<table>
<thead>
<tr>
<th><strong>1. REPORT NUMBER</strong></th>
<th>DTNRSRC-81/0241</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>2. GOVT ACCESSION NO.</strong></td>
<td>DT-M-81-0241</td>
</tr>
<tr>
<td><strong>3. RECIPIENT'S CATALOG NUMBER</strong></td>
<td>AD-A099125</td>
</tr>
<tr>
<td><strong>4. TITLE (and Subtitle)</strong></td>
<td>PAGED GIRS (GRAPH INFORMATION RETRIEVAL SYSTEM) USERS MANUAL</td>
</tr>
<tr>
<td><strong>5. TYPE OF REPORT &amp; PERIOD COVERED</strong></td>
<td>Final</td>
</tr>
<tr>
<td><strong>6. PERFORMING ORG. REPORT NUMBER</strong></td>
<td></td>
</tr>
<tr>
<td><strong>7. AUTHOR(s)</strong></td>
<td>Irving S. Zaritsky</td>
</tr>
<tr>
<td><strong>8. CONTRACT OR GRANT NUMBER(s)</strong></td>
<td></td>
</tr>
<tr>
<td><strong>9. PERFORMING ORGANIZATION NAME AND ADDRESS</strong></td>
<td>David W. Taylor Naval Ship Research and Development Center, Bethesda, Maryland 20084</td>
</tr>
<tr>
<td><strong>10. PROGRAM ELEMENT, PROJECT, TASK AREA &amp; WORK UNIT NUMBERS</strong></td>
<td>(See reverse side)</td>
</tr>
<tr>
<td><strong>11. CONTROLLING OFFICE NAME AND ADDRESS</strong></td>
<td>Naval Sea Systems Command, Washington, D.C. 20362</td>
</tr>
<tr>
<td><strong>12. REPORT DATE</strong></td>
<td>May 1981</td>
</tr>
<tr>
<td><strong>13. NUMBER OF PAGES</strong></td>
<td>156</td>
</tr>
<tr>
<td><strong>14. MONITORING AGENCY NAME &amp; ADDRESS (if different from Controlling Office)</strong></td>
<td></td>
</tr>
<tr>
<td><strong>15. SECURITY CLASS. (of this report)</strong></td>
<td>UNCLASSIFIED</td>
</tr>
<tr>
<td><strong>15a. DECLASSIFICATION/DETERMINING SCHEDULE</strong></td>
<td></td>
</tr>
<tr>
<td><strong>16. DISTRIBUTION STATEMENT (of this report)</strong></td>
<td>APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED</td>
</tr>
<tr>
<td><strong>17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)</strong></td>
<td></td>
</tr>
<tr>
<td><strong>18. SUPPLEMENTARY NOTES</strong></td>
<td></td>
</tr>
<tr>
<td><strong>19. KEY WORDS (Continue on reverse side if necessary and identify by block number)</strong></td>
<td>Associative Memory, Hashed Addressing, Data Base, Information Retrieval System, Data Definition Language, Data Base Management System, Data Management, Paging Schemes, Graph</td>
</tr>
<tr>
<td><strong>20. ABSTRACT (Continue on reverse side if necessary and identify by block number)</strong></td>
<td>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</td>
</tr>
</tbody>
</table>
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.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES.</td>
<td>v</td>
</tr>
<tr>
<td>ABSTRACT.</td>
<td>1</td>
</tr>
<tr>
<td>ADMINISTRATIVE INFORMATION.</td>
<td>1</td>
</tr>
<tr>
<td>INTRODUCTION.</td>
<td>1</td>
</tr>
<tr>
<td>BRIEF DESCRIPTION.</td>
<td>2</td>
</tr>
<tr>
<td>MOTIVATION.</td>
<td>3</td>
</tr>
<tr>
<td>MEMORY SCHEME.</td>
<td>4</td>
</tr>
<tr>
<td>THE GIRS BUFFER.</td>
<td>4</td>
</tr>
<tr>
<td>THE CONTINUANT</td>
<td>4</td>
</tr>
<tr>
<td>THE DIRECTORY.</td>
<td>6</td>
</tr>
<tr>
<td>THE BUFFER COMPOSITION</td>
<td>7</td>
</tr>
<tr>
<td>INITIALIZATION OF THE GIRS BUFFER.</td>
<td>7</td>
</tr>
<tr>
<td>COMMON LVBUF0R.</td>
<td>8</td>
</tr>
<tr>
<td>REPRESENTATION OF NODES AND LINKS.</td>
<td>9</td>
</tr>
<tr>
<td>PAGE AND CONTINUANT DETERMINATION FOR THE TRIPLE</td>
<td>10</td>
</tr>
<tr>
<td>Page Determination.</td>
<td>10</td>
</tr>
<tr>
<td>Continuant Determination.</td>
<td>10</td>
</tr>
<tr>
<td>THE FLAG FIELD.</td>
<td>11</td>
</tr>
<tr>
<td>DISK FORMAT.</td>
<td>13</td>
</tr>
<tr>
<td>PAGING SCHEME</td>
<td>15</td>
</tr>
<tr>
<td>GENERAL DISCUSSION</td>
<td>15</td>
</tr>
<tr>
<td>I/O FOR THE DIRECTORIES OF DISK LOCATIONS OF CONTINUANTS</td>
<td>15</td>
</tr>
<tr>
<td>I/O FOR THE CONTINUANTS.</td>
<td>16</td>
</tr>
<tr>
<td>PHILOSOPHY</td>
<td>17</td>
</tr>
<tr>
<td>USER-CALLABLE GIRS SUBROUTINES.</td>
<td>18</td>
</tr>
<tr>
<td>INITIALIZATION.</td>
<td>18</td>
</tr>
<tr>
<td>Getting Started</td>
<td>18</td>
</tr>
<tr>
<td>Subroutine LVSETP</td>
<td>22</td>
</tr>
<tr>
<td>Subroutine LVCRN.</td>
<td>24</td>
</tr>
<tr>
<td>RETRIEVAL OF VALUES.</td>
<td>25</td>
</tr>
<tr>
<td>Discussion.</td>
<td>25</td>
</tr>
<tr>
<td>Subroutine LVFDEX</td>
<td>26</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>RETRIEVAL OF MVL INDEX OF GIVEN VALUE OF A FUNCTION (INCLUSION)</td>
<td>30</td>
</tr>
<tr>
<td>Subroutine LVINCL</td>
<td>30</td>
</tr>
<tr>
<td>INSERTION</td>
<td>31</td>
</tr>
<tr>
<td>Discussion</td>
<td>31</td>
</tr>
<tr>
<td>Subroutine LVINEX</td>
<td>32</td>
</tr>
<tr>
<td>Subroutine LVREOR</td>
<td>38</td>
</tr>
<tr>
<td>DELETION</td>
<td>39</td>
</tr>
<tr>
<td>Discussion</td>
<td>39</td>
</tr>
<tr>
<td>Subroutine LVDLEX</td>
<td>40</td>
</tr>
<tr>
<td>DISK STORAGE AND RETRIEVAL OF A GRAPH.</td>
<td>43</td>
</tr>
<tr>
<td>Discussion</td>
<td>43</td>
</tr>
<tr>
<td>Subroutine LVDUMP</td>
<td>44</td>
</tr>
<tr>
<td>Subroutine LVFECH</td>
<td>46</td>
</tr>
<tr>
<td>EXECUTING A GIRS PROGRAM</td>
<td>47</td>
</tr>
<tr>
<td>GENERAL DISCUSSION</td>
<td>47</td>
</tr>
<tr>
<td>INDIRECT USE OF A GIRS SUBROUTINE VIA GIRL</td>
<td>47</td>
</tr>
<tr>
<td>Preprocessing and Compiling of GIRL/FORTRAN Program</td>
<td>49</td>
</tr>
<tr>
<td>DIRECT USE OF GIRS SUBROUTINES</td>
<td>50</td>
</tr>
<tr>
<td>Linking and Executing a GIRL/FORTRAN Program</td>
<td>50</td>
</tr>
<tr>
<td>OVERLAY STRUCTURE</td>
<td>52</td>
</tr>
<tr>
<td>LIMITATIONS AND MEMORY REQUIREMENTS</td>
<td>54</td>
</tr>
<tr>
<td>ADDING A USER-EMBEDDED STRATEGY</td>
<td>56</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>56</td>
</tr>
<tr>
<td>USE.</td>
<td>57</td>
</tr>
<tr>
<td>PROPOSED EXTENSIONS</td>
<td>57</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>57</td>
</tr>
<tr>
<td>APPENDIX A - VARIABLES IN LABELED COMMON</td>
<td>59</td>
</tr>
<tr>
<td>APPENDIX B - SUBROUTINE CALLING STRUCTURE</td>
<td>61</td>
</tr>
<tr>
<td>APPENDIX C - SUBROUTINE LISTINGS</td>
<td>65</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>149</td>
</tr>
</tbody>
</table>
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The Continuant Header</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>The Flag Field</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>The Disk Format</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>Overlay Region Sizes</td>
<td>52</td>
</tr>
<tr>
<td>5</td>
<td>The Overlay Structure</td>
<td>53</td>
</tr>
</tbody>
</table>
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 database. 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 database 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 database. 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. 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 Berkomwitz, "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 memory 4 will be merged with out-core GIRS. The result will be an enhanced system with high speed relational processing.
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 CPO.

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]  
<table>
<thead>
<tr>
<th></th>
<th>b) The REGISTER of Available SPACE (REGASP) for this continuant [REGAS]</th>
</tr>
</thead>
</table>
| 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) LWWRIT - 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:

\[ \text{LVDSRZ} = 64 \times (\text{LVNCOR}/64) + 1 \]

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:

\[ \text{LOC} = (\text{Page No.} + \text{Cont. No} + 1) \mod (\text{LVDSRZ}) \]
THE BUFFER COMPOSITION

A typical buffer may be illustrated as follows:

<table>
<thead>
<tr>
<th>Location</th>
<th>Length</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NODSPC</td>
<td>LSTSPC</td>
</tr>
<tr>
<td>1</td>
<td>LVDRSZ</td>
<td>Directory</td>
</tr>
<tr>
<td>CP1</td>
<td>LVHDRS</td>
<td>Continuant Header</td>
</tr>
<tr>
<td></td>
<td>LVVSZE</td>
<td>Body of Continuant</td>
</tr>
<tr>
<td>CP2</td>
<td>LVHDRS</td>
<td>Continuant Header</td>
</tr>
<tr>
<td></td>
<td>LVVSZE</td>
<td>Body of Continuant</td>
</tr>
<tr>
<td>CPn</td>
<td>LVPGHD</td>
<td>Continuant Header</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Body of Continuant</td>
</tr>
</tbody>
</table>

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

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

LVHDRE - Single array length of the header

\[ [HPRSZE] = 2 \]

LVPGHED - Single array length of the combined continuant and header

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

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
\[ \text{BLKSZE} = 4 \times \text{PAGHDR}/256 \]
LVPGH4 - Total length, in words, of one continuant
\[ \text{PACHG4} = 4 \times \text{PAGHDR} \]
LVNCO - 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
\[ \text{DIRSZ} = 64 \times ((\text{INCORE}/64) + 1) \]
LVBSZ - Total single array length of the in-core buffer. NODSPC, LSTSPC, LNKSPC, and FLGSPC are all dimensioned to this value
\[ \text{BUFSZ} = \text{DIRSZ} + (\text{INCORE} \times \text{PAGHDR}) \]
LVDBK - Number of blocks required to hold the in-core directory on disk
\[ \text{DIRBLK} = 4 \times \text{DIRSZ}/256 \]
LVMSAD - Location on disk (relative block number) of the in-core directory
\[ \text{MSADIR} = 2 \]

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 \(1 \leq \text{LVREQP}(1) \leq 63\) or assigned by LVGRN \(\text{LVREQP}(1) = -1\) 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:

<table>
<thead>
<tr>
<th>Page No.</th>
<th>Random Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
</tr>
</tbody>
</table>
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 ≤ n ≤ 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 ≤ n ≤ 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 MERGED\textsuperscript{2} and also a list may be SEPARATED\textsuperscript{2} 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

<table>
<thead>
<tr>
<th>14</th>
<th>13</th>
<th>1?</th>
<th>11</th>
<th>10</th>
<th>9-8</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6-7</th>
</tr>
</thead>
</table>

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

<table>
<thead>
<tr>
<th>FLAG</th>
<th>FLAG VALUE</th>
<th>CONTENTS OF ASSOCIATED LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flag 0</td>
<td>$2^7$</td>
<td>Head of a multivalued list.</td>
</tr>
<tr>
<td>Flag 1</td>
<td>$2^6$</td>
<td>Location already occupied.</td>
</tr>
<tr>
<td>Flag 2</td>
<td>$2^5$</td>
<td>A value on a multivalued list.</td>
</tr>
<tr>
<td>Flag 3</td>
<td>$2^4$</td>
<td>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.</td>
</tr>
<tr>
<td>Flag 4</td>
<td>$2^3$</td>
<td>Head of a multivalued list which has been modified either by an insertion or by an indexed deletion, thus bypassing the &quot;saved index&quot; upon retrieval feature. (See the description of Subroutine LVFNV for further details.)</td>
</tr>
<tr>
<td>Flag 5</td>
<td>$2^2$</td>
<td>Head of a conflict list.</td>
</tr>
</tbody>
</table>
| 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

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

†Each block contains 256 words.
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 LVCLOSE 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.

\[
LVDRPG = (\text{Page No.} - 1)/4 + 1 \\
\text{(value range } = 1-16)
\]

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

\[
LVDIRC = \text{Page No.} - 4*(LVDRPG - 1) \\
\text{(value range } = 1-4)
\]

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

\[
LVOTLC = 1 + 64*(LVDIRC - 1) + \text{Cont. No.} \\
\text{(value range } = 1-256)
\]

---

† 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. 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:

\[
\text{value} = A \times \text{order} + B \times \text{usage} + C \times \text{space} + D \times \text{write}
\]

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 LDVLEX 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, LVVPOS, LVVTYP, LVVAL, LVVNVL, LVSKIP,
1   LVVTR, LVVIN, LVNDXL, LVVALS(10), LVTYPE(10)
2   ,LVNSRF, LVVNSL, LVNSLF, LVNTYP
COMMON /LVSEQ/ LVSIZE, LVSEQ1, LVSEQ2, SEQSPC(1)
COMMON /LVRAND/ LVKPRMLVKS, LVKS, LVKY, LVKDY, LVKTK, LVTEMP, LVLIST, LVNTBL(256)
COMMON /LVVT1/ NOCTSPC(buffer size)
1   /LVVT2/ LSTSPC(buffer size)
2   /LVVT3/ LNKSPC(buffer size)
3   /LVVT4/ FLGSPC(buffer size)
COMMON /LVCRNT/ LVGSP, LVCTRL, LCTRL1, LVSTV, LVNFRE, LVFREE,
1   LVDREG, LVMMSA, LVPGLC, LVCRNT
COMMON /LVBUFR/ LVVSIZE, LVNWCH, LVOLCH, LVCMPR, LVPGHD, LVBFSZ
1   LVDRESZ, LVNCOR, LVHDRS, LVMSAD,
2   ,LVSFSZ, LVBKSZ, LVDRBK, LVPGH4
COMMON /LVREGS/ LVCUPG(4), LVREQF(4), LVLP(4), LVMSAR,
1   LVPHRPG, LVNMSA, LVHAPG(2), LVRCNT, LVUCNT, LVDRPG,
2   LVDIRC, LVOTLC, LVOTDR(256),
3   LVRWBF(4*continuant size)
COMMON /LVFRM/ LVFLC, LVNTH, LVERR, LVERBO, LVBNRY, LVBCD, LVMODE, LVPQS, LVNL
COMMON /LVRUN/ LVNRP, LVCORE
COMMON /LVSTAK/ LVLEVL, LVNVAR, LVSTAK(140)
COMMON /LVMASK/ LVWRIT,LVUSE,LVWNCN,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 subroutines 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,LVVPOS,LVVTYP,LVVAL,LVNVNL,LVSKIP,
1     LVVTR,LVVINC,LVNDXN,LVVALS(10),LVTYPE(10),
2     LVSRSF,LVLNSF,LVNSNSF,LVNTYP
COMMON /LVREGS/ LVCPUPG(4),LVRQEP(4),LVLPVG(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/ LVBLFC, 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†      LVMSSF†
LVFSZ†

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 (LDVRSZ) 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

$$64*(\text{LVNCOR}/64)+1 + \text{LVNCOR}*(\text{LVVSZE}+2)$$

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

$$\text{LVMSPF to 170000}$$
$$\text{and LVMSSF to 7777}$$

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 zero$^{th}$) 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 $>\frac{\text{VLVVSZE}}{2}$.

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:

$$\text{LVLUN = 17}$$
$$\text{CALL ASSIGN(17,'SY:ERROR.ERR',12)}$$

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/. 

21
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 \(\geq (\sqrt{LVVSZE})/2\).

(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".

22
Program Length: 
26158  142110

Subroutines Called:
LVFECH   LVDRWR   ICSI
LVGRN    LVMSA    IGETC
LVPGW    LVFIND   IFETCH
LVNSRT   LVPAGR   IENTER
          LOOKUP

Called by the Following Subroutines:
LVNPAG
LVNCWN
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 ≤ n ≤ 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 Berkowitz and Zaritsky.

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 DEFINE 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:

\[
\begin{array}{c}
750 \\
8 \\
488 \\
10
\end{array}
\]

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 IPOS\textsuperscript{th} 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 ≤ n ≤ 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 USRFDF1]

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 GRRR Code:**

```
C SET LVREQP(2) TO THE REQUESTED CONTINUANT, IF DESIRED.
G NODE+LINK.<t.s.J
```

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:

<table>
<thead>
<tr>
<th></th>
<th>12548</th>
<th>68610</th>
</tr>
</thead>
</table>

Subroutines Called:

LVSTAC LVFIND LVRTSH LVBOTM
LVPOP LVFNV LVEXCH

Called by the Following Subroutines:

LVSLER LVVINEX 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:
230 152

Subroutine Called:
LVFDEX
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: USRINI and USRIN2. USRINI precedes the actual insertion, but it is skipped if LVIN1 is .FALSE. (default). The insertion may be skipped if LVIN4 is set within USRINI to .FALSE. (default is .TRUE.). USRIN2 follows the insertion, but it is skipped if LVIN2 is .FALSE. (default). If USRINI is called, LVIN2 may be modified by LVIN3. USRINI 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 ≤ n ≤ 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

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

LVTYPE(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 Berkowitz for limitations on Integer data
- Hollerith data; see Berkowitz for limitations on Hollerith data

If LVTYPE(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.
                     Request new continuant
  = -1
  = -2  Continuant unspecified (default)
  = -3  Current continuant (if current page = re-
        quested page)

0 < n < 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
= 1 Function did exist prior to this
insertion
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 LTNKL have been initialized in a DEFINEn statement:
1) Add random number valuei to the (null) list:
   G NODE1 LINK1 valuei
2) Add integer I to the end of the list
   G NODE1 LINKI "I"
3) Place valuei in the third location from the bottom of the list.
   G NODE1 LINK1 .-3 valuei
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 valuei (for the current page) and place the triple on page 5, continuant 0.

```
NODEI = 5
LVREQP(2) = 0
VALUEI = 0
G NODEI LINK1 VALUEI
PRINT NODEI, VALUEI
```

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 914_10
```
Subroutines Called:

<table>
<thead>
<tr>
<th>LVSTAC</th>
<th>LVLFSH</th>
</tr>
</thead>
<tbody>
<tr>
<td>LVPOP</td>
<td>LVRTSH</td>
</tr>
<tr>
<td>LVNPAG</td>
<td>LVERR</td>
</tr>
<tr>
<td>LVGRN</td>
<td>LVEXCH</td>
</tr>
<tr>
<td>LVNCON</td>
<td>LVFDEX</td>
</tr>
<tr>
<td>LVREOR</td>
<td>LVFINH</td>
</tr>
<tr>
<td>LVOVER</td>
<td></td>
</tr>
</tbody>
</table>
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 USRD1 and USRD2, 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 USRDLL)

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

41
Example 2.
Delete the \textsuperscript{I}th value on an MVL, continuant is not known.
\texttt{G \ NODEI+LINK1=.I}

\textbf{Program Length:}
\begin{tabular}{ll}
762 & 498 \\
8 & 10
\end{tabular}

\textbf{Subroutines Called:}
\begin{tabular}{ll}
LVSTAC & LVDLET \\
LVPOP & USRDL1 \\
LVERR & USRDL2 \\
LVFDEX
\end{tabular}
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 IPI identifiers from COMMON /LVSUSER/ 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/)
LVMOGE 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
   > 0 Output all continuants of page n
LVLMN 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 LVLMN some of the pertinent GIRS variables found in the labeled commons
   = 1 Output to LVLMN all the pertinent GIRS variables found in the labeled commons
Program Length:

553

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

Subroutines Called:
LVERR LVDRRD
LVPAGR LVDRWR
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 GIRL 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 vari, 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 314 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:

   \[ 4 \times (64 + (\text{cont. size} \times \text{no. of conts. in buffer})) \]

   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.)

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 2513 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 4th 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):

```
.R PREP
ALL REAL VARIABLES MUST BE DECLARED
ERRORS ARE FLAGGED BY ****ERROR
PLEASE ENTER FILE NAMES IN COMMAND STRING FORM
*USER=USER
or, if a list file is also desired:
*USER,USER=USER
.R FORTRAN
.USER=USER/W
*C (control C)
```

DIRECT USE OF GIRLS SUBROUTINES

Calls to GIRLS 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,RKL:FINDEP,DELEOP/E/C
NSRTOP/0:1/C
DEXOP/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)
*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 $13653_{10}$ to $8332_{10}$ 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>
<thead>
<tr>
<th>OVERLAY REGION</th>
<th>OCTAL</th>
<th>DECIMAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root Segment</td>
<td>4531</td>
<td>2393</td>
</tr>
<tr>
<td>1</td>
<td>5434</td>
<td>2844</td>
</tr>
<tr>
<td>2</td>
<td>345</td>
<td>229</td>
</tr>
<tr>
<td>3</td>
<td>4123</td>
<td>2131</td>
</tr>
<tr>
<td>4</td>
<td>412</td>
<td>266</td>
</tr>
<tr>
<td>5</td>
<td>725</td>
<td>469</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>20214</strong></td>
<td><strong>8332</strong></td>
</tr>
</tbody>
</table>
## TABLE 5 - THE OVERLAY STRUCTURE

<table>
<thead>
<tr>
<th>Overlay Region</th>
<th>Subroutines Listed by Segment (Size in Decimal Words)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root Segment</td>
<td>LVFDEX, LVFIND, LVFNV, LVBOTM, LVEXCH, LVDLET (2393)</td>
</tr>
<tr>
<td>1</td>
<td>LVINEX, LVDLEX (481), LVNSRT, LVUPDT (2844)</td>
</tr>
<tr>
<td>2</td>
<td>LVMSA, LVDRCT, LVOPEN, LVRPLC (229, 196, 157)</td>
</tr>
<tr>
<td>3</td>
<td>LVSETP, LVFECH, LVREOR, LVGRN, LVDUMP, LVOVER, LVNFAG, LVDUMP, LVWRIT, LVINCL (2131, 1578)</td>
</tr>
<tr>
<td>4</td>
<td>LVPAGR, LVDRRD, LVPAGW, LVRWWR (266, 99)</td>
</tr>
<tr>
<td>5</td>
<td>LVALUE, LVSUM, LVESTAC, LVERR, LVSUM (469, 21)</td>
</tr>
<tr>
<td></td>
<td>LVPPO, LVRTRN [skeleton version], LVLFSH, LVRISH (348)</td>
</tr>
</tbody>
</table>
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:

   \[ \text{length} = 64 \times ((\text{LVNCOR}/64) + 1) + \text{LVNCOR} \times (\text{LVHDRS} + \text{LVVSZE}) \]

   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
LVDLEX 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)
LVDump Create either an ASCII dump of GIRS continuants
LVWRIT or a binary file which contains the graph and
LVCLSO 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:

<table>
<thead>
<tr>
<th>Overlay region</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2844</td>
</tr>
<tr>
<td>2</td>
<td>229</td>
</tr>
<tr>
<td>3</td>
<td>2131</td>
</tr>
<tr>
<td>4</td>
<td>266</td>
</tr>
<tr>
<td>5</td>
<td>469</td>
</tr>
</tbody>
</table>
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 database, 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 database and also when an attempt is made to retrieve information from the database.

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:

"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 database. 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 database 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  LVVT, LVVINC, LVNDXN, LVVALS(10), LVTYPE(10),
2  LVRSIF, LVLNSF, LVNSIF, LVNTYP
COMMON /LVVSEQ/ LVSIZE, LVSEQ1, LVSEQ2, SEQSC(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 /LVCNRT/ LVVGS, LVCTRL, LVCTR1, LVSTV, LVNFRE, LVFREE, LVDREG,
2  LVVMSA, LVPGLC, LVCNT
COMMON /LVBUFR/ LVVSE, LVNCH, LVOLCH, LVCMPL, LVPGHS, LVBFZ,
1  LVDRS, LVNCR, LVHDRS, LVMSAD,
2  LVFSZ, LVBHKS, LVDBK, LVPGH4
COMMON /LVREGS/ LVCPG(4), LVREQP(4), LVLPVG(4), LVMSAR
1  LVHRPG, LVVMSA, LVHAPG(2), LVRCNT, LVUCNT, LVDRPG,
2  LVDIR, LVOTLC, LVOTDR(256),
3  LVVWSB(4*continuant size)
COMMON /LVPRAM/ LVBFILC, LVNLTH, LVVER, LVENER, LVBNRY, LVBCD,
1  LVMODE, LVPR, LVUN
COMMON /LVUN/ LVRNTP, LVCORE
COMMON /LVSTAK/ LVLEVEL, LVNVAR, LVSTAK(140)
COMMON /LVMAK/ LVWRIT, LVUSE, LVWNCH, LVMSK3, LVMSIF, LVMSIF
COMMON /LVSWIT/ LVSTP,LVSNGL,LVNXTR,LVIN1,LVIN2,LVFD1,
1     LVFD2,LVDL1,LVDL2,LVDM3,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     FLAG67,FLAG8,FLAG9,FLAG10,FLAG11,FLAG12,FLAG13,
2     FLAG14,FLAG15
COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
1     USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
COMMON /LVFDNI/ IADD,THIS,LSTHED,LOC,LAST,LSTLC
COMMON /LVFDNI/ 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
   LVFCH
   LVGRN()
   LVPAGW
   LVDRWR
   LVMSA()
   LVPAGR()
   LVFIN
   LVNSRT

SUBROUTINE LVMSA(CONNUM)
   LVERR()
   LVDRRD()

SUBROUTINE LVGRN(NODE)
   LVFSH(,)
   LVEXCH
   LVSTAC
   LVERR()

SUBROUTINE LVGLOS
   LVERR()
   LVPAGW
   LVDRWR

SUBROUTINE LVDRRD(CHAN)
   LVERR()

SUBROUTINE LVDRWR
   LVERR()

SUBROUTINE LVPAGR(CHAN)
   LVERR()

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
LVPGW
LVRPLC
LVSUM
SUBROUTINE LVNCON
LVMSA()
LVOPEN
LVSETP
LVPGW
LVPGR()
LVRCPL
LVSUM

SUBROUTINE LVERR(DUMP)

SUBROUTINE LVOPEN
LVALUE
LVPGW

SUBROUTINE LVRCPL
LVSTAC
LVFIND
LVDLET
LVNSRT
LVPOP

SUBROUTINE LVSUM

SUBROUTINE LVALUE
LVDUMP()

SUBROUTINE LVFECH
LVERR()
LVPGR()
LVDRRD()
LVDWR
LVMSA()
LVPGW

SUBROUTINE LVDUMP(DUMP)
LVERR()
LVWRIT(,)
LVEDCH
LVPGW
LVCLS

SUBROUTINE LVWRIT(NBIAS,NUMBLK)
APPENDIX C

SUBROUTINE LISTINGS

C
C
C
C
8001 SUBROUTINE LVFDEX INDEX, INDXAD, KF'NC, XAR(,SA&VCON)
8002 IMPLICIT INTEGER(A-Z)
8003 LOGICAL*
8004 COMMON /lVARGS/
8005 COMMON /IVREGS/
8006 COMMON /1 VMASK/ MWRITE, NOTIISD, NEWCONFT GMSX MASKSF, MASXPF
8007 COMMON /IVFIAG/ FIOMSK, FI IMSK, FI2MSK, FI3MISK, FI4MLSK, FIG67,
8008 COMMON /IVCRNT/ REGASP, CTRI PT, CTR I, LEASTV, NTFRE E, FREE, DREGSP,
8009 COMMON /IVDIR/ PAGSZ E, NCHAM, OI CHAN, CMPAND, PAGHDR, BUFSE, DIRSZE,
8010 COMMON /IVHDI/ THSMA, REGAS, PAGENO, CONTNO, INSDEL,
8011 COMMON /IVSWIT/ SETUP, SNGI BK, NYTRAN, INISTR, IN2STR, FDISTR, FD2STR,
8012 COMMON /IVPRAN/ BIROC, LENGTH, IERR, ERMNUM, BINARY, BCP, MODE, PAGES,
8013 COMMON /IVVTR1/ NODSPC'I)
8014 COMMON /IVVTR2/ LSTSPC'I)
8015 COMMON /IVVTR3/ LIKSPC'I)
8016 COMMON /IVVTR4/ FISPC'I)

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
C UNTIL 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 FDISTR IS .TRUE., USER STRATEGY ROUTINE USRFD1 PRECEDES
C REtrieval ACTION AND CONTINUES IF FDNP1 IS .TRUE.
C USRFD2 IS CALLED AFTER THE RETRIEVAL IF FD2STR IS SET TO .TRUE.
C
0016 XXX=1000
0017 IF ((FISPC(I+REGASP).OR.FI 3MSK).NE.FI 3MSK) XXX=XXX*XXX
0019 XXX=XXX*XXX
0020 USE/LINKSPC(DIRSZE+ISECT)
0021 XXX=XXX*XXX
0022 XXX=XXX*XXX

65
C CALL USER'S FIRST RETRIEVAL STRATEGY ROUTINE?
0023 IF(FPISTR .EQ. .FALSE.) GO TO 100
C TO PREVENT RECURSION, INHIBIT FURTHER CALLS TO USER STRATEGY ROUTINES
0025 FP1TMP = FP1ISTR
0026 FP2TMP = FP2ISTR
0027 FP1ISTR = .FALSE.
0028 FP2ISTR = .FALSE.
0029 DL1TMP = DL1ISTR
0030 DL2TMP = DL2ISTR
0031 DL1ISTR = .FALSE.
0032 DL2ISTR = .FALSE.
0033 IN1TMP = IN1ISTR
0034 IN2TMP = IN2ISTR
0035 IN1ISTR = .FALSE.
0036 IN2ISTR = .FALSE.
C SET UP FOR FIRST USER ROUTINE
0037 CALL LVSTAC
0038 CALL USRFPI
0039 CALL LVPOP
0040 FDISTR = FDITMP
0041 FP2STR = FD2TMP
0042 DLISTR = DLITMP
0043 DL2STR = DL2TMP
0044 INISTR = INITMP
0045 IN2STR = IN2TMP
C PROCEED WITH RETRIEVAL?
0046 IF(FINDPI .EQ. .FALSE.) GO TO 600
C REQPAG(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 .LT. 2*SUFSIZE) RETURN
0050 REQPAG(1) = LVRTSH(IARG .AND. MASKPF, SUFSIZE)
0051 SRCSUF = IARG .AND. MASKSF
0052 REQPAG(2) = LVRTSH(IFUNC .AND. MASKPF, SUFSIZE)
0053 LNKSUF = IFUNC .AND. MASKSF
C IS SAVED INDEX OPTION ON?
0056 IF(NSKIP .EQ. 1) GO TO 150
0058 REQPAG(2) = SAVCON
C REQPAG(2) IS SET AT THE END OF LVFINP 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 150 RECON = REQPAG(2)
0059 IF(REQPAG(2) .EQ. -2) REQPAG(2) = 0
C *** BRING THE REQUESTED PAGE, CONTINUANT INTO CORE.
C MAKE IT THE CURRENT PAGE, CONTINUANT.
0062 ITESTR = -1
0063 200 CALL LVEXCH
C HAVE ALL CONTINUANTS OF REQ(PAGE) BEEN EXAMINED
   IF(MSARET .IE. 0) GO TO 600
C DESIRED PAGE, CONTINUANT IS NOW IN PLACE.
C ASSUME LIST DOES NOT CONTINUE BEYOND PRESENT CONTINUANT
   LSTCON = .FALSE.
C SEARCH FOR FUNCTION HEAD.
   CALL LVFIND
C FLAG CONTINUANT AS USED
   FIGSPC(CTRIPT+HDRFIG) = FIGSPC(CTRIPT+HDRFIG) .AND. .NOT. NOTUSD
   LNKSPC(CTRIPT+USECT) = LNKSPC(CTRIPT+USECT) + 1
C HAS THE FUNCTION HEAD BEEN FOUND?
   IF(ITESTR .GT. 0) GO TO 600
C IF THE CONTINUANT IS NOT SPECIFIED, EXAMINE NEXT CONTINUANT.
   IF(REQCON .NE. -2) GO TO 600
   REQPAG(2) = REQPAG(2) + 1
   GO TO 200
C FUNCTION HEAD FOUND
C SEARCH FROM TOP OR BOTTOM OF LIST?
   IF(IPOS) 500,600,410
   CALL LVFNV(INPEX,INPXAD,KFVNC,KARG,SAVCON)
C FLAG CONTINUANT AS USED
   FIGSPC(CTRIPT+HDRFIG) = FIGSPC(CTRIPT+HDRFIG) .AND. .NOT. NOTUSD
   LNKSPC(CTRIPT+USECT) = LNKSPC(CTRIPT+USECT) + 1
C SUCCESSFUL RETRIEVAL?
   IF(ITESTR .GT. 0) GO TO 600
C DOES THE LIST EXTEND TO ANOTHER CONTINUANT?
   IF(LSTCON .EQ. .FALSE.) GO TO 600
C UPDATE REQUESTED CONTINUANT AND BRING INTO THE BUFFER
   REQPAG(2) = REQPAG(2) + 1
   CALL LVFXCH
C HAVE ALL CONTINUANTS OF REQ(PAGE) BEEN EXAMINED?
   IF(MSARET .IE. 0) GO TO 600
   GO TO 400
C SEARCH FROM THE BOTTOM OF THE LIST
C BRING IN CONTINUANT CONTAINING LAST PORTION OF MVI
   CALL LVBOTM
   GO TO 530
C0094  520  CALL LVFIND
C C DOES A PORTION OF THE CORRECT LIST RESIDE ON THIS CONTINUANT?
0095 IF(IESTR .LT. 0) GO TO 550
C C BEGIN SEARCH UP THE LIST
0097 530  CALL LVFNVINDEX, INDEXAD, KFUNC, KARG, SAVCON)
C C FLAG CONTINUANT AS USED
0098 FLSPC(CTRLP+HDRFLG) = FLSPC(CTRLP+HDRFLG) .AND. .NOT. NOTUSP
0099 LNKSPC(CTRLP+USELECT) = LNKSPC(CTRLP+USELECT) + 1
C C SUCCESSFUL RETRIEVAL?
0100 IF(IESTR .GT. 0) GO TO 600
C C DOES THE LIST EXTEND TO ANOTHER CONTINUANT?
0102 IF(LSTCON .EQ. .FALSE.) GO TO 600
C C UPDATE REQUESTED CONTINUANT
0104 550  REQPAG(2) = REQPAG(2) - 1
C C HAVE ALL CONTINUANTS OF REQ(PAGE) BEEN EXAMINED?
0105 IF(REQPAG(2) .EQ. 0) GO TO 600
C C BRING REQ(P.C) INTO THE BUFFER
0107 CALL LVEXCH
0108 GO TO 520
C C CALL SECOND USER RETRIEVAL STRATEGY ROUTINE?
0109 600  IF(FP2STR .EQ. .FALSE.) GO TO 700
0111 FDITMP = FDISTR
0112 FD2TMP = FD2STR
0113 FD1STR = .FALSE.
0114 FD2STR = .FALSE.
0115 DL1TMP = DL1STR
0116 DL2TMP = DL2STR
0117 DL1STR = .FALSE.
0118 DL2STR = .FALSE.
0119 I1TMP = I1ISTR
0120 I2TMP = I2ISTR
0121 I1ISTR = .FALSE.
0122 I2ISTR = .FALSE.
0123 CALL LVSTAC
0124 CALL USRFP2
0125 CALL LVPOP
0126 FDISTR = FDITMP
0127 FD2STR = FD2TMP
0128 DL1STR = DL1TMP
0129 DL2STR = DL2TMP
0130 I1ISTR = I1TMP
0131 I2ISTR = I2TMP
C C RESET 'REQUESTED CONTINUANT' DEFAULT TO 'ANY'
0132 700  REQPAG(2) = -2
C C RESET TO DEFAULT VALUES
0133 IPOS = 1
0134 ITYP = 3
0135 RETURN
0136 END
SUBROUTINE LVFINP
IMPLICIT INTEGER(A-Z)
LOGICAL ISNO!
COMMON /IVARGS/ IFUNC, IARG, IPOS, ITP,IVAL,NVAL,NSKIP,ITESTR,
COMMON /IVREGS/ CURPA(4),REQPA(4),INSTR,FP1STR,FD2STR,DLISTR,
 COMMON /IVFI.AG/ FIONSK,FI INS,FIZMS,FI.3MS,FI.6NSK,F1.AG67,
 COMMON /IVVTR2/ VNOREQ,CTRL,CTRL1,LEFTSTY,NTFREE,FREE,DEFSEQ,
 COMMON /IVVTR3/ LSKDRG,FL,SKDRG,
 COMMON /IVVTR4/ FI.4STG,L}
 COMMON /IVPRAX/ BUFSIZE,LENGTH,IERRERRNITM,BINARY,BCP,NOPE,PAGES,
 COMMON/IVADDR/ IADD,THIS,ISTHED,LOC,LSTED,
 COMMON /IVVTR1/ NODSCO((1))
IADD = (RELATIVE) COMPUTED FUNCTION ADDRESS
THIS = (RELATIVE) LOCATION OF FUNCTION ON CONFLICT LIST
LOC = (RELATIVE) LOCATION OF RETRIEVED VALUE
LSTED = (RELATIVE) LOCATION OF RETRIEVED VALUE
* 0, NO LIST IS FOUND
* 1, RETRIEVAL IS SUCCESSFUL (IVAL = RETURNED VALUE)
* -1, RETRIEVAL IS FAILURE (IVAL = SOURCE NODE)
PAUSE 'IN LVFINP'
ITESTR = 1
IADD = SRCSUF + LNSUF
IF (IADD .GT. PAGES) IADD = IADD-PAGES
IF (IADD .LE. PAGES) GO TO 2
C IFUN OR IARG ARE INCORRECT, STOP
FORMAT(//,'****ERROR**** LINK ',15,' OR SOURCE NODE ',15,' ARE
UNDEFINED',//)
ERRNUM = 40
DUMP = 0
CALL LVERR(DUMP)
STOP

LSTHED = 0
THIS = IADD
IF(FIGSPC(CTRL1 + THIS) .AND. F1MSK) .EQ. 0) GO TO 99
C
SEARCH CONFLICT LIST FOR KEY (IFUNC OR LINK)
IF(NOISPSC(CTRL1 + THIS) .EQ. IFUNC) GO TO 4
LAST = THIS
LSTHED = LNKSPC(CTRL1 + THIS)
IF(FIGSPC(CTRL1 + THIS) .AND. F1MSK) .NE. 0) GO TO 99
GO TO 1
C
THE FUNCTION HAS BEEN FOUND.
C TEST FOR SINGLE VALUE LIST (SVL) OR MULTIVALUED LIST (MVL).
IF(FIGSPC(CTRL1 + THIS) .AND. F1MSK) .NE. 0) GO TO 14
C
SINGLE VALUED LIST.
LSTHED = -1
LOC = THIS
IVAL = LSTSPC(CTRL1 + LOC)
RETURN
C
MULTIVALUED LIST. OBTAIN FIRST VALUE.
LSTHED = LSTSPC(CTRL1 + THIS)
LOC = LSTHED
IVAL = NOISPSC(CTRL1 + LOC)
LASTLC = LNKSPC(CTRL1 + LSTHED)
RETURN
C
FUNCTION IS NOT ON THIS CONTINUANT
ITESTR = -1
IVAL = IARG
RETURN
END
SUBROUTINE LVFINP(INDEX, INDXAD, KFUNC, KARG, SAVCON)
IMPLICIT INTEGER(A-Z)
LOGICAL *1 SNGBK, SETUP, MVRAM, INSTR, IN2STR, FP1STR, FP2STR, DL1STR.
1 COMMON /IVARGS/ IFUTNC, IARG, IPOS, IYP, IVAL, IVAL, MSA., ITESTR.
2 COMMON /IVREGS/ CUTRPAG, RFOPAG, LSTVPG, MS., ARET, HREQPG, LNKSTF.
4 COMMON /IVVTR/ NFIAG4/ 177767/
C LVFINP Must be called immediately prior to the call to this routine
C Input is expected thru COMMONGS LVARGS, LVFINP, 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
D Does the function exist?
010 IF (ITESTR .LT. 0) GO TO 700
020 IF (LSTTHEQ .GT. 0) GO TO 100
C Svi. Does function qualify?
022 IF (ABSPOS .GT. 1) GO TO 699
024 IF (IYP .EQ. 3) GO TO 700
026 ISTYP = (FIGSPC(CTRIL + LOC) + LSTSPC(1)) + LSTSPC(1)
027 IF (IYP .EQ. 3) ISTYP = 2
029 IF (ISYP .EQ. 1) IYP + 10
031 GO TO 700
C C MVI - First value has already been found by LVFINP
032 IF (IPPOS .EQ. 1) GO TO 500
C *** BEGIN SEARCH
C IF THE SAVED INDEX FACILITY IS NOT TO BE USED. GO TO 200
0034 120 IF(WSHICP.EQ. 1) GO TO 200
0036 IF(INDX.EQ. 0) GO TO 200
C
C SAVED INDEX CAN'T BE USED IF IMMEDIATE PAST HISTORY = . EXED
C INSERTION OR DELETION
0038 IF((FSHPC(Control) .AND. F10MSK) .NE. 0) GO TO 200
C
C SAVED INDEX CAN'T BE USED IF SOURCE NODE OR LINK HAVE BEEN CHANGED
0040 IF((E1FUNC .NE. 1F1INC) .OR. (LARG .NE. 1ARG)) GO TO 200
C
C SAVED INDEX CAN'T BE USED IF DIRECTION OF SEARCH HAS SWITCHED
0042 IF((IP0S.INEQ. INDEX).I.E. 0) GO TO 200
0044 IF(INPXAD) GO TO 290
C SAVED INDEX CAN'T BE USED IF VALUE AT SAVED INDEX HAS BEEN MOVED
0045 IF(INPXAD .AND. F10MSK).NE. 0) GO TO 200
C SAVED INDEX CAN'T BE USED IF VALUE AT SAVED INDEX HAS BEEN REMOVED
0046 IF(INPXAD .AND. F10MSK).EQ. 0) GO TO 200
C IS SEARCH FROM BEGINNING FASTER THAN FROM SAVED INDEX ?
0047 IFABS(INDEX))
0050 IF(ABSPOS .LT. 2) GO TO 200
0052 IF(ABSPOS .NE. 0) GO TO 200
C SAVED INDEX CAN BE USED, BEGIN SEARCH AT INPXAD.
0054 IF(LENGTH) .LT. 000, 450, 170
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.
0055 LENGTH = INDEX - IP0S
0056 ABSPOS = IABS(LENGTH)
0057 IF(LENGTH) .LT. 000, 450, 170
C COUNT UP FROM INPXAD
C 0058 170 ITOP = 0
0059 GO TO 420
C DO NOT USE SAVED INDEX. START FROM THE BEGINNING OR END OF LIST
C 0060 200 FSHPD(Control) * THIS) = FSHPD(Control) * THIS) .AND. NFLAG
0061 IF(IP0S) 400, 450, 320
C COUNT DOWN
C 0062 300 LASTLC = LOC
0063 LOC = LSHPD(Control) + LOC
0064 IF((LSHPD(Control) .AND. F10MSK).NE. 0) GO TO 600
0066 320 IF(IYTP .EQ. 3) GO TO 320
0067 IYTP = (FSHPD(Control) * LOC) .AND. F1067)
0069 IF(IYTP .EQ. 3) IYTP = 2
0071 IF(IYTP .NE. IYTP) GO TO 320

72
COUNT = COUNT + 1
IF (COUNT .NE. ABSPOS) GO TO 300
GO TO 450

COUNT UP FROM THE BOTTOM OF THE LIST

ITOP = 1
LOC = LKSPC((CTRI1 + LOC)
IF (ITOP .EQ. 1) GO TO 430
IF (FLUSK) .NE. 0)
1 GO TO 650

ITOP = 0

LOC = LKSPC((CTRI1 + LOC)
IF (ITOP .EQ. 3) GO TO 440

ISTYP = (FIGSPC(CTRI1 + LOC) .AND. FI067)
IF (ISTYP .EQ. 3) IVAL = NOISPSC(CTRL1 + LOC)

SAVE INDEX PARAMETERS AFTER SUCCESSFUL RETRIEVAL

IF (NSKIP .EQ. 1) GO TO 700
KARG = IARG
KFUNC = IFUNC
INDEX = LOC
INDEX = IPNS
SAVCON = CURPAG(2)
GO TO 700

POSSIBLE FAILURE. DOES MVI EXTEND FORWARD TO ANOTHER CONTINUANT
IF (FIGSPC(CTRI1 + LASTLC) .AND. FLG11) .EQ. 0) GO TO 699
LSTCON = .TRUE.
GO TO 699

POSSIBLE FAILURE. DOES MVI EXTEND BACKWARD TO ANOTHER CONTINUANT
IF (FIGSPC(THIS) .AND. FLG10) .EQ. 0) GO TO 699

LSTCON = .TRUE.

FAILURE EXIT
ITESTR = -1
IF (NSKIP .EQ. 0) INDEX = 0
IVAL = IARG

SUCCESS EXIT. SET DEFAULTS.
ITYP = 3
RETURN
END
SUBROUTINE LVBOFN

IMPLICIT INTEGER(A-Z)

COMMON /IVARGS/ IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, IESTR,
1 INCLUD, INRDX, INVALS(10), ITYP(10), SRCSTRF,
2 LNSUBF, SNKSUF, INSTYP

COMMON /IVREGS/ CIURPAG, RREPAG, ISRVPAG, SNK, SSRCIF, 2 LNKSUTF, SNKSUTF, INSTYP

COMMON /IVARGS/ IFINC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, IESTR,
1 INCLUD, INRDX, INVALS(10), ITYP(10), SRCSTRF,
2 LNSUBF, SNKSUF, INSTYP

COMMON /1 VNASK/ IWRITE, NOTUSP, NFWCON, 2 FILLMSK, MASKSF, ASKPSF

COMMON /IVFAG/ FIOMSK, FI2MISK, FI3MSK, FI4MSK, FI5MSK, FI6MSK, FI7MSK, FI8MSK, FI9MSK, FI10MSK, FI11MSK, FI12MSK, FI13MSK, FI14MSK, FI15MSK, FI16MSK, FI17MSK, FI18MSK

COMMON /1 VPART/ IADD, THIS, LSTHED, LOC, LAST, LASTLC

COMMON /1 VPRAM/ BITLOC, LENGTH, IERR, ERRNUM, BINARY, BCP, MODE, PAGES,
1 LUN

COMMON /IVADDR/ IADD, THIS, LSTHED, LOC, LAST, LASTLC

COMMON /IVPELIN/ NITMRET, BAKCON

COMMON /IVVTR1/ NOPSPC(1)
1 /IVVTR3/ LSTSPC(1)
2 /IVVTR2/ LNKSPC(1)
3 /IVVTR4/ Bapkspc(1)

THIS ROUTINE BRINGS INTO THE BUFFER THE LAST CONTINUANT OF A PAGE WHICH
CONTAINS A PORTION OF THE REQUESTED LIST.

ASSUME THAT THE BUFFER CONTAINS THE CONTINUANT WHICH HOLDS THE FIRST
PORTION OF THE MWI.

PAUSE 'IN LVBOFN'

TMPREQ = CURTAG(2)

C SVI ?

100 IF(LSTHED .GT. 0) GO TO 120

120 LASTLC = THIS

GO TO 140

C GET FIRST 'VALUE' ON MWI

120 ILSLOC = LSTSPC(ICTRL + 1)

C GET LAST 'VALUE' ON MWI

LASTLC = LNKSPC(CTRL1 + 1)

C DOES THE LIST END ON THIS CONTINUANT ?
I0025 IF(FIGSPC(CTRL1 + LASTLC).AND. FLAG11) .EQ. 0) RETURN
I0027 LSTCON = .TRUE.
I0028 C EXAMINE NEXT (SEQUENTIAL) CONTINUANT FOR A PORTION OF THE MVI
I0029 200 REQPAG(2) = REQPAG(2) + 1
I0029 CALL LVECH
I0030 C ERROR IF SET OF CONTINUANTS IS EXHAUSTED
I0030 IF(MSARET .GT. 0) GO TO 250
I0030 C NO ERROR IF SEARCH ORIGINATED FROM LVIDEX
I0032 IF(BAKCON .EQ. .FALSE.) GO TO 220
I0032 REQPAG(2) = TMPREQ
I0033 RETURN
I0035 C 220 ERRNUM = 42
I0037 DUMP = 0
I0038 CALL LVERR(DUMP)
I0039 STOP
I0040 C DOES THIS CONTINUANT CONTAIN A PORTION OF THE MVI?
I0040 250 CALL LVFIND
I0041 IF(ITESTR .LT. 0) GO TO 200
I0043 TMPREQ = REQPAG(2)
I0044 GO TO 100
I0045 END
SUBROUTINE LVINEX
IMPLICIT INTEGER(A-Z)
LOGICAL*1 SNGIBK, SFTUP, NY(TRAN, INISTR, IN2STR, FPISTR, FP2STR, DL1STR,
   DL2STR, DUMPF1, CURRENT, FIND1, DLET1, NSRTF1, FP1TMP,
   DL2TMP, IN2TMP, FP2TMP, INSIDE, FULL, REORG, LSTCON, NYCON,
   RPLACE
COMMON /VAROS/ IFUNC, IARG, IPFS, IYP, IVAL, NVAL, NSKIP, ITESTR.
   INCLUDE, INDXON, IVALS(10), ITYP(10), SRCSUF.
   LNSUF, SNKSUF, INSTYP
COMMON /VREGS/ CTRPAG(4), REQPAG(4), LSTVPG(4), MSARET,
   HREQP, NXTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
   DIRCNT, OUTFLOC, OUTFDIR(256), RWBUF(1)
COMMON /VMASK/ MWT, NOTUSP, NRWCON, FlGMSK, MASKSF, *MASKPF
COMMON /VFREG/ REGASP, CTRIT, CTR1, LEASTV, NTFREE, FREE, DRPGSP,
   MSA, PAGLOC, CURREN
COMMON /VBPFR/ PAGSZE, NCHOP, OCHAN, CMPAND, PAGHDR, BUFZSE, DIRSZE,
   INCORE, HDRSZE, MSADIR, SPSZES, BJSZES, DIRBK, PGMD4
COMMON /VHDRVI/ THSMA, REGAS, PAGENO, CONTRO, INSDEL,
   USECT, HDRFLG, READVI, OI, DNDB, DOOPEN, KNOB, DROWH
COMMON /VSNIT/ SETUP, SNGL BK, XVT, INISTR, IN2STR, FPISTR, FP2STR,
   DL1STR, DL2STR, IN2TMP, FP2TMP, DL2TMP, DUMPF1,
   DLET1, NSRTF1
COMMON /VPFRMS/ BUFILOC, LENGTH, IERR, ERRNUM, BINARY, BCD, MODE, PAGES,
   LUN
COMMON /VADDH/ IADD, TH15, LSTHED, LOC, LAST, LASTLC
COMMON /VAPND/ COUNT, ABSPOS, LSTCON
COMMON /VINS1/ REORG, FULL, RPLACE
COMMON /VVT1/ NOSPC(1)
   /VVT2/ LSTSPC(1)
   /VVT3/ LNSPC(1)
   /VVT4/ *LNSPC(1)
DATA INSIDE .FALSE./
THE INSERT EXECUTIVE ROUTINE COMPLETES THE TRIPE IF NECESSARY AND
OBTAINS THE CORRECT P.C FOR SUBROUTINE LVNSRT TO OPERATE ON.
IS LVNSRT BEING CALLED FROM AN INSERT STRATEGY ROUTINE?
IF INSIDE .EQ. TRUE.) GO TO 100
TO PREVENT RECURSION. SAVE THE FIND STRATEGY FLAGS AND TURN THEM OFF
FD1TMP = FD1STR
FD2TMP = FD2STR
FD1STR = .FALSE.
FD2STR = .FALSE.
CALL USER'S FIRST INSERT STRATEGY ROUTINE?
IF(INISTR .EQ. .FALSE.) GO TO 100
TO PREVENT RECURSION, INIBIT CALLS TO ALL USER STRATEGY ROUTINES
INITMP = INISTR
IN2TMP = IN2STR
INISTR = .FALSE.
IN2STR = .FALSE.
DLITMP = DLISTR
DL2TMP = DL2STR
DLISTR = .FALSE.
DL2STR = .FALSE.

C SET UP FOR FIRST USER ROUTINE
CALL LVSTAC
INSIDE = .TRUE.
CALL USRINI
INSIDE = .FALSE.
CALL LVPOP
INISTR = INITMP
IN2STR = IN2TMP
DLISTR = DLITMP
DL2STR = DL2TMP

C PROCEED WITH INSERTION ?
IF(NSRTFI.EQ..FALSE.) GO TO 1000

C *** ENSURE THAT THE TRIPLE IS COMPLETELY DEFINED ***
C *** BRING IN REQ(P,C), DEFINE AS CURRENT(P,C) ***
TEST SOURCE NODE
IARG = -1 PLACE ON NEW PAGE AND DEFINE
* 2**SUFSZE, ALREADY DEFINED, SEPARATE PREFIX AND SUFFIX
* N<2**SUFSZE, PLACE ON PAGE N AND DEFINE

NYTCON = .FALSE.
RECON = REQPAG(2)
IF(IARG .GE. 2**SUFSZE) GO TO 140
IF((IARG) 110,120,130
IARG = LVIFSH(CURPAG(i),SUFSZE) OR. SRCSUF
GO TO 200

C PLACE ON NEW PAGE (CONT = 0)
CALL LVNPAG

C PLACE ON CURRENT PAGE (AND CONT) AND DEFINE SUFFIX
CALL LVGRN(SRCSUF);

C RECONSTRUCT IARG
IARG = LVIFSH(CURPAG(1),SUFSZE) OR. SRCSUF
GO TO 200

A SPECIFIC PAGE IS REQUESTED
IS ONLY THE SUFIX DEFINED ? (IF SO, IT IS A PAGE REQUEST W/O SUF)
IF(IARG .GE. 2**SUFSZE) GO TO 140
REQPAG(1) = IARG
GO TO 145
REQPAG(1) = LVRTSH(IARG,SUFSZE)

C IMPROPER PAGE REQUEST ?
IF(REQPAG(1) .EQ. HACPAG(1)) GO TO 145
ERRNUM = 60
DUMP = 0
CALL LVFRR(DUMP)
C STOP
C TEST "REQUESTED CONTINUANT"
C REPQAG(2) = 0, CONTINUANT SPECIFIED
C -1, NEW CONTINUANT
C -2, ANY CONTINUANT
C -3, CURRENT CONTINUANT IF CURRENT PAGE = REQUESTED PAGE
0065 IF(REPQAG(2) .GE. 0) GO TO 155
0066 IF(REPQAG(2) + 2) 159, 180, 178
C C CURRENT CONTINUANT OF REQUESTED PAGE IF ALSO CURRENT PAGE
0068 REPQAG(2) = CURPAG(2)
0069 IF(REPQAG(1) .NE. CURPAG(1)) REPQAG(2) = 0
C C BRING IN PROPER CONTINUANT
0071 CALL LVEXC4
0072 IF(IARG .GE. 2*SUFSIZE) GO TO 200
0074 GO TO 120
C C CONTINUANT NOT SPECIFIED, SET TO ZERO
0075 REPQAG(2) = 0
0076 GO TO 155
C C CREATE NEW CONTINUANT
0077 CALL LVNCON
0078 GO TO 157
C C TEST IFUNC AND RECONSTRUCT IF NECESSARY
0079 IF(IFUNC .GE. 2*SUFSIZE) GO TO 300
0081 TEMPAG = REPQAG(1)
0082 REPQAG(1) = IFUNC
0083 IF(IFUNC .EQ. 0) REPQAG(1) = CURPAG(1)
0085 REPQAG(3) = REPQAG(1)
0086 CALL LVGRN+LNKSUF
0087 IFUNC = LVFSH(REPQAG(3),SUFSIZE), OR. LNKSUF
0088 REPQAG(1) = TEMPAG
C C TEST THE SINK NODE AND RECONSTRUCT IF NECESSARY
C RANDOM NUMBER
0089 IF(TYPEP .NE. 0) GO TO 400
0091 IF(ILNALS(1) .GE. 2*SUFSIZE) GO TO 400
0093 TEMPAG = REPQAG(1)
0094 IF(ILNALS(1)) 310, 320, 330
C C SINK NODE POINTS TO NEW PAGE
0095 REPQAG(4) = HACTPG(1) + 1
0096 GO TO 340
C C SINK NODE POINTS TO CURRENT PAGE
0097 REPQAG(4) = CURPAG(1)
0098 GO TO 340
C C SINK NODE POINTS TO DIFFERENT PAGE
0099 REPQAG(4) = IVALS(1)
0100 REPQAG(1) + REPQAG(4)
0101 CALL LVGRN(SNKSUF)
0102 REPQAG(1) = TEMPAG

78
IVALS(1) = LVIFS(REQPAG(4),SUPSZE) .OR. SNKSUF

C BEGIN SEARCH FOR EXISTING LIST
0104 400  FULL = .FALSE.
0105  IF(INPXON .NE. 0) GO TO 420

C NORMAL INSERTION. FIND BOTTOM OF LIST
0107 410  IPOS = -1
0108 420  TEMPOS = IPOS
0109  TMPREQ = REQPAG(2)
0110  RFQPAG(2) = TMPREQ
0112  IPOS = TEMPOS

C LIST FOUND ON REQUESTED CONTINUANT ?
0113  IF(ITESTR .GT. 0) GO TO 500

C REQUEST FOR NEW CONTINUANT (OR PAGE) ?
0115  IF(REQCON .EQ. -1) GO TO 500

C PLACE NEW LIST ACCORDING TO RFQPAG(2) BUT FIRST SEARCH ELSEWHERE
0117  IPOS = TEMPOS
0118  TMPREQ = RFQPAG(2)
0119  RFQPAG(2) = -2
0120  CALL LVFDEX
0121  IPOS = TEMPOS
0122  RFQPAG(2) = TMPREQ

C LIST FOUND ON DIFFERENT CONTINUANT ?
0123  IF(ITESTR .GT. 0) GO TO 450

C LIST DOES NOT EXIST ON ANY CONTINUANT
0125  IPOS = TEMPOS
0126  CALL LVFDEX
0127  IPOS = TEMPOS
0128  RFQPAG(2) = TMPREQ
0129  GO TO 500

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  RFORG = .TRUE.
0133  CALL LVRFOR(REQCON)
0134  RFORG = .FALSE.
0135  GO TO 800

C PERFORM INSERTION
0136  500  CALL LVNSRT
0137  IF(.NOT. FULL) GO TO 800

C CONTINUANT IS FULL, PLACE ON NEXT CONTINUANT IF SPACE IS AVAILABLE
0139  FULL = .FALSE.

C SPECIAL HANDLING FOR OVERFLOW ON INDEXED INSERTION
0140  IF(INPXON .NE. 0) GO TO 600
C DOES A PORTION OF THE VVI RESIDE ON THE CURRENT CONTINUANT?

C SET VVI CONTINUATION FLAG
0142 506 IF(ITEMSTR .I.E. 0) GO TO 520
0144 FLGSPC(CONTL+LASTLC)+FLGSPC(CONTL+LASTLC) .OR. FLPG11
0145 NXTCON = .TRUE.
0146 520 REQPAG(2) = REQPAG(2) + 1
0147 FLGSPC(CONTL+HDRFLG)+FLGSPC(CONTL+HDRFLG) .AND. .NOT. NUSD
0148 CALL LVEXCH
0149 IF(MSARET .I.E. 0) CALL LVNC
0150 CALL LVFINP
0152 GO TO 506

C OVERFLOW ON INDEXED INSERTION
0153 500 IF(INPOS .EQ. -1) GO TO 505
0155 CALL LVOVER
0156 GO TO 505

C RESET CONTINUANT USAGE RATIO
0157 800 IF(PLACE .EQ. .TRUE.) GO TO 820
0159 LNSPCT(CONTL+INSDEL) = LNSPCT(CONTL+INSDEL) + NVAL

C CONTINUANT HAS BEEN MODIFIED
0160 820 FLGSPC(CONTL+HDRFLG)+FLGSPC(CONTL+HDRFLG) .AND. NOT NUSD
0161 NOTSPCP(CONTL+REGAS) = REGASP

C IF LIST IS CONTAINED ON MORE THAN ONE CONTINUANT,
C SET BACK POINTING FLAG
0162 IF(NXTCON .EQ. .TRUE.)
0163 FLGSPC(CONTL+THIS) = FLGSPC(CONTL+THIS) .OR. FLAG10

C IF A SPECIFIC CONTINUANT WAS REQUESTED, SET REORG INHIBIT FLAG
0164 IF(REQCON .NE. -2)
0165 FLGSPC(CONTL+THIS) = FLGSPC(CONTL+THIS) .OR. FLAG12
0166 REQPAG(2) = -2

C CALL SECOND USER INSERTION STRATEGY ROUTINE?
0167 1000 IF(IN2STR .EQ. .FALSE.) GO TO 1100
0168 INITMP = INITMP
0169 INZMP = INZMP
0170 INZSTR = .FALSE.
0171 INZSTR = .FALSE.
0172 DLTMP = DLSTR
0173 DLZMP = DLZSTR
0174 DLZSTR = .FALSE.
0175 DLZSTR = .FALSE.
0176 CALL LVSTAC
0177 INSIDE = .TRUE.
0178 CALL USRIN2
0179 INSIDE = .TRUE.
0180 INSIDE = .FALSE.
0181 CALL LVPOP
0179 INISTR = INITMP
0182 INZSTR = INZMP
0183 DLSTR = DLTMP
0184 DLZSTR = DLZMP
0185 DLTMP = DLTMP

C RESTORE FIND STRATEGY FLAGS
0186 1100 IF(INSIDE .EQ. .TRUE.) RETURN
0187 FDISTR = FDISTR
0188 FPZSTR = FPZSTR
0189 FPZSTR = FPZSTR
0190 RETURN
0191 END
SUBROUTINE LVNSRT
IMPLICIT INTEGER(A-Z)
LOGICAL*1 SNGLBK, SETUP, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR, DL1STR,
DL2STR, DUMPFI, CURRENT, FINDFI, DLETFI, NSRTFI, IN1TMP,
IN2TMP, FD1TMP, FD2TMP, FULL, REORG, LSTCON, RPLACE.
COMMON /VARGS/ IFUNC, IARG, IPOS, ITP, IVAL, IVAL, MSKIP, ITSTR,
INCLUD, INDXON, IVALS(10), ITYP(10), SRCBUF,
LNSUF, SNKSUF, SISTYP
COMMON /VREGS/ CTRPAG(4), RFOPAG(4), LSTVPG(4), MSARET,
BREOPG, XTMXSA, HACTPG(2), REACT, USECNT, DIFPAG,
DIREC, OUTLOC, OUTDIR(256), RWBUF(1)
COMMON /VFIA/ (FI 0MSK, FI 1MSK, FI 2MSK, FI 3MSK, FI 4MSK, FI 5MSK, FI 6MSK, FI 7MSK,
FI 8MSK, FI 9MSK, FI 10MSK, FI 11MSK, FI 12MSK, FI 13MSK, FI 14MSK),
 COMMON /VCNT/ (REGAS, CTRI 1, CTRI 2, CTRI 3, LEASTV, NTFREE, FREE, DREGS,
MSK, PAGIO.CURRENT,
COMMON /VBUFFR/ (PAGSZE, NCHAN, CHNCHAN, CMPFAD, PAGHDR, BUFZE, DIRSZE,
INCORE, HDRSZE, MSADIR, DFSZE, BLSZ, DIRB, L, PAGBD4)
COMMON /VHVIS/ (THSVISA, REGAS, PAGENO, CONTNO, INSDEL,
COMMON /VSBUF/ (SETUP, SNGLBK, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
DL1STR, DL2STR, IN1TMP, FD1TMP, FD2TMP, DUMPFI,
COMMON /VPRAM/ (BUFLOC, LENGTH, ERR, ERRNUM, BCP, BCPF, BIRW)
COMMON /VADDR/ (IADD, THIS, LSTTHEAD, LOC, LASTLC)
COMMON /VINFL/ (COUNT, ABSPOS, LSTCON)
COMMON /VINS/ (REORG, FULL, RPLACE)
COMMON /VVTY/ (NOISP(1))
1 /VVTY/ = LSTSPC(1)
2 /VVTY/ = LSTSPC(1)
3 /VVTY/ = LSTSPC(1)
DATA TWO/2/,THREE/3/,NFI067/-177774/,SV1RPL/-8/

C CALLS TO LVFIN & LVFNV MUST PRECEDE A CALL TO THIS ROUTINE.
C
D PAUSE 'IN LVNSRT'
RPLACE = .FALSE.
IF(INDXON-1) 125, 126, 127
C IS THE GIRS BUFFER FULL?
125 IF(REGAS.EQ.0) LSTSPC(CTRI 1 + REGAS) GO TO 98
C FORM FIRST WORD OF SINGLE OR MULTIVALUED FUNCTION
126 FDTP = FI 0MSK .OR. ITYP(11)
127 IF(INVAL .EQ. 1) GO TO 20
128 LSTTMP = REGAS
129 FDTP = FDTP .OR. FI 0MSK .OR. FI 2MSK
130 GO TO 21
131 LSTTMP = IVALS(1)
C IF THIS FUNCTION ALREADY EXISTS, GO TO 43
0029 IF (IFESTR .GT. 0) GO TO 43
C
C IF THAT ADDRESS IS ALREADY IN WORKING SPACE, GO TO 25
0031 IF ((FILEMSK .AND. FIGSPC(CTRL1 + IADD)) .NE. 0) GO TO 25
C
C UPDATE REGASP (IF NECESSARY)
0033 IF (IADD .EQ. REGASP) REGASP = LSTSPC(CTRL1 + IADD)
C
C UPDATE AVAILABLE SPACE
0035 LSTSPC(CTRL1 + NOPSPC(CTRL1 * IADD)) = LSTSPC(CTRL1 + IADD)
0036 NOPSPC(CTRL1 = LSTSPC(CTRL1 + IADD)) = NOPSPC(CTRL1 + IADD)
C
C INSERT FUNCTION
0037 NOPSPC(CTRL1 + IADD) = IFUNC
0038 LSTSPC(CTRL1 + IADD) = LSTTMP
0039 LNKSPC(CTRL1 + IADD) = IADD
0040 FIGSPC(CTRL1 + IADD) = FIGSPC(CTRL1 + IADD) .OR. FI2MP .OR. FILEMSK
C
C INSERT ANY ADDITIONAL FUNCTION VALUES
0041 IF (IVAL .EQ. 1) GO TO 100
0043 HEAD = IADD
0044 OIDLOC = IADD
0045 GO TO 50
C
C IF THAT ADDRESS CONTAINS THE HEAD OF A CONFLICT LIST, GO TO 60
0046 IF (FILEMSK .AND. FIGSPC(CTRL1 + IADD)) .GT. 0) GO TO 60
C
C IF THAT ADDRESS CONTAINS A VALUE ON A MULTIVALUE LIST, GO TO 35
0048 IF (FILEMSK .AND. FIGSPC(CTRL1 + IADD)) .GT. 0 .AND.
1 (FILEMSK .AND. FIGSPC(CTRL1 + 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 IF (LNKSPC(CTRL1 + LNKSPC(CTRL1 + THIS)) .EQ. IADD) GO TO 27
0053 THIS = LNKSPC(CTRL1 + THIS)
0054 GO TO 26
0055 LAST = LNKSPC(CTRL1 + THIS)
0056 NWLOC = REGASP
0057 IF (REGASP .EQ. LSTSPC(CTRL1 + REGASP)) GO TO 98
C
C UPDATE AVAILABLE SPACE AND REGASP
 CALL LVUPDT
C MOVE THE FUNCTION ON A CONFLICT LIST TO THE FIRST CELL OF AVAILABLE C SPACE
0060 NOPSPC(CTRL1 + NEWLOC) = NOPSPC(CTRL1 + IADD)
0061 LSTSPC(CTRL1 + NEWLOC) = LSTSPC(CTRL1 + IADD)
0062 LNKSPC(CTRL1 + NEWLOC) = LNKSPC(CTRL1 + IADD)
0063 FIGSPC(CTRL1 + NEWLOC) = FIGSPC(CTRL1 + IADD) .OR. FILEMSK
0064 FIGSPC(CTRL1 + IADD) = 0
0065 LNKSPC(CTRL1 + LAST) = NEWLOC
C
C INSERT THIS FUNCTION AS THE HEAD OF A CONFLICT LIST
0066 NOPSPC(CTRL1 + IADD) = IFUNC
0067    LNSPC(CTR I + IADD) = IADD
0068    LSTSPC(CTR I + IADD) = LSTTMP
0069    FIGSPC(CTR I + IADD) =
0070    FIGSPC(CTR I + IADD) .OR. FIGTMP .OR. FI4MSK .OR. FI5MSK
0071    IF((FIGSPC(CTR I + NEWLOC) .AND. FI4MSK) .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(CTR I + NEWLOC)
0073    NEXT = LSTSPC(CTR I + NEXT)
0074    IF(LSTSPC(CTR I + NEXT) .NE. IADD) GO TO 30
0076    LSTSPC(CTR I + NEXT) = NEWLOC
C
C INSERT ANY ADDITIONAL FUNCTION VALUES
0077    30   NEXT = IADD
0078    UILOC = IADD
0079    IF(NVAL .GT. 1) GO TO 50
0080    GO TO 100
C
C- THE ADDRESS CONTAINS A VALUE ON A MULTIVALUE LIST
0082    35   NEWLOC = REGASP
0083    IF(REGASP .EQ. FIGSPC(CTR I + REGASP)) GO TO 98
C
C UPDATE AVAILABLE SPACE AND REGASP
0085    CALL LVTFTD
C
C MOVE THE VALUE ON A MULTIVALUE LIST TO THE FIRST CELL OF
C AVAILABLE SPACE
0086    NODSPC(CTR I + NEWLOC) = NODSPC(CTR I + IADD)
0087    LSTSPC(CTR I + NEWLOC) = LSTSPC(CTR I + IADD)
0088    LNSPC(CTR I + NEWLOC) = LNSPC(CTR I + IADD)
0089    FIGSPC(CTR I + NEWLOC) = FIGSPC(CTR I + IADD)
0090    FIGSPC(CTR I + IADD) = 0
C
C RESET POINTERS
0091    LI = LSTSPC(CTR I + NEWLOC)
0092    IF((FIGMSK .AND. FIGSPC(CTR I + LI)) .EQ. 0) GO TO 200
0093    LNSPC(CTR I + LI) = NEWLOC
0094    GO TO 201
0095    200   NEWLOC = LI
0096    201   LI = LSTSPC(CTR I + NEWLOC)
0097    281   KZVAL = LSTSPC(CTR I + LNSPC(CTR I + NEWLOC))
0098    282   IF((FIGSPC(CTR I + KZVAL) .AND. FI4MSK) .NE. 0) GO TO 38
0100   LSTSPC(CTR I + LNSPC(CTR I + NEWLOC)) = NEWLOC
0101    GO TO 39
0102   38   LSTSPC(CTR I + LNSPC(CTR I + NEWLOC)) = LI
0103   39   NODSPC(CTR I + IADD) = IFUNC
C
C INSERT THIS FUNCTION AS THE HEAD OF A CONFLICT LIST
0104   LNSPC(CTR I + IADD) = IADD
0105   LSTSPC(CTR I + IADD) = LSTTMP
0106   FIGSPC(CTR I + IADD) = IADD
0107   FIGSPC(CTR I + IADD) .OR. FIGTMP .OR. FI4MSK .OR. FI5MSK
0108   GO TO 100
C
C- THE FUNCTION TO BE INSERTED IS ON THE CONFLICT LIST
0109   43   NEXT = IADD
83
C IS THIS A SINGLE VALUE LIST OR MULTIVALUE LIST?
0109 IF(LSTHEH .LT. 0) GO TO 51
C OIDLOC IS THE LOCATION OF THE LAST VALUE ON THE MULTIVALUE LIST
0111 OIDLOC = LNKSPC(CTR1 + LSTHEH)
C------------------------------------------------------------
C-INSERT ADDITIONAL FUNCTION VALUES
0112 50 LSTASP = NOPSPC(CTR1 + REGASP)
0113 IN = 0
0114 GO TO 56
C------------------------------------------------------------
C-FORM MULTIVALUE LIST TO ADD VALUE(S) TO SINGLE-VALUED FUNCTION
0115 51 IN = 0
0116 IF(REGASP .EQ. LSTSPC(CTR1 + REGASP)) GO TO 98
0118 LSTASP = NOPSPC(CTR1 + REGASP)
0119 NEWLOC = REGASP
0120 REGASP = LSTSPC(CTR1 + REGASP)
0121 NOPSPC(CTR1 + NEWLOC) = LSTSPC(CTR1 + THIS)
0122 TEMP = (FLGSPC(CTR1 + THIS) .AND. FIGSPC)
0123 FLGSPC(CTR1 + NEWLOC) = (TEMP .OR. FLGSPC(CTR1 + NEWLOC))
0124 FLGSPC(CTR1 + THIS) = (FLGSPC(CTR1 + THIS) .AND. NFIGSPC)
0125 FLGSPC(CTR1 + THIS) = (FLGSPC(CTR1 + THIS) .AND. FIGSPC)
0126 FLGSPC(CTR1 + THIS) = (FLGSPC(CTR1 + THIS) .AND. FIGSPC)
0127 LNKSPC(CTR1 + OIDLOC) = LNKSPC(CTR1 + OIDLOC + 1)
0128 OIDLOC + 1
C------------------------------------------------------------
C-INSERT ANOTHER VALUE ON MULTIVALUE LIST
0129 52 FIGSPC(CTR1 + NEWLOC) = (FLGSPC .OR. FLGSPC(CTR1 + NEWLOC))
0130 FIGSPC(CTR1 + NEWLOC) = (FLGSPC .OR. FLGSPC(CTR1 + NEWLOC))
0131 LSTSPC(CTR1 + NEWLOC) = NEWLOC
0132 LNKSPC(CTR1 + NEWLOC) = OLDLOC
0133 NEWLOC = NEWLOC
0134 56 NEWLOC = REGASP
0135 IF(IN .GT. 0) GO TO 57
C NO VALUES HAVE BEEN INSERTED YET
0137 IN = 1
0138 GO TO 58
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(CTR1 + REGASP)) GO TO 98
0144 REGASP = LSTSPC(CTR1 + REGASP)
0145 NOPSPC(CTR1 + NEWLOC) = NVALUES(IN)
0146 FIGSPC(CTR1 + NEWLOC) = (ITYPI(IN) .OR. FIGSPC(CTR1 + NEWLOC))
0147 ITYPI(IN) = 0
0148 GO TO 52
C END MULTIVALUE LIST AND UPDATE AVAILABLE SPACE
0149 67 LSTSPC(CTR1 + OIDLOC) = HEAD
C-THE FUNCTION TO BE INSERTED IS NOT ON THE CONFLICT LIST

0150 NOPSPCCTR1 = LSTASP
0151 LSTASP = NOPSPCCTR1 = REGASP
0152 LNKSCTR1 = LASTSPCTR1 = HEAD)) = OLDLOC
0153 GO TO 100

C

C-UPDATE AVAILABLE SPACE AND REGASP
0154 LNKSPCTR1 = LSTSPCTR1 = REGASP
0155 REGASP = RFGASP
0156 LSTASP = NOPSPCCTR1 * REGASP
0157 IF(REGASP .EQ. LSTSPCTR1) GO TO 98
0158 CALL LVPUDT

C INSERT FUNCTION IN FIRST CELL OF AVAILABLE SPACE
0159 NOPSPCCTR1 = ASPREG = IFUNC
0160 NV = NVL .EQ. 1) GO TO 611
0162 LSTSPCTR1 = ASPREG = REGASP
0163 FIGSCTR1 = ASPREG = (FLMSK .OR. FIGSCTR1 + ASPREG)
0164 FIGSCTR1 = ASPREG = (FIOMSK .OR. FIGSCTR1 + ASPREG)
0165 GO TO 612

0166611 LSTSPCTR1 = ASPREG = IVALS(1)
0167612 FIGSCTR1 = ASPREG = 1 FIGSCTR1 + ASPREG .OR. ITYP(1) .OR. FLMSK .OR. FIOMSK
0168 LNSPCTR1 = ASPREG = IADD
0169 LNSPCTR1 = LAST = ASPREG
0170 NV = NVL .EQ. 1) UC TO 100

C INSERT ADDITIONAL VALUES
0171 LSTASP = NOPSPCCTR1 = REGASP
0173 OLDLOC = ASPREG
0174 HEAD = ASPREG
0175 IN = 0
0176 GO TO 56

C DESTRUCTIVE INSERTION

C A CALL TO LVFINP 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 .IE. N+1

C DEFEAT SAVED INDEX UNTIL NEXT RETRIEVAL.
0177 126 FIGSCTR1 + THIS) = FIGSCTR1 + THIS) .OR. FIOMSK
0178 ABSPOS = IABS(IPOS)
0179 IPOS = IPAS
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 REPLAC = .TRUE.
0184 IF(LSTHED .GT. 0) GO TO 356

C SVI
0186 LSTSPCTR1 = LOC = IVALS(1)
0187 SVIRPL = 1
0188 GO TO 356
C MVI
0189 356 NOPSPCCTR1 + LOC = IVALS(1)
C REPLACE TYPE.
0190 365 FIGSPC(CTRL1 + LOC) = 
1
(FIGSPC(CTRL1 + LOC) .AND. NF1067) .OR. ITYP(1))
0191 GO TO 100
C
IPOS'TH VALUE WAS NOT FOUND. INDEXED INSERTION CAN STILL SUCCEED
I F (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 IF = KPOS
0200 CALL LVFIND(INDEX,INDEX,INDEX,INDEX)
C FAILURE IF NO VALUE IS FOUND.
0201 IF(IESTR .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 NWLOC = REGASP
0206 IF(REGASP .EQ. LSTSPC(CTRL1 + REGASP)) GO TO 98
0208 CALL LVFPDT
C SWI 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 DELETE SAVED INDEX UNTIL NEXT RETRIEVAL.
0214 FIGSPC(CTRL1 + THIS) = FIGSPC(CTRL1 + THIS) .OR. FI4MSK
0215 ABSPOS = IABS(IPOS)
0216 KPOS = IPOS
0217 INDEX = 0
0218 NWLOC = REGASP
C DOES THE IPOS'TH VALUE OF THE PROPER TYPE EXIST?
0219 IF(IESTR .IT. 0) GO TO 90
0221 IF(REGASP .EQ. LSTSPC(CTRL1 + REGASP)) GO TO 98
0223 CALL LVFPDT
C SWI 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 (WRIT ITYP) FROM THE TOP OF LIST
0228 377 INSTOC = INKSPC(CTRL1 + LOC)
0229 NINSTOC = INSTOC + NWLOC = IVALS(1)
0230 LSTSPC(CTRL1 + NWLOC) = LOC
0231 INKSPC(CTRL1 + NWLOC) = INSTOC
0232 FIGSPC(CTRL1 + NWLOC) = FI4MSK .OR. FI2MSK .OR. ITYP(1))

86
IF (LOC .NE. LSTHED) GO TO 321
0235 LSTSPC (CTRI1 + LSTSPC (CTRI1 + ISTLOC)) = NEWLOC
0236 GO TO 322
0237 321: LSTSPC (CTRI1 + ISTLOC) = NEWLOC
0238 322: LNKSPC (CTRI1 + LOC) = NEWLOC
0239 GO TO 100

C PLACE VALUE AT THE IPOS'TH POSITION (WRT ITYP) FROM THE BOTTOM OF THE LIST.

0240 347: NOPSPC (CTRI1 + NEWLOC) = IVALS(1)
0241 LSTSPC (CTRI1 + NEWLOC) = LSTSPC (CTRI1 + LOC)
0242 LNKSPC (CTRI1 + NEWLOC) = LOC
0243 FIGSPC (CTRI1 + NEWLOC) = FLMSK .OR. FL2MSK .OR. ITYP(1)
0244 IF (FIGSPC (CTRI1 + LSTSPC (CTRI1 + LOC)) .AND. FLMSK .EQ. 0)
     1 GO TO 323
0246 KZVAL = LSTSPC (CTRI1 + LOC)
0247 LNKSPC (CTRI1 + LSTSPC (CTRI1 + KZVAL)) = NEWLOC
0248 GO TO 324
0249 323: LNKSPC (CTRI1 + LSTSPC (CTRI1 + LOC)) = NEWLOC
0250 324: LSTSPC (CTRI1 + LOC) = NEWLOC
0251 GO TO 100

C CREATE MVI WITH NEW VALUE AT THE TOP OF THE LIST.

0252 344: IF (REGASP .EQ. LSTSPC (CTRI1 + REGASP)) GO TO 98
0254 NWLOC2 = REGASP
0255 CALL LVIPDT
0256 NOPSPC (CTRI1 + NEWLOC) = IVALS(1)
0257 LSTSPC (CTRI1 + NEWLOC) = NWLOC2
0258 LNKSPC (CTRI1 + NEWLOC) = NWLOC2
0259 FIGSPC (CTRI1 + NEWLOC) = FLMSK .OR. FL2MSK .OR. ITYP(1)
0260 NOPSPC (CTRI1 + NWLOC2) = LSTSPC (CTRI1 + THIS)
0261 LSTSPC (CTRI1 + NWLOC2) = THIS
0262 LNKSPC (CTRI1 + NWLOC2) = NEWLOC
0263 FIGSPC (CTRI1 + THIS) .AND. FL67
0264 FIGSPC (CTRI1 + NWLOC2) = (FLMSK .OR. FL2MSK) .OR. FIGSPC (CTRI1 + THIS)
0265 LSTSPC (CTRI1 + THIS) = NEWLOC
0266 FIGSPC (CTRI1 + THIS) = FIGSPC (CTRI1 + THIS) .OR. FLMSK .OR. FL2MSK

C FLAG 4 IS SET BECAUSE THIS INSERTION MIGHT BE A RECREATION OF AN OLD LIST.

0267 100: FIGSPC (CTRI1 + THIS) = FIGSPC (CTRI1 + THIS) .OR. FLMSK
0268 IVAL = IVALS(1)

C "FAILURE" IF IFUNC+IARG DID NOT PREVIOUSLY EXIST.

C IF (FIGSPC (CTRI1 + THIS) .AND. FLMSK .NE. 0) .OR.
C 
C SVIRPL .EQ. 1
0269 97: IPOS = 1
0270 SVIRPL = 0
0271 ITYP = 3
0272 INDXON = 0
0273 NVAL = 1
0274 ITYP(1) = 0
0275 RETURN

C CONTINUANT IS FULL, TRY AGAIN.

0276 98: FULL = .TRUE.
0277 RETURN
0278 END

87
C
C
SUBROUTINE LVUPDT
IMPLICIT INTEGER(A-Z)
LOGICAL*1 CURRENT
COMMON /VCRNT/ REGASP,CTRI1,LEASTV,NTFREE,FREE,DREGSP,

COMMON /VTRI/ NOSPCE
1 /VTR2/ LSTSPC
2 /VTR3/ LNSPC
3 /VTR4/ FLGSPC
COMMON /VTRI/ NODSPC
1 /VTR2/ LSTSPC
2 /VTR3/ LNSPC
3 /VTR4/ FLGSPC
COMMON /VTRI/ NODSPC
1 /VTR2/ LSTSPC
2 /VTR3/ LNSPC
3 /VTR4/ FLGSPC
COMMON /VTRI/ NODSPC
1 /VTR2/ LSTSPC
2 /VTR3/ LNSPC
3 /VTR4/ FLGSPC

THIS ROUTINE UPDATES AVAILABLE SPACE AND THE REGISTER OF AVAILABLE SPACE - REGASP

PAUSE 'IN LVUPDT'
LSTSPC(CTRI1 + NOSPCE(CTRI1 + REGASP)) = LSTSPC(CTRI1 + REGASP)
NOSPCE(CTRI1 + LSTSPC(CTRI1 + REGASP)) = NOSPCE(CTRI1 + REGASP)
REGASP = LSTSPC(CTRI1 + REGASP)
XXX = 1000
IF (FLGSPC(CTRI1+REGASP) .OR. FL3MSK) .NE. FL3MSK) XXX=XXX*XXX
RETURN
END
SUBROUTINE LVPLLEX

IMPLICIT INTEGER(A-Z)

LOGICAL*1 SNGI BK,SETUP,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,DL1STR.
1   DL2STR,DUMPP1,CURRENT,FINDF1,DELPFI,NSRTFI,FD1TMP,
2   DL2TMP,IN2TMP,FD2TMP,INSIDE,REORG,FULL,LSTCON,RPLACE,
3   BARCON

COMMON /TVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,
1   INCLUDE,INPNON,IVAL(10),ITYP(10),SNGI.BK.
2   LNSUP,SNKSUF,INSTYP

COMMON /TVREGS/ CURPGA(4),REOPG(4),LSTVPG(4),MSARET,
1   USEOPG,NXMSA,HACTPG(2),READCT,USECNTR,DILPAG,
2   DIREC,DUTFIC,DUHDR(256),IXBUF(1)

COMMON /LVMSK/ WRITE,NOTUSBD,NEWCON,FLMSK,MASKSF,MASKPF

COMMON /TVFLG/ FL0MSK,FL1MSK,FL2MSK,FL3MSK,FL4MSK,FL5MSK,FL6MSK,FL7MSK.
1   FL067,FL068,FL069,FL070,FL071,FL072,FL073,FL074.
2   FL075,FL076

COMMON /TVVRN/ REGASP,CTRLCT,CTRL1,LEASTV,NFFREE,FREE,DEGRES,
1   WSA,PAGLOC,CURRENT

COMMON /TVVFIR/ PAGSZE,NCCHAN,OCCHAN,CMPAND,PAGHDR,BUFZIE,DIRSZE,
1   INCORE,HDMSIZE,MSADIR,DFSZE,DKSKZE,DIRBK,PAGHD4

COMMON /TVPRM/ TMSKS.,REGAS,PAGENO,CONTNO,INSINE,
1   USECT,HDRT1,READW.,RMNH,NAMES,BISKZE,DROWH

COMMON /TVST/ SETUP,SNGL BK,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,
1   DL1STR,DL2STR,IN2TMP,FD1TMP,FD2TMP,INSIDE,REORG,FULL,LSTCON,
2   BARCON

COMMON /TVSCN/ COUNT,ABSPOS,LSTCON

COMMON /TVINS/ REORG,FULL,RPLACE

COMMON /TVDEL/ NPRET,BARCON

COMMON /TVV3/ NOSPSC(1)
1   /TVV3/ LSTSPC(1)
2   /TVV3/ LNSPC(1)
3   /TVV3/ FLSPC(1)

DATA INSIDE/.FALSE./

THE DELETE EXECUTIVE ROUTINE OBTAINS THE CORRECT P,C FOR SUBROUTINE
LVPLLEX TO OPERATE ON.

PAUSE 'IN LVPLLEX'.

IS LVPLLEX BEING CALLED FROM A DELETE STRATEGY ROUTINE ?

IF INSIDE .EQ .TRUE. ) GO TO 100

TO PREVENT RECURSION. SAVE THE FIND STRATEGY FLAGS AND TURN THEM OFF

FD1STR = FD1STR

FD2STR = FD2STR

FD1STR = .FALSE.

FD2STR = .FALSE.

CALL USER'S FIRST DELETE STRATEGY ROUTINE ?

IF DL1STR .EQ .FALSE.) GO TO 100

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 SET UP FOR FIRST USER ROUTINE
0034 CALL LVSTAC
0035 INSIDE = .TRUE.
0036 CALL USRDLY
0037 INSIDE = .FALSE.
0038 CALL LVPOP
0039 DL1STR = DL1TMP
0040 DL2STR = DL2TMP
0041 IN1STR = IN1TMP
0042 IN2STR = IN2TMP

C PROCEED WITH DELETION?
0043 IF(DELETFI.EQ. .FALSE.) GO TO 600

C BRING IN PROPER CONTINUANT
0045 100 J = 0
0046 CALL LVFPEX(J,J,J,J)

C NO LIST TO BE DELETED?
0047 IF(ITESTR .IT. 0) GO TO 600

C ASSUME LIST DOES NOT PROCEED TO ANOTHER CONTINUANT
0049 200 LSTCON = .FALSE.
0050 BAKCON = .FALSE.

C NUMRET COUNTS THE NUMBER OF LOCATIONS RETURNED TO AVAILABLE SPACE
0051 NUMRET = 0
0052 CALL LVINLET

C UPDATE CONTINUANT FILL QUANTITY
0053 LNKSPC(CTRPT+INSDEL) = LNKSPC(CTRPT+INSDEL) - NUMRET

C CONTINUANT HAS BEEN MODIFIED
0054 FIGSPC(CTRPT+HDRFIG) = FIGSPC(CTRPT+HDRFIG) .OR. MWRITE

C INDEXED DELETE?
0055 IF(INPXON .EQ. 1) GO TO 400

C FINISHED?
0057 IF(LSTCON .EQ. .FALSE.) GO TO 600

C EXAMINE NEXT CONTINUANT
0059 300 RFUPAG(2) = CUR*AG(2) + 1
0060 J = 0
0061 CALL LVFPEX(J,J,J, J)

C NO MORE CONTINUANTS?
0062 IF(MSARET .IE. 61) GO TO 500
C
C DOES A PORTION OF THE LIST RESIDE ON CURRENT CONTINUANT?
0064 IF (ITESTR .LT. 0) GO TO 300
0066 GO TO 200
C
C SET DEFAULT TO "DELETE ENTIRE LIST"
0067 400 INPRON = 0
0068 IF (BARCON .EQ. .FALSE.) GO TO 450
C
C LIST NO LONGER POINTS FORWARD TO A FOLLOWING CONTINUANT, REMOVE FLAG
0070 400 IPSS = -1
0071 CALL LVPFX
0072 FLGSPC (CTR11+LOC) = FLGSPC (CTR11+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 400 IPSS = 1
0076 CALL LVPFX
0077 FLGSPC (CTR11+THIS) = FLGSPC (CTR11+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 LVUUMP (DUMP)
0084 X = 10000
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 INITMP = INITMP
0093 INITSTR = INITSTR
0094 INIT2MP = INIT2MP
0095 INIT2STR = INIT2STR
0096 INIT2STR = .FALSE.
0097 CALL LVSTAC
0098 INSIDE = .TRUE.
0099 CALL USKRLZ
0100 INSIDE = .FALSE.
0101 CALL LVPOP
0102 DL1STR = DL1TMP
0103 DL1STR = DL2STR
0104 DL2STR = INITMP
0105 DL2STR = INIT2MP
0106 700 IF (INSIDE .EQ. .TRUE.) RETURN
0108 FD1STR = FD1TMP
0109 FD2STR = FD2TMP
0110 RETURN
0111 END
SUBROUTINE LVPLLET

IMPLICIT INTEGER(4)Z

LOGICAL*1 SNGI.BX, SETUP, NTTRAN, NT1STR, NT2STR, FP1STR, FP2STR, DL1STR, DL2STR, DUMPF1, CURRENT, FINDFI, DLETF1, NSRTP, DL1TMP,

DLETF2, DL2TMP, FP1TMP, FP2TMP, RO, NROG, FULL, LSTCON, REPLACE, BACON

COMMON /VARS/ FUNC, IARG, POS, ITYP, IVAL, NVAL, NSKIF, ITESTR,

1 INCLUD, INDNON, IVALS(10), ITYP(10), SRCSUF,

2 NSKIF, NSKIF, NSKIF

COMMON / VRESS/ CURPAG(4), BPAGAG(4), LSTPRG(4), MSARPT,

1 BREPAG, NTXNSA, HACTPRG(2), LR10C, USECTM, DDIRAG,

2 DIRECT, OUTLOC, OUTDIR(256), RNDVF1(4)

COMMON / VMARK/ WTMP, NTUSP, NWSIF, NNSDF, MASKSF, MASKPF

COMMON / VF1AG/ FT0NSK, FI 1NSK, FT 2NSK, FI 3NSK, FI 4NSK, FI 5NSK, FI 67,

1 FiAG1, FiAG1, FiAG1, FiAG1, FiAG1, FiAG1, FiAG1, FiAG1

2 FiAG1, FiAG1, FiAG1, FiAG1, FiAG1, FiAG1, FiAG1, FiAG1

COMMON / VCRNT/ REGASP, CRTR1, CTTRI 1, LEASTV, NTFREE, FREE, DEORG,

1 MSAPG OC, CURRENT

COMMON / VBUFR/ PAGPSZ, NCHAN, GI CHA, CMPAND, PAGHDR, BUFSZE, DIRSZE,

1 INCORE, HDRSSZ, MSADIR, SUFFSE, BLSSZE, DIRBK, KPDAG

COMMON / VHD/V1/ THSNSA, NREGS, PAGINO, CONTNO, MNSDEL,

1 USECT, HDBRT G, READVI, OF DMIV, DNOPER, NR4, KBIRD, Drown

COMMON / VSMIT/ SETUP, SNGI.BX, NTTRAN, NT1STR, NT2STR, FP1STR, FP2STR,

1 DL1STR, DL2STR, DUMPF1, DL1TMP, FP2TMP, DL2TMP, DUMPF1

COMMON / VF1AG/ DLETF1, NSRTP, DLETF2, DLETF2

COMMON / VFRPR/ BUFUC. LENGTH, ERR, ERRNUM, BINARY, BCP, MODE, PAGES,

1 LUN

COMMON / VAOR/ ADD, THIS, LSTHE D, LOC, LAST, LASTLC

COMMON / VSTAK/ CURVI, CTR, VAVAR, STACK(11)

COMMON / VFR/ COUNTER, ARSF, LSTCON

COMMON / TVNS1/ ROEG, FULL, REPLACE

COMMON / Vbohydr/ WRTCT, BACON

COMMON / VVRTR/ NOSRC(1)

1 / VVRTR2/ LSTSPC(1)

2 / VVRTR3/ LMSPC(1)

3 / VVRTR4/ F1GSPC(1)

DATA NF1023/ '177517/

DOES THE LIST EXIST?

PAUSE 'IN LVPLLET

IF(ITESTR .LT. 0) RETURN

C SVI OR NWI?

IF(LSTHE D .LT. 0) GO TO 200

C INDEXED DELETE?

IF(INDIVX .EQ. 1) GO TO 500

C DELETE ENTIRE MULTIVALUE LIST

ISADD = LSTHE D

LOC = THIS

100 NXTADD = LSTSPC(CTRL1 + ISADD)

IF((FL GSPC(CTR1 + NXTADD) .AND. FLAG1) .NE. 0) LSTCON = .TRUE.
DELETE SINGLE VALUED FUNCTION
C FORWARD OR BACK NEW CONTINUANT POINTER FLAGS MAY HAVE TO BE REMOVED
IF((FIGSPC('CTR1 + THIS) .AND. F1AG11) .NE. 0) LSTCON = .TRUE.
C IF THE LIST EXTENDED TO BOTH A PREVIOUS AND FOLLOWING CONTINUANT,
C DO NOT REMOVE POINTER FLAGS
IF(LSTCON .EQ. .FALSE.) GO TO 220
C IS THE FUNCTION HEAD OF A CONFLICT LIST
IF((FIGSPC('CTR1 + IADD) .AND. F1AGNSK) .EQ. 0) GO TO 220
C OTHERWISE, PLACE NEXT FUNCTION ON CONFLICT LIST IN /HEAD OF
C CONFLICT LIST/ LOCATION (IADD)
IF(NXFTNC .EQ. .FALSE.) GO TO 300
C IF THE MODIFIED FUNCTION IS A NVL. THE POINTER FROM THE LAST VALUE OF
C THE LIST TO THE HEAD MUST BE UPDATED.
KVAL = LSTSPC('CTR1 + IADD)
1 GO TO 250
LSTSPC('CTR1 + KVAL) = IADD
LOC = NXF1NC
C RETURN LOCATION TO AVAILABLE SPACE
IF((FIGSPC('CTR1 + LSTSPC('CTR1 + KVAL)) .AND. F1OMSK) .EQ. 0)
1 GO TO 250
LSTSPC('CTR1 + LOC) = IADD
LOC = NXF1NC
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
LKNSPC('CTR1 + LAST) = LKNSPC('CTR1 + THIS)
GO TO 300
C
**INDEXED DELETE**

FUNCTION MUST BE A MVI OR, IF SVI, ABS(IPOS) = 1 WITH PROPER TYPE.

DELETE VALUE AT LOC. DEFEAT SAVED INDEX FOR THIS LIST UNTIL AFTER NEXT RETRIEVAL.

INDEXED DELETE CAN BE REDUCED TO FOUR CASES. DELETE VALUE IN
FIRST, MIDDLE, OR LAST POSITION ON LIST, OR REDUCE TO SVI.

**INDEXED DELETE CAN BE REDUCED TO FOUR CASES. DELETE VALUE IN**
FIRST, MIDDLE, OR LAST POSITION ON LIST, OR REDUCE TO SVI.

INDEXED DELETE CAN BE REDUCED TO FOUR CASES. DELETE VALUE IN
FIRST, MIDDLE, OR LAST POSITION ON LIST, OR REDUCE TO SVI.

INDEXED DELETE CAN BE REDUCED TO FOUR CASES. DELETE VALUE IN
FIRST, MIDDLE, OR LAST POSITION ON LIST, OR REDUCE TO SVI.

INDEXED DELETE CAN BE REDUCED TO FOUR CASES. DELETE VALUE IN
FIRST, MIDDLE, OR LAST POSITION ON LIST, OR REDUCE TO SVI.

INDEXED DELETE CAN BE REDUCED TO FOUR CASES. DELETE VALUE IN
FIRST, MIDDLE, OR LAST POSITION ON LIST, OR REDUCE TO SVI.

INDEXED DELETE CAN BE REDUCED TO FOUR CASES. DELETE VALUE IN
FIRST, MIDDLE, OR LAST POSITION ON LIST, OR REDUCE TO SVI.

INDEXED DELETE CAN BE REDUCED TO FOUR CASES. DELETE VALUE IN
FIRST, MIDDLE, OR LAST POSITION ON LIST, OR REDUCE TO SVI.

INDEXED DELETE CAN BE REDUCED TO FOUR CASES. DELETE VALUE IN
FIRST, MIDDLE, OR LAST POSITION ON LIST, OR REDUCE TO SVI.

INDEXED DELETE CAN BE REDUCED TO FOUR CASES. DELETE VALUE IN
FIRST, MIDDLE, OR LAST POSITION ON LIST, OR REDUCE TO SVI.

INDEXED DELETE CAN BE REDUCED TO FOUR CASES. DELETE VALUE IN
FIRST, MIDDLE, OR LAST POSITION ON LIST, OR REDUCE TO SVI.
SUBROUTINE LVSETP
IMPLICIT INTEGER(A-Z)
REAL*4 DEFEXT, CORE, TOP, BOTTOM
LOGICAL*4 SINGBK, SETUP, NYTTRAN, INSTR, IN2STR, FDF1STR, FD2STR, DL1STR, DL2STR, DUNPFI, CURRENT, IN2TMP, FD2TMP, DL2TMP, FINDFI,
COMMON /VARS/ IFUNC, IARG, IPOS, ITYP,IVAL,IVAL,NSKIP, ITESTR,
INCLUDE, INWON, IVALS/10/, ITPF(10), SRCETF,
LOGICAL01 SINGBK, S2TUP, NXTRAN, INISTR, IN2ISTR, P2R, DL1STR, I
DL2STR, DUNPFI*CURENT, IN2TUP, FD2TNP, DL2TNP, FI67,
FLAGE, FLAG9, FLAGE5, FLAGE12, FLAG13, FLAG14,
COMMON /IVREGS/ CIAJRPAG(4), REQPAG,(4), READCT, USECNT, D1RPAG,
COMMON /IVASK/ MWRITE, NOTUSDN, NECN, GMSGK, MASKSF, MASKPF
COMMON /IVFIAG/ F1NSK, F1IMSK, F12NSK, P2NSK, F1AG9, F1AG12, F1AG13, F1AG14
COMMON /IVVRANP/ PRI ME, SEED, NROW, DNOPE, DROW, OLDNOP, LISTSZ,
LISTSZ(2), INCORE, HDRSZE, NSADIR, SITFSZ, BKSZE, DIRBIK, PHA5K,
COMMON /IVBUFR/ PAGSZ, NWCILLAN, CiJAN, CPANP, PAUWDR, BITFSZE, 
INCORE, HDRSZE, NSADIR, SITFSZ, BKSZE, DIRBIK, PHA5K,
COMMON /IVVTRI/ NOSPCE, IN3POS, LASTSO, SEQSPC(1)
COMMON /IVVTR2/ LSTSPC(1)
COMMON /IVVTR3/ LNSPCE(1)
COMMON /IVVTR4/ LISPC(1)
COMMON /IVVTR5/ NOSPCE(1)
COMMON /IVVTR6/ NOSPCE(1)
COMMON /IVVTR7/ NOSPCE(1)
COMMON /IVVTR8/ NOSPCE(1)
COMMON /IVVTR9/ NOSPCE(1)
COMMON /IVVTR10/ NOSPCE(1)
COMMON /IVVTR11/ NOSPCE(1)
COMMON /LVSTR/ CURVE, NUMVAR, STACK(1)
COMMON /LVVUL/ FILSPC(39), DEFEXT(2)
COMMON /LVVUN/ REORG, FULL, REPLAC
COMMON /LVVTP/ CORE
COMMON /LVVTR/ NOSPCE(1)
COMMON /LVVTR2/ LSTSPC(1)
COMMON /LVVTR3/ LNSPCE(1)
COMMON /LVVTR4/ LISPC(1)
COMMON /LVVTR5/ NOSPCE(1)
PAUSE 'IN LVSETP'
IP(SINGBK) GO TO 120
PAGHDR = PAGSZ + HDRSZE
BHKSZE = PHA5K * 8
PAGHDR = 4*PAGHDR
DIRSZE = 64*(INCORE/64) + 1
DIRBIB = DIRSZE/64
BKSZE = DIRSZE * (INCORE*PAGHDR)
TYPE 1
! FORMAT('PLEASE ENTER FILE NAMES OF OLD AND NEW GRAPHS'/
1. 'IN COMMAND STRING FORMAT (NEW.EXT = OLD.EXT)/
2. 'GRF IS ASSUMED EXTENSION'/
IF(ICSI(FILSPC,DEFEXT,.0).NE.0) STOP 'INVALID COMMAND STRING'
RUN TYPE 1 - CREATE NEW GRAPH
C RUN TYPE 2 - UPDATE OLD GRAPH
C RUN TYPE 3 - QUERY OLD GRAPH
0033 IF(RUNTYE.EQ. 3) GO TO 100
C ASSIGN CHANNEL TO OUTPUT (NEW) GRAPH
0035 NWCHAN = IGETC()
0036 IF(NWCHAN .LT. 0) STOP 'NO OUTPUT CHANNEL AVAILABLE'
0038 IF(IFETCH(FILSPC(1)).NE. 0) STOP 'OUTPUT DEVICE HANDLER FETCH FAILURE'
0040 IF(IENTER(NWCHAN,FILSPC(1),0) .LT. 0) STOP 'ENTRY FAILURE'
0042 IF(RUNTPY.EQ. 1) GO TO 110
C ASSIGN CHANNEL TO INPUT (OLD) GRAPH
0044 OCHAN = IGETC()
0045 IF(OCHAN .LT. 0) STOP 'NO INPUT CHANNEL AVAILABLE'
0047 IF(IFETCH(FILSPC(16)).NE. 0) STOP 'INPUT DEVICE HANDLER FETCH FAILURE'
0049 IF(ILOOK(0CHAN,FILSPC(16)).LT. 0) STOP 'INPUT FILE LOOKUP 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 CREATION RUN
0053 110 READCT = 1
0054 USECNT = 1
C
0055 120 SPED = PRINE/2
0056 NROW = SEED
0057 OIDN0P = SPED - PRIME
0058 DROW = PRIME
0059 DNOPE = PRIME
0060 LISTSZ = 1
0061 REGASP = 1
C SET UP SINGLE BLOCK?
0062 IF(SNGLBK) GO TO 160
0064 140 DO 145 1 = 1,84
0065 J = .5*(1-1)
0066 GRNTBI(J + OIDN0P) = OIDN0P
0067 GRNTBI(J + DNOPE) = DNOPE
0068 GRNTBI(J + NROW) = NROW
0069 145 GRNTBI(J + DROW) = DROW
0070 TOP = INCORE-1
0071 BOTTOM = INCORE
C SET UP DIRECTORY AVAILABLE SPACE
0072 IF(BOTTOM .EQ. 0) BOTTOM = 1
0074 CORE = TOP/BOTTOM
C SET UP DIRECTORY AVAILABLE SPACE
0075 DRETP = 1
0076 DO 150 I = 2,DIRSZF
0077 NOSPC(I) = -1
0078 LSTSPC(I-1) = 1
0079 LNSPC(I) = 0
0080 150 FISPC(I) = FI3MSK
0081 NOSPC(1) = DIRSZF
0082 LNSPC(I) = 0
0083 FLSPC(I) = FI3MSK
0084 LSTSPC(DIRSZF) = 1
C
D PAUSE 'LEAVING LVSETP 2'
RETURN
C C CREATE CONTINUANTS BY PAGE ORDER
0135 210 PAGNUM = 0
0137 CTRLPT = DIRSZ
0138 CTRI1 = CTRLPT + HDRSZ
C C CREATE PAGE PAGNUM, CONTINUANT CONTIN.
C COMPLETE HEADER WORDS FOR CONTINUANTS
0139 LIST = HREPFG
0140 DIRCNT = 0
0141 DIRPAG = 1
C INITIALIZE NXTMSA TO LOCATION OF PAGE 1, CONT 0
0142 NXTMSA = DIRBK + 10
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 NOISPC(CTRLPT + THSMSA) = NXTMSA
0154 LSTSPC(CTRLPT + PAGENO) = PAGNUM
0155 LSTSPC(CTRLPT + CONTNO) = CONTIN
C OUTPUT UNUSED CONTINUANT TO DISK
0156 LENGTH = PAGBRD
0157 BULOC = CTRLPT + 1
0158 ERRNUM = 0
0159 NSA = NOISPC(CTRLPT + THSMSA)
0160 CALL LVPGW
C ENTER CONTINUANT LOCATION INTO OUTCORE DIRECTORY
0161 OUTLOC = 1 + 64*(DIRCNT-1) + CONTIN
0162 OUTDIR(OUTLOC) = NSA
0163 NXTMSA = NXTMSA + BUFSZ
C UPDATE CONTROL POINTER
0164 CTRLPT=CTRLPT+PAGBRD
0165 IF(CTRLPT.GE.BUFSZ) CTRLPT=DIRSZ
0167 CONTINUE
C SAVE THIS BLOCK OF THE OUTCORE DIRECTORY IF ALL 4 SEGMENTS ARE FILLED
0168 IF(DIRCNT .LT. 4) GO TO 300
0170 CALL LVPRWR
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 LVPRWR
C C ZERO OUT REMAINING UNUSED OUTCORE DIRECTORY BLOCKS
0177 310 DO 312 J=1,256
0178 312 OUTDIR(J) = 0
C
0179 315 IF(DIRPAG .EQ. 16) GO TO 320
0181  DIRPAG = DIRPAG + 1
0182  CALL LVPRWR
0183  GO TO 315

C BRING FIRST OUTCORE DIRECTORY BLOCK INTO MAIN MEMORY

0184 320 REOPAG(1) = 1
0185  REOPAG(2) = 0
0186  CALL LVNSRA(DUMMY)

C IF IT IS NOT THERE, BRING PAGE 1, CONTINUANT 0 BACK INTO CORE

0187  CTRL1 = DIRSZE
0188  CTRL1 = CTRL1 + HDRSZE
0189  IF((LSTSPC(CTRL1 * PAGENO) .EQ. 1) .AND.
1   (LSTSPC(CTRL1 + CONTNO) .EQ. 0)) GO TO 340
0190  NSA = OUTDIR(1)
0191  BUFLOC = CTRL1 + 1
0192  LENGTH = PGHDR
0193  ERIMENT = 2
0194  CHAN = NVCHAN
0195  CALL LVPGAR(CHAN)

C INSERT INCORE CONTINUANTS INTO DIRECTORY PAGE

0196 340 CTRLPT = - HDRSZE
0197  CTRL1 = 0
0198  TMFSZE = PAGSZE
0200  PAGSZE = DIRSZE
0201  CNTR1 = DIRSZE - PGHDR
0202  INXMOV = 0
0203  NVAL=1
0204  INSTYP=1
0205  DO 400 K=1,INCORE
0206  CNTR1 = CNTR1 + PGHDR
0207  PAGE = LSTSPC(CNTR1 + PAGENO)
0208  CONT = LSTSPC(CNTR1 + CONTNO)
0209  IARG = CONT + 1
0210  IFUNC= PAGE
0211  SRCSUF = IARG
0212  LNKSUF = IFUNC
0213  CALL LVIFYND
0214  CALL LVPFIND
0215  CALL 'AFT FIND'
0216  SNESUF = IVALS(1)
0217  CALL LVNSRAT
0218  CONTINUE
0219  DREGSP = REGASP
0220  REGASP=1
0221  PAGSZE = TMFSZE

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
C

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

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

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

C DECLARE THE LAST CONTROL POINT AS AVAILABLE.

0235 LEASTV = FREE
0236 CNTR1 = BUFSZE - PAGHDR
0237 LSTVPG(1) = LSTSPC(CNTR1 + PAGENO)
0238 LSTVPG(2) = LSTSPC(CNTR1 + CONTNO)
0239 LSTVPG(3) = NODSPC(CNTR1 + THSWSA)
0240 LSTVPG(4) = CNTR1
D

PAUSE 'LEAVING LSSETP 3'
0241 RETURN
0242 END
C
C
0001    BLOCK DATA
0002    IMPLICIT INTEGER(A-Z)
0003    REAL*4, DEFEXT
0004    LOGICAL*1, SNGI.BK, SETUP, NXTRAN, IN1STR, IN2STR, PD1STR, PD2STR, DL1STR,
0005    DL2STR, DUMPFI, CURRENT, IN2TMP, DLTMP, DL2TM, FINDFI,
0006    DLETFL, NSRTFI, REORG, FULL, RPLACE
0007    COMMON /IVARGS/ IPUNC, IARG, IPES, ITP, IVAL, NVAL, NSKIP, ITESTR,
0008    INCLUD, INDXON, IVALS(10), ITYPE(10), SRCSUP,
0009    IPOS, ITYP, IVALS, NVAL, NSKIP, ITESTR,
0010    COMMON /I.VARGS/ IPOS, ITYP, IVALS, NVAL, NSKIP, ITESTR,
0011    COMMON /I.VARGS/ IPOS, ITYP, IVALS, NVAL, NSKIP, ITESTR,
0012    COMMON /I.VARGS/ IPOS, ITYP, IVALS, NVAL, NSKIP, ITESTR,
0013    COMMON /I.VARGS/ IPOS, ITYP, IVALS, NVAL, NSKIP, ITESTR,
0014    COMMON /I.VARGS/ IPOS, ITYP, IVALS, NVAL, NSKIP, ITESTR,
0015    COMMON /I.VARGS/ IPOS, ITYP, IVALS, NVAL, NSKIP, ITESTR,
0016    COMMON /I.VARGS/ IPOS, ITYP, IVALS, NVAL, NSKIP, ITESTR,
0017    COMMON /I.VARGS/ IPOS, ITYP, IVALS, NVAL, NSKIP, ITESTR,
0018    COMMON /I.VARGS/ IPOS, ITYP, IVALS, NVAL, NSKIP, ITESTR,
0019    COMMON /I.VARGS/ IPOS, ITYP, IVALS, NVAL, NSKIP, ITESTR,
0020    COMMON /I.VARGS/ IPOS, ITYP, IVALS, NVAL, NSKIP, ITESTR,
0021    COMMON /I.VARGS/ IPOS, ITYP, IVALS, NVAL, NSKIP, ITESTR,
0022    COMMON /I.VARGS/ IPOS, ITYP, IVALS, NVAL, NSKIP, ITESTR,

C
C
0023    IF THE FOLLOWING FLAGS ARE ON, THEY REPRESENT THE FOLLOWING:
0024    C
0025    FLG0MK- HEAD OF A MULTIVALUED LIST
0026    C
0027    FLG0MK- THE CELL IS IN WORKING SPACE, NOT AVAILABLE SPACE

101
C FL2MSK - VALUE ON A MULTIVALUE LIST
C FL3MSK - A NODE HAS BEEN DEFINED WITH THIS RELATIVE ADDRESS AS ITS VALUE
C FL4MSK - THE SAVED INDEX OPERATION IS NOT IN EFFECT FOR THIS LIST
C FL5MSK - HEAD OF A CONFLICT LIST
C FLOG7 - 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

0623 END
SUBROUTINE LVFECPI
IMPLICIT INTEGER(A-Z)
REAL*4 CORE, TOP, BOTTOM
LOGICAL*1 SINGBLK, SETUP, NYTRAN, IN1STR, IN2STR, FD1STR, FD2STR, DL1STR,
1 DL2STR, DUMPFI, CURENT, IN2TMP, FD2TMP, DL2TMP, FINDFI,
2 DLETFI, NSRTFI
COMMON /VARGS/ IFUNC, IARG, IPOS, ITYP, IVAL, IVAL(10), IVALP(10), SRCSTR,
1 LINKSUF, SNKSSUP, INSTR,
2 DL2STR ,DUMPFI .C&I N,
3 IN2STR ,FPMXP, DL2STR ,DUi§FI .C&I N,
4 IN2STR, FI STR, FP2STR, DLISTR,
5 DLETFI NSRTFI-
803 COMMON /VAROS/ CURSIG(4), REQSIG(4), LSTSIG(4), MSARET,
1 HREQPG, NTTMSA, HACTPG(2), REACT, USECN1, DIRSIG,
2 DLECN1, OUTLOC, OUTDIR(256), RWBUF(1)
805 COMMON /VMASS/ NWRITE, NWRITE, NWRITE, NWRITE, NVMS, NVMS, NVMS, NVMS,
1 NVMS, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
2 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
3 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
4 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
5 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
6 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
7 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
8 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
9 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
10 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
11 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
12 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
13 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
14 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
15 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
16 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
17 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
18 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
19 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
20 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
21 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
22 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
23 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
24 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
25 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
26 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
27 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
28 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
29 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
30 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
31 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
32 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
33 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
34 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
35 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
36 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
37 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
38 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
39 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
40 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
41 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
42 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
43 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
44 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
45 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
46 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
47 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
48 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
49 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
50 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
51 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
52 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
53 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
54 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
55 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
56 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
57 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
58 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
59 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
60 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
61 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
62 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
63 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
64 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
65 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
66 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
67 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
68 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
69 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
70 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
71 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
72 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
73 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
74 NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE, NWRITE,
0031  BUFSZE = RWBUF(5)
0032  DIRSZE = RWBUF(6)
0033  DRESIZE = RWBUF(7)
0034  INCORE = RWBUF(8)
0035  HDRSZE = RWBUF(9)
0036  HREPOG = RWBUF(10)
0037  BACTPG(1) = RWBUF(11)
0038  BACTPG(2) = RWBUF(12)
0039  REACTCT = RWBUF(13)
0040  BIKSZE = RWBUF(14)
0041  SUPSZE = RWBUF(15)
0042  DIRBLK = RWBUF(16)
0043  PRIKE = RWBUF(17)
0044  SPED = RWBUF(18)
0045  LISTSZ = RWBUF(19)
0046  ISENDZ = RWBUF(20)
0047  DO S I = 1, 4
0048  CURPAG(1) = RWBUF(20+1)
0049  LISTVPG(1) = RWBUF(24+1)
0050  S CONTINUE

C READ IN SAVED USER VARIABLES
0051  DO 7 I = 1, 228
0052  J = I * 28
0053  USER(I) = RWBUF(J)
C USECNT = 1
0054  LEASTV = NTFREE
0055  TOP = INCORE-1
0056  BOTTOM = INCORE
C CORE = TOP/BOTTOM

C READ IN GRN TABLE
0059  LENGTH = 256
0060  NSA = 1
0061  ERRNUM = 30
0062  IERR = IREADW(LENGTH,GRNBL(1),NSA,OLCHAN)
0063  DUMP = 1
0064  IF(IERR .LT. 0) CALL LVVERR(DUMP)

C READ IN IN-CORE DIRECTORY
0066  NSA = 2
0067  LENGTH = DRSIZE
0068  ERRNUM = 31
0069  BUFLOC = 1
0070  CHAN = OLCHAN
0071  CALL LVPRGR(Chan)

C COPY OUT-CORE DIRECTORY TO NEW DISK FILE
0072  DIRPAG = 17
0073  10  DIRPAG = DIRPAG - 1
0074  CALL LVPGRD(OLCHAN)
0075  CALL LVPRWR
0076  IF(DIRPAG .GT. 1) GO TO 10

C COPY OLD GRAPH TO NEW DISK FILE
0078  BUFLOC = DRSIZE + 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  REQPAG(1) = PAGE
0087  REQPAG(2) = CONT
0088  CALL LVMSA(CONNUM)
0089  IF(NEARET. .EQ. 0) GO TO 30
0091  NEA = NEARET
0092  CHAN = CHAN
0093  LENGTH = PAGHDR
0094  CALL LVPAGE(CHAN)
0095  CALL LVPGW
0096  CONTINUE
0097  30  CONTINUE

C EXAMINE IN-CORE DIRECTORY AND BRING IN LISTED CONTINUANTS INTO CORE
0098  DO 50 I = 1,DIRSZE
0099  IF((FNSPC(1).AND.FINSK).EQ.0) GO TO 50
0101  REQPAG(1) = NONSPC(1)
0102  REQPAG(2) = I - REQPAG(1) - 1
0103  IF(REQPAG(2).LT.0) REQPAG(2) = REQPAG(2) + Dirsze
0105  CTRLT = LSTSPC(1)
0106  CTRL1 = CTRLT + HDRSZE
0107  CALL LVMSA(CHAN)
0108  NEA = NEARET
0109  DUPLDC = CTRLT + 1
0110  LENGTH = PAGHDR
0111  CALL LVPAGE(CHAN)
0112  50  CONTINUE

C ESTABLISH REGISTERS
0113  CTRLT = CURTPAG(4)
0114  CTRL1 = CTRLT + HDRSZE
0115  CURRENT = .TRUE.
0116  REQPAG(2) = -2

C READ IN SEQUENCE SPACE (LATER VERSION)
0117  TYPE 68
0118  60  FORMAT('/',' GRAPH HAS BEEN PLACED INTO MEMORY','/)
0119  RETURN
0120  END

105`
THE PURPOSE OF THIS ROUTINE IS TO PROVIDE A SEQUENCE OF 'RANDOM' NUMBERS OF LENGTH LISTSZ TO THE REQUESTED PAGE (CONTINUANT = 0)

PROGRAM INITIALIZATION?

PAUSE 'IN LGRN'

IF(SETUP) GO TO 100

IS THE PAGE DEFINED?

IF(REQPAG(1) .GT. HACTPG(1)) HACTPG(1) = REQPAG(1)

OBTAIN CURRENT VALUES FOR GRN PARAMETERS FOR REQUESTED PAGE

PAGE = REQPAG(1)

GRNX = *(PAGE - 1)

DNOE = GRNTB(GRNEX + OLDMAP)

DNOE = GRNTB(GRNEX + DNOE)

DROH = GRNTB(GRNEX + DROH)

DO 200 J = 1, LISTSZ

200 NODE(1) = OLDMAP + DNOE

IF(NODE(1) .EQ. PAGE) GO TO 199
C RESIDUE GENERATION?
0034 IF(NROW.GT.PRIME) GO TO 150
C ROW UPDATE
0036 NROW = NROW+SEED
0037 IF(NROW.GT.PRIME) NROW = NROW-PRIME
0039 NODE(1) = NROW
0040 DNOPE = NOPE(1)
0041 DNOPE = PRIME+1
C RESIDUE GENERATION?
0042 IF(NODE(1).NE.SEED) GO TO 199
0044 NROW = 0
0045 DROW = PRIME
C RESIDUE GENERATION
0046 150 DROW = DROW+1
0047 NROW = NROW - DROW
0048 NOPE(1) = NROW
0049 DNOPE = NOPE(1)
0050 DNOPE = DROW
0051 IF(NOPE(1).GT.PAGSZ) GO TO 300
C OUTPUT NODE
0053 199 IF(SETUP.EQ..TRUE..) RETURN
0055 200 NODE(1) = NODE(1).OR.(1*FBSH(PAGE,SHPSZ))
C C UPDATE HEADER
0056 250 GRNTBI(GRNDEX + DNOPE) = DNOPE
0057 GRNTBI(GRNDEX + DNOPE) = DNOPE
0058 GRNTBI(GRNDEX + NROW) = NROW
0059 GRNTBI(GRNDEX + DROW) = DROW
0060 LISTSZ = 1
0061 RETURN
C C ORIGINAL CREATION SEQUENCE IS EXHAUSTED, RECOVER IN DEFINED NODES
C BRING IN CONTINUANT ZERO OF THE REQUESTED PAGE IF NECESSARY
0062 300 REQPAG = REQPAG(2)
0063 REQPAG(2) = 0
0064 CALL LVEXCH
0065 DO 430 L = 1,LISTSZ
0066 DO 400 K=1,PAGSZ
0067 LOC + CRITP + 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.PAGSZE) GO TO 440
0075 400 CONTINUE
0076 430 CONTINUE
C BRING IN ORIGINAL REQUESTED (PAGE,CONT)
0077 REQPAG(2) = REQPAG
0078 CALL LVEXCH
0079 LISTSZ = 1
0080 RETURN
C 440 TYPE 450
0082 450 FORMAT(1B,"ERROR...NUMBER OF NODES EXCEEDS REQUESTED MEMORY,"/"
1 PROGRAM IS TERMINATED.")
0083 ERRNUM = 10
0084 DUMP = 0
0085 CALL LVERR(DUMP)
0086 STOP
0087 END

107
SUBROUTINE LVNPAG

IMPLICIT INTEGER(A-Z)

LOGICAL*1 SIGH BK, SETUP, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR, DL1STR,
DL2STR, DUMPFI, CURRENT, I12TMP, FD2TMP, DL2TMP, FINDFI,
2 DLETFI, NISRTFI

COMMON /VPRGS/CURRAG(4), RDEPAG(4), LSTVPG(4), NSARET,
1 HREQPG, NXTHSA, RACTPG(2), READCT, USECNT, DUMPAG,
2 DIRECNT, OUTLOC, OUTDIR(256), HBOFP(1)

COMMON /VMASK/JMPTE, NOTUSFJ, NEWCOM, PINFO, PIPMSP, MASPX, MASKXP

COMMON /VCRNT/ REGASP, CRN I P, CRN I 1, LESTV, NTFREE, FREE, DREGFP,
1 NSA, PAGLOC, CURRENT

COMMON / VBUTF / PAGSIZE, NHCHAN, OI CHAN, CMPAND, PAGHDR, BUFSIZE, DIRSZF,
1 LNCORE, HDRSZF, MSADT, SUPSZF, DLEKSE, DIREB4, PAGNO4

COMMON /VHDFI/ THSHSA, REGAS, PAGENO, CONTNO, INSPFL

COMMON /VSNIT/ SETUP, SIGH BK, NXTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
1 DL1STR, DL2STR, IN1TMP, FD2TMP, DL2TMP, DUMPFI,
2 FINDFI, DLETFI, NISRTFI

COMMON /VPVRAH/ BUFTOC, EMOC, IENH, ERMUP, BINARY, BCP, MODE, PAGES,
1 LUN

COMMON /VVTRI/ NONSPC(:),
2 /VVTRJ/ LSTSPC(:),
3 /VVTRK/ FIPSPC(:)

C THIS ROUTINE WILL PLACE A NEW PAGE (ZERO 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 DEFINE PAGE NO. AND UPDATE HIGHEST ACTIVE PAGE
PAUSE 'IN LVNPAG'
HACTPG(1) = HACTPG(1)+1
HREQPG(1) = HREQPG(1)

C BRING IN OUT-CORE DIRECTORY PAGE AND DEFINE OUTLOC (LOC IN O-C D P)
RDEPAG(2) = -1
CALL LVOPEN
0016 REPAG(2) = 0
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 REQ(P, 0) WAS CREATED AT THE BEGINNING OF THE PROGRAM
0019 CALL LVEXEC
0020 HACTPG(2) = NSARET
0021 RETURN
C OPEN A PAGE-BLOCK IN THE BUFFER
0022 IF(LESTV.EQ.NTFREE) CALL LVOPEN
C NEW PAGE MUST BE ADDED TO THE DISK IMMEDIATELY FOLLOWING THE LAST
C CREATED CONTINUANT.
BACTPG(2) = NXTMSA

C PLACE LOCATION OF NEW PAGE INTO OUT-CORE DIRECTORY
0025 OUTDIR(OUTLOC) = BACTPG(2)
C
C SET UP AVAILABLE SPACE AND HEADER
0026 SNOWBK = .TRUE.
0027 CTRLPT = LSTVPG(4)
0028 CTRL1 = CTRLPT + HDRSZF
0029 CALL LVSETUP

C PLACE EMPTY PAGE ON DISK
0030 LENGTH = PAGEIZE
0031 BPLOC = CTRLPT + 1
0032 ERRNUM = 23
0033 MSN = BACTPG(2)
0034 CALL LVFASH

C UPDATE REGISTERS
0035 CURPAG(1) = BACTPG(1)
0036 CURPAG(2) = 0
0037 CURPAG(3) = BACTPG(2)
0038 CURPAG(4) = LSTVPG(4)

C PAGE HAS BEEN PLACED IN 'LEAST VALUED BLOCK'
C UPDATE IN-CORE DIRECTORY
0039 CALL LVRFLEC

C PROTECT THIS PAGE FROM BEING TAKEN OUT OF CORE BEFORE IT IS USED
0040 F1GSPC(CTRL1+HDRSZ) = NOTUSED OR NEWCON
0041 RETURN
0042 END
SUBROUTINE LVCON
IMPLICIT INTEGER(A-Z)
LOGICAL SING, BK, SETUP, INSTR, IN2STR, FP1STR, FP2STR, DL1STR,
DL2STR, DUMPFI, CURENT, NXTRAN, INSTR, IN2STR, FPISTR, FP2STR,
DLISTR, D2STR, DL2STR, DUMPFI,
COMMON /1VRREGS/ CURPAG(4), REOPAG(4), LSTVPAG(4), MSARET,
BREPAG, MSITMSA, MSITPG(2), REACCT, USECNT, DIRPAG,
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
COMMON /1VMASE/ WRITE, NOTUSP, NEWCON, FPISTR, MASKFP, MASEFP,
COMMON /1VCNT/ REGASP, CTRLPT, CTRLT, LEASTV, NTPFREE, FREE, DFBGSP,
COMMON /1VBUFR/ PFSZE, NWCHAN, 0ICHAN, MPAND, PAGDRA, BUFFSZ, D1RZSE,
COMMON /1VBUFP/ THSWSA, REGASP, PAGNO, CTRLNO, INSHELP,
COMMON /1VSORT/ SETUP, SING, BK, SETUP, INSTR, IN2STR, FP1STR, FP2STR,
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
COMMON /1VREQS/ CURPAG(4), REOPAG(4), LSTVPAG(4), MSARET,
BREPAG, MSITMSA, MSITPG(2), REACCT, USECNT, DIRPAG,
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
COMMON /1VSEQ/ MWRTE, NOTUSP, NEWCON, FPISTR, MASKFP, MASEFP,
COMMON /1VCRNT/ REGASP, CTRLPT, CTRLT, LEASTV, NTPFREE, FREE, DFBGSP,
COMMON /1VSR/ PAGSZE, NWCBAN, 0ICHAN, CMPANP, PAIG0DR, B1JFSE
DIMENSION CONIST(64)
DATA CONIST /64*0/
THIS ROUTINE PLACES AN UNUSED CONTINUANT OF AN ESTABLISHED PAGE INTO THE BUFFER. IF REQ(P,C) WAS NOT INITIALIZED AT THE BEGINNING OF THE PROGRAM, A CONTINUANT IS CREATED AND PLACED ON DISK. REGISTERS, IN-CORE, AND OUT-CORE DIRECTORIES ARE UPDATED. CONIST() IS A LIST OF HIGHEST ACTIVE CONTINUANTS FOR EACH PAGE. PAGE = REOPAG(1)
HICONT = CONIST(PAGE)
CONIST(PAGE) = REOPAG(2)
CALL LVMSA(3)
CALL LVOPEN
C OPEN A PAGE-BLOCK IN THE BUFFER
IF(LEASTV .EQ. NTPFREE) CALL LVOPEN
C ARE ANY PREINITIALIZED CONTINUANTS STILL AVAILABLE?
IF(MSARET .GT. 0) GO TO 10
C*** NEW CONTINUANT MUST BE ADDED TO THE DISK IMMEDIATELY FOLLOWING THE LAST CREATED CONTINUANT.
MSA = NXITMSA
C PLACE LOCATION OF NEW PAGE INTO OUT-CORE DIRECTORY
OUTDIR(OUTLOC + 1) = MSA
C SET UP AVAILABLE SPACE AND HEADER
0025 SINGBK = .TRUE.
0026 CTRLPT = LSTVPG(4)
0027 CTRL1 = CTRLPT + HDRSZE
0028 CALL LVSETP
C
C PLACE EMPTY PAGE ON DISK
0029 LENGTH = PAGHDR
0030 BUFLOC = CTRLPT + 1
0031 ERRNUM = 25
0032 CALL LVPGN
0033 GO TO 46
C
C REQ(P,0) WAS CREATED AT THE BEGINNING OF THE PROGRAM
C READ REQ(P,0) INTO CORE
0034 NSA = NSARET
0035 LENGTH = PAGHDR
0036 BUFLOC = LSTVPG(4) + 1
0037 ERRNUM = 26
0038 CALL LVPGN
0039 CALL LVPGN(CHAN)
C
C UPDATE REGISTERS
0040 CURPAG(1) = REQPAG(1)
0041 CURPAG(2) = REQPAG(2)
0042 CURPAG(3) = NSA
0043 CURPAG(4) = LSTVPG(4)
C
C PAGE HAS BEEN PLACED IN "LEAST VALUED BLOCK"
C UPDATE IN-CORE DIRECTORY
0044 CALL LVPLC
C
C PROTECT THIS PAGE FROM BEING TAKEN OUT OF CORE BEFORE IT IS USED
0045 FRSPC(CTRLPT+HDRF) = NOTUSB .OR. NEWCON
0046 RETURN
0047 END
This routine brings the requested (Page,Cont) into core if necessary and updates the in-core directory and "current" register.

To requested (P,C)
Failure return if MSARET LT 0
C Is REQ(P,C) in core?
D PAUSE 'IN LVEXC'
 TYPE 1
 D1 FORMAT(' REQ(1) REQ(2)')
 D TYPE 2, REQPA(1), REQPA(2)
 D2 FORMAT('2(2X,14)')
0011 MSARET = 10000
0012 CALL LVPrCT
0013 IF(PAGLOC .GT. 0) RETURN
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 gims buffer
0018 CALL LVOPEN
C Bring REQ(P,C) into "least-valued" page-block
0019 ERRNUM = 10
0020 LENGTH = PAGHDR
0021 HSA = MSARET
0022 BUFLOC = LSTVPG(4) + 1
0023 CHAN = NMCCHAN
0024 CALL LVPAGR(CHAN)
C Update "current page" registers
0025 CTRLPT = BUFLOC - 1
0026 CTRL1 = CTRLPT + HDRSZE
0627 PAGLOC = CTRLPT
0628 CURPAG(1) = LSTSPC(CTRLPT + PAGENO)
0629 CURPAG(2) = LSTSPC(CTRLPT + CONTNO)
0630 CURPAG(3) = NS#
0631 CURPAG(4) = CTRLPT
0632 CURSENT = .TRUE.
0633 REGASP = NOISPC(CTRLPT + REGAS)
0634 C UPDATE IN-CORE DIRECTORY
0635 CALL LVRLCLC
C UPDATE (P,C) READER
0636 READCT = READCT + 1
0637 FLSPC(CTRLPT + READI) = READCT
0638 LinksPC(CTRLPT * USECT) = 0
0639 100 LEASTV = NTFREE
0640 RETURN
0640 END
**C**

SUBROUTINE LVDRCT
IMPLICIT INTEGER(A-Z)
COMMON /IVARGS/ IARG,IPOS,IVAL,HVAL,NSKtP,ITESTR,
                 INCLUD,INDXON,IVAL$1(I),IVAL$2(I),SRCSUP,
                 LNKSUP,SNKSUP,INSTYP
COMMON /IVREGS/ CURPAG',4),
                 RQPAG 14),LSTVPG(4),MSARET,
                 HREQPG,NXTNSA,HACTPGI2),READCT,USECNT,DIRPAG,
                 LNKSUF,SNKSTF, INSTYP
COMMON /IVVCNT/ REGAS,PAGENO,CONTNO, INSPEL,
                 USECT, DRFI O ,READVI ,O.IDNPH ,
                 NOPE, NROWJI
COMMON /IVPRAX/ BUTFiOC,LENGTHI,I ERR, ERRNUM, B INARY,BD,
                 MODE, PAME, LUN
COMMON /IVSTAK/ CURLEV,NUNVAR,STACK(1)
COMMON /IVVTRI/ NODSPC(I)
COMMON /IVVTR2/ LSTSPC(I)
COMMON /IVVTR3/ LNKSPC(I)
COMMON /IVVTR4/ FLCSPC(I)
DIMENSION TEMPF(4)

**C**

THIS ROUTINE SEARCHES THE DIRECTORY TO SEE IF THE REQUESTED PAGE-
CONTINUANT (REQPAG) IS IN CORE. THE DIRECTORY IS A PAGE-BLOCK WHICH
STAYS IN CORE. IT IS AT THE FIRST CONTROL POINT (CTRIPT = 0). DIRSZE
IS THE DIRECTORY PAGE SIZE. FOR EACH PAGE-CONTINUANT THAT IS IN
CORE, A TRIPLE IS STORED. THE SOURCE NOPE IS THE CONTINUANT+I, THE
LINK (OR KEY) IS THE PAGE NUMBER, AND THE VALUE IS THE LOCATION
PRECEEDING THE FIRST WORD OF THAT PAGE IN CORE (= CTRIPT).
SUCCESS -- UPDATE CURPAG AND CTRIPT, CURRENT = .TRUE.
FAILURE -- PAGLOC = -1

**C**

*** DOES THE REQUESTED PAGE-CONTINUANT = THE CURRENT PAGE-CONTINUANT ?

**C**

PAUSE IN LVDRCT:  
IF (REQPAG(I) .EQ. CURPAG(I)) AND. (REQPAG(2) .EQ. CURPAG(2)))
1 GO TO 40
IF (INCORE .EQ. 1) GO TO 98
GO TO 50

**C**

GO TO 50

**C**

TEMPORARILY STORE SYSTEM VARIABLES FOR THE SEARCH

**C**
CALL LVSTAC
O1DCPT = CTRI.PT
C
C RESET SYSTEM VARIABLES FOR THE DIRECTORY
C
CTRI.PT = -HDRSZE
CTRI1 = 0
PAGSZ E = D IRSZE
LARG = REQPAG(2)+1
IIFUNC = REQPAG(1)
SRC SUF = LARG
LNK SUF = IIFUNC
CALL LVFINP
PAGIOC = ITEST*IVAL
C
C RESTORE SYSTEM VARIABLES
C
CALL LVPOP
IF(PAGIOC .IT. 0) GO TO 99
C
C PAGE-CONTINUANT HAS BEEN FOUND IN THE DIRECTORY.
CTRI.PT + PAGIOC
CTRI1 = CTRI.PT + HDRSZE
MSA = NONSPC(CTRI.PT + THSNSA)
C
C UPDATE CURPAG
CURPAG(1) = REQPAG(1)
CURPAG(2) = REQPAG(2)
CURPAG(3) = MSA
CURPAG(4) = CTRI.PT
CURRENT = .TRUE.
REGASP = NONSPC(CTRI.PT + REGAS)
RETURN
C
C FAILURE
PAGIOC = -1
CURRENT = .FALSE.
CTRI PT = O1DCPT
CTRI1 = CTRI.PT + HDRSZE
RETURN
END
SUBROUTINE LVMSA(CONNUM)

IMPLICIT INTEGER(4-2)
COMMON /VRSJK/ CURPAG(4), REQPAG(4), LSTVPG(4), MSARET,
  + BREPQG, NXTMSA, BACTPG(2), READCT, USECNT, DIRPAG,
  + DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)

COMMON /VBUFPR/ PAGSZE, NWCHAN, OLCHAN, ORPAND, PAGHDR, BUFSZ, DIRSZ, 
  + INCORE, HORSIZE, MSADIR, SUFSZE, BLSIZE, DIRBLK, PAGBD4

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'

PAGE = REQPAG(1)
CONT = REQPAG(2)
ERRNUM = 11
DUMP = 0
IF('PAGE .GT. 64') OR (CONT .GT. 63) CALL LVERP(DUMP)

C COMPUTE OUTCORE DIRECTORY BLOCK
NEWDIR = (PAGE - 1)/4 + 1
DIRCNT = PAGE - 4*(NEWDIR - 1)

C BRING IN DIRECTORY BLOCK IF NECESSARY
IF(NEWDIR .EQ. DIRPAG) GO TO 100
DIRPAG = NEWDIR
CHAN = NWCHAN
CALL LVVRWR(CHAN)

C DETERMINE IF "ANY", "SPECIFIC", OR "NEW" CONTINUANT IS REQUESTED
100 IF(CONT .LT. 1) 200, 300, 210
C "ANY" -- SET TO ZERO
200 CONT = 0
C "SPECIFIC"

OUTLOC = 1 + 64*(DIRCNT-1) + CONT
MSARET = OUTDIR(OUTLOC)

FORMAT ('REQPAG(1),REQPAG(2),DIRPAG,DIRCNT,CONT,OUTLOC,MSARET')
D
FORMAT ('X,N(25,1S)')

CONNUM = CONT
IF(CONT .LT. 0) GO TO 220
IF(MSARET .GT. 0) GO TO 220
CONT = CONT - 1
GO TO 210
C FAILURE ?
0029 220 IF(CONT .NE. REQPAG(2)) NSARET = -1
0031  RETURN
       C "NEW CONTINUANT" (PAGE MUST BE DEFINED, CONT NOT YET INITIALIZED)
0032 300 NSARET = -1
0033  CONNUM = -1
0034 310 CONT = CONT + 1
0035 IF(CONT .GE. 64) STOP 'REQUEST EXCEEDS ALLOWABLE NUMBER OF CONTINUANTS'
0037 OUTLOC = 1 + 64*(DIRCNT-1) + CONT
0038 NSARET = OUTDIR(OUTLOC)
0039 IF(NSARET .NE. 0) RETURN
0041 CONNUM = CONT
0042  GO TO 310
0043  END
THE PURPOSE OF THIS ROUTINE IS TO MAKE A PAGE-BLOCK AVAILABLE IN WRKSPC.

IF LEASTV = FREE, THEN LSTVPG CONTAINS THE CONTROL POINT FOR AN AVAILABLE PAGE-BLOCK.

TEST WRITE-BIT OF LEAST VALUED CONTINUANT. THE WRITE-BIT INDICATES WHETHER OR NOT A PAGE HAS BEEN CHANGED SINCE IT WAS READ INTO MEMORY.

ZERO OUT FLAGS.

CALL LVPGW.
SUBROUTINE LVSUN

IMPLICIT INTEGER(A-Z)

COMMON /IVREGS/
CUTRPAG(4), REQPAG(4), LSTVPG(4), MSARET,
1 REQPAG, NXTMSA, HACTPG(2), READCT, USECNT, DDIRPAG,
2 DIRCT,OUTLOC,OUTDIR(256), INBUF(1)

COMMON /IVWRSL/ WRITENOTUSD, NEWTYP, PAGHDR, MASKSP, MASKPF

COMMON /IVCRNT/ REGASP, CTRIP, CTIR, LASTV, NTFREE, FREE, DREGSP,
1 NSA, PAGLOC, CURRENT

COMMON /HVBUF/ PAGSIZE, NWCHAN, OLCHAN, CMPAND, PAGHDR, BUFSIZE, DIRSIZE,
1 INCRE, HDRSIZE, WSADIR, SUPSIZE, BLKSIZE, DIRBLK, PAGHDR

COMMON /IVHVDI/ TIESASA, REGAS, PAGEWO, CONTNO, INSVEL,
1 USECT, HDPFO, READVI, OLDIRF, NROWED, DROWE, DROWB

COMMON /IVVTR/ NOSPCE(1)
1 /IVVTR2/ LSTSPC(1)
2 /IVVTR3/ LNSPSC(1)
3 /IVVTR4/ PLSPC(1)

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 LVSUN'

MBIAS = DIRSIZE
C RECOMPUTE USE COUNT OF ALL INCORE CONTINUANTS

DO 10 I = 1, INCORE
USECNT = USECNT + LNSPSC(MBIAS + USECT)

MBIAS = MBIAS + PAGHDR

10 CONTINUE

IF(USECNT .EQ. 0) USECNT = 1

RETURN

END
SUBROUTINE LVALUE
IMPLICIT INTEGER(A-Z)
LOGICAL*1 CURRENT

REAL*4 A, B, C, D, BOTTOM, CORE, ORDER, TOP, TLUSE, USAGE,
VALUE, VALUE, WRITE, SPACE, CAPACY

COMMON /IVREGS/ CURPAG(4), REQPAG(4), LSTVPG(4), MSARET,
HREQPG, NXXMAS, BACTPG(2), READT, USECNT, DIRPAG,
DIRCNT, OUTLOC, OUTDIR(256), WNBUF(1)

COMMON /IVSWIT/ SETUP, SINGBK, RXTRAN, INISTR, IN2STR, FP1STR, FP2STR,
DL1STR, DL2STR, IN2TMP, F2TMP, DL2TMP, DUMPFL,
FINFL, DLETFI, NSRTFI

COMMON /IVSWIT/ SFFUP, SNGI.BK, NXXTRAN, INISTR, IN2STR, FPISTR, FD2STR,
DL1STR, DL2STR, IN2TMP, FD2MP, DL-TP, DJMPFI,
FINPFI, DLETFI, NSRTFI

COMMON /IVSWIT/ MITE, NOTUSD, NEWCON, FIPMK, ASKSF, MASKPF

COMMON /IVSWIT/ REGASP, CTMPLT, CTMI, LEASTV, NTFREE, FREE, DREGSP,
MSA, PGLOC, CORE

COMMON /IVSVT/ USPCT, HDRFIG, READVI, OLDMIN, DSORED, SGRW, DROMU

COMMON /IVSVT/ BUFLOC, LENSOH, IERR, ERRNUM, BINARY, BCP, NODE, PAGES,
LUN

COMMON /IVSVT/ BUFI.OC, LENTHI, IERR, ERRNUM, BINARY, PAGES,
LUN

COMMON /IVSVT/ BUFI.OC, LENTHI, IERR, ERRNUM, BINARY, PAGES,
LUN

COMMON /IVSVT/ BUFI.OC, LENTHI, IERR, ERRNUM, BINARY, PAGES,
LUN

DATA A, B, C, D /15.0, 20.0, 15.0, 50.0/

THE VALUE RANGE FROM 0 (LEAST NEEDED CONTINUANT) TO 100 (MOST USEFUL CONTINUANT).

D PAUSE 'IN LVALUE'

JBIAS = DIRSZE
MBIAS = JBIAS
IF(INCORE .EQ. 1) GO TO 20
LEASTV = NTFREE
DO 10 I = 1, INCORE
IF((FIPS(JBIAS + HDRFIG) .AND. NOTUSD) .NE. 0) GO TO 9
D PAUSE 'IN LVALUE LOOP'

C CALCULATE ORDER VALUE

INPOS = FIPS(JBIAS + HDRFIG) .AND. NOTUSD .NE. 0) GO TO 9

C CALCULATE WRITE VALUE

C CALCULATE USAGE VALUE

TTLUSE = USECNT
USAGE1 = LNSPC(JBIAS + Usect)
USAGE = USAGE1/TTLCSE

C CALCULATE SPACE VALUE
TOP1 = PAGSZF - LNSPC(JBIAS + INSDL)
BOTOM1 = PAGSZF
CAPACY = TOP1/BOTOM1
GO TO (1,2,3), RNTYI

C CREATION TYPE RUN
0038 SPACE = -4.0*CAPACY*(CAPACY-1.0)
0039 GO TO 5

C PRODUCTION TYPE RUN
0040 SPACE = 1.0
0041 IF( CAPACY .LT. .125) SPACE = 6.0*CAPACY
0043 IF( CAPACY .GE. .750) SPACE = 0.0
0045 IF((CAPACY .GT. .375).AND.(CAPACY .LT. .75))

1 SPACE = 2.0-(8.0/3.0)*CAPACY
0047 GO TO 5

C QUERY TYPE RUN
0048 SPACE = 1.0-CAPACY

C*** CALCULATE THE IN-CORE VALUE OF THIS CONTINUANT
0049 S = VALUE = A*ORDER+B*USAGE+C*SPACE+D*WRITE
0051 TYPE 111,ORDER,USAGE,SPACE,WRITE,VALUE1,VALUE
0053 IF((VALUE1 .GE. VALUE) GO TO 9
0052 LEASTV = FREE
0054 VALUE = VALUE1
0055 MBIAS = JBIAS
0056 9 JBIAS = JBIAS+PAGEOFF
0057 LEASTV = FREE
0058 RETURN

C UPDATE 'LEAST VALUED' REGISTER LSTVPGL UNTL NONE WAS FOUND
0059 IF(LEASTV .EQ. FREE) GO TO 20
0060 15 TYPE 15
0060 15 FORMAT(/'*** ERROR IN LVALUE ***'/)

C PREVENT POSSIBLE RECURSION
0061 IF(DUMPF1 .EQ. TRUE) STOP
0063 ERR = 0
0064 ERRNUM = 20
0065 MODE = BCP
0066 PAGES = -1
0067 DUMP = 0
0068 CALL LVDDUMP(DUMP)
0069 STOP

C LSTVPG(1) = LSTSPC(MBIAS + PAGENO)
LSTVPG(2) = LSTSPC(MBIAS + CONTNO)
LSTVPG(3) = NPSPC(MBIAS + THERSA)
LSTVPG(4) = MBIAS

C SET LEASTV FOR INCORF = 1
0074 LEASTV = FREE
0075 RETURN
0076 END
C C
SUBROUTINE LVRPLC
IMPLICIT INTEGER(A-Z)
LOGICAL*1 CURRENT, REORG, FULL, RPLACE
COMMON /VAROS/ IFUNC, IARG, IPOS, ITYP, IWAL, NWAL, NSKIP, ITESTR.
1 INCLUDE, INDXON, IVALS(10), ITYPE(10), SRCUSF,
2 LINKUS, SMKUSF, INSTYP
COMMON /VARUS/ CURPAG(4), REOPAG(4), LSTVPG(4), MSARET,
1 HREOPG, NXTMSA, HACHP(2), READC, USECNT, DIRPAG,
2 DIRECT, OUTLOC, OUTDIR(256), RBUF(1)
COMMON /VCRT/ REUASP, CTRIPT, CTRL1, LEASTV, NTFREE, FREE, DREOSP,
1 MSA, PAGLOC, CURRENT
COMMON /VBUR/ PAGSZE, NCPLAN, OLCBAN, CHPAND, PAGHDR, BUFPSZE, DIRSZE,
1 INCORE, HDRSZE, MSADIR, SUPSZE, BKLSZE, DIRBK, PAGHD4
COMMON /VPEL/ NUMET
COMMON /VINS1/ REORG, FULL, RPLACE
COMMON /VVRTR/ NDDSCPC(1)
1 /VVRTR2/ LSTSPC(1)
2 /VVRTR3/ LNKSPC(1)
3 /VVRTR4/ LGSPC(1)

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 B
PAUSE 'IN LVRPLC'
CALL LVSTAC

C C*** SAVE SYSTEM VARIABLES
C B
PAUSE 'IN LVRPLC'
CALL LVSTAC

C C*** DELETE OLD PAGE, CONTINUANT, LSTVPG, FROM DIRECTORY
C
CTRL1 = -HDRSZE
CTRL1 = 0
PAGSZE = DIRSZE
IF(LSTVPG(1).EQ.0) GO TO 5
IARG = LSTVPG(2) + 1
IARG = LSTVPG(1)
SRCUSF = IARG
LINKUSF = IFUNC
CALL LVFINP
ITYP = 1
IPOS = 1
INDXON = 0
REORG = .FALSE.
NUMET = 0
TEMP = REUASP
REUASP = DREOSP
CALL LVPLET

C C*** PLACE NEW PAGE-CONTINUANT INTO DIRECTORY
C
IARG = CURPAG(2) + 1
IFUNC = CURPAG(1)
SRCUSF = IARG
C** RESTORE SYSTEM VARIABLES
C
0042 REGSP = REGASP
0043 REGASP = TEMP
0044 CALL LVPOP
0045 CRIPTR = CURPAG(4)
0046 CRIPTR = CRIPTR + HDRSIZE
C
C PAGE-BLOCK HAS BEEN FILLED AND IS NO LONGER AVAILABLE
0047 LEAST = MTFREE
0048 RETURN
0049 END
SUBROUTINE LVSTAC

DIMENSION STORE(1), STOR(6)

COMMON /IVARGS/ IFUNC, IARG, IPOS, IITYP, IVAL, NVAL, NSkip, ITESTR,
1 INCLUD, INDEXON, IVALS(10), IITYP(10), SRCSUP,
2 LINSUP, SNSUP, DNSTYP

COMMON /IVRE43S/ LRPAGE4, REQPAG4, LSTVPG4, WSARET,
1 BREQGP, NKINSZA, NACTPG(2), READCT, USECT, DIRPAG,
2 DIRCT, OUTLOC, OUTDIR(256), RKBUP(1)

COMMON /IVBUFF/ PAGSZE, NCHAN, OCHAN, CMPAND, PAGHDR, BUFSZE, DIRSZE,
1 INCORE, HDSZE, MSADIR, SUPSZE, BKLSZE, DIRBK, PAGHDR

COMMON /IVPRAM/ BUFLOC, LENGTH, IERR, ERRNUM, BINARY, BCP, MOVE, PAGES,
1 LIN

COMMON /IVADDR/ LADD, THIS, LSTHEC, LOC, LAST, LASTLC

COMMON /IVSTAK/ CURLEV, NUMVAR, STACK(140)

EQUIVALENCE (IFUNC, STORE(1))
EQUIVALENCE (IADD, STORE(1))
EQUIVALENCE (IADD, STORE(1))
EQUIVALENCE (NUMVAR, MAXLEV, CURLEV/34, 3, 0/

C THIS ROUTINE SAVES UP TO 3 SETS OF /IVARGS/ VARIABLES AND REQUEST REGISTERS

C PAUSE 'IN LVSTAC'

IF(CURLEV .GT. MAXLEV) GO TO 99
ISTLOC = (CURLEV-1)*NUMVAR + 11 + 1
DO 10 I = 1, NUMVAR
STACK(I+ISTLOC) = STORE(I)
10 CONTINUE
DO 20 I = 1, 4
STACK(NUMVAR+ISTLOC+I) = REQPAG(I)
20 CONTINUE
DO 30 I = 1, 6
STACK(NUMVAR+ISTLOC+I+4) = STOR(I)
30 CONTINUE
STACK(NUMVAR+ISTLOC+4) = PAGSZE

RETURN

C FAILURE - ATTEMPT TO STACK TOO MANY SETS OF VARIABLES

ERRNUM = 21
DUMP = 1
CALL LVERR(DUMP)
RETURN
END
SUBROUTINE LVPOP

IMPLICIT INTEGER(A-Z)

DIMENSION STORF(1),STORF(6)

COMMON /IVARGS/ IFUNC,LARG,IPos,ITYP,IVAL,IVAL,WSKIP,ITESTR.
1  INCLUDE, INPMON, IVALS(16), ITYP1(16), SWSKIP.
2  LINKSUF, SWSKIP, INSTYP

COMMON /IVREGS/ CURFAG(4), REQPAG(4), LSTVPAG(4), WSARET.
1  HRQPG, NTHMSA, HACTPG(2), READCT, USECNT, DURPAG.
2  DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)

COMMON /IVBINF/ PAGESIZE, NWCHAN, OWCHAN, CHPAG, PAGEFU, BUFSIZE, DIRSIZE.
1  INCOMP, HDERSIZE, WSADIR, SYPSIZE, BUFSIZE, DIRBIB, PAGEFU

COMMON /IVPAM/ BUFSIZE, LENGTH, ERRNUM, BINARY, BCP, MODE, PAGES.
1  LUN

COMMON /VADDR/ IAADD, THISH, LSTMED, LOC, LASTLOC

COMMON /IVSTAR/ CURLEV, NUMVAR, STACK(1)

EQUIVALENCE (IFUNC,STORE(1))

EQUIVALENCE (IAADD,STOR(1))

C THIS ROUTINE RETURNS UP TO 3 SETS OF /IVARGS/ VARIABLES

D PAUSE 'IN LVPOP'

10  IF(CURLEV .LT. 0) GO TO 99
11  ISTLOC = (CURLEV)*(NUMVAR+1)+1
12  DO 10 I = 1,NUMVAR
13  IF(ISTLOC .LT. I) GO TO 10
14  STORE(I) = STACK(I+ISTLOC)
15  CONTINUE
16  10  CONTINUE
17  DO 20 I = 1,4
18  IF(REQPAG(I) .LT. ISTLOC) GO TO 20
19  IF(REQPAG(I) .GT. ISTLOC+NUMVAR) GO TO 20
20  CONTINUE
21  CONTINUE
22  DO 30 I = 1,6
23  IF(RFQPAG(I) .LT. ISTLOC+NUMVAR) GO TO 30
24  IF(RFQPAG(I) .GT. ISTLOC+NUMVAR+4) GO TO 30
25  CONTINUE
26  CONTINUE
27  CONTINUE
28  CONTINUE
29  DO 30 I = 1,11
30  CONTINUE
31  CONTINUE
32  CONTINUE
33  CONTINUE
34  CONTINUE
35  CONTINUE
36  CONTINUE
37  CONTINUE
38  CONTINUE
39  CONTINUE
40  CONTINUE
41  CONTINUE
42  CONTINUE
43  CONTINUE
44  CONTINUE
45  CONTINUE
46  CONTINUE
47  CONTINUE
48  CONTINUE
49  CONTINUE
50  CONTINUE
51  CONTINUE
52  CONTINUE
53  CONTINUE
54  CONTINUE
55  CONTINUE
56  CONTINUE
57  CONTINUE
58  CONTINUE
59  CONTINUE
60  CONTINUE
61  CONTINUE
62  CONTINUE
63  CONTINUE
64  CONTINUE
65  CONTINUE
66  CONTINUE
67  CONTINUE
68  CONTINUE
69  CONTINUE
70  CONTINUE
71  CONTINUE
72  CONTINUE
73  CONTINUE
74  CONTINUE
75  CONTINUE
76  CONTINUE
77  CONTINUE
78  CONTINUE
79  CONTINUE
80  CONTINUE
81  CONTINUE
82  CONTINUE
83  CONTINUE
84  CONTINUE
85  CONTINUE
86  CONTINUE
87  CONTINUE
88  CONTINUE
89  CONTINUE
90  CONTINUE
91  CONTINUE
92  CONTINUE
93  CONTINUE
94  CONTINUE
95  CONTINUE
96  CONTINUE
97  CONTINUE
98  CONTINUE
99  ERRNUM = 22
100  DUMP = 1
101  CALL LVERR(DUMP)
102  RETURN
103  END
SUBROUTINE LVRFOR(RECON)
IMPLICIT INTEGER(A-Z)
LOGICAL* SNGIX, SFTUP, NXTRAN, IN1STR, IN2STR, FP1STR, FP2STR, DL1STR,
   1 D2STR, DUMFIP, CURRENT, FINDIP, DLETFT, NSTRFT, FD1TMP,
   2 DL2TMP, IN2TMP, FP2TMP, INSIDE, FULL, REORG, LSTCON, REPLAC,
   3 FINISH

COMMON /TVARS/ IFUNC, IARG, IP1OS, ITYP, IVAL, NVAL, XVAL, ITESTR,
   1 INCLUDE, IMFKNX, IVALS(10), ITYP(110), SRSFIP,
   2 LNKSUF, SNSKSF, INSTYP

COMMON /VRFGS/ RFQPAG(4), RFQPAG, LSTVPG, MSA4RF, HRFQPG,
   1 NXTRAN, RFADCT, USFCNT, DIRPAG, DIRCNT, OUTLOC, OUTDIR,
   2 DUMPFI, FULL, RFORG, STCON, RPLAC

COMMON /VMASK/ WRITE, NOTUSD, NEWCON, FI SNSK, MASKSF, MASKTF

COMMON /VFAGS/ F1 (10), F2MSK(10), F3MSK(10), F4MSK(10), F5MSK(10),
   1 F6MSK, F7MSK, F8MSK, F9MSK, F10MSK, F11MSK, F12MSK, F13MSK, F14MSK

COMMON /VMASK/ M8RITE, NOTUSJ, NFWSNF, F1GMSK, F1NSKSF, MASKPF

COMMON /VFAGS/ F1, F2MSK(10), F3MSK(10), F4MSK(10), F5MSK(10),
   1 F6MSK, F7MSK, F8MSK, F9MSK, F10MSK, F11MSK, F12MSK, F13MSK, F14MSK

COMMON /VCRNT/ RFORG, RPLAC, CTRIPT, CTRI, LEASTV, NTFREE, FREE, DREPSP,
   1 WA, PAGLOC, CURRENT

COMMON /VFBFR/ F1SFRE, NCHAN, OI CHAN, CMPAND, PAGHDR, BUFSP座位, DIRSZF,
   1 INCFR, HDRSZE, SADIR, STFSPF, BLKSF, DIRBF, PAGF04

COMMON /VHBFR/ F1SF, REGAN, PAGEFQ, CONTRO, INSFPL,
   1 URFNT, HDRF1F, READVI, OIP DMPH, DSONEH, NROWH, DROUN

COMMON /VTWNT/ SETUP, SNSGFI, NYTRAN, IN1STR, IN2STR, FD1STR, FD2STR,
   1 D2STR, D1NSTR, IN2TMP, FP2TMP, DLSTMP, DUMFIP,
   2 F1NDF, DLETFT, SFTIP

COMMON /VRFQR/ BVFLC, LENGTH, IFER, ERRNUM, BINARY, BCD, MOD, PAGS,
   1 LIV

COMMON /VAPBY/ L1ND, TH1N, LSTTH, LOC, LAST, LSTLC

COMMON /VF1S/ COUNT, ABSPOS, LSTCON

COMMON /VCRNT/ RFORG, RFORG, FULL, REPLAC

COMMON /VF1S/ 1OSF(10), S1OSF(10),
   1 /VF1S2/ LSTSPC(*)
   2 /VF1S3/ LNKSF(*)
   3 /VF1S4/ FUSPC(*)

C RFORG MOVFS: THE LIST FROM ITS PRESENT LOCATION (CONTINUANT = RFQPAG(2))
C TO CONTINUANT RECON.

CP ERRNUM = 62
CP DUMP = 1
CP ORDER = FO (TRMPAG(2)) CALL LVFRR (DUMP)
C WAS THE ORIGINAL LIST PLACED ON (TRMPAG(2)) AS A RESULT OF A SPECIFIC
C REQUEST? IF SO, ERROR.
CP TYP = 10, SNSGFI, NYTRAN, IN1STR, IN2STR, RFQPAG(4), RFQPAG, CURMPAG(2)
CP DUMP = 10, FORMAT = *** ERROR *** THE FOLLOWING TRIPLE: '13..08.23',
CP OF PAGE '13..13..15' WILL BE PLACED ON CONTINUANT '13..13..13',
CP DUMP = 20, ORDER = (TRMPAG(2))

C RFORG MOVFS: THE LIST FROM ITS PRESENT LOCATION (CONTINUANT = RFQPAG(2))
C TO CONTINUANT RECON.

CP ERRNUM = 62
CP DUMP = 1
CP ORDER = FO (TRMPAG(2)) CALL LVFRR (DUMP)
C WAS THE ORIGINAL LIST PLACED ON (TRMPAG(2)) AS A RESULT OF A SPECIFIC
C REQUEST? IF SO, ERROR.
CP TYP = 10, SNSGFI, NYTRAN, IN1STR, IN2STR, RFQPAG(4), RFQPAG, CURMPAG(2)
CP DUMP = 10, FORMAT = *** ERROR *** THE FOLLOWING TRIPLE: '13..08.23',
CP OF PAGE '13..13..15' WILL BE PLACED ON CONTINUANT '13..13..13',
CP DUMP = 20, ORDER = (TRMPAG(2))

CP ERRNUM = 62
CP DUMP = 1
CP ORDER = FO (TRMPAG(2)) CALL LVFRR (DUMP)
C WAS THE ORIGINAL LIST PLACED ON (TRMPAG(2)) AS A RESULT OF A SPECIFIC
C REQUEST? IF SO, ERROR.
CP TYP = 10, SNSGFI, NYTRAN, IN1STR, IN2STR, RFQPAG(4), RFQPAG, CURMPAG(2)
CP DUMP = 10, FORMAT = *** ERROR *** THE FOLLOWING TRIPLE: '13..08.23',
CP OF PAGE '13..13..15' WILL BE PLACED ON CONTINUANT '13..13..13',
CP DUMP = 20, ORDER = (TRMPAG(2))
OLDCPT = CTRIP
OLDCPI = CTII
C SPECIAL HANDLING IF BUFFER HOLDS ONLY ONE CONTINUANT
0028 IF(INCORE .EQ. 1) GO TO 30
C KEEP CONTINUANT WITH OLD LIST IN CORE
0030 FIGSPC(CTRIP+HDRFIG) = FIGSPC(CTRIP+HDRFIG) .OR. NOTINS
C LOCATE NEW CONTINUANT
0031 IFDOCAG(2) .EQ. RFOCON
0032 CALL LVFIN
0033 NEWCPT = CTRIP
0034 NEWCP1 = CTII
C SAVE THE VALUE TO BE INSERTED
0035 CALL LVSTAC
C
0036 IF(LSTHEF .GT. 0) GO TO 50
C SVI
0037 KVAL = 1
0039 IVALS(1) = LSTSPC(OI DCP1+THIS)
0040 ITYPE(1) = FIGSPC(OI DCP1+THIS) .AND. FIG67
0041 IF(INCORE .GT. 1) GO TO 45
0043 IFDOCAG(2) .EQ. RFOCON
0044 CALL LVFIN
0045 CALL LVFIN
0046 CALL LVFIN
0047 FIGSPC(CTRIP+FIG67) .OR. WRITE
0048 FIGSPC(CTRIP+HDRFIG) = FIGSPC(CTRIP+HDRFIG) .OR. WRITE
0049 GO TO 70
C SNI
0050 IF ILOC = LSTHEF
0051 KVAL = 0
0052 MVAL = 0
0053 IF(INCORE .EQ. 1) GO TO 63
0055 KVAL = KVAL .EQ. 1
0056 IVALS(1) = SOSPCC(OI DCP1+OI LOC)
0057 ITYPE(1) = FIGSPC(OI DCP1+OI LOC) .AND. FIG67
0058 CALL LVFIN
0059 CALL LVFIN
0060 IF ILOC = LSTSPC(OI DCP1+OI LOC)
0061 IF FIGSPC(OI DCP1+OI LOC) .AND. FIG67 .EQ. 0) GO TO 60
0063 GO TO 70
C FOR MVI (INCORE .EQ. 1) SWAP OLD AND NEW CONTINUANTS IN AND OUT WHILE
C RE-INSERTING UP TO TEN VALUES AT A TIME
0064 FINISH = FALSE
0065 FINISH = FALSE
0066 CALL LVFIN
0067 IFDOCAG(2) .EQ. RFOCON
0069 MVAL = MVAL .EQ. 1
0070 IVALS-KVAL = SOSPCC(OI DCP1+OI LOC)
0071 ITYPE-KVAL = FIGSPC(OI DCP1+OI LOC) .AND. FIG67
0072 IF ILOC .NE. FIGSPC(OI DCP1+OI LOC)
0073 IF FIGSPC(OI DCP1+OI LOC) .AND. FIG67 .NE. 0) FINISH = TRUE
0076           IF (.NOT. Finish) G0 T0 66
0078  67       RFQPAG(2) = RFQCON
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. WRITE
0086           IF (.NOT. Finish) G0 T0 65
0087           KVAL = NVAL
0088           RFQPAG(2) = OIDCON
0089           CALL LVEXCH

C
C DELETE OLD LIST
0090  70       INDEX = 0
0091           CTRIPT = OI DCPT
0092           CTRI1 = OI DCPT
0093       IF (.INCRG .GT. 1) G0 T0 75
0094           RFQPAG(2) = OIDCON
0095           CALL LVEXCH
0096  75       CTRIPT = NGCPT
0097           CALL LVFIND
0098           CALL LVNSRT

C
C UPDATE HEADER
C OLD CONTINUANT HAS BEEN MODIFIED
0100       FIGSPC(CTRIPT+HDRFIG) = FIGSPC(CTRIPT+HDRFIG) .AND. (.NOT. NOTUSD
0101           1).OR. WRITE

C
C RESET CONTINUANT USAGE RATIO
0102       LNKSPC(0I DCPT+INSDEL) = LNKSPC(0I DCPT+INSDEL) - KVAL

C
C INSERT NEW VALUE
0103           CALL LVPOP
0104       RFQPAG(2) = RFQCON
0105           IF (.INCRF .EQ. 1) CALL LVEXCH
0106           CTRIPT = NEWCPT
0107           CTRI1 = NEWCPT
0108           CALL LVFIND
0109           CALL LVNSRT

C
C UPDATE HEADER
C RESET CONTINUANT USAGE RATIO
0110       LNKSPC(NEWCPT+INSDEL) = LNKSPC(NEWCPT+INSDEL) - KVAL
0111           RETURN
0112           END
SUBROUTINE LVNIC

IMPLICIT INTEGER(A-Z)
LOGICAL*1 SINGBK, SETUP, INSTR, IN2STR, FD1STR, FD2STR, DL1STR,
1 DLSSTR, DUMPFL, CURRENT, FINDFL, DLETFL, NSKIPFL,
2 DLSTRMP, IN2STRMP, FD1STRMP, FD2STRMP, REORG, FULL, REPLACE

COMMON /VARGS/ IFINC, IARG, IPOS, ITYP, IVAL, INVAL, NSKIP, ITESTR,
1 INCLUDE, INDXON, IVALS(10), ITYPF(10), SRCSUF,
2 LNASKUF, SNKSUF, INSTR

COMMON /VRFOS/ CHPAG(4), RFOG(4), LSTVG(4), MSARET,
1 HREOOG, MXTRSA, HACTPG(2), READCT, USECNT, DIREAG,
2 DIREAG, OUTLOC, OUTDIR(256), RWBUF(1)

COMMON /VWASK/ MRITIT, NOTUSP, NEWCNO, FLGMSK, MASKSF, MASKPF

COMMON /VFJAG/ FI 0MSK, FI 1MSK, FI 2MSK, FI 3MSK, FI 4MSK, FI 5MSK, FI 6MSK.
1 FI AGS, FI AG10, FI AG11, FI AG12, FI AG13, FI AG14,
2 FLAG16

COMMON /VCRNT/ REGASP, CRTP, CTRI1, LEASTV, NTFRE, FREE, DREGSP,
1 MSA, PAGI OC, CURFNT

COMMON /VFUPR/ PAGSZ, NCHON, OLCCHAN, CMFAND, PAGHDR, BUFSIZE, DLRSIZE,
1 INCSEG, HDRSZ, MSADIR, SUFFSZE, BLKSIZE, DREM K, PAGE4

COMMON /VMDVN/ THMSRA, REGAS, PAGENO, CONTNO, INSPFL,
1 USECT, HDRPG, READV1, OL DNP, DNGDEH, NWPMP, DNOFP,
2 DLSSTR, DL2STR, DL3STR, FD1STR, FD2STR,
3 DLSTRMP, IN2STRMP, FD1STRMP, FD2STRMP, DUMPFL,
4 FINDFL, DLETFL, NSKIPFL

COMMON /VPRAM/ BUFLOC, LENGTH, ERRNUM, BINARY, BCD, MODE, PAGES,
1 LUN

COMMON /VADDR/ 1ADD, THSVID, LOC, LAST, LASTLC

COMMON /VVTR/ NOITSC(1)
1 /VVTR3/ LSTSPC(1)
2 /VVTR1/ NSPSC(1)
3 /VVTR4/ FSPEC(1)

C THIS ROUTINE SEARCHES THE LIST TO FIND THE VALUE IN INCLUD. IF IT
C IS FOUND, ITS POSITION WRT THE TOP OF THE LIST IS RETURNED.
C DOES THE LIST EXIST ?
D PAUSE 'IN LVNIC'

IPOS = 1
RFOG(2) = -2
J = 0
CALL LVFPEX(J,3,J,J,J)
IF (ITESTR .lt. ) GO TO 31
IF (IVAL .eq. INCLUD) GO TO 25
IF (LSTHEA .lt. ) GO TO 31

C NOT FOUND
JVAL = IVAL
KSKIP = NSKIP
NSKIP = 0
INDEX = 0
INDEXD = 0
KFUNC = 0
KARG = 0
SAGCON = 0
0034 KPOS = 0
0035 10 KPOS = KPOS + 1
0036 IPOS = KPOS
0037 CALL LVDFEX(INP, INPMAD, KFUNC, KARG, SAVCON)
0038 IF(ITESTR .LT. 0) GO TO 36
0039 IF(IVAL .NE. INCLUD) GO TO 10
C
C EXIT FROM LOOP, INCLUD = 1, SUCCESS
C INCLUD = -1, FAILURE
C EXCEPT FOR IPOS, OUTPUT MUST APPEAR AS IF IT WAS FROM LVFINP
0042 30 IVAL = JVAL
0043 NSKIP = KSKIP
0044 LOC = LSTHED
0045 25 INCLUD = ITESTR
0046 ITESTR = 1
0047 RETURN
C
C SVI FAILURE RETURN
0048 31 INCLUD = -1
0049 RETURN
0050 END
This routine handles continuant overflow on indexed insertion.

C SUBROUTINE LVOVER
C
C IF FUNCTION DOES NOT EXIST, RETURN AND PLACE ON NEXT CONTINUANT
C
0017 IF(ITESTR .IT. 0) RETURN
C
C PICK UP LAST VALUE ON LIST,
C
0019 IF(LSTHED .GT. 0) GO TO 50
C
C SVI
C
0021 NVAL = 2
C
0022 IVALS(2) = IVALS(1)
C
0023 ITYP(2) = ITYP(1)
C
0024 TMIVAL = LSTSPC(CTRL + THIS)
0025 TMISP = FLSPC(CTRL + THIS) .AND. FL067
C
C NVI
C
C SAVE THE VALUE TO BE INSERTED
C
0027 GO TO 100
C
C CALL LYSTAC
C
C COMMON /IVARGS/ IFUNC, IARG, IPOS, ITYP, IVAL, NVAL, NSKIP, ITESTR,
C
C COMMON /IVREGS/ CUTRYAG44, REQPAG44, LSTVPC44, NSARET,
C
C COMMON /IVFAG/ FIOFSK, FINSK, FINSK, FINSK, FINSK, FL067,
C
C COMMON /IVADDR/ IADD, THIS, THISTHED, LOC, LASTL, LSTCON,
C
C COMMON /IVVTR1/ NSPC(1)
C
C COMMON /IVVTR2/ LSTSPC(1)
C
C COMMON /IVVTR3/ LSTSPC(1)
C
C COMMON /IVVTR4/ FLSPC(1)
C
0028    TMPVAL = NODSPC(Ctrl-1-LASTLC)
0029    TMPTYP = FISPC(Ctrl-1-LASTLC). AND. FIG67
0030    C DELETE LAST VALUE,
0031     INDEXN = 1
0032     IPOS = -1
0033     CALL LVPLET
0033     C RESET CONTINUANT USAGE RATIO
0034     LNSPC(CTRL+INSDEL) = LNSPC(CTRL+INSDEL) - 1
0034     IF'LSTHEQ .LT. 0) GO TO 150
0036     C INSERT ORIGINAL VALUE,
0036     CALL LVPOP
0037     CALL LVNSRT
0038     C INSERT LAST VALUE ON NEXT CONTINUANT
0038     IVALS(1) = TMPVAL
0039     ITYP(1) = TMPTYP
0040     INDEXN = 0
0041     RETURN
0042     END
0001 C
0002 C SUBROUTINE LVDPUMP (DUMP)
0003 IMPLICIT INTEGER (A-Z)
0004 LOGICAL * CURRENT, SETUP, SNGL BK, NVTRAN, INI1STR, INI2STR, FD1STR, FD2STR,
0005 DL1STR, DL2STR, IN12TMP, FD12TMP, DL22TMP, DUMPFI,
0006 FNDPFI, DLETFL, NSRTFl
0007 COMMON /LREGS/ CURPAG(4), REUPAG (4), LSTRPAG(4), MSARET,
0008 HDRPAG, NSWMSA, HACTPO (2), READCT, USECNT, DIREPAG,
0009 DIRCNT, OUTLOC, OUTDIR (256), RBUF (1)
0010 COMMON /VMASK/ WRITE, NUSPD, NNEWCN, PWSK, MASKEP, MASKFF
0011 COMMON /VCRENT/ REGAS, CTRLPT, CTRL1, LEASTV, NLFREE, FREE, DREMSP,
0012 MSA, PAGLOC, CURRENT
0013 COMMON /NBUFFER/ PACSZ, NWCHAN, OCHAN, CMPAND, PAGBDR, BUFŞZE, DİRSZB,
0014 INCORE, HDRSZF, MSADIR, BUFŞZE, BLKSZE, DIRBlK, PAGBD4
0015 COMMON /THSD/ THSNSA, REGAS, FADNO, CONTNO, INSHPL,
0016 USECT, HDRFIG, READV1, OI DUMP, DNOPEH, NROWH, DROWN
0017 COMMON /VSF/ SETUP, SNGL BK, NVTRAN, INI1STR, INI2STR, FD1STR, FD2STR,
0018 DL1STR, DL2STR, IN12TMP, FD12TMP, DL22TMP, DUMPFI,
0019 FNDPFI, DLETFL, NSRTFl
0020 COMMON /LVPRM/ BUFI0C, LENGTH, IERR, ERRNUM, BINARY, BPWD, MODE, PAGES,
0021 LUN
0022 COMMON /ISEO/ SSEQCH, ISOSOS, LSNSTO, SEQSPC(1)
0023 COMMON /LVTR/ NODSPC(1)
0024 1 /LVTR2/ LSTSPC(1)
0025 2 /LVTR3/ INKSPC(1)
0026 3 /LVTR4/ FLSGPC(1)
0027 EQUIVALENCE (LUN, CHAN)

C THIS ROUTINE HAS TWO MODES OF OPERATION:
A) BCP DUMP - OUTPUTS SYSTEM VARIABLES IN INTERNAL STRUCTURE OF GIRS
   PAGES AS A DEBUGGING AID
B) BINARY DUMP
   COPIES ALL MODIFIED IN-CORE CONTINUANTS TO THE NEW DISK FILE.
   A CALL TO LV PRM SAVES THE SYSTEM PARAMETERS.
   C) PAGES = -1, DUMP IN-CORE PAGE-BLOCKS
   C) PAGES = 0, DUMP ALL PAGES w/ ALL CONTS
   C) PAGES = N, DUMP PAGE N
C IS THIS A BINARY OR BCP DUMP?
D PAUSE 'IN LVDPUM()' IF/NODE .E0. BINARY) GO TO 100

C*** BCP
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 DUMPFI = TRUE,
0020 CALL LVPRM(DUMP)
0021 WRITE (LUN, 21)
0022 21 FORMAT (//)
0023 WRITE (LUN, 21)
C WRITE IN-CORE DIRECTORY

133
WRITE(LUN, 35)
FORMAT(' IN-CORE DIRECTORY', //)
WRITE(LUN, 45)
FORMAT(' CONTINUANT 'IOC-1' LNKSPC FLSPC', //)
NBIAS = -HDRSZ
NUMBLK = 1
CALL LVWRIT(NBIAS, NUMBLK)

C TYPE OF DUMP ?
IF(PAGES .GE. 0) GO TO 50
WRITE OUT ONLY THOSE CONTINUANTS THAT ARE IN CORE
NBIAS = DIRSZ
NUMBLK = INCORE
CALL LVWRIT(NBIAS, NUMBLK)
DUMPFI = .FALSE.
RETURN

C TEST TO WRITE OUT EITHER A SINGLE PAGE AND ALL OF ITS CONTINUANTS
C OR ALL PAGES
NPAGE = PAGES
HIPAGE = HACTPG(1)
IF(HREQPG .GT. HIPAGE) HIPAGE = HREQPG
IP(PAGES, NP, 0) GO TO 60
PAGES = PAGES + 1
REQPAG(2) = -1
REQPAG(2) = REQPAG(2) + 1
ATTEMPT TO LOCATE OR BRING IN PAGE, CONT
CALL LVEXCH
IS REQ(P, C) IN CORE?
IF(REQ(P, C) .GT. 0) GO TO 67
ANY MORE CONTINUANTS TO THIS PAGE?
IF(HREQPG .EQ. 0) GO TO 70
WRITE OUT PAGE
NBIAS = CTRIPT
NUMBLK = 1
CALL LVWRIT(NBIAS, NUMBLK)
GO TO 65
WRITE SINGLE PAGE ONLY?
IF(PAGE .NE. 0) GO TO 80
IS THIS THE HIGHEST ACTIVE PAGE?
IF(PAGES .LT. HIPAGE) GO TO 55
DUMPFI = .FALSE.
RETURN

C ***BINARY WRITE

CTRIP = DIRSZ
CTRL = CTRIPT + HDRSZ
DO 150 I = 1, INCORE
COPY TO DISK IF CONTINUANT HAS BEEN MODIFIED IN CORE
IF(FLSPC(CTRLPT + HDRF) .AND. WRITE) .EQ. 0) GO TO 145
FLSPC(CTRLPT + HDRF) = 0
LENGTH = PAGHDR
BUFLOC = CTRLPT + 1
WSA = NORSPC(CTRLPT + THWSA)
CALL LPAGNW
CTRLPT = CTRLPT + PAGHDR
CTRL = CTRLPT + HDRSZ
CONTINUE
CALL LVCIOS
RETURN
END
SUBROUTINE LVWRIT(NBIAS,NUMBK)

IMPLICIT INTEGER(A-Z)

COMMON /1VCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
  
M, PAGLOC,CURRENT

COMMON /1VBUR/ PAGSIZE,NMCHAN,OLCHAN,CMPPAND,PAGHDR,BURPSIZE,DIRSZE,
  
INCORE,BURSZE,MSADIR,SUPFSIZE,BLKSZE,DIRBKI,PAHD4

COMMON /1FLAG/ FLMSK,FI1MSK,FI2MSK,FI3MSK,FI4MSK,FI5MSK,FI6MSK,
  
FLG8,FLG9,FLG10,FLG11,FLG12,FLG13,FLG14,
  
2, FLG15

COMMON /1VHDV/ THWSA,REGAS,PAGENO,CONTNO,INSPEL,
  
U, USECT,HDRFLG,READVL,OLDDW,DDNW,HROWN,DROWN

COMMON /1VPRAM/ BUTFCNT,BUFFLOC.LENGTH,IERR,ERRNB,BCP,MODE,PAGES,
  
LUN

COMMON /1VCASK/ MIWRITE,NOTUSP,NEWCON,FLGMSK,MASKSF,MASKPF

COMMON /1VTR1/ NOTDSPC(1)
  
1 /1VTR2/ LSTSPC(1)
  
2 /1VTR3/ LNKSPC(1)
  
3 /1VTR4/ FIGSPC(1)

C THIS ROUTINE PERFORMS A BCD WRITE OF NUMBK PAGE-BLOCKS BEGINNING AT
C GIRS BUFFER LOCATION NBIAS

C FLAG CONTINUANT AS USED
D PAUSE 'IN LVWRIT'
D 0010 FIGSPC(CTRLPT+HDRFLG) = FIGSPC(CTRLPT+HDRFLG) .AND. .NOT. NOTUSP
D 0011 DO 100 M = 1, NUMBK
D 0012 ISTLOC = NBIAS+HDRSIZE+1
D 0013 LAST = ISTLOC-PAGSIZE-1
D 0014 IF(NBIAS .GT. 0) GO TO 10
D 0015 LAST = DIRSZ
D 0016 GO TO 30
D 0017 C EXTRACT HEADER VALUES
D 0018 MSA = NOTDSPC(NBIAS * THWSA)
D 0019 REGAS = NOTDSPC(NBIAS * REGAS)
D 0020 PAGE = LSTSPC(NBIAS * PAGENO)
D 0021 CONT = LSTSPC(NBIAS * CONTNO)
D 0022 USEAGE = LNKSPC(NBIAS * USECT)
D 0023 USEAGE = LNKSPC(NBIAS * USECT)
D 0024 FLAGS = FIGSPC(NBIAS * HDRFLG)
D 0025 RPVAL = FIGSPC(NBIAS * READVL)
D 0026 WRITE(LUN,1)
D 0027 WRITE(LUN,2) PAGE,CONT
C WRITE HEADER
D 0028 WRITE(LUN,3)
D 0029 WRITE(LUN,4) MSA, REGAS, INSPLT, USEAGE, FLAGS, RPVAL
D 0030 WRITE(LUN,5)
D 0031 COUNT = 0
D 0032 DO 30 BUFCTR = ISTLOC, LAST
D 0033 COUNT = COUNT - 1
D 0034 M1 = NOTDSPC(BUFCTR)
D 0035 M2 = LSTSPC(BUFCTR)
D 0036 M3 = LNKSPC(BUFCTR)
D 0037 M4 = FIGSPC(BUFCTR)
D 0038 IF(NBIAS .GT. 0) GO TO 45
C !X-CORE DIRECTORY

0040 CONTIN = -1
0041 IF (.GT. FLGSPC(BUCFNT), AMP, FLMSK) .EQ. 0) GO TO 40
0042 CONTIN = BUCFNT - M1 - 1
0043 IF (.GT. CONTIN, IT, 0) CONTIN = CONTIN + DIRSZE
0044 GO TO 47
0045 M1 = 0
0046 M2 = 0
0047 WRITE(LUN, 7) BUFCNT, COUNT, M1, CONTIN, M2, M3, M4, COUNT
0048 GO TO 50
0049 WRITE(LUN, 6) BUFCNT, COUNT, M1, M2, M3, M4, COUNT
0050 CONTINUE
0051 CONTINUE
0052 CONTINUE
0053 CONTINUE
0054 CONTINUE
0055 FORMAT(///, 10X, 'PAGE', 3X, 'CONTINUANT', //)
0056 FORMAT(8X, 16, 3X, 16, /)
0057 FORMAT(1X, 'USA REGASP (INSERTIONS-DELETIONS) USAGE')
0058 FORMAT(1X, 'USA REGASP (INSERTIONS-DELETIONS) USAGE')
0059 FORMAT(1X, 'USA REGASP (INSERTIONS-DELETIONS) USAGE')
0060 FORMAT(1X, 'USA REGASP (INSERTIONS-DELETIONS) USAGE')
0061 FORMAT(1X, 'USA REGASP (INSERTIONS-DELETIONS) USAGE')
0062 RETURN
0063 END
SUBROUTINE LVCLOS

IMPLICIT INTEGER(A-Z)

COMMON /VREGS/ CURPAG(4), REQPA.3(4), ISTVPG(4), MSARET,
                  HREQPG, NXTMSA, HACTPG(2), READCT, USECNT, DIRPAG,
                  DIRCNT, OUTLOC, OUTDIR(256), RWBUF(1)

COMMON /VMASK/ MWRITE, NOTTUSD, NEWCOMP, FILEONX, MASKSP, MASKPF

COMMON /VCRNT/ PRIME, USECNT, CTXTPT, CTXTRT, LASTV, *NTFREE, *FREE, DREGSP,
                  MSA, PAGLOC, CURRENT

COMMON /VHREQ/ THRMSA, REGAS, PAGENO, CONTNO, INSDEL,
                  USECT, HDRPFG, READVI, OLDNDB, DNOBH, DROWB

COMMON /VHDVI/ THSMA, REGAS, PAGENO, CONTNO, INSDEL,
                  USECT, HDRPFG, READVI, OLDNDB, DNOBH, DROWB

COMMON /IVSEQ/ ISEQSZ, ISOPOS, LASTSV, SEQSPC(1)

COMMON /IVPRAM/ BWPOC, LENGTH, IERR, ERRNTM, BINARY, BCP, MODE, PAGES,
                  LUN

SAVE SYSTEM VARIABLES ON FIRST BLOCK OF DISK

DO 10 I = 1, 4
  RWBUF(1) = REGASP
  RWBUF(2) = NXTMSA
  RWBUF(3) = PAGSZF
  RWBUF(4) = PAGHDR
  RWBUF(5) = BUFZIE
  RWBUF(6) = DIRZIE
  RWBUF(7) = DRENSP
  RWBUF(8) = INCORE
  RWBUF(9) = HDNSZIE
  RWBUF(10) = HREQPG
  RWBUF(11) = HACTPG(1)
  RWBUF(12) = HACTPG(2)
  RWBUF(13) = READCT
  RWBUF(14) = BLSZIE
  RWBUF(15) = SUPSZIE
  RWBUF(16) = DIRBLX
  RWBUF(17) = PRIME
  RWBUF(18) = SEED
  RWBUF(19) = LISTSZ
  RWBUF(20) = LSEQSZ

DO 10 I = 1, 4
  RWBUF(20+I) = CURPAG(I)
  RWBUF(24+I) = LSTVPG(I)

CONTINUE

USER WILL HAVE ACCESS TO WORDS 29 THRU 256 OF THE FIRST BLOCK TO STORE
VARIABLES IF A PERMANENT FILE IS TO BE CREATED.

DO 15 I = 29, 256
  RWBUF(I) = USFR(J)

LENGTH = 256

MSA = 0
ERRNUM = 3
IERR = IWRITW(LENGTH, RBUF(1), MSA, NWCHAN)
DUMP = 0
IF (IERR .NE. 0) CALL LVERR(DUMP)

C SAVE GRN VARIABLES
LENGTH = 256
MSA = 1
ERRNUM = 4
IERR = IWRITW(LENGTH, GRNTBL(1), MSA, NWCHAN)
DUMP = 0
IF (IERR .NE. 0) CALL LVERR(DUMP)

C SAVE INCORE DIRECTORY BEGINNING AT MSA = 2
MSA = 2
BUFLOC = 1
LENGTH = DRSIZE
ERRNUM = 5
CALL LVPAGW

C SAVE CURRENT OUTCORE DIRECTORY
CALL LVPRWR

C CLOSE CHANNEL
CALL CLOSE(NWCHAN)
RETURN
END
SUBROUTINE LVPAGR(CHAN)

IMPLICIT INTEGER(A-Z)
LOGICAL*I, CITRENT

COMMON /1VREGS/ CURPAG(4), REQPAG(4), LSTVPG(4), NSARET,
          HREQP, NXTMSA, HACTFG(2), READCT, USERCT, DIRPAG,
          DIRECT, OUTLOC, OUTDIR(256), RWBUF(1)

COMMON /1VCRNT/ REGASP, CTRIP, CTR1, LEASTVNTFR, FREE,
          NSARET, I, HREQP(3, NXTMSA, NACTPG, READCT, USI, TDIPAG,
          DIRCNT, OUTTLOC, OITTDIR(256), RWITF

COMMON /1VBR1/ PAG1SZE, NCLUSION, OILCJAN, CBENIAN, PAG14D Ri,
          INCORE, HDRSZE, MSADR, SUPSZE, BI KSZE, DIRBK, PAGH4

COMMON /1VPR1/ BUFSZ, LENGTH, IERR, ERRNUM, BINARY, BCD, ROWNUM, MESSAGES

COMMON /1VMASK/ WRITE, NOTUSD, NEWCON, FIGSKE, MASKSF, MASKPF

COMMON /1VHDR1/ THOSMSA, REGAS, PAGENO, CONTNO, INSPDL,

LISTSPC(1)

COMMON /1VVT1/ NORSPEC(1)
1 /1VVT2/ LISTSPC(1)
2 /1VVT3/ LNKSPC(1)
3 /1VVT4/ FIGSPC(1)

THIS ROUTINE READS DATA FROM DISK INTO RWBUF AND PLACES IT INTO WRKSPC

PAUSE "IN LVPAGR()"

NEWLEN = 4*LENGTH
IERR = IREADW(NEWLEN, RWBUF(1), NSA, CHAN)
ERRNUM = 8
DUMP = 0
IF(IERR.LT.0) CALL LVERR(DUMP)
DO 10 I = 1, LENGTH
10 NOPLOC = BUFLOC + 1
IF(ISTLOC .EQ. 0) RETURN
NORSPEC(ISTLOC + 1) = RWBUF(1)
LISTSPC(ISTLOC + 1) = RWBUF(LENGTH + 1)
LINKSPC(ISTLOC + 1) = RWBUF(2*LENGTH + 1)
IF(ISTLOC .LE. 0) RETURN

IF NOT DIRECTORY, FLAG CONTINUANT AS NOT USED
FIGSPC(ISTLOC+HDRFIG) = FIGSPC(ISTLOC+HDRFIG) .OR. NOTUSD
RETURN
END
SUBROUTINE LVPGWN
DILIMPLICIT INTEGER(A-Z)
LOGICAL*1, CURRENT
COMMON /VPPSS/ CURPAG(4), RQPAG(4), LSTVPAG(4), CURNT.
1 HREPO, NTHSHA, HACTPT(2), RFADCT, USECNT, DIRPAG.
2 DIRENT, OUTLOC, OUTDIR(256), RWBUF(1)
COMMON /VCNTN/ RXGAS, CPRST, CURPAG, L.EASTV, NTPRFE, FREE, DRFGSP.
1 RSA, PAGLOC, CURRENT
COMMON /VBUFR/ PAGESZ, NCHAN, OLCHAN, COMPN, PAGEHR, BUTSZF, DBRSZF,
1 INCORE, HDRSFZ, NSRDIR, SUPSFZ, BI KSFZ, DIRBI K, PAGHR.
COMMON /VPPAM/ BUFI LOC, LENGTH, IERR, ERRNM, BINARY, BCP, NODE, PAGES.
1 LEN
COMMON /VHDWI/ THRDPSA, RPQG, PAGNO, CONTNO, INSDEL,
1 USSCT, HDRSF, READV, OI DMP, DNPEN, NROWB, DROWB
COMMON /VVTIN/ WRITE, NOTUSP, NEWCOX, FIGMSK, MASKSF, MASKFP
COMMON /VVTR/ NOSPC(1)
1 /VVTR2/ LSTSPC(1)
2 /VVTR3/ LNKSPC(1)
3 /VVTR4/ FIGSPC(1)

THIS ROUTINE TRANSFERS THE CONTENTS OF WRA SP TO RWBUF TO BE WRITTEN
OUT TO DISK
PAUSE 'IN LVPGWN'

IF ISTLOC .NE. LENGTH
ISTLOC = BUFLOC - 1

IF NOT DIRECTORY, TURN FLAGS OFF
IF(ISTLOC .EQ. 0) GO TO 5
DO 10 1 = 1, LENGTH
RWBUF(1) = NOSPC(ISTLOC + 1)
10 RWBUF(LENGTH + 1) = LSTSPC(ISTLOC + 1)
18 RWBUF(LENGTH + 1) = FIGSPC(ISTLOC + 1)
20 RWBUF(3*LENGTH + 1) = FIGSPC(ISTLOC + 1)

IERR = IWRIT(NEWLEN, RWBUF(1), RSA, NCHAN)
ERRNM = 9
DUMP = 0
IF(IERR .NE. 0) CALL LVERR(DUMP)
RETURN
END
SUBROUTINE LVRPR (MAN)

IMPLICIT INTEGER (A-Z)

COMMON /LVRFUG/ CBUFAG(4), RBUFAG(4), LSTVPAG(4), RSTVPAG(4), RSTVPAG(4), MSAR1T

1. HBUFAG, NTMSA, NTCTPG(2), READT, UNFRONT, DIPAG.
2. DIREP(DOTLOC OUTDIR(256), DIREP(256)).

COMMON /LVBFHR/ PAGSZ, MSCHAN, CMPAR, DIREP, BPNFZ, DIREP(DIREP), RUFTAG(DIREP).

COMMON /LVPRAF/ BUFLOC, KGMO, LENGTH, IERR, ERR, ERR, ERR, BINARY, OUTDIR, PAGE(20).

LIN

THIS ROUTINE READS A SELECTED OUTDIR DIRECTORY BLOCK INTO OUTDIR.

PAUSE 'IN LVPRMR'.

LENGTH = 256.

MFA = MSADIR + DIREP + DIPAG.

ERRNUM = 6.

IERR = IREAD + LENGTH, OUTDIR(1), MSCHAN.

PUMP = 4.

IF IERRIT = 0 CALL LVFRR(DUMP)

RETURN

END

SUBROUTINE LVWPR

IMPLICIT INTEGER (A-Z)

COMMON /LVRFUG/ CBUFAG(4), RBUFAG(4), LSTVPAG(4), RSTVPAG(4), RSTVPAG(4), MSAR1T

1. HBUFAG, NTMSA, NTCTPG(2), READT, UNFRONT, DIPAG.
2. DIREP(DOTLOC OUTDIR(256), DIREP(256)).

COMMON /LVBFHR/ PAGSZ, MSCHAN, CMPAR, DIREP, BPNFZ, DIREP(DIREP), RUFTAG(DIREP).

COMMON /LVPRAF/ BUFLOC, KGMO, LENGTH, IERR, ERR, ERR, ERR, BINARY, OUTDIR, PAGE(20).

LIN

THIS ROUTINE WRITES THE OUTDIR DIRECTORY BLOCK FROM OUTDIR TO DISK.

PAUSE 'IN LVWPR'.

LENGTH = 256.

MFA = MSADIR + DIREP + DIPAG - 1.

ERRNUM = 7.

IERR = IWRITE + LENGTH, OUTDIR(1), MSCHAN.

PUMP = 4.

IF IERRIT = 0 CALL LVFRR(DUMP)

RETURN

END
THE PURPOSE OF THIS ROUTINE IS TO 'UNDEFINE' A NODE.

D PAUSE 'IN LVRTRN'

0010  FISPC('CTRI 1 +IVAL) = FISPC('CTRI 1 +ival) .AND. .NOT.FI3MSK

0011  RETURN

0012  END
C C C

0001 FUNCTION LVRTSH(WORD,BITS)
0002 IMPLICIT INTEGER(A-Z)

C THIS FUNCTION PERFORMS A RIGHT LOGICAL SHIFT

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 LVRSH = 0
0012 RETURN
0013 END

C C C

0001 FUNCTION LVIFSL(WORD,BITS)
0002 IMPLICIT INTEGER(A-Z)

C THIS FUNCTION PERFORMS A LEFT LOGICAL SHIFT

0003 IF (BITS .EQ. 0) GO TO 10
0005 IF (BITS .GE. 16) GO TO 20
0007 LVIFSL = WORD * 2 ** (BITS)
0008 RETURN
0009 10 LVIFSL = WORD
0010 RETURN
0011 20 LVIFSL = 0
0012 RETURN
0013 END
COMMON BLOCK /IVARGS/ LENGTH 000104
IFUNC 000000 INTEGER*2 VARIABLE
LARG 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
INCLUD 000020 INTEGER*2 VARIABLE
INDXON 000022 INTEGER*2 VARIABLE
IVALS 000024 INTEGER*2 ARRAY (10)
ITYP1 000025 INTEGER*2 ARRAY (10)
SRCSUP 000026 INTEGER*2 VARIABLE
LNKSUP 000028 INTEGER*2 VARIABLE
SNKSUP 000030 INTEGER*2 VARIABLE
INSTYP 000032 INTEGER*2 VARIABLE
COMMON BLOCK /IVREGS/ LENGTH 000108
C1RPAG 000000 INTEGER*2 ARRAY (4)
REQPAG 000010 INTEGER*2 ARRAY (4)
LSTVPG 000020 INTEGER*2 VARIABLE
NAME OFFSET ATTRIBUTES
MSARET 000030 INTEGER*2 VARIABLE
HREQPG 000032 INTEGER*2 VARIABLE
NXMSA 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)
RWBUF 001054 INTEGER*2 ARRAY (1)
COMMON BLOCK /IVMASK/ LENGTH 000014
MWRITE 000000 INTEGER*2 VARIABLE
NOTUSID 000002 INTEGER*2 VARIABLE
NEWCON 000004 INTEGER*2 VARIABLE
FIGMSK 000006 INTEGER*2 VARIABLE
MASISF 000010 INTEGER*2 VARIABLE
MASKPF 000012 INTEGER*2 VARIABLE
COMMON BLOCK /IVFLAG/ LENGTH 000036
FI0MSK 000000 INTEGER*2 VARIABLE
FI1MSK 000002 INTEGER*2 VARIABLE
FI2MSK 000004 INTEGER*2 VARIABLE
FI3MSK 000006 INTEGER*2 VARIABLE
FI4MSK 000010 INTEGER*2 VARIABLE
FI5MSK 000012 INTEGER*2 VARIABLE
FI657 000014 INTEGER*2 VARIABLE
FLG8 000016 INTEGER*2 VARIABLE
FLG9 000020 INTEGER*2 VARIABLE
FLG10 000022 INTEGER*2 VARIABLE
FLG11 000024 INTEGER*2 VARIABLE
FLG12 000026 INTEGER*2 VARIABLE
FLG13 000030 INTEGER*2 VARIABLE
FLG14 000032 INTEGER*2 VARIABLE
FLG15 000034 INTEGER*2 VARIABLE
COMMON BLOCK /IVRAND/ LENGTH 001016
PRIME 000000 INTEGER*2 VARIABLE
SIZE 000002 INTEGER*2 VARIABLE
NROW 000004 INTEGER*2 VARIABLE
DNOPE 000006 INTEGER*2 VARIABLE
DROW 000010 INTEGER*2 VARIABLE
ODDNOD 000012 INTEGER*2 VARIABLE
LISTSZ 000014 INTEGER*2 VARIABLE
GRNTBI 000016 INTEGER*2 ARRAY (256)
COMMON BLOCK /VCRNT/ LENGTH 000023
REGAS 000000 INTEGER*2 VARIABLE
CTRLPT 000002 INTEGER*2 VARIABLE
CTRL1 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
CURREN 000022 LOGICAL*1 VARIABLE
COMMON BLOCK /VBUFR/ LENGTH 000034
PAGSZE 000000 INTEGER*2 VARIABLE
NCHAN 000002 INTEGER*2 VARIABLE
OLCHAN 000004 INTEGER*2 VARIABLE
CMNDAND 000006 INTEGER*2 VARIABLE
PAGHDR 000010 INTEGER*2 VARIABLE
BUFSZE 000012 INTEGER*2 VARIABLE
DIRSIZE 000014 INTEGER*2 VARIABLE
INCORE 000016 INTEGER*2 VARIABLE
HDRSZE 000020 INTEGER*2 VARIABLE
MSADIR 000022 INTEGER*2 VARIABLE
SUPSZE 000024 INTEGER*2 VARIABLE
BLKSIZE 000026 INTEGER*2 VARIABLE
DIRBLK 000030 INTEGER*2 VARIABLE
PAGHDR 000032 INTEGER*2 VARIABLE
COMMON BLOCK /VBVDI/ LENGTH 000030
THBSIZE 000000 INTEGER*2 VARIABLE
REGAS 000002 INTEGER*2 VARIABLE
PAGENO 000004 INTEGER*2 VARIABLE
CONTNO 000006 INTEGER*2 VARIABLE
INSPEL 000010 INTEGER*2 VARIABLE
USECT 000012 INTEGER*2 VARIABLE
HDRFIL 000014 INTEGER*2 VARIABLE
READVI 000016 INTEGER*2 VARIABLE
OLNDNB 000020 INTEGER*2 VARIABLE
DNOUEB 000022 INTEGER*2 VARIABLE
NROWB 000024 INTEGER*2 VARIABLE
DROWB 000026 INTEGER*2 VARIABLE
COMMON BLOCK /LVSWIT/ LENGTH 000020
SETUP 000000 LOGICAL*1 VARIABLE
SNGLBK 000001 LOGICAL*1 VARIABLE
NXTTRAN 000002 LOGICAL*1 VARIABLE
IN1STR 000003 LOGICAL*1 VARIABLE
IN2STR 000004 LOGICAL*1 VARIABLE
PR1STR 000005 LOGICAL*1 VARIABLE
PR2STR 000006 LOGICAL*1 VARIABLE
DL1STR 000007 LOGICAL*1 VARIABLE
DL2STR 000010 LOGICAL*1 VARIABLE
INFTMP 000011 LOGICAL*1 VARIABLE
FTZTMP 000012 LOGICAL*1 VARIABLE
DLZTMP 000013 LOGICAL*1 VARIABLE
DMPFPL 000014 LOGICAL*1 VARIABLE
FINPFL 000015 LOGICAL*1 VARIABLE
DLETPL 000016 LOGICAL*1 VARIABLE
NSKIPFL 000017 LOGICAL*1 VARIABLE
COMMON BLOCK /LVSE0/ LENGTH 000010
ISEOSZ 000000 INTEGER*2 VARIABLE
ISOFPOS 000002 INTEGER*2 VARIABLE
LASTSO 000004 INTEGER*2 VARIABLE
SPOSFC 000006 INTEGER*2 ARRAY (11)
COMMON BLOCK /VPRAM/ LENGTH 000022
BUFFLOC 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
NODE 000014 INTEGER*2 VARIABLE
PAGES 000016 INTEGER*2 VARIABLE
LUN 000020 INTEGER*2 VARIABLE

COMMON BLOCK /VSTAK/ LENGTH 000006
CURLEV 000000 INTEGER*2 VARIABLE
NUMVAR 000002 INTEGER*2 VARIABLE
STACK 000004 INTEGER*2 VARIABLE

COMMON BLOCK /VUTIL/ LENGTH 000126
FILSPC 000000 INTEGER*2 ARRAY (39)
DEFEXT 000116 REAL*4 ARRAY (2)

COMMON BLOCK /VINSJ/ LENGTH 000003
REORG 000000 LOGICAL*1 VARIABLE
FILL 000001 LOGICAL*1 VARIABLE
RPLACE 000002 LOGICAL*1 VARIABLE

COMMON BLOCK /VVRN/ LENGTH 000006
RUNTYP 000000 INTEGER*2 VARIABLE
CORE 000002 REAL*4 VARIABLE

COMMON BLOCK /VVTR1/ LENGTH 000002
NOPSPC 000000 INTEGER*2 ARRAY (1)

COMMON BLOCK /VVTR2/ LENGTH 000002
LSTSPC 000000 INTEGER*2 ARRAY (1)

COMMON BLOCK /VVTR3/ LENGTH 000002
LNKSPC 000000 INTEGER*2 ARRAY (1)

COMMON BLOCK /VVTR4/ LENGTH 000002
FIGSPC 000000 INTEGER*2 ARRAY (1)

COMMON BLOCK /VVEL1/ LENGTH 000003
NUMET 000000 INTEGER*2 VARIABLE
BAKCON 000002 LOGICAL*1 VARIABLE

COMMON BLOCK /VADDR/ 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 /VFNPV/ LENGTH 000005
COUNT 000000 INTEGER*2 VARIABLE
ABSPOS 000002 INTEGER*2 VARIABLE
LSTCON 000004 LOGICAL*1 VARIABLE
REFERENCES


### INITIAL DISTRIBUTION

<table>
<thead>
<tr>
<th>Copies</th>
<th>Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CHONR</td>
<td>1826 L. Culpepper</td>
</tr>
<tr>
<td>1</td>
<td>NRL</td>
<td>184 J. Schot</td>
</tr>
<tr>
<td>1</td>
<td>NSWC</td>
<td>184.1 H. Feingold</td>
</tr>
<tr>
<td>1</td>
<td>NUSC</td>
<td>1843 H. Haussling</td>
</tr>
<tr>
<td>1</td>
<td>NOSC</td>
<td>1844 S. Dhir</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1844 J. McKee</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>185 T. Corin</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1850 A. Cinque</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1851 J. Brainin</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1854 H. Sheridan</td>
</tr>
<tr>
<td>2</td>
<td>NAVSEA</td>
<td>1855 R. Brengs</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>187 M. Zubkoff</td>
</tr>
<tr>
<td>1</td>
<td>Rome Air Development Center</td>
<td>189 G. Gray</td>
</tr>
<tr>
<td>12</td>
<td>DTIC</td>
<td>5211.1 Reports Distribution</td>
</tr>
<tr>
<td></td>
<td></td>
<td>522.1 Unclass Lib (C)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>522.2 Unclass Lib (A)</td>
</tr>
</tbody>
</table>

### CENTER DISTRIBUTION

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