMMON /VADDR/ IADD,THIS,ISTHED,LOC,LAST,LASTL
00032 COMMON /VSTAK/ CURLEV,NUMVAR,STACK,0,0,0
00033 COMMON /VINS/ REORG,FULL,RPLACF
00034 COMMON /VCRN/ RUSTYV,CORE

```

00035 DATA PART 1

THIS ROUTINE IS USED TO DISPOINT DISK I/O ERRORS

```

00036 IF PAUSE IN LVERR DUMP
00037 IF ERRNUM GT 0 GO TO 4
00038 IERR = ERRNUM IERR
00039 FORMAT (24H **** DISK I/O ERROR NO. 13 SA IERRVAL = 13)
00040 FULL DUMP PLACED ON LV ERROR ERR
00041 FORMAT (24H 13)
00042 FORMAT (24H 13)
00043 FORMAT (24H 13)
00044 FORMAT (24H 13)
00045 FORMAT (24H 13)
00046 FORMAT (24H 13)
00047 WRITE IUN 10
00048 FORMAT (1ADD,THIS,ISTHED,LOC,LAST,LASTL)
00049 WRITE IUN 21 (1ADD,THIS,ISTHED,LOC,LAST,LASTL)
00050 WRITE IUN 30
00051 WRITE IUN 31

```

```

0035 11  FORMAT(' IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, ITESTR, INCIUD,
1  INPXON' /
1  ' IVALS(1), ITYP1(1), SRCSUF, LNKSF, SNKSUF, INSTYP')
0036  WRITE(LUN, 2)
1  IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, ITESTR, INCIUD, INPXON,
1  IVALS(1), ITYP1(1),
0037  2  SRCSUF, LNKSF, SNKSUF, INSTYP
1  IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, ITESTR, INCIUD, INPXON,
1  IVALS(1), ITYP1(1),
0038  2  SRCSUF, LNKSF, SNKSUF, INSTYP
0039  WRITE(LUN, 4)
0040  12  FORMAT(' CURPAG(4)', 30X, 'REQPAG(1)', /' LSTVPG(4)')
0041  WRITE(LUN, 6) CURPAG, REQPAG, LSTVPG
0042  WRITE(LUN, 4)
0043  122  FORMAT(' MSARFT, HREQPG, NXTMSA, HACTPG(1), HACTPG(2), RFADCT, USECNT,
2  DIRPAG, DIRCNT, OUTLOC')
0045  WRITE(LUN, 2)
1  MSARFT, HREQPG, NXTMSA, HACTPG, RFADCT, USECNT, DIRPAG,
2  DIRCNT, OUTLOC
0046  WRITE(LUN, 4)
0047  13  FORMAT(' PRIME, SFED, NROW, DNODE, DROW, OI DNOP, LISTSZ')
0048  WRITE(LUN, 2) PRIME, SEED, NROW, DNODE, DROW, OI DNOP, LISTSZ
0049  WRITE(LUN, 4)
0050  14  FORMAT(' REGASP, CTRI PT, CTRI 1, LEASTV, NTFREE, FREE, DREGSP, MSA,
1  PAGIOC')
0053  WRITE(LUN, 2)
1  REGASP, CTRI PT, CTRI 1, LEASTV, NTFREE, FREE, DREGSP, MSA,
1  PAGIOC
0054  WRITE(LUN, 4)
0055  15  FORMAT(' PAGESZ, NWCHAN, OI CHAN, CMPAND, PAGHDR, BUFSZ, DIRSZ,
1  INCORE, HDRSZF, MSADIR' /' SUFSZ, BIKSZ, DIRBIK, PAGHD4')
0056  WRITE(LUN, 2)
1  PAGESZ, NWCHAN, OI CHAN, CMPAND, PAGHDR, BUFSZ, DIRSZ,
1  INCORE, HDRSZF, MSADIR, SUFSZ, BIKSZ, DIRBIK, PAGHD4
0058  WRITE(LUN, 4)
0059  16  FORMAT(' SETUP, SNGI BK, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR, DL1STR,
1  DL2STR, FINDFI' /' DLETFI, NSRTFI')
0060  WRITE(LUN, 5) SETUP, SNGI BK, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
1  DL1STR, DL2STR, FINDFI, DLETFI, NSRTFI
0061  WRITE(LUN, 4)
0062  17  FORMAT(' FULL, REORG, CURENT')
0063  WRITE(LUN, 5) FULL, REORG, CURENT
0064  WRITE(LUN, 4)
0065  18  FORMAT(' BUFI OC, LENGTH, IERR, ERRNUM')
0066  WRITE(LUN, 2) BUFI OC, LENGTH, IERR, ERRNUM
0067  IF (DUMP .EQ. PART) GO TO 30
0068  WRITE(LUN, 4)
0069  WRITE(LUN, 123)

```

```

0074 123 FORMAT(' OUTDIR(256)')
0075 WRITE(LUN,6) OUTDIR
0076 WRITE(LUN, 4)
0077 WRITE(LUN, 124)
0078 124 FORMAT(' RWBUF(256)')
0079 WRITE(LUN,6) RWBUF
0080 WRITE(LUN, 4)
0081 WRITE(LUN, 131)
0082 131 FORMAT(' GRNTBI(256)')
0083 WRITE(LUN,6) GRNTBI
0084 WRITE(LUN, 4)
0085 WRITE(LUN, 19)
0086 19 FORMAT(' STACK')
0087 WRITE(LUN, 2) STACK
0088 30 WRITE(LUN, 4)
0089 IF(DUMPF1.EQ.'TRUE.') RETURN
0091 CALL CLOSEFC(NWCHAN)
0092 STOP
0093 END

C
C
C

0001 SUBROUTINE LVRTRN
0002 IMPLICIT INTEGER(A-Z)
0003 LOGICAL*1 CURENT
0004 COMMON /I VARG/ IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, ITESTR,
1 INCIUD, INDXON, IVALS(10), ITYP1(10), SRCSUF,
2 LNKSUF, SNKSUF, INSTYP
0005 COMMON /I VREGS/ CURPAG(4), REQOPAG(4), LSTVPG(4), MSARET,
1 HREQPG, NXTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
2 DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)
0006 COMMON /I VFI AG/ FI 0MSK, FI 1MSK, FI 2MSK, FI 3MSK, FI 4MSK, FI 5MSK, FI 67,
1 FI AG8, FI AG9, FI AG10, FI AG11, FI AG12, FI AG13, FI AG14,
2 FI AG15
0007 COMMON /I VCRNT/ REGASP, CTRI PT, CTRI 1, LEASTV, NTFREE, FREE, DREGSP,
1 MSA, PAGLOC, CURENT
0008 COMMON /I VHDVI/ THMSA, REGAS, PAGENO, CONTNO, INSTEL,
1 USECT, HDRFIG, READVI, OI DNP, DNODEH, NROWH, DROWH
0009 COMMON /I VVTR1/ NOPSPC(1)
1 /I VVTR2/ LSTSPC(1)
2 /I VVTR3/ LNKSPC(1)
3 /I VVTR4/ FIGSPC(1)

C
C THE PURPOSE OF THIS ROUTINE IS TO "UNDEFINE" A NODE.
D PAUSE 'IN LVRTRN'
0010 FIGSPC(CTRI 1 + IVAL) = FIGSPC(CTRI 1 + IVAL) .AND. .NOT.FI 3MSK
0011 RETURN
0012 END

```

```

      C
      C
      C
0001      FUNCTION LVRTSH(WORD,BITS)
0002      IMPLICIT INTEGER(A-Z)
      C
      C THIS FUNCTION PERFORMS A RIGHT LOGICAL SHIFT
      C
0003      IF(BITS .EQ. 0) GO TO 10
0005      IF(BITS .GE. 16) GO TO 20
0007      LVRTSH = WORD / 2 ** (BITS)
0008      RETURN
0009  10      LVRTSH = WORD
0010      RETURN
0011  20      LVRTSH = 0
0012      RETURN
0013      END

      C
      C
      C
0001      FUNCTION LVIFSH(WORD,BITS)
0002      IMPLICIT INTEGER(A-Z)
      C
      C THIS FUNCTION PERFORMS A LEFT LOGICAL SHIFT
      C
0003      IF(BITS .EQ. 0) GO TO 10
0005      IF(BITS .GE. 16) GO TO 20
0007      LVIFSH = WORD * 2 ** (BITS)
0008      RETURN
0009  10      LVIFSH = WORD
0010      RETURN
0011  20      LVIFSH = 0
0012      RETURN
0013      END

```

COMMON BLOCK /IVARGS/ LENGTH 000104

IFUNC	000000	INTEGER*2	VARIABLE
IARG	000002	INTEGER*2	VARIABLE
IPOS	000004	INTEGER*2	VARIABLE
ITYP	000006	INTEGER*2	VARIABLE
IVAL	000010	INTEGER*2	VARIABLE
NVAL	000012	INTEGER*2	VARIABLE
NSKIP	000014	INTEGER*2	VARIABLE
ITESTR	000016	INTEGER*2	VARIABLE
INCIUD	000020	INTEGER*2	VARIABLE
INPXON	000022	INTEGER*2	VARIABLE
IVALS	000024	INTEGER*2	ARRAY (10)
ITYPI	000050	INTEGER*2	ARRAY (10)
SRCSTF	000074	INTEGER*2	VARIABLE
LNKSUF	000076	INTEGER*2	VARIABLE
SNKSUF	000100	INTEGER*2	VARIABLE
INSTYP	000102	INTEGER*2	VARIABLE

COMMON BLOCK /IVREGS/ LENGTH 001056

CURPAG	000000	INTEGER*2	ARRAY (4)
REQPAG	000010	INTEGER*2	ARRAY (4)
LSTVPG	000020	INTEGER*2	ARRAY (4)
NAME	OFFSET	ATTRIBUTES	
MSARET	000030	INTEGER*2	VARIABLE
HREQPG	000032	INTEGER*2	VARIABLE
NXTMSA	000034	INTEGER*2	VARIABLE
HACTPG	000036	INTEGER*2	ARRAY (2)
READCT	000042	INTEGER*2	VARIABLE
USECNT	000044	INTEGER*2	VARIABLE
DIRPAG	000046	INTEGER*2	VARIABLE
DIRCNT	000050	INTEGER*2	VARIABLE
OUTLOC	000052	INTEGER*2	VARIABLE
OUTDIR	000054	INTEGER*2	ARRAY (256)
RWBUP	001054	INTEGER*2	ARRAY (1)

COMMON BLOCK /VMASK/ LENGTH 000014

MWRITE	000000	INTEGER*2	VARIABLE
NOTUST	000002	INTEGER*2	VARIABLE
NEWCON	000004	INTEGER*2	VARIABLE
FI GMSK	000006	INTEGER*2	VARIABLE
MASKSF	000010	INTEGER*2	VARIABLE
MASKPF	000012	INTEGER*2	VARIABLE

COMMON BLOCK /VFIAG/ LENGTH 000036

FI 0MSK	000000	INTEGER*2	VARIABLE
FI 1MSK	000002	INTEGER*2	VARIABLE
FI 2MSK	000004	INTEGER*2	VARIABLE
FI 3MSK	000006	INTEGER*2	VARIABLE
FI 4MSK	000010	INTEGER*2	VARIABLE
FI 5MSK	000012	INTEGER*2	VARIABLE
FI G67	000014	INTEGER*2	VARIABLE
FI AG8	000016	INTEGER*2	VARIABLE
FI AG9	000020	INTEGER*2	VARIABLE
FI AG10	000022	INTEGER*2	VARIABLE
FI AG11	000024	INTEGER*2	VARIABLE
FI AG12	000026	INTEGER*2	VARIABLE
FI AG13	000030	INTEGER*2	VARIABLE
FI AG14	000032	INTEGER*2	VARIABLE
FI AG15	000034	INTEGER*2	VARIABLE

COMMON BLOCK /VVRAND/ LENGTH 001016

PRIME	000000	INTEGER*2	VARIABLE
SEED	000002	INTEGER*2	VARIABLE
NROW	000004	INTEGER*2	VARIABLE
DNODE	000006	INTEGER*2	VARIABLE
DROW	000010	INTEGER*2	VARIABLE
OLDNOD	000012	INTEGER*2	VARIABLE
LISTSZ	000014	INTEGER*2	VARIABLE
GRNTBI	000016	INTEGER*2	ARRAY (256)

COMMON BLOCK /1VCRNT/ LENGTH 000023

REGASP	000000	INTEGER*2	VARIABLE
CTRIPT	000002	INTEGER*2	VARIABLE
CTRI 1	000004	INTEGER*2	VARIABLE
LEASTV	000006	INTEGER*2	VARIABLE
NTFREE	000010	INTEGER*2	VARIABLE

NAME OFFSET ATTRIBUTES

FREE	000012	INTEGER*2	VARIABLE
DREGSP	000014	INTEGER*2	VARIABLE
MSA	000016	INTEGER*2	VARIABLE
PAGLOC	000020	INTEGER*2	VARIABLE
CURRENT	000022	LOGICAL*1	VARIABLE

COMMON BLOCK /1VBVFR/ LENGTH 000034

PAGSZE	000000	INTEGER*2	VARIABLE
NWCHAN	000002	INTEGER*2	VARIABLE
OICHAN	000004	INTEGER*2	VARIABLE
COMPAND	000006	INTEGER*2	VARIABLE
PAGHDR	000010	INTEGER*2	VARIABLE
BUFSZE	000012	INTEGER*2	VARIABLE
DIRSZE	000014	INTEGER*2	VARIABLE
INCORE	000016	INTEGER*2	VARIABLE
HDRSZE	000020	INTEGER*2	VARIABLE
MSADIR	000022	INTEGER*2	VARIABLE
SUFSZE	000024	INTEGER*2	VARIABLE
BIKSZE	000026	INTEGER*2	VARIABLE
DIRBLK	000030	INTEGER*2	VARIABLE
PAGHDR4	000032	INTEGER*2	VARIABLE

COMMON BLOCK /1VBDVL/ LENGTH 000030

THSWSA	000000	INTEGER*2	VARIABLE
REGAS	000002	INTEGER*2	VARIABLE
PAGENO	000004	INTEGER*2	VARIABLE
CONTNO	000006	INTEGER*2	VARIABLE
INSDVL	000010	INTEGER*2	VARIABLE
USECT	000012	INTEGER*2	VARIABLE
HDRFLG	000014	INTEGER*2	VARIABLE
READVL	000016	INTEGER*2	VARIABLE
OLDNTH	000020	INTEGER*2	VARIABLE
DNONEH	000022	INTEGER*2	VARIABLE
NROWB	000024	INTEGER*2	VARIABLE
DROWB	000026	INTEGER*2	VARIABLE

COMMON BLOCK /1VSWIT/ LENGTH 000020

SETUP	000000	LOGICAL*1	VARIABLE
SGLBK	000001	LOGICAL*1	VARIABLE
NXTRAN	000002	LOGICAL*1	VARIABLE
IN1STR	000003	LOGICAL*1	VARIABLE
IN2STR	000004	LOGICAL*1	VARIABLE
FD1STR	000005	LOGICAL*1	VARIABLE
FD2STR	000006	LOGICAL*1	VARIABLE
DL1STR	000007	LOGICAL*1	VARIABLE
DL2STR	000010	LOGICAL*1	VARIABLE
IN2TMP	000011	LOGICAL*1	VARIABLE
FD2TMP	000012	LOGICAL*1	VARIABLE
DL2TMP	000013	LOGICAL*1	VARIABLE
DUMPF1	000014	LOGICAL*1	VARIABLE
FINDF1	000015	LOGICAL*1	VARIABLE
DETF1	000016	LOGICAL*1	VARIABLE
NSRTF1	000017	LOGICAL*1	VARIABLE

COMMON BLOCK /1VVSEQ/ LENGTH 000010

ISEQSZ	000000	INTEGER*2	VARIABLE
ISOPOS	000002	INTEGER*2	VARIABLE
LASTSO	000004	INTEGER*2	VARIABLE
SEQSPC	000006	INTEGER*2	ARRAY (1)

COMMON BLOCK /IVPRAM/ LENGTH 000022

BUFILOC 000000 INTEGER*2 VARIABLE
LENGTH 000002 INTEGER*2 VARIABLE
IERR 000004 INTEGER*2 VARIABLE
ERRNUM 000006 INTEGER*2 VARIABLE
BINARY 000010 INTEGER*2 VARIABLE
BCD 000012 INTEGER*2 VARIABLE
MODE 000014 INTEGER*2 VARIABLE
PAGES 000016 INTEGER*2 VARIABLE
LUN 000020 INTEGER*2 VARIABLE

COMMON BLOCK /IVSTAK/ LENGTH 000006

CURLEV 000000 INTEGER*2 VARIABLE
NUMVAR 000002 INTEGER*2 VARIABLE
STACK 000004 INTEGER*2 ARRAY (1)

COMMON BLOCK /IVUTIL/ LENGTH 000126

FILSPC 000000 INTEGER*2 ARRAY (39)
DEFEXT 000116 REAL*4 ARRAY (2)

COMMON BLOCK /IVINS1/ LENGTH 000003

REORG 000000 LOGICAL*1 VARIABLE
FULL 000001 LOGICAL*1 VARIABLE
RPLACE 000002 LOGICAL*1 VARIABLE

COMMON BLOCK /IVRUN/ LENGTH 000006

RUNTP 000000 INTEGER*2 VARIABLE
CORE 000002 REAL*4 VARIABLE

COMMON BLOCK /IVVTR1/ LENGTH 000002

NOTSPC 000000 INTEGER*2 ARRAY (1)

COMMON BLOCK /IVVTR2/ LENGTH 000002

LSTSPC 000000 INTEGER*2 ARRAY (1)

COMMON BLOCK /IVVTR3/ LENGTH 000002

LNKSPC 000000 INTEGER*2 ARRAY (1)

COMMON BLOCK /IVVTR4/ LENGTH 000002

FIGSPC 000000 INTEGER*2 ARRAY (1)

COMMON BLOCK /IVDEL1/ LENGTH 000003

NUMRET 000000 INTEGER*2 VARIABLE
BAKCON 000002 LOGICAL*1 VARIABLE

COMMON BLOCK /IVADDR/ LENGTH 000014

IADD 000000 INTEGER*2 VARIABLE
THIS 000002 INTEGER*2 VARIABLE
LSTHED 000004 INTEGER*2 VARIABLE
LOC 000006 INTEGER*2 VARIABLE
LAST 000010 INTEGER*2 VARIABLE
LASTLC 000012 INTEGER*2 VARIABLE

COMMON BLOCK /IVFND/ LENGTH 000005

COUNT 000000 INTEGER*2 VARIABLE
ABSP0S 000002 INTEGER*2 VARIABLE
LSTCON 000004 LOGICAL*1 VARIABLE

REFERENCES

1. Zaritsky, I., "GIRS (Graph Information Retrieval System) Users Manual," DTNSRDC Report 79/036 (Apr 1979).
2. Berkowitz, S., "Design Trade-offs for a Software Associative Memory," DTNSRDC Report 3531 (May 1973).
3. Zaritsky, I., "Feasibility Study for Incorporating A Data Structure Definition and Manipulation Facility Within The COMRADE Data Management System," DTNSRDC Report 78/045 (May 1978).
4. Carlberg, J., "A Paged Hardware Associative Memory - Preliminary Report," DTNSRDC Report 77-0083 (Aug 1977).
5. Berkowitz, S., "Graph Information Retrieval Language; Programming Manual for FORTRAN Complement; Revision One," DTNSRDC Report 76-0085 (Feb 1976).
6. "Data Handler, Version 1.0, Reference Manual, Revision B," Control Data Corporation Publication No. 17322100 (Jan 1976).

INITIAL DISTRIBUTION

Copies		Copies	Code	Name
1	CHONR	1	1826	L. Culpepper
		1	184	J. Schot
1	NRL	1	184.1	H. Feingold
1	NSWC	1	1843	H. Haussling
1	NUSC	1	1844	S. Dhir
		1	1844	J. McKee
1	NOSC	1	185	T. Corin
1	NAVSUP/0431C, G. Bernstein	1	1850	A. Cinque
		1	1851	J. Brainin
2	NAVSEA	1	1854	H. Sheridan
	1 SEA 312	1	1855	R. Brengs
	1 SEA 612	1	187	M. Zubkoff
1	Rome Air Development Center	1	189	G. Gray
12	DTIC	10	5211.1	Reports Distribution
		1	522.1	Unclass Lib (C)
		1	522.2	Unclass Lib (A)

CENTER DISTRIBUTION

Copies	Code	Name
1	18	G. Gleissner
1	1802.2	F. Frenkiel
1	1803	S. Rainey
1	1804	L. Avrunin
1	1805	E. Cuthill
1	1806	R. Santamaria
2	1809.3	D. Harris
1	182	A. Camara
1	1821	D. Jefferson
1	1822	T. Rhodes
1	1824	S. Berkowitz
1	1824	J. Carlberg
1	1824	J. Garner
1	1824	P. Marques
1	1824	C. Slominski
20	1824	I. Zaritsky

