IBM VSE/Enterprise Systems Architecture
VSE Central Functions

Librarian Diagnosis Reference

Version 6 Release 5

SC33-6330-01

IBM VSE/Enterprise Systems Architecture
VSE Central Functions

Librarian Diagnosis Reference

Version 6 Release 5

SC33-6330-01

Notice!

Before using this information and the product ist supports, be sure to read the general information under “Notices” on page xi.

Forth Edition (Oct 1999)

This edition applies to Version 6 Release 5 of IBM VSE/Advanced Functions, which is part of VSE/Central Functions, Program Number 5686-066, and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters.

This manual is not available in print. It has been updated for softcopy Oct 1999.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the addresses given below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Deutschland Entwicklung GmbH
Department 3248
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany

You may also send your comments by FAX or via the Internet:

Internet: s390id@de.ibm.com
FAX (Germany): 07031-16-3456
FAX (other countries): (+49)+7031-16-3456

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Notices

References in this publication to IBM* products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

This publication is intended primarily for use by IBM personnel responsible for program service. The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. It is not intended as a description of a programming interface. The use of this information is a customer reponsibility. Service for errors, ommissions, accuracy, or completeness will not be provided.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Corporation, IBM Director of Licensing, North Castle Drive, Armonk, New York 10504-1785, U.S.A.

Programming Interface Information

This book documents information that is NOT intended to be used as a Programming Interface of VSE/ESA.

Trademarks and Service Marks

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in certain countries:

ACF/VTAM
ESA/370
ESA/390
CICS/VSE
IBM
System/370
VM/ESA
VSE
VSE/ESA
VTAM

Preface

This manual is intended for use by IBM personnel responsible for servicing the VSE/ESA VSE/Central Functions Librarian or for implementing new functions. The manual supplements the Librarian listings; it is organized into the following chapters:

The first chapter is an **INTRODUCTION** outlining the purpose and method of operation of the Librarian, and discussing the library organization.

The next chapter, **DESIGN INFORMATION**, explains the Librarian functions and the function flow. Detail information (such as function description, entry point, input/output, and use of registers) is given for each module and macro.

Then follows **PROGRAM ORGANIZATION**, providing cross reference information. It helps you, for example, to find out which modules are called by a given module, or which macros are invoked by a certain module.

The next chapter, **DATA AREAS**, contains the functional description and layout of all Librarian data areas and the library record types.

The last chapter, **DIAGNOSTIC AIDS**, contains some hints how to determine problems in the area of Librarian services and Librarian data structures.

Bibliography:

Information about using Librarian functions is contained in other VSE/ESA manuals as follows:

Guide to System Functions, SC33-6711
It describes the library concept and how to use the librarian.

Planning and *Installation,* SC33-6703, SC33-6704
They provide Librarian resource requirements information, which is needed for planning and running the system; they list the complete VSE/Advanced Functions system library.

System Control Statements, SC33-6713
It describes the Librarian commands and the JCL statements which are necessary to define and access libraries.

Listed below are the diagnosis reference publications that document diagnosis information for related components of VSE/Central Functions:

VSE/Central Functions Diagnosis Reference:

Supervisor, SC33-6323

Error Recovery and Recording Transients, SC33-6326

Initial Program Load and Job Control, SC33-6325

Serviceability Aids, SC33-6327

System Utilities, SC33-6617

Linkage Editor, SC33-6328

Introduction

The Librarian is a group of programs which serve to organize and maintain the libraries of a VSE resident system. It also contains service functions to display and punch parts of them or display their directories and to set up and change the library concatenation chains and their control tables.

The VSE Librarian is a new librarian for the VSE/Advanced Functions Version 5 Release 2 which has replaces the DOS Librarian.

The concept of the VSE Librarian as compared to the DOS Librarian is based on the following aspects:

- common logical library structure
- VSE library data format
- unified Librarian command language
- interactive usage in ICCF partition

Common logical library structure: A library is logically divided into sublibraries. Each sublibrary may contain members of any type (procedures, source books, object modules, phases, user-defined types). This makes a library common for all types of members. Therefore, a component consisting of procedures, source books, object modules, and phases may be put into *one* library and even into one sublibrary.

VSE library data format: The VSE library data format allows space to be reused, that is space freed due to deletion of a member is reused without requiring a condense run.
The block size (1K) selected for the libraries results in improved utilization of DASD space (support of large track DASDs).
The member directories are sorted.

Unified command language: The VSE Librarian commands follow a general syntax. They are more powerful and easier to use than the DOS Librarian statements. They reflect the functional enhancements of the VSE Librarian.
The Librarian functions can be invoked from SYSLOG or SYSRDR/SYSIPT.

Interactive usage in ICCF partition: The VSE Librarian runs in interactive partitions of VSE/ICCF; it can attach VSE libraries/sublibraries dynamically through the VSE Librarian commands. This allows the customer to move members from any VSE library/sublibrary into the ICCF library and vice versa. All Librarian services can be invoked from ICCF terminals for all libraries. The extents of the libraries have to be defined by DLBL/EXTENT JCL statements before starting ICCF.

Library Structure

Physical Organization

A Library is a data set which resides either in BAM space (VTOC controlled) or in VSAM managed space. When in VSAM managed space, it may be extended dynamically. (The Program Product *VSE/VSAM Space Management for SAM Files* is prerequisite for having libraries in VSAM managed space.)

There is a distinction between system and non-system libraries. System libraries are used for IPL and start at a fixed disk address. They are contained in one extent. A system library cannot reside in VSAM managed space. A non-system library data set consists of one or more (up to 16) extents on one or more CKD or FBA DASDs supported by the VSE Librarian. If the extents of a library are spread over several volumes all of the devices must be of the same type.

The format of a system library and non-system libraries is identical. Each library is organized as a sequence of Library Blocks (LBs). Figure 1 on page 2 shows the layout of a library block.

The library blocks of one library are numbered from 0 to n-1 where n is the maximum number of library blocks which fit into the size defined by the DLBL and EXTENT statements for this library. From the addressing point of view the information stored in a library is regarded as a continuous byte string (concatenation of all LBs of a library). Data within a library is addressed by the Physical Relative Byte Address (PRBA) in connection with the extent definitions of the library. The six-byte PRBA is subdivided in an offset part of two bytes, which specifies the offset within a LB, and a four-byte LB number. The first LB of a library has the block number 0. Library data does not contain disk addresses. Therefore, library data sets can be moved to other extents/volumes through the FCOPY program if the source and target library data sets reside in BAM managed space on the same device type.

Each library block has a fixed size (1K) and consists of a data part, free space, and a Library Block Control Field (LBCF - see data area INLCLBCF). The LBCF resides at the end of the library block and its information controls the data part and the free space within the library block as well as the connection (chaining) between the blocks. The data part of a library block either consists of a set of data records of fixed or variable (record type FIXED or VARIABLE) or of a single byte string (record type UNDEFINED). If a library block contains entries of variable record size, the lengths of the entries are stored in reverse order of the entries (in decreasing PRBAs) beginning at the starting PRBA of the Library Block Control Field (LBCF) minus one. The length field of the last entry is preceded by a zero length.

Logical Structure Overview

A library is a general repository for system data (such as phases, modules, procedures, and books) and for any kind of user data.

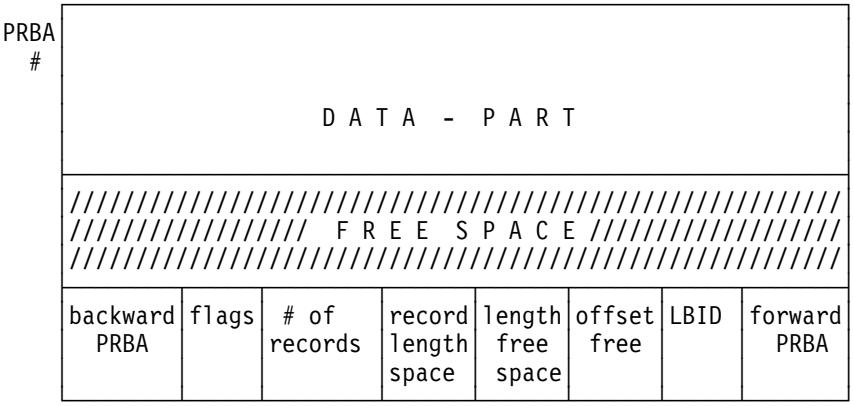


Figure 1. Library Block. A Library Block consists of a data part, free space, and control information (see data area INLCLBCF). Note, that the fields given in this figure do not correspond exactly to the layout of data area INLCLBCF.

It consists of several components:

```
Library
|
|-- Library Header
|-- Free Space Inventory
|-- Member Indexes, each possibly consisting of several levels
|-- Member Data
|-- Space Reclamation Chains
|-- Free Space
```

Correspondingly, a library block which is not part of the free space belongs to one of five classes of objects:

- Library Header
- Free Space Inventory
- Member Index Level
- Member Data
- Space Reclamation Chain

The library blocks of one class are chained together by forward and backward pointers. The forward pointer contains the PRBA of the logically succeeding library block, the backward pointer the PRBA of the logical predecessor block in the chain. The backward pointer of the first block and the forward pointer of the last block have the value "End-of-Chain" (hexadecimal value 'FFFFFFFF' or decimal value '-1').

Each library block (LB) has a Library Block Identification Field (LBCFIDEN - see data area INLCLBCF) of four bytes in its LBCF. This field contains the Library Block Identifier (LBID) which identifies the LB as part of a specific logical library unit (for example, Library Header and Free Space Inventory, member index, specific member). Whenever a library is defined, all LBs of the Library Header and the Free Space Inventory receive a fixed LBID of value 1. This value is never changed during the life-time of the library. Any extension to the Library Header (growing number of entries) or to the Free Space Inventory (additional extent in VSAM managed space) will get the LBID of 1. Each sublibrary of the library receives a unique LBID which is contained in all LBs of its member index. Each member of a sublibrary gets a LBID unique within the library which is attached to all LBs of the member data chain. The LBID of a library object is never changed during its life-time. If LBs of the member index or of a member are inserted into the Space Reclamation Chain of the corresponding sublibrary, their LBIDs are kept unchanged.

Figure 2 shows the chaining for a member with 4 library blocks and the starting block number 27.

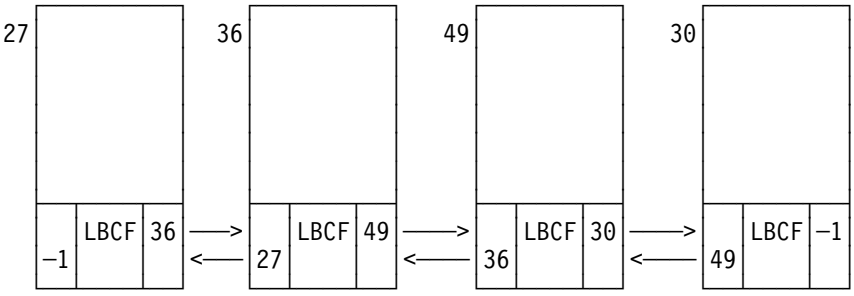


Figure 2. Example for Chain of Library Blocks

The data part of a library block differs depending on the type of data stored in the library block. For detailed information refer to the following sections describing library header, free space inventory, member index, member data, and space reclamation chain.

The logical overall structure of a library is a tree. The root is the library descriptor and the leaves are the members (see Figure 3).

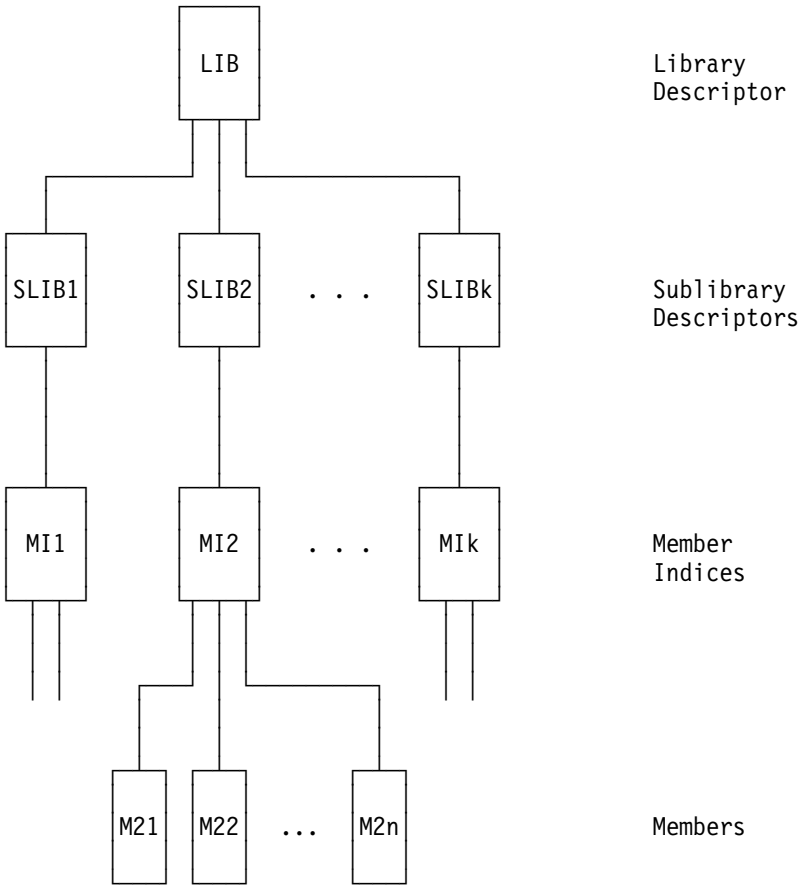


Figure 3. Tree Structure of a Library

The path length through the tree is not fixed because a member index itself is a tree with a variable number of levels to address any number of members.

Library Header

The Library Header starts in the first library block of a library; it has two main parts:

- library descriptor
- sublibrary index consisting of sublibrary descriptors

Library Descriptor: The Library Descriptor (LDES) is stored at the beginning of the library block with PRBA 0 and offset 0. It contains information and attributes related to the entire library. This includes

- the library identifier "LIBRARY" (LABL)
- the name of the library (NAME)
- the PRBA of the sublibrary directory (PRBA)

- the creation date (CRDT)
- the size of a library block (LBSZ)
- the length of the library block control field (LBCF)
- the length of the library descriptor (LNG)
- the number of sublibraries (NOSL)
- the number of library blocks for the library header (NOLB)
- the highest chain identification number within the library (IDEN)
- the number of locked members contained in the library (NLCK)

See data area INLCLDES for the layout of the library descriptor.

Sublibrary Index: A library is logically divided into sublibraries. For each sublibrary there exists a Sublibrary Descriptor (SLXE) in the Library Header. Library descriptor and sublibrary descriptors build one chain. The first sublibrary descriptor immediately follows the library descriptor with PRBA 0 and has an offset which is equal to the length of the library descriptor. If the library has more sublibraries than sublibrary descriptors fit into the remaining space of the first library block, one or more library blocks are appended to it via the PRBA chain (see Figure 4). The sublibrary index is not sorted.

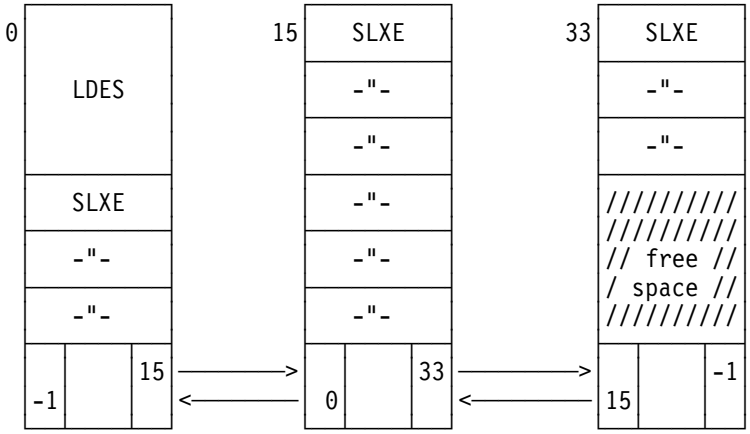


Figure 4. Library Header with Library Descriptor and Sublibrary Index

A sublibrary descriptor (SLXE) includes

- the sublibrary name (NAME)
- the creation date (CRDT)
- the number of library blocks used for the sublibrary (RESV)
- the number of reclaimed library blocks (RECL)
- the PRBA of the chain of reclaimed library blocks (DELA)
- the PRBA of the highest member index level (HXRA)
- the PRBA of the 1st library block in the member directory (LXRA)
- the number of index levels (NXLV)
- the space reclamation attribute of the sublibrary (0: AUTOMATIC, 1: IMMEDIATE) (RCLM)
- the identification number for the sublibrary index (IDEN)
- the number of locked members contained in the sublibrary (NLCK)

See data area INLCSLXE for the layout of the sublibrary descriptor.

Free Space Inventory

The Free Space Inventory maps the allocation state of the library blocks to a bit pattern, thus reflecting the blocks available for allocation. It consists of the Space Descriptor (SPAD) and of as many free space maps as there are extents for the library.

The space descriptor consists of

- the length of the Space Descriptor (LK)
- the number of LBs used for the Free Space Inventory (BMCN)
- the number of allocated library blocks in the library (ALLC)
- the number of free library blocks in the library (AVAL)
- the LB number where the free space begins (BFRE)
- an indication whether space is reused (HIFG)

See data area INLCSPAD for the layout of the space descriptor.
The space descriptor of a library is contained in the free space map of the first library extent.

Each Free Space Map is located at the beginning of the corresponding library extent and occupies one or more contiguously allocated library blocks (the number depends on the extent size).
A free space map consists of an Extent Descriptor (EXTD) and a bit map describing the reservation state of the LBs of the extent (1: used / 0: free).

An extent descriptor includes

- the length of the extent descriptor (LK)
- a free space map sequence number (NO)
- the lowest PRBA of the extent (LODA)
- the highest PRBA of the extent (HIDA)
- the length of the bit map in LBs (EBMLK)
- the length of the bit map in bytes (EBMBYT)
- the length of the bit map in bits (EBMBIT)
- the first allocatable PRBA of the extent (FSP)

See data area INLCEXTD for the layout of an extent descriptor.

The free space map of the first library extent starts at LB-# 1; it contains the space descriptor of the library.

The bit map size is calculated from the size given in the EXTENT statement:
Bit map size (in bits) for CKD Devices =
((# of tracks) * (# library blocks per track)).
Bit map size (in bits) for FBA Devices =
((# of FBA blocks * FBA block size) / size of library block).

All library blocks which belong to the free space inventory are chained together. Figure 5 shows an example of the free space inventory for a library which has two extents. The first extent owns the library blocks 0 up to 1897, the second extent starts at PRBA 1898. For each extent a free space map is allocated, the first one consisting of two LBs, the second of one LB.

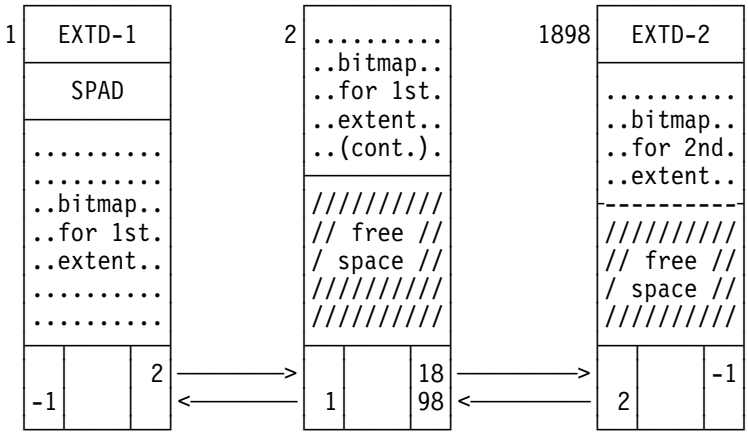


Figure 5. Free Space Inventory Consisting of Two Free Space Maps

Member Index

The index structure consists of a sublibrary index and one or more levels of member indexes.

The sublibrary index has one entry for each sublibrary. The sublibrary index is part of the library header and has been already described. The member index of a sublibrary constitutes the index component; it is described in this section.

The lowest level of the member index, which could also be the only level of the member index, is called the member directory. It contains one entry for each member of the sublibrary.

The individual levels of the member index are numbered from level 1 (lowest level, member directory) up to level n, which is the level immediately below the sublibrary index. The number of index levels for a given sublibrary depends on the number of members stored in that sublibrary. The index entries of the intermediate levels (level i > 1) each describe exactly one LB of level i-1 by highest member key and LB-# of that LB.

The LBs of each level are chained together by forward and backward PRBAs. Since there is no lock done for reading the member index, it can happen that during a search for a member the member index is concurrently updated. In this case, forward and backward PRBAs may point to different chains for a short time. The index update is, however, done in such a way that , for a search operation, the forward chain is guaranteed to be in a consistent state.

Each member index is sorted. The order is arranged by member type, and within each type by member name. For all members of the same type there is only one Type Entry (TYPE), which precedes the respective member names (key compression). If index entries of the same type spread over more than

one LB, the type entry is repeated as first entry of each additional LB. So each LB of a member index starts with a type entry (followed by at least one name entry of that type), and it may contain further (different) type entries if the type changes within that LB.

Member Directory (Member Index Level 1): The members of a sublibrary are indexed by a Member Directory. It has at least one library block - even if the sublibrary is empty. This LB is the first LB of the member index level 1; it is kept during the whole life-time of the sublibrary and its address is used for locking purposes. All library blocks of the member directory are chained.
The member directory is sorted. The order is arranged by member type, and within each type by member name. If there are more directory entries of the same type than fit into one library block, the type entry is repeated as first entry of each additional library block.

The Type Entry (TYPE) consists of

- name of the member type
- identification indicating that this is a 'type' entry

(See data area INLCTYPE.)

The naming convention for the six basic member types is as follows:

Type	Member type name
Phases	PHASE
Object Decks	OBJ
Job Control Procedures	PROC
Dumps	DUMP
Source Books	A - Z , 0 - 9 , # , $, @
User Types	Alphanumeric string of 2 to 8 characters unequal to PHASE, OBJ, PROC, DUMP

A Member Directory Entry (DENT) also called 'member descriptor' consists of two parts - a fixed part common for all types of members and an optional part with varying size.

The fixed part, which has the same layout for all types of members, contains

- member name (NAM)
- identification indicating that this is a 'member' entry (DEF1)
- creation date (DORI)
- date of the last replace (DUPD)
- number of library blocks for the member (MBLK)
- PRBA of the first library block of the member (PRBA)
- PRBA of the last library block of the member (LSTA)
- logical record type (DEF2)
- logical record length (RLEN)
- number of logical records (NORL)

- various flags showing the record type, MSHP control etc.
- flag for the existence of a Variable Information Field (VIF)
- identification number for the member LB chain (IDEN)

(See data area INLCDENT.)

The variable part of the member descriptor contains type-specific information or user-specified directory data and is contained in Variable Information Fields (VIFs), which are appended in arbitrary order to the common part of the descriptor.

Each VIF includes:

- an identifier (EID)
- its length (ELEN)
- a flag for a following VIF (VIF) and
- attribute fields for the various uses of the VIF

(See data area INLCVIFD.)

There exist two types of system-defined VIFs:

- Phase VIF
- Lock VIF

Members of type PHASE and those members whose type was originally PHASE and which have been renamed to user types have a phase vif. Lock vifs belong to locked members locked by the lock command or LIBRM LOCK request.

The VIF of a phase (UPHA) includes information like:

- flags indicating relocatable, SVA eligible, active etc.
- switches setting the system directory list
- length and
- entry point
- AMODE/RMODE information for the phase.

(See data area INLCUPHA.)

The VIF of a locked member (LCKV) contains

- the lockid by which the member was locked.

(See data area INLCLCKV.)

Inserting a new Member into the Directory: Figure 6 on page 11 shows a member directory with a size of 2 library blocks. It contains the entries for the members ABC.OBJ, ..., XYZ.OBJ, ABC.PROC, ..., XYZ.PROC and PQ.ZA ordered according to their types OBJ, PROC and the user type ZA.

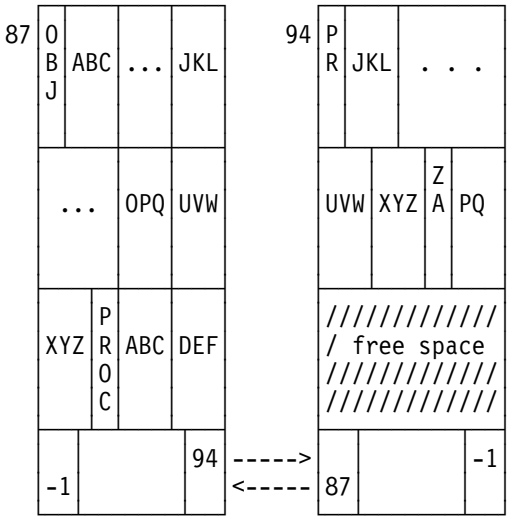


Figure 6. Example of a Member Directory

If a new member is to be inserted into the member directory, different actions take place depending on the name and type of the new member:

- Adding Member TU.ZA
  The directory entry is placed into the free space behind the entry PQ.ZA.
- Adding Member VWX.PROC
  The directory entry is placed behind the entry UVW.PROC and all other entries are shifted to the right as far as necessary into the free space.
- Adding Member LM.OBJ
  If the library block where LM.OBJ belongs to is full, the block is split in the middle. A new block is placed between the two; it takes over the type of the first entry in the second half of the old block. All entries of the old second half are moved into the new block. The new member entry is inserted after the entry JKL.OBJ and the following entries are moved to the right occupying some of the free space (see Figure 7).

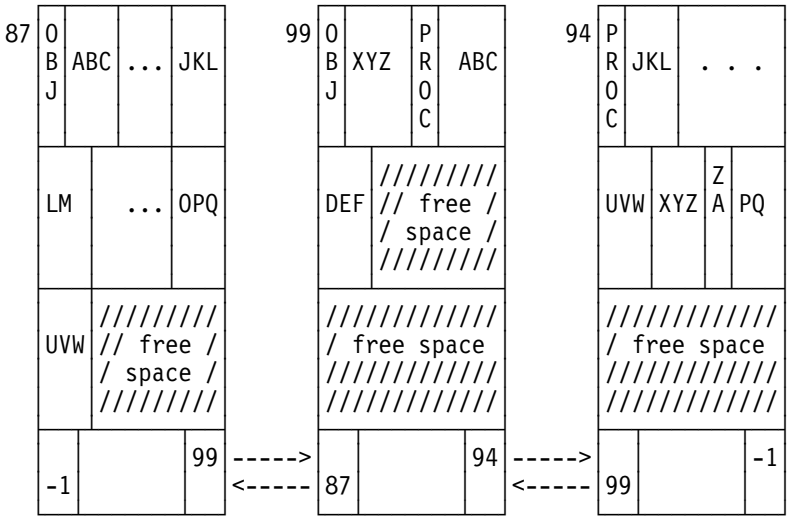


Figure 7. Member Directory with New Member LM.OBJ

- Adding Member XYZ.BOOK
  If a member with a new user type called XYZ.BOOK is added and the library block is full, the block is split and a new block is inserted. The new block receives the entries of the second half of the old block and the respective type entry.
  The entries for the new type and the new member are placed in front of type OBJ in the first block, and the old entries of this block are shifted into the free space that resulted from the split (see Figure 8).

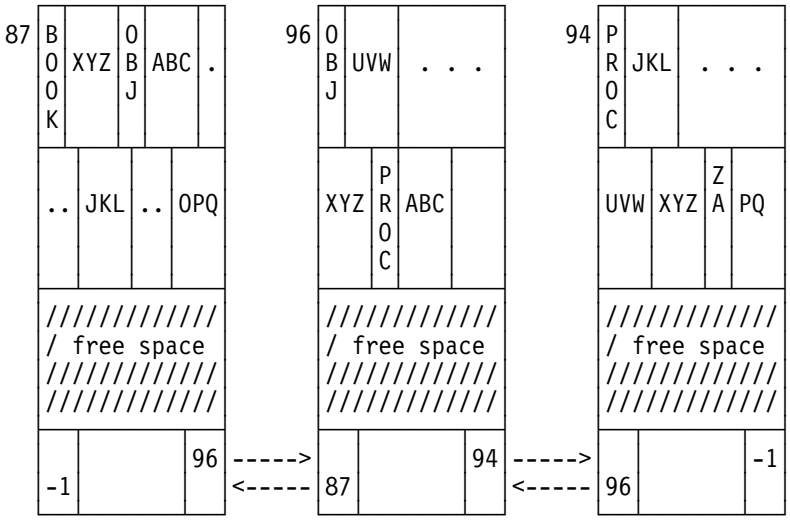


Figure 8. Member Directory with New Member XYZ.BOOK

Member Index with Higher Levels: If the member directory of one sublibrary fits into two library blocks, one member index level is sufficient. In this case the "PRBA of the highest member index level" (HXRA) and the "PRBA of the first LB of the member directory" (LXRA) in the sublibrary descriptor are the same: they point to the first library block of member index level 1.

If more than two library blocks are necessary, a member index level 2 is created. It contains a LB with a type entry followed by a Higher Level Member Index Entry (MBRX) for each library block of the lower level (in this case the member directory). If the member type changes in one library block, a new type field is inserted.

The higher level member index entry (MBRX) includes

- the highest key (name/type) in the lower level library block
- the PRBA of the lower level library block

(See data area INLCMBRX.)

In the case of two index levels "PRBA of the highest member index level" (HXRA) in the sublibrary descriptor points to the first library block of member index level 2, while the "PRBA of the first LB of the member directory" (LXRA) points to the first library block of member index level 1.

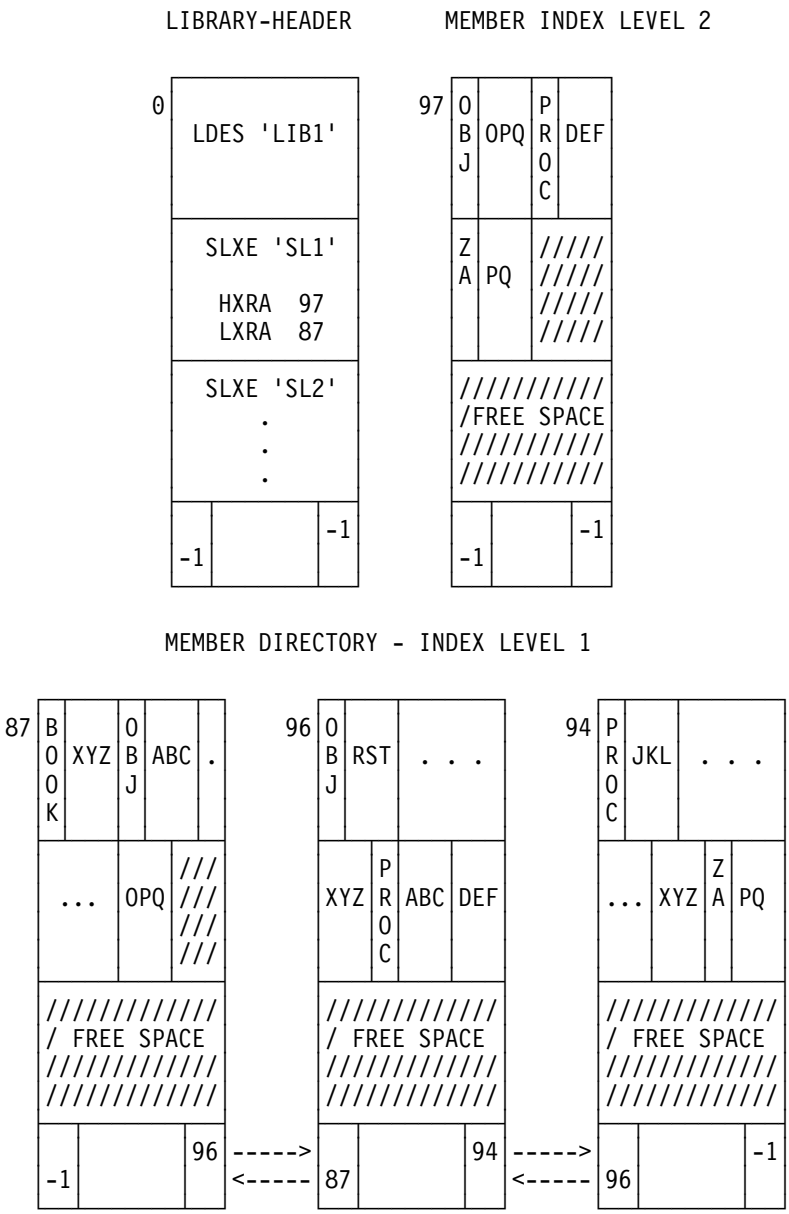


Figure 9. Example of a Member Index with 2 Levels

Figure 9 shows the sublibrary descriptor, the member index level 2 and the member directory. When searching for the member XYZ.OBJ, the algorithm looks into the member index level 2 first, finds the type entry OBJ for the first library block of the member directory but its highest key OPQ is lower than the wanted member name. The next entry, however, is the type entry PROC. Therefore, member XYZ.OBJ, if present, must be in the library block pointed to by this entry.

While the sublibrary is growing and new member directory entries are inserted into the member directory, new library blocks are created on this level, by splitting or appending. Each new library block gets an entry in the index level 2. If these entries do not fit into one library block, a new library block is appended to the first one chained via the PRBAs.

For the insertion of entries in a second level member index, the same rules are applying as for the insertion of entries in the member directory.

If more than two library blocks are necessary for index level 2, an index level 3 is created and operates in the same way.

The deletion of a member causes the deletion of its member directory entry in the library block of level 1; all entries following this deleted entry are shifted to the left. If all member directory entries of one library block are deleted, the library block can be reused and its entry on the next higher level is deleted.

According to this procedure the number of index levels for one sublibrary can increase and decrease depending on the number of members currently existing in this sublibrary. All library blocks of one level are chained via PRBAs and contain entries for the next lower level except the entries of level 1 which contain the member directory information.

Because the member index is updated while there can be concurrent read operations upon it, for a search request each index level must be read until a key is found which is greater or equal to the search argument (possibly more than one library block can be affected on each level). If the search argument is greater than the last key of one index level, the search must be continued on the library block addressed by that last index entry (for a level greater one). The search stops at index level 1 if a member is found whose key is greater or equal to the search argument or the end of the directory is reached.

Member

The library blocks of a member need not be physically contiguous. The LBs are chained together to establish the sequence. Records within the LBs are in physically contiguous order.

Record Structure: The layout of the records in a member library block depends on the type of the member:

- Phases and Dumps are each stored in one record which can span over a number of library blocks (record type UNDEFINED)
- The records of member types OBJ, PROC, Source or User have the record structure shown in Figure 10 (record type FIXED). Each 80-byte record is compressed by eliminating the blanks. The information is stored starting at the beginning of the library block while the record length is stored immediately before the library block control field. Thus, the length fields of newly added records are stored in reverse order at the end of the free space.

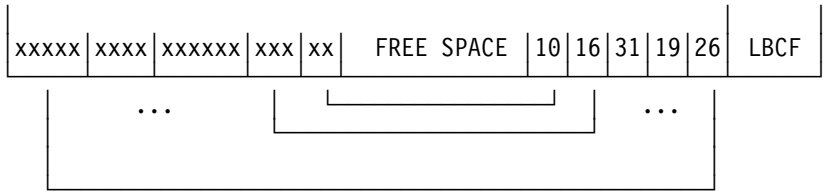


Figure 10. Library Block with 5 Source Book Records

Member Space Allocation: The algorithm for the space allocation of members searches for the first available library block. The search always begins at the point where the last search ended (sequential search). As long as no deletion has occurred in the library, the members are stored contiguously. Otherwise, the algorithm finds one or more library blocks within the library which have become available because of a deletion and starts to store the member data there. If the number of blocks is insufficient for the member, the algorithm goes on searching for the next free library block(s).

If a member is stored into more than one gap it becomes scattered. Scattering of a member occurs if

- the free library space is already scattered, or

- the number of output buffers is too small to keep the entire member in storage (for example if the partition size is too small) and more than one partition catalogs concurrently in the library.

To allow minimization of START I/Os when retrieving a member, the number of contiguous LBs starting from the beginning of the member is stored in the member descriptor. In addition, the control field of each LB contains a number of LBs which follow contiguously this LB.
The library space allocation strategy together with the contiguity counters allows fast retrieval of members; this is especially important for loading phases.

Space Reclamation Chain

Each sublibrary of a library has a Space Reclamation Chain which can be empty. The Space Reclamation Chain contains members whose directory entries are removed (due to a delete function), but whose library blocks are not yet added to the Free Space Inventory and therefore are not yet available for re-usage. Also LBs which are discarded from the member index due to index changes can be contained in the Space Reclamation Chain. The reason not to free these LBs immediately when they are taken out of a chain is that there may be an access to this sublibrary at that time. The LBs are chained together by means of their backward pointers so that the forward pointers still show a valid chain. Thus, a member which has been included in the Space Reclamation Chain can no longer be found in the member directory, but may still be pointed to by in-storage directories.

Librarian - Library Access Services

Overall Structure

To store, retrieve and modify data within a library, a set of services, the Library Access Services, are provided.

The Library Access Services are hierarchically structured into 5 levels. Each level performs a given task, providing an interface to the higher level which uses the functions of the lower levels. A 6th level is introduced with VSE/ESA* 1.3 to call librarian functions from a transaction oriented environment.

The 6 levels are:
Level 0: Supervisor Services
Level 1: Space Management, DASD Access
Level 2: Library Management
Level 3: Command Execution and Execution of the API
Level 4: User Commands / Call Interface / API
Level 5: Transaction Server
Level 1 to level 3 and the Call Interface are internal Librarian levels and are described in more detail in this section. The Transaction Server and the internals of the API will be described in *VSE/ESA Diagnostic Reference: OCO* . Level 0, the Librarian commands and the Application Programmer Interface (API) of Level 4 are described in the *VSE/ESA Diagnosis Reference: Supervisor*, the *VSE/ESA System Control Statements* and the *VSE/ESA System Macro's User's Guide*, respectively.

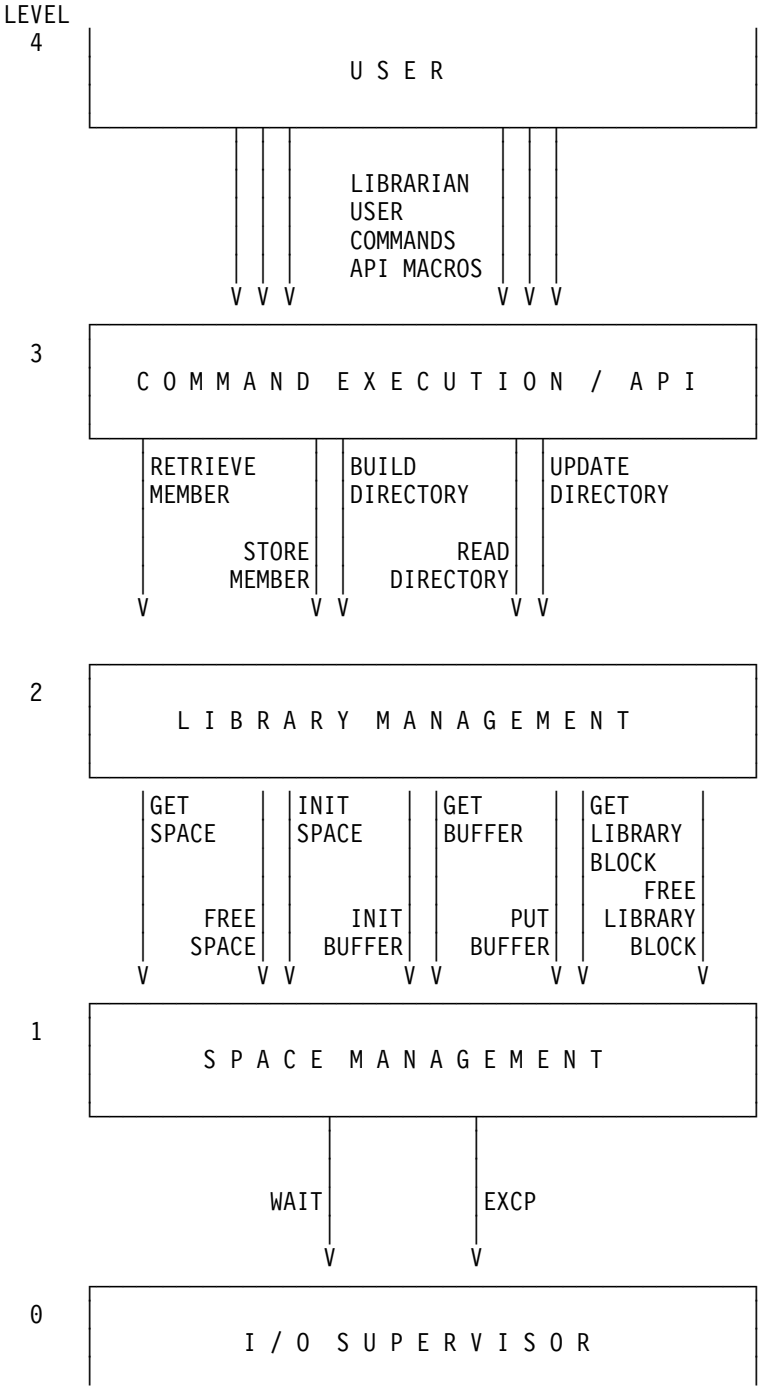


Figure 11. Librarian Level Structure

Figure 11 shows the structure of the Library Access Services:

- The interface to Level 0 is given by the I/O Supervisor which provides the macros EXCP and WAIT. These two macros allow the Librarian to perform all necessary operations to store, retrieve, delete etc. information either on CKD or FBA disk.
- Level 1 works with the physical layout of the library and is responsible for the organization of the library blocks on disk. This includes the mapping of the PRBA to the real disk address. Level 1 provides a set of functions to manage the disk space (buffer and space management).

- Level 2 manages the library block chaining via the PRBAs and maintains the organization of the directories and the member indexes. Level 2 provides the logical structure of a library; a set of macros perform all logical operations like create-directories, update-directories, insert-member etc.
- Level 3 gets the Librarian user commands as input, analyzes them and invokes the necessary level 2 macros to perform the desired function.
- Level 4 is the user who calls the Librarian with "EXEC LIBR" and asks for specific library operations by using the Librarian user commands. They can be supplied as an input stream either via SYSRDR or, separately, via SYSLOG or an ICCF terminal. Or, they can be invoked from a program using the Librarian Call Interface. is the Transaction Server (TPS). The TPS runs in an own Partition and communicates with the requestor task via XPCC. The requestor puts his librarian requests (commands or API) into so called frames which are send to the TPS via XPCC. The results are send back in frames to the requestor. For building up frames there exist a special macro in the API.

Level 1 and 2 services are contained in phase $IJBLBR which is executed as reentrant code in the SVA. Level 3 services run in the partitions except the API functions. The modules executing the API functions (INLPIALC, INLPIASV, INLPIAAC) and the module for building up frames for the TPS (INLPFRSP) are integrated into the phase $IJBLBR.

Level 1 - Space Management

Level 1 is responsible for 3 different functions:

- Buffer management
- Library space management
- Physical I/O.

The operations on Level 1 allow the other levels to work device-independently and to ignore the physical layout of the library, for example how many extents are used.

Buffer Management:

There are two types of buffers

- shared buffers
  used for housekeeping purposes such as the library descriptor, bit map, sublibrary descriptors etc.
- private buffers
  used for the information stored in the members.

The user of the Librarian has no influence on the number, type and size of the buffers except by specifying the partition size. Module INLPGST (Get STorage) is used by the Level 3 modules to determine the free space in the partition and to reserve space for the needed buffers.

Module INLPBBUF (Build BUFfer) builds 2 buffer pools for the Librarian using the free space of the partition. Each buffer pool has a number of buffers of one buffer type. The layout of a buffer pool is shown in Figure 12 on page 19.

It includes

- a buffer control block (BUCB) as a work area to control the buffers (see data area INLCBUCB)
- a buffer header (BHDR) for each buffer (see data area INLCBHDR)
- a number of buffers as data areas, each having the size of one library block

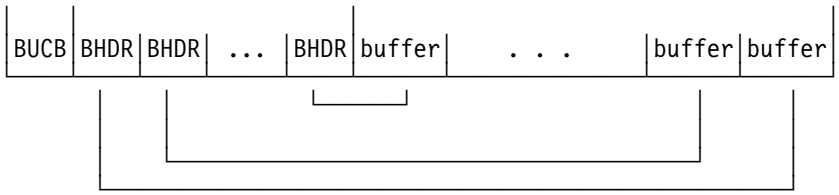


Figure 12. Layout of a Buffer Pool

Each buffer in a buffer pool can be in one of three states:

- free
- in-use
- pre-emptable

The state free means that the buffer is available for use, the state in-use means the buffer is currently being accessed, and the state pre-emptable means that the buffer has been previously accessed and is no longer in use but its information can be useful for another request. The pre-emptable state allows to save Start-I/Os (SIOs) and is helpful for storing directory information. A pre-emptable buffer contains the PRBA of the accessed library block in its buffer header.
Within the pool all buffers of the same state are chained together in their corresponding buffer headers. The anchors of these queues are stored in the buffer control block. After the execution of INLPBBUF, all buffers are in the free queue.

The Level 2 services manage and access the buffers using the modules INLPGBUF (Get BUFfer) and INLPFBUF (Free BUFfer). INLPGBUF takes the first free buffer off the requested buffer pool and puts it into the in-use queue. If the free queue is empty, the first buffer of the pre-emptable queue is taken and marked in-use (first-in - first-out strategy).

A call of INLPFBUF takes the buffer out of the in-use queue and puts it either into the free queue or into the pre-emptable queue depending on a request parameter.

A call of INLPGBUF with parameter OLD means that the pre-emptable queue has to be searched for a given PRBA to avoid a SIO. If the PRBA is not found in this queue, a free buffer if possible, otherwise a preemptive buffer is taken and a disk access is necessary to get the information.

Space Management: Space management is responsible for the initialization of the disk space and the Library Header during the execution of the DEFINE LIBRARY command and during an extension of a library in VSAM managed space; it is also responsible for the allocation and freeing of library blocks on disk for the indexes and for member information.

Initialization is done by module INLPBSPA (Build SPAce). It distinguishes between CKD and FBA disks. For CKD disks it has to format all tracks of all extents of the library into records of the same library block size. This is not necessary for FBA devices.

In the next step INLPBSPA writes at the beginning of each extent an Extent Descriptor (EXTD). The extent descriptor of the first extent is followed by a Library Space Descriptor (SPAD). The extent descriptor on each extent is followed by the free-space bit map for the extent which is initialized by setting all bits to zero except the bits for the library blocks containing the Free Space Map.

If new library blocks are needed - either for new indexes or for member data - module INLPGSPA (Get SPAce) is used to allocate them on disk. It gets as a parameter the number of library blocks needed and searches the Free Space Inventory for bits with value zero (free library blocks). If it succeeds, it changes these bits to one to indicate that these library blocks are now reserved. If there are not enough library blocks available on disk, the library-full condition is raised.

A second entry point, INLPFSPA (Free SPAce) in module INLPGSPA, is used to free DASD space by setting the corresponding bits in the bit map to zero. It is invoked, for example, as a result of the DELETE command of Level 4.

Physical I/O: Physical I/O is prepared by module INLPLBGP (Library Block Get/Put) and done by module INLPDOIO (DO Input/Output).

Module INLPLBGP gets as its input a list of buffers with their buffer headers. The buffers are chained via the I/O request queue field BIQUE of the buffer header (see Figure 13).

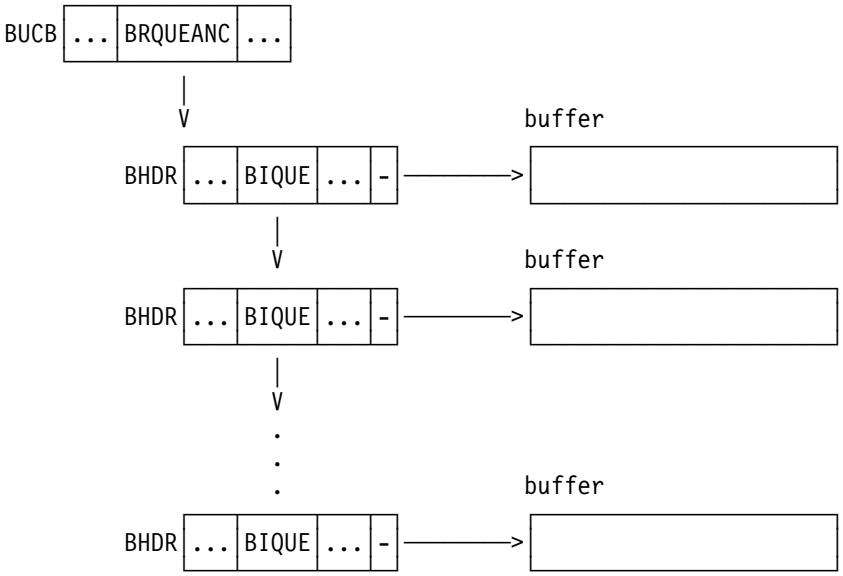


Figure 13. Buffer Request List

The anchor of this list is stored in the buffer control block (BUCB). The requested library blocks are addressed by their PRBAs and these are translated into the physical disk addresses (cylinder, track, record for CKD device, block number for FBA device).

In the second step, INLPLBGP sorts the buffer request list by physical addresses to minimize the Start-I/Os. In an iteration process it builds the CCW chain for processing one SIO, i.e. adjacent records on the same track, extent and volume, prepares the Channel Command Block (CCB) and calls module INLPDOIO via the Supervisor SENTER interface to obtain physical I/O capabilities (physical addressing). INLPDOIO submits the request via EXCP and WAIT macros. After completion of the EXCP request, INLPDOIO releases the physical I/O capabilities and returns to INLPLBGP.

Level 2 - Library Management

Control Blocks: The Level 2 services form a uniform macro interface to get access to a library, a sublibrary, a member, or a member record.
All operations are built in the same way: They require a Library Access Method Block (LAMB, see data area INLCLAMB) which contains the access related information which is not bound to a specific library, and a Library Request Parameter List (LRPL, see data area INLCLRPL) which contains all the parameters defining a specific request.

The buffer storage required for a read or write operation on a library must be provided by Level 3. Two types of buffers are required: shared buffers connected to the LAMB are used for all index I/O, and private buffers connected to the LRPL are used for all member I/O.

Information about the accessed library/sublibrary is contained in the Library Control Tables (LCTs) and passed to the Level 2 services (via the LBRACCES macro).

Control information about the library, sublibrary, or member being accessed is held in corresponding Access Control Blocks invisible to Level 3.

To understand the Level 2 services, it is helpful to understand the control table structure of the Librarian and its relationship to the Level 2 control block structure (see Figure 14 on page 22).

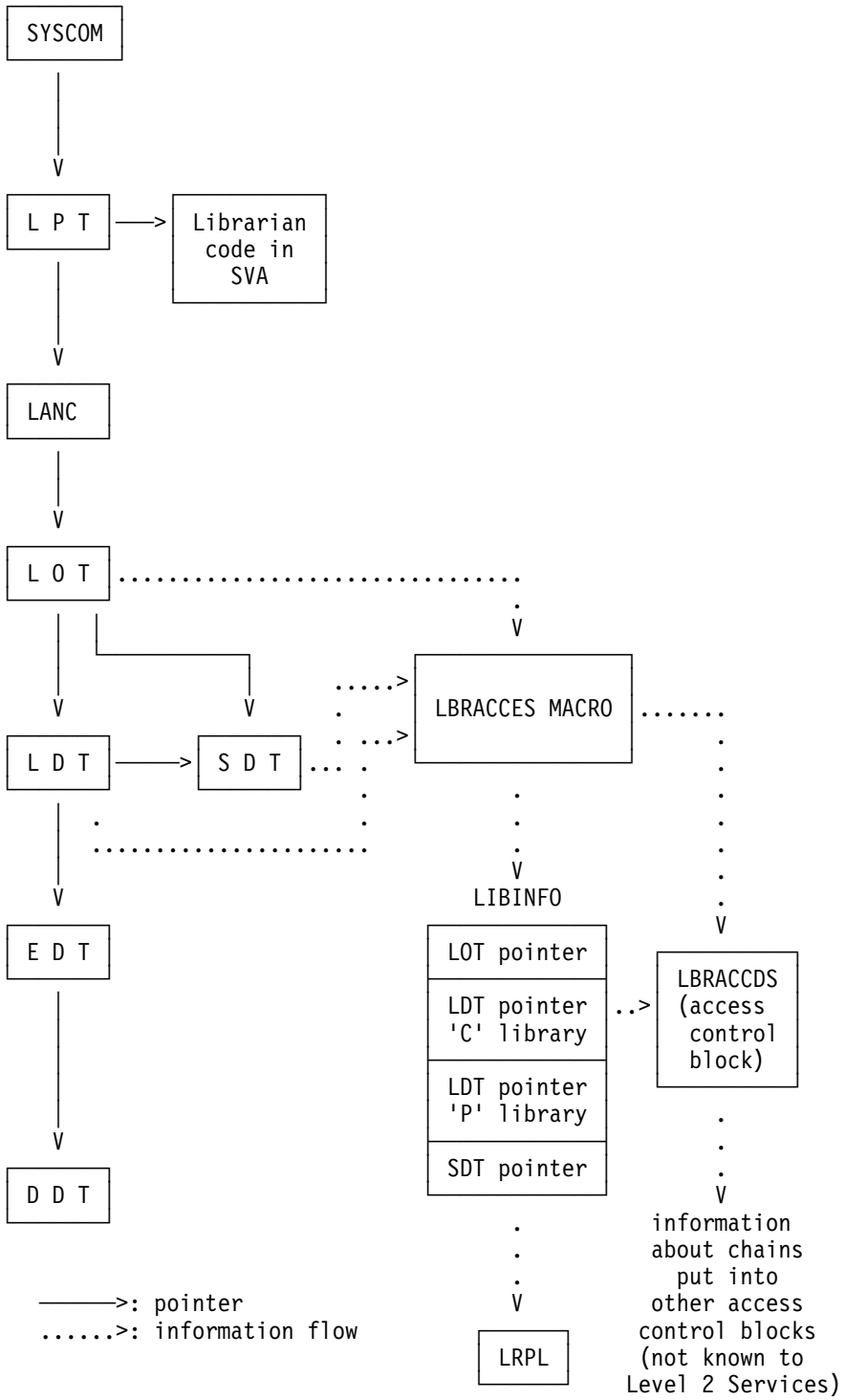


Figure 14. Connection between the Library Control Tables and the Level 2 Interface

The library control table chain starts with the System Communication Region (SYSCOM - see *VSE/Advanced Functions Diagnosis Reference: Supervisor*). Its field IJBPLCT contains the address of the Library Pointer Table (LPT - see data area INLCLPT). The LPT points to the library services code in the SVA and contains the anchors for the Library Control Tables:

- The Librarian LOT anchor table (see data area INLCLANC)
- the Library Offset Tables (LOTs - see data area INLCLOT)
- the Library Definition Table (LDT - see data area INLCLDTE)
- the Sublibrary Definition Table (SDT - see data area INLCSDTE)
- the Extent Definition Table (EDT - see data area INLCEDTE)
- the Device Definition Table (DDT - see data area INLCDDTE)
- the OPEN Identification Table (IDT - see data area INLCIDTE)

The Librarian anchor table (LANC) contains a row of anchors (pointers) for each partition. These pointers reflect the JCL-LIBDEF statements and the task related 'LIBDEF' respectively. They point to a set of specified library chains (LIBDEFs) within the common LOT-Pool. A Library Offset Table (LOT) contains the library chains established by a LIBDEF command or the LBRACCES macro. For each partition, there exists a LOT for each possible type of the LIBDEF statement (PHASE, OBJ, SOURCE, PROC, DUMP) and for type LBR whose chains can be defined via the LBRACCES macro and which are established on a task basis. A LOT entry mainly consists of two offsets, pointing to the corresponding LDT/SDT entries for a specific library/sublibrary.

The Library Definition Table (LDT) and the Sublibrary Definition Table (SDT) keep the definitions of all libraries/sublibraries currently known to the system. Each LDT entry points to a list of Extent Definition Table Entries (EDT Entries) describing the physical locations of the library extents on the DASD. Each EDT entry contains a pointer to a Device Definition Table (DDT) describing the device on which the extent resides. For each library which is contained in the LDT there exists an entry in the OPEN Identification Table (IDT). This entry contains information about the library which allows the Librarian to avoid a further OPEN when another entry for the library shall be made in the LCTs.

The bridge between the Library Control Tables and the Level 2 interface is established by the LBRACCES macro. This macro defines a library/sublibrary chain and stores parameters and other information in an area called the Library Access Control Information Area (see data area LBRACCDS).

LBRACCES builds also a Library Information Area (see data area LIBINFO) which contains the pointers to the LOT entry, to the LDT entry of type 'C', to the LDT entry of type 'P', and to the SDT entry. A type 'C' LDT entry is a complete LDT entry while a type 'P' LDT entry refers to a library which is already defined by another LDT. It only contains the library name and a pointer to the corresponding 'C' entry. A type 'P' entry exists when a library can be accessed also under another name (i.e. a different file name in the DLBL statement is used) by the same or another partition. A pointer to the LIBINFO area is stored in the LBRACCDS.

Library Services: The Level 2 library services can be roughly divided into two categories. The services belonging to the first class are responsible for the creation and deletion of libraries, sublibraries and members while the services of the second class are used to retrieve the stored information.

Creation and Deletion:

Create Library: To create a library (see Figure 15) the first step is to invoke macro LBRACCES with parameters LEVEL(LIB) and DEFINE(NEW) first. It calls module IJBLBHLS which calls modules IJBLBBRN and IJBCTUPD. Module IJBLBBRN opens the library using the OPEN(OUTPUT) macro and builds the LDT, EDT and DDT entries. Module IJBCTUPD includes these entries in the Library Control Tables. Now a connection to the library is possible.
The next step is to establish the connection to the library by invoking macro INLMLCON with the parameter CONNECT(NEW). It calls module INLPLCON (Library CONnect) which claims GETVIS Space as working space for the Library Access Control Block (LACB) and calls module INLPCONL (CONnect to Library). This one initializes the shared buffers by calling INLPBBUF, formats the library with INLPBSPA and finally builds the Library Descriptor. After this step the library exists. Finally the library is disconnected via the INLMLDIS macro to release all system resources.

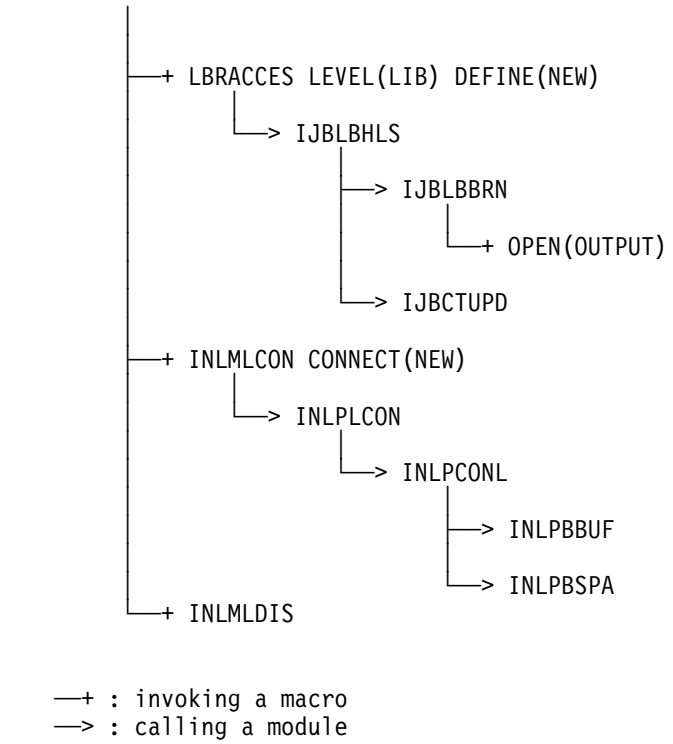


Figure 15. Creation of a Library

Create Sublibrary: A sublibrary can only be created in an existing library. This is done by invoking macro LBRACCES with parameters LEVEL(BOTH) and DEFINE(NEW). An OPEN of the library can be omitted if the library is already opened and the corresponding LIBINFO is provided in conjunction with parameter LEVEL(SUBLIB).

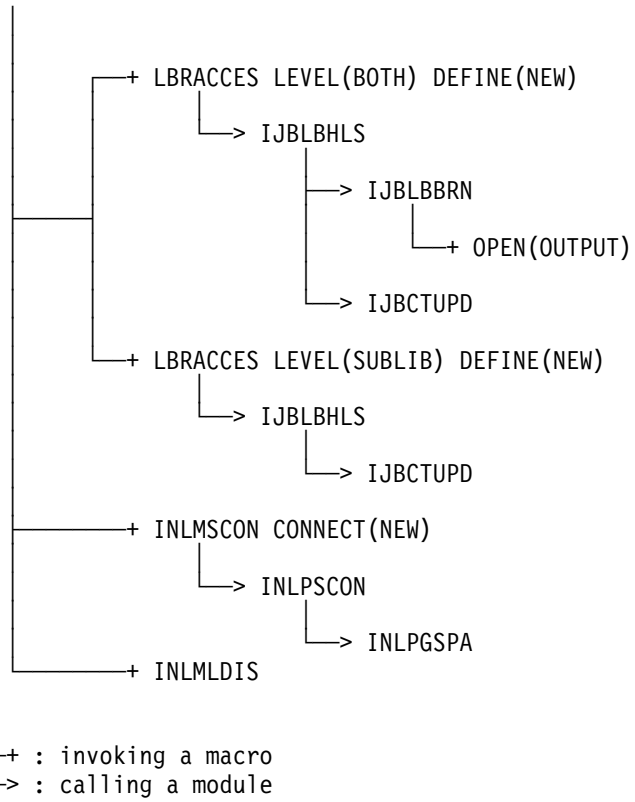


Figure 16. Creation of a Sublibrary

As shown in Figure 16 LBRACCES calls module IJBLBLHS which calls - if necessary - module IJBLBBRN to open the library and to build the LCT entries. Then IJBLBHLS calls module IJBCTUPD to include the entries into the Library Control Tables, to build the SDT entry, and to add it to the SDT.
In the next step macro INLMSCON with parameter CONNECT(NEW) establishes the connection to the sublibrary. It calls module INLPSCON (Sublibrary CONnect) which implicitly connects to the library, allocates the Sublibrary Access Control Block (SACB), creates a sublibrary descriptor and stores it into the Library Header. It also calls module INLPGSPA to allocate one library block permanently to the sublibrary. This block is used as the first Member Index Level 1 block and is necessary to lock/unlock the sublibrary. It will be kept during the whole life-time of the sublibrary. Finally, the library is disconnected to release all formerly requested system resources.

Create Member: The creation of a member (see Figure 17 on page 26) requires the existence of the LDT, SDT, EDT and DDT entries. It is done by invoking macro INLMMCON with parameter CONNECT(NEW) which calls module INLPMCON (Member CONnect). This module implicitly connects to the sublibrary, allocates the Member Access Control Block (MACB), and creates a dummy member directory entry in the partition storage. This is followed by invoking macro INLMPUTR (PUT Record) which calls the Level 1 module INLPGBUF with a request for all free private buffers. It receives as return values the number of buffers and their addresses. Then it writes the data records into the buffers and updates the dummy entry. If the number of records exceeds the number of free buffers, INLMPUTR calls INLPLBGP to write the content of the buffers on disk and return the freed buffers.

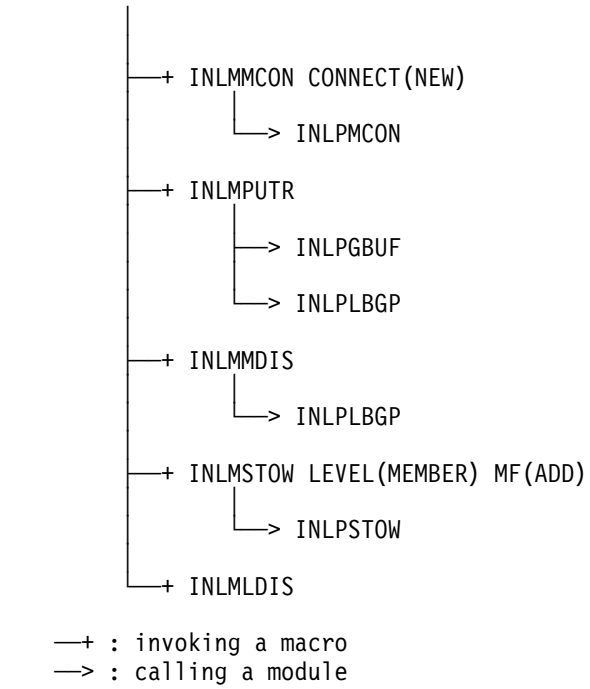


Figure 17. Creation of a Member

After the last record is written into a buffer, macro INLMMDIS is invoked which writes the last buffers on disk using INLPLBGP and completes the dummy directory entry.

At this time the member is disconnected but the connection to the sublibrary still exists. Macro INLMSTOW with parameters LEVEL(MEMBER) and MF(ADD) is invoked to catalog the member into the sublibrary. The macro calls module INLPSTOW which updates the sublibrary descriptor, loads the member into the SVA if it is possible and desired, splits the library blocks of the member indices if it is necessary and stores the new member directory entry into the member directory on disk. After the successful completion of INLMSTOW the new member is cataloged. Finally, macro INLMLDIS is issued to free the system resources (the sublibrary is implicitly disconnected).

Delete Member: To delete a member a connection to the library and sublibrary is necessary. Macro INLMSTOW with parameters LEVEL(MEMBER) and MF(DELETE) realizes the deletion by calling module INLPSTOW. This module removes the member's directory entry from the Member Index and releases the member space. If the sublibrary is not uniquely assigned and the space reclamation attribute of the sublibrary is AUTOMATIC, the freed library blocks are inserted into the Reclamation Chain. In this case, a new access to the member is not possible but a parallel one can be finished.
Entry INLPFSPA which actually releases the library block is only taken if the sublibrary is uniquely assigned or owns the space reclamation attribute IMMEDIATE.

Rename Member: The renaming of a member is done by the macro INLMSTOW with the parameter MF(RENAME). This function combines the ADD and the DELETE functions of the INLMSTOW macro.

Delete Sublibrary: The deletion of a sublibrary (see Figure 18 on page 27) is only possible if the sublibrary is uniquely assigned. Otherwise the execution is terminated.

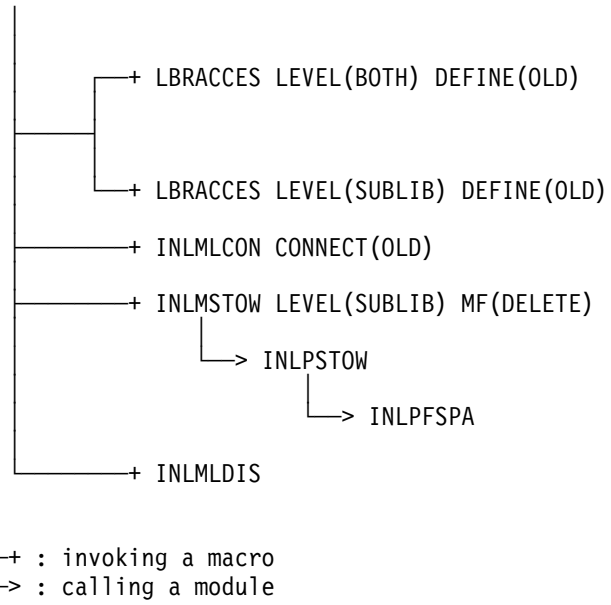


Figure 18. Deletion of a Sublibrary

The deletion requires to invoke (1) macro LBRACCES with parameters DEFINE(OLD) and either LEVEL(SUBLIB) - if the corresponding library is already opened - or LEVEL(BOTH) - if the library has to be opened - and
(2) macro INLMLCON with parameter CONNECT(OLD). After the library is connected, macro INLMSTOW is invoked with parameters LEVEL(SUBLIB) and MF(DELETE). It calls module INLPSTOW which removes the sublibrary descriptor from the Library Header and releases all library blocks containing the member indices and the member data by using INLPFSPA.

Delete Library: The deletion of a library cannot be done with Level 2 Services but is a task of the Level 3 routines.

It is performed by marking the library descriptor as deleted. For deleting libraries in BAM space, module INLPDEL locks the library using macro LBRUPDAT and calls the Common VTOC Handler. This program eliminates the VTOC entry of the library on the disk.
If the user tries to delete a library in VSAM managed space, he receives a message to use the VSAM service 'IDCAMS DELETE' in order to delete the VSAM cluster.

Information Retrieval: To access the information stored in a library either in the directories or in the member it is always necessary to invoke the LBRACCES macro with parameters LEVEL(LIB) and MF(GET) which provides the library information of the Library Control Tables.

Access to Directory Information: To get the information from the library descriptor macro INLMLCON has to be invoked, and macro INLMGDIR with parameter LEVEL(LIB) provides the access to the library descriptor. Using this macro with parameter LEVEL(SUBLIB) provides sequential access to all sublibrary descriptors of the connected library.
Using macro INLMSCON followed by invoking INLMGDIR with parameter LEVEL(MEMBER) establishes sequential access to all member directory entries of the connected sublibrary.

In distinction to the sequential access of macro INLMGDIR to all directory entries it is possible to access a selection of directory entries using macro INLMBLDL. This does not need a preceding INLMSCON but gets the connection implicitly. Depending on the LEVEL parameter, which can be either LEVEL(SUBLIB) or LEVEL(MEMBER), macro INLMBLDL gets as input a chain of libraries or sublibraries, respectively. These chains are built either by the LIBDEF statement or the LBRACCES macro.
INLMBLDL scans the chain for the desired directory entry and returns a name or the whole information of the entry depending on the INFORM-parameter (INFORM(KEY) or INFORM(ENTRY)). For LEVEL(SUBLIB) and INFORM(RESOURCE) resource names which are used for locking purpose can be obtained.
The request parameter of INLMBLDL can be NONGENERIC or GENERIC. In the first case it is possible to provide a single name or a list of names. In both cases it retrieves the first entry in the chain whose name matches to the given name totally (NONGENERIC) or partially (GENERIC). In the case of GENERIC it can happen that the work area is too small to satisfy all requests. Then it is possible to reinvoke INLMBLDL with the CONTINUED parameter and the starting point of the residual search. After completion of INLMBLDL all connections will be released.

Access to Member Information: This is done via a provided chain of sublibraries. Macro INLMFIND uses module INLPFIND to scan the chain until the first occurrence of the member name. It returns to the caller and gives him the connection to the member.

Macro INLMGETR calls entry point INLPGETR (GET Record) of module INLPPUTR (PUT Record) to access the member records. This can be done either sequentially to all records or with an offset to one specific record. INLPGETR performs its task by calling the Level 1 modules INLPGBUF, INLPFBUF and INLPLBGP.

The DTFSL interface used to access source members (macros, include- books) is preserved from the DOS Librarian. All of its functions (FNDSL, GETSL, NTSL, PTSL) remain object and source code compatible. The requests are passed in the form of a request list (DTFSRQL) which indicates the operation requested and points to all resources needed. DTFSL calls phase $IJBLBSL which resides in the SVA. A first call from a partition causes the sublibraries in the SOURCE search chain of the LIBDEF statement to be made accessible. This is done by issuing the Level 2 macros INLMFIND, INLMGETR, INLMNOTE, and INLMPOIN.

The code of the DTFSL macro is invoked by the internal imperative macros FNDSL, GETSL, NTSL, and PTSL. They have the following functions:

FNDSL: Find a book and save its disk address if found. An alternative branch address must be specified in case the book is not found. Register 1 points to a 9-byte book name (1 byte for source type name, 8 bytes for member name).

GETSL: Retrieve a book sequentially, one record with each GETSL request.

NTSL: Return to the caller the position where retrieving is to continue.

PTSL: Restore the position of earlier processing.

For releases prior to DOS/VSE, the position of a NTSL/PTSL request is contained in register 1. For DOS/VSE and VSE systems, register 1 contains the address of an entry into an internal LIFO stack called note word table if the caller does not provide, in register 1, the address of a field where the macro is to store the information.

The DTFSL has the following format:

```
&DTFSL   DTFSL  NOTEPNT=(NO|YES),
                PRIVATE=(NO|YES),
                ERROR=label,
                BL=(NO|YES),
                LBR=(NO|YES),
                NADR=(NO|YES),
                NLEN=constant

where the parameters have the meaning:

NOTEPNT=NO ......... Only FNDSL and GETSL allowed.
NOTEPNT=YES ........ FNDSL, GETSL, NTSL, and PTSL allowed.
PRIVATE=NO ......... Operates on system sublibrary only.
PRIVATE=YES ........ Operates on all sublibraries of search chain.
LBR=NO ............. Compiler and Assembler option.
LBR=YES ............ Librarian processing option (DOS/VSE and VSE).
BL=NO .............. DTFSL runs only on DOS/VSE and VSE.
BL=YES ............. DTFSL runs on a release prior to DOS/VSE.
ERROR=label ........ Entry point of error routine (for bad records).
NADR=NO ............ Note information is returned in register 1 or in a
                     user specified register, for a release prior to
                     DOS/VSE. For DOS/VSE and VSE systems, the note
                     information is kept in GETVIS space with 20 nesting
                     levels.
NADR=YES ........... The caller sets up register 1 for NTSL/PTSL to point
                     to a field which contains the note information.
NLEN=constant ...... Length of the note information field specified with
                     NADR=YES.  It must be 32 bytes (length of data area
                     INLCNTPT) or higher. Otherwise, a default length of
                     12 bytes is taken.  Since 12 bytes do not allow to
                     save the LBID value, this value is lost after NTSL/
                     PTSL.  This has the effect that the library blocks
                     cannot be checked anymore whether they belong to
                     the member currently retrieved or not.
REENTR=YES|NO........If YES is specified reentrant code will be
                     expanded. The operand will only be considered for
                     BL=NO. To provide reentrant code the user has to
                     supply a work area. This work area must be addressed
                     with Register-13 on each request. The work area must
                     start on a DW boundary.
                     A DSECT will be generated for this area. The length
                     is equated by DTFSLPRL.
                     On the first FNDSL request the following fields
                     are to be set:
                             PRCID = 'DTFSLPRC'
                             DTFSLSW = '80'X
                     PRCID will be used for the validation of that area
                     on each subsequent request.
                     The area must not be manipulated by the user.
```

```
Processing Error Codes : ( Message 3M17I .. Program Error )
  Register 15 contains the Error Code:

    X'101' : Nesting depth of NOTE exceeded
    X'102' : No expansion for NOTE/POINT or READ
    X'103' : Too many POINT requests
    X'104' : Invalid request or Library Open failure
    X'105' : GET, NOTE, POINT, or READ requested w/o FIND
    X'106' : Concurrent Deletion of member being read
    X'107' : For REENTR=YES no valid work area
```

Level 3 - Command Execution

Level 3 provides the interface to the user. It initializes the Librarian tables during the execution of EXEC LIBR and interprets the Librarian commands.

Root Phase: The execution of the JCL statement EXEC LIBR involves a load of the Librarian Root Phase LIBR. LIBR mainly consists of

- Initialization module INLPMAIN
- Parser modules INLPSYNA, INLPSYNX, INLPSYPA
- Parser tables
- Message handler modules INLPDIAG, INLPWOR, INLPWTO, INLPWTP
- Dictionary tables
- Several Level 3 service routines
- DTFs and I/O areas for SYSIPT, SYSLOG, and SYSPCH

Figure 19 on page 32 shows the execution of the root phase.
After module INLPMAIN is loaded it builds the Level 3 Communication Region COMR (see data area INLCCOMR) and the Command Parser Control Block CMDP (see data area INLCCMDP). The communication region is used for the communication between the root phase and the command processing modules. INLPMAIN distinguishes between the different execution modes like BATCH, SYSLOG or a call from other programs (SPF, etc.), determines the environment of the Librarian like MSHP, migration etc. and indicates this by setting the corresponding flags in the communication region. It also stores there the addresses of the DTFs and the I/O areas.
The command parser control block is used for the internal format of the commands.

After the initialization INLPMAIN branches to the syntax checker. It calls module INLPSYNA (SYntax NAme) which reads the first input line and checks it for a valid command name. All other input lines are read by other modules but if they recognize their command end they branch back to INLPSYNA.
If the EOF condition or the END-command occur, INLPSYNA returns to INLPMAIN using the entry INLPEND to leave the Librarian in a clean way. Otherwise it stores the command name into the parser control block, calls module INLPSYPA (SYntax PArameter) and hands over to it the address of the parameter table.
INLPSYPA checks the parameters for these commands and sets the flags in the command parser control block. If the command is syntactically correct, the execution phase will be loaded behind the root phase into the partition.

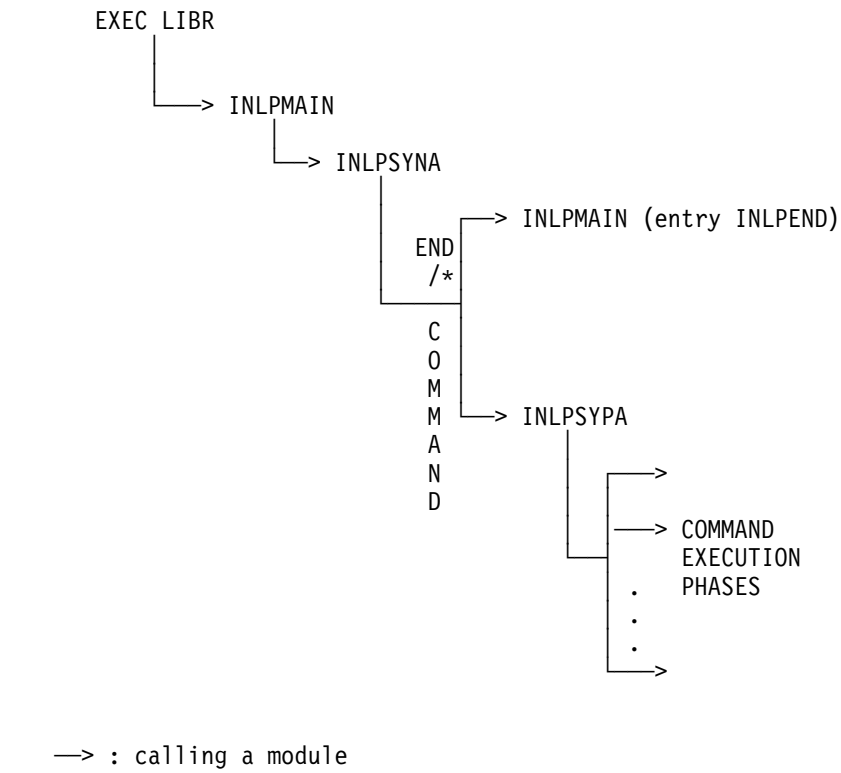
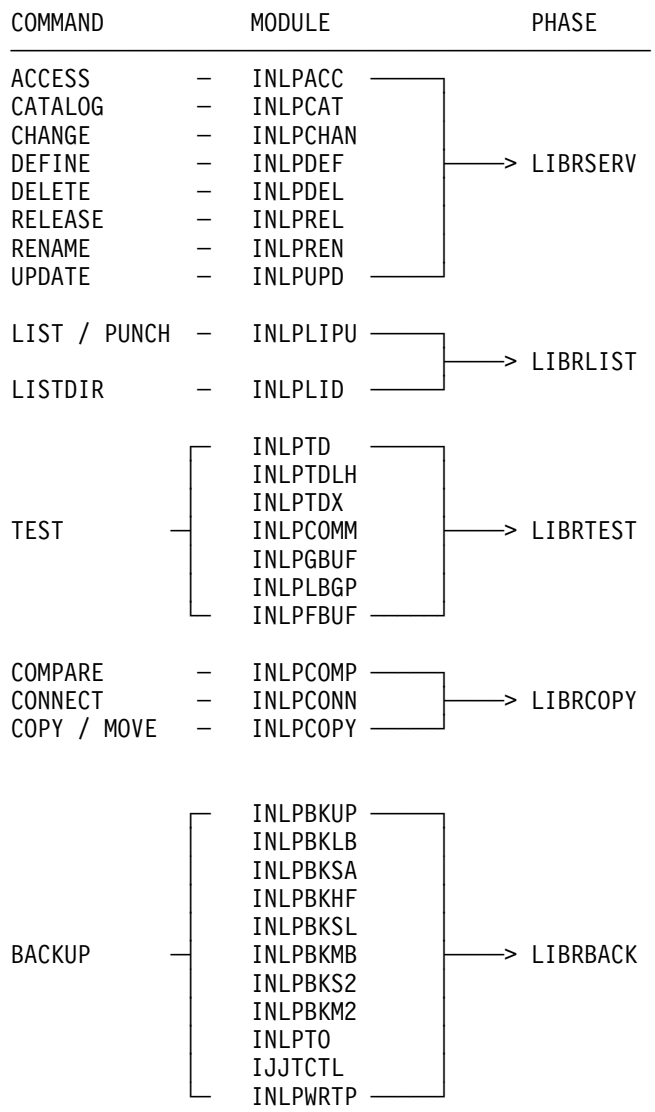
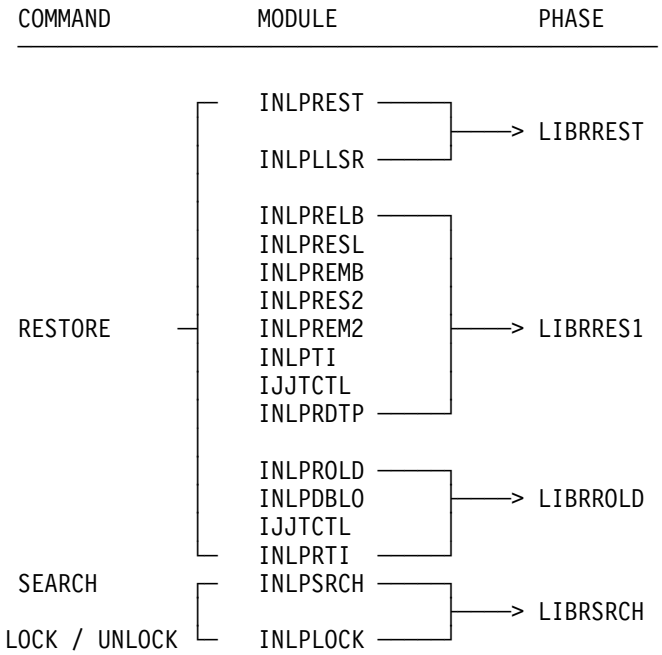


Figure 19. Execution of the Root Phase

Commands: The command processing modules are contained in executable phases. According to the command name stored in the INLCCMDP data area a phase is loaded. Since some phases contain several command execution modules they have a selection routine (INLPLEV3) at the beginning which branches to the desired module. For migration purposes module INLPMIGR is contained in several phases. The relationship between the user command, the module names and the execution phase is given below:





Notes:

1. The INPUT command needs no module but causes just a switch of SYSLOG mode to SYSIPT mode in the INLCCOMR data area.
2. The commands for conditional execution (ON, GOTO, /.) are processed in module INLPSYNA (phase LIBR).

The execution steps of all command processing modules are similar. The module

1. allocates the remaining space of the partition for an internal working area.
2. makes the semantic checks of the parameters.
3. provides the environment for the Level 2 services.
4. accomplishes the command processing by invoking Level 2 services.

Librarian Call Interface: This section describes the invocation of the Librarian from a program within a partition and the input/output interfaces provided.

The Librarian may be invoked from a (problem) program by loading the Librarian root phase LIBR into partition storage and branching to its entry point. The Librarian will take the rest of the partition as working space. Therefore, LIBR must be loaded in the upper part of the partition (recommended size reserved for LIBR and working space: 256 K).

A program may pass commands to the Librarian and receive the output in its own areas. It may get control back from the Librarian whenever information is put into a caller-supplied area.

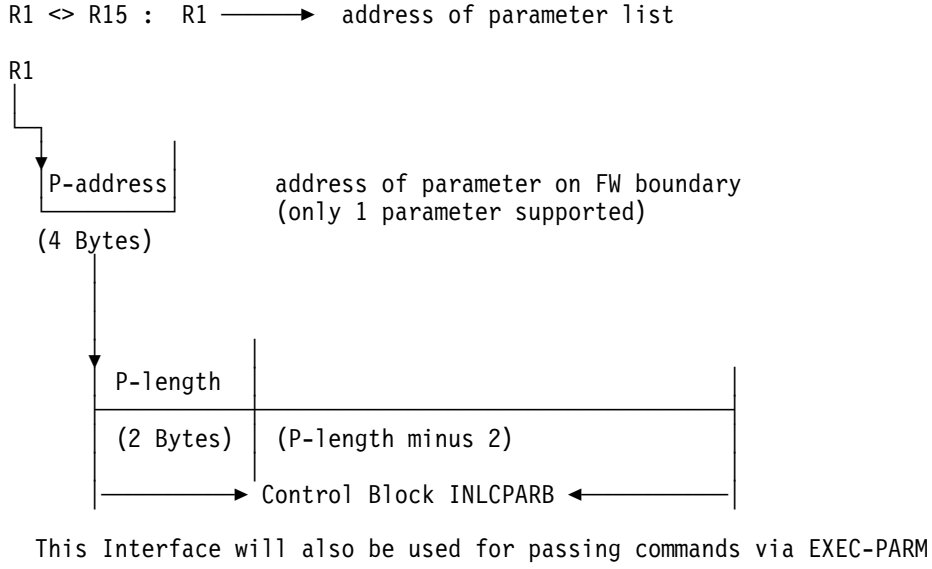
Interface Description:

Invocation of the Librarian:

```
Load phase LIBR (entry point in R15).
Load address of 18-FW register save area in R13.
Load address of parameter list in R1.
BALR R14,R15.
```

The Librarian returns to the R14-address and passes back a return code in R15.
The Librarian behaves the same as if the input comes from SYSIPT, that is messages and listings (default output unit) are displayed on SYSLST or the SYSLST Exit is taken.

Passing a Parameter:



This Interface will also be used for passing commands via EXEC-PARM

```
Control Block Layout (use INCLUDE-book INLCPARB):
       DC    AL2(64)        Length of control block (P-length)
       DC    XL2'00'        Reserved
       DC    XL4            Identification of caller
* identifications: SPF - XL4'00', MSHP - XL4'01', SIPO - XL4'02'
                   DSNX - XL4'03'
       DC    AL4            Address of SYSIPT exit routine
       DC    AL4            Address of SYSIPT input area
       DC    AL4            Address of SYSLOG exit routine
       DC    AL4            Address of SYSLOG output area (and
                                  input area for A-messages)
       DC    AL4            Address of SYSLST exit routine
       DC    AL4            Address of SYSLST output area
       DC    AL4            Address of SYSPCH exit routine
       DC    AL4            Address of SYSPCH output area
       DC    XL1            Current line count (SYSLST)
       DC    XL1            Internal function request number
       DC    AL1            Flag byte ( see INLCPARB )
       DC    AL1            Flag byte ( see INLCPARB )
       DC    AL4            User area (not used by Librarian)
* pointers for formatted output functions (only used if an internal
* function request number is specified)
       DC    AL4            Address of formatted output exit routine
       DC    AL4            Address of formatted output area
       DC    CL2            EOD characters for MSHP
       DC    AL1            Flag byte ( see INLCPARB )
       DC    AL1            Return Code of last execution
       DC    CL4            RESERVED
```

The user area is not read or modified by the Librarian. It may be used by the routines invoking the Librarian to save information (for example, information which is used in an exit routine).

Any of the exit routine addresses may be zero, which indicates that this exit must not be taken by the Librarian,

i.e. the Librarian

- reads from SYSIPT if SYSIPT addresses are zero
- writes/punches on SYSLOG/SYSLST/SYSPCH if the corresponding addresses in the control block are zero.

If an exit routine address is not zero, the corresponding area address must also be unequal zero.
Before entering an exit routine, the Librarian loads the address of the input control block into Register 1 :

```
Calling routine:       CALL xyzproc(arg);
Invoked routine:       xyzproc:PROC(arg);  or  xyzproc:ENTRY(arg);
```

Either both addresses (exit and area) must be zero or both addresses must be unequal zero (otherwise: system error).

The SYSLOG/SYSLST/SYSPCH output areas will contain one logical record when the corresponding exit is taken. If the logical record does not fit into the output area provided by the invoking routine, the record is cut without notice.
If the logical record is smaller than the output area, the rest of the area is padded with blanks for SYSLOG/SYSLST/SYSPCH.
In addition, the Librarian provides records with formatted output for the RESTORE SCAN function if the internal function request number is XL1'01' and the corresponding address pair for formatted output is given. The formatted output area may contain one or more records of fixed or variable length. If the

internal function request number is specified then both addresses (formatted output exit and area) must be unequal zero (otherwise: system error).

The input area may contain more than one logical record of 80 bytes. Each input command is displayed on SYSLST or moved into the SYSLST output area (and the SYSLST exit is taken) before the command is executed.
The Librarian processes one input record after the other until either a '/*' (outside member data) is encountered or the input area is exhausted.
If the input area is exhausted before '/*' is encountered, the SYSIPT exit is taken. When '/*' is encountered, a cleanup of the Librarian functions is performed (i.e. ACCESS and CONNECT commands are lost).

If the SYSIPT addresses are zero, the Librarian will read from SYSIPT.

Layout of Input Area:

```
DC    XL2                Length of input area (length >= 88 bytes)
DC    XL6                Reserved
DC    CL(length minus 8) Input record(s)
                                 (record length: 80 bytes)
```

The input area may contain one or more logical records of 80 bytes. They are stored one behind the other without physical delimiters.

Layout of Output Areas (SYSLOG,SYSLST,SYSPCH):

```
DC   XL2                Length of output area (8<length<256 bytes)
DC   XL6                Reserved
DC   CL(length minus 8) Output record (one)
```

The output area contains one output record at a time. The maximum length of the output area is 255 bytes and, consequently, the maximum length of the output record is 247 bytes.
For SYSLST: The first byte contains the Printer Control Character.
For SYSPCH: No Control Character is passed.
Layout of RESTORE SCAN formatted output record:
(Only used if sublibraries are on tape, i.e. for a tape containing only products)

```
DC    AL2(80)        Length of RESTORE SCAN record
DC    XL2'01'        RESTORE SCAN formatted output identification
DC    CL16           BACKUP file id
DC    AL4            Minimum library blocks required
DC    AL4            Number of cylinders required for 3330
DC    AL4            Number of tracks required for 3330
DC    AL4            Number of cylinders required for 3340
DC    AL4            Number of tracks required for 3340
DC    AL4            Number of cylinders required for 3350
DC    AL4            Number of tracks required for 3350
DC    AL4            Number of cylinders required for 3375
DC    AL4            Number of tracks required for 3375
DC    AL4            Number of cylinders required for 3380
DC    AL4            Number of tracks required for 3380
DC    AL4            Reserved
DC    XL1'80'        PID-V2-STACKED tape
DC    AL3            Reserved
DC    AL4            Required FBA blocks
DC    AL4            Reserved
```

(See data area INLCFSRL.)

Restrictions: When invoking the Librarian from a program the following restrictions apply:

- The Librarian command INPUT is not accepted.
- The 'Delayed Cancel' feature must not be activated by the calling program since it will be activated by the Librarian (i.e. flag IJBARCNA in field JCSW8 of the partition communication region must not be set).
- System Files assigned to DASD must be closed before calling the LIBR, if the corresponding EXIT will not be taken.
- The Return Code field in INLCPARB is only valid after the execution of a Librarian command. If it is not possible to enter the execution due to a wrong command (Syntax or Semantic error) control is transferred back to the caller with the return code only in Register 15.

Implementation Hints in Librarian Root Phase:

```
   Module INLPMAIN:

     Entry Point:
     Save registers in save area of invoking routine (pointed to by R13).
     Provide own register save area.
     IF R1¬ = R15 & P-address¬ = 0 THEN
     DO;
       IF identification=XL4'00' or XL4'02'
          THEN activate INPUT/EXIT interface.
       ELSE
          IF value='MSHP'
             THEN initialize 'bypass MSHP control'.
             ELSE process command.
     END;
     ELSE; /* no parameter list */

     ......

   Module INLPREAD:

     Processing input records (SYSIPT addresses in control block ¬= 0):
        DO until input record = '/*' (outside member data);
          DO until input area is exhausted;
          Process input record;
          IF input record = '/*' (outside member data)
             THEN indicate EOJ and return;
          END;
        Branch to SYSIPT-exit via 'BALR R14,R15';
        RETURN;
        END;

     ......

   Modules INLPWTP (SYSLST), INLPWTO (SYSLOG), INLPPUN (SYSPCH),
           INLPOUT (formatted output RESTORE SCAN):

     Invoking an exit routine:
       LA   R1,A(pointer-to-input-control-block)
       LA   R13,A(libr-savearea)
       LA   R15,A(exit-routine)
       BALR R14,R15  Exit routine saves/restores registers

     ......
   Module INLPMAIN:

     At EOJ:
       Cleanup Librarian functions;
       Load return code in R15;
       IF identification = XL4'00' or XL4'02'
          THEN GOTO R14;
          ELSE MSHEOJ;
```

MSHP macros for OPEN and EOJ are used.

Conditional Command Execution

Functional Details (Module INLPSYNA)

- Each valid ON statement is stored in a stack with a maximum of 30 entries.
- If more than one ON condition are active, checking starts with the most recently stored ON condition. If this condition is not fulfilled, the preceding condition is checked etc.
- An ON condition remains active till it is overridden by a new one (with the same comparison operator and number or with different ones making the existing condition obsolete) or till End-of-SYSIPT.
- When a new ON condition is stored in the stack, all previous ON conditions are checked if they get obsolete due to the new condition. Obsolete entries are removed and the stack is compressed.

```
Example:        ON $RC > 8 GOTO A           Condition 1
                ON $RC = 8 CONTINUE         Condition 2
                (Librarian commands)
                ON $RC > 4 GOTO B           Condition 3

 Condition 3 makes conditions 1 and 2 obsolete.
```

- At stack overflow (in spite of compression), message L110I is issued and the job terminated with RC=16.

Controlled Operator Cancel: Whenever the operator cancels a Librarian command or LNKEDT job normally, that is not with CANCEL-FORCE, the Librarian or Linkage Editor continues processing up to a point where a consistent state of the library is reached. The delayed cancel function ensures that the library in process cannot be destroyed.
CANCEL-FORCE, however, terminates always immediately.
The delayed cancel function is especially useful when working in SYSLOG-prompting mode: long-running commands may be cancelled and the Librarian will ask for entering the next command. The same function is supported also when running under POWER or, for Librarian, when running in an interactive (ICCF) partition.

The delayed cancel function is controlled by the flags IJBARCNA and IJBCNCPD in field JCSW8 of the partition communication region. These flags are set/reset/checked by the Level 3 services of the Librarian and by the LNKEDT phase.

The implementation for the Librarian Level 3 services is done in the following way:

The delayed cancel option is set in module INLPSYPA just before branching to the command execution phase, and it is reset in INLPSYPA just after control returns from the command execution phase. The command execution phase checks if there is any operator CANCEL pending by calling INLPMAIN (entry point INLPCACK). If there is a CANCEL pending INLPMAIN issues message L140I, indicates 'CANCEL pending' in the Librarian communication region INLCCOMR, resets flag IJBCNCPD, and returns to the command execution phase. If there is no CANCEL pending INLPMAIN indicates 'no CANCEL pending' in the Librarian communication region INLCCOMR and returns to the command execution phase.
After a consistent library state is reached the execution phase decides the terminating action depending on batch or SYSLOG processing mode.

Batch mode: The execution phase cancels the task by calling INLPMAIN (entrypoint INLPCANC). INLPMAIN resets the 'Delayed Cancel' option (flag IJBARCNA) and forces cancel by issuing macro TREADY with cancel code X'24' if the Librarian is not running under control of ICCF. Under control of ICCF the Librarian resets the 'Delayed Cancel' option, sets the 'Cancel pending' flag (IJBCNCPD) as cancel indication for ICCF, and goes into Wait State in order to wait for being canceled by ICCF.

SYSLOG mode: The execution phase returns to the parser (calling module INLPEXIT) in order to force prompting for the next command.

Migration: Migration phases with the names of the old Librarian phases MAINT (module INLPMIMA), CORGZ (module INLPMICO), CSERV (module INLPMICS), RSERV (module INLPMIRS), SSERV (module INLPMISS), PSERV (module INLPMIPS), DSERV (module INLPMIDS) are shipped preparing the online migration of these functions. The migration phases load phase LIBR into the partition and build a migration identification string which is passed via the PARM parameter interface to module INLPMAIN in phase LIBR. Module INLPMAIN recognizes the different migration identification strings and sets the corresponding migration flags in the Librarian communication region INLCCOMR. The parser module INLPSYNA checks the migration mode and uses the parser table INLTMIGR (used for old command syntax description) instead of INLTPARS (used for new command syntax description). After successful parsing the normal Librarian execution phases are loaded and processed, each of them containing the migration module INLPMIGR. Module INLPLEV3 which is processed at first in each command execution phase recognizes migration and gives the control to INLPMIGR.

INLPMIGR does the following:

1. Complete the parsed command block INLCCMDP by inserting the library and sublibrary names and transforming old functions to the corresponding new functional flags in INLCCMDP. The new library and sublibrary names are received either via the Library Migration Table (LMT) and the LIBDEF FROM/TO library names or by prompting the user for specification of 'library.sublibrary'.
2. Call the corresponding new Librarian command processing module for executing the required function.

Special Considerations

"Library full" Condition

Whenever the "library is full"-condition occurs while requesting one or more free library blocks from space management and the library space resides in VSAM managed SAM space, the Librarian tries to get an additional extent for the library.

There are two cases to be distinguished:

(1) If the library already owns 16 extents, no more extension is allowed and message L268 is given.

(2) Otherwise, VSAM tries to get a secondary extent for the library. If this extension is successful, the new extent is added to the VSAM catalog, appended to the other library extent(s) and initialized by writing its free space map (and formatting the library blocks when they reside on a CKD device). In this case, the "library is full"- condition disappears.
If the extension fails (e.g. VSAM space is insufficient for an extension) VSAM exits to the Librarian (module IJBLBBRN) which issues message L278.

If the "library is full"-condition remains because a library in VSAM managed space cannot be extended via secondary allocation or the full library resides in BAM managed space where no extension is possible, the subsequent action depends on whether the condition is raised while cataloging a member or while updating the member index of a sublibrary.

(a) The condition occurs while cataloging a member (INLMPUTR or INLMMDIS has been given):

The already written member space is freed and return code 12 (library is full) is given back to the requestor.

(b) The condition occurs while updating the member index (macro INLMSTOW has been given):

If the condition occurs while adding several members out of the stow table, the following iteration sequence is tried to add as many complete entries as possible to the directory.

1. If immediate reclamation can be done (because some members have been previously replaced within the same INLMSTOW request) new space can be made available and the update proceeds. (Return code 0 is given.)
2. If automatic reclamation can be forced (because some members have been replaced and the sublibrary is uniquely assigned to the requestor task) new space can be made available and the update proceeds. (Return code 0 is given.)
3. If the condition occurs while not all members of the stow table are added, the space of the last member in the stow table is immediately freed, the entry in the stow table is flagged as deleted and the update proceeds. (Return code 4 is given.)
4. If no more members of the stow table are to be processed, or the "library is full"-condition occurs while splitting a library block in a higher index level, the upper index levels which cannot be updated are freed until the index is in a consistent state. In this case, message L164 is given and return code 12 is passed back to the requestor.

(c) The condition occurs during the processing of the COPY/MOVE command (module INLPCOPY):

If the "library is full"-condition is indicated by return code 12 from the macros INLMPUTR or INLMMDIS and the stow table contains members which are already copied, but not yet reflected in the member index, then macro INLMSTOW is issued for an ADD request of these members. When INLMSTOW was suc-

cessfully completed the copy process is continued with that member in the stow table for which the "library is full" condition occurred. This process is repeated until there is no member left which could be copied completely. In this case message L201 is issued and the processing is terminated.

If the "library is full"-condition is indicated by return code 12 from the macro INLMSTOW, then message L201 is issued and the processing is terminated. If INLMSTOW returns with return code 4 and already copied members were deleted during the processing of INLMSTOW because of the occurrence of a "library is full" condition, then the copy process for these deleted members is repeated when MOVE is executed and at least one member in the stow table could be processed successfully by macro INLMSTOW. Otherwise, message L201 is issued and the processing is terminated.

(d) The condition occurs during the processing of the RESTORE library, sublibrary, or member command (module INLPREM2):

If the "library is full"-condition is indicated by return code 12 from the macros INLMPUTR or INLMMDIS and the stow table contains members which are already restored, but not yet reflected in the member index, then macro INLMSTOW is issued for an ADD request of these members. When INLMSTOW was successfully completed it is tested whether that member for which the "library is full" condition occurred is starting in the current tape buffer. If so, Restore continues until return code 12 occurs and the stow table is empty, or all members have been restored successfully. If not, message L201 is issued and the processing is terminated.

If the "library is full"-condition is indicated by return code 12 from macro INLMSTOW, or if INLMSTOW returns with return code 4 and already restored members were deleted during the processing of INLMSTOW because of the occurrence of a "library is full" condition, then message L201 is issued and the processing is terminated.

(e) The condition occurs during the processing of the RESTORE of a pre-Version 2 private library (module INLPROLD):

If the "library is full"-condition is indicated by return code 12 from the macros INLMPUTR or INLMMDIS and the stow table contains members which are already restored, but not yet reflected in the member index, then macro INLMSTOW is issued for an ADD request of these members. When INLMSTOW was completed message L201 is issued and the processing is terminated.

If the "library is full"-condition is indicated by return code 12 from macro INLMSTOW, or if INLMSTOW returns with return code 4 and already restored members were deleted during the processing of INLMSTOW because of the occurrence of a "library is full" condition, then message L201 is issued and the processing is terminated.

GETVIS Management

Librarian Level 1 and Level 2 services reside as reentrant code in the SVA. Therefore, for each requestor there must be a separate storage area to hold the variable part of the data.

Two types of data are distinguished:

(a) the storage which keeps the program variables which are internal to the called modules (i.e. the AUTOMATIC storage),

(b) the storage for the control blocks which can be kept for the execution of several modules (i.e. the global data).

Both types of storage requests are satisfied by GETVIS calls (SVC 61) with subpool identifiers. The requestor of Level 2 services can specify by a parameter in the LAMB whether the space is to be given from the partition from the system GETVIS area or dynamic GETVIS space. The dynamic GETVIS space will be used instead of system GETVIS for requests running in a dynamic partition.

The subpool ID for the AUTOMATIC storage of a module is contained in the LAMB and is defaulted to 'INLG' concatenated with the task identifier. To avoid a heavy GETVIS/FREEVIS overhead when using (nested) Level 2 services, the first request for the given subpool ID takes a fixed amount of GETVIS storage (currently 12K) and the Librarian manages further requests for this subpool by itself in a simple stack mechanism. If the stack becomes eventually empty, the space is given back by a FREEVIS request. The user of Level-2 services has furthermore the possibility to indicate in the LAMB that the space shall be kept for different service nestings. In this case, the user is responsible to free the space, if it is no longer needed.

The subpool ID for control blocks is contained in the LRPL and is defaulted to 'INLC' concatenated with the task ID.

GETVIS requests with subpooling have the advantage that if the system abnormally terminates the used GETVIS space can be released by freeing the corresponding subpools.

At end-of-subtask processing the SVA subpools INLC and INLG for the terminating subtask are freed by module INLPSEOT. At end-of-job processing (/&) the SVA subpools INLC and INLG are freed for the executing task by module IJBCTUPD if its used working space is not within these subpools (i.e. LAMBSVIS is off).

The space for the Second Level Directory (SLD) is taken from the system GETVIS area. The used subpool ID is stored in the LPT (field LPTPOLID) and is initialized to 'INLSLD'.

The special subpool name 'SPDUMP' is used by IJBEOTSK (normal or abnormal termination of a task) and IJBXLBIO (dump access to library).

Locking Mechanism

Locking ensures that other partitions/CPUs are unable to modify the Library Header or the sublibrary locked.

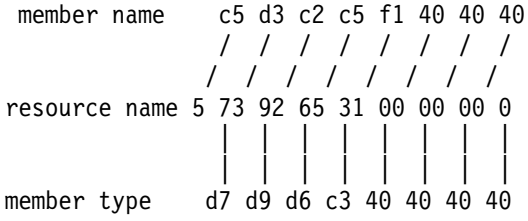
Locking Units: The following main locking units are used:

```
Library Header                          )
Free Space Inventory                    )   library space
Sublibrary (member index)               )
Library Control Tables (LCT)      )   tables in SVA
System Directory List (SDL)       )
Member data
The locking units are represented internally by resource names which are
12 characters long.
They are built as follows:

Library Header: 'C' || VOLID of first extent || start (disk) address
                                                of library


Free Space
Inventory:      'C' || VOLID of first extent || start (disk) address of
                                                first Free Space Map

Sublibrary:     'C' || VOLID of index level 1 || start (disk) address of
                                                 index level 1

LCT:            'C' || 'LCTABLES   '

SDL:            'C' || 'SYSDIRLIST '

Member:         'C' || member resource name   || start (disk) address of
                                                 the sublibrary
```

The length of the VOLID is 6 bytes and the length of the disk address 5 bytes (CCHHR for CKD and block number for FBA).

The resource names for the library header or sublibrary starting with 'L' has been removed with ESA130. In the locking hierarchy they stood below the LCT (see Order of Locking, below).

The start disk address contains the start track number truncated to 3 bytes from the left. The length of the member resource name is 8 character long. Therefore it cannot be unique. Member name and type are hashed into 8 character for shared libaries by the following method:



Locking Rules: The basic rules for locking are as follows:

A sublibrary is locked

- EXCLUSIVE only when its index is modified, that is when members are cataloged, deleted, updated, renamed
- SHARED when a read request requires a stable status of the sublibrary (e.g. Backup of sublibraries).

A library header is locked

- EXCLUSIVE when it is modified, i.e. when a sublibrary is created, deleted, renamed, updated or the Library Descriptor is updated
- SHARED when a read request requires a stable status of the library (for example, Backup of libraries).

A library / sublibrary is not locked when it is searched for a specific member, e.g. by DTFSL, FETCH, LNKEDT-INCLUDE function, execution of procedures. But the sublibrary is locked SHARED if a generic search request is done (INLPBLDL).

The Free Space Inventory is locked EXCLUSIVE when it is updated.

The kind of locking request is determined by the location of the library header of a library. Independent of whether single extents of a given library reside on CPU-shared DASDs or not:

- if the Library Header resides on a DASD which is shared between CPUs, locking is done across CPUs;
- if the Library Header resides on a DASD which is not shared between CPUs, locking is done only within the requesting CPU.

Member Locking:

Why Member Locking?

- The TP-server puts the librarian into an interactive environment.
- Via PWS support it's possible to hold a shadow of a member for a long time. There should be a possibility to protect members on the host for write access during this time.

These two cases leads to two different lockings:

- a supervisor lock for members.
- a long range lock realized by a new VIF in the directory entry of the member.

The Directory Lock for Members: This Lock can be controlled by LOCK/UNLOCK command or API LOCK/UNLOCK functions. In Level 2 it is done by module INLPCLK. It simply calls INLMSTOW to add or delete the lockvif for the specified member(s). The format of the lockvif:

```
DECLARE                        /* VIF                          */
  1 INLCLCKV BASED(*),         /*                              */
   2 LCKVELEN FIXED(15),       /* LENGTH OF V.I.F.             */
   2 LCKVEATR BIT(16),         /* RESERVED                     */
    3 LCKVVIF BIT(1),          /* VIF FOLLOWS                  */
   2 LCKVEID CHAR(4),          /* V.I.F. IDENTIFIER            */
   2 LCKVLID CHAR(8),          /* LOCKID                @D52RDUT*/
   2 * CHAR(16);               /* RESERVED              @D52RDUT*/
```

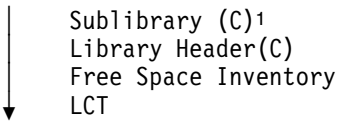
The Supervisor Lock for Members: The member data is locked exclusively with any call of INLPMCON with CONNECT=NEW or update. The member will be unlocked when disconnected (CONNECT=UPDATE) or with the stow function (CONNECT=NEW). Because locking and unlocking are done in two different Level2 Modules even in different API services the time frame in which the member is locked is outside the control of the Librarian. Therefore the Librarian never waits for this lock.

```
MCON_NEW
        lock member
PUTR
 ...


MDIS


STOW
MF(ADD|PURGE)
        lock sublibrary
        unlock member
        add member
        unlock sublibrary
```

```
MCON_UPDATE
        lock member
GETR

PUTR
....

MDIS
        unlock member
...
```

The locking/unlocking is done in the nodule INLPMLK. It computes the resource name and also controls setting and clearing of the LARGSLK bit. It is set when the member was locked with this INLCLARG and it remains cleared when the resource is already owned by the task.

Order of Locking: To avoid deadlocks, the following order for locking is defined:



1 see the conventions for the resource name

That means, for example, a sublibrary must not be locked if the Library Header is already locked.

Unique Access to a Library/Sublibrary: A second locking mechanism is implemented to ensure unique assignment of a single task during special functions like space reclamation, definition or deletion of a library/sublibrary. This has become necessary because the usual Supervisor locking support is not used for read accesses. The mechanism is based on flags in the LDT/SDT entries of the Library Control Tables and works only within one CPU. (For this reason, a library which resides on a shared DASD is never uniquely assigned.)

'Uniquely assigned' means:
The library or sublibrary is in use by one task only
and
does not reside on a volume which is shared across CPUs
and
the sublibrary does not show up on a LIBDEF-PROC statement if the job is executed from a procedure (for the "no access allowed" flag) or on any LIBDEF statement (for the "delete" flag).

- The "no access allowed" flag of a LDTE/SDTE is used whenever (AUTOMATIC) space reclamation takes place for a library/sublibrary. It is set and reset via macro LBRUPDAT. This macro also locks the library/sublibrary as long as the flag is set. If during that time an attempt is made to access that library/sublibrary (to establish a LOT entry in the LCTs), the flag indicates that the library/sublibrary is not accessible. The requesting task abends with message L160/L162, because waiting on the lock of the library/sublibrary may lead to a corruption of the library (module IJBCTUPD).
  The "no access allowed" flag is also used for optimizing I/Os for the Free Space Inventory and the library descriptor. When a COPY library, a RESTORE library, a RESTORE sublibrary/member, or a RESTORE OLDLIB is done and the target library is uniquely assigned, the free-space map and the library descriptor are kept in storage as long as the "no access flag" is set during allocation of member space and assignment of a unique library block identification. The update on the disk is done for each STOW function.
- The "delete" flag of a LDTE/SDTE is used (by module INLPSLIB via LBRUPDAT) whenever the corresponding library/sublibrary is deleted. An attempt to access the library/sublibrary results in the return information:
  - for defining a library/sublibrary (new access): library/sublibrary already exists
  - for accessing an existing library/sublibrary: library/sublibrary is deleted
- The "new" flag for the LDTE/SDTE is set for the period of time when a library/sublibrary is defined. It is used by module INLPCONL/INLPCONS and operates in a comparable manner as the "delete" flag: An attempt to access the library/sublibrary results in the return information:
  - for defining a library/sublibrary (new access): library/sublibrary already exists
  - for accessing an existing library/sublibrary: library/sublibrary does not exist

Space Reclamation

General Description: The process of freeing used library blocks and making them available for re-usage is called space reclamation. The information about the free space of a library is kept

- in the Free Space Map covering the entire library space
- in Space Reclamation Chains attached to sublibraries.

The Free Space Map indicates the library blocks available for usage, i.e. for cataloging members.
The Space Reclamation Chain of a sublibrary contains members whose directory entries are removed, but whose library blocks (index and data) are not added to the Free Space Map and therefore are not available for re-usage. Such a member can no longer be found in the member directory, but may still be pointed to by in-storage directories.

To allow space reclamation for libraries shared between CPUs, a bridge between CPUs is built using the following technique:
Each library block has a Library Block Identifier (LBID) field with an identification value in it. All library blocks belonging to the same library unit get the same unique value.

The library units are:

- Library Header and Free Space Inventory
- member index of a sublibrary
- data of a member

The Library Descriptor contains the highest LBID allocated in the library (LBID high-watermark).

A fixed LBID is allocated to the library blocks which contain the Library Descriptor, the Free Space Map (of all extents), and the Sublibrary Directory. This value is never changed even if the Sublibrary Directory is updated (which may result in an extension).

When a sublibrary is defined, it gets a LBID which is stored in the Sublibrary Directory entry and is allocated to all library blocks which contain the member index of this sublibrary. This LBID is unique to the sublibrary and not changed during the whole lifetime of the sublibrary.

When a member is created, it gets a LBID which is stored in the member directory entry and allocated to all library blocks containing the member data. Each time the member is updated it gets a new LBID (stored in the library blocks of the member data and in the member directory entry).

Such identifier fields allow the detection of a member change. Since they are stored in the library, the changes can be detected on all CPUs accessing the library.

This technique provides the possibility to reclaim the space of a deleted member immediately whether the sublibrary is in use by another task or not, eliminating the necessity for extensive locking and avoiding system integrity exposures.

However, reclaiming space always immediately under any circumstances may cause some side effects which may not always be acceptable. Therefore the sublibrary attributes AUTOMATIC and IMMEDIATE are introduced. They can be specified at sublibrary creation time.

When a member is deleted from a sublibrary, its library blocks are reclaimed by adding them to the Free Space Map. This may happen immediately as part of the member deletion process or delayed depending on the sharing status of the sublibrary and on the sublibrary attribute.

Figure 20 on page 50 describes the conditions of the different space reclamation algorithms.

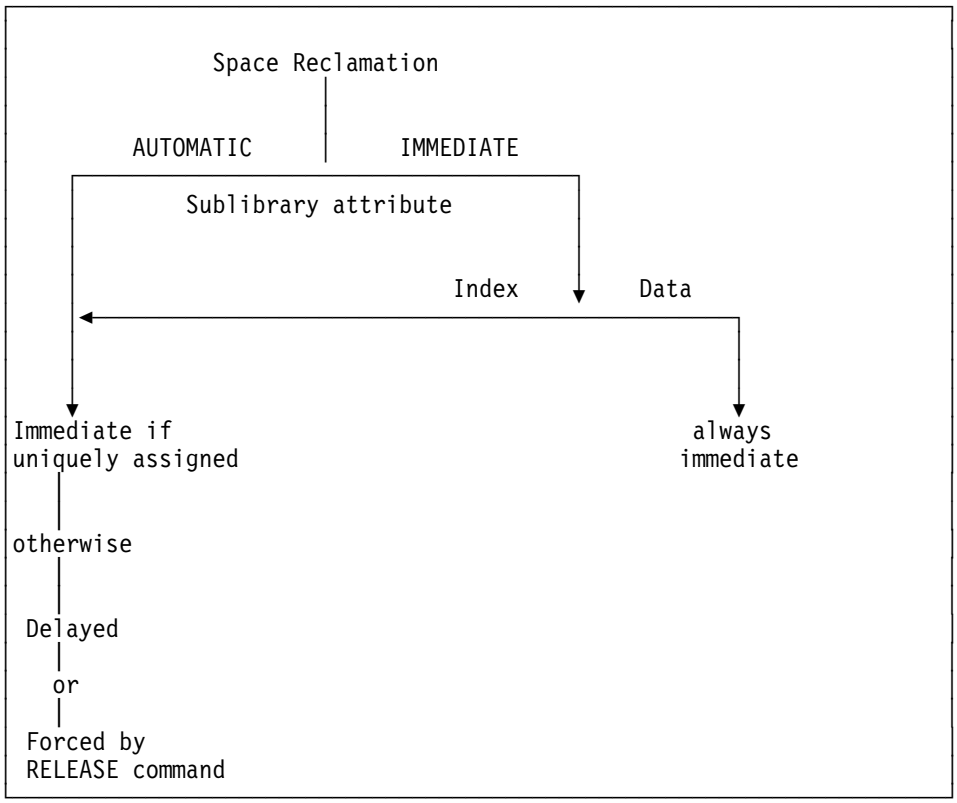


Figure 20. Space Reclamation Overview

Sublibraries with Attribute AUTOMATIC: *Immediate Space Reclamation:* For sublibraries with attribute AUTOMATIC, immediate space reclamation takes place for

- private sublibraries which are uniquely assigned and not shared across CPUs
- system sublibraries in BG when FG is inactive and not shared across CPUs.

Otherwise, the library blocks are added to the Space Reclamation Chain of the sublibrary for delayed reclamation, i.e they are not immediately available for re-usage.
Since the entry of the deleted member is removed from the member index, it cannot be found anymore. However, programs which already have the address of the member may continue reading the member data, i.e. those jobs or tasks are not cancelled.
This feature is valuable e.g. when long-running programs or procedures have to be updated or replaced without affecting the execution of the version in use.

Delayed Space Reclamation: The library blocks of a Space Reclamation Chain are made available for re-usage.

- When an update of a sublibrary is requested and conditions under "Immediate Space Reclamation" are fulfilled.
- For private libraries: whenever a private library is entered into or removed from the Library Control Tables and not shared across CPUs, the space of the Space Reclamation Chains of all sublibraries in this library is reclaimed.
- For private sublibraries: whenever a private sublibrary is entered into or removed from the Library Control Tables and not shared across CPUs, the space of the Space Reclamation Chain of this sublibrary is reclaimed. During end-of-task processing, space reclamation does not take place.
- For the system sublibrary during IPL.

Reclamation is done by module INLPRCLM via macro INLMRCLM.

According to the above rules, libraries (system as well as private) on CPU-shared DASDs are never considered uniquely assigned and therefore the space for deleted members is not reclaimed.

For DELETE (incl.MOVE-from), RENAME, RESTORE operations it is requested that the libraries or sublibraries are uniquely assigned and don't show up on any LIBDEF statement (not even of the own partition).

For delayed space reclamation the CPU time and SIOs may be accounted to a partition other than the partition which actually deleted the members.

Forced Space Reclamation: When the conditions for delayed space reclamation are not fulfilled for a long time and cannot be triggered without inconvenience, a user may force reclamation of library blocks in the Space Reclamation Chain of a sublibrary through the RELEASE SPACE command.
The RELEASE SPACE command frees library blocks with member and index data for sublibraries with attribute AUTOMATIC and library blocks with index data for sublibraries with attribute IMMEDIATE.
As long as such library blocks are not reused, the situation for member retrieval and index search is the same as before execution of the RELEASE SPACE command. (The reclaimed library blocks remain unchanged, i.e. member data or index data are not destroyed.) If a sublibrary is shared between CPUs and space reclamation has to be forced, it is recommended to issue the RELEASE SPACE command in all CPUs (which access the sublibrary) before new members are cataloged. This ensures that the tasks in all CPUs see a consistent member index tree if index blocks are affected through forced reclamation.
If a user does not submit the RELEASE job in all affected CPUs before cataloging new members, the SLDs in the other CPUs may be obsolete with the effect that FETCH may not find a phase or may take longer to find a phase.
When sharing IJSYSRS.SYSLIB across CPUs, a 'SET SDL' command should be given in all other CPU's if a CPU changed a SVA-eligible phase in IJSYSRS.SYSLIB. The 'SET SDL' command should be given before a RELEASE command for IJSYSRS.SYSLIB is executed. The reason is that SDL entries which contain pointers to phases in the system sublibrary should always point to members which have not already been reclaimed.
The RELEASE command does not destroy a library or cause integrity exposures.
A task is cancelled when (due to its in-storage information) it searches an index with reused library blocks. If data of deleted members are still being retrieved by other programs, such programs may be cancelled with message L135 (similar as with attribute IMMEDIATE).

If the library/sublibrary is protected by Access Control Facility: The user must have the ALTER access right for the library or sublibrary specified on the command.
For command RELEASE-LIB: An informational message (L144) is displayed on SYSLOG to document that space reclamation was forced. If the library resides on a CPU-shared DASD: the operator is told by a message (L047) to ask the operators of the other CPUs to execute the same job for the libraries / sublibraries released. This action should be done on all CPUs before any new members are cataloged.

For command RELEASE-SUBLIB: Appropriate messages (L145, L048) are given only if there is space actually released.

Sublibraries with Attribute IMMEDIATE: For sublibraries with attribute IMMEDIATE, the data library blocks of a deleted member are marked as free in the Free Space Map always immediately, i.e. as part of the member deletion process, independent whether the sublibrary is uniquely assigned or not.
Since the library blocks get another identification (LBID) when they are reused, their re-usage is detectable (even across CPUs) and the retrieval process is terminated by the Librarian routines.
Index library blocks are always reclaimed according to the rules of AUTOMATIC space reclamation in order not to destroy the basic assumptions of the index-update algorithm. Therefore a sublibrary with attribute IMMEDIATE may contain 'delayed' Library Blocks (containing index information).
Since library blocks have to be read in main storage before their identification can be checked, it may happen that a partition dump shows data of a member which the respective user is not authorized to read. To avoid such a situation, members with sensitive data should be stored in sublibraries of attribute AUTOMATIC.
There is one situation in which the Librarian routines cannot check the library block identifications: when a program uses DTFSL for member retrieval and manages the NOTE information itself and does not provide a NOTE length of 32 bytes (or higher) via the new parameter NLEN. Such programs may get inconsistent member data. To avoid this, it should be ensured (in an administrative way) that the sublibraries accessed by such programs have the attribute AUTOMATIC.

Notes for Implementation of the RELEASE Command: Actions to be done during RELEASE

For all sublibraries (incl.IJSYSRS.SYSLIB):

- Add library blocks of Space Reclamation Chain to Free Space Map.
- Update SLD
- Update Highest Index PRBA

For sublibraries on CPU-shared DASDs: These actions must be performed also when the Space Reclamation Chain is empty. In this case neither the ALTER access right is required nor is the operator-action message given. (Reason: Reclamation may have taken place in another CPU and the RELEASE command has just the task to update a SLD.)

Backup Tape Layout

The tape layout depends on the options specified. Three or four files, separated by a tape mark, will be created with a single BACKUP command depending on the specification of the RESTORE=STANDALONE parameter.

The tape will not be re-positioned when the backup function starts, except for a stand-alone version. In this case, the tape will be positioned to the load point before the backup operation is started. Thus, the stand-alone backup version can only be the first backup version on a "stacked" backup tape.

```
stand-alone backup version                    online  backup version
_________________________________________  ____________________________________

1. file: "Stand-alone IPL file":               1. file: header file or empty file
          - [header]
          - IPL bootstrap phases
          - Console support phases
          - VSE/ESA Supervisor
          - Further IPL phases
            (See module INLPBKUP for a complete
             list of the phases in this file)
2. file: "Stand-alone Utilities file":
          All phases needed in the stand-alone
          environment in the SVA including the
          - Customization phase (IJWCUST)and
          - the stand-alone utility phases for:
            FCOPY,RESTORE,ICKDSF and DITTO
          (See stand-alone SVA loadbook $SVASA)
3. file: backup-file-ID record                 2. file: backup-file-ID record
         [system history file]                        [system history file]
4. file: backup file, i.e.                     3. file: backup file, i.e.
         libraries/sublibraries/                        libraries/sublibraries/
         members                                        members
```

At completion of a BACKUP command a tapemark is written to complete the Backup file. Then another tape mark, an End-of-Backup record, and another 2 tape marks are written. Then the tape is re-positioned in front of the last 3 tape marks. Thus, another online backup version will overwrite these 3 tape marks and the End-of-Backup record.

For a detailed description of the format of labeled and unlabeled backup tapes see Module INLPBKUP. The phases in the "standalone" files except for the IPL bootstrap phases are preceded by a tape block header and a phase record header (see DSECT's INLCSABL and INLCSAPH).

The Backup File: (For the logical description of the backup file and the tape format see also description of module INLPBKUP.)

Each tape block in the backup file starts with a 16-byte block header (see data area BLKHDR) which contains the length of the tape block, the number of the tape block and a 4-byte descriptor.

Each item in a tape block has the same format:

The first 2 bytes contains the length of tape item followed by a 4-byte descriptor and the data.

The following is a list of the possible descriptors of tape items in a tape block and the meaning of the tape item:

LIBR	Backup-file-header (see data area BFHDR). The backup file contains libraries.
SUBL	Backup-file-header (see data area BFHDR). The backup file contains only sublibraries.
MEMB	Backup-file-header (see data area BFHDR). The backup file contains only members.
LHDR	Library header (see data area LIBRHDR). Contains the file name, the file-id and information from the library descriptor record.
SHDR	Sublibrary header (see data area SUBLHDR). Contains the file name of the library, the sublibrary name and information from the sublibary index entry.
XTNT	Extent-record (see data area EXTENTR). Contains information about the original extents.
IPLK	CKD Disk IPL phase (see data area DIPLHDR).
IPLF	FBA Disk IPL phase (see data area DIPLHDR).
MHDR	Member header (see data area MEMBHDR). Contains the library,sublibrary and member name and the member type and information from the member directory record.
MDAT	Member data (see data area MEMBDAT). Contains the data of a member. The data of members of types OBJ,PROC,Source or any user type are contained in the tape block in compressed form as in the library on disk.
DHDR	Dummy member header. Indicates the end of a sublibrary.
EOBF	End-of-backup-file record (see data area EOBKFID). Indicates the end of the backup file. The tape block containing this record is followed by the dummy tape blocks which are necessary to allow multiple tape buffering.

PID-V2-STACKED Tape: To avoid the need for separate DLIB format tapes for Version 2 programs and VSE/SP product tapes a PID-V2-STACKED tape format has been introduced for use as VSE/SP distribution tapes only.

PID copies a "Start of PID-V2-STACKED tape" indicator to the front of the tape to indicate to the VSE/SP installation process that this is a PID-V2-STACKED tape. This allows the individual VSE/SP Optional Program tapes and the DLIB format tapes for Version 2 program products to be identical.

The tape stacking process is as follows:

1. Copy "Start of PID-V2-STACKED tape" indicator to stacked tape.
2. Copy Optional Program(s) to stacked tape.
3. Copy "End of PID-V2-STACKED tape" indicator to stacked tape.

This process allows the shipment of one copy of the individual product tapes to PID to be used either as the DLIB format program product tape or as the VSE/SP 2.1 Optional Program tape.

Format of the PID-V2-STACKED Tape:

```
       - START OF PID-V2-STACKED TAPE INDICATOR (3 files)
                                  (see detailed layout below)

FILE 4 - COPYRIGHT INFO FOR FIRST PRODUCT
       --- tape mark ---
FILE 5 - BACKUP-FILE-ID RECORD and
         MSHP HISTORY FOR FIRST PRODUCT
       --- tape mark ---
FILE 6 - PRODUCT SUBLIBRARY FOR FIRST PRODUCT
       --- tape mark ---
FILE 7 - NULL FILE
       --- tape mark ---
FILE 8 - END OF BACKUP RECORD FOR FIRST PRODUCT
       --- tape mark ---
FILE 9 - NULL FILE
       --- tape mark ---
    ...
    ...
    ...  Files 4-9 are repeated for each product to be stacked
    ...
    ...
       - END OF PID-V2-STACKED TAPE INDICATOR (3 files)
                                  (see detailed layout below)
```

Notes:

1. If the number of products ordered requires more than one stacked tape then the "Start of PID-V2-STACKED Tape" and "End of PID-V2-STACKED Tape" indicators are present on each and every stacked tape.
2. The stacking process also allows for products which contain more than one sublibrary backup file on the product tape. (i.e. the VSE/SP Optional Program tape consists of 9 or 12 files.)
3. If the number of products ordered requires more than one stacked tape then products do not span volumes on PID-V2-STACKED tapes.
4. PID-V2-STACKED tapes are always "unlabeled" and contains only unlabeled backup files.

Format of "Start of PID-V2-STACKED tape" indicator and "End of PID-V2-STACKED tape" indicator for the VSE/SP 2.1 Stacked Optional Program Tape:

The format of the "Start of PID-V2-STACKED tape" and "End of PIV-V2-STACKED tape" indicator required for the stacked tape containing VSE/SP Optional Programs is as follows:

1. tape mark
2. 80 byte record
3. tape mark
4. tape mark

Both the "start of tape indicator" and the "end of tape indicator" have the same format.

1. Layout of the 80 byte record for "start of stacked tape" indicator:

```
0050000000000001START.OF.STACKED.TAPE.FOR.VSE/SP.ONLY_00000000000000000000000000
<--hex data----><------character string-------------><-----------hex data---->

DETAILED LAYOUT OF 80 BYTE 'START OF STACKED TAPE' RECORD

Bytes 00-01 - 2 byte length of ID  x'0050'

Bytes 02-03 - 2 bytes reserved     x'0000'

Bytes 04-07 - 4 byte block number  x'00000001'

Bytes 08-45 - 38 bytes of characters - c'START.OF.STACKED.TAPE.FOR.VSE/SP.ONLY_'

Bytes 46-79 - 34 bytes of hex zeros    x'00000000...'
```

2. Layout of the 80 byte record for "end of stacked tape" indicator:

```
0050000000000001END.OF.STACKED.TAPE.FOR.VSE/SP.ONLY___00000000000000000000000000
<--hex data----><------character string-------------><-----------hex data----->


DETAILED LAYOUT OF 80 BYTE 'END OF STACKED TAPE' RECORD

Bytes 00-01 - 2 byte length of ID  x'0050'

Bytes 02-03 - 2 bytes reserved     x'0000'

Bytes 04-07 - 4 byte block number  x'00000001'

Bytes 08-45 - 38 bytes of characters - c'END.OF.STACKED.TAPE.FOR.VSE/SP.ONLY___'

Bytes 46-79 - 34 bytes of hex zeros    x'00000000...'
```

Note that the format of the two indicators is the same. The only difference is the text of the 38 byte character string.

Processing of PID-V2-STACKED Tapes by Restore Command: Restore gets in two ways the information that the input tape is a VSE/SP distribution tape (i.e. is a PID-V2-STACKED tape):

1. When Restore recognizes the "Start of PID-V2-STACKED tape" indicator it turns on bit PIDV2SW in Backup/Restore communication region INLCBRCR. If this bit is on Restore scans the input tape for the specified backup file-ID until the "End of PID-V2-STACKED tape" indicator is found.
2. MSHP indicates to Librarian in the interface control block INLCPARB (bit PARMPDV2) that the input tape is a PID-V2-STACKED tape. Librarian module INLPMAIN then turns on bit CRGPIDV2 in librarian communication region INLCCOMR which is recognized by Restore. In this case Restore scans the input tape starting at the current position of the tape until the specified backup file-ID or the "End of PID-V2-STACKED tape" indicator is found (i.e. Restore is able to locate the specified backup file-ID if the product specified by this backup file-ID is located on the input tape behind the current position of the tape).

This means that a specific product on a PID-V2-STACKED tape may be restored if

- restore is called by the VSE/SP installation process or
- the tape is positioned at the 'load point' or
- the tape is positioned directly before the specified backup file-ID of the product sublibrary to be restored.

Tape Buffer Management

For the BACKUP and RESTORE commands a special buffering technique for the tape I/O is used. This is done to achieve at least partial streaming for the 8809 tape device without causing performance degradation for other tape devices. Thus the tape output routine and the tape input routine have the following principal concepts:

Multiple buffers are used and the tape operations are scheduled as soon as possible and the WAIT is issued as late as possible (see modules INLATI and INLATO for the description of the technique).

Depending on the partition size available a maximum of 8 tape buffers are allocated. The space for a minimum of 3 tape buffers is required. The size of each buffer is 16632 bytes. For 3480 tapes only 2 buffers are used because these tapes have addtionally an internal buffer mechanisme.

For each tape buffer allocated a buffer definition block (BDB) is initialized (see data area BDB). The BDB contains the buffer address and pointers to the associated CCB and CCWs and a pointer to the next BDB. For each buffer allocated there exists a separate CCB.

The BDB for which the last buffer is allocated points to the first BDB. So the BDBs for which a buffer is allocated are chained in a loop as shown by Figure 21 for a BDB chain with 4 BDBs.

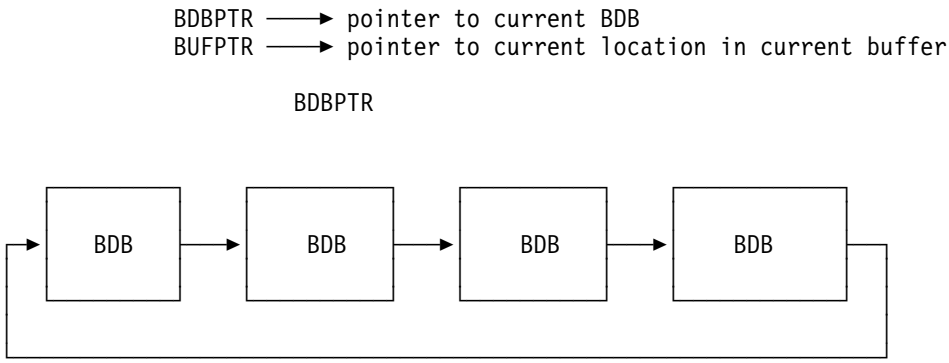


Figure 21. Example of a Buffer Definition Block List

The pointers BDBPTR and BUFPTR are located in the Backup/Restore Communication Region (see data area INLCBRCR) in module INLPRABF of root phase LIBR. The identification of the Backup/Restore Communication Region 'BRCR' is followed by BDBPTR and BUFPTR. BUFPTR points to the item in the backup file currently in process.

Because of this buffering technique Backup writes at the end of the backup file 8 dummy blocks.

Librarian/MSHP Interface for History File Records on Tape: To allow Tape Managers to support Librarian Backup/Restore only one OPEN and only one CLOSE must be done for the tape during one Backup or Restore run for unlabeled or labeled tapes. Therefore, when the backup of the history file together with some libraries or sublibraries is requested, the Librarian backup module INLPBKLB (or INLPBKSL) passes the address of an exit routine (INLPBKHF) to the MSHP module PTFBKUP. MSHP passes the history file records to this exit routine which writes these records to the output tape.
Similarly, when the restore function is called by MSHP, the Librarian restore module INLPREST passes the address of an exit routine (INLPRSHF) to MSHP. When MSHP then wants to read a history file record MSHP passes control to this exit routine which reads the record from the input tape.
Additionally, as during MSHP INSTALL the librarian restore function is called twice for the same backup file (once with SCAN=YES and once with SCAN=NO), MSHP provides in the call interface block INLCPARB the address of an area to be used by the Librarian Restore function. (Field PARMUSER in INLCPARB). The Librarian sets up in this area the DTF to be used for tape input; i.e. to guarantee that only one OPEN and one CLOSE is done per restore run.

Dependencies

This section describes dependencies which exist between the Librarian and other VSE components or, via global data, between different Librarian modules.

IPL

- The Library Control Tables (LCTs) are allocated and initialized during IPL time. For this purpose, phase $INITCON (module IJBLBIPL) is loaded and called twice:
  1. Call from phase $IPLRT6 to calculate the necessary system GETVIS space for the LCTs and return the space requirement,
  2. Call from phase $IPLRT7 to request the system GETVIS space and initialize the LCTs.
- The size of the LCTs is dependent on the input parameter IPLSDTE which gives the maximum number of sublibrary definitions in the Sublibrary Definition Table (SDT). The space requirement is returned in output parameter ICTABLEN.
- The System Directory List (SDL) is built and initialized and the phases given in the SDL are loaded into the System Virtual Area (SVA) at IPL time. IPL phase $IPLRT7 puts directory information of phase names given by appropriate load lists into the SDL. Then phase INLPSDL is loaded into the background (BG) partition which puts the corresponding system phases into the SVA by means of the Supervisor service SLOAD.

ICCF

- Interactive ICCF partitions are run by VSE tasks. If the ROLLOUT function suspends an ICCF user from work until shorter-running commands of other interactive partitions have been processed, there is no guarantee that the same VSE task is used to reinstate the previous command. However, Librarian functions rely on the assumption that the same VSE task is used for a sequence of Librarian services (that is the task ID is used to determine the accessed library within the LCTs, for locking purposes, for GETVIS subpooling, etc.). In order to avoid the ROLLOUT function of ICCF, a monitor call (MC 10,4) is given after each LBRACCES macro invocation to signal Librarian processing to ICCF which, then, suspends the ROLLOUT function.
- The Level 3 delayed cancel function (module INLPMAIN) checks if LIBR is running under control of ICCF. The macro 'GETFLD FIELD=ICCFPP' is issued and the result is contained in register 1. The cancel indication for ICCF is given via the partition COMREG flags IJBARCNA=OFF and IJBCNCPD=ON in JCSW8.

- The check for ICCF control is also done in the message handler (module INLPDIAG) in order to avoid additional message writing on SYSLST if LIBR is running in SYSLOG mode under control of ICCF.

Supervisor

- At abnormal termination of a VSE task it can happen that some entries in the LCTs have not been removed, or that the LCTs are in an inconsistent state (for example partially built entries). Therefore, the Librarian provides a clean-up service, which is invoked at task termination. To avoid unnecessary calls of this service, the clean-up (module INLPSEOT, phase LIBRSEOT) is only performed if the Librarian has requested it from the Supervisor task terminator. The signal is given by module IJBLBHLS (macro LBRACCES) via the instruction 'MODFLD FIELD=LIBRSERV,NEWVAL=(1)', where register 1 contains the value 1. The indication is reset by the task terminator when INLPSEOT is invoked.
- The Supervisor FETCH/LOAD service uses the Second Level Directory (SLD) when searching for phases in a given LIBDEF search chain to avoid an index search in the sublibraries. The SLD for the system sublibrary IJSYSRS.SYSLIB is built at IPL-time by phase $INITCON (via macro INLMSLD) in the system GETVIS area. SLDs for private sublibraries are installed for each sublibrary of a LIBDEF PHASE job control statement by module IJBCTUPD via INLMSLD. The SLD is rebuilt every time an update is done to the corresponding sublibrary (a new system GETVIS area is taken and the old one is freed) by module INLPSTOW via INLPSLD. If FETCH finds a SLD in a back-level state (for example, a SLD for a sublibrary on a shared DASD which has been rebuilt only within the other CPU) it uses the Librarian macro INLMSLD to rebuild the SLD. A SLD is dropped (that is the GETVIS space is freed) if the corresponding sublibrary is removed from the SDT (module IJBCTUPD).
- The Librarian I/O to libraries is done by physical addressing. No logical unit is needed. To prevent an unintentional removal of a volume on which an accessed library resides, the device is signalled as 'in-use' (via the Supervisor MSAT macro) for each extent which is entered in the EDT. The corresponding 'device release' function is performed whenever an extent is removed from the EDT (module IJBCTUPD). A logical unit is taken for each extent in order to do an OPEN for it (module IJBLBBRN). This internal system logical unit is unassigned as soon as the MSAT macro for 'device use' has been given (module IJBCTUPD). The logical unit is communicated between modules IJBLBBRN and IJBCTUPD in the EDT entry (field EDTEPUBX). MSAT changes the logical unit to the PUB index. A flag indicates whether the contents of the field is a logical unit or a PUB index.
- At end-of-job processing (/&) a VIO(CLOSE) is given for the partition and the VIO pointer for this partition is cleared (module IJBCTUPD).
- Space reclamation is performed immediately whenever the sublibrary is uniquely assigned to the requesting task. Otherwise, space reclamation is delayed until the point when the sublibrary will become uniquely assigned (Exception: If the sublibrary owns the space reclamation attribute IMMEDIATE the member data space is reclaimed always immediately, or if the RELEASE SPACE command is given all space is reclaimed immediately). The sublibrary is uniquely assigned if it does not reside on a CPU shared DASD and is entered to or removed from the SDT. Therefore, delayed reclamation takes place when the library/sublibrary is entered to or removed from the LDT/SDT (module IJBCTUPD via INLMRCLM macro). Reclamation is suspended at end-of-task processing. For the system sublibrary IJSYSRS.SYSLIB, delayed reclamation takes place at IPL time (module IJBLBIPL via macro INLMRCLM).

  Since read operations do not lock the member space, it is necessary to signal Supervisor FETCH when the space for a phase is freed although the sublibrary is not uniquely assigned (IMMEDIATE reclamation). There are two indications for FETCH to recognize that a phase is deleted:
  (1) the last library block of the phase which contains TXT data has set the flag LBCFMDEL (done by modules INLPSTOW, INLPSLIB), (2) if the space has been reused, the LB identifier (LBCFIDEN) is different from that given in the member descriptor.
  If a phase is chained in the reclamation chain, the flag LBCFIDEL is set to indicate the intention to set the flag LBCFMDEL when the reclamation takes place.

- The phase LIBR (modules INLPMAIN, INLPSYPA) uses the partition COMREG flags IJBARCNA and IJBCNCPD in JCSW8 as interface for the delayed cancel function.
- To allow cataloging of /& and /* when the input is read from SYSIPT, the flag 'Allow-/&' (pos.8) in the INSIZE field of the partition COMREG is set via SVC 13 (modules INLPCAT, INLPUPD). The flag is reset via SVC 12 (modules INLPCAT, INLPUPD, INLPMAIN).
- To recognize whether 80 or 81 byte records are read from SYSIPT the flag 'TEMP 81' (pos.4) in the INSIZE field of the partition COMREG is checked (module INLPMAIN).
- To recognize whether LIBR is executed in SYSLOG mode or in batch mode the flag JCINRDR (pos.3) in the JCSW1 field of the partition COMREG is checked (module INLPMAIN).

Job Control

- If a LIBDEF PHASE is given which contains the SDL in its SEARCH chain, module IJBCTUPD sets the flag JCSW4(6) ("Test mode") in the partition communication region (COMREG) to signal FETCH the changed search sequence: SDL is not searched in the first place but in the place which is indicated by the LIBDEF statement ("Test mode"). The flag is reset by module IJBCTUPD when the corresponding LIBDROP statement is executed.

MSHP

- The phase LIBR uses the macros MSHOPEN (module INLPREAD), MSHCLOSE and MSHEOJ (both in module INLPMAIN), and MSHTAPE (modules INLPROLD and INLPREST).

BAM

- The phase LIBR (modules INLPMAIN, INLPAREA, INLPREAD, INLPPUN, INLPWOR, INLPWTO, INLPWTP) uses the macros DTFCP, DTFCN, OPEN, CLOSE, GET, PUT. The phases LIBRBACK and LIBRRES1 (modules INLPTO, INLPTI) use the macros DTFMT, OPEN, CLOSE. Furthermore they use the B-transient $$BCEOV1 to force the end-of-volume handling for tapes. Flags in the DTFMT area are set in order to control tape moving operations.

Librarian

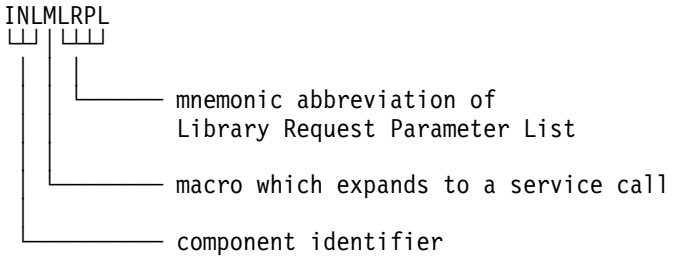
- To force the Librarian internal GETVIS optimization, flag LAMBGVIN in INLCLAMB is set by the phase LIBR (module INLPMAIN). When terminating the LIBR session the GETVIS space is freed by invoking macro INLMFREE (module INLPMAIN).

Design Information

Naming Conventions

All names for modules and macros which are new in VSE/AF Version 5 Release 2 adhere to the following naming conventions: they start with the component identifier INL; the following character denotes the type of the referred object. The last four characters are a mnemonic abbreviation of the object's intended function.

Example:



The type character is used as follows:

```
P ... for modules which are used in the on-line environment and/or
      in the stand-alone environment,
X ... for modules which are used in the stand-alone environment,
A ... for macros which are able to generate P modules,
M ... for macros which expand to a service call,
C ... for macros which describe control blocks,
T ... for tables (mainly for the command parser),
```

The function code used with a M-type macro owns the type character 'D'.

Related P-, M-, C-, D-type objects have the same mnemonic abbreviation. Moreover, the mnemonic abbreviation used for a structure (C-type object) has been mostly used as a prefix for the substructure names.

Modules and macros which had been taken over from releases prior to VSE/AF Version 5 Release 2 kept their names (component identifier IJB for the modules, the macro names start with LBR with few exceptions).

Link Books

The following link books are used:

```
INLPLNK1   -   for all command processing phases
INLPLNK2   -   for migration phases
INLPLNK3   -   for Transaction Server phase
INLXLNK1   -   for the stand-alone RESTORE phase
IJBLBLNK   -   for Level-1 and Level-2 services (phase $IJBLBR)
IJBLBIPL   -   for IPL phase $INITCON
INLPSDL    -   for phase INLPSDL (initialization and update of SVA)
INLPLNKB   -   for BACKUP migration
INLPLNKR   -   for RESTORE migration
INLPLNKC   -   for COPYSERV migration
INLPLNKP   -   for PDZAP migration
```

Linkage Conventions and Function Codes

The following calling conventions between Librarian modules are observed:

Register 1 is loaded with the address of the module-specific parameter list,

Register 13 contains the address of a caller provided save area,

Register 14 contains the return address,

Register 15 is loaded with the entry point address.

When Level 2 modules are invoked via the macro interface, Register 15 generally contains the address of phase $IJBLBR and Register 0 additionally provides a function code in order to route control to the requested entry point.

The function codes are constants with the following values:

```
LBRFCOPN  X'00'  (0) .... for module INLPOPEN,
LBRFCBDL  X'04'  (4) .... for module INLPBLDL (used by macro INLMBLDL),
LBRFCFND  X'08'  (8) .... for module INLPFIND (used by macro INLMFIND),
LBRFCGET  X'0C' (12) .... for entry  INLPGETR (used by macro INLMGETR),
LBRFCNTE  X'10' (16) .... for module INLPNOTE (used by macro INLMNOTE),
LBRFCPNT  X'14' (20) .... for module INLPPOIN (used by macro INLMPOIN),
INLDSLD   X'18' (24) .... for module INLPSLD  (used by macro INLMSLD),
LBRFCGLP  X'1C' (28) .... for module IJBLBLLS (used by macro LBRLCTAC),
LBRFCACC  X'20' (32) .... for module IJBLBHLS (used by macro LBRACCES),
LBRFCUPD  X'24' (36) .... for module IJBLBULT (used by macro LBRUPDAT),
LBRFCBRN  X'28' (40) .... for module IJBLBBRN (used by macro LBRBLDRN),
LBRFCACL  X'2C' (44) .... for module IJBCTUPD (used by macro LBRCTUPD),
INLDLCON  X'30' (48) .... for module INLPLCON (used by macro INLMLCON),
INLDSCON  X'34' (52) .... for module INLPSCON (used by macro INLMSCON),
INLDMCON  X'38' (56) .... for module INLPMCON (used by macro INLMMCON),
INLDMDIS  X'3C' (60) .... for module INLPMDIS (used by macro INLMMDIS),
INLDSDIS  X'40' (64) .... for module INLPSDIS (used by macro INLMSDIS),
INLDLDIS  X'44' (68) .... for module INLPLDIS (used by macro INLMLDIS),
INLDPUTR  X'48' (72) .... for module INLPPUTR (used by macro INLMPUTR),
INLDSTOW  X'4C' (76) .... for module INLPSTOW (used by macro INLMSTOW),
INLDGDIR  X'50' (80) .... for module INLPGDIR (used by macro INLMGDIR),
INLDDOIO  X'54' (84) .... for module INLPDOIO,
INLDRCLM  X'58' (88) .... for module INLPRCLM (used by macro INLMRCLM),
INLDLSIM  X'5C' (92) .... for module INLPLSIM (used by macro INLMLSIM),
INLDDIAG  X'60' (96) .... for module INLPDIAG (used by macro INLMDIAG),
INLDPIDT  X'64' (100).... for module INLPPIDT (used by macro INLMPIDT),
INLDRESN  X'68' (104).... for module INLPRESN (used by macro INLMRESN),
INLDIALC  X'6C' (108).... for module INLPIALC (used by macro LIBR)    ,
INLDLDS   X'70' (112).... for module INLPLDS  (used by macro LBRACCES),
INLDCLK   X'74' (116).... for module INLPCLK  (used by macro INLMLOCK),
INLDFRSP  X'78' (120).... for module INLPFRSP (used by macro LIBRM FRAME).
```

Return Codes and Message Handling of Level 1 and Level 2 Services

The internal return codes provided by Librarian Level 1 and 2 modules adhere to the following conventions:

```
 0 ............. requested function successfully completed,

 4, 8, 12 ...... function-specific exceptional conditions,

16 ............. external error (e.g. system resources exhausted,
                 invalid parameter list when called),

20 ............. internal error (e.g. processing consistency check
                 failed),

32 ............. security violation,

36 ............. library chaining error (library block identification
                 number changes within a chain).
```

Additional to return codes 16 and 20, a feedback code is returned in the corresponding parameter list (usually in the LRPL) denoting the reason for the failure (see Chapter 5.1 for the list of feedback codes).

Return codes which are equal to or greater than 16 are accompanied by a message which is returned in the message areas for SYSLST and SYSLOG (if specified in the LAMB) and printed or logged according to the message routing indicator if the corresponding DTFs are provided via the LAMB. If neither DTFs nor message areas are specified in the LAMB only the message number is returned in the LAMB.

Generation of common "Stand-Alone" and "Online" (P) Modules

As described under naming conventions, "A" is the type character for macros which are able to generate P modules.

These macros are introduced to have code which is common to "stand-alone" and "online" RESTORE only once in the system.

In the Versions preceding to Librarian Version 6 Release 1 the code was generated for the online environment and for the stand-alone environment (P- and X-modules) depending on the contents of the macro variable %ENV.

In the Librarian Version 6 Release 1 the macro variable %ENV is not used anymore. Now only common P-modules for the online environment and for the stand-alone environment are generated.

The stand-alone environment is indicated by the flag SYSCOM.IJBFLG08.IJBSAENV set by IPL.

The invocation of some services which are only used in the "online"-environment is skipped by the P-modules dependent on flag SYSCOM.IJBFLG08.IJBSAENV.

Some other services are ignored in the stand-alone environment (for example SVC 75 (SECTVAL) or SVC 110 (LOCK/UNLOCK)) or the code is not executed at all in the stand-alone environment (as for example the services INLMFIND,INLMBLDL,INLPNOTE or INLPNOTE (see module INLXSAR1)).

"Stand-alone" SVA Load Book

The following SVA Load book is used in the stand-alone environment:

```
$SVASA     -   This phase contains the names of all phases
               to be loaded in the stand-alone environment
               into the SVA.
               (This phase is used by Stand-Alone IPL and
                module INLPBKSA.)
```

Modules

IJBCTUPD

```
MODULE IJBCTUPD UPDATES LIBRARY CONTROL TABLES
DESCRIPTIVE NAME: ADD/CLEAR LIBRARIES IN LIB. CONTROL TABLES

COPYRIGHT =       SEE ABOVE

STATUS =          VSE/AF VERSION 5 RELEASE 2

FUNCTION =        ADD / CLEAR OF LIBRARY AND SUBLIBRARY
                  DEFINITIONS IN LIBRARY CONTROL TABLES
                  (LOT, LDT, SDT, EDT, DDT).
                  TRIGGERS RECLAMATION OF DELAYED SPACE
                  OF SUBLIBRARIES. BUILD  SECOND LEVEL
                  DIRECTORY FOR PHASES.
          1. MF=ADD (AT LABEL IJBADD):
          THIS ROUTINE IS CALLED IF A LIBRARY/SUBLIBRARY
          REFERENCE SHALL BE ADDED TO THE LIBRARY CONTROL
          TABLES.
          - THE CORRESPONDING LOT-ENTRY IS BUILT, AN EXISTENT
            REFERENCE AT THIS PLACE IS OVERWRITTEN.
          - MISSING LDT-/SDT-/EDT-/DDT-ENTRIES ARE INSERTED.
          - DURING THE WHOLE INSERTION PROCESS THE LIBRARY
            CONTROL TABLES (LCT) ARE KEPT LOCKED.
          - THE POINTERS TO THE CORRESPONDING ENTRIES
            (LIBINFO: LOT-ENTRY, LDT-C-ENTRY, LDT-P-ENTRY,
            SDT-ENTRY) ARE PASSED BACK TO THE INVOKER.
          2. MF=CLEAR (AT LABEL CLNORML):
          THIS ROUTINE IS PROCESSED AT A LBRCTUPD MF=DELETE
          REQUEST FOR A SPECIFIC CHAIN.
          - FOR LIBTYPE=ALL, THE LOT-CHAINS FOR TYPES PHASE,
            OBJ, SOURCE, AND PROC ARE REMOVED (E.G. LIBDROP).
          - OTHERWISE, THE ADDRESSED CHAIN OF THE GIVEN TYPE
            IS REMOVED.
          3. MF=CLEAR,SET=EOSTEP (AT LABEL CLEOSTEP):
          THIS ROUTINE IS CALLED AT SVC 14 (TASK TERMINATION).
          THE CHAINS FOR TYPES PHASE, OBJ, SOURCE, PROC, AND
          DUMP ARE CHECKED FOR PARTIAL ENTRIES (CAUSED E.G.
          BY CANCELATION OF A LIBDEF STMT) AND THE TABLES
          CLEANED-UP.
          ALL LOT-ENTRIES OF TYPE LBR FOR THIS PARTITION ARE
          REMOVED.
          4. MF=CLEAR,SET=EOJOB  (AT LABEL CLEOJOB):
          THIS ROUTINE IS CALLED AT END-OF-JOB (/&).
          ALL TEMPORARY LOT-ENTRIES FOR TYPES PHASE, OBJ,
          SOURCE, PROC, DUMP (TEMPORARY LIBDEFS) ARE REMOVED.
          THE VIRTUAL I/O AREA FOR OPTION LINK OF THE PARTITION
          IS CLEARED.
          THE SVA-GETVIS SUBPOOL INLC AND INLG ARE FREED.
```

```
          5. MF=EXTEND  (AT LABEL IJBXTND):
          THIS ROUTINE PROCESSES THE MF=EXTEND PARAMETER OF THE
          LBRCTUPD MACRO. FOR AN ALREADY EXISTING LDT-ENTRY
          ADDITIONAL EDT-ENTRIES ARE ADDED WHILE THE LIBRARY
          CONTROL TABLES ARE LOCKED.
          IF THE LDT-ENTRY DOES NOT EXIST MESSAGE L152
          IS GIVEN.

REGISTER CONVENTIONS = R10, R11, R12, R9 USED AS BASE REGISTERS
                       R5 USED AS DATA REGISTER
                       OTHERS USED AS NEEDED

PATCH-LABEL =     PATCH1 LOCAL INITIALIZED IN LOCAL STORAGE
                  PATCH2 IN AUTOMATIC STORAGE

MODULE TYPE =     LIBRARIAN

                  PROCESSOR = PL/S COMPILER

                  MODULE SIZE = 15500 DECIMAL BYTES

                  ATTRIBUTES  = REENTRANT

ENTRY POINT =     IJBCTUPD

                  PURPOSE = SEE FUNCTION

                  LINKAGE = R1 LOADED WITH PARAMETER LIST ADDR.
                            R13 LOADED WITH SAVE AREA ADDRESS
                            R14 LOADED WITH RETURN ADDRESS
                            R15 LOADED WITH ENTRY POINT

INPUT =           PARAMETER LIST TOLCRQL
                  LIBRARY CONTROL TABLES

OUTPUT =          POINTERS TO ENTRIES (LIBINFO)
                  UPDATED LIBRARY CONTROL TABLES
                  MESSAGES, RETURN CODES
                  0 ... FUNCTION SUCCESSFULLY COMPLETED
                  4 ... CHAIN/ENTRY WAS ALREADY CLEARED
                        (ONLY FOR MF=CLEAR)
                  8 ... LOT OVERFLOW
                        (MORE THAN 15 CHAIN ENTRIES, OR
                         MORE THAN 31 TASK ENTRIES/PARTITION,
                         OR IALOT-POOL OVERFLOW)
                 12 ... FOR DEFINE=NEW:
                        (SUB-)LIBRARY DOES ALREADY EXIST
                        FOR DEFINE=OLD:
                        (SUB-)LIBRARY DOES NOT EXIST
                 16 ... EXTERNAL ERROR
                 20 ... INTERNAL ERROR
                 32 ... SECURITY VIOLATION
```

IJBLBBRN

```
MODULE IJBLBBRN
DESCRIPTIVE NAME: OPEN/EXTEND A LIBRARY

FUNCTION = 1. OPEN A LIBRARY (INPUT,OUTPUT) IN BAM OR VSAM
           MANAGED SPACE. THE LIBRARY MAY BE A PRIVATE LIBRARY
           OR A SYSTEM LIBRARY NAMED IJSYSR1..9, BUT NOT THE
           SYSTEM LIBRARY IJSYSRS.
           EACH EXTENT IS OPENED BY A SEPARATE INVOCATION OF
           $$BOPEN.
           CLOSE A LIBRARY WHICH RESIDES IN VSAM MANAGED SPACE.
           2. FOR INPUT: READ THE FIRST RECORD OF EACH EXTENT
           AND PERFORM BASIC CHECKS TO VERIFY THAT THE EXTENT
           ADHERES TO THE FORMAT OF A VSE LIBRARY.
           3. BUILD THE LDT/EDT/DDT ENTRIES IN AREAS PROVIDED
           BY THE CALLING ROUTINE.
           4. ASSIGN A TEMPORARY INTERNAL SYSTEM LOGICAL UNIT
           FOR EACH EXTENT. (THESE LOGICAL UNITS ARE FREED
           BY MODULE IJBCTUPD.)
           5. DELETE EXTENT-JIBS OF THE LOGICAL UNITS USED.
           6. EXTEND A LIBRARY IN VSAM MANAGED SPACE AND
           PASS BACK ONE OR MORE ADDITIONAL EDT ENTRIES
           (TOGETHER WITH THE ORIGINAL ONES).
           7. BEFORE A LIBRARY IS OPENED, MODULE INLPPIDT
           IS INVOKED TO CHECK IF THE LIBRARY IS ALREADY OPENED.
           8. AFTER A LIBRARY IS OPENED, MODULE INLPPIDT IS
           INVOKED TO ADD LIBRARY-RELATED INFORMATION TO THE
           ID-TABLE.
           9. LOCK LDT EXCLUSIVE TO PREVENT OTHER TASKS FROM
           MODIFYING LDT ENTRIES BEFORE THIS ENTRY IS INSERTED.
           10. BUILD THE LDT/DDT/EDT ENTRIES FOR ALL EXTENTS OF
           A LIBRARY IN VSAM MANAGED SPACE (MF+EXTUPD) AND PASS
           THEM BACK TO UPDATE A POSSIBLY BACK-LEVEL EDT IF THE
           LIBRARY IS ON A CPU-SHARED DASD.

INPUT = PARAMETER LIST BLDCBMAP (CREATED THROUGH MACROS LBRBLDRN/
                       LBRBLDCB).
        REGISTER 13 MUST CONTAIN THE ADDRESS OF A SAVE AREA
               OF 18 FULLWORDS.

OUTPUT = RETURN IN BLDCBLDT:  COMPLETE LDT-ENTRY (WITHOUT
                               PARTITION FLAGS)
                 IN BLDCBEDT:  ONE OR MORE EDT ENTRIES
                 IN BLDCBDDT:  ONE DDT ENTRY
         MESSAGES ON SYSLOG/SYSLST VIA MODULE INLPDIAG.
         RETURN CODE IN R15:
                               0 - SUCCESSFUL
                               8 - LIBRARY NOT EXTENDABLE
                              12 - LIBRARY DOES ALREADY EXIST
                                   (FOR OUTPUT)
                              16 - USER ERROR (THIS INCLUDES
                                   'LIBRARY DOES NOT EXIST')
                              20 - SYSTEM ERROR
```

```
ENTRY POINT: SAME AS MODULE NAME

PURPOSE = SEE FUNCTION DESCRIPTION

LINKAGE = R0  LOADED WITH FUNCTION CODE
          R1  LOADED WITH PARAMETER LIST ADDRESS
          R13 LOADED WITH SAVE AREA ADDRESS (18 FULLWORDS)
          R14 LOADED WITH RETURN ADDRESS
          R15 LOADED WITH ENTRY POINT
```

IJBLBHLS

```
MODULE IJBLBHLS
DESCRIPTIVE NAME: HIGH LEVEL SERVICES FOR LIB. CONTROL TABLES

ATTRIBUTES:       REENTRANT

PROCESSOR:        PL/S

PATCH AREA:       STATIC STORAGE   200 BYTES
                  DYNAMIC STORAGE  200 BYTES

FUNCTION =        ADD, DELETE AND PROVIDE ACCESS TO ENTRIES
                  IN THE LIBRARY CONTROL TABLES
                  THE FUNCTION IS INVOKED BY THE MACRO
                  LBRACCES.
                  (FOR A DETAILED FUNCTION DESCRIPTION SEE
                  DESCRIPTION OF MACRO LBRACCES.)

INPUT =           PARAMETER LIST LBRACCDS
                  LIBRARY CONTROL TABLES (LPT,LOT,LDT,SDT,EDT,
                                          DDT)

OUTPUT =          LIBINFO (POINTERS TO ENTRIES)
                  UPDATED LIBRARY CONTROL TABLES
                  MESSAGES, RETURN CODES
                   0 ... REQUEST HAS BEEN HONORED
                   4 ... MF=GET:
                         END OF CHAIN HAS BEEN REACHED FOR A
                         CHAINING REQUEST. POINTERS ARE STILL
                         POINTING TO THE LAST VALID LIB/SUBLIB.
                   8 ... MF=GET: REQUESTED LIBRARY HAS NOT BEEN
                                 DEFINED.
                         MF=ADDLIB: END OF CHAIN CAPACITY
                                 REACHED.
                         MF=EXTEND: NO EXTENSION POSSIBLE.
                         MF=EXTUPD: NO UPDATE OF EDT NECESSARY.
                  12 ... MF=ADDLIB:
                         FOR LEVEL=LIB AND DEFINE=NEW:
                             LIBRARY ALREADY EXISTS
                         FOR LEVEL=LIB AND DEFINE=OLD:
                             LIBRARY DOES NOT EXIST
                         FOR LEVEL=SUBLIB AND DEFINE=NEW:
                             SUBLIBRARY ALREADY EXISTS
                         FOR LEVEL=SUBLIB AND DEFINE=OLD:
                             SUBLIBRARY DOES NOT EXIST
                  16 ... EXTERNAL ERROR WITH FEEDBACK CODE
                  20 ... INTERNAL ERROR WITH FEEDBACK CODE
                  (FIELD LBRRET CONTAINS FEEDBACK CODE,
                   FIELD LBRRET1 CONTAINS RETURN CODE OF
                                 FAILING SERVICE)
                  32 ... SECURITY VIOLATION
                 255 ... LAMB IS MISSING
```

IJBLBIPL

```
MODULE IJBLBIPL
DESCRIPTIVE NAME: INITIALIZE LIBRARY CONTROL TABLES

  PHASE NAME  :         $ I N I T C O N

  MODULE NAME :         I J B L B I P L

  FUNCTION    :  ALLOCATION AND INITIALIZATION OF LIBRARY
                 CONCATENATION TABLES.

                 PHASE WILL BE EXECUTED TWICE DURING IPL TIME:
                 1. CALL - CALCULATION OF SYSTEM GETVIS SPACE
                           FOR LIBRARY CONCATENATION.
                           (MINIMUM SYSTEM GETVIS SPACE).
                         - INPUT  ->   IPLSDTE   (IPL LIST)
                         - OUTPUT ->   ICTABLEN  (IPL LIST)
                 2. CALL - BUILD AND INITIALIZATION OF LIBRARY
                           CONCATENATION CONTROL TABLES.
                         - INPUT  ->    IPLSDTE  (IPL LIST)
                         - OUTPUT ->    CONTROL TABLES

  ENTRY POINT
     - CALL 1 :  FETCHED BY $IPLRT6 AND ENTERED AT INITSTRT
     - CALL 2 :  FETCHED BY $IPLRT7 AND ENTERED AT INITSTRT

  I P L -     :  1   IPLLIST   BASED      POINTER IN REG 1
  INTERFACE       2  IPLCALL   BIT(8)     IPL CALL BYTE :
                   3 IPLCALL1  BIT(1)     - FIRST CALL
                   3 IPLCALL2  BIT(1)     - SECOND CALL
                   3           BIT(6)     - NOT USED
                  2            BIT(8)     RESERVED
                  2  IPLSDTE   BIT(16)    # OF SUBLIBS IN SDT
                  2  ICTABLEN  FIXED(31)  LENGTH CONCAT. CHAIN
                  2  IPLNPART  FIXED(15)  MAX # OR PARTITIONS

  REGISTER    :   R1   POINTS TO IPL PARAMETER LIST
                  R13  POINTS TO REG SAVE AREA
                  R14  RETURN REGISTER
                  R15  RETURN CODE

  INPUT:          MAXIMUM NUMBER OF ACTIVE SUBLIBRARIES

  OUTPUT:         INITIALIZED LIBRARY CONTROL TABLES
                  RETURN CODES:
                   0      SUCCESSFUL COMPLETION
                  30-34   ERROR OCCURRED DURING EXECUTION
                          SEE ERROR HANDLING CODE
                  XX      RETURN CODE (R>0) OF GETVIS REQUEST
```

```
EXIT-NORMAL:    VIA PLS RETURN STATEMENT

ENTRY POINT:    SAME AS MODULE NAME

EXT. REFERENCES

MACROS USED :  SYSCOM,GETVIS,SUBSID-INQUIRY,GETVCE,AVRLIST,
               INLCLPT,INLCLOT,INLCLDTE,INLCEDTE,INLCDDTE,
               INLCIDTE,INLCSDTE,INLCRESN,INLCLDES,INLCLACB,
               INLCSACB,INLCLDES,INLCEXTD,INLCSLXE,MAPCOMR
               LIBINFO,DCTENTRY,IORB,CCW,SGENL,INLMLAMB,
               INLMLRPL
```

IJBLBLLS

```
MODULE IJBLBLLS
DESCRIPTIVE NAME: LOW LEVEL SERVICE FOR LIBRARY CONTROL TABLES

FUNCTION =        PROVIDE ACCESS TO LIBRARY CONTROL TABLES
                  WITH EXTRACT INFORMATION OF REQUESTED
                  LIBRARY/SUBLIBRARY.

INPUT =           PARAMETER BLOCK LBRLCTCB
                  LIBRARY CONTROL TABLES

OUTPUT =          LIBINFO (POINTERS TO LOT, LDT, SDT)
           POINTER TO LIBRARY OFFSET TABLE ENTRY,
           LIBRARY DEFINITION TABLE ENTRIES, AND
           SUBLIBRARY DEFINITION TABLE ENTRY
           FOR THE REQUESTED LIBRARY, IF THE REQUEST
           CAN BE HONORED. IF THE REQUEST WAS FOR AN SDL
           ENTRY (IN PHASE SEARCH CHAIN), ONLY THE POINTER
           TO THE LOT ENTRY IS RETURNED, THE LDT POINTERS
           ARE SET TO ZERO.
           IF A CHAINING REQUEST RESULTS IN AN 'END-OF-CHAIN'
           CONDITION THIS IS PASSED BACK IN THE RETURN CODE.
           THE POINTERS ARE STILL POINTING TO THE LAST VALID
           LIBRARY IN THE CHAIN (SDT-PTR NO MORE VALID).
           RETURN CODE IN REGISTER 15:
   0 ...    REQUEST HAS BEEN HONORED
   4 ...    END OF CHAIN REACHED
   8 ...    REQUESTED LIBRARY HAS NOT BEEN DEFINED
 255 ...    SYSTEM ERROR: INVALID PARAMETERS HAVE BEEN
                          SPECIFIED FOR GET REQUEST, FEEDBACK
                          CODE AVAILABLE IN LBRLRET

 ENTRY POINT:     SAME AS MODULE NAME
```

IJBLBRT

```
MODULE IJBLBRT
DESCRIPTIVE NAME: BRANCH ROUTINE

FUNCTION =        WITH A FUNCTION CODE IN REGISTER 0 A BRANCH
                  TABLE IS INDEXED TO FIND THE MODULE FOR
                  THE REQUESTED FUNCTION

INPUT =           FUNCTION CODE IN REGISTER 0

OUTPUT =          ADDRESS OF MODULE IN REGISTER 15

 ENTRY POINT:     BRANCH
```

IJBLBSL

```
MODULE IJBLBSL
DESCRIPTIVE NAME: LIBRARY INTERFACE FOR SOURCE BOOKS

FUNCTION =        ON REQUEST FROM A DTFSL MACRO CALL FROM A
                  PARTITION THE FOLLOWING MACRO CALLS ARE
                  TRANSFORMED TO THE LIBRARY INTERFACE FOR
                  CONCATENATED SUBLIBRARIES IN THE SOURCE
                  SEARCH CHAIN OF THE LIBDEF STATEMENT:
                  - SEARCH A BOOK IN A SUBLIBRARY'S DIRECTORY
                    (FINDSL) - INLMFIND
                  - RETRIEVE SOURCE STATEMENT CARDS
                    SEQUENTIALLY FROM A LIBRARY
                    (GETSL) - INLMGETR
                  - NOTE THE POSITION OF MEMBER PROCESSING
                    (NTSL) - INLMNOTE
                  - RESTORE AN EARLIER POSITION OF MEMBER
                    PROCESSING (PTSL) - INLMPOIN

INPUT =           REQUEST LIST AS PREPARED BY THE MACRO
                  EXPANSION OF DTFSL TRIGGERED BY ONE OF THE
                  IMPERATIVE MACROS FINDSL, GETSL, ...

OUTPUT =          CONNECTION TO MEMBER (FINDSL)
                  NEXT CARD IMAGE, DECOMPRESSED (GETSL)
                  POSITION OF CURRENT MEMBER PROCESSING (NTSL)
                  RESTORED EARLIER STATE OF CONTROL BLOCK AND
                  BUFFER FOR MEMBER PROCESSING (PTSL)

 ENTRY POINT:     SAME AS MODULE NAME
```

IJBLBULT

```
MODULE IJBLBULT
DESCRIPTIVE NAME: SET/RESET LDT/SDT CONTROL FLAGS

FUNCTION = CONTROL SHARING OF LIBRARIES AND SUBLIBRARIES
           ACROSS PARTITIONS AND BETWEEN CPUS.
           THIS IS ACCOMPLISHED BY SETTING/RESETTING THE
           LDT/SDT FLAGS NOAC,DELT,ACC AND NEWL/NEWS FOR
           PRIVATE LIBRARIES/SUBLIBRARIES AND FOR THE SYSTEM
           SUBLIBRARY.
           FUNCTION REQUESTS:
           SET=NOACC :  SET LDTENOAC/SDTENOAC IF LIBRARY/
                        SUBLIBRARY IS UNIQUELY ASSIGNED.
                        RESET LDTENEWL FOR LIBRARIES.
                        LOCK LIBRARY/SUBLIBRARY.
           SET=ACC :    RESET FLAG LDTENOAC/SDTENOAC AND
                        LDTENEWL/SDTENEWS AND LDTEDELT/SDTEDELT.
                        UNLOCK LIBRARY/SUBLIBRARY FOR
                        LDTENOAC/SDTENOAC=ON.
           SET=DEL :    SET FLAG LDTEDELT/SDTEDELT IF LIBRARY/
                        SUBLIBRARY IS UNIQUELY ASSIGNED.
                        NOT ACCECTED FOR SYSTEM SUBLIBRARY.
           'UNIQUELY ASSIGNED' MEANS:
           GENERAL:
            - LIBRARY/SUBLIBRARY MUST NOT RESIDE ON A DASD
              SHARED BETWEEN CPUS.
            - LIBRARY/SUBLIBRARY MUST BE ATTACHED TO ONE
              TASK ONLY.
           SPECIFIC FOR SET=NOACC:
            - LIBRARY/SUBLIBRARY MUST NOT SHOW UP IN THE
              PROC SEARCH CHAINS (NEITHER TEMP NOR PERM)
              IF A PROCEDURE IS BEING EXECUTED.
            - FOR IJSYSRS OR IJSYSRS.SYSLIB: NOT MORE THAN
              ONE PARTITION MUST BE ACTIVE. NO SUBTASKS MUST
              BE ACTIVE.
           SPECIFIC FOR SET=DEL:
            - LIBRARY/SUBLIBRARY MUST NOT SHOW UP ON ANY
              LIBDEF SPECIFICATION.
            - WHEN RUNNING UNDER MSHP, A LIBRARY OR SUBLIBRARY
              CAN BE DELETED EVEN IF IT RESIDES ON A
              CPU-SHARED DASD.
            - A PROMPTING MESSAGE IS DISPLAYED ON THE CONSOLE
              WHEN A DELETE REQUEST IS GIVEN FOR A LIBRARY
              OR SUBLIBRARY AND THE FIRST EXTENT OF THE LIBRARY
              RESIDES ON A CPU-SHARED DASD.
```

```
INPUT = PARAMETER LIST (CREATED THROUGH MACRO LBRUPDAT).
        REGISTER 13 MUST CONTAIN THE ADDRESS OF A SAVE AREA
        OF 18 FULLWORDS.

OUTPUT = LDT/SDT ENTRY IS UPDATED.
         MESSAGES ON SYSLST/SYSLOG VIA MODULE INLPDIAG.
         RETURN CODE IN R15:
                     0 - SUCCESSFUL
                     8 - NOT UNIQUELY ASSIGNED FOR SET=NOACC
                    16 - NOT UNIQUELY ASSIGNED FOR SET=DEL
                    20 - SYSTEM ERROR

 ENTRY POINT: SAME AS MODULE NAME

 PURPOSE = SEE FUNCTION DESCRIPTION

 LINKAGE =  R0  LOADED WITH FUNCTION CODE
            R1  LOADED WITH THE PARAMETER LIST ADDRESS
            R13 LOADED WITH SAVE AREA ADDRESS (18 FULLWORDS)
            R14 LOADED WITH RETURN ADDRESS
            R15 LOADED WITH ENTRY POINT
```

INLACONL

```
MODULE INLPCONL
DESCRIPTIVE NAME: CONNECT TO LIBRARY

FUNCTION =        MAKES LIBRARY AVAILABLE FOR ACCESS.
                  POSITIONS TO START OF SUBLIBRARY INDEX.

   =====> THE LIBRARY GIVEN WITH LRPLINFO IS CONNECTED AND
          THE CURSOR POSITIONED TO THE START OF THE SUBLIBRARY
          INDEX.
          - AT MODULE ENTRY: NO CONNECTION MUST HOLD.
                             THE SPACE FOR THE LACB IS ALLOC.
          - THE SHARED BUFFERS ARE INITIALIZED
          IF THE LIBRARY IS NEW, ADDITIONALLY
          - THE FREE SPACE MAP IS INITIALIZED
          - THE LIBRARY DESCRIPTOR IS INITIALIZED

INPUT =           LIBRARY REQUEST PARAMETER LIST INLCLRPL
                  LIBRARY ACCESS CONTROL BLOCK INLCLACB
          - PARAMETERS ARE: LRPLPTR - ADDRESS OF LRPL
                            LACBPTR - ADDRESS OF LACB
                            CONNECT - INDICATION WHETHER OLD OR
                                      NEW CONNECTION REQUESTED

OUTPUT =          INITIALIZED LIBRARY ACCESS CONTROL BLOCK
                  (INLCLACB)
                  MESSAGES, RETURN CODES:
                  0... SUCCESSFULLY COMPLETED
                 16... EXTERNAL ERROR WITH FEEDBACK CODE
                 20... INTERNAL ERROR WITH FEEDBACK CODE

 ENTRY POINT:     SAME AS MODULE NAME
```

INLACONS

```
MODULE INLPCONS / INLXCONS
DESCRIPTIVE NAME: CONNECT TO SUBLIBRARY

FUNCTION =        MAKES SUBLIBRARY AVAILABLE FOR ACCESS
                  POSITIONS TO START OF MEMBER INDEX

  ======> THIS PROCEDURE IS CALLED TO CONNECT TO AN EXISTING
          SUBLIBRARY (CONNECT=OLD) WHICH ALLOWS OPERATIONS ON
          SUBLIBRARY LEVEL OR TO CREATE A NEW SUBLIBRARY
          (CONNECT=NEW).
          - THE PASSED SUBLIBRARY NAME IS LOOKED UP IN THE
          CONNECTED LIBRARY.
          - FOR CONNECT=NEW: IF THE SUBLIBRAY EXISTS RET.CODE
          8 IS PASSED BACK. OTHERWISE, WHILE THE LIBRARY IS
          LOCKED THE FIRST LEVEL 1 INDEX LIBRARY BLOCK IS
          ALLOCATED TO THE SUBLIBRARY, AND THE SUBLIBRARY
          DESCRIPTOR IS BUILT AND INITIALIZED.
          - FOR CONNECT=OLD: IF THE SUBLIBRARY DOES NOT EXIST
          RETURN CODE 8 IS PASSED BACK. OTHERWISE, THE EXISTING
          SUBLIBRARY ACCESS CONTROL BLOCK (SACB) IS INITIALIZED
          WITH INFORMATION OUT OF THE SUBLIBRARY DESCRIPTOR.

INPUT =           LIBRARY REQUEST PARAMETER LIST (INLCLRPL)
                  SUBLIBRARY ACCESS CONTROL BLOCK (INLCSACB)

OUTPUT =          INITIALIZED INLCSACB
                  MESSAGES, CONTROL BLOCKS
                  RETURN CODES: 0 SUCCESSFULLY
                                8 SUBLIBRARY DOES NOT EXIST
                                  (FOR CONNECT=OLD)
                                  SUBLIBRARY ALREADY EXIST
                                  (FOR CONNECT=NEW)
                               12 LIBRARY IS FULL
                               16 PARAMETER ERROR
                               20 PROCESSING ERROR
                               32 SECURITY VIOLATION
                               (FEEDBACK CODES WITH 16 AND 20)

 ENTRY POINT:     SAME AS MODULE NAME
```

INLADOIO

```
MODULE INLPDOIO
DESCRIPTIVE NAME:  EXECUTE I/O REQUESTS  AT SVC LEVEL

FUNCTION =       SUBMIT AN I/O
                 REQUEST VIA EXCP (READ OR WRITE)

INPUT =          R1: ADDR OF LRPL OR LAMB
                 R3: ADDRESS OF IORB

OUTPUT =         RETURN CODE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLAGST

```
 MODULE INLPGST
 DESCRIPTIVE NAME: STORAGE MANAGEMENT

 FUNCTION =

     THIS ROUTINE IS TO PROVIDE STORAGE ALLOCATION SUPPORT AND
     STORAGE FIXING SUPPORT (VSE ONLY) FOR THE LEVEL 3 COMMANDS.

 ENTRY-POINTS = AS FOLLOWS

     INLPGST:  ACQUIRE A SPECIFIED AMOUNT OF CONTIGUOUS STORAGE
               OUT OF THE AVAILABLE ADDRESS SPACE

                    THE AVAILABLE ADDRESS SPACE IS INITIALLY
                    THE RANGE BETWEEN THE END ADDRESS OF
                    THE LAST PHASE LOADED AND THE HIGHEST
                    ADDRESS AVAILABLE TO THE PARTITION
                    (STANDALONE: HIGHEST MAIN STORAGE ADDRESS)
                    EACH CALL OF INLPGST UPDATES THE BEGIN OF
                    THE NOW AVAILABLE ADDRESS SPACE.

INPUT =     BUFFSIZE    THE LENGTH OF THE REQUIRED STORAGE AREA
OUTPUT =    BUFFADDR    THE ADDRESS OF THE BUFFER
            REMSPACE    THE LENGTH OF THE REMAINING SPACE IN
                            THE PARTITION
            RETURN CODES :
                        0 BUFFER IS AVAILABLE AS REQUESTED
                        4 BUFFER WAS ALLOCATED BUT
                            PFIX REQUEST COULD NOT BE SATISFIED
                        8 ACQUISITION OF BUFFER SPACE FAILED

     INLPGSTH: RESET THE START ADDRESS TO THE END ADDRESS
                    OF THE LAST PHASE LOADED.
                     (IN EFFECT A FREE STORAGE)

OUTPUT =    REMSPACE    THE LENGTH OF THE REMAINING SPACE IN
                            THE PARTITION

     INLPGSTI: RESET THE START OF THE AVAILABLE ADDRESS SPACE TO
                    THE VALUE PASSED AS ARGUMENT.
                     (IN EFFECT A FREE STORAGE)

INPUT =     RESETPT     POINTER TO THE START ADDRESS TO
                        BE USED FOR THE NEXT BUFFER REQUEST(S)
OUTPUT =    REMSPACE    THE LENGTH OF THE REMAINING SPACE IN
                            THE PARTITION
```

INLAMCON

```
MODULE INLPMCON
DESCRIPTIVE NAME: CONNECT TO MEMBER

FUNCTION =        MAKES MEMBER AVAILABLE FOR ACCESS
                  POSITIONS TO START OF MEMBER
 ======> THIS MODULE IS CALLED BY THE INLMMCON MACRO FROM
         LEVEL 3 SERVICES OR BY OTHER LEVEL 2 SERVICES.
         IT IS USED TO CONNECT TO A MEMBER (BUILD THE STRING
         LACB-SACB-MACB, ATTACHED TO THE LRPL). AFTERWARDS
         THE MEMBER IS ACCESSIBLE FOR INLMGETR, INLMPUTR,
         INLMNOTE, INLMPOIN  AND DISCONNECT OPERATIONS.
         - THE ADDRESSED SUBLIBRARY (BY LIBINFO PARAMETER) MUST
           HAVE BEEN ENTERED IN THE LIBRARY CONTROL TABLES
           (LOT, LDT, SDT, EDT, DDT).
         - THE MEMBER TO BE CONNECTED IS PASSED IN THE
           ARGUMENT (PARAMETER ARG).
         - ADDITIONAL PARAMETERS (LOCKID, TIME FRAME) ARE
           PASSED IN THE INLCLRPL.
         - CONNECT=OLD|UPDATE: THE MEMBER IS SUPPOSED TO EXIST
           ON THE SUBLIBRARY ADDRESSED BY LIBINFO.IF IT DOESN'T
           EXIST OR DOES NOT MATCH THE LOCKID OR TIME FRAME, IF
           PASSED IN THE INLCLRPL, RETURN CODE 12 IS PASSED   .
           PASSED BACK TO THE INVOKER.
         - CONNECT=UPDATE: IF THE MEMBER IS DIRECTORY LOCKED,
           RETURN CODE 12 IS RETURNED AND LARGDLCK IS SET.
           THE MEMBER SPACE WILL BE SUPERVISOR LOCKED.
         - CONNECT=NEW: THE MEMBER IS SUPPOSED TO BE CREATED
           IN THE SUBLIBRARY. NO CHECK FOR ITS EXISTENCE IS
           DONE. THE MEMBER SPACE WILL BE SUPERVISOR LOCKED.
         - SECURITY CHECK IS DONE AS FOLLOWS:
           CONNECT=OLD REQUIRES READ ACCESS TO THE MEMBER.
           CONNECT=NEW REQUIRES UPDATE ACCESS TO THE MEMBER.
           CONNECT=UPDATE REQUIRES UPDATE ACCESS TO THE MEMBER.
           (IF THE LIBRARY IS NOT PROTECTED, THIS HOLDS BY DEF)
           ANY SECURITY VIOLATION CAUSES RETURN CODE 32.
         - THE ACCESS CONTROL BLOCKS ARE CONNECTED IN THE
           SEQUENCE LACB,SACB,MACB BY ISSUING A GETVIS REQUEST
           AND INITIALIZING IT. (NOTE: AT ENTRY, EITHER A
           CONNECTION TO THE LIBRARY, TO THE SUBLIBRARY, OR TO
           THE MEMBER CAN HOLD, OR NO CONNECTION AT ALL.
           MISSING CONNECTIONS ARE ADDED).
         - IF THE PROPER LIBRARY OR SUBLIBRARY IS ALREADY
           CONNECTED, IT WILL NOT BE CONNECTED NEWLY. OTHERWISE
           THE CORRESPONDING LIBRARY AND/OR SUBLIBRARY IS
           DISCONNECTED (KEEPING THE SPACE FOR THE ACC.CONTROL
           BLOCKS) AND THE CORRECT LIBRARY AND/OR SUBLIBRARY
           IS CONNECTED. A ALREADY CONNECTED MEMBER WILL ALWAYS
           BE DISCONNECTED AND A NEW CONNECTION DONE (FORCES
           INITIALIZATION OF MACB AND PRIVATE BUFFERS).
```

```
          - THE INITIALIZATION OF THE MACB EFFECTS THAT THE
            FOLLOWING MEMBER OPERATION STARTS FROM THE BEGINNING
            OF THE MEMBER.
          - THE FUNCTION ALSO APPLIES TO A GENERIC MEMBER
            SPECIFICATION. IN THIS CASE THE FIRST MEMBER
            FULFILLING THE NAME AND TYPE SPECIFICATION WILL BE
            CONNECTED.
          - A MEMBER BEING CONNECTED WITH CONNECT=OLD MAY ONLY
            BE READ (NO WRITE ACCESS ALLOWED).
          NOTE THE SPECIAL IMPLEMENTATION CONSIDERATIONS:
          - IF FOR CONNECT=OLD|UPDATE THE MEMBER DESCRIPTOR
            (PARAMETER DENT, E.G. AS IT IS RETURNED BY A FORMER
            INLMBLDL INVOCATION) IS PASSED
            WITH THE MACRO INLMMCON, NO SEARCH FOR THE MEMBER
            IN THE SUBLIBRARY IS DONE, BUT THE MACB INITIALIZED
            WITH INFORMATION PASSED WITH THE GIVEN DIRECTORY
            ENTRY (PERFORMANCE REASON).
          - IF FOR CONNECT=NEW AN MACB IF PASSED WITH THE
            MACRO INLMMCON (PARAMETER SAVE), THE MACB IS NOT
            INITIALIZED, BUT THE GIVEN MACB IS USED FOR THE
            CONNECTION (USED BY LNKEDT).
          - IF THE INLMMCON SERVICE IS CALLED DURING THE NESTED
            PROCESSING WITHIN A CHAIN, AFTER A POINT OPERATION
            (MACRO INLMPOIN) INFORMATION IN THE LACB, SACB, AND
            MACB HAS BEEN LOST (DUE TO THE RESTRICTED LENGTH OF
            THE NOTE INFORMATION WORD). THEREFORE, WHENEVER A
            CONNECT-TO-MEMBER IS GIVEN AFTER A POINT OPERATION
            AND THE CORRECT SUBLIBRARY AND/OR CORRECT LIBRARY
            IS ALREADY CONNECTED, THIS INFORMATION IS NOT
            TRUSTED BUT THE LACB AND SACB ARE BUILT NEWLY.
          - THE MODULE CAN NOT WAIT FOR A LOCK FOR MEMBER SPACE.
            IF THIS LOCK FAILS (CONNECT=NEW|UPDATE) RETURN CODE.
            16 AND FEEDBACK CODE FDBCMLCK IS RETURNED.         .

INPUT =           LIBRARY REQUEST PARAMETER LIST (INLCLRPL)

OUTPUT =          MEMBER ACCESS CONTROL BLOCK (INLCMACB)
                  INLCLARG :
                    LARGDLCK : TRUE ==> MEMBER DIRECTORY LOCKED
                    LARGSLCK : TRUE ==> MEMBER SPACE     LOCKED
                    LARGDLID : TRUE ==> LOCKID MATCHES
                  MESSAGES, RETURN CODES
                     0 ... FUNCTION SUCCESSFULLY COMPLETED
                    12 ... MEMBER NOT FOUND OR DIRECT.-LOCKED
                               (FOR CONNECT=OLD|UPDATE)
                    16 ... EXTERNAL ERROR WITH FEEDBACK CODE
                           OR MEMBER IN WRITE MODE
                    20 ... INTERNAL ERROR WITH FEEDBACK CODE
                    32 ... SECURITY VIOLATION

 ENTRY POINT:     SAME AS MODULE NAME

EXT.REFERENCES    INLPXMLK
```

INLARABF

```
MODULE INLPRABF
DESCRIPTIVE NAME: BACKUP/RESTORE - ALLOCATE BUFFERS

FUNCTION =      ALLOCATE BUFFER SPACE

   ENTRY POINT: INLPRABF
                - ALLOCATE TAPE BUFFERS AND
                - ALLOCATE SPACE FOR PRIVATE AND
                          SHARED DISK BUFFER POOLS

   ENTRY POINT: INLPRAB1
                - REALLOCATE SHARED AND PRIVATE BUFFERS
                  AND BACKUP/RESTORE WORK AREA.

   ENTRY POINT: INLPRAB2
                - ALLOCATE SHARED AND PRIVATE BUFFERS
                  AND RESTORE WORK AREA FOR RESTORE OLDLIB

INPUT =          INLCBRCR  -  BACKUP/RESTORE COMREG

OUTPUT =         BDBPTR  - POINTING TO LOOP OF TAPE BUFFERS
                 PRVBFPTR - ADDR OF PRIVATE DISK BUFFER POOL
                 PRVBFLEN - LENGTH OF PRIVATE DISK BUFFER POOL
                 SHRBFPTR - ADDR OF SHARED DISK BUFFER POOL
                 SHRBFLEN - LENGTH OF SHARED DISK BUFFER POOL
                 BRWAPTR  - BACKUP/RESTORE WORK AREA

                 RETURN CODES
                   0 ... ALLOCATION SUCCESSFUL
                   4 ... PARTION TOO SMALL

 ENTRY POINT:    SAME AS MODULE NAME

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLARDTP

```
MODULE INLPRDTP
DESCRIPTIVE NAME: READ TAPE BLOCK

FUNCTION =      READ TAPE BLOCK

   ENTRY POINT: INLPRDTP  -  READ TAPE BLOCK

                - THE NEXT TAPE BLOCK IS READ INTO THE NEXT
                  BUFFER OF THE BDB CONTROL BLOCK LOOP.
                  (MODULE INLPTI IS CALLED WHICH
                   CONTROLS THE READING OF THE TAPE
                   BUFFERS USING ALL BUFFERS OF THE
                   BDB-LOOP.)

INPUT =       BDBPTR - POINTER TO CURRENT BDB.

OUTPUT =      BDBPTR - PTR TO BUFFER DEFINITION BLOCK
                       CONTAINING THE NEXT INPUT BUFFER
              BUFPTR - PAST BLKHDR.
              BUFEND - BUFAD + BLKLEN.

              RETURN CODES
                   0 ... BUFFER READ SUCCESSFUL
                   4 ... BUFFER CONTAINS INVALID DATA

   ENTRY POINT: INLPRDT1

                - POSITION TAPE TO FIRST BACKUP FILE ID
                  OR TO FIRST INFO-RECORD (IF 'OLD' TAPE)
                  BY FSF MAXIMALLY TWO FILES IF NECESSARY.
                  THIS POSITIONING IS REPEATED IF THE START
                  RECORD OF A PID-V2-STACKED TAPE IS FOUND.
                  IF AN END-OF BACKUP RECORD IS FOUND AND A
                  PID-V2-STACKED TAPE IS INDICATED THIS
                  POSITIONING IS ALSO REPEATED BUT ALLOWING
                  MAXIMALLY FOUR FILES SKIPPED IF NECESSARY.

OUTPUT =      RETURN CODES
                   0 ... BACKUP-FILE-ID FOUND
                   4 ... TAPE CONTAINS INVALID DATA

   ENTRY POINT: INLPRDT2

                - POSITION TAPE TO NEXT BACKUP FILE ID
```

```
OUTPUT =      EOSCAN  - TRUE: END OF BACKUP RECORD FOUND OR
                            END OF STACKED ID IS FOUND IF
                            INPUT TAPE IS A PID-V2-STACKED TAPE
              RETURN CODES:
                   0 ... BACKUP-FILE-ID FOUND
                   4 ... TAPE CONTAINS INVALID DATA OR
                         IS NOT POSITIONED CORRECTLY

   ENTRY POINT: INLPRDT3

                - CHECK IF HISTORY FILE FOLLOWS BACKUP FILE ID
                  ON TAPE AND SKIP HISTORY FILE

OUTPUT =      HFFOUND - TRUE: HISTORY FILE FOUND
```

INLARELB

```
MODULE INLPRELB
DESCRIPTIVE NAME: RESTORE LIBRARIES

FUNCTION =    RESTORE LIBRARIES

              THIS FUNCTION RESTORES LIBRARIES ON TAPES
              CREATED WITH THE BACKUP COMMAND

              - THE LIBRARY FILENAME FOR EACH LIBRARY FOUND
                ON THE BACKUP TAPE IS COMPARED WITH THE
                LIBRARY FILENAMES IN THE LIBRARY NAME LIST
                ADDRESSED BY INLCCMDP. THE LIBRARY IS RESTORED
                IF FOUND IN THE LIST.

              - RESTLIB: RESTORE LIBRARY
                 - BUILD NEW LIBRARY:
                     - ADD TEMP. LIBRARY ENTRY TO THE
                       LIBRARIAN TABLES (LBRACCES)
                     - CONNECT TO LIBRARY
                 - CHECK FOR SYSTEM LIBRARY BUILD (NAMES
                   IJSYSR1 TO IJSYSR9):
                     - RESTORE IPL RECORDS IF SYSTEM LIBRARY
                       RESTORED (STORIPL)
                 - DO FOR ALL SUBLIBRARIES FOUND ON TAPE FOR
                   THE LIBRARY:
                     - ADD TEMP. LIB/SUBLIB ENTRY TO THE
                       LIBRARIAN TABLES (LBRACCES)
                     - RESTORE SUBLIBRARY (INLPRES2)

       NOTE: IF THE TARGET LIBRARY IS UNIQUELY USED (I.E NO
             ACCESS IS ACTIVE FOR ANOTHER TASK, THE LIBRARY
             IS NOT ON A VOLUME WHICH IS SHARED ACROSS
             CPUS AND NO LIBDEF IS SPECIFIED) THE LIBRARY
             IS LOCKED (THE "NO ACCESS ALLOWED" FLAG IS
             TURNED ON) TO IMPROVE THE PERFORMANCE OF
             THE RESTORE PROCESS.

INPUT =          - LIBRARY-SPECIFICATION-LIST
                 - BACKUP FILE ON INPUT TAPE CONTAINING
                   ONLY LIBRARIES
```

```
OUTPUT =          EACH LIBRARY FOUND ON TAPE AND CONTAINED IN
                  THE SPECIFICATION-LIST IS RESTORED TO
                  THE TARGET LIBRARY

                  INFORMATION ON SYSLOG/SYSLST INDICATING
                  WHETHER THE LIBRARIES IN THE LIBRARY-
                  SPECIFICATION-LIST ARE RESTORED OR NOT

                  RETURN CODES
                    0 ... NORMAL EXIT :
                              CRGRETCD = 0 :
                                  RESTORE SUCCESSFUL
                              CRGRETCD = 8 AND MESSAGE:
                                  IF A LIBRARY IS SKIPPED
                    4 ... INTERNAL OR EXTERNAL ERROR
                          WITH FEEDBACK CODE

 ENTRY POINT:     SAME AS MODULE NAME

   LINKAGE:       PLS/III CALL CONVENTIONS
```

INLAREMB

```
MODULE INLPREMB
DESCRIPTIVE NAME: RESTORE MEMBERS

FUNCTION =    RESTORE MEMBERS

              THIS FUNCTION RESTORES MEMBERS CONTAINED
              IN SUBLIBRARIES/LIBRARIES ON BACKUP TAPES
              CREATED WITH THE BACKUP COMMAND.

              - THE MEMBER NAME (FULLY QUALIFIED) FOR
                EACH MEMBER FOUND ON THE BACKUP TAPE IS
                COMPARED WITH THE MEMBER NAMES IN THE
                MEMBER SPECIFICATION LIST ADDRESSED BY
                INLCCMDP. THE MEMBER IS RESTORED IF FOUND
                IN THE LIST TO THE TARGET SUBLIBRARY.
                (THE TARGET SUBLIBRARY IS EITHER SPECIFIED
                 IN THE MEMBER SPECIFICATION AS SECOND
                 PARAMETER OR WITH THE ACCESS COMMAND.)
              - RESTMEMB: RESTORE MEMBER
                - IF THE TARGET LIBRARY/SUBLIBRARY NAME
                  CHANGES:
                    - STOW MEMBERS OF LAST SUBLIBRARY(INLPMSTW)
                    - DISCONNECT FROM LAST TARGET LIBRARY
                    - ADD TEMP LIB/SUBLIB ENTRY TO THE
                      LIBRARIAN TABLES (LBRACCES)
                    - CONNECT TO THE SUBLIBRARY
                - RESTORE MEMBER (INLPREM2)
                - DISCONNECT FROM SUBLIB
              - AT END OF PROCESSING:
                - STOW MEMBERS OF LAST SUBLIBRARY (INLPMSTW)
                - DISCONNECT FROM LIBRARY LAST USED

       NOTE: IF THE TARGET LIBRARY IS UNIQUELY USED (I.E NO
             ACCESS IS ACTIVE FOR ANOTHER TASK, THE LIBRARY
             IS NOT ON A VOLUME WHICH IS SHARED ACROSS
             CPUS AND NO LIBDEF IS SPECIFIED) THE LIBRARY
             IS LOCKED (THE "NO ACCESS ALLOWED" FLAG IS
             TURNED ON) TO IMPROVE THE PERFORMANCE OF
             THE RESTORE PROCESS.

INPUT =          - MEMBER-SPECIFICATION-LIST
                 - BACKUP FILE ON INPUT TAPE CONTAINING
                   LIBRARIES OR SUBLIBRARIES
```

```
OUTPUT =          EACH MEMBER FOUND ON TAPE AND CONTAINED IN
                  THE SPECIFICATION-LIST IS RESTORED TO
                  THE TARGET SUBLIBRARY

                  INFORMATION ON SYSLOG/SYSLST INDICATING
                  WHETHER THE MEMBERS IN THE MEMBER-
                  SPECIFICATION-LIST ARE RESTORED OR NOT

                  RETURN CODES
                    0 ... NORMAL EXIT :
                              CRGRETCD = 0 :
                                  RESTORE MEMBERS SUCCESSFUL
                              CRGRETCD = 8 AND MESSAGE :
                                  IF A MEMBER IS SKIPPED
                    4 ... INTERNAL OR EXTERNAL ERROR
                          WITH FEEDBACK CODE

 ENTRY POINT:     SAME AS MODULE NAME

   LINKAGE:       PLS/III CALL CONVENTIONS
```

INLAREM2

```
MODULE INLPREM2
DESCRIPTIVE NAME: RESTORE MEMBER

FUNCTION =       RESTORE ONE MEMBER FROM TAPE INTO THE
                 CONNECTED SUBLIBRARY
                 PERFORM STOW MEMBER DIRECTORY ENTRIES WHEN
                 THE STOW TABLE IS FILLED

               - IF RESTORE MEMBERS:
                  - CHECK IF MEMBER EXISTS AND CHECK SECURITY
               - CONNECT TO NEW MEMBER (INLMMCON)
               - PROCESS ALL MEMBER RECORDS (INLMPUTR)
               - DISCONNECT FROM MEMBER (INLMMDIS)
               - IF STOW TABLE FILLED:
                  - CALL MEMBSTOW (STOW MEMBERS
                                   AND INITIALIZE STOW TABLE)

ENTRY-POINTS = AS FOLLOWS

    INLPREM2:    RESTORE MEMBER

INPUT =          INPUT TAPE POSITIONED AT MEMBER HEADER
                 OF THE MEMBER TO RESTORE

OUTPUT =         RETURN CODES
                   0 ... MEMBER CREATED AND RESTORED
                        (IF A MEMBER WITH THE SAME NAME EXIST
                         IT IS REPLACED)
                        NOTE HOWEVER THAT THE RESTORING OF THE
                        MEMBER IS NOT COMPLETE BEFORE THE STOW
                        MEMBER DIRECTORY IS PERFORMED. THIS IS
                        DONE WHEN THE STOW TABLE IS FILLED OR
                        THE RESTORE FOR THE CURRENT SUBLIBRARY
                        IS COMPLETE.
                    4 ... INTERNAL OR EXTERNAL ERROR
                           WITH FEEDBACK CODE
                   32 ... MEMBER HAS TO BE SKIPPED DUE TO
                           (FOR EXAMPLE DUE TO
                            A SECURITY VIOLATION OR BECAUSE
                            THE TARGET MEMBER IS LOCKED)

 ENTRY POINT:    SAME AS MODULE NAME

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLARESL

```
MODULE INLPRESL
DESCRIPTIVE NAME: RESTORE SUBLIBRARIES

FUNCTION =     RESTORE SUBLIBRARIES

               THIS FUNCTION RESTORES SUBLIBRARIES
               CONTAINED ON BACKUP TAPES CREATED
               WITH THE BACKUP COMMAND

               - THE LIBRARY FILENAME AND THE SUBLIBRARY NAME
                 FOR EACH SUBLIBRARY FOUND ON THE BACKUP TAPE
                 ARE COMPARED WITH THE SUBLIBRARY NAMES IN THE
                 SUBLIBRARY SPECIFICATION LIST ADDRESSED BY
                 INLCCMDP. THE SUBLIBRARY IS RESTORED IF FOUND
                 IN THE LIST.
               - RESTSUBL: RESTORE SUBLIBRARY
                 - IF THE TARGET LIBRARY NAME CHANGES:
                    - ADD TEMP LIBRARY ENTRY TO THE
                      LIBRARIAN TABLES (LBRACCES)
                 - RESTORE SUBLIB (INLPRES2)

       NOTE: IF THE TARGET LIBRARY IS UNIQUELY USED (I.E NO
             ACCESS IS ACTIVE FOR ANOTHER TASK, THE LIBRARY
             IS NOT ON A VOLUME WHICH IS SHARED ACROSS
             CPUS AND NO LIBDEF IS SPECIFIED) THE LIBRARY
             IS LOCKED (THE "NO ACCESS ALLOWED" FLAG IS
             TURNED ON) TO IMPROVE THE PERFORMANCE OF
             THE RESTORE PROCESS.

INPUT =        - SUBLIBRARY-SPECIFICATION-LIST
               - BACKUP FILE ON INPUT TAPE CONTAINING
                 LIBRARIES OR SUBLIBRARIES

OUTPUT =       EACH SUBLIBRARY FOUND ON TAPE AND CONTAINED
               IN THE SPECIFICATION-LIST IS RESTORED TO
               THE TARGET SUBLIBRARY

               INFORMATION ON SYSLOG/SYSLST INDICATING
               WHETHER THE SUBLIBRARIES IN THE SUBLIBRARY-
               SPECIFICATION-LIST ARE RESTORED OR NOT

               RETURN CODES
                 0 ... NORMAL EXIT :
                          CRGRETCD = 0 :
                              RESTORE SUBLIBRARY SUCCESSFUL
                          CRGRETCD = 8 AND MESSAGE:
                              IF A SUBLIBRARY IS SKIPPED
                 4 ... INTERNAL OR EXTERNAL ERROR
                        WITH FEEDBACK CODE

 ENTRY POINT:  SAME AS MODULE NAME
```

INLARES2

```
MODULE INLPRES2
DESCRIPTIVE NAME: RESTORE SUBLIBRARY

FUNCTION =       RESTORE SUBLIBRARY

                - IF NOT STAND-ALONE ENVIRONMENT:

                  - IF RESTORING SUBLIBRARIES:
                    ADD TEMP. LIB/SUBLIB ENTRY TO THE LIBRARIAN
                    TABLES (LBRACCES) WITH DEFINE(OLD)
                    TO CHECK IF SUBLIBRARY EXISTS ALREADY

                  - IF RESTORING SUBLIBRARIES AND
                    THE SUBLIBRARY EXISTS:
                    - DELETE THE SUBLIBRARY USING MACRO INLMSTOW
                      (I.E EMPTY THE SUBLIBRARY)
                    - CONNECT THE SUBLIBRARY (INLMSCON)
                  - ELSE:
                    - ADD TEMP. LIB/SUBLIB ENTRY TO THE LIBRARIAN
                      TABLES (LBRACCES) WITH DEFINE(NEW)
                    - CONNECT THE NEW SUBLIBRARY (INLMSCON)

                - ELSE (STAND-ALONE ENVIRONMENT):
                  - CONNECT THE NEW SUBLIBRARY (INLMSCON)

                - DO FOR ALL MEMBERS FOUND FOR THIS SUBLIBRARY
                  ON TAPE:
                   - RESTORE MEMBER (INLPREM2)
                - STOW MEMBERS CONTAINED STILL IN THE STOW
                  TABLE FOR THE SUBLIBRARY (INLPMSTW)
                - DISCONNECT FROM LIBRARY (INLMLDIS)

INPUT =          INPUT TAPE POSITIONED AT SUBLIBRARY HEADER
                 OF SUBLIBRARY TO RESTORE

OUTPUT =         SUBLIBRARY CREATED AND RESTORED
                 (IF A SUBLIBRARY WITH THE SAME NAME EXIST
                  IT IS DELETED, I.E. ALL MEMBERS OF THAT
                  SUBLIBRARY ARE DELETED AND THE RESTORE
                  OF THE SUBLIBRARY WORKS ON THE EMPTY
                  'OLD' SUBLIBRARY.)

                 RETURN CODES
                   0 ... SUBLIBRARY CREATED AND RESTORED
                   4 ... INTERNAL OR EXTERNAL ERROR
                         WITH FEEDBACK CODE
                  32 ... SUBLIBRARY HAS TO BE SKIPPED
                         (FOR EXAMPLE DUE TO A SECURITY VIOLATION)

 ENTRY POINT:    SAME AS MODULE NAME

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLARIPL

```
MODULE INLARIPL
DESCRIPTIVE NAME: REPLACE IPL-PHASES

FUNCTION =       IF CALLED FROM INLPDEF THEN LOAD
                 THE IPL-BOOTSTRAP PHASES;
                 IF CALLED FROM INLPRELB THEN GET THEM
                 IN TWO DISTINCT BUFFERS;
                 WRITE THE PHASES ON PREDEFINED POSITIONS
                 ON SYSTEM DISK (EITHER CKD OR FBA)

INPUT =          - PTR TO BUFFER AREA TO LOAD PHASES (DEFINE)
                 - OR PTR TO ALREADY LOADED PHASES (RESTORE)

OUTPUT =         MESSAGES

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLARTI

```
MODULE INLPRTI
DESCRIPTIVE NAME: TAPE INPUT ROUTINE (VSE) FOR RESTORE OLDLIB'S

FUNCTION =

    THIS MODULE PROVIDES A TAPE INPUT INTERFACE FOR THE
         'RESTORE OLD LIBRARY'-COMMAND OF THE VSE LIBRARIAN
         FOR BACKUP TAPES OF VSE AF RELEASE 3.5 OR EARLIER.
         SYSTEM MACROS ARE USED FOR OPEN/CLOSE OF THE TAPE
         AND FOR HANDLING OF ALTERNATE TAPE ASSIGNMENTS.
         FEATURES ARE DOUBLE OR SINGLE BUFFERING
         RECFORM=UNDEF
         OPERATION CAN BE EITHER NEXT EXCP BEFORE WAIT FOR
         PREVIOUS, OR (FOR "OLD" R34 TAPES) WAIT BEFORE EXCP.

         THE DIFFERENCE BETWEEN "OLD" AND "NEW" TAPES IS,
         THAT THE FORMER HAVE ONE TAPEMARK AT THE END OF A
         VOLUME, AND THE LATTER TWO. EXCP BEFORE WAIT IS
         THEREFORE NOT POSSIBLE FOR OLD TAPES.

ENTRY-POINTS = AS FOLLOWS

    INLPRTIO: OPEN THE TAPE FILE

        THE CALLER FIRST OPENS THE TAPE FILE VIA ENTRY INLPRTIO
         PROVIDING THE ADDRESSES OF TWO BUFFERS, THE MAXIMUM
         RECORD LENGTH (WHICH IS THE SIZE OF THE BUFFERS),
         A SET OF FLAGS TELLING WHETHER,
           A. TAPES HAVE STANDARD LABELS OR ARE UNLABELED
           B. TAPES ARE TO BE REWOUND AT OPEN, REWOUND AND
              UNLOADED AT CLOSE OR REWOUND AT NEITHER OPEN
              NOR CLOSE
           C. CCB OR IORB - APPLIES TO DOS/VS CASE ONLY
         AND THE SYSTEM LOGICAL UNIT NUMBER TO BE USED.
         INLPRTIO MODIFIES THE DTFMT FOR FILE UIN ACCORDING TO
         THE USER'S SPECIFICATIONS AND OPENS THE FILE.

INPUT =     P1      -   POINTER TO PRIMARY BUFFER
            P2      -   POINTER TO ALTERNATE BUFFER
            TICNTRL -   CONTROL INFORMATION

    INLPRTI: INPUT REQUEST EMPLOYING DOUBLE BUFFERING

        DOUBLE BUFFERING MUST BE INITIATED BY A CALL OF INLPRTIL
        INDICATING WHICH MODE OF OPERATION IS REQUESTED, I.E.
        WHETHER WAIT SHOULD BE ISSUED BEFORE OR AFTER NEXT EXCP
```

```
INPUT =   BUFFADDR -
              THE ADDRESS OF THE BUFFER INTO WHICH THE
              DATA WILL BE READ
          DATALEN -
              A 4-BYTE FIELD IN WHICH THE LENGTH
              OF THE DATA READ IS RETURNED

OUTPUT =  DATALEN     LENGTH  OF TAPE RECORD READ IN
          BUFFADDR    ADDRESS OF BUFFER INTO WHICH HAS BEEN READ
          RETCODE     0 RECORD HAS BEEN READ SUCCESSFULLY
                      2 RECORD LONGER THAN BLKSIZE HAS BEEN READ
                      4 END-OF-FILE HAS BEEN ENCOUNTERED

     INLPRTI1: INPUT REQUEST USING A SINGLE BUFFER

          THE CALLER NEEDS PROVIDE ONLY ONE PARAMETER IN WHICH
          HE RECEIVES THE LENGTH OF THE DATA READ.
          THE DATA READ IS RETURNED IN THE FIRST OF THE TWO
          BUFFERS SPECIFIED FOR THE INLPRTIO CALL.

INPUT =   DATALEN -
              A 4-BYTE FIELD IN WHICH THE LENGTH
              OF THE DATA READ IS RETURNED

OUTPUT =  DATALEN     LENGTH  OF TAPE RECORD READ IN
          RETURN CODES
                   0    RECORD HAS BEEN READ SUCCESSFULLY
                   2    RECORD LONGER THAN BLKSIZE HAS BEEN READ
                   4    END-OF-FILE HAS BEEN ENCOUNTERED

     INLPRTIL: TAPE INPUT CONTROL FUNCTIONS

INPUT =    ACTION -   CONTROL ACTION REQUIRED:
             - '07'X    REWIND TAPE TO LOAD POINT.
             - '01'X    OLD R34 TAPE (ONE TRAILING TM ON VOLUME)
             - '02'X    NEW TAPE (TWO TRAILING TM ON VOLUME)
             - 'FF'X    WAIT - PROVIDES A PAUSE DURING DOUBLE
                        BUFFERED INPUT, MAKING BOTH BUFFERS
                        AVAILABLE TO THE CALLER
             - '3F'X    FORWARD SPACING OVER A FILE AND/OR TM.

     INLPRTIR: ALLOWS AN INVALID TAPE TO BE REJECTED
                 (APPLIES TO UNLABELED TAPES ONLY)

OUTPUT =  DATALEN - LENGTH OF RECORD READ FROM CORRECT VOLUME
                    MOUNTED IN PLACE OF INVALID ONE.
          BUFFADDR    ADDRESS OF BUFFER INTO WHICH HAS BEEN READ
          RETURN CODES (SEE ENTRY POINT INLPRTI OR INLPRTI1)

     INLPRTIC: CLOSE THE INPUT FILE PROPERLY AFTER DOING ALL
               INPUT
```

INLASTOW

```
MODULE INLPSTOW

DESCRIPTIVE NAME: MAINTAIN SUBLIBRARY AND MEMBER INDEX

FUNCTION =        UPDATE (ADD/DELETE/RENAME) SUBLIBRARY AND
                  MEMBER INDEX ON A LIBRARY (CONSISTENTLY FOR
                  CONCURRENT READ REQUESTS).
                  UPDATE SECOND LEVEL DIRECTORY (SLD) FOR
                  PHASES.
                  UPDATE SYSTEM DIRECTORY LIST (SDL) BY
                  CALLING INLPSDL.

          ===> MF=ADD:
          ADD ALL DIRECTORY ENTRIES GIVEN IN THE STOW TABLE
          TO THE DIRECTORY ACCORDING TO THE COLLATING
          SEQUENCE AND UPDATE THE HIGHER LEVEL INDEX.
          - THE TIMESTAMP IS ONLY SET IF LRPLDATE=FALSE.
          IF LRPLDATE=TRUE, THE TIMESTAMPS AS GIVEN WITH THE
          DIRECTORY ENTRIES ARE LEFT UNCHANGED.
          - THE INSERTION OF THE ENTRIES AND THE SETTING OF THE
          TIMESTAMP IS DONE ACCORDING TO THE OLDNTRY PARAMETER:
          NOREPLACE & MBR EXISTS: NO INSERTION AT ALL
          NOREPLACE & MBR DOES NOT EXIST: SET ORIGINATION TIME
          REPLACE & MBR EXISTS: SET UPDATE TIME
          REPLACE & MBR DOES NOT EXIST: SET ORIGINATION TIME.
          WHENEVER THE ORIGINATION TIMESTAMP IS SET THE UPDATE
          TIMESTAMP IS CLEARED.
          - SECURITY CHECK DEPENDS ON THE EXISTENCE OF THE
          MEMBER IN THE DIRECTORY: IF THE MEMBER ALREADY EXISTS
          AN UPDATE ACCESS RIGHT FOR THE MEMBER IS REQUIRED,
          OTHERWISE AN ALTER RIGHT. ANY SECURITY VIOLATION IS
          FLAGGED IN THE ARGUMENT (LARGSEC). A MEMBER WHICH
          IS MSHP-CONTROLLED CAN ONLY BE REPLACED BY MSHP.
          - IF A MEMBER IS REPLACED, ITS DIRECTORY ENTRY IS
          UPDATED IN-PLACE AND - FOR PURPOSE=UPDATE (MEMBER
          SPACE IS AFFECTED) - THE SPACE OF THE OLD VERSION IS
          RECLAIMED.
          - IF A MEMBER CANNOT BE ADDED (BECAUSE OF SECURITY
          VIOLATION, BECAUSE AN OLD VERSION SHOULD NOT BE
          REPLACED OR THE MEMBER IS LOCKED), THE ALREADY
          WRITTEN MEMBER SPACE IS FREED (ONLY FOR PURPOSE=
          UPDATE). THIS IS REPORTED IN THE ARGUMENT (LARGDEL).
          - IF A MEMBER HAS BEEN SUCCESSFULLY ADDED TO THE
          DIRECTORY THIS WILL BE REPORTED IN THE ARGUMENT
          (LARGCAT).
          - THE UPDATING OF THE HIGHER LEVEL IS DONE BOTTOM-UP.
          - FINALLY THE SUBLIBRARY DIRECTORY ENTRY IS UPDATED.
```

```
          ===> MF=DELETE:
          ALL MEMBERS GIVEN IN THE STOWTABLE ARE DELETED FROM
          MEMBER DIRECTORY OF THE CONNECTED SUBLIBRARY AND THE
          HIGHER LEVEL INDEX IS UPDATED BOTTOM-UP. FINALLY, THE
          SUBLIBRARY DESCRIPTOR IS UPDATED.
          - WHILE PROCESSING THE HIGHER LEVEL INDEX, AN INDEX
          SHRINKAGE CAN OCCUR (USALLY, WHEN THE CURRENT INDEX
          LEVEL (THE LEVEL BEING PROCESSED) NEEDS NO MORE THAN
          TWO LIBRARY BLOCKS, ALL LEVELS ABOVE THIS LEVEL ARE
          THROWN AWAY.
          - MEMBER ALTER RIGHT IS REQUIRED.

          ===> MF=RENAME:
          ALL ENTRIES IN THE STOWTABLE REQUESTED FOR RENAME ARE
          PROCESSED.
          - ALL ENTRIES IN THE STOWTABLE WITH AN ODD INDEX ARE
          REGARDED AS THE ENTRIES TO BE RENAMED AND ALL ENTRIES
          WITH AN EVEN INDEX ARE REGARDED AS THE CORRESPONDING
          NEW KEYS (NO SORT OF THE STOWTABLE HAS TO BE DONE).
          - IF A RENAME IS DONE, THE OLD KEY EXISTS AND THE
          NEW NAME DOES NOT EXIST WITHIN THE SUBLIBRARY.
          - WHEN A RENAME TO TYPE "PHASE"  IS REQUESTED
          THE CONCERNED MEMBER MUST HAVE THE CHARACTERISTICS OF
          A PHASE.
          - A RENAME IS DONE BY FIRST PERFORMING A COMPLETE
          ADD OF THE NEW ENTRY IN THE MEMBER INDEX AND THEN
          DELETING THE OLD ENTRY. SO, TEMPORARILY TWO DIRECTORY
          ENTRIES POINTING TO THE SAME MEMBER SPACE ARE IN THE
          SUBLIBRARY.
          - BY THIS ALGORITHM THE MEMBER INDEX CAN GROW AND
          SHRINK, THUS RESERVING LIBRARY BLOCKS WHICH ARE
          AFTERWARDS IN THE SPACE RECLAMATION CHAIN.
          - FINALLY, THE SUBLIBRARY DESCRIPTOR IS UPDATED.
          - FOR A RENAME THE CALLER MUST HAVE THE ALTER MEMBER
          RIGHT FOR THE OLD NAME AS WELL AS FOR THE NEW ONE.

          ===> MF=PURGE:
          THE MEMBER SPACE OF THE ENTRIES GIVEN IN THE STOW
          TABLE IS IMMEDIATELY FREED.

INPUT =           STOWTABLE
                  LIBRARY REQUEST PARAMETER LIST (LRPL)
                  LIBRARY DESCRIPTOR
                  SUBLIBRARY DIRECTORY
                  MEMBER INDEX

OUTPUT =          UPDATED LIBRARY INDICES
                  MESSAGES, RETURN CODES
                         0   NORMAL RETURN
                         4   FUNCTION (PARTIALLY) FAILED,
                             SEE RETURN INFORMATION IN STOWTAB
                        12   LIBRARY IS FULL
                        16   EXTERNAL ERROR WITH FEEDBACK CODE
                        20   INTERNAL ERROR WITH FEEDBACK CODE
                        32   SECURITY VIOLATION
```

```
         INFORMATION IN STOW TABLE FOR RETURNCODE 4:
              LARGCAT(N) = TRUE -> N.TH ENTRY STOWED SUCCESSFUL
              LARGCAT(N) = FALSE-> N.TH ENTRY NOT STOWED,
              LARGDEL(N) = TRUE -> MEMBERSPACE HAS BEEN FREED
              LARGSEC(N) = TRUE -> SECURITY VIOLATION FOR N.TH
                                   ENTRY
              LARGFND(N) = TRUE -> N.TH ENTRY EXISTED ALREADY
              LARGDLCK(N)= TRUE -> N.TH ENTRY WAS LOCKED

ENTRY POINT:     SAME AS MODULE NAME
```

INLASTOX

```
     MODULE INLPSTOX
     DESCRIPTIVE NAME:
               EXTERNAL SUBROUTINES FOR INLPSTOW AND
               INLPSLIB AND OTHER SERVICES

     ENTRY-POINTS:
                  UPDLBDIR  - UPDATE LDES
                  SORT - CALLED BY STOW
                  PARMCHCK - CALLED BY STOW
                  TEST     - CALLED BY STOW
                  TRACE    - CALLED BY STOW
                  PMSG286  - CALLED BY STOW
                  INLPSARG - FILL ARGUMENT CALLED BY MCON, BLDL
                  INLXMRS  - BUILD RESOURCE NAME FOR MEMBER SPACE
                             BELONGS TO DATATYPE LOCK MEMBER SPACE
                  DELLVIF - DELETES LOCK VIF
                             CALLED BY STOW, COPY
                             MODULE ALSO COPIED TO LEVEL 3 PHASE

     LINKAGE:
                  REGISTER 1 CONTAINS ADDRESS OF LIBRARY REQUEST
                    PARAMETER LIST (LRPL).
                  REGISTER 13 POINTS TO A SAVEAREA.
                    (THESE AREAS HAVE TO BE ALIGNED ON DWORD BOUNDARY.
                  REGISTER 14 IS RETURN REGISTER.
                  REGISTER 15 CONTAINS ENTRY-POINT.

     INPUT:      STOWTABLE.
                 LRPL
                 LAMB
                 SUBLIBRARY DIRECTORY
                 MEMBER INDEX
                 LACB, SACB

     OUTPUT:         UPDATED
                         MEMBER INDEX OF SUBLIBRARY,
                         SUBLIBRARY DIRECTORY,
                         LIBRARY DESCRIPTOR,
                         SYSTEM DIRECTORY LIST,
                         SECOND LEVEL DIRECTORY FOR PHASES,
                         LIBRARY POINTER TABLE.

EXIT:           VIA REGISTER 14 WITH RETURNCODE

     CALLED BY:    INLPSTOW, INLPSLIB AND OTHER SERVICES
```

INLASTOY

```
MODULE INLPSTOY
DESCRIPTIVE NAME:
          SUBROUTINES FOR DIRECTORY UPDATE FROM
          INLASTOW  -- RESULT OF MODULE SPLITTING

MODULE TYPE:  REENTRANT (READ-ONLY) PROCEDURE IN SVA

ENTRY-POINT:  MODULE BEGIN

LINKAGE:
              REGISTER 1 CONTAINS ADDRESS OF LIBRARY REQUEST
                PARAMETER LIST (LRPL).
              REGISTER 13 POINTS TO A SAVEAREA.
                (THESE AREAS HAVE TO BE ALIGNED ON DWORD BOUNDARY.
              REGISTER 14 IS RETURN REGISTER.
              REGISTER 15 CONTAINS ENTRY-POINT.

INPUT:      STOWTABLE.
            LRPL
            LAMB
            SUBLIBRARY DIRECTORY
            MEMBER INDEX
            LACB, SACB

OUTPUT:         DEPENDS ON FUNCTION

EXIT:          VIA REGISTER 14 WITH RETURNCODE 0 ,4, 12, 16, 20, 32
               THE RETURN CODE IS IN REGISTER 15.

               RETURN CODE: 0   NORMAL RETURN
                            4   FUNCTION (PARTIALLY) FAILED,
                                SEE RETURN INFORMATION IN STOWTABLE
                           12   LIBRARY IS FULL
                           16   EXTERNAL ERROR WITH FEEDBACK CODE
                           20   INTERNAL ERROR WITH FEEDBACK CODE
                           32   SECURITY VIOLATION

CALLED BY:    INLPSTOW
```

INLATI

```
 MODULE INLPTI  (VSE)
 DESCRIPTIVE NAME: TAPE INPUT ROUTINE

 FUNCTION =

     THIS MODULE PROVIDES A TAPE INPUT INTERFACE
          FOR THE 'RESTORE'-COMMAND OF THE VSE LIBRARIAN
          PROGRAM AND THE 'STAND ALONE RESTORE'.
          SYSTEM MACROS ARE USED FOR OPEN/CLOSE OF THE TAPE
          AND FOR HANDLING OF ALTERNATE TAPE ASSIGNMENTS.
          FEATURES ARE MULTIPLE OR SINGLE BUFFERING
          AND RECFORM=UNDEF.

 ENTRY-POINTS = AS FOLLOWS

     INLPTIO:  OPEN THE TAPE FILE

INPUT =   BDBPTR,TICNTRL
          THE CALLING ROUTINE (INLPREST OR INLXREST)
          PROVIDES THE ADDRESS OF A POINTER (BDBPTR)
          TO A BUFFER POOL (THE BUFFERS OF THIS POOL ARE
          ARE CHAINED TOGETHER IN A LOOP AND BDBPTR
          POINTS TO ONE BUFFER CONTROL BLOCK (BDB) OF THIS
          THIS BUFFER POOL)
          AND A SET OF FLAGS (TICNTRL) TELLING WHETHER,
            A. TAPES ARE TO BE REWOUND AT OPEN, REWOUND AND
               UNLOADED AT CLOSE OR REWOUND AT NEITHER OPEN
               NOR CLOSE
            B. CCB OR IORB - APPLIES TO THE VSE CASE ONLY
          AND THE SYSTEM LOGICAL UNIT NUMBER TO BE USED.
          INLPTIO MODIFIES THE DTFMT FOR FILE UIN ACCORDING TO
          THE USER'S SPECIFICATIONS AND OPENS THE FILE.
          UNLABELED BACKUP TAPES ARE SUPPORTED ONLY FOR
          VSE/AF VERSION 2 RELEASE 1.

     INLPTI:  INPUT REQUEST USING MULTIPLE BUFFERING

INPUT =   BDBPTR  -
              THE ADDRESS OF THE BUFFER DEFINITION BLOCK (BDB)
              POINTING TO THE BUFFER INTO WHICH THE DATA
              WILL BE READ
          DATALEN -
              A 4-BYTE FIELD IN WHICH THE LENGTH
              OF THE DATA READ IS RETURNED
```

```
          IT IS THE PRINCIPAL CONCEPT OF THIS TAPE INPUT
          FUNCTION THAT TAPE OPERATIONS ARE SCHEDULED AS
          SOON AS POSSIBLE AND WAITED FOR AS LATE AS
          POSSIBLE. AS MANY TAPE REQUESTS AS POSSIBLE
          ARE PLACED INTO THE CHANNEL QUEUE BEFORE A
          TAPE WAIT IS ISSUED. THE MORE REQUESTS RESIDING
          IN THE CHANNEL QUEUE, THE LONGER THE TIME AVAILABLE
          UNTIL THE NEXT TAPE REQUEST MUST BE FORWARDED
          TO THE SUPERVISOR IN ORDER TO MAINTAIN STREAMING
          FOR THE 8809 TAPE DEVICE.
          IF IT HAPPENS THAT ALL TAPE BUFFERS ARE ALREADY
          FILLED WHEN A CALL TO INLPTI IS ISSUED NO TAPE
          I/O IS SCHEDULED ANY MORE UNTIL ALL BUFFERS OF
          THE BUFFER POOL ARE PROCESSED IN ORDER TO
          HAVE AT LEAST 'PARTIAL' STREAMING.

OUTPUT =  DATALEN     LENGTH  OF TAPE RECORD READ IN
          BDBPTR      ADDRESS OF BUFFER DEFINITION BLOCK POINTING
                      TO THE BUFFER INTO WHICH HAS BEEN READ
          RETURN CODES
                   0    RECORD HAS BEEN READ SUCCESSFULLY
                   2    RECORD LONGER THAN BLKSIZE HAS BEEN READ
                   4    TAPE MARK READ AND NOT EOF

     INLPTI1: INPUT REQUEST USING A SINGLE BUFFER

INPUT =   BDBPTR  -
              THE ADDRESS OF THE BUFFER DEFINITION BLOCK (BDB)
              POINTING TO THE BUFFER INTO WHICH THE DATA
              WILL BE READ
          DATALEN -
              A 4-BYTE FIELD IN WHICH HE GETS RETURNED
              THE LENGTH OF THE DATA READ

OUTPUT =  DATALEN     LENGTH  OF TAPE RECORD READ IN
          RETURN CODES
                   0    RECORD HAS BEEN READ SUCCESSFULLY
                   2    RECORD LONGER THAN BLKSIZE HAS BEEN READ
                   4    TAPE MARK READ

     INLPTIL: TAPE INPUT CONTROL FUNCTIONS

INPUT =    ACTION -   CONTROL ACTION REQUIRED:
             - '07'X    REWIND TAPE TO LOAD POINT.
             - '3F'X    FORWARD SPACING OVER A FILE AND/OR TM.
             - '2F'X    BACKWARD SPACING OVER A FILE AND/OR TM.

     INLPTIR: ALLOWS AN INVALID TAPE TO BE REJECTED
                 (APPLIES TO UNLABELED TAPES ONLY)
```

```
OUTPUT =  DATALEN - LENGTH OF RECORD READ FROM CORRECT VOLUME
                    MOUNTED IN PLACE OF INVALID ONE.
          BDBPTR  - ADDRESS OF BUFFER DEFINITION BLOCK POINTING
                    TO THE BUFFER INTO WHICH THE DATA IS READ
          RETURN CODES (SEE ENTRY POINT INLPTI)

     INLPTIC: CLOSE THE INPUT FILE PROPERLY AFTER DOING ALL
              INPUT

     FOR THE FORMAT OF THE RESTORE TAPE SEE MODULE
     INLPBKUP (BACKUP MAIN MODULE)
```

INLATO

```
 MODULE INLPTO
 DESCRIPTIVE NAME:  TAPE OUTPUT ROUTINE (VSE) FOR BACKUP

 FUNCTION =

     THIS MODULE PROVIDES A TAPE OUTPUT INTERFACE
          FOR THE 'BACKUP'-COMMAND OF THE VSE LIBRARIAN
          PROGRAM.
          SYSTEM MACROS ARE USED FOR OPEN/CLOSE OF THE TAPE
          AND FOR HANDLING OF ALTERNATE TAPE ASSIGNMENTS.

          FEATURES ARE MULTIPLE OR SINGLE BUFFERING
          AND RECFORM=UNDEF.

 ENTRY-POINTS = AS FOLLOWS

     INLPTOO:  OPEN THE TAPE FILE

INPUT =   BDBPTR,TOCNTRL
          THE CALLING ROUTINE (INLPBKUP)
          PROVIDES THE ADDRESS OF A POINTER (BDBPTR)
          TO A BUFFER POOL (THE BUFFERS OF THIS POOL
          ARE CHAINED TOGETHER IN A LOOP AND BDBPTR
          POINTS TO ONE BUFFER CONTROL BLOCK (BDB) OF
          THIS BUFFER POOL)
          AND  A FIELD (TOCNTRL)
          WHICH CONTAINS THE FOLLOWING INFORMATION:

           NOTM     TRUE  - NO LEADING TAPE MARK IS TO BE WRITTEN
                    FALSE - A LEADING TAPE MARK IS WRITTEN
                           (SIGNIFICANT ONLY FOR UNLABELED CASE)
           NOREWND  TRUE  - NO REWIND (CLOSE/OPEN)
                    FALSE - REWIND TO LOAD POINT (CLOSE/OPEN)
           USEIORB  TRUE  - THE I/O BUFFERS HAVE BEEN FIXED.
                    FALSE - IORB CANNOT BE USED
           TOSYSNUM THE LOGICAL UNIT TO BE USED (IN CCB FORMAT)
           TOWRDRCP ADDR(WRITE DUMMY RECORDS CHANNEL PROGRAM)

          INLPTOO MODIFIES THE DTFMT FOR FILE UIN ACCORDING TO
          THE THE FLAGS 'LABEL' AND 'NOTM' AND OPENS THE FILE.
          UNLABELLED BACKUP TAPES ARE SUPPORTED ONLY WITH
          VSE/AF VERSION 2 RELEASE 1.

     INLPTO: OUTPUT REQUEST USING MULTIPLE BUFFERING
```

```
INPUT =   BDBPTR  -
             THE ADDRESS OF THE BUFFER DEFINITION BLOCK (BDB)
             POINTING TO THE BUFFER FROM WHICH THE DATA
             WILL BE WRITTEN TO TAPE AND
          DATALEN -
             A 4-BYTE FIELD  WHICH HAS TO CONTAIN THE
             LENGTH OF THE DATA TO BE WRITTEN.

          IT IS THE PRINCIPAL CONCEPT OF THIS TAPE OUTPUT
          FUNCTION THAT TAPE OPERATIONS ARE SCHEDULED AS
          SOON AS POSSIBLE AND WAITED FOR AS LATE AS
          POSSIBLE.
          BASICALLY, THE TAPE OUTPUT ALGORITHM FOR BACKUP
          WORKS AS FOLLOWS:
          - INITIALLY ALL BUFFERS OF THE BUFFER POOL ARE
            FILLED WITH THE DATA TO BE WRITTEN.
          - THEN THE TAPE I/O FOR ALL THESE BUFFERS IS
            SCHEDULED BUT NOT WAITED FOR.
          - AFTER THE INITIALIZATION PROCESS, TAPE
            OPERATIONS ARE SCHEDULED AS SOON AS POSSIBLE
            AND WAITED FOR AS LATE AS POSSIBLE.
            SO AS MANY TAPE REQUEST AS POSSIBLE ARE
            PLACED INTO THE CHANNEL QUEUE BEFORE A TAPE
            WAIT IS ISSUED. THE MORE REQUESTS RESIDING IN
            THE CHANNEL QUEUE, THE LONGER THE TIME AVAILABLE
            UNTIL THE NEXT TAPE REQUEST MUST BE FORWARDED
            TO THE SUPERVISOR IN ORDER TO MAINTAIN STREAMING
            FOR THE 8809 TAPE DEVICE.
            IF IT HAPPENS THAT ALL TAPE BUFFERS ARE ALREADY
            EMPTY (I.E. THEY ARE ALL ALREADY WRITTEN TO TAPE)
            WHEN A CALL TO INLPTO IS ISSUED NO TAPE- I/O
            IS SCHEDULED ANY MORE UNTIL ALL BUFFERS OF THE
            BUFFER POOL ARE FILLED AGAIN IN ORDER TO
            HAVE AT LEAST 'PARTIAL' STREAMING.

            THIS ALGORITHM RESULTS IN AN EXCEPTIONAL SITUATION
            IF THE WAIT INDICATES THAT THE REFLECTIVE SPOT
            WAS SENSED  (VIA UNIT EXCEPTION IN THE CCB)
            AT THE END OF THE TAPE. THERE IS, HOWEVER, SPACE
            ON THE TAPE FOR THE N EXTRA RECORDS REQUESTED TO
            BE WRITTEN BY THE 'EARLY' EXCP'S.
            OCCURRENCE OF UNIT EXCEPTION IS NOTED IN THE FLAG
             'DOFEOV'.

            UPON THE NEXT CALL OF INLPTO, IF DOFEOV IS ON, THE
            FEOV ACTION IS TAKEN, WHICH INCLUDES WRITING
            OF N DUMMY RECORDS AND AN EXTRA TAPE MARK
            AT THE END OF AN UNLABELED TAPE
              (SEE THE CORRESPONDING REQUIREMENT OF THE
               TAPE INPUT ROUTINE)

    INLPTO1: OUTPUT REQUEST USING A SINGLE BUFFER
```

```
INPUT =    BDBPTR  -
              THE ADDRESS OF THE BUFFER DEFINITION BLOCK (BDB)
              POINTING TO THE BUFFER FROM WHICH THE DATA
              WILL BE WRITTEN TO TAPE AND
           DATALEN -
              A 4-BYTE FIELD  WHICH HAS TO CONTAIN THE
              LENGTH OF THE DATA TO BE WRITTEN.

              END OF TAPE IS INDICATED BY UNIT EXCEPTION -
              RETURN CODE 8 IS SET AND DOFEOV SWITCH IS
              TURNED ON AND ON THE NEXT ENTRY TO INLPTO1
              OR INLPTO THE FEOV ACTION IS TAKEN.

     INLPTOL: TAPE OUTPUT CONTROL FUNCTIONS

INPUT =   ACTION -   CONTROL ACTION REQUIRED:
            - '07'X  CAUSES THE TAPE TO BE REWOUND TO THE
                     LOAD POINT
            - '1F'X  CAUSES A TM TO BE WRITTEN
            - '2F'X  BACKWARD SPACING OVER A FILE AND/OR TM.

     INLPTOC: WRITE DUMMY RECORDS AND A TAPE MARK AT THE END
              OF A BACKUP FILE
              CLOSE THE OUTPUT FILE PROPERLY AFTER DOING ALL
              OUTPUT

     INLPTOCH:  TAPE OUTPUT REQUEST FOR A CCW-CHAIN

INPUT =   CCWPTR  - POINTER TO CCW-CHAIN
                    (WRITING RECORDS TO TAPE).
OUTPUT =  RETURN CODES
             0    - REQUEST SUCCESSFUL EXECUTED
             8    - EOV CONDITION HAS OCCURRED

     FOR THE FORMAT OF THE RESTORE TAPE SEE MODULE
     INLPBKUP (BACKUP MAIN MODULE)
```

INLPACC

```
MODULE INLPACC
DESCRIPTIVE NAME: ACCESS COMMAND PROCESSOR

FUNCTION =       PROCESS ACCESS STATEMENT AND DO SEMANTIC
                 CHECKS:
                 - LABEL INITACC:
                    DO PREPARATORY WORK FOR ACCESS PROCESSING
                 - LABEL DISPACC:
                    PRINT/DISPLAY ACCESS INFO ON REQUEST
                 - LABEL BLDACC:
                    BUILD ACCESS INFO ENTRY VIA LBRACCES
                 - LABEL FINACC:
                    FINISH ACCESS PROCESSING AND RETURN

INPUT =          LIBRARIAN COMMUNICATION REGION INLCCOMR
                 PARSED COMMAND CONTROL BLOCK INLCCMDP

OUTPUT =         ACCESS INFORMATION LINE ON REQUEST
                 RETURN CODE 0  - SUCCESSFUL COMPLETION
                 RETURN CODE 8  - ANY FAILURE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPAREA

```
MODULE INLPAREA
DESCRIPTIVE NAME: DEFINE CONTROL BLOCKS AND DATA AREAS

FUNCTION =       THIS MODULE PROVIDES THE SPACE FOR THE
                 FOLLOWING CONTROL BLOCKS AND DATA AREAS:
                     INLCCOMR  - LIBR. COMMUNICATION REGION
                     INLCCMDP  - PARSED COMMAND BLOCK
                     INLCSCAN  - SCAN CONTROL BLOCK
                     INLSAVSC  - SCAN CONTROL BLOCK SAVE AREA
                     INLSAVIO  - SAVED INPUT I/O-AREA
                     INLDTFLS  - SYSLST DTF
                     INLDTFPC  - SYSPCH DTF
                     INLDTFCN  - SYSLOG DTF
                     INLIPTIO  - SYSIPT I/O-AREA
                     INLLSTIO  - SYSLST I/O-AREA
                     INLPCHIO  - SYSPCH I/O-AREA
                     INLLOGIO  - SYSLOG I/O-AREA
                     INLNMLIS  - NAME LIST AREA

INPUT =          NONE

OUTPUT =         NONE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPBBUF

```
MODULE INLPBBUF
DESCRIPTIVE NAME: SET UP BUFFERS AND BUFFER CONTROL BLOCKS

FUNCTION =       FOR BOTH PRIVATE AND SHARED BUFFER  AREAS
                 SET UP BUFFER CONTROL BLOCK(BUCB) AND
                 ONE HEADER (BHDR) FOR EACH DATA BUFFER.
                 INITIALIZE CHAINING OF BUFFER HEADERS,
                 ALL BHDRS IN FREE CHAIN,ALL OTHER CHAINS
                 EMPTY.

INPUT =          LRPLPTR  PTR(31)  - LRPL POINTER
                 BBUFTYPE CHAR(1)  - BUFFER TYPE REQUESTED
                                     SHAREDB | PRIVARB

OUTPUT =         RETC000           - NORMAL END
                 RETC016           - MINIMUM STORAGE SIZE
                                     NOT AVAILABLE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPBKHF

```
FUNCTION =      THIS MODULE WRITES THE HISTORY FILE BLOCKS
                PROVIDED BY MSHP ONTO THE OUTPUT TAPE.


INPUT =         FUNCTION CODES
                  1 ... WRITE BLOCK TO TAPE
                  2 ... ERROR DURING MSHP PROCESSING:
                            CLOSE OUTPUT TAPE

                ADDRESS AND LENGTH OF HISTORY FILE BLOCK


OUTPUT =        HISTORY FILE BLOCK ON THE OUTPUT TAPE

                RETURN CODES
                  0 ... NORMAL EXIT :
                            WRITE OF HISTORY FILE
                            BLOCK SUCCESSFUL
                  8 ... END OF VOLUME HAS OCCURRED:
                            BACKUP TERMINATED
                  16... INVALID FUNCTION CODE
```

INLPBKLB

```
MODULE INLPBKLB
DESCRIPTIVE NAME: BACKUP LIBRARIES

FUNCTION =  BACKUP LIBRARIES

            - DUMP BACKUP FILE HEADER
            - FOR EACH LIBRARY IN LIBRARY NAME LIST ADDRESSED
              BY INLCCMDP WHICH EXISTS IN THE SYSTEM:
              PROCEDURE  BKUPLIB:
                - LOCK ALL SUBLIBRARIES
                - BACKUP LIBRARY HEADER
                - BACKUP IPL PHASES IF IT IS A SYSTEM
                  LIBRARY (INLPDIPL)
                - BACKUP ALL SUBLIBRARIES FOR THIS LIBRARY:
                     - ACCESS SUBLIBRARY
                     - BACKUP SUBLIBRARY (INLPBKS2)
                     - UNLOCK SUBLIBRARY

INPUT =          LIBRARY-SPECIFICATION-LIST (LS) IN
                 (INLCCMDP - PARSED COMMAND CONTROL BLOCK )

OUTPUT =         ONE BACKUP-FILE ON TAPE CONTAINING THE
                 LIBRARIES FOUND IN THE SYSTEM

                 INFORMATION ON SYSLOG/SYSLST INDICATING
                 WHETHER THE LIBRARY IS DUMPED OR NOT

                 RETURN CODES
                   0 ... NORMAL EXIT :
                             CRGRETCD = 0 :
                               BACKUP SUCCESSFUL
                             CRGRETCD = 8 AND MESSAGE:
                               IF A LIBRARY,SUBLIBRARY OR MEMBER
                               IS SKIPPED
                   4 ... AN INTERNAL OR EXTERNAL ERROR
                         WITH FEEDBACK CODE

 ENTRY POINT:    SAME AS MODULE NAME

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPBKMB

```
MODULE INLPBKMB
DESCRIPTIVE NAME: BACKUP MEMBERS

FUNCTION =  BACKUP MEMBERS

            - DUMP BACKUP FILE HEADER
            - PROCESS  NAME LIST ADDRESSED BY INLCCMDP :
              - BACKUP MEMBER (BKUPMEMB):
                - IF LIB/SUBLIB CHANGE:
                  - IF NOT FIRST PATH:
                    DISCONNECT FROM LAST TARGET LIBRARY
                  - ADD TEMP. LIB/SUBLIB ENTRY TO THE
                    LIBRARIAN TABLES (LBRACCES)
                  - CONNECT THE NEW SUBLIBRARY (INLMSCON)
                - CHECK IF SPECIFICATION IS GENERIC (CHECKGEN)
                - IF GENERIC:
                  - CHECK READ AUTHORITY FOR SUBLIBRARY
                  - BUILD MEMBER NAME LIST FOR THE GENERIC
                    SPECIFICATION (BLDLBKUP) AND CALL FOR EACH
                    MEMBER IN LIST BACKUP MEMBER (INLPBKM2)
                  ELSE
                  - BACKUP MEMBER (INLPBKM2)
            - WRITE DUMMY MEMBER RECORD TO TAPE BUFFER AND
              GET NEW TAPE BUFFER
            - AT END OF PROCESSING: DISCONNECT FROM LIBRARY
              LAST USED (INLMLDIS)

INPUT =          MEMBER-SPECIFICATION-LIST (MS)
                 (INLCCMDP - PARSED COMMAND CONTROL BLOCK)

OUTPUT =         ONE BACKUP-FILE ON TAPE CONTAINING THE
                 MEMBERS FOUND IN THE SYSTEM

                 INFORMATION ON SYSLOG/SYSLST INDICATING
                 WHETHER THE MEMBER IS DUMPED OR NOT

                 RETURN CODES
                   0 ... NORMAL EXIT :
                            CRGRETCD = 0 :
                              BACKUP SUCCESSFUL
                            CRGRETCD = 8 AND MESSAGE:
                              IF A MEMBER IS SKIPPED
                   4 ... AN INTERNAL OR EXTERNAL ERROR
                         WITH FEEDBACK CODE

 ENTRY POINT:    SAME AS MODULE NAME

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPBKM2

```
MODULE INLPBKM2
DESCRIPTIVE NAME: BACKUP MEMBER

FUNCTION =   THE PURPOSE OF THIS MODULE IS TO PERFORM
             THE BACKUP OF A MEMBER TO TAPE.

             - CONNECT TO MEMBER (INLPMCON)
             - BUILD AND DUMP MEMBER HEADER (DUMPMHDR)
             - PROCESS ALL RECORDS OF MEMBER:
                 - GET MEMBER RECORDS (INLPGETR)
                 - WRITE FILLED BUFFER TO TAPE (INLPWRTP)
             - DISCONNECT FROM MEMBER (INLPMDIS)

INPUT =          ARGPTR : PTR TO ARGUMENT (MEMBER NAME,TYPE)

OUTPUT =         MEMBER WRITTEN TO TAPE OR INSERTED INTO
                 BUFFERS TO BE WRITTEN TO OUTPUT TAPE

                 RETURN CODES
                   0 ... NORMAL EXIT :
                         BACKUP SUCCESSFUL
                   4 ... AN INTERNAL OR EXTERNAL ERROR
                         WITH FEEDBACK CODE
                  32 ... MEMBER HAS BEEN SKIPPED

 ENTRY POINT:    SAME AS MODULE NAME

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPBKSA

```
MODULE INLPBKSA
DESCRIPTIVE NAME: BACKUP STAND-ALONE PROGRAMS, BACKUP HEADER
                  AND BACKUP DISK IPL PHASES

FUNCTION =       BACKUP STAND ALONE PROGRAMS, BACKUP HEADER
                 AND BACKUP DISK IPL PHASES

                 1. BACKUP THE STAND ALONE PROGRAMS:
                    ( ENTRY POINT INLPBKSA )
                    THE STAND ALONE PROGRAMS ARE RETRIEVED OUT
                    OF THE FIRST SYSTEM LIBRARY CONTAINED IN
                    THE NAME LIST OF THE LIBRARIES TO BE
                    BACKED UP.
                    IF NO SYSTEM LIBRARY IS SPECIFIED THE SA
                    PROGRAMS ARE RETRIEVED OUT OF THE IPL'D
                    SYSTEM LIBRARY.
                    AN EXCEPTION TO THIS RULE ARE THE SA DITTO
                    PHASES AND THE CUSTOMIZATION PHASE:
                    - THE SA DITTO PHASES ARE RETRIEVED (IF NOT
                      CONTAINED IN THE SYSTEM SUBLIBRARY
                      IJSYSRX.SYSLIB) OUT OF THE SUBLIBRARY
                      PRD1.BASE .
                    - IF THE CUSTTABLE OPERAND IS SPECIFIED
                      THE CUSTOMIZATION PHASE IS RETRIEVED OUT
                      OF THE SPECIFIED SUBLIBRARY, ELSE THAT
                      THAT PHASE IS RETRIEVED OUT OF THE
                      THE SAME SYSTEM SUBLIBRARY AS THE OTHER
                      STAND ALONE PROGRAMS.

                 2. BACKUP THE DISK IPL PHASES:
                    ( ENTRY POINT INLPDIPL )
                    THE DISK IPL PHASES ARE RETRIEVED OUT
                    OF THE SYSTEM LIBRARY TO BE BACKED UP.

                 3. BACKUP THE HEADER FILE:
                    ( ENTRY POINT INLPBKHD )
                    THE HEADER FILE MUST BE OF CARD IMAGE
                    FORMAT
                    (I.E. OF RECORD TYPE FIXED AND RECLEN 80)

 ENTRY POINT:    INLPBKSA

PURPOSE =        BACKUP THE STAND ALONE PROGRAMS

INPUT =          FILENAME =   NAME OF THE SYSTEM LIBRARY

 ENTRY POINT:    INLPDIPL

PURPOSE =        BACKUP THE DISK IPL PHASES

INPUT =          FILENAME =   NAME OF THE SYSTEM LIBRARY
```

```
 ENTRY POINT:    INLPBKHD

PURPOSE =        BACKUP THE HEADER FILE

INPUT =          INLCLIB  =   NAME OF THE LIBRARY
                 INLCSUB  =   NAME OF THE SUBLIBRARY
                              CONTAINING THE HEADER FILE
                 INLCMEM  =   MEMBER NAME OF THE HEADER FILE
                 INLCTYP  =   MEMBER TYPE OF THE HEADER FILE

OUTPUT =         RETURN CODES
                   0 ... NORMAL EXIT :
                            BACKUP SUCCESSFUL
                            CRGRETCD = 8 AND MESSAGE :
                              IF A SA-PHASE OR DISK-IPL-PHASE
                              IS SKIPPED
                   4 ... AN INTERNAL OR EXTERNAL ERROR
                         WITH FEEDBACK CODE
                         OR A MESSAGE
```

INLPBKSL

```
MODULE INLPBKSL
DESCRIPTIVE NAME: BACKUP SUBLIBRARIES

FUNCTION =  BACKUP SUBLIBRARIES

            - DUMP BACKUP FILE ID
            - BACKUP HISTORY FILE (IF REQUESTED)
            - DUMP BACKUP FILE HEADER
            - PROCESS  NAME LIST ADDRESSED BY INLCCMDP:
                     - ACCESS SUBLIBRARY
                     - GET SUBLIBRARY INDEX ENTRY
                     - BACKUP SUBLIBRARY (INLPBKS2)

INPUT =          SUBLIBRARY-SPECIFICATION-LIST (SS)
                 (INLCCMDP - PARSED COMMAND CONTROL BLOCK)

OUTPUT =         ONE BACKUP-FILE ON TAPE CONTAINING THE
                 SUBLIBRARIES FOUND IN THE SYSTEM

                 INFORMATION ON SYSLOG/SYSLST INDICATING
                 WHETHER THE SUBLIBRARY IS DUMPED OR NOT

                 RETURN CODES
                   0 ... NORMAL EXIT:
                            CRGRETCD = 0:
                              BACKUP SUCCESSFUL
                            CRGRETCD = 8 AND MESSAGE:
                              IF A SUBLIBRARY OR MEMBER
                              IS SKIPPED
                   4 ... AN INTERNAL OR EXTERNAL ERROR
                         WITH FEEDBACK CODE

 ENTRY POINT:    SAME AS MODULE NAME

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPBKS2

```
MODULE INLPBKS2
DESCRIPTIVE NAME: BACKUP SUBLIBRARY

FUNCTION =   THE PURPOSE OF THIS MODULE IS TO PERFORM
             THE BACKUP OF A SUBLIBRARY TO TAPE.

             THE SUBLIBRARY IS ALREADY LOCKED SHARED BY THE
             CALLING MODULE INLPBKLB(BACKUP LIBRARIES) OR
             INLPBKSL(BACKUP SUBLIBRARIES) ( I.E. NO UPDATE IS
             ALLOWED DURING THE BACKUP OF THE SUBLIBRARY)

             - CONNECT TO SUBLIBRARY
             - BUILD AND DUMP SUBLIBRARY TAPE HEADER (DUMPSHDR)
             - PROCESS ALL MEMBERS OF SUBLIBRAY:
                CALL INLPBKM2 (BACKUP MEMBER)
             - DISCONNECT FROM SUBLIBRARY
             - WRITE DUMMY MEMBER RECORD TO TAPE TO CLOSE
                 BACKUP SUBLIBRARY PROCESSING (WRTEDMBR)

ENTRY-POINTS = AS FOLLOWS

    INLPBKS2 :  BACKUP SUBLIBRARY

INPUT =          SUBLIBRARY ACCESSED

OUTPUT =         SUBLIBRARY ON TAPE
                 (FOR TAPE LAYOUT SEE MODULE INLPBKUP)

                 RETURN CODES
                   0 ... NORMAL EXIT:
                            BACKUP SUBLIBRARY SUCCESSFUL
                            CRGRETCD = 8 AND A MESSAGE:
                              IF A MEMBER IN ERROR IS SKIPPED
                   4 ... AN INTERNAL OR EXTERNAL ERROR
                         WITH FEEDBACK CODE
                  32 ... SUBLIBRARY IS SKIPPED

    INLPBKLK :  LOCK/UNLOCK SUBLIBRARY

             CALLED BY MODULE INLPBKLB(BACKUP LIBRARY) OR
             INLPBKSL(BACKUP SUBLIBRARY)

INPUT =          RESOURCE NAME OF SUBLIBRARY
                 LOCK OR UNLOCK FUNCTION REQUESTED

OUTPUT=          RETURN CODES
                   0 ... NORMAL EXIT:
                         LOCK/UNLOCK SUCCESSFUL
```

INLPBKUP

```
FUNCTION =      THIS PROGRAM CREATES A BACKUP FILE, ON TAPE,
                CONTAINING: LIBRARIES OR
                            SUBLIBRARIES

                BACKUP MAIN MODULE:

                - INITIALIZATION OF BACKUP
                - ALLOCATE SPACE FOR CONTROL BLOCKS AND BUFFERS
                - OPEN TAPE
                - BACKUP HEADER FILE (INLPBKHD) IF REQUESTED
                - BACKUP STANDALONE PROGRAMS (INLPBKSA) IF
                  RESTORE=STANDALONE IS SPECIFIED:
                      THE STANDALONE-PROGRAMS ARE RETRIEVED
                      OUT OF THE FIRST SYSTEM LIBRARY
                      FOUND IN THE SPECIFICATION LIST.
                      (NAMES IJSYSR1 TO IJSYSR9  AND IJSYSRS)
                      ELSE THE SA-PROGRAMS ARE RETRIEVED
                      OUT OF THE IPL'D SYSTEM LIBRARY
                     AN EXCEPTION TO THIS RULE ARE THE SA DITTO
                     PHASES AND THE CUSTOMIZATION PHASE:
                     - THE SA DITTO PHASES ARE RETRIEVED (IF NOT
                       CONTAINED IN THE SYSTEM SUBLIBRARY
                       IJSYSRX.SYSLIB) OUT OF THE SUBLIBRARY
                       PRD1.BASE .
                     - IF THE CUSTTABLE OPERAND IS SPECIFIED
                       THE CUSTOMIZATION PHASE IS RETRIEVED OUT
                       OF THE SPECIFIED SUBLIBRARY, ELSE THAT
                       THAT PHASE IS RETRIEVED OUT OF THE
                       THE SAME SYSTEM SUBLIBRARY AS THE OTHER
                       STAND ALONE PROGRAMS.

                - DUMP BACKUP-FILE-ID
                - BACKUP HISTORY FILE IF REQUESTED (LOADPTFB)
                - CALLS DEPENDING ON THE SPECIFICATION LIST :
                    BACKUP LIBRARIES (INLPBKLB) OR
                    BACKUP SUBLIBRARIES (INLPBKSL) OR
                    BACKUP MEMBERS (INLPBKMB)
                - CLOSE BACKUP FILE AND RETURN TO PARSER


INPUT =         LIBRARY-SPECIFICATION-LIST OR
                SUBLIBRARY-SPECIFICATION-LIST OR
                MEMBER-SPECIFICATION-LIST :
                (INLCCMDP - PARSED COMMAND CONTROL BLOCK)


OUTPUT =        - INFORMATION ON SYSLOG/SYSLST INDICATING
                  WHETHER THE BACKUP OF THE LIBRARIES,
                  SUBLIBRARIES OR MEMBERS CONTAINED IN THE
                  SPECIFICATION-LIST IS PERFORMED OR NOT
```

```
                    - DEPENDING ON WHETHER TAPELABEL=LABELNAME IS
                      SPECIFIED OR NOT THE BACKUP COMMAND CREATES
                      A LABELED OR UNLABELED TAPE.
                      THE OUTPUT PRODUCED BY ONE BACKUP COMMAND
                      IS A SINGLE-VOLUME OR MULTI-VOLUME FILE
                      CONSISTING OF 3 SUBFILES (THE HEADER FILE,
                      THE HISTORY FILE AND THE BACKUP FILE
                      CONTAINING LIBRARIES OR SUBLIBRARIES) WHICH
                      ARE SEPARATED BY TAPEMARKS AND DO NOT
                      CONTAIN THEIR OWN SET OF LABELS.

                      IF  RESTORE = STANDALONE IS SPECIFIED:
                         A REWIND OF THE OUTPUT TAPE IS
                         PERFORMED, STANDALONE CONTROL PROGRAMS
                         ARE WRITTEN TO THE 1ST SUBFILE AND THE
                         STANDALONE UTILITIES ARE WRITTEN TO
                         AN ADDITIONAL SUBFILE.
                      (SEE FORMAT OF THE BACKUP TAPE ON NEXT PAGE)

                      NOTE: IF STD LABELS ARE WRITTEN ONTO A
                         STANDALONE TAPE THE LABELS HAVE TO BE
                         SKIPPED BEFORE THE TAPE CAN BE IPL'D.

                    RETURN CODES
                      0 ... NORMAL EXIT :
                                CRGSAVRC = 0 :
                                  BACKUP SUCCESSFUL
                                CRGSAVRC = 8 :
                                  IF A LIBRARY,SUBLIBRARY OR MEMBER
                                  IS SKIPPED

REMARKS =

  FORMAT OF THE BACKUP TAPE:
```

```
UNLABELED TAPE:

     HEADER  (OPTIONAL - FOR THE STANDALONE ENVIRONMENT AN
                         IBM SUPPLIED HEADER IS REQUIRED)
     STAND-ALONE PROGRAMS (OPTIONAL):
             PRESENT IF RESTORE=STANDALONE SPECIFIED:
                    COMMON PHASES NEEDED TO IPL THE
                    STAND-ALONE ENVIRONMENT:
                     $$A$IPL2   1ST TAPE BOOTSTRAP
                     $$A$PLBT   2ND TAPE BOOTSTRAP
                     $$A$IPLR   3RD IPL MODULE
                     $$ACISS    CONSOLE SUPPORT PHASE
                     $$A$CDL0   COMMUNICATION DEVICE LIST
                     $$A$SUPX   ESA SUPERVISOR
                     $IJBDSPU   ESA STANDARD DISPATCHER
                     $IJBTFCH   TAPE FETCH
                     $IJBSPDT   INTEGRATED CONSOLE SUPPORT
                     $IJBCRT    CRT SUPPORT
                     $IJBXDEF   NLS DEFINITIONS
                     $IJBCSIW   CONSOLE ROUTER WORK AREA
                     $IJBCSIO   CONSOLE ROUTER
                     $IPLRT2    IPL CMD PROCESSORS
                     $SENSDEV   SENSES ALL DEVICES
                     $SVASA     SA SVA LOAD LIST
                     $IJBSSM    SPACE AND PARTIT. ALLOCATION
                     $INTVIRT   VIRT. STORAGE INITIALIZATION
                     INLPSDL    BUILDS SDL, LOADS SVA
           ( This file is called the "Stand-alone IPL file")
                    -- TAPEMARK --

                    IN ALPHABETICAL ORDER ALL THE SVA
                    PROGRAMS NEEDED IN THE STAND-ALONE
                    ENVIRONMENT AS SPECIFIED IN THE SA SVA
                    LOAD LIST ($SVASA) INCLUDING THE
                    STAND-ALONE UTILITIES : FCOPY, RESTORE,
                    ICKDSF AND DITTO.
           ( This file is called the "Stand-alone Utility file" )
  -- TAPEMARK --
     BACKUP-FILE-ID RECORD
     HISTORY-FILE  (OPTIONAL)
  -- TAPEMARK --
     BACKUP-FILE
  -- TAPEMARK --

     HEADER  (OPTIONAL)           2ND AND FOLLOWING
  -- TAPEMARK --                  BACKUP (OPTIONAL)
     BACKUP-FILE-ID RECORD               "
     HISTORY-FILE  (OPTIONAL)            "
  -- TAPEMARK --                         "
     BACKUP-FILE                         "
  -- TAPEMARK --                         "

  -- TAPEMARK --
     END-OF-BACKUP RECORD
  -- TAPEMARK --
  -- TAPEMARK --
```

```
LABELED TAPE:

     VOL1 LABEL
     HDR1 LABEL
  -- TAPEMARK --
     HEADER  (OPTIONAL)
  -- TAPEMARK --
     BACKUP-FILE-ID RECORD
     HISTORY-FILE  (OPTIONAL)
  -- TAPEMARK --
     BACKUP-FILE
  -- TAPEMARK --
     EOF1 LABEL
  -- TAPEMARK --


     HDR1 LABEL                   2ND AND FOLLOWING
  -- TAPEMARK --                  BACKUP (OPTIONAL)
     HEADER  (OPTIONAL)                  "
  -- TAPEMARK --                         "
     BACKUP-FILE-ID RECORD               "
     HISTORY-FILE  (OPTIONAL)            "
  -- TAPEMARK --                         "
     BACKUP-FILE                         "
  -- TAPEMARK --                         "
     EOF1 LABEL                          "
  -- TAPEMARK --                         "


  -- TAPEMARK --
     END-OF-BACKUP RECORD
  -- TAPEMARK --
  -- TAPEMARK --


- TAPE POSITIONING

DURING BACKUP:

NORMALLY AT START AND END OF BACKUP NO REWIND OF THE
OUTPUT TAPE IS PERFORMED, I.E. :
IF DUMPING OF THE SA-PROGRAMS IS NOT REQUESTED
( RESTORE = ONLINE ) NO REWIND OF THE OUTPUT TAPE
IS DONE AT START OF A BACKUP COMMAND.
ONLY IF DUMPING OF THE SA-PROGRAMS (RESTORE = STANDALONE)
IS REQUESTED A REWIND OF THE OUTPUT TAPE IS PERFORMED AT
START OF A BACKUP COMMAND.

AT THE END OF A BACKUP COMMAND THE TAPE IS POSITIONED
IN FRONT OF THE TAPEMARK PRECEDING THE END-OF-BACKUP
RECORD.
```

```
NOTE :
 IF FOR A LABELED TAPE ONLY THE SA PROGRAMS ARE
 WRITTEN, NO TRAILER LABEL IS WRITTEN AT END OF BACKUP.


DURING RESTORE:

AT START AND END OF RESTORE NO REWIND OF THE INPUT
TAPE IS PERFORMED WITH THE FOLLOWING EXCEPTION:
IF FOR SA-RESTORE THE INPUT TAPE IS DIFFERENT FROM THE
IPL'D ONE THE INPUT TAPE IS REWOUND.

AT THE END OF A RESTORE COMMAND THE TAPE IS POSITIONED
 - FOR AN UNLABELED BACKUP TAPE
   OR A SELECTIVE RESTORE:
       WITHIN THE CURRENT BACKUP FILE AFTER THE RESTORED
       OBJECT
 - FOR A NON-SELECTIVE RESTORE
   AND A LABELED BACKUP TAPE:
       AFTER THE TAPEMARK WHICH FOLLOWS THE EOF1 LABEL



- HEADER AND STANDALONE PROGRAMS

IF HEADER = L.S.M.T IS SPECIFIED FIRST THE HEADER
IS WRITTEN TO TAPE.
IF THE HEADER IS NOT SPECIFIED AN EMPTY FILE IS
WRITTEN TO TAPE IF RESTORE = ONLINE IS REQUESTED.
IF DUMPING OF THE SA-PROGRAMS IS REQUESTED THE
HEADER AND THE SA-PROGRAMS ARE WRITTEN AS TWO FILES,
THE FIRST FILE CONSISTING OF THE HEADER (OPTIONAL) AND
THE STAND-ALONE IPL AND CONTROL PROGRAMS,
THE SECOND OF ALL PROGRAMS NEEDED IN THE STAND-ALONE
ENVIRONMENT IN THE SVA AND THE STAND-ALONE UTILITIES.
(SEE TABLE PHSTAB IN ROUTINE DUMPSA IN
 MODULE INLPBKSA AND THE STAND-ALONE SVA LOAD LIST
 ($SVASA) FOR PHASE NAMES)
THE HEADER IS WRITTEN IN CARD IMAGE FORMAT TO THE TAPE

(NOTE: FOR SA RESTORE PROCESSING THE HEADER MUST START WITH
       IPL BOOTSTRAP RECORDS. OTHERWISE, THE TAPE CANNOT BE
       IPL'D AND THE HEADER HAS TO BE SKIPPED BY USER
       ACTION.
       IF THE SA RESTORE TAPE STARTS WITH STANDARD TAPE
       LABELS THESE LABELS HAVE TO BE SKIPPED BY USER
       ACTION BEFORE THE TAPE CAN BE IPL'D.)

(NOTE: IF THE SA-PROGRAMS ARE WRITTEN TO TAPE (I.E.
       RESTORE=STANDALONE IS SPECIFIED), THE SPECIFICATION OF
       LIBRARIES/SUBLIBS/MEMBERS TO BE BACKED UP IS OPTIONAL;
       I.E. IF NOT SPECIFIED NO BACKUP FILE IS CREATED.)
```

```
- BACKUP FILE :

EACH TAPE BLOCK OF THE BACKUP FILE STARTS
WITH A BLOCK HEADER (SEE DESCRIPTION OF BLKHDR)
WHICH CONTAINS THE LENGTH OF THE BLOCK, THE BLOCK COUNT
AND A DESCRIPTOR INDICATING WHETHER THE BLOCK CONTAINS
NORMAL DATA (DATA), CONTAINS THE 'BACKUP FILE HEADER'
(FHDR), IS A DUMMY BLOCK (DRID) AT THE END OF THE BACKUP
FILE OR IS AN ERROR RECORD (ERR ) (WRITTEN TO TAPE IN CASE
OF AN ABNORMAL END OF THE BACKUP COMMAND).
THE BACKUP FILE HEADER INDICATES WHETHER THE BACKUP FILE
CONTAINS LIBRARIES,ONLY SUBLIBRARIES OR ONLY MEMBERS.
EACH ITEM IN THE BACKUP FILE HAS THE SAME FORMAT: THE
FIRST 2 BYTES CONTAIN THE LENGTH OF THE ITEM FOLLOWED
BY A 4-BYTE DESCRIPTOR AND THE DATA.
THE FOLLOWING ARE THE POSSIBLE TAPE ITEMS ON THE BACKUP
FILE (IN PARENTHESES THE DESCRIPTOR AND THE DATA AREA
NAME):

    BACKUP-FILE-HEADER (LIBR,SUBL OR MEMB) - (SEE BFHDR)
    LIBRARY-HEADER             (LHDR) - (SEE LIBRHDR )
    EXTENT-RECORD              (XTNT) - (SEE EXTENTR )
    DISK IPL PHASES    (IPLK OR IPLF) - (SEE DIPLPHDR)
    SUBLIBRARY-HEADER          (SHDR) - (SEE SUBLHDR )
    MEMBER-HEADER              (MHDR) - (SEE MEMBHDR )
    MEMBER-RECORDS             (MDAT) - (SEE MEMBDAT )
    DUMMY MEMBER HEADER        (DHDR)
    END-OF-BACKUP-FILE RECORD  (EOBF) - (SEE EOBKFID )


THE LAYOUT OF THE BACKUP FILE
IS AS FOLLOWS:

    BACKUP-FILE-HEADER --
    LIBRARY-HEADER  --
      EXTENT-RECORDS --
    DISK IPL PHASES(IF A SYSTEM LIBRARY IS FOLLOWING) --
      SUBLIBRARY-HEADER  --
        MEMBER-HEADER  -- MEMBER-RECORDS --
        MEMBER-HEADER  -- MEMBER-RECORDS --
        .....
        MEMBER-HEADER  -- MEMBER-RECORDS --
        DUMMY MEMBER HEADER --
      SUBLIBRARY-HEADER  --
        MEMBER-HEADER  -- MEMBER-RECORDS --
        MEMBER-HEADER  -- MEMBER-RECORDS --
        .....
        MEMBER-HEADER  -- MEMBER-RECORDS --
        DUMMY MEMBER RECORD --
    .....
    END-OF-BACKUP-FILE RECORD--
    DUMMY BLOCK ... DUMMY BLOCK --
    TM --
```

```
  THE LIBRARY HEADER OF A SYSRES FILE IS FOLLOWED
  BY THE DISK IPL PHASES (CKD AND FBA - 2 PHASES EACH,
  SEE TABLE IPLTAB (IN MODULE INLPBKSA) FOR PHASE NAMES -
  EVERY PHASE WILL OCCUPY ONE TAPE BLOCK).


  A NEW LIBRARY OR A NEW SUBLIBRARY STARTS A NEW TAPE BLOCK.
  EACH LIBRARY IS PRECEDED BY A LIBRARY HEADER,
  EACH SUBLIBRARY IS PRECEDED BY A SUBLIBRARY HEADER AND
  EACH MEMBER IS PRECEDED BY A MEMBER HEADER. A MEMBER
  MAY OVERFLOW INTO THE NEXT TAPE BLOCKS, BUT THE MEMBER
  HEADER WILL NOT BE SPLIT ACROSS BLOCKS.
  AFTER THE LAST LIBRARY OR LAST SUBLIBRARY DUMPED
  AN END-OF-BACKUP FILE RECORD AND N DUMMY RECORDS
  ARE WRITTEN.
  (THE NUMBER N OF DUMMY RECORDS WRITTEN IS CONTAINED
   IN THE BACKUP-FILE-ID RECORD :
   IN THIS VERSION N=8 WHICH MEANS THAT DURING
   RESTORE UP TO 8 BUFFERS MAY BE USED)

  MULTI-VOLUME BACKUP TAPES AND ALTERNATE TAPE ASSIGNMENT
  ARE SUPPORTED.

  IF AN END OF VOLUME CONDITION OCCURS DURING BACKUP AN
  END-OF-VOLUME RECORD (DESCR='EOV ' IN BLKHDR) IS WRITTEN
  AFTER THE BACKUP FILE ON TAPE (FOR UNLABELED AND FOR
  LABELED TAPES). THIS EOV RECORD CONTAINS THE BLOCK NUMBER
  OF THE NEXT TAPE BLOCK OF THE BACKUP FILE. THIS BLOCK
  NUMBER IS NEEDED TO INSURE FOR UNLABELED TAPE THAT THE
  CORRECT CONTINUATION TAPE IS MOUNTED WHEN SCANNING THE
  TAPE FOR A SPECIFIC BACKUP-FILE-ID DURING RESTORE.



FORMAT OF A 'PID-V2-STACKED' BACKUP TAPE FOR VSE/SP:


  FOR DISTRIBUTION OF A VSE/SP SYSTEM SEVERAL UNLABELED
  BACKUP TAPES AS DESCRIBED ABOVE (INCLUDING THE
  END-OF-BACKUP RECORD) ARE STACKED ON A SINGLE TAPE.
  (I. E. PID-V2-STACKED TAPES ARE ALWAYS UNLABELED.)
  A PID-V2-STACKED TAPE STARTS WITH A START-OF-STACKED-TAPE
  IDENTIFICATION AND ENDS WITH AN END-OF-STACKED-TAPE
  IDENTIFICATION AS FOLLOWS:


    -- TAPEMARK --
       START-OF-PID-V2-STACKED-TAPE RECORD  - (SEE SPIDRECD)
    -- TAPEMARK --
    -- TAPEMARK --
```

```
     -- CONTENTS OF A BACKUP TAPE CONTAINING ONLY A SINGLE
        PRODUCT (INCLUDING THE END-OF-BACKUP RECORD FOLLOWED
        BY THE TWO TAPEMARKS)

       .....
       .....

     -- CONTENTS OF A BACKUP TAPE CONTAINING ONLY A SINGLE
        PRODUCT (INCLUDING THE END-OF-BACKUP RECORD FOLLOWED
        BY THE TWO TAPEMARKS)



     -- TAPEMARK --
        END-OF-PID-V2-STACKED-TAPE RECORD    - (SEE EOSPRECD)
     -- TAPEMARK --
     -- TAPEMARK --



   WHEN THE RESTORE COMMAND FINDS THE START-OF-STACKED-TAPE
   IDENTIFICATION ON THE BACKUP TAPE, THE TAPE IS SCANNED
   (IF REQUESTED VIA THE ID OPERAND) ACROSS THE END-OF-BACKUP
   RECORDS UNTIL THE END-OF-STACKED-TAPE IDENTIFICATION
   IS FOUND.
   IN THIS CASE THE RESTORE COMMAND POSITIONS THE BACKUP
   TAPE IN FRONT OF THE TAPEMARK PRECEDING THE
   END-OF-PID-V2-STACKED TAPE RECORD.



ENTRY POINT:    SAME AS MODULE NAME

  LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPBLDL

```
MODULE INLPBLDL
DESCRIPTIVE NAME: BUILD LIST OF DIRECTORY ENTRIES

FUNCTION =        BUILD IN-CORE LIST OF SUBLIBRARY OR MEMBER
                  DIRECTORY ENTRIES OUT OF (SUB-)LIBRARY CHAIN.
   =====> THIS PROCEDURE IS CALLED BY MACRO INLMBLDL.
          THE FUNCTION GETS A LIST OF ARGUMENTS (IN STOW TABLE
          FORMAT) AS INPUT AND PASSES BACK AN IDENTIFICATION
          ABOUT THE EXISTENCE OF THE GIVEN LIBRARY OBJECTS IN
          THE SEARCH CHAIN,TOGETHER WITH ADDITIONAL DESCRIPTOR
          INFORMATIONS.
          - THE FUNCTION WORKS FOR:
          1. A LIST OF SUBLIBRARY NAMES OR A LIST OF MEMBER
          SPECIFICATIONS, WHICH WILL BE FLAGGED AS EXISTENT OR
          NOT (INFORM=KEY) OR FILLED UP WITH ADDITIONAL SUBLIB
          OR MEMBER DESCRIPTOR INFORMATION (INFORM=ENTRY),
          2. A LIST OF FULL SPECIFIED NAMES OR ONE GENERIC
          SPECIFICATION, WHICH WILL BE EXPANDED TO A LIST OF
          ENTRIES FOUND IN THE FIRST LIBRARY OR SUBLIBRARY
          WHERE THE GENERIC SPECIFICATION MATCHES.
          - FOR SUBLIBRARIES, THE LOCKING RESOURCE NAMES WILL
          BE BUILT WITH REQUEST INFORM=RESOURCE.
          - IF NO MATCH IS FOUND IN THE SEARCHING CHAIN, RETURN
          CODE 12 IS PASSED BACK.
          - IF THE INFORMATION AREA (PARAMETER STOWLEN) IS
          EXCEEDED BY THE ACTUAL INFORMATION, RETURN CODE 8 IS
          PASSED BACK. NOW IT IS POSSIBLE TO GIVE A CONTINUED
          REQUEST (PARAMETER CONTINUE) TO GET THE REMAINING
          INFORMATION.
          - IF SOME OF THE ARGUMENTS IN A LIST DO   NOT OCCUR
          IN THE SEARCHING CHAIN OR DO NOT PASS THE SECURITY
          CHECK THE CORRESPONDING ENTRIES ARE FLAGGED AND A
          RETURN CODE OF 4 IS PASSED BACK TO THE INVOKER.
          - THE SECURITY CHECK FOR A FULLY SPECIFIED ARGUMENT
          IS FOR READ AUTHORIZATION.
          - THE SECURITY CHECK FOR A GENERIC SUBLIBRARY
          SPECIFICATION IS DONE FOR READ AUTHORIZATION ON
          LIBRARY LEVEL, FOR A GENERIC MEMBER SPECFICATION FOR
          READ AUTHORIZATION ON THE SUBLIBRARY.

INPUT =           LIBRARY REQUEST PARAMETER LIST (INLCLRPL)
                  STOW TABLE WITH SEARCH ARGUMENT SPECIFICATION
                  (GENERIC OR NON-GENERIC)
```

```
OUTPUT =          LIST OF DIRECTORY ENTRIES
                  MESSAGES, RETURN CODES:
          0 = REQUEST HAS BEEN HONORED
          4 = SOME OF THE ARGUMENTS WERE NOT FOUND IN THE GIVEN
              CHAIN AND THEIR STOW ENTRIES ARE FLAGGED.
              (SEE MACRO INLMBLDL).
          8 = THE AREA SUPPLIED FOR THE STOW TABLE WAS NOT
              SUFFICIENT AND THE REQUEST COULD NOT BE COMPLETED.
              A NONGENERIC REQUEST CAN BE CONTINUED BY
              SUPPLYING THE ENTRIES THAT WERE NOT PROCESSED.
              A GENERIC REQUEST CAN BE CONTINUED BY SUPPLYING
              THE ORIGINAL ARGUMENT AS THE FIRST ENTRY OF THE
              STOW TABLE AND THE LAST RETURNED ARGUMENT AS THE
              SECOND ENTRY ALONG WITH THE KEYWORD 'CONTINUED'.
         12 = NO ENTRY FOUND
         16 = EXTERNAL ERROR WITH FEEDBACK CODE
         20 = INTERNAL ERROR WITH FEEDBACK CODE
         32 = SECURITY VIOLATION

 ENTRY POINT:     SAME AS MODULE NAME
```

INLPBSPA

```
MODULE INLPBSPA
DESCRIPTIVE NAME:  INITIALIZE LIBRARY DISK SPACE

FUNCTION =       DISK SPACE, ALLOCATED IN UNITS OF ONE OR MORE
                 EXTENTS, IS INITIALIZED:
                   1. THE CKD SPACE IS FORMATTED WITH RECORDS
                      OF 1K LENGTH
                   2. FOR EACH EXTENT, CREATE A BIT MAP INITIA-
                      LISED TO BINARY ZERO  FOR EACH FREE LB
                   3. FOR EACH EXTENT, CREATE AN EXTENT DESCRI-
                      PTOR
                   4. FOR THE LIBRARY, CREATE A LIBRARY SPACE
                      DESCRIPTOR (PRBA=1)
                 FOR SECONDARY ALLOCATION IN VSAM SPACE, INITI-
                 ALIZE NEW EXTENTS AND UPDATE SPACE DESCRIPTOR

INPUT =          LRPLPTR  PTR(31) -LRPL POINTER
                 BSPATYPE CHAR(1) -REQUEST TYPE
                                   FORMATL FORMAT NEW LIBRARY
                                   FORMATE FORMAT NEW EXTENT
                 FOLLOWING PARAMETERS USED ONLY FOR VSAM SPACE:
                 BSPAPTR  PTR(31) -EDTE PTR, FOR NEW EXTENT
                 BSPABMCN BIN(31) -CURRENT EXTENTS COUNT,FOR
                                   NEW EXTENT

OUTPUT =         BSPABMCN BIN(31) -BIT MAP LBS COUNT
                 RETC000          -NORMAL RETURN

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPCAT

```
MODULE INLPCAT
DESCRIPTIVE NAME: CATALOG COMMAND PROCESSOR

FUNCTION =       PROCESS CATALOG STATEMENT AND DO SEMANTIC
                 CHECKS:
                 - LABEL INITCAT:
                    DO PREPARATORY WORK FOR CATALOG PROCESSING
                    AND SEMANTIC CHECKING
                 - LABEL LIBCAT:
                    FIND THE LIBRARY/SUBLIBRARY IN WHICH TO
                    CATALOG THE MEMBER
                 - LABEL CHECKCAT:
                    CHECK REPLACE OPTION, MSHP SECURITY FLAG
                 - LABEL SELCAT:
                    CONNECT THE NEW MEMBER AND SELECT THE
                    CATALOGUING PROCEDURE DEPENDING ON THE
                    MEMBER TYPE
                 - LABEL SOURCAT
                    CATALOG MEMBERS OF A SOURCE-TYPE
                 - LABEL USPROCAT
                    CATALOG MEMBERS OF A USER-TYPE OR OF TYPE
                    OBJ, PROC
                 - LABEL FINCAT:
                    UPDATE THE DIRECTORY ENTRY AND WRITE IT,
                    FINISH CATALOG PROCESSING AND RETURN

INPUT =          LIBRARIAN COMMUNICATION REGION INLCCOMR
                 PARSED COMMAND CONTROL BLOCK INLCCMDP

OUTPUT =         RETURN CODE 0 - SUCCESSFUL COMPLETION
                 RETURN CODE 4 - MESSAGE L031I
                 RETURN CODE 8 - ANY OTHER FAILURE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPCHAN

```
MODULE INLPCHAN
DESCRIPTIVE NAME: CHANGE COMMAND PROCESSOR

FUNCTION =       PROCESS CHANGE STATEMENT
                 - LABEL INITCHA:
                    DO PREPARATORY WORK FOR CHANGE PROCESSING
                 - LABEL INLPCHAS:
                    CHANGE SUBLIBRARY ATTRIBUTE
                 - LABEL FINCHA:
                    FINISH CHANGE PROCESSING AND RETURN

INPUT =          LIBRARIAN COMMUNICATION REGION INLCCOMR
                 PARSED COMMAND CONTROL BLOCK INLCCMDP

OUTPUT =         RETURN CODE 0  - SUCCESSFUL COMPLETION
                 RETURN CODE 4  - MESSAGE L105I
                 RETURN CODE 8  - ANY OTHER FAILURE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPCLK

```
MODULE INLPCLK                                                    00600052
DESCRIPTIVE NAME: MEMBER DIRECTORY LOCKING                        00700052

                                                                  01350052
FUNCTION =                                                        01400052
                - INLPCLCK: DIRECTORY LOCKS MEMBER                01450052
                - INLPCULK: DIRECTORY UNLOCKS MEMBER              01500052

                                                                  01550052
INPUT =         LOCK:  ARGUMENT  (LRPLARG)                        01600052
                       LOCKID    (LRPLKID)                        01600053
                UNLOCK: STOWTABLE                                 01600054
                        LOCKID    (LRPLKID)                       01600055

                                                                  01650052
OUTPUT =        RETURN CODE (R15)                                 01700052
                    0 MEMBER(S) SUCCESSFULLY LOCKED/UNLOCKED      01700053
                    2 MEMBER(S) WERE ALREADY LOCKED/UNLOCKED      01700054
                    4 FUNCTION PARTIALLY SUCCESSFUL (UNLOCK)      01700055
                    8 NO MEMBER FOUND                             01700056
                   12 MEMBER WAS LOCKED WITH OTHER LOCKID         01700057
                   16 EXTERNAL ERROR WITH FEEDBACK CODE           01700058
                   20 EXTERNAL ERROR WITH FEEDBACK CODE           01700059
                   32 SECURITY VIOLATION                          01700060
                                                                  01750052
                                                                  01850052
 ENTRY POINT:                                                     01900052
                         INLPCLCK, INLPCULK                       01950052

                                                                  02000052
EXT.REFERENCES  INLPSDTL                                          02050052
                INLPXQNM                                          02100052
                INLPXMSG                                          02150052
                DELLVIF                                           02200052

                                                                  02250052
MACRO CALLS     INLMFIND                                          02350052
                INLMSTOW                                          02400052
                INLMLDIS                                          02450052
                INLMBLDL                                          02500052
                GENDTL                                            02550052
                LOCK                                              02600052
```

INLPCMPR

```
MODULE INLPCMPR
DESCRIPTIVE NAME: DECOMPRESS LOGICAL RECORD

FUNCTION =       DECOMPRESS RECORDS OF A GIVEN LENGTH

INPUT =          POINTER TO PARAMETER LIST CONTAINING THE
                 INPUT/OUTPUT AREAS AND THE LENGTH FIELDS
                 INPUT AREA CONTAINS COMPRESSED RECORD

OUTPUT =         DECOMPRESSED LOGICAL RECORD AND ITS LENGTH

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPCOMM

```
MODULE INLPCOMM
DESCRIPTIVE NAME: COMMON MESSAGE PREPARATION ROUTINES

FUNCTION =        PROVIDES THE INTERFACE TO THE MESSAGE
                  GENERATION MODULE INLPDIAG.
                  AT ENTRY POINT
                  - INLPXRCD: A HEX-VALUED RETURN CODE IS
                        TRANSLATED INTO PRINTABLE CHARACTERS.
                  - INLPXMSG: THE I/F TO INLPDIAG IS BUILT
                        AND INLPDIAG IS INVOKED.
                  - INLPXPRM: THE MESSAGE L152 (ERROR IN
                        PARAMETER LIST) IS PREPARED AND GIVEN.
                  - INLPXSYS: THE MESSAGE L151 (SYSTEM ERROR)
                        IS PREPARED AND GIVEN.
                  - INLPXLIB: THE MESSAGE L157 (DEFECT IN
                        LIBRARY) IS PREPARED AND GIVEN.
                  - INLPXVIS: THE MESSAGES L257 AND L258
                        (SYSTEM / PARTITION GETVIS SPACE
                        EXHAUSTED) IS PREPARED AND GIVEN.
                  - INLPXQNM: A QUALIFIED SUBLIBRARY OR
                        MEMBER NAME IS BUILT.
                  - INLPXSLB: THE MESSAGE L158 (DEFECT IN
                        SUBLIBRARY IS PREPARED AND GIVEN
                  - INLPXTRC: TRACE FUNCTION FOR LEVEL 1/2
                        SERVICES

INPUT =           ADDRESS OF LAMB
                  MESSAGE NUMBER
                  VARIABLE PARTS OF MESSAGE

OUTPUT =          DEPENDING OF FUNCTION:
                  PREPARED MESSAGE OR MESSAGE PARTS.

 ENTRY POINT:     INLPXRCD, INLPXMSG, INLPXPRM, INLPXSYS,
                  INLPXLIB, INLPXVIS, INLPXQNM, INLPXSLB,
                  INLPXTRC
```

INLPCOMP

```
MODULE INLPCOMP
DESCRIPTIVE NAME: COMPARE  COMMAND PROCESSOR

FUNCTION =       PROCESS COMPARE    STATEMENT AND DO SEMANTIC
                 CHECKS:
                 - LABEL INITCOMP:
                    DO PREPARATORY WORK FOR COMPARE  PROCESSING
                 - LABEL SELCOMP:
                    SELECT THE PROCESSING FUNCTIONS:
                      LIBCOMP - PROCESS WHOLE LIBRARIES
                      SUBCOMP - PROCESS WHOLE SUBLIBRARIES
                      MEMCOMP - PROCESS SINGLE MEMBERS
                 - LABEL LIBCOMP:
                    PROCESS LIBRARY LIST OF THE COMMAND INPUT
                    ADD TEMPORARY SEARCH/CATALOG ENTRIES
                    DO SECURITY CHECK
                    BUILD THE SUBLIB LIST OF THE FROM LIBRARY
                 - LABEL LIB1COMP:
                    PROCESS THE SUBLIB NAME LIST
                 - LABEL SUBCOMP:
                    PROCESS SUBLIB LIST OF THE COMMAND INPUT
                    DO SEMANTIC CHECK
                    PROCESS COMPARING FOR EACH SUBLIB
                 - LABEL SUB1COMP:
                    PROCESS ONE FROM-/TO-SUBLIB PAIR
                    ADD TEMPORARY SEARCH/CATALOG ENTRIES
                    BUILD THE MEMBER LIST OF THE FROM SUBLIBRARY
                 - LABEL MEMCOMP:
                    PROCESS MEMBER LIST OF THE COMMAND INPUT
                    GET PERMANENT SEARCH/CATALOG ENTRIES
                    DO SEMANTIC CHECK
                    BUILD MEMBER LIST FOR GENERIC SPECIFICATION
                 - LABEL MDIRCOMP:
                    COMPARE THE MEMBER LIST OF THE FROM-SUBLIB
                    WITH THAT OF THE CORRESPONDING TO-SUBLIB
                 - LABEL MDATCOMP:
                    PROCESS THE MEMBER LIST AND COMPARE CORRES-
                    PONDING MEMBER PAIRS IN BOTH SUBLIBS
                    COMPARE EACH BYTE OF THE MEMBER DATA  UNTIL
                    A MISMATCH IS FOUND
                 - LABEL FINCOMP:
                    FINISH COMPARE    PROCESSING AND RETURN

INPUT =          LIBRARIAN COMMUNICATION REGION INLCCOMR
                 PARSED COMMAND CONTROL BLOCK INLCCMDP
```

```
OUTPUT =          RESULTS OF COMPARING ON SYSLST/SYSLOG
                  RETURN CODE 0 - SUCCESSFUL COMPLETION
                  RETURN CODE 2 - COMPARE RESULT IS UNEQUAL
                  RETURN CODE 4 - MESSAGES L136I, L139I, L153I,
                                          L046I
                  RETURN CODE 8 - ANY OTHER FAILURE

 ENTRY POINT:     SAME AS MODULE NAME

   PURPOSE:       SEE FUNCTION DESCRIPTION

   LINKAGE:       PLS/III CALL CONVENTIONS
```

INLPCONN

```
MODULE INLPCONN
DESCRIPTIVE NAME: CONNECT COMMAND PROCESSOR

FUNCTION =       PROCESS CONNECT STATEMENT AND DO SEMANTIC
                 CHECKS:
                 - LABEL INITCONN:
                    DO PREPARATORY WORK FOR CONNECT PROCESSING
                 - LABEL DISPCONN:
                    PRINT/DISPLAY CONNECT INFO ON REQUEST
                 - LABEL BLDCONN:
                    BUILD CONNECT INFO ENTRY VIA LBRACCES
                 - LABEL FINCONN:
                    FINISH CONNECT PROCESSING AND RETURN

INPUT =          LIBRARIAN COMMUNICATION REGION INLCCOMR
                 PARSED COMMAND CONTROL BLOCK INLCCMDP

OUTPUT =         CONNECT INFORMATION LINE ON REQUEST
                 RETURN CODE 0  - SUCCESSFUL COMPLETION
                 RETURN CODE 8  - ANY FAILURE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPCOPY

```
MODULE INLPCOPY
DESCRIPTIVE NAME: COPY/MOVE  COMMAND PROCESSOR

FUNCTION =       PROCESS COPY/MOVE STATEMENT AND DO SEMANTIC
                 CHECKS:
                 - LABEL INITCOPY:
                    DO PREPARATORY WORK FOR COPY  PROCESSING
                 - LABEL SELCOPY:
                    SELECT THE PROCESSING FUNCTIONS:
                      LIBCOPY - PROCESS WHOLE LIBRARIES
                      SUBCOPY - PROCESS WHOLE SUBLIBRARIES
                      MEMCOPY - PROCESS SINGLE MEMBERS

                 - LABEL LIBCOPY:
                    DO  SEMANTIC CHECK
                    ADD FROM-LIBRARY AS TEMPORARY SEARCH ENTRY
                    ADD TO-LIBRARY AS TEMPORARY ACCESS ENTRY
                    DO SECURITY CHECK ON LIBRARY LEVEL
                    GET ALL SUBLIBNAMES OF THE FROM-LIBRARY
                 - LABEL LIB1COPY:
                    PROCESS ALL SUBLIBRARIES

                 - LABEL SUBCOPY:
                    DO  SEMANTIC CHECK
                    ADD TO-LIBRARY AS TEMPORARY ACCESS ENTRY
                    DO SECURITY CHECK ON SUBLIBRARY LEVEL
                 - LABEL SUB1COPY:
                    ADD TEMPORARY SEARCH AND CATALOG ENTRIES
                      FOR FROM- AND TO-SUBLIBRARY
                    DO FURTHER SEMANTIC CHECKS
                    EMPTY AN EXISTING TO-SUBLIBRARY I.E THE
                      CURRENT CONTENTS IS LOST
                    A NON EXISTING TO-SUBLIBRARY IS DEFINED
                      IMPLICITLY
                    GET MEMBER LIST OF FROM-SUBLIBRARY
                    FOR MOVE, THE FROM-SUBLIB IS DELETED

                 - LABEL MEMCOPY:
                    GET PERMANENT SEARCH AND CATALOG ENTRIES
                    DO  SEMANTIC CHECK
                    COLLECT NON GENERIC MEMBERS INTO THE STOW
                      TABLE
                    GET MEMBER LIST FOR GENERIC SPECIFICATION
                      AND PROCESS IT (BLDLCOPY)
```

```
                - LABEL MEM1COPY:
                   PROCESS ALL MEMBERS IN THE STOW TABLE
                   CONNECT FROM-MEMBER IN THE FROM-SUBLIB
                   FIND TO-MEMBER IN THE CATALOG-CHAIN
                   COPY THE MEMBER SPACE
                   UPDATE THE TO-DIRECTORY ENTRY(RLD-INFO)
                   ISSUE STOW REQUEST FOR THE TO-MEMBERS
                   FOR MOVE, DELETE THE FROM-MEMBERS
                   LIST ALL COPIED MEMBERS IF REQUESTED

                - LABEL FINCOPY:
                   FINISH COPY/MOVE PROCESSING AND RETURN

INPUT =         LIBRARIAN COMMUNICATION REGION INLCCOMR
                PARSED COMMAND CONTROL BLOCK INLCCMDP

OUTPUT =        RETURN CODE 0 - SUCCESSFUL COMPLETION
                RETURN CODE 4 - MESSAGES L031I, L136I, L139I,
                                         L148I, L046I, L153I,
                                         L284I
                RETURN CODE 8 - ANY OTHER FAILURE

 ENTRY POINT:   SAME AS MODULE NAME

   PURPOSE:     SEE FUNCTION DESCRIPTION

   LINKAGE:     PLS/III CALL CONVENTIONS
```

INLPDBLO

```
MODULE INLPDBLO
DESCRIPTIVE NAME:  DEBLOCK OBJECT DECKS SUBROUTINE

FUNCTION =
                ONE 322 BYTE LIBRARY BLOCK FROM
                A MEMBER OF THE RELOCATABLE LIBRARY ON
                A BACKUP TAPE OF VSE/AF RELEASE 3.5 OR
                AN EARLIER RELEASE IS DEBLOCKED
                INTO 80 BYTE OBJECT RECORDS

INPUT =         BUFPTR  - POINTER TO BUFFER
                          CONTAINING THE 322 BYTE RECORD
                RCOUNT  - LIBRARY RECORD COUNT
                          = 1 IF FIRST 322 BYTE RECORD
                IBUFPTR - PTR TO OUTPUT AREA

OUTPUT =        OUTPUT AREA CONTAINING THE
                DEBLOCKED OBJECT RECORDS

                IBUFPTR - PTR TO NEXT FREE BYTE IN OUTPUT AREA

                RETURN CODES: 0 - NORMAL RETURN
                              4 - INVALID INPUT
                                  (LOG RECORD COUNT > 2)

 ENTRY POINT:   SAME AS MODULE NAME

   PURPOSE:     SEE FUNCTION DESCRIPTION

   LINKAGE:     PLS/III CALL CONVENTIONS
```

INLPDEF

```
MODULE INLPDEF
DESCRIPTIVE NAME: DEFINE COMMAND PROCESSOR

FUNCTION =       PROCESS DEFINE STATEMENT AND DO SEMANTIC
                 CHECKS:
                 - LABEL INITDEF:
                    DO PREPARATORY WORK FOR DEFINE PROCESSING
                 - LABEL LIBRDEF:
                    DEFINE LIBRARY
                 - LABEL SUBLDEF:
                    DEFINE SUBLIBRARY
                 - LABEL FINDEF:
                    FINISH DEFINE PROCESSING AND RETURN

INPUT =          LIBRARIAN COMMUNICATION REGION INLCCOMR
                 PARSED COMMAND CONTROL BLOCK INLCCMDP

OUTPUT =         RETURN CODE 0 - SUCCESSFUL COMPLETION
                 RETURN CODE 4 - MESSAGES L128I, L148I
                 RETURN CODE 8 - ANY OTHER FAILURE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPDEL

```
MODULE INLPDEL
DESCRIPTIVE NAME: DELETE COMMAND PROCESSOR

FUNCTION =       PROCESS DELETE STATEMENT AND DO SEMANTIC
                 CHECKS:
                 - LABEL INITDEL:
                    DO PREPARATORY WORK FOR DELETE PROCESSING
                 - LABEL LIBRDEL:
                    DELETE LIBRARY
                 - LABEL SUBLDEL:
                    DELETE SUBLIBRARY
                 - LABEL MEMBDEL:
                    DELETE MEMBER
                 - LABEL FINDEL:
                    FINISH DELETE PROCESSING AND RETURN

INPUT =          LIBRARIAN COMMUNICATION REGION INLCCOMR
                 PARSED COMMAND CONTROL BLOCK INLCCMDP

OUTPUT =         RETURN CODE 0 - SUCCESSFUL COMPLETION
                 RETURN CODE 4 - MESSAGES L042I, L082I, L101I,
                                 L149I
                 RETURN CODE 8 - ANY OTHER FAILURE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPDIAG

```
MODULE INLPDIAG
DESCRIPTIVE NAME: ISSUE DIAGNOSTICS

FUNCTION =       THE MODULE TRANSLATES AN INPUT MESSAGE NUMBER
                 INTO THE CORRESPONDING FULL TEXT MESSAGE AND
                 WRITES IT OUT TO SYSLST, SYSLOG OR BOTH(DEPEN-
                 DING ON THE MESSAGE AND ON THE SPECIFICATION
                 OF THE DTF NEEDED).
                 IF A DTF IS MISSING:
                  THE MESSAGE IS PLACED INTO THE I/O-AREA ONLY.
                  A-TYPE MESSAGES ARE WRITTEN VIA EXCP MACRO.
                 FOR THE MESSAGES L150I, L151I, L152I (INTERNAL
                 ERRORS) THE DUMP MACRO IS ISSUED.
                 FOR A-TYPE MESSAGES SOLICITING A REPLY, THE
                 REPLY IS READ. FOR SOME OF SUCH MESSAGES THE
                 REPLY CAN ALSO BE PRINTED ON SYSLST.

INPUT =          LIBRARIAN ACCESS METHOD BLOCK (LAMB):
                   FOUR-BYTE MESSAGE NUMBER.
                   UP TO FOUR INSERTION TEXTS TO REPLACE THE
                   CORRESPONDING PLACEHOLDERS IN THE MESSAGE
                   SKELETON.
                   SYSLST/SYSLOG DTF AND I/O-AREA INFORMATION.
                 REGISTER 1 CONTAINS  ADDRESS OF LAMB.

OUTPUT =         MESSAGE SYSLST AND/OR SYSLOG
                 REPLY TEXT IN THE I/O-AREA FOR A-TYPE MESSAGES
                 RETURN CODES: 0 = SUCCESSFUL
                               4 = WARNING: MISSING DTF-MESSAGE
                                   IS NOT WRITTEN BUT STORED TO
                                   THE CORRESPONDING I/O-AREA.
                              12 = MORE PLACEHOLDERS THAN
                                   ARGUMENTS
                              16 = NO DTF AND I/O-AREA INFO
                                   SPECIFIED
                              20 = REQUIRED SYSLST I/O-AREA IS
                                   MISSING
                              24 = SYSLST I/O-AREA LENGTH IS
                                   INVALID
                              28 = REQUIRED SYSLOG I/O-AREA IS
                                   MISSING
                              32 = SYSLOG I/O-AREA LENGTH IS
                                   INVALID
                              36 = MAXIMUM MESSAGE LENGTH IS
                                   EXCEEDED
 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPEXIT

```
MODULE INLPEXIT
DESCRIPTIVE NAME: ERROR EXIT ROUTINE

FUNCTION =       EXIT FOR ABNORMAL TERMINATION:
                    RESET THE DELAYED CANCEL OPTION
                    FREE GETVIS SPACE WHICH MIGHT BE USED FOR
                    LONG COMMAND LISTS
                    SET FAILING RETURN CODE IN LIBR.COMREG
                    FLUSH COMMAND LINES WITH CONTINUATION SIGN
                    WHEN RUNNING BATCH
                    1) CALL INTERFACE ACTIVE OR RUNNING UNDER
                       CONTROL OF MSHP:
                         RETURN TO INLPMAIN FOR NORMAL
                         TERMINATION
                    2) ELSE:
                         RETURN TO INLPSYNA TO PROCESS NEXT
                         COMMAND

INPUT =         NONE

OUTPUT =        NONE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPFBUF

```
MODULE INLPFBUF
DESCRIPTIVE NAME:  FREE BUFFER LIST

FUNCTION =       RETURN BUFFERS TO THE FREE OR TO PREEMPTABLE
                 QUEUE

INPUT =          RBLKPTR  PTR(31) -REQUEST BLOCK PTR
                 LRPLPTR  PTR(31) -LRPL PTR
                 FBUFTYPE CHAR(1) -TYPE OF BUFFER REQUESTED
                                   FREEBUF FREE BUFFER
                                   PREEMPT MAKE PREEMPTABLE

OUTPUT =         RETC000          -NORMAL RETURN

 ENTRY POINT:    INLPFBUF
   PURPOSE:      FREE ALL BUFFERS IN REQUEST LIST

 ENTRY POINT:    INLPPBUF
   PURPOSE:      FREE ALL BUFFERS IN PRE-EMPTABLE LIST
                 OF SPECIFIED BUFFER POOL (SHARED OR PRIVATE)
                 BUFFERS MARKED AS CHANGED ARE WRITTEN OUT
                 TO DISK

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPFIND

```
MODULE INLPFIND
DESCRIPTIVE NAME: FIND MEMBER IN A CHAIN OF SUBLIBRARIES

FUNCTION =        THE CHAIN OF ACCESSED LIBRARIES IS SEARCHED
                  FOR THE MEMBER GIVEN IN ARGUMENT INLCLARG.
                  IF THE MEMBER IS FOUND IT WILL BE CONNECTED.
          FOR EACH SUBLIBRARY IN THE SEARCHING CHAIN (DEFINED
          BY PARAMETERS LIBTYPE AND LIBUSE), A CONNECT-TO-
          MEMBER (MEMBER IS GIVEN IN PARAMETER ARG) IS TRIED.
          THE FIRST SUBLIBRARY IN WHICH THE MEMBER IS FOUND IS
          REPORTED IN THE ARGUMENT (INLCLARG). THE MEMBER WILL
          BE CONNECTED AFTERWARDS.
          - IF THE MEMBER DOES NOT EXIST IN THE GIVEN CHAIN,
          RETURN CODE 8 IS PASSED BACK.
          - IF THE SEARCHING CHAIN IS EMPTY, RETURN CODE 12 IS
          PASSED BACK.
          - THE FIND OPERATION WORKS ON A GENERIC MEMBER
          SPECIFICATION, TOO.

INPUT =           REQUEST PARAMETER LIST INLCLRPL

OUTPUT =          MEMBER DIRECTORY INFORMATION.
                  MESSAGES, RETURN CODES.
                  0 ... REQUEST HAS BEEN HONORED
                  8 ... MEMBER NOT FOUND
                 12 ... SEARCHING CHAIN NOT DEFINED
                 16 ... EXTERNAL ERROR WITH FEEDBACK CODE
                 20 ... INTERNAL ERROR WITH FEEDBACK CODE
                 32 ... SECURITY VIOLATION

 ENTRY POINT:     SAME AS MODULE NAME
```

INLPGBUF

```
MODULE INLPGBUF
DESCRIPTIVE NAME:  ALLOCATE REQUESTED BUFFERS

FUNCTION =       ALLOCATE REQUESTED NUMBER OF BUFFERS.
                 IF NOT ENOUGH BUFFERS CAN BE OBTAINED
                 FROM THE FREE QUEUE, AN ATTEMPT IS MADE
                 TO OBTAIN THEM FROM THE PRE-EMPTABLE QUEUE.

INPUT =          RBLKPTR  PTR(31) -REQ.BLOCK POINTER
                 LRPLPTR  PTR(31) -LRPL POINTER
                 GBUFPRBA CHAR(6) -1TH PRBA (CAN BE EMPTY)
                 GBUFFUN  CHAR(1) -TYPE OF REQUEST
                                   NEWBUF NEW BUFFER
                                   OLDBUF POSSIBLE PREEMPTED
                 GBUFTYPE CHAR(1) -TYPE OF BUFFER
                                   SHAREDB SHARED BUFFERS
                                   PRIVATB PRIVATE BUFFERS
                 GBUFNO   BIN(15) -NO. DATA BUFFERS REQUESTED
                                   IF ZERO, ALL AVAILABLE
OUTPUT =         RBLKPTR  PTR(31) -REQ.BLOCK PTR, POINTS TO
                                   BUCB,BHDR-LIST ALLOCATED
                 RETC000          -NORMAL RETURN
                 RETC020          -REQUEST NOT SATISFIED

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPGDIR

```
MODULE INLPGDIR
DESCRIPTIVE NAME: GET NEXT DIRECTORY RECORD

FUNCTION =        PROVIDES NEXT SUBLIBRARY OR MEMBER DIRECTORY
                  ENTRY. POSITIONS TO FOLLOWING ENTRY (IF ANY).
  ======> THE PROCEDURE IS CALLED BY THE MACRO INLMGDIR.
          THE  SEQUENTIALLY NEXT DESCRIPTOR OF THE LIBRARY
          (LEVEL=SUBLIB) OR SUBLIBRARY (LEVEL=MEMBER) IS PASSED
          BACK TO THE INVOKER. FOR REQUEST LEVEL=LIBRARY THE
          LIBRARY DESCRIPTOR IS PASSED BACK.
          - SECURITY CHECK IS DONE FOR LEVEL=LIBRARY AND LEVEL=
          SUBLIBRARY FOR READ AUTHORIZATION ON LIBRARY LEVEL,
          FOR LEVEL=MEMBER FOR READ AUTHORIZATION ON SUBLIBRARY
          LEVEL.
          - FOR READING THE LIBRARY DESCRIPTOR OR THE SUBLIB
          DIRECTORY, THE LIBRARY HAS TO BE CONNECTED.
          - FOR READING THE MEMBER DIRECTORY OF A SUBLIBRARY
          THE SUBLIBRARY HAS TO BE CONNECTED.

INPUT =           LIBRARY REQUEST PARAMETER LIST (INLCLRPL)
                  LIBRARY ACCESS CONTROL BLOCK (INLCLACB)
                  SUBLIBRARY ACCESS CONTROL BLOCK (INLCSACB)

OUTPUT =          DIRECTORY ENTRY
                  MESSAGES, RETURN CODES:
                  0 = SUCCESSFULLY COMPLETED
                  4 = SUCCESSFULLY COMPLETED, LAST ENTRY
                 12 = REQUEST OUTSIDE DIRECTORY
                 16 = EXTERNAL ERROR (FEEDBACK CODE)
                 20 = INTERNAL ERROR (FEEDBACK CODE)
                 32 = SECURITY VIOLATION

 ENTRY POINT:     SAME AS MODULE NAME
```

INLPGETR

```
ENTRY INLPGETR
DESCRIPTIVE NAME:  MEMBER DATA READ

INLPGETR is an ENTRY Point in Module INLPPUTR. For a detailed
description see MODULE INLPPUTR.
```

INLPGSPA

```
MODULE INLPGSPA
DESCRIPTIVE NAME:  LIBRARY DISK SPACE MANAGEMENT

FUNCTION =       ALLOCATE AND FREE LIBRARY DISK SPACE
                 IF LIBRARY IS IN VSAM MANAGED SPACE,
                 TRY SECONDARY ALLOCATION TO FULFILL
                 REQUIREMENT.
                 DURING ALLOCATION/DEALLOCATION THE LIBRARY
                 SPACE IS LOCKED WITH  UNIQUE RESOURCE NAME

INPUT =          RBLKPTR  PTR(31) -POINTS TO BHDR-LIST FOR
                                   WHICH LBS ARE ALLOCATED
                 LRPLPTR  PTR(31) -LRPL POINTER
                 FSPAPRBA CHAR(6) -1TH PRBA IN RECLAMATION LIST
                 FSPANO   BIN(15) -COUNT OF LBS TO BE FREED
                                   IF ZERO, TILL END OF LIST
OUTPUT =         RETC000          -NORMAL RETURN
                 RETC012          -LIBRARY FULL
                 RETC016          -RESOURCE ALREADY LOCKED
                 RETC020          -LIBRARY/RECL.CHAIN ERROR
 ENTRY POINT:    INLPGSPA

   PURPOSE:      ALLOCATE ONE PRBA TO EACH BHDR IN REQUEST LIST

 ENTRY POINT:    INLPFSPA

   PURPOSE:      FREE LBS IN RECLAMATION CHAIN

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPGVFV

```
MODULE INLPGVFV
DESCRIPTIVE NAME: ISSUE GETVIS/FREEVIS

FUNCTION = 1. ISSUE GETVIS TO ALLOCATE SPACE FOR AUTOMATIC
           STORAGE OR CONTROL BLOCKS.
           BUILD SUBPOOL ID IN LRPLSPID OR LAMBSPID - IF IT IS
           INCOMPLETE - BY INSERTING THE TASK-ID.
           REQUEST PARTITION OR SYSTEM GETVIS SPACE DEPENDENT
           ON LAMB FLAG LAMBSVIS.
           INSERT ADDRESSES IN LAMBGVS,LAMBGVE,LAMBGVU.
           THE SPACE ALLOCATED FOR AUTOMATIC STORAGE IS 8K
           (A MULTIPLE OF 2K). THE STORAGE REQUESTS OF THE
           INDIVIDUAL SVA ROUTINES ARE SATISFIED WITH SPACE
           FROM THIS 8K AREA.
           2. ISSUE FREEVIS TO FREE SPACE FOR AUTOMATIC STORAGE
           OR CONTROL BLOCKS IN PARTITION OR SYSTEM GETVIS AREA.
           RESET ADDRESSES IN LAMBGVS,LAMBGVE,LAMBGVU.

INPUT = ADDRESS OF LRPL OR LAMB
        OR ADDRESS AND LENGTH OF AREA TO BE FREED.
        INVOCATION THRU MACROS INLMDSTO AND INLMFREE.

OUTPUT = UPDATED LAMB.
         MESSAGES ON SYSLST/SYSLOG VIA MODULE INLPDIAG.
         GETVIS/FREEVIS RETURN CODES IN R15.

 ENTRY POINT: INLPEGLL, INLPEGLR, INLPEGRR, INLPEFL, INLPEFR

 PURPOSE = SEE DESCRIPTIONS AT ENTRY POINTS

 LINKAGE =  R0  LOADED WITH GETVIS/FREEVIS LENGTH
            R1  LOADED WITH ADDRESS OF LRPL/LAMB OR WITH
                ADDRESS OF AREA TO BE FREED
            R13 LOADED WITH SAVEAREA ADDRESS (18 FULLWORDS)
            R14 LOADED WITH RETURN ADDRESS
            R15 LOADED WITH ENTRY POINT
```

INLPLBGP

```
MODULE INLPLBGP
DESCRIPTIVE NAME:  PERFORM LIBRARY BLOCK PHYSICAL I/O

FUNCTION =       FOR SPECIFIED PRBA-LIST
                 PREPARE IORB,CCWS AND SUBMIT  AN I/O
                 REQUEST VIA SENTER SVC TO INLPDOIO

INPUT =          RBLKPTR  PTR(31) -REQUEST BLOCK POINTER
                 LRPLPTR  PTR(31) -LRPL POINTER
                 LBGPCOMM CHAR(1) -COMMAND REQUESTED
                                   LBREAD READ
                                   LBWRITE WRITE
OUTPUT =         RETC000          -NORMAL RETURN
                 RETC020          -INVALID PRBA-LIST

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS

 GENERATED CCW-CHAINS:

 ------------------  C K D  ---------------------

   READ CCW CHAIN:

      07     SEEK
      23     SET SECTOR
      31     SEARCH ID EQUAL
      08     TIC
      86     READ DATA MULT. TRACK
      86     READ DATA MULT. TRACK
       .
       . (USING COMMAND CHAINING)
       .
      86     READ DATA MULT. TRACK
```

```
  WRITE CCW CHAIN:

     07     SEEK
     23     SET SECTOR
     31     SEARCH ID EQUAL
     08     TIC
     05     WRITE DATA
     31     SEARCH ID EQUAL
     08     TIC
     05     WRITE DATA
      .
      . (USING COMMAND CHAINING)
      .
     31     SEARCH ID EQUAL
     08     TIC
     05     WRITE DATA

------------------  E C K D  --------------------

  READ CCW CHAIN:

     63     DEFINE EXTENT
        DEFINE EXTENT PARAMETERS :
         MASK BYTE        :  NORMAL OPERATION
         GLOBAL ATTRIBUTE :  NORMAL ECKD* OPERATION
         BLOCKSIZE        :  LIBRARY BLOCK SIZE
         EXTENT BOUNDARIES:  CCHH OF LOWEST AND HIGHEST TRACK
     47     LOCATE RECORD
        LOCATE RECORD PARAMETERS :
         OPERATION CODE   :  X'16' (COUNT AREA ORIENTATION)
         AUXILIARY BYTE   :  X'00'
         COUNT            :  NUMBER OF CONTIGUOUS LB'S TO READ
         SEEK ADDRESS     :  CCHH  LOWEST TRACK ADDRESS
         SEARCH ARGUMENT  :  CCHHR ADDRESS OF FIRST LB TO READ
         SECTOR NUMBER    :  X'SS'  SECTOR NUMBER
         TLF              :  X'0000' (NOT USED)
     86     READ DATA MULT. TRACK
     86     READ DATA MULT. TRACK
      .
      . (USING COMMAND CHAINING)
      .
     86     READ DATA MULT. TRACK

  WRITE CCW CHAIN:

     63     DEFINE EXTENT
         (SEE READ CCW CHAIN FOR DEF EXTENT PARAMETERS )
     47     LOCATE RECORD
         OPERATION CODE   :  X'01' (COUNT AREA ORIENTATION)
         (ELSE SEE READ CCW CHAIN FOR LOCATE RECORD PARAMETERS)
     85     WRITE UPDATE DATA
     85     WRITE UPDATE DATA
      .
      . (USING COMMAND CHAINING)
      .
     85     WRITE UPDATE DATA
```

```
------------------  F B A  --------------------

  READ CCW CHAIN:

     63     DEFINE EXTENT
     43     LOCATE RECORD
     42     FBA READ
     42     FBA READ
      .
      . (USING DATA CHAINING)
      .
     42     FBA READ

  WRITE CCW CHAIN:

     63     DEFINE EXTENT
     43     LOCATE RECORD
     41     FBA WRITE
     41     FBA WRITE
      .
      . (USING DATA CHAINING)
      .
     41     FBA WRITE
```

INLPLCON

```
MODULE INLPLCON
DESCRIPTIVE NAME: CONNECT TO LIBRARY

FUNCTION =
          THE REQUESTED LIBRARY (DEFINED BY LRPLINFO IN LRPL)
          IS CONNECTED AND THE CURRENT POSITION PLACED TO THE
          START OF THE SUBLIBRARY INDEX.
          - CONNECT=OLD: THE LIBRARY ALREADY EXISTS.
          - CONNECT=NEW: THE LIBRARY IS NEW, IT HAS TO BE
                         PRE-FORMATTED.
          - IF A CONNECTION TO ANOTHER LIBRARY OR ANY
            SUBLIBRARY HOLDS, THE DISCONNECTION
            OF THE OTHER LIBRARY/SUBLIBRARY IS DONE.
          - AN EXISTING CONNECTION TO A MEMBER RESULTS IN ERROR
          - IF NO CONNECTION EXISTS, THE CORRESPONDING LACB IS
            ALLOCATED BY A GETVIS REQUEST.

INPUT =   REQUEST PARAMETER LIST INLCLRPL

OUTPUT =  MESSAGES, RETURN CODES
                  0 REQUEST HAS BEEN HONORED
                  4 SUCCESSFUL, ALREADY CONNECTED
                 16 EXTERNAL ERROR WITH FEEDBACK CODE
                 20 INTERNAL ERROR WITH FEEDBACK CODE
                 32 SECURITY VIOLATION

 ENTRY POINT: SAME AS MODULE NAME
```

INLPLDIS

```
MODULE INLPLDIS
DESCRIPTIVE NAME: DISCONNECT FROM LIBRARY

FUNCTION =        MAKES LIBRARY UNAVAILABLE FOR ACCESS
 =======> ANY CONNECTION TO A LIBRARY OBJECT IS RELEASED.
          - IF NO CONNECTION IS ACTIVE, RETURN CODE 8 IS PASSED
          BACK.
          - IF A LIBRARY IS CONNECTED, THE LACB IS FREED.
          - IF A SUBLIB IS CONNECTED, LACB AND SACB ARE FREED.
          - IF A MEMBER IS CONNECTED, LACB AND SACB AND MACB
          ARE FREED.
          - ADDITIONALLY, THE LIBRARY SEARCH TABLE LBTB IS
          FREED (IF IT EXISTS).
          - IF PRIVATE BUFFERS HAVE BEEN UPDATED (WITH MEMBER
          CONNECTION) A WRITE OPERATION IS FORCED.

INPUT =           LIBRARY REQUEST PARAMETER LIST (INLCLRPL)

OUTPUT =          MESSAGES, RETURN CODES
                  0... REQUEST HAS BEEN HONORED
                  8... NO CONNECTION WAS ACTIVE
                 12... LIBRARY IS FULL
                 16... EXTERNAL ERROR WITH FEEDBACK CODE
                 20... INTERNAL ERROR WITH FEEDBACK CODE

 ENTRY POINT:     SAME AS MODULE NAME
```

INLPLEV3

```
MODULE INLPLEV3
DESCRIPTIVE NAME: SELECT COMMAND EXECUTION MODULES

FUNCTION =       CHECK THE COMMAND NAMES SPECIFIED IN THE
                 COMMAND CONTROL BLOCK(INLCCMDP) AND BRANCH
                 TO THE CORRESPONDING EXECUTION MODULE.
                 DO SPECIAL ACTIONS IF RUNNING IN MIGRATION
                 MODE.

INPUT =         PARSED COMMAND BLOCK (INLCCMDP)

OUTPUT =        NONE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPLID

```
MODULE INLPLID
DESCRIPTIVE NAME: LIST DIRECTORY COMMAND PROCESSOR

FUNCTION =       PROCESS LISTD STATEMENT AND DO SEMANTIC
                 CHECKS:

                 - LABEL INITLID:
                   DO PREPARATORY WORK FOR LISTD PROCESSING
                   AND SEMANTIC CHECKING

                 - LABEL LIBDIR:
                   GET LIBRARY DIRECTORY INFORMATION
                   IF LIBRARY SPECIFICATION IN COMMAND

                 - LABEL SUBDIR:
                   GET LIBRARY DIRECTORY INFORMATION
                   IF SUBLIBRARY OR MEMBER SPECIFICATION
                   IN COMMAND

                 - LABEL PARTSUB:
                   GET SUBLIBRARY DIRECTORY INFORMATION

                 - LABEL PARTMEM:
                   GET SUBLIBRARY INFORMATION

                 - LABEL STATLIB:
                   PRINT LIBRARY STATUS

                 - LABEL STATSUB:
                   PRINT SUBLIBRARY STATUS

                 - LABEL PRTSLINE:
                   PRINT INFORMATION FOR SUBLIBRARY PER LINE

                 - LABEL PRTMEM:
                   PRINT INFORMATION PER MEMBER

                 - LABEL PRTMLINE:
                   PRINT ONE LINE INFORMATION PER MEMBER

                 - LABEL SUBLIST:
                   PRINT PAGE WITH NAMES OF MEMBERS
                   IF OUTPUT = SHORT

                 - LABEL SDLDIR:
                   GET AND DISPLAY SYSTEM DIRECTORY LIST
                   IF SDL SPECIFICATION IN COMMAND

                 - LABEL ABNORMAL:
                   TERMINATE LISTD PROCESSING
```

```
INPUT =          LIBRARIAN COMMUNICATION REGION INLCCOMR
                 PARSED COMMAND CONTROL BLOCK INLCCMDP

OUTPUT =         DIRECTORY LISTS IN VARIOUS FORMATS ON
                 SYSLST/SYSLOG
                 RETURN CODE 0 - SUCCESSFUL COMPLETION
                 RETURN CODE 4 - EXISTENCE CHECK: MESSAGES
                                 L042I, L082I, L101I, L187I
                 RETURN CODE 8 - ANY OTHER FAILURE
                 RETURN CODE 16 - MESSAGE L151I

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPLIPU

```
MODULE INLPLIPU
DESCRIPTIVE NAME: LIST/PUNCH COMMAND PROCESSOR

FUNCTION =       PROCESS LIST/PUNCH STATEMENT AND DO SEMANTIC
                 CHECKS:
                 - LABEL INITLIPU:
                    DO PREPARATORY WORK FOR LIST/PUNCH PROCESS
                 - LABEL LIBLIPU:
                    FIND THE LIBRARY/SUBLIBRARY IN WHICH TO
                    LIST/PUNCH   THE MEMBER
                    DISTINGUISH BETWEEN ACCESS-CHAIN AND NON
                    ACCESS-CHAIN PROCESSING
                    SELECT THE LIST/PUNCH PROCEDURE DEPENDING
                    ON THE MEMBER TYPE
                 - LABEL PHASLIPU:
                    LIST/PUNCH MEMBERS WITH RECORD TYPE
                    UNDEFINED (PHASES, DUMPS)
                 - LABEL M80LIPU:
                    LIST/PUNCH MEMBERS WITH RECORD TYPE FIXED
                    80 BYTES (OBJECT DECKS, PROCEDURES,...)
                 - LABEL FINLIPU:
                    FINISH LIST/PUNCH PROCESSING AND RETURN

INPUT =          LIBRARIAN COMMUNICATION REGION INLCCOMR
                 PARSED COMMAND CONTROL BLOCK INLCCMDP

OUTPUT =         VARIOUS MEMBER LISTINGS ON SYSLST/SYSLOG

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPLLSR

```
MODULE INLPLLSR
DESCRIPTIVE NAME: LIST LIBRARY SPACE REQUIREMENTS

FUNCTION =      LIST SPACE REQUIREMENT INFORMATION
                FOR LIBRARIES/SUBLIBRARIES ON BACKUP TAPES

   ENTRY POINT: INLPLLSR

                 LIST LIBRARY SPACE REQUIREMENTS

INPUT =        BUFPTR  -  POINTER TO LIBRARY HEADER

OUTPUT =       ORIGINAL AND MINIMUM SPACE REQUIREMENTS FOR
               THE LIBRARY ARE LISTED ADJUSTED FOR
               THE SUPPORTED DASD DEVICES

   ENTRY POINT: INLPLLBS

                 LIST LIBRARY BLOCKS NEEDED FOR THE
                 SUBLIBRARY OF THE LIBRARY IN PROCESS

INPUT =        BUFPTR  -  POINTER TO SUBLIBRARY HEADER
               QSLNAME -  QUALIFIED SUBLIBRARY NAME

OUTPUT =       THE QUALIFIED SUBLIBRARY NAME AND
               THE MINIMUM NUMBER OF LIBRARY BLOCKS
               NEEDED FOR THIS SUBLIBRARY ARE LISTED

   ENTRY POINT: INLPLSSR

                 LIST LIBRARY BLOCKS NEEDED FOR THE
                 SUBLIBRARY OF THE LIBRARY IN PROCESS

INPUT =        BUFPTR  -  POINTER TO SUBLIBRARY HEADER

OUTPUT =       THE MINIMUM NUMBER OF LIBRARY BLOCKS
               NEEDED FOR THIS SUBLIBRARY ARE LISTED

   ENTRY POINT: INLPLLRO

                 LIST EXPECTED LIBRARY SPACE REQUIREMENTS
                 WHICH A LIBRARY ON A VSE/AF 1.3.5 (OR EARLIER)
                 INPUT TAPE WILL PROBABLY NEED PROBABLY
                 IN WHEN RESTORED TO A VSE/AF 2.1 LIBRARY.

INPUT =        EXPDLBKS - NO OF LIBRARY BLOCKS NEEDED PROBABLY
```

```
OUTPUT =       THE EXPECTED NUMBER OF LIBRARY BLOCKS
               AND THE CORRESPONDING SPACE REQUIREMENTS FOR
               THE 'OLD' LIBRARY ARE LISTED ADJUSTED FOR
               THE SUPPORTED DASD DEVICES

   ENTRY POINT: INLPLBLN

                 PRINT A BLANK LINE

   ENTRY POINT: INLPINFR

                 INITIALIZE FORMATTED SPACE REQUIREMENT
                 LIST RECORD

INPUT =        BFIDPTR  -  POINTER TO BACKUP FILE ID

 ENTRY POINT:    SAME AS MODULE NAME

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPLSIM

```
MODULE INLPLSIM
DESCRIPTIVE NAME: LIBRARY SECURITY INTERFACE MODULE

FUNCTION =        CHECKS WHETHER THE ACTUAL ACCESS REQUEST
                  TO A LIBRARY OBJECT IS ALLOWED. ANY ACCESS
                  VIOLATION IS RECORDED.
  ======> THIS ROUTINE IS CALLED IN THREE DIFFERENT WAYS,
          1. THE PASSED CONTROL BLOCK IS THE LRPL, I.E.
          THE SECURITY CHECK HAS TO BE MADE DEPENDENT ON
          THE CURRENT LEVEL 2 SERVICE REQUEST.
          2. THE PASSED CONTROL BLOCK IS THE TOLCRQL, I.E.
          THE ACCESS RIGHTS TO A SUBLIBRARY ARE EXTRACTED FROM
          THE DTSECTAB AND STORED IN THE LIBRARY CONTROL TABLES
          3. THE PASSED CONTROL BLOCK IS THE LIBINFO, I.E.
          A SPECIFIC SECURITY CHECK HAS TO BE DONE. THE REQ'D
          ACCESS RIGHT IS GIVEN, THE ACTUAL ACCESS RIGHT IS
          RETURNED.

INPUT =           PARAMETER BLOCK INLCLSIM

OUTPUT =          MESSAGES, RETURN CODES:
                         0 ... ACCESS ALLOWED
                        16 ... PARAMETER ERROR (FEEDBACK CODE)
                        20 ... PROCESSING ERROR (FEEDBACK CODE)
                        32 ... SECURITY VIOLATION

 ENTRY POINT:     SAME AS MODULE NAME
```

INLPMAIN

```
MODULE INLPMAIN
DESCRIPTIVE NAME: LIBRARIAN INITIALIZATION ROUTINE

FUNCTION =       CHECK IF LIBRARIAN IS USED VIA THE CALL INTER-
                 FACE AND INDICATE IN LIBR. COMM. REGION
                 INITIALIZE DATA AREAS, CONTROL BLOCKS, ...
                 OPEN SYSLST/SYSLOG DTFS
                 CHECK IF LIBRARIAN IS CALLED FROM SYSLOG OR
                 FROM BATCH AND INDICATE IN LIBR. COMM. REGION
                 CHECK IF LIBRARIAN IS CALLED FROM MSHP  AND
                 INDICATE IN LIBR. COMM. REGION
                 CHECK THE // EXEC PARM= INFORMATION:
                   IF 'MSHP' SPECIFIED THEN SET MSHP BYPASS
                   FLAG IN THE LIBR. COMMUNICATION REGION
                   IF MIGRATION REQUESTED THEN SET THE MIGRATION
                   FLAGS IN THE LIBR. COMMUNICATION REGION
                   IF 'ACCESS' COMMAND SPECIFIED THEN SET ACCESS
                   FLAG IN THE LIBR. COMMUNICATION REGION AND
                   MOVE THE COMMAND SPECIFICATION TO THE
                   SYSIPT/SYSLOG I/O-AREA
                 GIVE CONTROL TO PARSER MODULE (INLPSYNA)
                 CLOSE SYSLST, SYSIPT, SYSLOG, SYSPCH DTFS
                 RESET DELAYED CANCEL FUNCTION
                 TERMINATE LIBRARIAN SESSION AND RETURN TO
                 JCL OR TO THE USER OF THE CALL INTERFACE

INPUT =          NONE

OUTPUT =         RETURN CODE:
                   0 = SUCCESSFUL
                   8 = FUNCTIONS SKIPPED OR PARTIALLY COMPLETED
                  16 = INTERNAL ERROR

 ENTRY POINT:    INLPMAIN   - START OF LIBR SESSION
                 INLPCACK   - CHECK DELAYED CANCEL REQUEST
                 INLPCANC   - FORCE CANCEL
                 INLPEND    - TERMINATE LIBR SESSION

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPMDIS

```
MODULE INLPMDIS
DESCRIPTIVE NAME: DISCONNECT FROM MEMBER

FUNCTION =        MAKES MEMBER UNAVAILABLE FOR ACCESS
                  SUBLIBRARY REMAINS ACCESSIBLE
  ======> IF A MEMBER IS CONNECTED, THE CONTROL BLOCK STRING
          LACB-SACB-MACB IS REDUCED TO LACB-SACB.
          - IF NO MEMBER IS CONNECTED, RETURN CODE 8 IS PASSED
          BACK.
          - IF THERE WAS A CHANGE TO THE CONNECTED MEMBER, FOR
          WHICH THE BUFFERS HAVEN'T BEEN WRITTEN OUT, A
          WRITE REQUEST IS FORCED.
          - THE PRIVATE BUFFERS ARE INVALIDATED.
          - THE MACB IS UNCHAINED AND
          IF LRPLSAVE CONTAINS THE VALUE -1 THE MACB ADDRESS
          IS PASSED BACK IN LRPLSAVE. OTHERWISE, THE MACB IS
          FREED.

INPUT =           LIBRARY REQUEST PARAMETER LIST (INLCLRPL)

OUTPUT =          MESSAGES, RETURN CODES
                      0 ... FUNCTION SUCCESSFULLY COMPLETED
                      8 ... MEMBER WAS NOT CONNECTED
                     12 ... LIBRARY IS FULL (FOR CONNECT=NEW)
                     16 ... EXTERNAL ERROR WITH FEEDBACK CODE
                     20 ... INTERNAL ERROR WITH FEEDBACK CODE

 ENTRY POINT:     SAME AS MODULE NAME
```

INLPMIBK

```
MODULE INLPMIBK
DESCRIPTIVE NAME: MIGRATION BACKUP MESSAGE PHASE

FUNCTION =       DISPLAY HELP INFORMATION TO
                 USE LIBRARIAN COMMAND 'BACKUP'

INPUT =          NONE

OUTPUT =         MESSAGES ON SYSLST

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPMICO

```
MODULE INLPMICO
DESCRIPTIVE NAME: CORGZ MIGRATION ROUTINE

FUNCTION =       COMPUTE THE LOAD ADDRESS AND LOAD THE LIBRARIAN
                 PHASE 'LIBR'.
                 PROVIDE MIGRATION-IDENTIFICATION STRING AS
                 PARM PARAMETER VIA REGISTER 1.
                 GIVE CONTROL TO LIBRARIAN.
                 TERMINATE MIGRATION SESSION AND RETURN TO
                 JCL.

INPUT =          NONE

OUTPUT =         NONE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPMICS

```
MODULE INLPMICS
DESCRIPTIVE NAME: CSERV MIGRATION ROUTINE

FUNCTION =       COMPUTE THE LOAD ADDRESS AND LOAD THE LIBRARIAN
                 PHASE 'LIBR'.
                 PROVIDE MIGRATION-IDENTIFICATION STRING AS
                 PARM PARAMETER VIA REGISTER 1.
                 GIVE CONTROL TO LIBRARIAN.
                 TERMINATE MIGRATION SESSION AND RETURN TO
                 JCL.

INPUT =          NONE

OUTPUT =         NONE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPMICV

```
MODULE INLPMICV
DESCRIPTIVE NAME: MIGRATION COPYSERV MESSAGE PHASE

FUNCTION = DISPLAY HELP INFORMATION AND TERMINATE THE JOBSTEP
           WHEN THE PHASE "COPYSERV" IS TO BE EXECUTED.

INPUT =          NONE

OUTPUT =         MESSAGES ON SYSLST,
                 RETURN CODE = 16

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPMIDS

```
MODULE INLPMIDS
DESCRIPTIVE NAME: DSERV MIGRATION ROUTINE

FUNCTION =       COMPUTE THE LOAD ADDRESS AND LOAD THE LIBRARIAN
                 PHASE 'LIBR'.
                 PROVIDE MIGRATION-IDENTIFICATION STRING AS
                 PARM PARAMETER VIA REGISTER 1.
                 GIVE CONTROL TO LIBRARIAN.
                 TERMINATE MIGRATION SESSION AND RETURN TO
                 JCL.

INPUT =          NONE

OUTPUT =         NONE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPMIGR

```
MODULE INLPMIGR
DESCRIPTIVE NAME: SELECT EXECUTION MODULES FOR MIGRATED
                  COMMANDS

FUNCTION =       GET LIBRARY INFORMATION FOR MIGRATED FROM-
                 AND TO-LIBRARIES OF TYPE PHASE,PROC,OBJ AND
                 SOURCE.
                 ADD MISSING COMMAND INFORMATION (LIB- AND
                 SUBLIB-NAMES,MEMBERTYPES) DEPENDING ON THE
                 MIGRATED FUNCTION.
                 CHECK THE COMMAND NAMES SPECIFIED IN THE
                 COMMAND CONTROL BLOCK(INLCCMDP) AND BRANCH
                 TO THE CORRESPONDING EXECUTION MODULE.

INPUT =         LIBRARIAN COMMUNICATION REGION INLCCOMR
                PARSED COMMAND CONTROL BLOCK INLCCMDP

OUTPUT =        RETURN CODE 8 FOR FAILURE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPMIMA

```
MODULE INLPMIMA
DESCRIPTIVE NAME: MAINT MIGRATION ROUTINE

FUNCTION =       COMPUTE THE LOAD ADDRESS AND LOAD THE LIBRARIAN
                 PHASE 'LIBR'.
                 PROVIDE MIGRATION-IDENTIFICATION STRING AS
                 PARM PARAMETER VIA REGISTER 1.
                 GIVE CONTROL TO LIBRARIAN.
                 TERMINATE MIGRATION SESSION AND RETURN TO
                 JCL.

INPUT =          NONE

OUTPUT =         NONE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPMIPD

```
MODULE INLPMIPD
DESCRIPTIVE NAME: MIGRATION PDZAP MESSAGE PHASE

FUNCTION = DISPLAY HELP INFORMATION AND TERMINATE THE JOBSTEP
           WHEN THE PHASE "PDZAP" IS TO BE EXECUTED.

INPUT =          NONE

OUTPUT =         MESSAGES ON SYSLST,
                 RETURN CODE = 16

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPMIPS

```
MODULE INLPMIPS
DESCRIPTIVE NAME: PSERV MIGRATION ROUTINE

FUNCTION =       COMPUTE THE LOAD ADDRESS AND LOAD THE LIBRARIAN
                 PHASE 'LIBR'.
                 PROVIDE MIGRATION-IDENTIFICATION STRING AS
                 PARM PARAMETER VIA REGISTER 1.
                 GIVE CONTROL TO LIBRARIAN.
                 TERMINATE MIGRATION SESSION AND RETURN TO
                 JCL.

INPUT =          NONE

OUTPUT =         NONE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPMIRE

```
MODULE INLPMIRE
DESCRIPTIVE NAME: MIGRATION RESTORE MESSAGE PHASE

FUNCTION =       DISPLAY HELP INFORMATION TO
                 USE LIBRARIAN COMMAND 'RESTORE'

INPUT =          NONE

OUTPUT =         MESSAGES ON SYSLST

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPMIRS

```
MODULE INLPMIRS
DESCRIPTIVE NAME: RSERV MIGRATION ROUTINE

FUNCTION =       COMPUTE THE LOAD ADDRESS AND LOAD THE LIBRARIAN
                 PHASE 'LIBR'.
                 PROVIDE MIGRATION-IDENTIFICATION STRING AS
                 PARM PARAMETER VIA REGISTER 1.
                 GIVE CONTROL TO LIBRARIAN.
                 TERMINATE MIGRATION SESSION AND RETURN TO
                 JCL.

INPUT =          NONE

OUTPUT =         NONE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPMISS

```
MODULE INLPMISS
DESCRIPTIVE NAME: SSERV MIGRATION ROUTINE

FUNCTION =       COMPUTE THE LOAD ADDRESS AND LOAD THE LIBRARIAN
                 PHASE 'LIBR'.
                 PROVIDE MIGRATION-IDENTIFICATION STRING AS
                 PARM PARAMETER VIA REGISTER 1.
                 GIVE CONTROL TO LIBRARIAN.
                 TERMINATE MIGRATION SESSION AND RETURN TO
                 JCL.

INPUT =          NONE

OUTPUT =         NONE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPMLCK

```
MODULE INLPMLCK
DESCRIPTIVE NAME: COMMON SUPERVISOR LOCKING ROUTINES

FUNCTION =
                  - INLPXMLK: LOCKS/UNLOCK   A MEMBER RESOURCE
                  - INLPSDTL: BUILDS DTL FOR SUBLIBRARY
                  - BUILDDTL: BUILDS DTL FOR LIB/SUBLIB
                  - INLPCMPT: COMPARES TIMESTAMPS
                  - CHKPHASE: CHECKS, IF MEMBER IS A PHASE
                  - INLPXMRS: BUILDS A MEMBER RESOURCE NAME
                              THIS ENTRY IS IN MODULE INLPSTOX
                              BECAUSE IT IS CALLED BY INLPXMLK

INPUT =           DEPENDANT FROM FUNCTION
                  R1 MUST CONTAIN ADRESS OF LRPL

OUTPUT =          DEPENDING OF FUNCTION:

 ENTRY POINT:
                  INLPSDTL, INLPXMLK, INLPCMPT

EXT.REFERENCES    INLPRESN
                  INLPXQNM
                  INLPXMSG
                  INLPXPRM

MACRO CALLS
                  GENDTL
                  LOCK
```

INLPMSG

```
MODULE INLPMSG
DESCRIPTIVE NAME: PREPROCESSOR FOR MESSAGE HANDLER

FUNCTION =       STORE MESSAGE NUMBER AND THE ADDRESSES OF THE
                 INSERTION TEXTS TO THE LAMB.
                 CALL THE MESSAGE HANDLER (INLPDIAG).

INPUT =          PARAMETERS:
                  1) MSGNUM  BIN(15)   MESSAGE NUMBER
                  2) UP TO 4 INSERTION TEXTS WITH VARIABLE
                     LENGTH:
                       INSERT1 CHAR( )   INSERTION 1
                       INSERT2 CHAR( )   INSERTION 2
                       INSERT3 CHAR( )   INSERTION 3
                       INSERT4 CHAR( )   INSERTION 4

OUTPUT =         NONE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPLDS

```
MODULE INLPLDS
DESCRIPTIVE NAME: LOW LEVEL SERVICE FOR ACCESS OF ALL OPEN
                  LIBRARIES

FUNCTION =        RETURNS NAMES OF ALL LIBRARIES IN LDT

INPUT =
                  LIBRARY CONTROL TABLES
                  WORKAREA
                  IF INLPLDS IS CALED THE FIRST TIME THE
                  FIRST WORD IN WORKAREA MUST BE HEX 0.
                  FOR A CONTINUATION REQUEST THIS WORD MUST
                  CONTAIN THE WORKPOINTER RETURNED BY THE
                  PREVIOUS CALL OF INLPLDS.

OUTPUT =          LIST OF NAMES OF ALL LIBRARIES IN LDT
                   STARTING AT OFFSET 5 FROM BEGINNING OF
                   WORKAREA.
           THE FIRST WORD IN THE WORKAREA CONTAINS THE CURRENT
           CURRENT POINTER IN THE LDT.
           IF THE END OF LDT HAS BEEN REACHED AND THERE IS
           STILL SPACE LEFT IN THE WORKAREA, THEN THE LAST
           ENTRY IN THE WORKAREA IS 'EOL'.
           THE RETURN CODE SHOWS IF A CONTINUATION REQUEST IS
           NECESSARY OR NOT.
           RETURN CODE IN REGISTER 15:
    0 ...   END OF LDT HAS BEEN REACHED. NAMELIST IS COMPLETE.
    8 ...   WORKAREA IS FULL BUT END OF LDT HAS NOT BEEN
            REACHES. CONTINUATION REQUEST POSSIBLE.
   16 ...   NO WORKAREA HAS BEEN SPECIFIED OR THE LENGTH OF
            THE WORKAREA IS SMALLER THAN THE LENGTH OF ONE
            ENTRY
                        INFORMATION AVAILABLE IN LBRLRET

 ENTRY POINT:     SAME AS MODULE NAME

EXT.REFERENCES

 MACROS USED:  LBRACCCB,ASYSCOM ,INLCLAMB

 MODULES CALLED:  INLPXPRM, INLPXTRC
```

INLPNOTE

```
MODULE INLPNOTE
DESCRIPTIVE NAME:  NOTE POSITION FOR SEQUENTIAL READ

FUNCTION =       TRANSFER INFORMATION ON CURRENT POSITION
                 IN MODULE TO USER PROVIDED AREA

INPUT =          LRPLPTR IN REGISTER 1

OUTPUT =         RETC000       - NORMAL RETURN
```

INLPOPEN

```
MODULE INLPOPEN
DESCRIPTIVE NAME: CONNECT PROGRAM TO (SUB-)LIBRARY CHAIN

FUNCTION =        BUILDS THE (SUB-)LIBRARY CHAIN TABLE
                  REQUIRED TO MANAGE ACCESS TO THE (SUB-)
                  LIBRARIES.
 =======> THE CHAIN OF (SUB-)LIBRARIES SPECIFIED WITH THE LRPL
          (LRPLTYPE, LRPLLUSE) IS RETRIEVED AS A TABLE
          (DSECT: INLCLBTB) OF LIBINFO ENTRIES (POINTERS TO
          LOT-, LDT-C-, LDT-P-, AND SDT-ENTRY). WHEN THE
          TABLE IS BUILT ITS ADDRESS IS INSERTED INTO THE LRPL
          (OTHERWISE 0).
          - THE SPACE OF INLCLBTB IS ALLOCATED WITH GETVIS.
          - THE CORRESPONDING FREEVIS REQUEST HAS TO BE DONE
            AT DISCONNECT-LIBRARY-TIME (INLPLDIS).
          - RATIONALE FOR THIS MODULE IS SAVINGS OF LBRACCES
            REQUESTS FOR FIND OR BUILD-LIST FUNCTIONS.

INPUT =           LIBRARY REQUEST PARAMETER LIST (INLCLRPL)

OUTPUT =          LIBRARY CHAIN TABLE INLCLBTB
                  RETURN CODES: 0.... SUCCESSFUL
                                8.... CHAIN IS EMPTY
                               16.... ERROR IN PARAMETER
                               20.... PROCESSING ERROR
                               (FEEDBACK CODES WITH 16 AND 20)

 ENTRY POINT:     SAME AS MODULE NAME
```

INLPOUT

```
MODULE INLPOUT
DESCRIPTIVE NAME:  STORE RECORDS INTO USER OUTPUT AREA AND
                   BRANCH TO USER OUTPUT EXIT (CALL INTERFACE)

FUNCTION =       THIS MODULE IS PART OF THE CALL INTERFACE.
                 IT STORES 1 RECORD WITH FORMATTED CONTENTS
                 TO THE USER'S OUTPUT AREA. - IF THE INPUT
                 PARAMETER 'DELAYED' IS SPECIFIED, THE RECORDS
                 ARE STORED SEQUENTIALLY INTO THE OUTPUT
                 AREA WITHOUT BRANCHING TO THE USER EXIT
                 ROUTINE. THE BRANCH TO THE USER EXIT IS DONE
                 WHEN THE AREA IS FILLED UP. - IF RECPTR ¬= 0
                 AND 'IMMEDIAT' IS SPECIFIED, THE BRANCH
                 TO THE USER EXIT IS DONE IMMEDIATELY AFTER
                 THE RECORD IS STORED INTO THE OUTPUT AREA.
                 - IF RECPTR = 0 AND 'IMMEDIAT' IS SPECIFIED,
                 THE BRANCH TO THE USER EXIT IS DONE IMMEDIATELY
                 WITHOUT STORING A RECORD TO THE OUTPUT AREA.

INPUT =          1ST PARAMETER: RECPTR (POINTER TO THE RECORD)
                 2ND PARAMETER: ACTION BIN(8) (0 = DELAYED,
                                               1 = IMMEDIATE)

OUTPUT =         RETURN CODES IN REG. 15:
                   0 = SUCCESSFUL
                   8 = OUTPUT AREA TOO SMALL FOR STORING 1
                       RECORD
                  12 = INVALID PARAMETERS

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPPIDT

```
MODULE NAME: INLPPIDT

DESCRIPTIVE NAME: PROCESS OPEN IDENTIFICATION TABLE

EXIT: REG14

FUNCTION:  THE OPEN IDENTIFICATION TABLE FOR A LIBRARY IS
           UPDATED OR SEARCHED.

 LOOKUP: SCANS IDT FOR AN ENTRY WITH SAME FILE-ID AND PHYSICAL
         UNIT (BAM), OR SAME CATALOG-ID AND FILE-ID(VSAM). IF
         ENTRY IS FOUND THE LDT IS LOOKED UP FOR AN EQUAL
         RESOURCE NAME; THE CORRESPONDING LDT, EDT AND DDT
         ENTRIES ARE PROVIDED FOR THE CALLER.
         IF THE IDT ENTRY OR THE LDT ENTRY IS MISSING
         RETURN CODE 4 IS PASSED BACK.

 INSERT: SCANS IDT FOR AN ENTRY WITH SAME FILE-ID AND PHYSICAL
         UNIT (BAM), OR SAME CATALOG-ID AND FILE-ID(VSAM). IF
         ENTRY IS NOT FOUND IT IS INSERTED.  FOR LIBRARIES IN
         VSAM MANAGED SPACE TWO ENTRIES ARE PROVIDED (THE
         ADDITIONAL ENTRY CONTAINING THE VSAM CATALOG-ID).
         IF THE IDT IS FULL, RETURN CODE 12 IS PASSED BACK TO
         THE INVOKER.

 DELETE: SCANS IDT FOR AN ENTRY WITH SAME RESOURCE NAME AND
         PHYSICAL UNIT. IF THE ENTRY IS FOUND, IT IS REMOVED.
         IF THE ENTRY BELONGS TO A LIBRARY IN VSAM-MNGD. SPACE
         AND IT WAS THE LAST ENTRY BELONGING TO A CATALOG THE
         THE CATALOG ENTRY IS ALSO REMOVED.
         IF THE ENTRY WAS NOT FOUND RETURN CODE 4 IS PASSED
         BACK TO THE INVOKER.

INPUT  : INLCPIDT

OUTPUT : LDT/EDT/DDT ENTRIES

RETURN CODES:
          0 = REQUEST HAS BEEN HONORED
          4 = ENTRY NOT FOUND
         12 = IDT IS FULL
         16 = PARAMETER INPUT ERROR WITH FEEDBACK CODE AND MSG

 ENTRY POINT:    SAME AS MODULE NAME
```

INLPPOIN

```
MODULE INLPPOIN
DESCRIPTIVE NAME:  REPOSITION FOR SEQUENTIAL READ

FUNCTION =       REPOSITION IN MEMBER, FOR SEQUENTIAL READ
                 ONLY, FROM USER PROVIDE NOTE INFORMATION

INPUT =          LRPLPTR IN REGISTER 1

OUTPUT =         RETC000          - NORMAL RETURN

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPPUN

```
MODULE INLPPUN
DESCRIPTIVE NAME:  PUNCH A 80-BYTE RECORD

FUNCTION =       USE CALL INTERFACE EXIT OR PUNCH A RECORD.
                 IF THE CALL INTERFACE IS NOT REQUESTED THEN:
                  OPEN THE SYSPCH DTF IF NOT YET DONE AND
                  SET 'OPEN' INDICATION IN THE LIBRARIAN COMREG.
                  MOVE PUNCH DATA TO SYSPCH OUTPUT AREA.
                  PUNCH THE RECORD.

INPUT =          PARAMETER:
                     1 PCLINE BDY(WORD),
                       3 PCCC CHAR(1),     ASA CONTROL CHAR.
                       3 PCLEN BIN(8),     LENGTH OF PUNCH DATA
                       3 PCDATA PTR(31) BDY(WORD,3)  ADDR. DATA

OUTPUT =         80-BYTE RECORD ON SYSPCH

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPPUTR

```
MODULE INLPPUTR
DESCRIPTIVE NAME: PERFORM PUT/GET RECORD

FUNCTION =       TRANSFER INFORMATION BETWEEN USER AND A
                 CONNECTED MEMBER IN THE LIBRARY
INPUT =          LRPLPTR IN REGISTER 1
OUTPUT =         RETC000     - NORMAL RETURN
                 RETC004     - SIGNAL LAST RECORD IN MEMBER
                 RETC008     - SIGNAL LAST RECORD TRUNCATED
                 RETC012     - FOR INLMGETR:
                               REQUEST OUTSIDE MEMBER SIZE
                             - FOR INLMPUTR:
                               LIBRARY FULL
                 RETC016     - EXTERNAL ERROR
                 RETC020     - INTERNAL ERROR
                               CONFLICTING ATTRIBUTES
                               INVALID REQUEST
                 RETC036     - MEMBER SPACE REUSED

 ENTRY POINT:    INLPPUTR
   PURPOSE:      TRANSFER INFORMATION BETWEEN USER WORK
                   AREA AND BUFFERS
                 REQUEST ALLOCATION OF LIBRARY SPACE FOR
                   BUFFERS TO BE WRITTEN
                 REQUEST WRITING OF ALLOCATED BUFFERS

                 FOR CONTINUOUS MEMBERS, THE WORK AREA CONTENT
                 AS IDENTIFIED BY FIELDS LRPLWAEA AND LRPLLNWA
                 IS MOVED TO THE MEMBER STARTING AT THE LRBA
                 VALUE GIVEN IN LRPLLRBA. IF THIS VALUE IS PAST
                 MEMBER HIGHEST LRBA, THE MEMBER PORTION IN
                 BETWEEN IS PATCHED WITH HEXADECIMAL ZEROS.

                 FOR NON CONTINUOUS MEMBERS, ONE OR MORE
                 RECORDS ARE ADDED TO THE MEMBERS, STARTING AT
                 THE CURRENT RELATIVE RECORD NUMBER.
                 FOR MULTIPLE RECORDS REQUESTS AND IF THE
                 RECORDS TO BE WRITTEN ARE ALREADY COMPRESSED,
                 A TWO BYTE LENGTH FIELD IS ASSUMED TO PRECEDE
                 EACH RECORD.

                 PREREQUISITE TO A WRITE REQUEST IS
                 A CONNECT MEMBER REQUEST.
```

```
ENTRY POINT:    INLPGETR
  PURPOSE:      READ CONNECTED MEMBER FROM LIBRARY INTO
                  BUFFERS
                TRANSFER INFORMATION BETWEEN BUFFERS AND
                  USER SPECIFIED AREA

                FOR CONTINUOUS MEMBERS, THE WORKABLE IS FILLED
                UP WITH THE MEMBER CONTENT, STARTING AT LRBA
                SPECIFIED IN LRPLLRBA AND UP TO THE LENGTH
                SPECIFIED IN LRPLLNWA.

                FOR NON CONTINUOUS MEMBERS AND LOCATE MODE,
                A DECOMPRESSED RECORD IS PRESENTED ADDRESSED
                BY POINTER LRPLWAEA.
                FOR MOVE MODE, ONE OR MORE RECORDS ARE MOVED
                TO THE AREA SPECIFIED IN LRPLWAEA. THEY MAY
                BE PRESENTED IN COMPRESSED OR DECOMPRESSED
                FORMAT, AS REQUESTED. IF MORE THAN ONE RECORD
                ARE REQUESTED IN COMPRESSED FORMAT, THEY ARE
                PRECEDED BY A TWO BYTES LENGTH FIELD.

  LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPQNAM

```
MODULE INLPQNAM
DESCRIPTIVE NAME: BUILD AN EXTERNAL QUALIFIED NAME STRING

FUNCTION =       BUILD AN EXTERNAL QUALIFIED NAME STRING FROM
                 SINGLE 8 BYTE NAMES. UP TO 4 QUALIFICATIONS
                 ARE ALLOWED (E.G. 'NAME1.NAME2.NAME3.NAME4').

INPUT =          PARAMETERS:
                  1) 1 QNAME          AREA FOR OUTPUT STRING
                       2 QLEN  BIN(8) LENGTH OF OUTPUT TEXT
                       2 QTEXT CHAR(34) MAX. TEXT AREA
                  2) UP TO 4 NAMES (EACH 8 BYTES LONG)
                       1. NAME1
                       2. NAME2
                       3. NAME3
                       4. NAME4

OUTPUT =       USER AREA QNAME IS FILLED WITH QLEN AND QTEXT

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPRALU

```
MODULE INLPRALU
DESCRIPTIVE NAME: DYNAMIC ASSIGN OF LOGICAL TAPE UNIT

FUNCTION =       DYNAMICALLY ASSIGN/UNASSIGN LOGICAL UNIT FOR
                 TAPE DEVICE FOR BACKUP/RESTORE

 ENTRY POINT :  INLPRALU

                 DYNAMICALLY ASSIGN LOGICAL UNIT FOR
                 TAPE DEVICE

INPUT =          INLCUNV  - PHYSICAL TAPE UNIT IN (INLCCMDP)
                            PARSED COMMAND CONTROL BLOCK
OUTPUT =         SYSNO    - LOGICAL UNIT

                 RETURN CODES
                   0 ... ASSIGN SUCCESSFUL
                   4 ... AN EXTERNAL OR INTERNAL ERROR OCCURRED
                         AND A MESSAGE IS ISSUED

 ENTRY POINT :  INLPRUAS

                 FREE LOGICAL UNIT FOR TAPE DEVICE

                 RETURN CODES
                   0 ... UNASSIGN SUCCESSFUL
                   4 ... AN EXTERNAL OR INTERNAL ERROR OCCURRED
                         AND A MESSAGE IS ISSUED
```

INLPRCLM

```
MODULE INLPRCLM
DESCRIPTIVE NAME: RECLAIM DELAYED LIBRARY SPACE

FUNCTION =        THE LIBRARY BLOCKS OF THE RECLAMATION CHAIN
                  OF A SPECIFIC SUBLIBRARY OR OF THE
                  RECLAMATION CHAINS OF ALL SUBLIBRARIES WITHIN
                  A LIBRARY ARE ADDED AS AVAILABLE LIBRARY
                  BLOCKS TO THE FREE-SPACE-MAP.
                  IF NECESSARY, THE SECOND LEVEL DIRECTORY
                  (SLD) FOR THE CORRESPONDING SUBLIBRARY,
                  THE HIGHEST INDEX PRBA AND THE RECLAIM
                  COUNTER (SDTE) ARE UPDATED.
                  - THIS FUNCTION RUNS WITH STORAGE KEY 0.
                  - DURING RECLAMATION THE LIBRARY HEADER IS
                  KEPT LOCKED.
                  - THE RECLAMATION CHAIN POINTER IN THE
                  SUBLIBRARY DESCRIPTOR IS FIRST RESET BEFORE
                  UPDATING THE BITMAP OF THE LIBRARY TO AVOID
                  FREEING LIBRARY BLOCKS TWICE IN CASE OF
                  ABNORMAL TERMINATION.

INPUT =           LIBINFO

OUTPUT =          UPDATED FREE-SPACE-MAP, SLD, SDTE
                  RETURN CODES, MESSAGES
                       0 ... FUNCTION SUCCESSFULLY COMPLETED
                      16 ... EXTERNAL ERROR WITH FEEDBACK CODE
                      20 ... INTERNAL ERROR WITH FEEDBACK CODE

 ENTRY POINT:     SAME AS MODULE NAME
```

INLPREAD

```
MODULE INLPREAD
DESCRIPTIVE NAME:  READ COMMANDS AND INPUT DATA

FUNCTION =       CHECK IF THE LIBRARIAN IS EXECUTED FROM SYSLOG
                 OR FROM BATCH AND DO THE CORRESPONDING READ
                 REQUESTS ON SYSLOG/SYSIPT.
                 DO CALL-INTERFACE EXIT PROCESSING IF REQUESTED
                 INSTEAD OF ISSUING I/O-S (BATCH MODE ONLY).
                 OPEN THE SYSIPT DTF IF REQUIRED.
                 CHECK FOR EOF ON INPUT AND INDICATE IT IN THE
                 LIBRARIAN COMMUNICATION REGION.

INPUT =          80/81-BYTE RECORD FROM SYSIPT/SYSLOG OR
                 80 BYTE RECORDS FROM USER EXIT AREA

OUTPUT =         80-BYTE RECORD IN THE PARTITION I/O-AREA

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPREL

```
MODULE INLPREL
DESCRIPTIVE NAME: RELEASE-SPACE COMMAND PROCESSOR

FUNCTION =       PROCESS RELEASE-SPACE STATEMENT
                 - LABEL INITREL:
                    DO PREPARATORY WORK FOR RELEASE PROCESSING
                 - LABEL SELREL:
                    SELECT THE PROCESSING FUNCTIONS:
                      LIBREL - PROCESS WHOLE LIBRARIES
                      SUBREL - PROCESS SUBLIBRARIES
                 - LABEL FINREL:
                    FINISH RELEASE PROCESSING AND RETURN

INPUT =          LIBRARIAN COMMUNICATION REGION INLCCOMR
                 PARSED COMMAND CONTROL BLOCK INLCCMDP

OUTPUT =         RETURN CODE 0 - SUCCESSFUL COMPLETION
                 RETURN CODE 4 - MESSAGE L137I
                 RETURN CODE 8 - ANY OTHER FAILURE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPREN

```
MODULE INLPREN
DESCRIPTIVE NAME: RENAME COMMAND PROCESSOR

FUNCTION =       PROCESS RENAME STATEMENT AND DO SEMANTIC
                 CHECKS:
                 - LABEL INITREN:
                    DO PREPARATORY WORK FOR RENAME PROCESSING
                 - LABEL SUBLREN:
                    RENAME SUBLIBRARY
                 - LABEL MEMBREN:
                    RENAME MEMBER
                 - LABEL FINREN:
                    FINISH RENAME PROCESSING AND RETURN

INPUT =          LIBRARIAN COMMUNICATION REGION INLCCOMR
                 PARSED COMMAND CONTROL BLOCK INLCCMDP

OUTPUT =         RETURN CODE 0 = SUCCESSFULLY COMPLETED
                 RETURN CODE 8 = ANY FAILURE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPRESN

```
MODULE NAME: INLPRESN

DESCRIPTIVE NAME: BUILD SUBLIBRARY RESOURCE NAME

EXIT: REG14

FUNCTION:  THE RESOURCE NAME FOR A SUBLIBRARY IS BUILT.

INPUT  : LAMBPTR,LDTEPTR,PRBA,SUBLIBRARY NAME

OUTPUT : RESOURCE NAME OF SUBLIBRARY

RETURN CODES
         0 = REQUEST HAS BEEN HONORED
        20 = PRBA OUTSIDE LIBRARY EXTENT(S)

 ENTRY POINT:    SAME AS MODULE NAME
```

INLPREST

```
MODULE INLPREST
DESCRIPTIVE NAME: RESTORE - MAIN MODULE

FUNCTION =       RESTORE MAIN MODULE

                 - INITIALIZATION OF RESTORE
                 - IF RESTORE 'OLD' LIBRARY
                     SPECIFICATION LIST :
                     LOAD PHASE  'LIBRROLD'
                     CALL RESTORE LIBRARIES (INLPROLD)
                 - ALLOCATE  SPACE FOR TAPE AND DISK BUFFERS
                 - OPEN TAPE
                 - POSITION TAPE TO BACKUP FILE HEADER AND
                   CHECK IF TAPE IS PRODUCED WITH VSE/AF 2.1
                 - PROCESS THE BACKUP FILE AND IF 'ID= ' IS
                   SPECIFIED ALL FOLLOWING BACKUP FILES ON
                   THE INPUT TAPE :
                   - CALLS DEPENDENT ON THE SPECIFICATION LIST
                      OR IF 'RESTORE  ' WAS SPECIFIED DEPENDENT
                      ON THE CONTENTS OF THE BACKUP FILE :
                     RESTORE LIBRARIES (INLPRELB) OR
                     RESTORE SUBLIBRARIES (INLPRESL) OR
                     RESTORE MEMBERS (INLPREMB)

INPUT =          LIBRARY-SPECIFICATION-LIST OR
                 SUBLIBRARY-SPECIFICATION-LIST OR
                 MEMBER-SPECIFICATION-LIST OR
                 OLDLIB-SPECIFICATION-LIST OR

                 AND SCAN=YES OR SCAN=NO

OUTPUT =         THE LIBRARIES OR SUBLIBRARIES OR MEMBERS
                 FOUND ON THE INPUT TAPE RESTORED

                 INFORMATION ON SYSLOG/SYSLST INDICATING
                 WHETHER THE LIBRARIES,SUBLIBRARIES OR
                 MEMBERS CONTAINED IN THE SPECIFICATION-
                 LIST ARE RESTORED OR NOT

                 RETURN CODES
                   0 ... NORMAL EXIT :
                            CRGSAVRC = 0 :
                                RESTORE SUCCESSFUL
                            CRGSAVRC = 8 AND MESSAGE:
                                IF A LIBRARY, SUBLIBRARY
                                OR MEMBER IS SKIPPED
```

```
 ENTRY POINT:    INLPRSHF

FUNCTION =    READ THE NEXT HISTORY FILE BLOCK FROM
              THE INPUT TAPE INTO AN AREA PROVIDED
              BY MSHP.

INPUT =       FUNCTION CODES
                1 ... READ BLOCK FROM TAPE
                2 ... ERROR DURING MSHP PROCESSING:
                          CLOSE INPUT TAPE

              ADDRESS AND LENGTH OF HISTORY FILE BLOCK

OUTPUT =      HISTORY FILE BLOCK (HFBLKADR)
              LENGTH OF BLOCK (HFBLKLEN)

              RETURN CODES
                0 ... NORMAL EXIT :
                          READ OF HISTORY FILE
                          BLOCK SUCCESSFUL
                4 ... END OF HISTORY FILE :
                          TAPE IS POSITIONED NEW FOR
                          FOLLOWING RESTORE COMMAND ISSUED
                          BY MSHP WITHOUT SCAN=YES
                8 ... HISTORY FILE CONTAINS BLOCK WHICH
                          IS GREATER THAN BLOCK LENGTH :
                          INPUT TAPE IS CLOSED
               16... INVALID FUNCTION CODE



 ENTRY POINT:    SAME AS MODULE NAME

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPROLD

```
MODULE INLPROLD
DESCRIPTIVE NAME: RESTORE OLDLIB

FUNCTION =     RESTORE 'OLD' LIBRARIES OF DOS/VS RELEASE 34
               OR VSE/AF1-AF3.5

               THIS FUNCTION RESTORES PRIVATE LIBRARIES ON TAPES
               CREATED WITH THE BACKUP UTILITY OF DOS/VS R34
               OR OF VSE/AF1 - AF3.5
               THE LIBRARIES ARE RESTORED INTO THE TARGET
               SUBLIBRARIES SPECIFIED IN THE OLDLIB-
               SPECIFICATION-LIST. THE TARGET LIBRARY MUST
               ALREADY EXIST. IF THE TARGET SUBLIBRARY
               ALREADY EXISTS IT IS REPLACED.

               THE LIBRARY FILENAME FOR EACH LIBRARY FOUND
               ON THE BACKUP TAPE IS COMPARED WITH THE
               LIBARY FILENAMES IN THE OLDLIB NAME LIST
               ADDRESSED BY INLCCMDP. THE LIBRARY IS RESTORED
               IF FOUND IN THE LIST.

               RESTORE LIBRARY (RESTOLDL):
               - IF LIBRARY NAME CHANGES:
                 - ADD TEMP. LIB ENTRY TO THE LIBRARIAN
                   TABLES (LBRACCES)
               - CONNECT THE SUBLIBRARY  (INLMSCON)
               - RESTORE SUBLIBRARY (RESTSUBL):
                - ADD TEMP. LIB/SUBLIB ENTRY TO THE LIBRARIAN
                  TABLES (LBRACCES) WITH DEFINE(OLD)
                  TO CHECK IF SUBLIBRARY EXISTS ALREADY
                - IF SUBLIBRARY EXISTS:
                  - CONNECT TO OLD SUBLIBRARY (INLMSCON)
                    (RESTORE OLDLIB WORKS ADDITIVE)
                - ELSE:
                  - ADD TEMP. LIB/SUBLIB ENTRY TO THE LIBRARIAN
                    TABLES (LBRACCES) WITH DEFINE(NEW)
                  - CONNECT THE NEW SUBLIBRARY (INLMSCON)
                - DO FOR ALL MEMBERS FOUND FOR THIS SUBLIBRARY
                  ('OLD' LIBRARY) ON TAPE:
                    - PROCESS MEMBER DIRECTORY RECORD (PROCDIRE)
                        - CONNECT TO MEMBER
                    - RESTORE MEMBER (RESTMBR)
                    - DISCONNECT FROM MEMBER (DISCMBR)
                - STOW MEMBERS CONTAINED STILL IN THE STOW
                  TABLE FOR THE SUBLIBRARY (INLPMSTW)
                - DISCONNECT FROM LIBRARY (INLMLDIS)
```

```
       NOTE: IF THE TARGET LIBRARY IS UNIQUELY USED (I.E NO
             ACCESS IS ACTIVE FOR ANOTHER TASK, THE LIBRARY
             IS NOT ON A VOLUME WHICH IS SHARED ACROSS
             CPUS AND NO LIBDEF IS SPECIFIED) THE LIBRARY
             IS LOCKED (THE "NO ACCESS ALLOWED" FLAG IS
             TURNED ON) TO IMPROVE THE PERFORMANCE OF
             THE RESTORE PROCESS.

INPUT =          OLDLIB-SPECIFICATION-LIST

OUTPUT =         EACH LIBRARY FOUND ON TAPE AND CONTAINED IN
                 THE OLDLIB-SPECIFICATION-LIST IS RESTORED TO
                 THE TARGET SUBLIBRARY

                 INFORMATION ON SYSLOG/SYSLST INDICATING
                 WHETHER THE LIBRARY ARE RESTORED OR NOT

                 RETURN CODES
                   0 ... NORMAL EXIT:
                             CRGRETCD = 0:
                               IF RESTORE SUCCESSFUL
                             CRGRETCD = 4 IF A MBR NOT REPLACED
                             CRGRETCD = 8 AND A MESSAGE
                               IF A LIBRARY IS NOT RESTORED
                   4 ... AN INTERNAL OR EXTERNAL ERROR
                         WITH FEEDBACK CODE

 ENTRY POINT:    SAME AS MODULE NAME

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPSCON

```
MODULE INLPSCON
DESCRIPTIVE NAME: CONNECT TO SUBLIBRARY

FUNCTION = MAKES SUBLIBRARY AVAILABLE FOR ACCESS
           POSITIONS TO START OF MEMBER DIRECTORY
  ======> THE CONNECTION STRING (LACB-SACB) IS ATTACHED TO THE
          LRPL NO MATTER, WHETHER THE SAME OR ANOTHER LIBRARY
          OR SUBLIBRARY WAS CONNECTED, OR NO CONNECTION WAS
          ACTIVE.
          - IF A MEMBER CONNECTION PREVIOUSLY IS ACTIVE, A
          RETURN CODE OF 16 IS PASSED BACK.
          - MISSING ACCESS CONTROL BLOCKS (LACB OR SACB) ARE
          ALLOCATED, MISSING CONNECTIONS TO THE CORRECT LIBRARY
          AND SUBLIBRARY (GIVEN IN PARAMETER LIBINFO) ARE DONE.
          - AFTERWARDS, THE LRPL POINTS TO THE INITIALIZED SACB
          OF THE REQUESTED SUBLIBRARY.

INPUT =    LIBRARY REQUEST PARAMETER LIST INLCLRPL

OUTPUT =   MESSAGES, RETURN CODES
           SUBLIBRARY ACCESS CONTROL BLOCK (INLCSACB)
           RETURN CODES:
                  0 ... SUCCESSFUL
                  4 ... SUCCESSFUL, SUBLIB ALREADY CONNECTED
                 12 ... LIBRARY IS FULL (FOR CONNECT=NEW)
                 16 ... EXTERNAL ERROR WITH FEEDBACK CODE
                 20 ... INTERNAL ERROR WITH FEEDBACK CODE
                 32 ... SECURITY VIOLATION

 ENTRY POINT: SAME AS MODULE NAME
```

INLPSDIS

```
MODULE INLPSDIS
DESCRIPTIVE NAME: DISCONNECT FROM SUBLIBRARY

FUNCTION =        MAKES SUBLIBRARY UNAVAILABLE FOR ACCESS
                  LIBRARY REMAINS ACCESSIBLE
  =======> IF A SUBLIB IS CONNECTED, IT WILL BE DISCONNECTED,
           I.E. IF A CONTROL BLOCK STRING EXISTS WHICH CONTAINS
           THE SACB (LACB-SACB-MACB OR LACB-SACB) IT WILL BE
           REDUCED TO LACB. IF NO SUBLIBRARY IS CONNECTED, RET.
           CODE 8 IS PASSED BACK.
           - IF A MEMBER IS CONNECTED,ANY BUFFERS BEING CHANGED
           ARE WRITTEN TO THE LIBRARY. THE MACB AND
           THE SACB ARE DECHAINED AND FREED.
           - IF A SUBLIBRARY IS CONNECTED, THE SACB IS FREED.

INPUT =           LIBRARY REQUEST PARAMETER LIST (INLCLRPL)

OUTPUT =          MESSAGES, RETURN CODES
                      0 ... SUCCESSFULLY
                      8 ... SUBLIBRARY WAS NOT CONNECTED
                     16 ... EXTERNAL ERROR WITH FEEDBACK CODE
                     20 ... INTERNAL ERROR WITH FEEDBACK CODE

 ENTRY POINT:     SAME AS MODULE NAME
```

INLPSDL

```
MODULE INLPSDL
DESCRIPTIVE NAME: MAINTAIN SYSTEM DIRECTORY LIST (SDL)

FUNCTION =        BUILD AND UPDATE SYSTEM DIRECTORY LIST (SDL)
                  AND SHARED VIRTUAL AREA (SVA)
                  UPDATE LIBRARY POINTER TABLE (ADDRESS OF
                  $IJBLBR).
-------------------------------------------------------------

            I N T E R F A C E S

-------------------------------------------------------------
CALLER    :  JCL FOR SET SDL
STOWTABLE :  CONSISTS ONLY OF ENTRIES AFTER THE ACTUAL
             SDL.
TYPE BYTE :  'J'.
OTHER INT.:  'DSVAENT' IN SVA HEADER GIVES THE NUMBER OF
             ENTRIES ALREADY IN THE SDL.
FUNCTION  :  INLPSDL SORTS AND COMPLETES THE SDL
             WITH PHASE INFORMATION AND LOADS THE
             REQUESTED PHASES INTO THE SVA,IF POSSIBLE.
-------------------------------------------------------------
CALLER    :  LIBRARIAN WHILE UPDATING THE LIBRARY DIRECTORY
             (INLPSTOW) OF IJSYSRS.SYSLIB
STOW TABLE:  ONLY PHASE ENTRIES
TYPE BYTE :  'A'(ADD), 'D'(DELETE), 'R'(RENAME)
OTHER INT.:
FUNCTION  :  INLPSDL UPDATES THE SDL.
-------------------------------------------------------------
CALLER    :  IPL.
STOW TABLE:  IDENTICAL WITH SDL, SDL IS ALREADY SORTED
             AND COMPLETE.
TYPE BYTE :  'I'/'O' (ONLINE IPL) or 'S'/'T' (STAND-ALONE IPL)
OTHER INT.:  HEADER OF SVA.
FUNCTION  :  INLPSDL LOADS THE REQUESTED PHASES INTO
             THE SVA.
-------------------------------------------------------------
```

```
INPUT =           STOWTABLE
                  BYTE 0 OF ADDRESS OF SVA IN SYSCOM
                  SVA HEADER
                  SYSTEM DIRECTORY LIST (SDL)

OUTPUT =          UPDATED
                    BYTE 0 OF SVA ADDRESS IN SYSCOM
                    SVA HEADER
                    SYSTEM DIRECTORY LIST
                    ADDRESS OF $IJBLBR IN LPT
                  RETURN CODES, MESSAGES
                  0 ... ALL ENTRIES OF STOW TABLE PROCESSED
                  8 ... SDL IS FULL
                 16 ... EXTERNAL ERROR WITH FEEDBACK CODE
                 20 ... INTERNAL ERROR WITH FEEDBACK CODE

 ENTRY POINT:     16 BYTES BEHIND MODULE BEGIN BECAUSE OF
                  MODULE IDENTIFICATION AT MODULE START.
```

INLPSEOT

```
MODULE  INLPSEOT
DESCRIPTIVE NAME: LIBRARIAN END-OF-TASK HANDLER

ENTRY POINT: LIBRSEOT

EXIT: REG14

FUNCTION =
THIS ROUTINE IS INVOKED BY THE SUPERVISOR TERMINATOR ROUTINE
WHICH ENSURES "DELAYED CANCEL".
TO AVOID DEADLOCKS AN "UNLOCK ALL" REQUEST IS GIVEN BEFORE
REQUESTING ANY LOCKING SERVICES.
IF THIS ROUTINE IS INVOKED FOR A SUBTASK, THE SVA SUBPOOLS
"INLC" AND "INLG" ARE REQUESTED TO BE FREED.
MODULE IJBCTUPD OF PHASE $IJBLBR IS INVOKED TO PROCESS
LIBRARY CONTROL TABLES CLEAN-UP.
BECAUSE THIS MODULE IS CALLED WITHOUT ANY SAVE AREA, SERIAL
REUSABILITY IS ESTABLISHED BY GATING THE CODE.
FOR THIS REASON KEY ZERO MUST BE PROVIDED.

INPUT =
NONE.

OUTPUT =
MESSAGES ON SYSLOG.

RETURN CODES: NONE (ERROR CODES IN REQUEST LIST)

MACROS USED:  INLCLAMB,CRGSPID,FREEVIS,LBRCTUPD,INLMLAMB
              GETFLD,ASYSCOM,LOCK,UNLOCK

MODULES CALLED: $IJBLBR-IJBCTUPD
```

INLPSLD

```
MODULE INLPSLD
DESCRIPTIVE NAME: BUILD SECOND LEVEL DIRECTORY (SLD)

FUNCTION =        BUILD A SECOND LEVEL DIRECTORY (SLD) FOR
                  PHASES (CORRESPONDS TO MEMBER INDEX LEVEL 2)
                  IN-STORAGE FOR FAST FETCHING OF PHASES
 ======> A COPY OF MEMBER INDEX LEVEL 2 OF THE CONNECTED SUBLIB
         IS PROVIDED IN THE SYSTEM GETVIS AREA, FOR PHASE ONLY,
         TO HAVE A FAST FETCH ACCESS TO     PHASE DIRECTORY
         ENTRIES IN THE SUBLIBRARY.

INPUT =           REQUEST PARAMETER LIST (INLCLRPL)
                  SUBLIBRARY DEFINITION TABLE ENTRY (SDTE)
                  NUMBER OF MEMBERS IN SUBLIBRARY

OUTPUT =          SECOND LEVEL DIRECTORY
                  MESSAGES, RETURN CODES
                      0 ... FUNCTION SUCCESSFULLY COMPLETED
                     16 ... EXTERNAL ERROR WITH FEEDBACK CODE
                     20 ... INTERNAL ERROR WITH FEEDBACK CODE

 ENTRY POINT:     SAME AS MODULE NAME
```

INLPSLIB

```
MODULE INLPSLIB
DESCRIPTIVE NAME: DELETE / RENAME SUBLIBRARY DIRECTORY ENTRY

FUNCTION =        RENAME A SUBLIBRARY DIRECTORY ENTRY IN-PLACE
                  OR DELETE A SUBLIBRARY DIRECTORY ENTRY.
                  IF A LIBRARY BLOCK BECOMES FREE IT IS
                  ENTERED INTO THE FREE SPACE MAP.
  ======> 1. MF=DELETE, LEVEL=LIB
          THE LIBRARY WHICH IS CONNECTED IS DELETED.
          - WITH A STOW REQUEST, ONLY ONE LIBRARY    CAN BE
          DELETED.
          - THIS LIBRARY    HAS TO BE UNIQUELY ASSIGNED TO THE
          REQUESTOR.
          - THE LIBRARY NAME IN THE LIBRARY DESCRIPTOR GETS THE
          LABEL "DELETED".
          - THE REQUESTOR NEEDS ALTER ACCESS RIGHT TO THE LIB.
          2. MF=DELETE, LEVEL=SUBLIB
          THE SUBLIBRARY GIVEN IN THE STOW TABLE IS DELETED FROM
          THE CONNECTED LIBRARY.
          - WITH A STOW REQUEST, ONLY ONE SUBLIBRARY CAN BE
          DELETED.
          - THIS SUBLIBRARY HAS TO BE UNIQUELY ASSIGNED TO THE
          REQUESTOR.
          - ALL MEMBERS OF THE SUBLIBRARY ARE DELETED AND THE
          LIBRARY BLOCKS IMMEDIATELY FREED.
          - BY DELETING THE SUBLIBRARY DESCRIPTOR A LIB. BLOCK
          MERGE IS DONE IF THE FOLLOWING LIBRARY BLOCK (THE LB
          LOGICALLY FOLLOWING THE BLOCK WITH THE SUBLIBRARY TO
          BE DELETED) FITS INTO THE CURRENT LIBRARY BLOCK.
          - THE REQUESTOR NEEDS ALTER ACCESS RIGHT TO THE
          SUBLIBRARY TO BE DELETED.
          3. MF=RENAME, LEVEL=SUBLIB
          THE REQUESTED SUBLIBRARY IS RENAMED IN PLACE IF IT
          EXISTS, OTHERWISE A RETURN CODE IS GIVEN.
          - THE REQUESTOR NEEDS ALTER ACCESS RIGHT TO THE
          SUBLIBRARY TO BE RENAMED.
          4. MF=EMPTY,  LEVEL=SUBLIB
          THE SUBLIBRARY GIVEN IN THE STOWTABLE IS EMPTIED (ALL
          MEMBERS ARE DELETED, BUT THE SUBLIBRARY STILL EXISTS)
          - WITH A STOW REQUEST, ONLY ONE SUBLIBRARY CAN BE
          EMPTIED.
          - THIS SUBLIBRARY GETS A NEW TIME-STAMP.
          - ALL MEMBERS OF THE SUBLIBRARY ARE DELETED AND THE
          LIBRARY BLOCKS FREED ACCORDING TO DELAYED SPACE
          RECLAMATION RULES.
          - THE REQUESTOR NEEDS UPDATE ACCESS RIGHT TO THE
          SUBLIBRARY TO BE EMPTIED.
```

```
INPUT =           STOW TABLE
                  LIBRARY REQUEST PARAMETER LIST (LRPL)
                  SUBLIBRARY DIRECTORY

OUTPUT =          UPDATED SUBLIBRARY DIRECTORY
                  MESSAGES, RETURN CODES
                   0 ... REQUEST HAS BEEN SUCCESSFULLY HONOURED
                   4 ... FUNCTION (PARTIALLY) FAILED (SEE
                         RETURN INFORMATION IN STOWTAB ENTRIES,
                         MACRO INLMSTOW)
                  16 ... EXTERNAL ERROR WITH FEEDBACK CODE
                  20 ... INTERNAL ERROR WITH FEEDBACK CODE
                  32 ... SECURITY VIOLATION

 ENTRY POINT:     SAME AS MODULE NAME
```

INLPSRCH

```
MODULE INLPSRCH
DESCRIPTIVE NAME: SEARCHES SPECIFIED MEMBER IN SPECIFIED
                  SUBLIBRARIES

FUNCTION =        PROCESS SEARCH     STATEMENT AND DO SEMANTIC
                  CHECKS:
                  - LABEL PRSRCH:
                     DO PREPARATORY WORK FOR SEARCH PROCESSING
                  - LABEL SELSRCH:
                     SELECT THE PROCESSING FUNCTIONS AND CALL
                     APPROPRIATE SUBROUTINES
                  - LABEL ERR1:
                     DOES THE ERROR PROCESSING
                  - LABEL LISTCHAIN:
                     PRODUCES A LIST OF NAMES OF SUBLIBRARIES
                     WHICH ARE CONTAINED IN THE SPECIFIED
                     CHAIN.
                  - LABEL REDOUB:
                     DELETES ALL MULTIPLE ENTRIES OUT OF
                     THE LIST OF SUBLIBRARY NAMES
                  - LABEL PROLI:
                     PROCESS A LIST OF SUBLIBRARY NAMES
                  - LABEL LIBSEARCH:
                     PROCESS A LIST OF LIBRARY NAMES
                  - LABEL LISTFIND:
                     CONNECTS SPECIFIED SUBLIBRARY AND CALLS
                     FINDM
                  - LABEL FINDM:
                     SEARCHES MEMBER IN ONE SUBLIBRARY AND ADDS
                     ALL NECESSARY INFORMATION INTO THE VARIOUS
                     LISTS
                  - LABEL PRINTL:
                     PRINTS THE RESULT OF SEARCH FROM THE LISTS
                     PRODUCED BY FINDM

INPUT =           LIBRARIAN COMMUNICATION REGION INLCCOMR
                  PARSED COMMAND CONTROL BLOCK INLCCMDP

OUTPUT =          RESULTS OF SEARCH ON SYSLST/SYSLOG
                  RETURN CODE  IN CRGRETCD
                  RETURN CODE 0 - SEARCH PROCESSED
                  RETURN CODE 4 - FUNCTION BYPASSED
                  RETURN CODE 8 - SEARCH TERMINATED


NOTES             |
```

```
 DEPENDENCIES=   THE COMMAND BLOCK INLCCMDP AND THE LIBRARIAN
                 COMMUNICATION REGION INLCCOMR MUST BE PREPARED
                 BY THE CALLER

 ENTRY POINT:    SAME AS MODULE NAME

EXT.REFERENCES

 ROUTINES=       INLPEXIT
                 INLPGST
                 INLPGSTI
                 INLPMSG
                 INLPPUN
                 INLPQNAM
                 INLPWTP
                 INLPCACK (ENTRYPOINT IN INLPMAIN)
                 INLPCANC (ENTRYPOINT IN INLPMAIN)

 DATA AREAS =    |

 CNTL.BLOCK =    INLCCMDP
                 INLCCOMR
                 INLCDENT
                 INLCLAMB
                 INLCLRPL

 TABLES=         LIBRARIAN CONTROL TABLES USED BY MACRO
                 LBRACCES
                 STOW TABLE (INLCSTOH, INLCLARG)

 MACROS=         LBRACCES
                 INLMLAMB
                 INLMMCON
                 INLMLDIS
                 INLMMDIS
                 INLMSDIS
                 INLMLSIM
                 INLMLRPL
                 INLMBLDL
                 INLMGETR
                 INLMPRAS
                 AVRLIST
                 GETVCE
                 GETVIS
                 GETFLD
                 GETIME
              BUILDING BLOCK BPQLST2

REMARKS =     BUILDING BLOCK USES PARTITION SPACE FOR LISTS
```

INLPSYNA

```
MODULE INLPSYNA
DESCRIPTIVE NAME: SYNTAX CHECK FOR COMMAND NAMES

FUNCTION =       INITIALIZE THE SCAN CONTROL BLOCK INLCSCAN USED
                 BY THE SCAN ROUTINE INLPSYNX.
                 PROCESS THE ACCESS COMMAND IF SPECIFIED IN
                 THE // EXEC PARM= STATEMENT.
                 READ STATEMENT FROM SYSIPT/SYSLOG AND POSITION
                 TO FIRST NON BLANK ITEM.
                 PRINT THE STATEMENT LINE.
                 CHECK THE COMMAND NAME BY THE TABLES INLTPARS
                 OR INLTMIGR (MIGRATION COMMANDS).
                 INITIALIZE THE PARSED COMMAND CONTROL BLOCK
                 (INLCCMDP) AND CALL INLPSYPA TO PARSE THE
                 CORRESPONDING COMMAND PARAMETERS.
                 PROCESS THE STATEMENTS:  "ON $RC ..",
                 "GOTO ..", AND "/. LABEL".
                 RETURN TO INLPMAIN AFTER EOF FOR INPUT.

INPUT =          INLTPARS, INLTMIGR

OUTPUT =         COMMAND NAME IN INLCCMDP
                 RETURN CODE  0 = SUCCESSFULLY COMPLETED
                 RETURN CODE 16 = MESSAGE L110I
                 RETURN CODE  8 = ANY OTHER FAILURE (SET BY
                                                    INLPEXIT)

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPSYNX

```
MODULE INLPSYNX
DESCRIPTIVE NAME: DO SEVERAL SCAN FUNCTIONS IN I/O-AREA

FUNCTION = THIS ROUTINE IS TO PROVIDE LOW LEVEL SYNTAX SUPPORT
           FOR THE SYNTAX MODULES INLPSYNA AND INLPSYPA.

           THE ROUTINE MUST FIRST BE CALLED VIA ENTRY INLSCNIN
           TO INITIALIZE THE SCAN CONTROL BLOCK.

           A NUMBER OF ENTRIES ARE PROVIDED TO RECOGNIZE
           VARIOUS SYNTACTIC ENTITIES IN THE STRING TO BE
           SCANNED.

           THE FOLLOWING ENTRIES ARE PROVIDED:

           CONTROL ENTRIES:

             INLSCNIN   INITIALIZE THE SCAN CONTROL BLOCK
             INLSCNBK   CHECK WHETHER THE REST OF THE
                        STRING IS BLANK
             INLSCNBR = INLSCNBK BUT POINT TO NON BLANK CHAR

           SCANNING ENTRIES PROPER:

               INLSCNAN  SCAN FOR ALPHANUMERIC STRING
               INLSCNCC  SCAN FOR A SPECIAL CHARACTER
               INLSCNNN  SCAN FOR DECIMAL NUMERICS
               INLSCNF   SCAN FOR DECIMAL NUMERICS AND CONVERT
                         INTO A FULLWORD
               INLSCNXN  SCAN FOR HEXADECIMAL NUMERICS
               INLSCNXF  SCAN FOR HEXADECIMAL NUMERICS AND
                         CONVERT INTO A FULLWORD
               INLSCNKW  SCAN FOR A KEYWORD (ALPHA)
               INLSCNLN  SCAN FOR A LIBRARY NAME
               INLSCNGN  SCAN FOR A GENERIC NAME (ALPHANUM.
                         AND LAST CHAR. = ' ' POSSIBLE)
               INLSCNGS  SCAN FOR A GENERIC SUBLIB (ALPHANUM.
                         ONLY OR ' ' ONLY)
               INLSCNST  SCAN FOR ANY STRING
               INLSCNQS  SCAN FOR ANY STRING ENCLOSED IN QUOTES

               FOR ALL THESE ENTRIES
                 FLAG 'FOUNDIND' IN INLCSCAN WILL INDICATE
                 WHETHER THE REQUESTED SCAN WAS SUCCESSFUL. IN
                 THIS CASE THE POINTER 'START' IN INLCSCAN
                 POINTS TO THE RECOGNIZED ENTITY, 'LEN' IN
                 INLCSCAN CONTAINS THE LENGTH OF THE RECOGNIZED
                 ITEM, AND 'ENDSTR' INDICATES WHETHER THE
                 STRING IS EXHAUSTED.
```

```
INPUT =     PARAMETERS AND FIELDS OF INLCSCAN

OUTPUT =    PARAMETERS AND FIELDS OF INLCSCAN
            RETURN CODE 0 = SUCCESSFULLY COMPLETED
            RETURN CODE 8 = ANY FAILURE  (SET BY INLPEXIT)

 ENTRY POINT:    SEE FUNCTION DESCRIPTION

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPSYPA

```
MODULE INLPSYPA
DESCRIPTIVE NAME: SYNTAX CHECK OF COMMAND PARAMETERS

FUNCTION =       THIS MODULE PARSES THE COMMAND PARAMETERS
                 WITH AID OF THE CORRESPONDING PARSER TABLE
                 MODULE. EVERY COMMAND NAME IN INLTPARS IS
                 CONNECTED WITH ONE PARSER TABLE MODULE
                 CONTAINING THE DESCRIPTION OF THE PARAMETERS
                 (E.G. INLTACC IS THE PARSER TABLE MODULE
                 BELONGING TO THE ACCESS COMMAND).

                 THE LOGIC FLOW IS SUBDIVIDED IN:
                   POSITIONAL PARAMETER CHECK AND
                   KEYWORD PARAMETER CHECK
                 DEPENDING ON THE CONTENTS OF THE PARSER TABLE
                 MODULE.

                 IN CASE OF SUCCESSFUL SYNTAX CHECK THE PARSED
                 COMMAND BLOCK INLCCMDP IS COMPLETED, THE
                 DELAYED CANCEL OPTION IS INITIALIZED, AND THE
                 CORRESPONDING COMMAND SEMANTIC CHECK/EXECUTION
                 PHASE IS LOADED AND EXECUTED.
                 AFTER THE COMPLETION OF THE COMMAND EXECUTION
                 OR AFTER AN UNSUCCESSFUL SYNTAX CHECK, THE
                 CONTROL GOES BACK TO INLPSYNA IN ORDER TO READ
                 THE NEXT COMMAND OR TO TERMINATE IN CASE OF
                 EOF.

INPUT =          PARAMETERS:
                   ADDRESS OF PARSER TABLE MODULE

OUTPUT =         INLCCMDP
                 RETURN CODE  0 = SUCCESSFULLY COMPLETED
                 RETURN CODE 16 = MESSAGE L111I
                 RETURN CODE  8 = ANY OTHER FAILURE (SET BY
                                                     INLPEXIT)

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPTD

```
MODULE INLPTD
DESCRIPTIVE NAME: TEST LIBRARY FOR CONSISTENCY

FUNCTION =        ACCORDING THE THE COMMAND,
                  - TEST LIB=...,AREA=SPACE: THE BITMAP OF THE
                         FREE SPACE OF THE LIBRARY IS DISPLAYED
                         (TOGETHER WITH
                         THE CONTROL BLOCKS INLCLDES, INLCEXTD,
                         INLCSPAD). ERRORS ARE REPORTED.
                  - TEST LIB=...,AREA=ALL:     THE WHOLE
                         LIBRARY IS CHECKED FOR CONSISTENCY.
                         ALL CHAINS ARE FOLLOWED. ALL CONTROL
                         BLOCK INFORMATION IS CHECKED AGAINST
                         THE ACTUAL STATE OF THE LIBRARY.
                         WASTED LIBRARY BLOCKS ARE DETECTED.
                         A DETAILED ERROR REPORT IS GIVEN.
                       - REPAIR=YES: IF THERE ARE ANY LIBRARY
                         BLOCKS IN THE BITMAP SHOWN AS
                         RESERVED BUT DOES NOT OCCUR IN ANY
                         LOGICAL LIBRARY BLOCK CHAIN THEY ARE
                         FREED. MISMATCH OF FREE SPACE WITH
                         VALUE IN SPACE DESCRIPTOR IS CORRECTED
                  - TEST SUBLIB=... : THE MEMBER INDEX OF THE
                         SUBLIBRARY IS DISPLAYED AND CHECKED
                         FOR CONSISTENCY. DETECTED ERRORS ARE
                         REPORTED.
                  - TEST MEMBER: THE DIRECTORY ENTRY OF THE
                         GIVEN MEMBER IS DISPLAYED AND ITS
                         CONTENTS IS CHECKED AGAINST THE
                         ACTUAL STATE OF THE LIBRARY.
                         THE CHAIN  OF THE LIBRARY BLOCKS
                         BELONGING TO THAT MEMBER IS SHOWN
                         FOLLOWING THE FORWARD CHAIN.
                         DETECTED ERRORS ARE REPORTED.

INPUT =           PARSED COMMAND LIST TABLE

OUTPUT =          MEMBER INDEX OF A SUBLIBRARY
                  LIBRARY FREE SPACE MAP
                  REPORT OF LIBRARY INCONSISTENCY
                  RETURN CODES:
                  0 ... SUCCESSFULLY COMPLETED
                  2 ... DEFECT DETECTED IN LIBRARY
                  8 ... FUNCTION FAILED

 ENTRY POINT:     SAME AS MODULE NAME
```

INLPTDLH

```
MODULE INLPTDLH
DESCRIPTIVE NAME: TEST LIBRARY HEADER FOR CONSISTENCY

FUNCTION =        ACCORDING TO  THE COMMAND,
                  - TEST LIB=...,AREA=SPACE: THE BITMAP OF THE
                         LB   SPACE OF THE LIBRARY IS DISPLAYED
                         (TOGETHER WITH
                         THE CONTROL BLOCKS INLCLDES, INLCEXTD,
                         INLCSPAD). ERRORS ARE REPORTED.
                  - TEST TRACE=...: THE LEVEL1/LEVEL2 LIBRARIAN
                         TRACE IS ACTIVATED OR DEACTIVATED.

INPUT =           PARSED COMMAND LIST TABLE

OUTPUT =          LIST OF LIBRARY, SPACE AND EXTENT DESCRIPTORS
                  SPACE MAP
                  REPORT OF LIBRARY INCONSISTENCY
                  RETURN CODES:
                =  0 ... FUNCTION SUCCESSFULLY COMPLETED
                  2 ... DEFECT DETECTED IN LIBRARY
                  8 ... FUNCTION FAILED

 ENTRY POINT:     SAME AS MODULE NAME
```

INLPTDX

```
MODULE INLPTDX
DESCRIPTIVE NAME: COMMON ROUTINES FOR TESTCOMMAND

FUNCTION =        LISTDENT: PRINT DENT INFORMATION

INPUT =           DEPENDENT FROM ENTRY

OUTPUT =          DEPENTEND FROM ENTRY

EXIT-NORMAL:      ACCORDING TO STANDARDS

ENTRY POINT:      LISTDENT
                  STATUSLB
```

INLPTIME

```
 MODULE NAME      = INLPTIME
 DESCRIPTIVE NAME = DATA TYPE TIME FRAME

 ENTRY-POINTS:     INLPPTIM    PARSES TIME FRAME
                  (INLPCMPT  IN THE FUTURE)
                             BISHER INTERN. PROC. IN INLPSARG
                       CHECKS IF A GIVEN VALUE LIES IN TIME FRAME

 INLPPTIM:

 INPUT: TS        POINTER TO START VALUE OF TIME WUNDOW
        TE        POINTER TO END VALUE OF TIMEWINDOW
        TW        POINTER TO TIMEWINDOW SPEC. IN COMMAND

 OUTPUT: PARSED AND CHECKED TIMEWINDOW AT TS AND TE

FUNCTION:

  PARSES TIMEVALUE IN PARSED COMMAND BLOCK. PARSED TIMEVALUE
  IS STORED AT TS AND TE.
  TS        WILL CONTAIN THE START OF THE TIMEWINDOW AND
  TE        THE END. EACH CHAR(10) FIELD OF THE TIMEWINDOW WILL
  CONTAIN A DATE IN THE FORM YYMMDDHHNN.
  YY  YEAR       MM MONTH      DD DAY
  HH  HOUR       NN MINUTE
  '00' <= YY <= '99' OR BLANK
  '01' <= MM <= '12' OR BLANK
  '01' <= DD <= '31' OR BLANK
  '01' <= HH <= '24' OR BLANK
  '00' <= NN <= '59' OR BLANK

 RETURN CODES

         0        FIELD PARSED
         8        SYNTAX ERROR IN STRING
        12        TE < TS                                  @KX40596
```

INLPUPD

```
MODULE INLPUPD
DESCRIPTIVE NAME: UPDATE COMMAND PROCESSOR

FUNCTION =       PROCESS UPDATE  STATEMENT AND DO SEMANTIC
                 CHECKS:
                 - LABEL INITUPD:
                    DO PREPARATORY WORK FOR UPDATE  PROCESSING
                    AND SEMANTIC CHECKING
                 - LABEL LIBUPD:
                    FIND THE LIBRARY/SUBLIBRARY IN WHICH TO
                    UPDATE THE MEMBER
                 - LABEL CHECKUPD:
                    CHECK MSHP SECURITY FLAG
                 - LABEL PREPUPD:
                    CONNECT THE NEW MEMBER
                 - LABEL SAVEUPD:
                    PROCESS 'SAVE' PARAMETER
                 - LABEL SEQUPD:
                    PROCESS 'SEQUENCE' PARAMETER
                 - LABEL COLUPD:
                    PROCESS 'COLUMN' PARAMETER
                 - LABEL SELUPD:
                    SELECT THE UPDATE FUNCTION DEPENDING ON
                    THE UPDATE SUBCOMMAND IN INPUT AREA
                 - LABEL ADDUPD:
                    PROCESS FUNCTION: ADD
                 - LABEL DELUPD:
                    PROCESS FUNCTION: DELETE
                 - LABEL REPUPD:
                    PROCESS FUNCTION: REPLACE
                 - LABEL FINUPD:
                    FINISH UPDATE  PROCESSING AND RETURN.

INPUT =          LIBRARIAN COMMUNICATION REGION INLCCOMR
                 PARSED COMMAND CONTROL BLOCK INLCCMDP

OUTPUT =         RETURN CODE 0 = SUCCESSFULLY COMPLETED
                 RETURN CODE 8 = ANY FAILURE

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPUSDL

```
MODULE INLPUSDL
DESCRIPTIVE NAME: UPDATE SYSTEM DIRECTORY LIST

FUNCTION =
                  UPDATE SYSTEM DIRECTORY LIST (SDL) BY
                  CALLING INLPSDL.

INPUT =           STOW TABLE
                  LIBRARY REQUEST PARAMETER LIST (LRPL)

OUTPUT =          UPDATED LIBRARY INDICES
                  MESSAGES, RETURN CODES
                         0   NORMAL RETURN
                         4   FUNCTION (PARTIALLY) FAILED,
                             SEE RETURN INFORMATION IN STOWTAB
                        16   EXTERNAL ERROR WITH FEEDBACK CODE
                        20   INTERNAL ERROR WITH FEEDBACK CODE
                        32   SECURITY VIOLATION

 ENTRY POINT:     SAME AS MODULE NAME
```

INLPWOR

```
MODULE INLPWOR
DESCRIPTIVE NAME: CONSOLE COMMUNICATION

FUNCTION =       ISSUE MESSAGE ON THE CONSOLE AND READ THE
                 ANSWER.
                 THE ANSWER RESIDES IN THE I/O-AREA SPECIFIED
                 AND IS MADE UPPER CASE.
                 IF THE DTF IS NOT AVAILABLE THE ANSWER IS READ
                 VIA THE EXCP MACRO.
                 IF THE CALL INTERFACE IS ACTIVE THEN TAKE THE
                 ANSWER FROM THE OUTPUT AREA OF THE SYSLOG EXIT.

INPUT =          PARAMETERS:
                  1) MESSAGE           MESSAGE TO BE DISPLAYED
                       MSGLEN  BIN(8)  LENGTH OF THE MESSAGE
                       MSG     PTR(31) ADDRESS OF MESSAGE TEXT
                  2) LAMBPTR   PTR(31) LAMB ADDRESS
                                          LAMB CONTAINS DTF INFO

                 OPERATOR ANSWER FROM SYSLOG

OUTPUT =         MESSAGE ON SYSLOG OR TO CALL INTERFACE I/O-AREA

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPWRTP

```
MODULE INLPWRTP
DESCRIPTIVE NAME: WRITE TAPE BLOCK

FUNCTION =     WRITE TAPE BLOCK

               THE PURPOSE OF THIS FUNCTION IS TO
               INITIALIZE THE WRITING OF A BUFFER OF
               THE BACKUP FILE TO TAPE.
               THE BLOCK HEADER IS INITIALIZED AND
               MODULE INLPTO IS CALLED WHICH CONTROLS
               THE TRANSFER OF THE BUFFER TO TAPE AND
               RETURNS ANOTHER EMPTY BUFFER FOR THE
               FOLLOWING PROCESSING BY BACKUP.

INPUT =        BDBPTR = PTR TO BDB OF BUFFER TO BE WRITTEN
               BUFPTR = ADDR LAST BYTE TO BE WRITTEN + 1,
               BLKNBR = BLOCK NUMBER OF NEXT TAPE BLOCK

OUTPUT =       BDBPTR = PTR TO NEXT BDB OF LOOP (SET BY INLPTO)
               BUFPTR = 1ST BYTE PAST BLK HDR
               BUFEND = PTR TO END OF BUFFER

 ENTRY POINT:    SAME AS MODULE NAME

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPWTO

```
MODULE INLPWTO
DESCRIPTIVE NAME: WRITE ON CONSOLE

FUNCTION =       ISSUE A LINE ON THE CONSOLE OR USE THE
                 LIBRARIAN CALL INTERFACE EXIT.
                 IF THE DTF IS NOT AVAILABLE THE LINE IS ISSUED
                 VIA EXCP.

INPUT =          PARAMETERS:
                  1) LINE              LINE TO BE DISPLAYED
                       PLEN    BIN(8)  LENGTH OF THE DATA
                       PDATA   PTR(31) ADDR. OF DATA TO DISPLAY
                  2) LAMBPTR   PTR(31) LAMB ADDRESS
                                          LAMB CONTAINS DTF INFO

OUTPUT =         LINE ON SYSLOG OR TO CALL INTERFACE I/O-AREA

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLPWTP

```
MODULE INLPWTP
DESCRIPTIVE NAME: PRINT A LINE

FUNCTION =       PRINT A LINE OR USE CALL-INTERFACE EXIT
                 PERFORM CARRIAGE CONTROL REQUESTED BY THE
                 CALLER AND TAKE CARE OF THE ACTUAL PAGE SIZE.

INPUT =          PARAMETERS:
                  1) LAMBPTR PTR(31)  ADDRESS OF LAMB
                                        LAMB CONTAINS DTF INFO
                  2) LINE    LINE TO BE PRINTED
                       CC    CHAR(1) CONTROL CHARACTER(STANDARD)
                       PLEN  BIN(8)  LENGTH OF THE PRINT AREA
                       PDATA PTR(31) ADDR. OF DATA TO PRINT
                  3) HEADER  HEADER LINE IF SKIPPING TO
                             CHANNEL 1 (OPTIONAL)
                       HLEN  BIN(8)  HEADER LENGTH
                       HDATA PTR(31) ADDR. OF HEADER TO PRINT

OUTPUT =         LINE ON SYSLST OR TO CALL-INTERFACE I/O-AREA
                 RETURN CODES:  0 = SUCCESSFUL
                                8 = INVALID PARAMETER SPECIFIC.

 ENTRY POINT:    SAME AS MODULE NAME

   PURPOSE:      SEE FUNCTION DESCRIPTION

   LINKAGE:      PLS/III CALL CONVENTIONS
```

INLXREST

```
MODULE INLXREST
DESCRIPTIVE NAME: STAND ALONE RESTORE

FUNCTION =       THIS PROGRAM (STAND-ALONE VERSION OF
                 THE RESTORE FUNCTION) RESTORES SYSRES FILES.
                 (RESTORING INTO VSAM MANAGED SPACE CAN
                  ONLY BE DONE ONLINE)
                 RESTORING OF A LIBRARY CONTAINED ON THE
                 BACKUP TAPE IS CONTROLLED BY PROMPTS
                 AND ANSWER VIA SYSLOG.

                 - INITIALIZATION OF STAND-ALONE RESTORE
                 - ALLOCATE SPACE FOR TAPE AND DISK BUFFERS
                 - OPEN TAPE (INLPTIO)
                 - POSITION TAPE TO BACKUP FILE HEADER
                 - PROMPT THE USER WHETHER A SYSRES FILE FOUND
                   ON TAPE SHOULD BE RESTORED AND IF YES, FOR
                   THE FOLLOWING PARAMETERS:
                   - FILE ID
                   - ALLOCATION FOR THE SYSRES FILE
                   - RESTORE OF SUBLIBRARIES SELECTIVELY OR NOT
                 - WRITE THE F1 LABEL INTO VTOC (ALOCFILE)
                 - SIMULATE ONLINE ENVIRONMENT FOR LIBRARIAN
                   SUPPORT: (SIMONLIN)
                   - SET UP LIBRARY POINTER TABLE HEADER
                   - SET UP LIBRARY DEFINITION TABLE ENTRY
                   - SET UP DEVICE DEFINITION TABLE ENTRY
                   - SET UP EXTENT DEFINITION TABLE ENTRY
                   - SET UP LIBINFO POINTERS
                 - CONNECT TO THE NEW LIBRARY
                   (IF CKD DEVICE THE LIBRARY IS FORMATTED)
                 - DISCONNECT FROM THE LIBRARY
                 - DO FOR ALL SUBLIBRARIES FOUND ON TAPE FOR
                   THE LIBRARY:
                     IF THE SUBLIBRARY IS NOT SELECTIVELY
                     EXCLUDED FROM RESTORE:
                     (SUBLIBRARY 'SYSLIB' CANNOT BE EXCLUDED)
                     - SET UP SUBLIBRARY DEFINITION TABLE ENTRY
                     - RESTORE SUBLIBRARY (INLPRES2)
                 - DISCONNECT FROM THE LIBRARY
                 - ISSUE RESTORE COMPLETE MESSAGE
                 - CLOSE INPUT TAPE (INLPTIC)
                 - CLOSE SYSLST AND SYSLOG DTF'S
                 - RETURN TO STAND-ALONE JCL VIA EOJ

INPUT =          A TAPE BACKUP FILE CONTAINING:
                   LIBRARIES (INCLUDING AT LEAST ONE SYSRES FILE)

 ENTRY POINT:    SAME AS MODULE NAME

   LINKAGE:      NONE
```

Macros

DTFSL

```
  DESCRIPTIVE NAME: READ ACCESS TO SOURCE BOOKS
  FUNCTION: ALLOW RETRIEVAL OF SOURCE BOOKS OF A SUB-
          LIBRARY BY A MACRO INTERFACE AS FOLLOWS:

1. FIND : FIND A BOOK AND SAVE ITS POSITION IF FOUND.
(FNDSL)   A BRANCH ADDRESS MUST BE SPECIFIED WHERE TO BRANCH
          TO IF THE BOOK IS NOT FOUND.
          REGISTER 1 POINTS TO A 9-BYTE BOOK NAME.
          THE ADDRESS OF THE FOUND BOOK IS STORED IN THE
          DTFSL CONTROL BLOCKS FOR FOLLOWING GET'S AND READ'S.

2. GET  : RETRIEVE A BOOK SEQUENTIALLY (BY ONE CARD WITH
(GETSL)   EACH GET REQUEST).

3. NOTE : IF INNER MACROS ENCOUNTERED NOTE POSITION
(NTSL)    OF INNER MACRO.

4. POINT: AFTER PROCESSING OF INNER MACRO RESTORE POSITION
(PTSL)

  RETURN CODES FROM SERVICE FUNCTIONS:

ERROR RETURN AND ERROR CODES ( NOT IN BACK-LEVEL MODE ) :

REGISTER 15  CONTAINS A NUMBER TO INDICATE THE REASON FOR FAILURE
2. X'101' NESTING DEPTH OF NOTE EXCEEDED  (USER ERROR)
3. X'102' NOTE, POINT OR READ GIVEN, BUT NO EXPANSION FOR  (USER)
4. X'103' TOO MANY POINT GIVEN  (USER ERROR)
5. X'104' INVALID REQUEST OR LIBRARY OPEN FAILURE  (USER/SYSTEM)
6. X'105' GET, NOTE, READ, OR POINT GIVEN WITHOUT FIND  (USER )
7. X'106' CONCURRENT DELETION OF MEMBER BEING READ (USER)
8. X'108' REENTR=YES, WORKAREA VALIDATION FAILED
```

INLMBLDL

```
MACRO NAME:  INLMBLDL

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, BILINGUAL

DESCRIPTIVE NAME:  MACRO FOR BUILDING LIST OF ENTRIES
FUNCTION:

THIS MACRO BUILDS A LIST OF INFORMATION IN AN AREA CALLED A STOW
table.   THE FUNCTION GETS A LIST OF ARGUMENTS (IN THE STOW TABLE
FORMAT) AS INPUT AND PASSES BACK AN IDENTIFICATION
ABOUT THE EXISTENCE OF THE GIVEN LIBRARY OBJECTS IN THE
SEARCHING CHAIN, TOGETHER WITH ADDITIONAL DESCRIPTOR
INFORMATION.
THE INFORMATION CAN BE ABOUT SUBLIBRARIES OR MEMBERS AND
THE QUANTITY OF INFORMATION RETURNED IS CONTROLLED BY THE CALLER.
THE CALLER CAN REQUEST TO BUILD A LIST ABOUT A SPECIFIC LIST
OF SUBLIBRARIES OR MEMBERS,   OR THE CALLER CAN REQUEST THAT A
GENERIC SEARCH BE DONE.  THE MACRO WILL START ITS SEARCH AT THE
BEGINNING OF THE SPECIFIED LIBRARY CHAIN AND CONTINUE THROUGH
THE LIBRARIES UNTIL IT CAN FIND THE ARGUMENTS OR THE CHAIN ENDS.
ON A GENERIC REQUEST, THE SEARCH STOPS AT THE FIRST OCCURRENCE
OF A MATCHING ARGUMENT.
THE LRPL MAY CONTAIN ADDITIONAL SEARCH PARAMETER LIKE LOCKID
OR TIMEFRAME. AN ENTRY IS FOUND IF ALL SEARCH PARAMETER MATCH.
         - THE FUNCTION WORKS FOR:
         1. A LIST OF SUBLIBRARY NAMES OR A LIST OF MEMBER
         SPECIFICATIONS, WHICH WILL BE FLAGGED AS EXISTENT OR
         NOT (INFORM=KEY) OR FILLED UP WITH ADDITIONAL SUBLIB
         OR MEMBER DESCRIPTOR INFORMATION (INFORM=ENTRY),
         2. A LIST OF FULL SPECIFIED NAMES OR ONE GENERIC
         SPECIFICATION, WHICH WILL BE EXPANDED TO A LIST OF
         ENTRIES FOUND IN THE FIRST LIBRARY OR SUBLIBRARY
         WHERE THE GENERIC SPECIFICATION MATCHES.
         - FOR SUBLIBRARIES, THE LOCKING RESOURCE NAMES WILL
         BE BUILT WITH REQUEST INFORM=RESOURCE.
         - IF NO MATCH IS FOUND IN THE SEARCHING CHAIN, RETURN
         CODE 12 IS PASSED BACK.
         - IF THE INFORMATION AREA (PARAMETER STOLEN) IS
         EXCEEDED BY THE ACTUAL INFORMATION, RETURN CODE 8 IS
         PASSED BACK. NOW IT IS POSSIBLE TO GIVE A CONTINUATION
         REQUEST (PARAMETER CONTINUE) TO GET THE REMAINING
         INFORMATION.
         - IF SOME OF THE ARGUMENTS IN A LIST DO   NOT OCCUR
         IN THE SEARCHING CHAIN OR DO NOT PASS THE SECURITY
         CHECK THE CORRESPONDING ENTRIES ARE FLAGGED AND A
         RETURN CODE OF 4 IS PASSED BACK TO THE INVOKER.
         - THE SECURITY CHECK FOR A FULLY SPECIFIED ARGUMENT
         IS FOR READ AUTHORIZATION.
         - THE SECURITY CHECK FOR A GENERIC SUBLIBRARY
         SPECIFICATION IS DONE FOR READ AUTHORIZATION ON
         LIBRARY LEVEL, FOR A GENERIC MEMBER SPECIFICATION FOR
         READ AUTHORIZATION ON THE SUBLIBRARY.
```

```
RETURN CODES FROM SERVICE FUNCTION:
   0      REQUEST HAS BEEN HONORED
   4      SOME OF THE ARGUMENTS WERE NOT FOUND IN THE GIVEN CHAIN
          AND THEIR STOW ENTRIES ARE FLAGGED
   8      THE AREA SUPPLIED FOR THE STOW TABLE WAS NOT SUFFICIENT
          AND THE REQUEST COULD NOT BE COMPLETED.  A NONGENERIC
          REQUEST CAN BE CONTINUED BY SUPPLYING THE ENTRIES THAT
          WERE NOT PROCESSED.  A GENERIC REQUEST CAN BE CONTINUED
          BY SUPPLYING THE ORIGINAL ARGUMENT AS THE FIRST ENTRY
          OF THE STOW TABLE AND THE LAST RETURNED ARGUMENT AS THE
          SECOND ENTRY ALONG WITH THE KEYWORD 'CONTINUED'
  12      NO MATCH FOUND
  16      EXTERNAL ERROR WITH FEEDBACK CODE
  20      INTERNAL ERROR WITH FEEDBACK CODE
  32      SECURITY VIOLATION

FLAGS SET IN THE ARGUMENT ENTRIES (INLCLARG):

  LARGFND = TRUE ... ENTRY FOUND IN THE LIBRARY
  LARGIPT = TRUE ... MEMBER EXPECTS DATA ON SYSIPT FOR PROCESSING
                     (PROCEDURE WITH DATA=YES)
  LARGSYS = TRUE ... MEMBER FOUND IN SYSTEM SUBLIBRARY
  LARGSEC = TRUE ... SECURITY VIOLATION FOR THIS ENTRY
  LARGDLCK= TRUE ... MEMBER IS LOCKED
```

INLMDSTO

```
MACRO NAME:  INLMDSTO

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, ASM

DESCRIPTIVE NAME:  GET DYNAMIC STORAGE.
FUNCTION:

       1) FOR SPID=LAMB:
          IF LAMBGVS=0 THEN ISSUE GETVIS FOR PARTITION OR
          SYSTEM GETVIS AREA DEPENDENT ON FLAG LAMBSVIS.
          IF THE SUBPOOL ID IN THE LAMB IS INCOMPLETE IT IS
          COMPLETED BY INSERTING THE TASK-ID.
          INSERT BEGIN/END ADDRESS IN LAMBGVS/LAMBGVE/LAMBGVU.
          LOAD ADDRESS OF NEXT-AVAILABLE SPACE INTO R1 AND
          UPDATE HIGH-USED POINTER LAMBGVU BY LENGTH VALUE
          OF R0.
          IF LAMBGVS>0 THEN LOAD ADDRESS OF NEXT-AVAILABLE
          SPACE IN R1 AND UPDATE LAMBGVU BY LENGTH OF R0.
       2) FOR SPID=LRPL:
          ISSUE GETVIS FOR PARTITION OR SYSTEM DEPENDENT
          ON LAMB FLAG LAMBSVIS.
          IF THE SUBPOOL ID IN THE LRPL IS INCOMPLETE IT IS
          COMPLETED BY INSERTING THE TASK-ID.
```

INLMFIND

```
MACRO NAME:  INLMFIND

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, BILINGUAL

DESCRIPTIVE NAME:  MACRO FOR FINDING MEMBER IN CHAIN
FUNCTION:

THIS MACRO IS USED TO FIND A MEMBER WITHIN A LIBRARY/SUBLIBRARY
CHAIN WHICH HAS BEEN DEFINED BY USING THE LBRACCES MACRO.
THE MEMBER AND TYPE GIVEN IN ARGUMENT IS SEARCHED IN THE CHAIN OF
SUBLIBRARIES DENOTED BY 'LIBUSE' AND 'LIBTYPE'.
IF THE REQUEST CAN BE HONORED
THE ARGUMENT IS COMPLETED WITH THE REQUESTED INFORMATION FROM
THE LIBRARY AND SUBLIBRARY.
ADDITIONALLY, THE MEMBER IS CONNECTED FOR A
FOLLOWING GET/PUT MEMBER RECORD REQUEST (CONNECT=OLD|UPDATE).
IF THE REQUESTED MEMBER IS GIVEN AS A GENERIC NAME, THE FIRST
MEMBER FOUND WITH THE SAME PREFIX IN ITS NAME IS CONNECTED.
IF THE REQUESTED MEMBER IS NOT FOUND, THE SUBLIBRARY WHICH WAS
SEARCHED LAST FOR REMAINS CONNECTED.
THE LRPL MAY CONTAIN ADDITIONAL SEARCH PARAMETER LIKE LOCKID
OR TIMEFRAME. A MEMBER IS FOUND IF ALL SEARCH PARAMETER MATCH.
IF THE LIBRARY WHERE THE MEMBER RESIDES IN IS PROTECTED VIA
VSE/ACF, THE REQUESTOR NEEDS READ (FOR CONNECT=OLD) OR UPDATE
(FOR CONNECT=UPDATE) AUTHORIZATION FOR THE MEMBER.

RETURN CODES FROM SERVICE FUNCTION:
   0       REQUEST HAS BEEN HONORED
   8       NO MATCH FOUND OR MEMBER LOCKED (CONNECT=UPDATE)
           LARGFND = FALSE, IF NO MATCH FOUND
           LARGFND = TRUE AND LARGDLCK = TRUE IF MEMBER LOCKED
           MEMBER IS NOT CONNECTED
  12       SEARCHING CHAIN IS EMPTY
  16       EXTERNAL ERROR WITH FEEDBACK CODE
  20       INTERNAL ERROR WITH FEEDBACK CODE
  32       SECURITY VIOLATION
```

INLMFREE

```
MACRO NAME:  INLMFREE

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, ASM

DESCRIPTIVE NAME:  FREE DYNAMIC STORAGE
FUNCTION:
       1) LAMB SPECIFIED, FREEVIS MISSING:
          DECREASE LAMB POINTER LAMBGVU BY VALUE IN R0.
          IF LAMBGVU=LAMBGVS AND FLAG LAMBGVIN=OFF,
          ISSUE FREEVIS WITH VALUES TAKEN FROM LAMB FIELDS
          LAMBGVS AND LAMBGVE AND CLEAR FIELDS LAMBGVS,
          LAMBGVE, LAMBGVU.
          IF LAMGVU=LAMBGVS AND FLAG LAMBGVIN=ON, KEEP
          GETVIS AREA DENOTED BY FIELDS LAMBGVS AND LAMBGVE.
       2) LAMB SPECIFIED, FREEVIS=FORCE:
          ISSUE FREEVIS WITH VALUES TAKEN
          FROM LAMB FIELDS LAMBGVS AND LAMBGVE.
          CLEAR FIELDS LAMBGVS, LAMBGVE, LAMBGVU.
       3) LAMB MISSING, ADDRESS AND LENGTH SPECIFIED:
          ISSUE FREEVIS FOR PARTITION OR SYSTEM DEPENDENT
          ON START ADDRESS OF AREA TO BE FREED.
```

INLMGDIR

```
MACRO NAME:  INLMGDIR

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, BILINGUAL

DESCRIPTIVE NAME:  MACRO FOR READING DIRECTORY SEQUENTIALLY
FUNCTION:

THE NEXT DIRECTORY ENTRY OF THE CONNECTED INDEX IS RETURNED
IN THE 'ARG' PARAMETER.
FOR LEVEL=SUBLIB, THE SEQUENTIALLY NEXT DESCRIPTOR OF THE
SUBLIBRARY INDEX,
FOR LEVEL=MEMBER, THE SEQUENTIALLY NEXT DESCRIPTOR OF THE
MEMBER INDEX, AND
FOR LEVEL=LIB, THE LIBRARY DESCRIPTOR
IS PASSED BACK.
IF THE LIBRARY IS PROTECTED VIA VSE/ACF, SECURITY CHECK IS DONE
FOR READ AUTHORIZATION ON LIBRARY LEVEL (FOR LEVEL=LIB|SUBLIB),
FOR READ AUTHORIZATION ON SUBLIBRARY LEVEL (FOR LEVEL=MEMBER).
RETURN CODES FROM SERVICE FUNCTION:
   0       REQUEST HAS BEEN HONORED
   4       SUCCESSFULLY, LAST ENTRY OF DIRECTORY
  12       REQUEST OUTSIDE OF DIRECTORY
  16       EXTERNAL ERROR WITH FEEDBACK CODE
  20       INTERNAL ERROR WITH FEEDBACK CODE
  32       SECURITY VIOLATION
```

INLMGETR

```
MACRO NAME:  INLMGETR

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, BILINGUAL

DESCRIPTIVE NAME:  MACRO FOR READING MEMBER RECORD
FUNCTION:

A MEMBER RECORD IS RETURNED IN THE WORK AREA OR THE ADDRESS OF
THE REQUESTED RECORD IS RETURNED IN THE LRPL.

- FOR MODE=MOVE & REQUEST=NONCONTINUOUS:
AS MANY RECORDS AS ARE SPECIFIED BY PARAMETER 'LRECNO' ARE MOVED
TO THE WORK AREA. IF LRECNO=0, THE WORK AREA IS FILLED WITH
AS MANY RECORDS AS POSSIBLE. A MEMBER RECORD IS PRECEDED
BY A TWO-BYTES LENGTH FIELD.
IF LRECNO=1, ONLY ONE RECORD WITHOUT LENGTH FIELD IS TRANSFERRED
TO THE WORK AREA OF THE REQUESTOR.

- FOR MODE=MOVE & REQUEST=CONTINUOUS:
THE WORK AREA OF THE REQUESTOR IS FILLED UP WITH THE MEMBER
UNTIL EITHER THE AREA IS FULL OR THE MEMBER IS COMPLETELY
READ. THE MOVED LENGTH IS REPORTED IN PARAMETER 'MOVELEN'.
RETURN CODES FROM SERVICE FUNCTION:
   0       REQUEST HAS BEEN HONORED SUCCESSFULLY
   4       REQUEST HAS BEEN HONORED SUCCESSFULLY, END OF MEMBER
   8       LOGICAL RECORD TRUNCATED
  12       REQUEST OUTSIDE OF MEMBER
  16       EXTERNAL ERROR WITH FEEDBACK CODE
  20       INTERNAL ERROR WITH FEEDBACK CODE
  36       MEMBER CHAINING ERROR
```

INLMIGR

```
MACRO NAME:  INLMIGR

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO

DESCRIPTIVE NAME:  MACRO FOR BUILDING THE LIBRARIAN MIGRATION
FUNCTION:

THE NAMES OF THE INPUT PARAMETERS ARE CHANGED TO ASSEMBLER
DEFINITION STATEMENTS
A PUNCH PHASE STATEMENT AND THE CSECT STATEMENT ARE INSERTED

RETURN CODES FROM LANGUAGE PROCESSOR:
   0       MACRO EXPANSION SUCCESSFUL
   8       INVALID PARAMETERS SPECIFIED
```

INLMLAMB

```
MACRO NAME:  INLMLAMB

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, BILINGUAL

DESCRIPTIVE NAME:  MACRO FOR LIBRARIAN ACCESS METHOD BLOCK
FUNCTION:

THE CONTROL BLOCK INLCLAMB
IS DEFINED AND FILLED UP WITH THE SPECIFIED
PARAMETERS.
```

INLMLCON

```
MACRO NAME:  INLMLCON

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, BILINGUAL

DESCRIPTIVE NAME:  MACRO FOR CONNECTING TO LIBRARY
FUNCTION:

THIS MACRO IS USED TO CONNECT TO A LIBRARY WITHIN A CHAIN WHICH
HAS BEEN DEFINED AND ACCESSED USING THE LBRACCES MACRO.

- FOR CONNECT=OLD:
THE LIBRARY DEFINED IN LIBINFO IS CONNECTED AND
THE ACCESS CONTROL BLOCK FOR IT IS INITIALIZED.
THE LIBRARY MUST HAVE BEEN ADDED TO THE LCTS WITH DEFINE=OLD
PREVIOUSLY.
- FOR CONNECT=NEW:
THE NEW LIBRARY IS PREFORMATTED AT THE SPACE GIVEN BY LIBINFO,
THE LIBRARY DESCRIPTOR IS WRITTEN, AND
THE LIBRARY IS CONNECTED FOR ACCESS.
THE LIBRARY MUST HAVE BEEN ADDED TO THE LCTS WITH DEFINE=NEW
PREVIOUSLY.

ANY CONNECTION TO A SUBLIBRARY OR A DIFFERENT LIBRARY WILL BE
DISCONNECTED IMPLICITLY BEFORE THE REQUESTED LIBRARY IS
CONNECTED.
IF A MEMBER WAS PREVIOUSLY CONNECTED THE FUNCTION IS NOT
PERFORMED.
RETURN CODES FROM SERVICE FUNCTION:
   0       REQUEST HAS BEEN HONORED
   4       SUCCESSFUL, ALREADY CONNECTED
  16       EXTERNAL ERROR WITH FEEDBACK CODE
  20       INTERNAL ERROR WITH FEEDBACK CODE
```

INLMLDIS

```
MACRO NAME:  INLMLDIS

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, BILINGUAL

DESCRIPTIVE NAME:  MACRO FOR DISCONNECTING FROM LIBRARY
FUNCTION:

THE LIBRARY IS DISCONNECTED FROM ACCESS AND STORAGE USED FOR
INTERNAL CONTROL BLOCKS IS FREED.
ANY CONNECTION TO A LIBRARY OBJECT IS RELEASED. IF PRIVATE
BUFFERS HAVE BEEN UPDATED (WITH MEMBER CONNECTION) A WRITE
OPERATION IS FORCED.
RETURN CODES FROM SERVICE FUNCTION:
   0       REQUEST HAS BEEN HONORED
   8       LIBRARY WAS NOT CONNECTED
  12       LIBRARY IS FULL
  16       EXTERNAL ERROR WITH FEEDBACK CODE
  20       INTERNAL ERROR WITH FEEDBACK CODE
```

INLMLMIT

```
MACRO NAME:  INLMLMIT

MACRO TYPE:  INTERNAL LIBRARIAN MACRO, PLS

DESCRIPTIVE NAME:  MACRO FOR SEARCHING THE MIGRATION TABLE
FUNCTION:

FOR A NAME OF AN OLD LIBRARY THE CORRESPONDING LIBRARY/SUBLIBRARY
NAMES ARE SEARCHED IN THE LIBRARIAN MIGRATION TABLE INLPLMT

REGISTER USAGE FOR INLMLMIT MACRO:

  REG  0   USED BY MACRO SLOAD
  REG  1   USED AS DIRECTORY ENTRY POINTER
  REG 15   USED FOR RETURN CODES

  ALL OTHER REGISTERS REMAIN UNCHANGED

RETURN CODES FROM LANGUAGE PROCESSOR:
   0       MACRO EXPANSION SUCCESSFUL
   8       INVALID PARAMETERS SPECIFIED

RETURN CODES FROM SERVICE FUNCTION:
   0       REQUEST HAS BEEN HONORED
   4       MIGRATION TABLE(INLPLMT) DOESN'T EXIST
   8       OLD LIBRARY NAME NOT IN MIGRATION TABLE
```

INLMLRPL

```
MACRO NAME:  INLMLRPL

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, BILINGUAL

DESCRIPTIVE NAME:  MACRO FOR LIBRARIAN REQUEST PARAMETER LIST
FUNCTION:

THE CONTROL BLOCK INLCLRPL IS DEFINED AND FILLED UP
WITH THE SPECIFIED PARAMETERS.
```

INLMLSIM

```
MACRO NAME:  INLMLSIM

MACRO TYPE:  INTERNAL LIBRARIAN MACRO, PLS

DESCRIPTIVE NAME:  MACRO FOR INVOKING INLPLSIM
FUNCTION:

THE LIBRARY-SECURITY INTERFACE MODULE INLPLSIM IS INVOKED  FOR
SECURITY CHECK.
RETURN CODES FROM SERVICE FUNCTION:
   0       REQUEST HAS BEEN HONORED
  16       EXTERNAL ERROR WITH FEEDBACK CODE
  20       INTERNAL ERROR WITH FEEDBACK CODE
  32       SECURITY VIOLATION
```

INLMMCON

```
MACRO NAME:  INLMMCON

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, BILINGUAL

DESCRIPTIVE NAME:  MACRO FOR CONNECTING TO MEMBER
FUNCTION:

THIS MACRO ALLOWS THE USER TO CONNECT TO AN EXISTING MEMBER
OR BEGIN THE PROCESS OF CREATING A NEW MEMBER WITHIN A
SUBLIBRARY WHICH HAS BEEN DEFINED AND ACCESSED USING THE
LBRACCES MACRO.
THE LRPL MAY CONTAIN ADDITIONAL SEARCH PARAMETER LIKE LOCKID
OR TIMEFRAME. A MEMBER IS FOUND IF ALL SEARCH PARAMETER MATCH.

- FOR CONNECT=OLD | UPDATE:
A SEARCH IS MADE IN THE SUBLIBRARY DENOTED BY LIBINFO FOR
THE MEMBER GIVEN IN 'ARG'.  IF THE MEMBER IS FOUND, THE ACCESS
CONTROL BLOCK FOR IT IS ALLOCATED.
THE FUNCTION ALSO APPLIES TO A GENERIC MEMBER SPECIFICATION.
IN THIS CASE THE FIRST MEMBER FULFILLING THE NAME AND TYPE
SPECIFICATION WILL BE CONNECTED.
- FOR CONNECT=UPDATE THE CONNECTION OF A LOCKED MEMBER WILL
  BE REJECTED BY RETURN CODE 12.
- FOR CONNECT=NEW:
THE ACCESS CONTROL BLOCK FOR A NEW MEMBER IS INITIALIZED
INDEPENDENT OF WHETHER THE MEMBER ID ALREADY EXISTS IN THE
SUBLIBRARY.

THE ADDRESSED SUBLIBRARY MUST HAVE BEEN ENTERED IN THE LCTS.

SECURITY CHECK IS DONE AS FOLLOWS:
IF THE LIBRARY IS PROTECTED BY VSE/ACF,
- CONNECT=OLD REQUIRES READ ACCESS RIGHT TO THE MEMBER,
- CONNECT=NEW | UPDATE REQUIRES UPDATE ACCESS RIGHT TO THE
  MEMBER.

IF THE PROPER LIBRARY OR SUBLIBRARY IS ALREADY CONNECTED, IT
WILL NOT BE CONNECTED NEW. OTHERWISE, THE CORRESPONDING LIBRARY
AND / OR SUBLIBRARY IS DISCONNECTED AND THE CURRENT LIBRARY
AND / OR SUBLIBRARY IS CONNECTED. AN ALREADY CONNECTED MEMBER
WILL ALWAYS BE DISCONNECTED AND A NEW CONNECTION IS DONE
(FORCING INITIALIZATION OF THE MEMBER ACCESS CONTROL BLOCK
AND THE PRIVATE BUFFERS), THEREFORE THE FOLLOWING MEMBER
OPERATIONS STARTS FROM THE BEGINNING OF THE MEMBER.

A MEMBER BEING CONNECTED WITH CONNECT=OLD MAY ONLY BE READ
(NO WRITE ACCESS ALLOWED).
```

```
NOTE THE SPECIAL IMPLEMENTATION CONSIDERATIONS:

- IF FOR CONNECT = OLD | UPDATE THE MEMBER DESCRIPTOR
(PARAMETER DENT, E.G. AS IT IS RETURNED BY A FORMER INLMBLDL
INVOCATION) IS PASSED WITH THE MACRO INLMMCON, NO SEARCH FOR
THE MEMBER IN THE SUBLIBRARY IS DONE, BUT THE MEMBER ACCESS
CONTROL BLOCK IS INITIALIZED WITH INFORMATION PASSED WITH THE
GIVEN DIRECTORY ENTRY (PERFORMANCE REASON).

- IF THE INLMMCON SERVICE IS CALLED DURING THE NESTED PROCESSING
WITHIN A CHAIN, AFTER A POINT OPERATION (BY MACRO INLMPOIN)
INFORMATION IN THE ACCESS CONTROL BLOCKS HAS BEEN LOST
(DUE TO THE RESTRICTED LENGTH OF THE NOTE INFORMATION WORD).
THEREFORE, WHENEVER A CONNECT-TO-MEMBER IS GIVEN AFTER A POINT
OPERATION AND THE CORRECT SUBLIBRARY AND/OR LIBRARY IS ALREADY
CONNECTED, THIS INFORMATION IS NOT TRUSTED BUT THE ACCESS
CONTROL BLOCKS ARE INITIALIZED NEW.

RETURN CODES FROM SERVICE FUNCTION:
   0       REQUEST HAS BEEN HONORED
  12       MEMBER IS NOT IN SUBLIBRARY (FOR CONNECT=OLD|UPDATE)
           MEMBER IS LOCKED            (FOR CONNECT=UPDATE)
                IN THIS CASE IS LARGDLCK = TRUE.
  16       EXTERNAL ERROR WITH FEEDBACK CODE
  20       INTERNAL ERROR WITH FEEDBACK CODE
  32       SECURITY VIOLATION
```

INLMMDIS

```
MACRO NAME:  INLMMDIS

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, BILINGUAL

DESCRIPTIVE NAME:  MACRO FOR DISCONNECTING FROM MEMBER
FUNCTION:

THE MEMBER IS DISCONNECTED FROM ACCESS. REMAINING
LIBRARY BLOCKS ARE WRITTEN OUT ONTO DISK, IF MEMBER WAS CONNECTED
WITH CONNECT=NEW|UPDATE. FOR CONNECT=NEW, THE DIRECTORY ENTRY
IS BUILT FOR A SUBSEQUENT STOW REQUEST. INTERNAL ACCESS CONTROL
BLOCKS ARE DEALLOCATED.
THE SUBLIBRARY WILL BE CONNECTED AFTERWARDS.
RETURN CODES FROM SERVICE FUNCTION:
   0       REQUEST HAS BEEN HONORED
   8       MEMBER WAS NOT CONNECTED
  12       LIBRARY IS FULL
  16       EXTERNAL ERROR WITH FEEDBACK CODE
  20       INTERNAL ERROR WITH FEEDBACK CODE
```

INLMNOTE

```
MACRO NAME:  INLMNOTE

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, BILINGUAL

DESCRIPTIVE NAME:  MACRO FOR POSITIONING IN MEMBER
FUNCTION:

THE POSITION OF THE CONNECTED MEMBER IS NOTED IN THE GIVEN
ARGUMENT.
RETURN CODES FROM SERVICE FUNCTION:
   0       REQUEST HAS BEEN HONORED
  16       EXTERNAL ERROR WITH FEEDBACK CODE
  20       INTERNAL ERROR WITH FEEDBACK CODE
```

INLMPIDT

```
MODULE INLMPIDT MACRO FOR UPDATING OPEN IDENTIFICATION TABLE

  MODULE NAME: INLMPIDT     - LIBRARIAN MACRO

 DESCRIPTIVE NAME: UPDATE/RETRIEVE OPEN IDENTIFICATION
 FUNCTION:
               FOR THE DEFINITION AND/OR MAPPING OF THE REQUEST
               BLOCK THE MACRO INLCPIDT IS INVOKED.

             - FOR MF = INSERT | DELETE | LOOKUP
               MODIFICATION OF REQUEST BLOCK FOR UPDATING/LOOKUP
               OF OPEN IDENTIFICATION TABLE (IDT).
               INTERFACE TO THE SVA-PHASE $IJBLBR, MODULE INLPPIDT
               FOR MAINTAINING OPEN IDENTIFICATION TABLE.
 RETURN CODES FROM SERVICE FUNCTION:
                0 ... REQUEST HAS BEEN HONORED
                4 ... ENTRY NOT FOUND
               12 ... IDT IS FULL, ENTRY HAS NOT BEEN INSERTED
               16 ... EXTERNAL ERROR WITH FEEDBACK CODE AND MSG
               20 ... INTERNAL ERROR WITH FEEDBACK CODE AND MSG

             IF THE RETURN CODE IS GREATER THAN OR EQUAL TO 16,
             A MESSAGE HAS BEEN STORED IN THE
             CORRESPONDING I/O AREA SPECIFIED IN THE LAMB.
```

INLMPOIN

```
MACRO NAME:  INLMPOIN

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, BILINGUAL

DESCRIPTIVE NAME:  MACRO FOR POSITIONING IN MEMBER
FUNCTION:

THE MEMBER GIVEN IN 'ARG' IS POSITIONED AT AN ADDRESS
SPECIFIED IN 'ARG'.  'ARG' MUST HAVE BEEN PREVIOUSLY RETURNED
FROM INLMNOTE.  WHEN THE MEMBER BEING POINTED TO IS NOT THE
SAME AS THE CURRENTLY CONNECTED MEMBER, THE MACRO HAS THE EFFECT
OF DISCONNECTING FROM THE CURRENT MEMBER AND CONNECTING TO THE
MEMBER SPECIFIED IN 'ARG'.

RETURN CODES FROM SERVICE FUNCTION:
   0       REQUEST HAS BEEN HONORED
   4       SUCCESSFULLY, END OF MEMBER
  12       POSITION OUTSIDE OF MEMBER
  16       EXTERNAL ERROR WITH FEEDBACK CODE
  20       INTERNAL ERROR WITH FEEDBACK CODE
```

INLMPROC

```
MACRO NAME:  INLMPROC

MACRO TYPE:  INTERNAL LIBRARIAN MACRO, PLS ONLY

DESCRIPTIVE NAME: GENERATE PROCEDURE HEADER
FUNCTION:    GENERATE A STANDARD PROCEDURE PROLOGUE
             FOR LIBRARIAN MODULES.

PARAMETERS:  MAIN(YES|NO)  DOS MAIN PROCEDURE
             BASEREGS(NUMBER) NO. OF BASE REGISTERS NEEDED
             AUTOREGS(NUMBER) NO. OF DATA REGISTERS NEEDED
             OPTIONS(......) PLS OPTIONS
             MOD(CC) TWO CHARACTERS, DEFAULT '01'
             VER(CC) TWO CHARACTERS, DEFAULT DRIVER LEVEL
             PROL(-|LRPLPTR) POSITION OF LRPL PTR IN PARM.LIST
             ID(-|YES|CCCC..) CHARACTER STRING ID FOR MODULE
```

INLMPUTR

```
MACRO NAME:  INLMPUTR

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, BILINGUAL

DESCRIPTIVE NAME:  MACRO FOR WRITING MEMBER RECORD
FUNCTION:

A MEMBER RECORD GIVEN IN THE WORK AREA IS APPENDED TO A MEMBER
OR USED TO UPDATE AN EXISTING MEMBER RECORD.

- MF=APPEND & REQUEST=NONCONTINUOUS
THE RECORDS GIVEN IN THE WORK AREA ARE STORED SEQUENTIALLY
AT THE END OF THE ALREADY WRITTEN RECORDS OF THE MEMBER.
IF THE NUMBER OF RECORDS  GIVEN IN PARAMETER 'LRECNO' IS ZERO,
THE WORK AREA IS EXPECTED TO BE FILLED WITH RECORDS WHICH ARE
TO BE WRITTEN TO THE MEMBER. IF MORE THAN ONE RECORD ARE TO
BE WRITTEN THEY ARE SUPPOSED TO HAVE A 2-BYTES LENGTH FIELD
PRECEDING EACH ONE IN THE WORK AREA. IF LRECNO=1, THE FIRST
RECORD (IN THE LENGTH OF THE VALUE OF PARAMETER 'LRECL') WHICH
MUST NOT HAVE A LENGTH FIELD IN FRONT IS APPENDED TO THE MEMBER.

- MF=APPEND & REQUEST=CONTINUOUS:
THE CONTENTS OF THE WORK AREA (ITS LENGTH IS SPECIFIED BY
PARAMETER 'WALEN') IS STORED CONTINUOUSLY AT THE END OF THE
ALREADY WRITTEN MEMBER PART.

- MF=UPDATE:
THIS FUNCTION WORKS ONLY WITH MEMBERS OF RECORD TYPE 'UNDEFINED'.
AN ALREADY WRITTEN PART OF A MEMBER IS EXCHANGED IN-PLACE WITH
THE CONTENTS OF THE WORK AREA.
RETURN CODES FROM SERVICE FUNCTION:
   0       REQUEST HAS BEEN HONORED
  12       LIBRARY IS FULL
  16       EXTERNAL ERROR WITH FEEDBACK CODE
  20       INTERNAL ERROR WITH FEEDBACK CODE
  36       MEMBER CHAINING ERROR
```

INLMRCLM

```
MACRO NAME:  INLMRCLM

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, BILINGUAL

DESCRIPTIVE NAME:  MACRO FOR RECLAIMING LIBRARY SPACE
FUNCTION:

THIS MACRO IS USED TO REUSE SPACE OF A LIBRARY/SUBLIBRARY
WHICH HAS BEEN COLLECTED IN THE SPACE RECLAMATION CHAINS OF
THE REQUESTED SUBLIBRARIES.

RETURN CODES FROM SERVICE FUNCTION:
   0       REQUEST HAS BEEN HONORED
  16       EXTERNAL ERROR WITH FEEDBACK CODE
  20       INTERNAL ERROR WITH FEEDBACK CODE
```

INLMRESN

```
MACRO NAME:  INLMRESN

MACRO TYPE:  INTERNAL LIBRARIAN MACRO, BILINGUAL

DESCRIPTIVE NAME:  MACRO FOR INVOKING INLPRESN
FUNCTION:

THE LIBRARIAN MODULE INLPRESN IS INVOKED FOR BUILDING THE
RESOURCE NAME OF A SUBLIBRARY.

REGISTER USAGE FOR INLMRESN MACRO:

  REG  0   FUNCTION INDICATOR
  REG  1   POINTER TO CONTROL BLOCK
  REG 13   ADDRESS OF SAVE AREA
  REG 14   RETURN ADDRESS
  REG 15   USED AS BRANCH REGISTER AND FOR RETURN CODES

  ALL OTHER REGISTERS REMAIN UNCHANGED

RETURN CODES FROM LANGUAGE PROCESSOR:
   0       MACRO EXPANSION SUCCESSFUL
   4       PARAMETERS HAVE BEEN IGNORED DUE TO CONFLICTS
   8       INVALID PARAMETERS SPECIFIED

RETURN CODES FROM SERVICE FUNCTION:
   0       REQUEST HAS BEEN HONORED
  16       EXTERNAL ERROR WITH FEEDBACK CODE
  20       INTERNAL ERROR WITH FEEDBACK CODE
```

INLMSCON

```
MACRO NAME:  INLMSCON

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, BILINGUAL

DESCRIPTIVE NAME:  MACRO FOR CONNECTING TO SUBLIBRARY
FUNCTION:

THE INLMSCON MACRO IS USED TO CONNECT TO AN EXISTING SUBLIBRARY
OR CREATE A NEW SUBLIBRARY WITHIN A CHAIN WHICH HAS BEEN DEFINED
AND ACCESSED USING THE LBRACCES MACRO.  WHEN USING INLMSCON TO
CREATE A NEW SUBLIBRARY, IT HAS THE EFFECT OF STOWING THE NEW
SUBLIBRARY ENTRY IN THE LIBRARY DIRECTORY.

- FOR CONNECT=OLD:
THE SUBLIBRARY MUST HAVE BEEN ADDED TO THE LCTS PREVIOUSLY WITH
DEFINE=OLD.
THE SUBLIBRARY DEFINED IN LIBINFO IS SEARCHED IN THE LIBRARY
DEFINED BY LIBINFO. IF THE SUBLIBRARY IS FOUND THE ACCESS CONTROL
BLOCK FOR IT IS INITIALIZED.
- FOR CONNECT=NEW:
THE SUBLIBRARY MUST HAVE BEEN ADDED TO THE LCTS PREVIOUSLY WITH
DEFINE=NEW. THE REQUESTOR MUST HAVE AN ALTER RIGHT FOR THE
SUBLIBRARY IF THE LIBRARY IS PROTECTED.
THE SUBLIBARY ACCESS CONTROL BLOCK IS INITIALIZED. IF THE
SUBLIBRARY ALREADY EXISTS, A RETURN CODE OF 8 IS RETURNED.
OTHERWISE, THE SUBLIBRARY DESCRIPTOR AND THE MEMBER INDEX ARE
INITIALIZED.

ANY CONNECTION TO ANOTHER LIBRARY OR SUBLIBRARY IS RELEASED
BEFORE THE REQUESTED SUBLIBRARY IS CONNECTED. THE CONNECTION
TO THE LIBRARY IS DONE IMPLICITLY, IF NECESSARY.
IF A MEMBER IS PREVIOUSLY CONNECTED THE FUNCTION WILL NOT BE
PERFORMED.

RETURN CODES FROM SERVICE FUNCTION:
   0       REQUEST HAS BEEN HONORED
   4       SUCCESSFUL, SUBLIBRARY WAS ALREADY CONNECTED
   8       SUBLIBRARY ALREADY EXISTS (FOR CONNECT=NEW)
           SUBLIBRARY DOES NOT EXIST (FOR CONNECT=OLD)
  12       LIBRARY IS FULL (NO SPACE AVAILABLE FOR CONNECT=NEW)
  16       EXTERNAL ERROR WITH FEEDBACK CODE
  20       INTERNAL ERROR WITH FEEDBACK CODE
  32       SECURITY VIOLATION
```

INLMSDIS

```
MACRO NAME:  INLMSDIS

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, BILINGUAL

DESCRIPTIVE NAME:  MACRO FOR DISCONNECTING FROM SUBLIBRARY
FUNCTION:

THE SUBLIBRARY IS DISCONNECTED FROM ACCESS AND STORAGE USED FOR
INTERNAL CONTROL BLOCKS IS FREED.
IF A MEMBER WAS CONNECTED ANY PRIVATE BUFFERS HAVING BEEN
CHANGED ARE WRITTEN TO THE LIBRARY. THE LIBRARY REMAINS
CONNECTED.
RETURN CODES FROM SERVICE FUNCTION:
   0       REQUEST HAS BEEN HONORED
   8       SUBLIBRARY WAS NOT CONNECTED
  12       LIBRARY IS FULL
  16       EXTERNAL ERROR WITH FEEDBACK CODE
  20       INTERNAL ERROR WITH FEEDBACK CODE
```

INLMSLD

```
MACRO NAME:  INLMSLD

MACRO TYPE:  INTERNAL LIBRARIAN MACRO, PLS

DESCRIPTIVE NAME:  MACRO FOR INVOKING INLPSLD
FUNCTION:

THE LIBRARIAN MODULE INLPSLD IS INVOKED FOR BUILDING A SECOND
LEVEL DIRECTORY FOR THE PHASES OF THE CONNECTED SUBLIBRARY.

REGISTER USAGE FOR INLMSLD MACRO:

  REG  0   FUNCTION INDICATOR
  REG  1   POINTER TO CONTROL BLOCK
  REG 13   ADDRESS OF SAVE AREA
  REG 14   RETURN ADDRESS
  REG 15   USED AS BRANCH REGISTER AND FOR RETURN CODES

  ALL OTHER REGISTERS REMAIN UNCHANGED

RETURN CODES FROM LANGUAGE PROCESSOR:
   0       MACRO EXPANSION SUCCESSFUL
   4       PARAMETERS HAVE BEEN IGNORED DUE TO CONFLICTS
   8       INVALID PARAMETERS SPECIFIED

RETURN CODES FROM SERVICE FUNCTION:
   0       REQUEST HAS BEEN HONORED
  16       EXTERNAL ERROR WITH FEEDBACK CODE
  20       INTERNAL ERROR WITH FEEDBACK CODE
```

INLMSTOW

```
MACRO NAME:  INLMSTOW

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, BILINGUAL

DESCRIPTIVE NAME:  MACRO FOR UPDATING DIRECTORY
FUNCTION:

UPDATE(ADD/DELETE/RENAME) SUBLIBRARY AND MEMBER INDEX ON A
LIBRARY (CONSISTENTLY FOR CONCURRENT READ REQUESTS).
UDATE SYSTEM DIRECTORY LIST (SDL) FOR PHASES IN THE
SYSTEM SUBLIBRARY IJSYSRS.SYSLIB.

THE SUBLIBRARY OR MEMBER DIRECTORY OF THE CONNECTED LIBRARY OR
SUBLIBRARY IS UPDATED BY THE ENTRIES OF THE STOW TABLE AS
INDICATED BY THE 'MF' PARAMETER.

LEVEL=MEMBER
         ===> MF=ADD:
         ADD ALL DIRECTORY ENTRIES GIVEN IN THE STOW TABLE
         TO THE DIRECTORY ENTRY ACCORDING TO THE COLLATING
         SEQUENCE AND UPDATE THE HIGHER LEVEL INDEX.
         - THE INSERTION OF THE ENTRIES AND THE SETTING OF THE
         TIME STAMP IS DONE ACCORDING TO THE OLDNTRY PARAMETER:
         NOREPLACE & MBR EXISTS: NO INSERTION AT ALL
         NOREPLACE & MBR DOES NOT EXIST: SET ORIGINATION TIME
         REPLACE & MBR EXISTS: SET UPDATE TIME
         REPLACE & MBR DOES NOT EXIST: SET ORIGINATION TIME.
         WHENEVER THE ORIGINATION TIME STAMP IS SET THE UPDATE
         TIME STAMP IS CLEARED.
         - SECURITY CHECK DEPENDS ON THE EXISTENCE OF THE
         MEMBER IN THE DIRECTORY: IF THE MEMBER ALREADY EXISTS
         AN UPDATE ACCESS RIGHT FOR THE MEMBER IS REQUIRED,
         OTHERWISE AN ALTER RIGHT. ANY SECURITY VIOLATION IS
         FLAGGED IN THE ARGUMENT (LARGSEC). A MEMBER WHICH
         IS MSHP-CONTROLLED CAN ONLY BE REPLACED BY MSHP.
         - IF A MEMBER IS REPLACED, ITS DIRECTORY ENTRY IS
         UPDATED IN-PLACE AND - FOR PURPOSE=UPDATE (MEMBER
         SPACE IS AFFECTED) - THE SPACE OF THE OLD VERSION IS
         REUSED.
         - IF A MEMBER CANNOT BE ADDED (BECAUSE OF SECURITY
         VIOLATION, BECAUSE AN OLD VERSION SHOULD NOT BE
         REPLACED OR BECAUSE A MEMBER IS LOCKED)
         THE ALREADY WRITTEN MEMBER SPACE IS FREED
         (ONLY FOR PURPOSE=UPDATE). THIS IS REPORTED IN THE
         ARGUMENT (LARGDEL).
         - IF A MEMBER HAS BEEN SUCCESSFULLY ADDED TO THE
         DIRECTORY THIS WILL BE REPORTED IN THE ARGUMENT
         (LARGCAT).
         - THE UPDATING OF THE HIGHER LEVEL IS DONE BOTTOM-UP.
         - FINALLY THE SUBLIBRARY DIRECTORY ENTRY IS UPDATED.
```

```
         ===> MF=DELETE:
         ALL MEMBERS GIVEN IN THE STOW TABLE ARE DELETED FROM
         MEMBER DIRECTORY OF THE CONNECTED SUBLIBRARY AND THE
         HIGHER LEVEL INDEX IS UPDATED BOTTOM-UP. FINALLY, THE
         SUBLIBRARY DESCRIPTOR IS UPDATED.
         - MEMBER ALTER RIGHT IS REQUIRED.
         ===> MF=RENAME:
         ALL ENTRIES IN THE STOW TABLE REQUESTED FOR RENAME ARE
         PROCESSED.
         - ALL ENTRIES IN THE STOW TABLE WITH AN ODD INDEX ARE
         REGARDED AS THE ENTRIES TO BE RENAMED AND ALL ENTRIES
         WITH AN EVEN INDEX ARE REGARDED AS THE CORRESPONDING
         NEW KEYS (NO SORT OF THE STOW TABLE HAS TO BE DONE).
         - IF A RENAME IS DONE, THE OLD KEY EXISTS AND THE
         NEW NAME DOES NOT EXIST WITHIN THE SUBLIBRARY.
         - WHEN A RENAME TO TYPE "PHASE"  IS REQUESTED
         THE CONCERNED MEMBER MUST HAVE THE CHARACTERISTICS OF
         A PHASE.
         - A RENAME IS DONE BY FIRST PERFORMING A COMPLETE
         ADD OF THE NEW ENTRY IN THE MEMBER INDEX AND THEN
         DELETING THE OLD ENTRY. SO, TEMPORARILY TWO DIRECTORY
         ENTRIES POINTING TO THE SAME MEMBER SPACE ARE IN THE
         SUBLIBRARY.
         - BY THIS ALGORITHM THE MEMBER INDEX CAN GROW AND
         SHRINK, THUS RESERVING LIBRARY BLOCKS WHICH ARE
         AFTERWARDS IN THE SPACE RECLAMATION CHAIN.
         - FINALLY, THE SUBLIBRARY DESCRIPTOR IS UPDATED.
         - FOR A RENAME THE CALLER MUST HAVE THE ALTER MEMBER
         RIGHT FOR THE OLD NAME AS WELL AS FOR THE NEW ONE.
         ===> MF=PURGE:
         ALL ENTRIES GIVEN IN THE STOWTABLE WHICH MUST NOT HAVE
         DIRECTORY ENTRIES IN THE CONNECTED SUBLIBRARY ARE USED
         TO FREE IMMEDIATELY THEIR RESERVED MEMBER SPACE.
         - MEMBER UPDATE RIGHT IS REQUIRED.

LEVEL=SUBLIB
         ===> MF=DELETE:
         THE SUBLIBRARY GIVEN IN THE STOWTABLE IS DELETED FROM
         THE CONNECTED LIBRARY.
         - WITH A STOW REQUEST, ONLY ONE SUBLIBRARY CAN BE
         DELETED.
         - THIS SUBLIBRARY HAS TO BE UNIQUELY ASSIGNED TO THE
         REQUESTOR.
         - ALL MEMBERS OF THE SUBLIBRARY ARE DELETED AND THE
         LIBRARY BLOCKS IMMEDIATELY FREED.
         - THE REQUESTOR NEEDS ALTER ACCESS RIGHT TO THE
         SUBLIBRARY TO BE DELETED.
         ===> MF=RENAME:
         THE REQUESTED SUBLIBRARY IS RENAMED IN PLACE IF IT
         EXISTS, OTHERWISE A RETURN CODE IS GIVEN.
         - THE REQUESTOR NEEDS ALTER ACCESS RIGHT TO THE
         SUBLIBRARY TO BE RENAMED.
```

```
         ===> MF=EMPTY:
         THE SUBLIBRARY GIVEN IN THE STOW TABLE IS EMPTIED (ALL
         MEMBERS ARE DELETED, BUT THE SUBLIBRARY STILL EXISTS)
         - WITH A STOW REQUEST, ONLY ONE SUBLIBRARY CAN BE
         EMPTIED.
         - THIS SUBLIBRARY GETS A NEW TIME-STAMP.
         - ALL MEMBERS OF THE SUBLIBRARY ARE DELETED AND THE
         LIBRARY BLOCKS FREED ACCORDING TO AUTOMATIC SPACE
         RECLAMATION RULES.
         - THE REQUESTOR NEEDS UPDATE ACCESS RIGHT TO THE
         SUBLIBRARY TO BE EMPTIED.
         ===> MF=CHANGE:
         ATTRIBUTES OF THE SUBLIBRARY GIVEN IN THE STOW TABLE ARE
         CHANGED:
         A) REUSE
         - WITH A STOW REQUEST, ONLY ONE SUBLIBRARY CAN BE
         EMPTIED.
         - ALL MEMBER LBS OF THE SPACE RECLAMATION CHAIN ARE
         FREED IMMEDIATELY IF REUSE IS SET FROM AUTOMATIC TO
         IMMEDIATE.
         - THE REQUESTOR NEEDS ALTER  ACCESS RIGHT TO THE
         SUBLIBRARY TO BE CHANGED.

LEVEL=LIB
         ===> MF=DELETE:
         THE LIBRARY WHICH IS CONNECTED IS DELETED.
         NOTE: THE VTOC ENTRY FOR THE LIBRARY STILL EXISTS
               AFTERWARDS.
         - WITH A STOW REQUEST, ONLY ONE LIBRARY    CAN BE
         DELETED.
         - THIS LIBRARY    HAS TO BE UNIQUELY ASSIGNED TO THE
         REQUESTOR.
         - THE LIBRARY NAME IN THE LIBRARY DESCRIPTOR GETS THE
         LABEL "DELETED".
         - THE REQUESTOR NEEDS ALTER ACCESS RIGHT TO THE LIB.

REGISTER USAGE FOR INLMSTOW MACRO:

  REG  0   FUNCTION INDICATOR
  REG  1   POINTER TO CONTROL BLOCK
  REG 13   ADDRESS OF SAVE AREA
  REG 14   RETURN ADDRESS
  REG 15   USED AS BRANCH REGISTER AND FOR RETURN CODES

  ALL OTHER REGISTERS REMAIN UNCHANGED
```

```
RETURN CODES FROM LANGUAGE PROCESSOR:
   0       MACRO EXPANSION SUCCESSFUL
   4       PARAMETERS HAVE BEEN IGNORED DUE TO CONFLICTS
   8       INVALID PARAMETERS SPECIFIED
RETURN CODES FROM SERVICE FUNCTION:
   0       REQUEST HAS BEEN HONORED
   4       PARTIALLY SUCCESSFUL, SEE STOW TABLE RETURN CODES
  12       LIBRARY IS FULL
  16       EXTERNAL ERROR WITH FEEDBACK CODE
  20       INTERNAL ERROR WITH FEEDBACK CODE
  32       SECURITY VIOLATION

THE STOW TABLE RETURN CODES ARE DEFINED BY THE FLAG BYTE (LARGSW)
IN THE ARGUMENT ENTRY (INLCLARG):

- FOR MF=ADD:

LARGCAT LARGFND LARGSEC LARGDEL
   1       0       0       0      A NEW MEMBER WAS CATALOGED.
   1       1       0       0      AN EXISTING MEMBER WAS REPLACED
                                  (ONLY WITH OLDNTRY=REPLACE).
   0       1       0       *      AN EXISTING MEMBER WAS NOT RE-
                                  PLACED (ONLY WITH OLDNTRY=
                                  NOREPLACE).
   0       0       1       *      SECURITY VIOLATION OCCURRED FOR
                                  THE MEMBER.
   0       0       0       1      THE MEMBER SPACE HAS BEEN FREED
                                  (IN ORDER TO GET FREE SPACE TO
                                  COMPLETE THE INSERTION OF OTHER
                                  DIRECTORY ENTRIES).

LARGCAT LARGFND LARGDLCK LARGDEL
   0       1        1       1     A MEMBER WAS NOT REPLACED
                                  BECAUSE IT WAS LOCKED

*   : FOR PURPOSE=UPDATE THE MEMBER SPACE IS FREED
      (LARGDEL=1, OTHERWISE 0).

- FOR MF=DELETE:

LARGDEL LARGFND LARGSEC LARGDLCK
   1       1       0      *       THE ENTRY  HAS BEEN DELETED
                                  FROM THE DIRECTORY.
   0       0       0      0       THE ENTRY WAS NOT FOUND IN THE
                                  DIRECTORY.
   0       0       1      *       SECURITY VIOLATION OCCURRED FOR
                                  THE ENTRY (IS SKIPPED).
   0       1       1      1       MEMBER WAS LOCKED
```

- FOR MF=RENAME:

```
LARGREN LARGFND LARGCAT LARGDEL LARGSEC
   1       1       0       1       0     THE OLD ENTRY EXISTED
                                         AND HAS BEEN DELETED.
   1       0       1       0       0     THE NEW ENTRY DID NOT
                                        EXIST AND HAS BEEN CATAL.
   0       0       0       0       0     OLD NAME NOT FOUND.
   0       1       0       0       0     NEW NAME ALREADY EXISTS.
   0       0       0       0       1     SECURITY VIOLATION.

LARGREN LARGFND LARGCAT LARGDEL LARGDLCK
    0      1       0       0       1     NEW NAME EXISTS AND IS LOCKED
```

- FOR MF=CHANGE:

```
LARGFND LARGCAT LARGSEC
   1       1       0                   ATTRIBUTE OF THE ENTRY
                                       HAS BEEN CHANGED.
   1       0       0                   ATTRIBUTE ALREADY OWNED
                                       BY ENTRY.
   0       0       0                   ENTRY NOT FOUND.
   0       0       1                   SECURITY VIOLATION.
```

LBRACCCB

```
MACRO NAME:  LBRACCCB

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, BILINGUAL

DESCRIPTIVE NAME:  MACRO FOR HIGH LEVEL ACCESS CONTROL BLOCK
FUNCTION:

THE CONTROL BLOCK FOR THE LBRACCES MACRO AND A LOCK
TABLE ENTRY ARE DEFINED AND FILLED UP WITH THE SPECIFIED
PARAMETERS.
```

LBRACCES

```
MACRO NAME:  LBRACCES

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, BILINGUAL

DESCRIPTIVE NAME:  MACRO FOR HIGH LEVEL ACCESS TO LIBRARY
FUNCTION:

THIS MACRO ALLOWS THE LEVEL 2 USER TO BUILD LIBRARY CHAINS,
ACCESS THEM ONCE THEY HAVE BEEN BUILT, AND DELETE THEM WHEN
THEY ARE NO LONGER NEEDED.

FOR MF=ADDLIB:
THE REQUESTED LIBRARY/SUBLIBRARY IS ADDED TO THE LIBRARY CONTROL
TABLES FOR 'MF=ADDLIB'.  UP TO FIFTEEN LIBRARY / SUBLIBRARY
MAY BE ADDED TO A CHAIN.
IT IS POSSIBLE TO BUILD SINGLE LIBRARY/SUBLIBRARY ENTRIES OR
A CHAIN OF LIBRARIES/SUBLIBRARIES.
THE ENTRIES/CHAINS ARE ADDRESSED BY THE PARAMETER TRIPLE
(LIBTYPE,LIBUSE,CHAIN) AND THE TASK ID OF THE REQUESTOR.
IF THE LIBTYPE IS NOT "LBR" A CHAIN EXISTS ONLY FOR LIBUSE=
SEARCH, BOTH FOR CHAIN=TEMP AND CHAIN=PERM. SINGLE ENTRIES
EXIST FOR LIBUSE=CATALOG|TO|FROM. THESE ENTRIES/CHAINS ARE
RESERVED FOR LIBDEF STATEMENTS.
IF LIBTYPE=LBR, CHAINS EXIST FOR LIBUSE=SEARCH|CATALOG|ACCESS,
ONLY FOR CHAIN=PERM. SPECIFICATION OF CHAIN=TEMP WILL ADDRESS
A SINGLE ENTRY OF THE SPECIFIED LIBUSE PARAMETER.
FOR A TASK RUNNING UNDER ICCF, THE REQUESTOR CAN SPECIFY OWN
CHAINS/ENTRIES WITH LIBUSE=USER AND A USER-DEFINED CHAIN NAME
WITH THE CHAINED PARAMETER.
TO BUILD A CHAIN THE LBRACCDS HAS TO BE INITIALIZED. SUCCESSIVE
LBRACCES REQUESTS FOR THE SAME (LIBTYPE, LIBUSE, CHAIN) PARAMETER
TRIPLE CONCATENATES THE LIBRARY/SUBLIBRARY COMBINATION TOGETHER,
UNLESS THE LBRACCDS CONTROL BLOCK IS INITIALIZED NEW.
SUCCESSIVE LBRACCES REQUEST FOR A SINGLE ENTRY WILL ALWAYS
REPLACE THE OLD BY THE NEW ONE.
IF LEVEL=LIB|BOTH IS SPECIFIED, AN OPEN IS PERFORMED FOR THE
LIBRARY. FOR PERFORMANCE REASONS, THE OPEN CAN BE AVOIDED FOR
A LIBRARY ALREADY ADDED IN A CHAIN (OR SINGLE ENTRY) WHEN LEVEL=
SUBLIB IS SPECIFIED AND THE LIBINFO VALUE RETURNED BY THAT VERY
REQUEST IS PASSED TO THE SERVICE.
```

```
- FOR MF=GET:

THE ACTIVE LIBRARY/SUBLIBRARY CHAIN (ENTRY) CAN BE RETRIEVED.
THE INVOCATIONS OF LBRACCES MAY BE CHAINED TOGETHER BY PASSING
THE LIBINFO FIELD FROM ONE TO ANOTHER.  TO BEGIN A CHAINING
SEQUENCE, LIBINFO IS SET TO ZERO
AND LBRACCES WITH 'MF=GET' COULD BE ISSUED.  THIS WOULD ESTABLISH
ACCESS TO THE FIRST LIBRARY / SUBLIBRARY.  TO ESTABLISH, ACCESS
TO THE SECOND LIBRARY / SUBLIBRARY,  THE LBRACCES MACRO WITH
'MF=GET' WOULD BE ISSUED AND THE SAME LIBINFO FIELD WOULD BE
USED.  THE PROCESS WOULD BE REPEATED FOR THE FURTHER LIBRARIES/
SUBLIBRARIES IN THE CHAIN.
IF THE 'GET' REQUEST CAN BE HONORED, THE ADDRESS OF THE
LOT ENTRY AND THE ADDRESS OF THE LDT ENTRY ARE RETURNED
AS FULLWORDS IN THAT SEQUENCE AT THE ADDRESS SPECIFIED
BY LIBINFO.  IF THE 'GET' REQUEST WAS FOR A 'P' LIBRARY,
THE POINTER TO IT IS RETURNED IN THE THIRD FULLWORD
ADDRESSED BY LIBINFO. THE SECOND WORD ADDRESSED BY LIBINFO
ALWAYS CONTAINS THE POINTER TO THE 'C' LIBRARY.
THE FOURTH WORD ADDRESSED BY LIBINFO CONTAINS THE ADDRESS OF
THE SDT ENTRY IF SPECIFIED, OTHERWISE IT IS 0.
FOR A CHAINING REQUEST, THE POINTERS RETURNED IN THE
PREVIOUS CALL HAVE TO BE PASSED BACK IN 'LIBINFO'.
FOR THE FIRST CHAINING REQUEST 'LIBINFO' HAS TO CONTAIN
ZEROS.

- FOR MF=REMOVE:

AN ENTRY/CHAIN SPECIFIED BY (LIBTYPE,LIBUSE,CHAIN) IS DELETED
FOR 'MF=REMOVE'.

THE REQUESTED LIBRARY IS LOCKED OR UNLOCKED ACCORDING
TO THE MF SPECIFICATION 'LOCK|UNLOCK' WITH FAIL=WAIT.

THE REQUESTED LIBRARY IS EXTENDED IN VSAM MANAGED SPACE FOR
'MF=EXTEND'.

FOR 'MF=EXTUPD' REQUESTS THE EDT OF THE LIBRARY
IN VSAM MANAGED SPACE ON A SHARED DASD IS  UPDATED BY
THE ADDITIONAL EXTENT(S) ALLOCATED BY ANOTHER CPU.

IF A LIBRARY IS ADDED TO THE LIBRARY CONTROL TABLES, AN
OPEN HAS TO BE PERFORMED. TO ALLOW
USAGE OF THE DOS/VSE ACCESS CONTROL FACILITY, THE OPEN IS
PERFORMED WITH THE FOLLOWING ACCESS ATTRIBUTE:

    LIBUSE          OPEN FOR

    TO               INPUT
    FROM             INPUT
    SEARCH           INPUT
    ACCESS           OUTPUT IF DEFINE=NEW
                     INPUT  IF DEFINE=OLD
    CATALOG          INPUT
    USER             INPUT
```

```
REGISTER USAGE FOR LBRACCES MACRO:

  REG  0   FUNCTION INDICATOR
  REG  1   POINTER TO CONTROL BLOCK
  REG 13   ADDRESS OF SAVE AREA
  REG 14   RETURN ADDRESS
  REG 15   USED AS BRANCH REGISTER AND FOR RETURN CODES

  REGISTERS 0 AND 1 ARE ALSO USED BY THE MACROS GENDTL MODDTL

  ALL OTHER REGISTERS REMAIN UNCHANGED

RETURN CODES FROM LANGUAGE PROCESSOR:
   0       MACRO EXPANSION SUCCESSFUL
   4       PARAMETERS HAVE BEEN IGNORED DUE TO CONFLICTS
   8       INVALID PARAMETERS SPECIFIED

RETURN CODES FROM SERVICE FUNCTION:
   0       REQUEST HAS BEEN HONORED
   4       MF=GET:
           END OF CHAIN HAS BEEN REACHED FOR A CHAINING REQUEST
           POINTERS ARE STILL POINTING TO LAST VALID LIB/SUBLIB
   8       MF=GET: REQUESTED LIBRARY HAS NOT BEEN DEFINED
           MF=ADDLIB: END OF CHAIN CAPACITY REACHED.
           MF=EXTEND: NO EXTENSION POSSIBLE.
  12       MF=ADDLIB:
           FOR LEVEL=LIB AND DEFINE=NEW:
               LIBRARY ALREADY EXISTS
           FOR LEVEL=LIB AND DEFINE=OLD:
               LIBRARY DOES NOT EXIST
           FOR LEVEL=SUBLIB AND DEFINE=NEW:
               SUBLIBRARY ALREADY EXISTS
           FOR LEVEL=SUBLIB AND DEFINE=OLD:
               SUBLIBRARY DOES NOT EXIST
  16       EXTERNAL ERROR WITH FEEDBACK CODE (LBRRET FIELD).
  20       INTERNAL ERROR WITH FEEDBACK CODE (LBRRET FIELD).
  32       SECURITY VIOLATION
 255       LAMB MISSING

  FOR LOCK AND UNLOCK THE RETURN CODES FROM THE SUPERVISOR
  SERVICE LOCK|UNLOCK ARE HANDLED BY THE MACRO EXPANSION.
```

LBRBLDCB

```
MACRO NAME:  LBRBLDCB

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, PLS

DESCRIPTIVE NAME: BUILD PARAMETER BLOCK FOR MODULE IJBLBBRN.
FUNCTION:

          DEFINES A PARAMETER BLOCK WHICH IS USED FOR
          INVOCATION OF MODULE IJBLBBRN.
          MODULE IJBLBBRN IS INVOKED VIA MACRO LBRBLDRN.
          IJBLBBRN OPENS OR EXTENDS A LIBRARY AND RETURNS
          LDT/EDT/DDT ENTRIES.
```

LBRBLDRN

```
MACRO NAME:  LBRBLDRN

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, PLS

DESCRIPTIVE NAME:  OPEN/EXTEND A LIBRARY.
FUNCTION:

          THE PARAMETER BLOCK DEFINED BY AREA IS UPDATED
          (BY INVOKING MACRO LBRBLDCB) AND MODULE IJBLBBRN
          IS INVOKED.
          THE LIBRARY IS OPENED OR EXTENDED AND THE
          LDT/EDT/DDT ENTRIES ARE RETURNED.

RETURN CODES:

             0             COMPLETED SUCCESSFULLY
             8             LIBRARY NOT EXTENDABLE
            12             LIBRARY DOES ALREADY EXIST
                           (FOR OUTPUT)
            16             USER ERROR
                           (THIS INCLUDES "LIBRARY DOES NOT EXIST)
            20             INTERNAL SYSTEM ERROR
            32             SECURITY VIOLATION
```

LBRCTUPD

```
MODULE LBRCTUPD MACRO FOR UPDATING  LIBRARY CONTROL TABLES

  MODULE NAME: LBRCTUPD     - LIBRARIAN MACRO

 DESCRIPTIVE NAME: UPDATE LIBRARY CONTROL TABLES
 FUNCTION:   - FOR MF = DCL | GEN | MOD | MAP
               DEFINITION, MODIFICATION AND/OR MAPPING OF
               REQUEST BLOCK FOR UPDATING LIBRARY CONTROL TABLES

               FOR THE DEFINITION AND/OR MAPPING OF THE REQUEST
               BLOCK THE MACRO TOLCRQL IS INVOKED.

             - FOR MF = ADD | CLEAR | EXTEND
               MODIFICATION OF REQUEST BLOCK FOR UPDATING LIBRARY
               CONTROL TABLES.
               INTERFACE TO THE SVA-PHASE $IJBLBR, MODULE IJBCTUPD
               FOR UPDATING LIBRARY CONTROL TABLES.

      RETURN CODES:

              0             COMPLETED SUCCESSFULLY
              4             COMPLETED SUCCESSFULLY -
                                NO ACTIVE ENTRIES IN LOT ROW
              8             MAXIMUM CHAIN REACHED
             12             FOR DEFINE=NEW, LEVEL=LIB:
                               LIBRARY DOES ALREADY EXIST
                            FOR DEFINE=OLD, LEVEL=LIB:
                               LIBRARY DOES NOT EXIST
                            FOR DEFINE=NEW, LEVEL=SUBLIB|BOTH:
                               SUBLIBRARY DOES ALREADY EXIST
                            FOR DEFINE=OLD, LEVEL=SUBLIB|BOTH:
                               SUBLIBRARY DOES NOT EXIST
             16             EXTERNAL SYSTEM ERROR
             20             INTERNAL SYSTEM ERROR
             32             SECURITY VIOLATION

             IF THE RETURN CODE IS GREATER OR EQUAL TO 16,
             A MESSAGE HAS BEEN STORED IN THE
             CORRESPONDING I/O AREA SPECIFIED IN THE LAMB.
```

LBRLCTAC

```
MACRO NAME:  LBRLCTAC

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, BILINGUAL

DESCRIPTIVE NAME:  MACRO FOR ACCESS TO LIBRARY CONTROL TABLES
FUNCTION:

THIS MACRO PROVIDES THE CALLER WITH THE INFORMATION ABOUT
A SPECIFIC LIBRARY/SUBLIBRAY WHICH HAS BEEN ADDED INTO THE
LIBRARY CONTROL TABLES.

IF THE REQUEST CAN BE HONORED, THE ADDRESS OF THE LOT
ENTRY AND THE ADDRESSES OF THE 'C'- AND 'P'- LDT AND SDT ENTRIES
RETURNED AS FULLWORDS IN THAT SEQUENCE AT THE ADDRESS
SPECIFIED BY 'LIBINFO' TOGETHER WITH A RETURN CODE OF 0
IN REGISTER 15. THE THIRD FULLWORD CONTAINS ZEROS IF THE
REQUESTED LIBRARY WAS NOT A 'P'- LIBRARY.
RETURN CODE 4 INDICATES THAT THE END OF A CHAIN
HAS BEEN REACHED, THE RETURNED LOT AND LDT POINTERS ARE
STILL POINTING TO THE LAST LIBRARY IN THE CHAIN.
RETURN CODES FROM SERVICE FUNCTION:
   0       REQUEST HAS BEEN HONORED
   4       END OF CHAIN REACHED FOR A 'NEXT' REQUEST
            POINTERS ARE STILL POINTING TO LAST VALID
            LIBRARY.
   8       REQUESTED LIBRARY HAS NOT BEEN DEFINED
 255       SYSTEM ERROR: INVALID PARAMETERS HAVE BEEN
            SPECIFIED FOR GET REQUEST
            A FEEDBACK CODE IS RETURNED IN LBRLRET
```

LBRLCTCB

MACRO NAME: LBRLCTCB

MACRO TYPE: EXTERNAL LIBRARIAN MACRO, BILINGUAL

DESCRIPTIVE NAME: MACRO FOR CONTROL BLOCK FOR ACCESS TO LIBRARY
FUNCTION:

THE CONTROL BLOCK FOR THE LBRLCTAC MACRO IS DEFINED
AND FILLED UP WITH THE SPECIFIED PARAMETERS.

LBRUPDAT

```
MACRO NAME:  LBRUPDAT

MACRO TYPE:  EXTERNAL LIBRARIAN MACRO, PLS

DESCRIPTIVE NAME: SET/RESET LDT/SDT ACCESS CONTROL FLAGS.
FUNCTION:

           CREATES A PARAMETER BLOCK AND INVOKES MODULE
           IJBLBULT.
           MODULE IJBLBULT SETS AND RESETS THE ACCESS CONTROL
           FLAGS LDTENOAC/SDTENOAC,LDTEDELT/SDTEDELT AND
           LDTENEWL/SDTENEWS IN AN LDT/SDT ENTRY. IT LOCKS
           AND UNLOCKS THE LIBRARY/SUBLIBRARY.

RETURN CODES:

             0             COMPLETED SUCCESSFULLY
             8             NOT UNIQUELY ASSIGNED FOR SET=NOACC
            16             NOT UNIQUELY ASSIGNED FOR SET=DEL
            20             INTERNAL SYSTEM ERROR
```

Further Modules and Macros

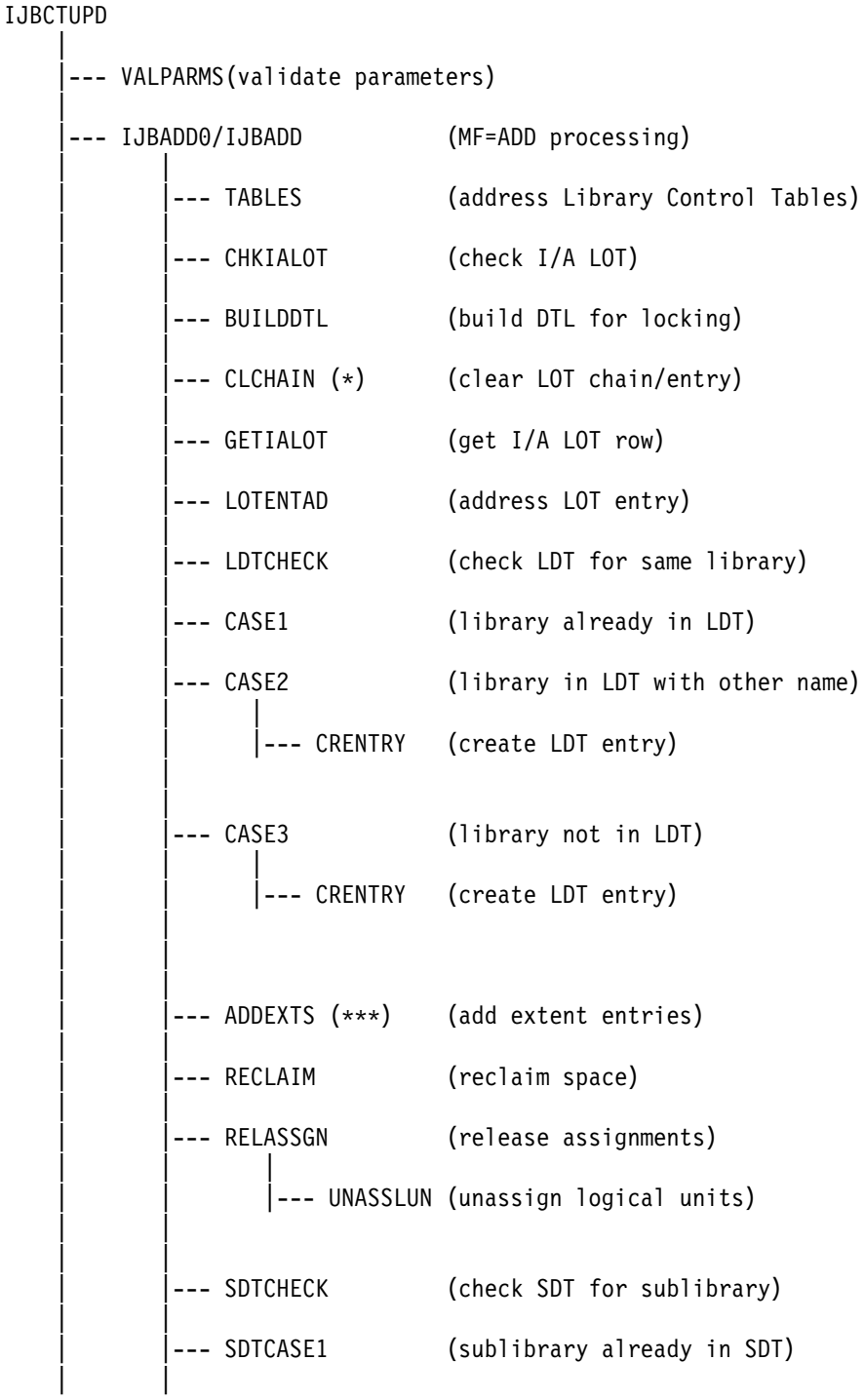
```
FNDSL    - DTFSL Find request
GETSL    - DTFSL Get request
INLPCONL - common version of INLACONL
INLPCONS - common version of INLACONS
INLPDOIO - common version of INLADOIO
INLPGST  - common version of INLAGST
INLPMCON - common version of INLAMCON
INLPRABF - common version of INLARABF
INLPRDTP - common version of INLARDTP
INLPRELB - common version of INLARELB
INLPREMB - common version of INLAREMB
INLPREM2 - common version of INLAREM2
INLPRESL - common version of INLARESL
INLPRES2 - common version of INLARES2
INLPRIPL - common version of INLARIPL
INLPRTI  - common version of INLARTI
INLPSTOW - common version of INLASTOW
INLPSTOX - common version of INLASTOX
INLPSTOY - common version of INLASTOY
INLPTI   - common version of INLATI
INLPTO   - common version of INLATO
INLTACC  - parser table for ACCESS command
INLTBKP  - parser table for BACKUP command
INLTCAT  - parser table for CATALOG command
INLTCATP - parser table for CATALP statement
INLTCATR - parser table for CATALR statement
INLTCATS - parser table for CATALS command
INLTCHAN - parser table for CHANGE command
INLTCOMP - parser table for COMPARE command
INLTCONN - parser table for CONNECT command
INLTCOPC - parser table for COPYC statement
INLTCOPP - parser table for COPYP statement
INLTCOPR - parser table for COPYR statement
INLTCOPS - parser table for COPYS statement
INLTCOPY - parser table for COPY   command
INLTDEF  - parser table for DEFINE command
INLTDEL  - parser table for DELETE command
INLTDELC - parser table for DELETC statement
INLTDELP - parser table for DELETP statement
INLTDELR - parser table for DELETR statement
INLTDELS - parser table for DELETS statement
INLTDICT - message text dictionary
INLTDSP  - parser table for DSPLY statement
INLTDSPC - parser table for DSPCH statement
INLTINPU - parser table for INPUT command
INLTIOLN - length values for I/O areas
INLTLCK  - parser table for LOCK command
INLTLD   - parser table for LISTDIR command
INLTLIST - parser table for LIST command
INLTMIGR - parser table for migration
INLTMLIB - message library
```

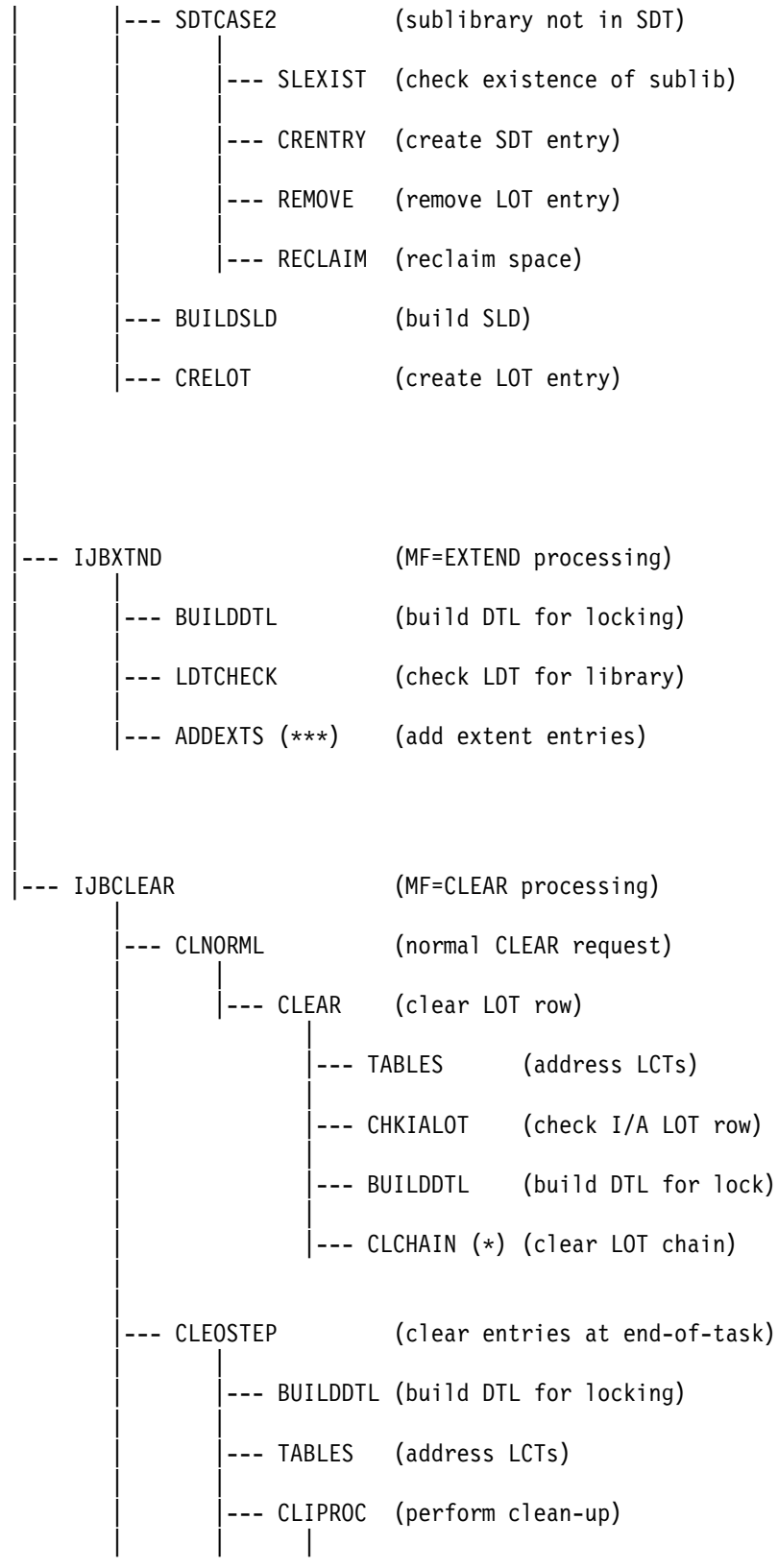
```
INLTMOVE - parser table for MOVE command
INLTPARS - parser table
INLTPUN  - parser table for PUNCH command
INLTPUNM - parser table for PUNCH statement
INLTREL  - parser table for RELEASE command
INLTREN  - parser table for RENAME command
INLTRENC - parser table for RENAMC statement
INLTRENP - parser table for RENAMP statement
INLTRENR - parser table for RENAMR statement
INLTRENS - parser table for RENAMS statement
INLTRES  - parser table for RESTORE command
INLTSRCH - parser table for SEARCH  command
INLTTEST - parser table for TEST command
INLTULCK - parser table for UNLOCK command
INLTUPD  - parser table for UPDATE command
INLXSAR1 - stand-alone module: dummy entry points
NTSL     - DTFSL Note request
PTSL     - DTFSL Point request
IJBDCTL  - access interface to dynamic class member
           (GET/PURGE/SAVE)
```

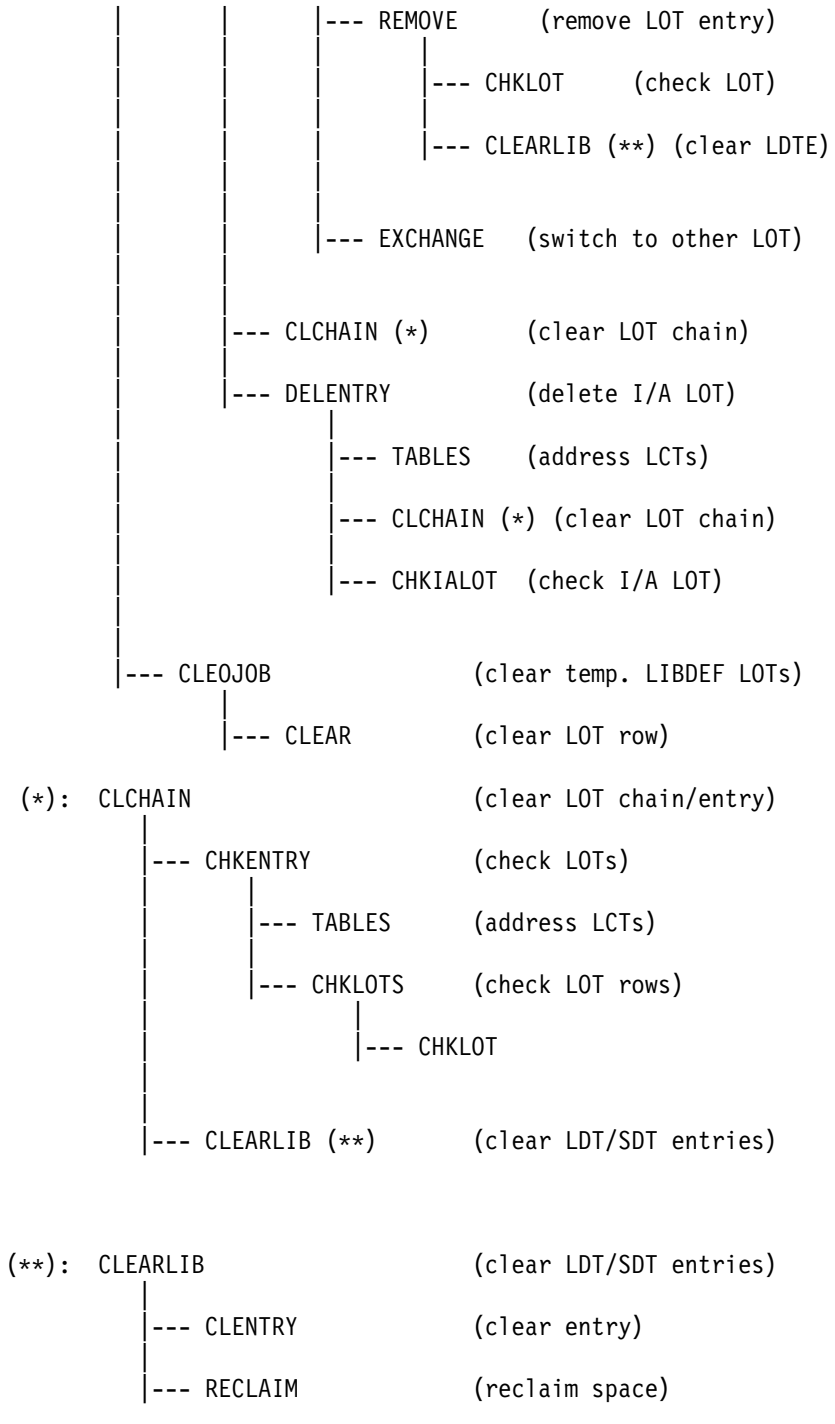
Program Organization

Calling Structure

IJBCTUPD

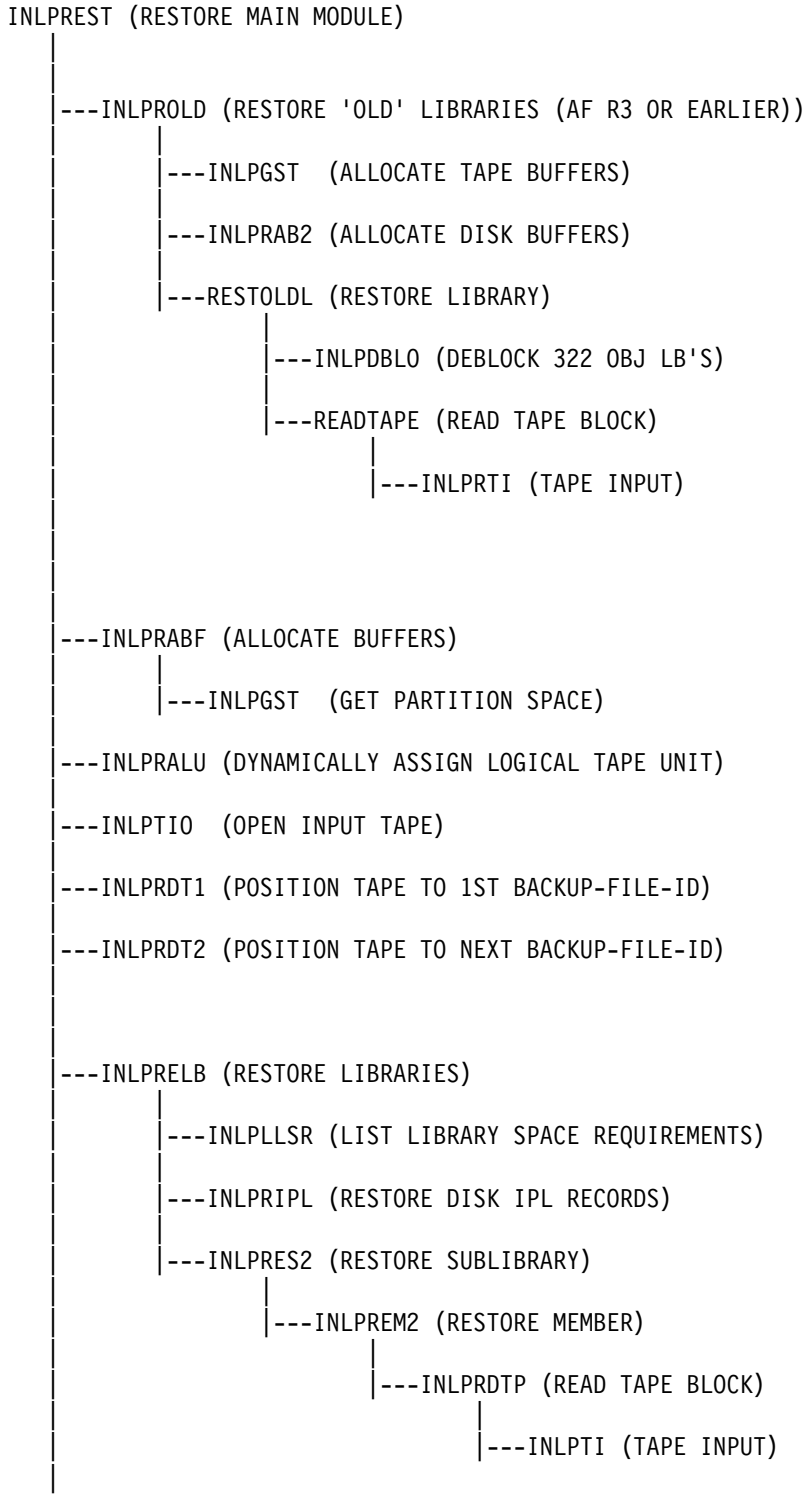


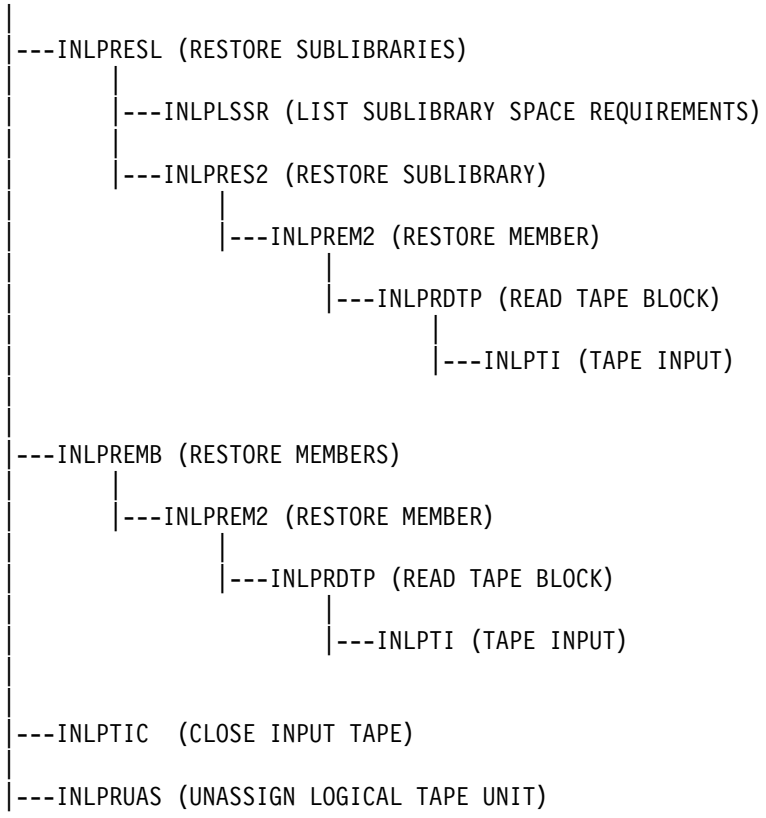

|--- SDTCASE2 (sublibrary not in SDT)
|--- SLEXIST (check existence of sublib)
|--- CRENTRY (create SDT entry)
|--- REMOVE (remove LOT entry)
|--- RECLAIM (reclaim space)
|--- BUILDSLD (build SLD)
|--- CRELOT (create LOT entry)
|--- IJBXTND (MF=EXTEND processing)
|--- BUILDDTL (build DTL for locking)
|--- LDTCHECK (check LDT for library)
|--- ADDEXTS (***) (add extent entries)
|--- IJBCLEAR (MF=CLEAR processing)
|--- CLNORML (normal CLEAR request)
|--- CLEAR (clear LOT row)
|--- TABLES (address LCTs)
|--- CHKIALOT (check I/A LOT row)
|--- BUILDDTL (build DTL for lock)
|--- CLCHAIN (*) (clear LOT chain)
|--- CLEOSTEP (clear entries at end-of-task)
|--- BUILDDTL (build DTL for locking)
|--- TABLES (address LCTs)
|--- CLIPROC (perform clean-up)


|--- REMOVE (remove LOT entry)
|--- CHKLOT (check LOT)
|--- CLEARLIB (**) (clear LDTE)
|--- EXCHANGE (switch to other LOT)
|--- CLCHAIN (*) (clear LOT chain)
|--- DELENTRY (delete I/A LOT)
|--- TABLES (address LCTs)
|--- CLCHAIN (*) (clear LOT chain)
|--- CHKIALOT (check I/A LOT)
|--- CLEOJOB (clear temp. LIBDEF LOTs)
|--- CLEAR (clear LOT row)
(*): CLCHAIN (clear LOT chain/entry)
|--- CHKENTRY (check LOTs)
|--- TABLES (address LCTs)
|--- CHKLOTS (check LOT rows)
|--- CHKLOT
|--- CLEARLIB (**) (clear LDT/SDT entries)
(**): CLEARLIB (clear LDT/SDT entries)
|--- CLENTRY (clear entry)
|--- RECLAIM (reclaim space)

```
(***):  ADDEXTS                    (add extent entries)
          |
          |--- DDTCHECK            (check DDT)
          |
          |--- ADDEVICE            (add DDT entry)
          |
          |--- UNASSLUN            (unassign logical unit)
          |
          |--- CRENTRY             (create entry)
          |
          |--- REMOVE              (remove LOT entry)
```

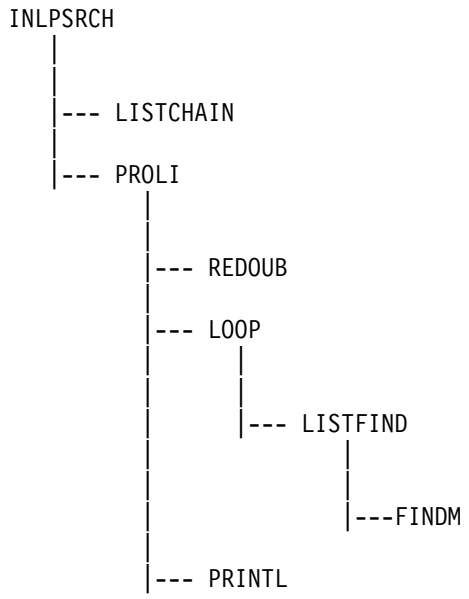
INLPREST


INLPREST (RESTORE MAIN MODULE)
INLPROLD (RESTORE 'OLD' LIBRARIES (AF R3 OR EARLIER))
INLPGST (ALLOCATE TAPE BUFFERS)
INLPRAB2 (ALLOCATE DISK BUFFERS)
RESTOLDL (RESTORE LIBRARY)
INLPDBLO (DEBLOCK 322 OBJ LB'S)
READTAPE (READ TAPE BLOCK)
INLPRTI (TAPE INPUT)
INLPRABF (ALLOCATE BUFFERS)
INLPGST (GET PARTITION SPACE)
INLPRALU (DYNAMICALLY ASSIGN LOGICAL TAPE UNIT)
INLPTIO (OPEN INPUT TAPE)
INLPRDT1 (POSITION TAPE TO 1ST BACKUP-FILE-ID)
INLPRDT2 (POSITION TAPE TO NEXT BACKUP-FILE-ID)
INLPRELB (RESTORE LIBRARIES)
INLPLLSR (LIST LIBRARY SPACE REQUIREMENTS)
INLPRIPL (RESTORE DISK IPL RECORDS)
INLPRES2 (RESTORE SUBLIBRARY)
INLPREM2 (RESTORE MEMBER)
INLPRDTP (READ TAPE BLOCK)
INLPTI (TAPE INPUT)

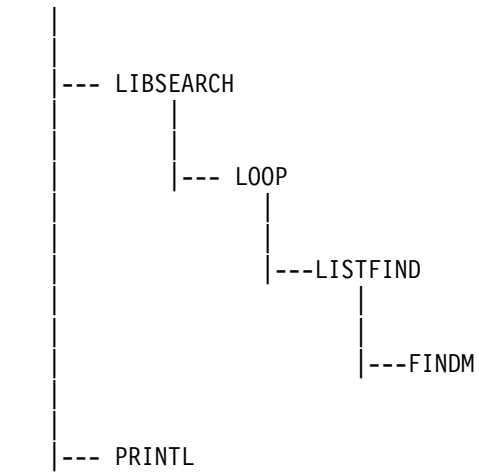

|---INLPRESL (RESTORE SUBLIBRARIES)
|---INLPLSSR (LIST SUBLIBRARY SPACE REQUIREMENTS)
|---INLPRES2 (RESTORE SUBLIBRARY)
|---INLPREM2 (RESTORE MEMBER)
|---INLPRDTP (READ TAPE BLOCK)
|---INLPTI (TAPE INPUT)
|---INLPREMB (RESTORE MEMBERS)
|---INLPREM2 (RESTORE MEMBER)
|---INLPRDTP (READ TAPE BLOCK)
|---INLPTI (TAPE INPUT)
|---INLPTIC (CLOSE INPUT TAPE)
|---INLPRUAS (UNASSIGN LOGICAL TAPE UNIT)

INLPSRCH

SEARCH IN CHAIN

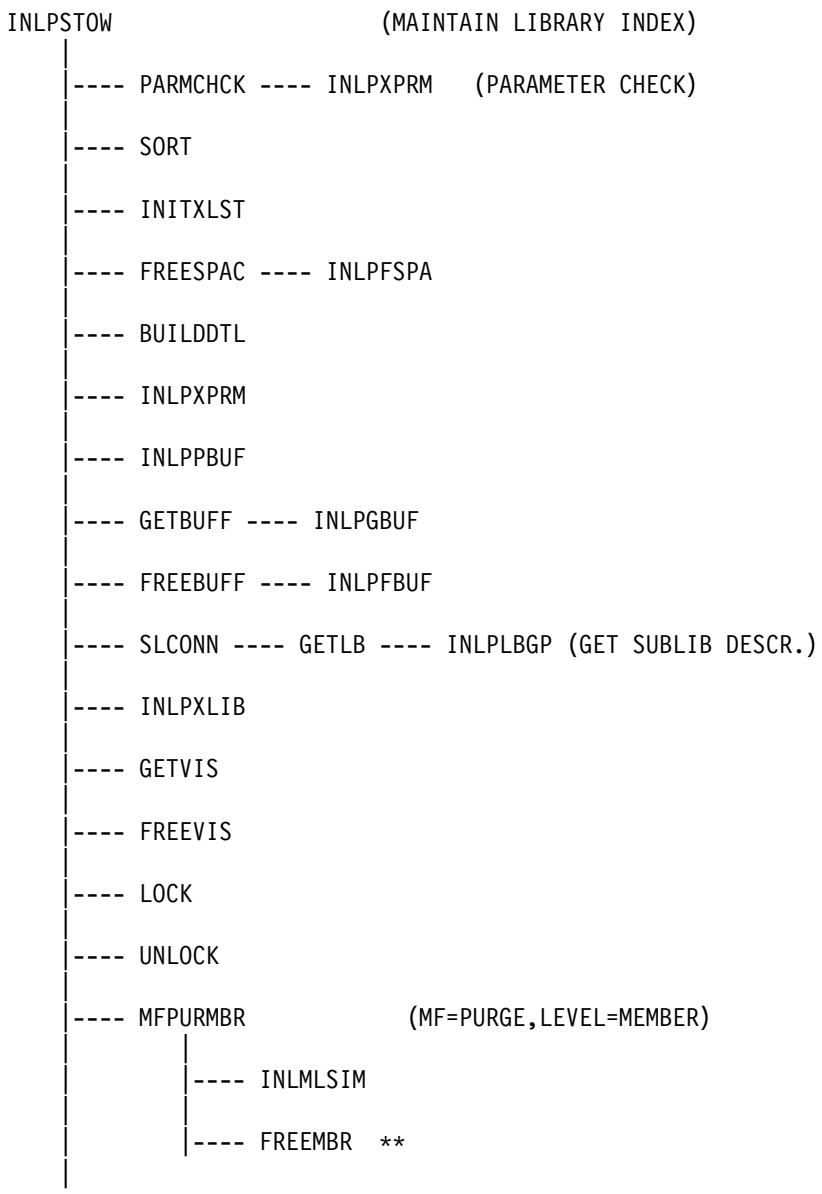


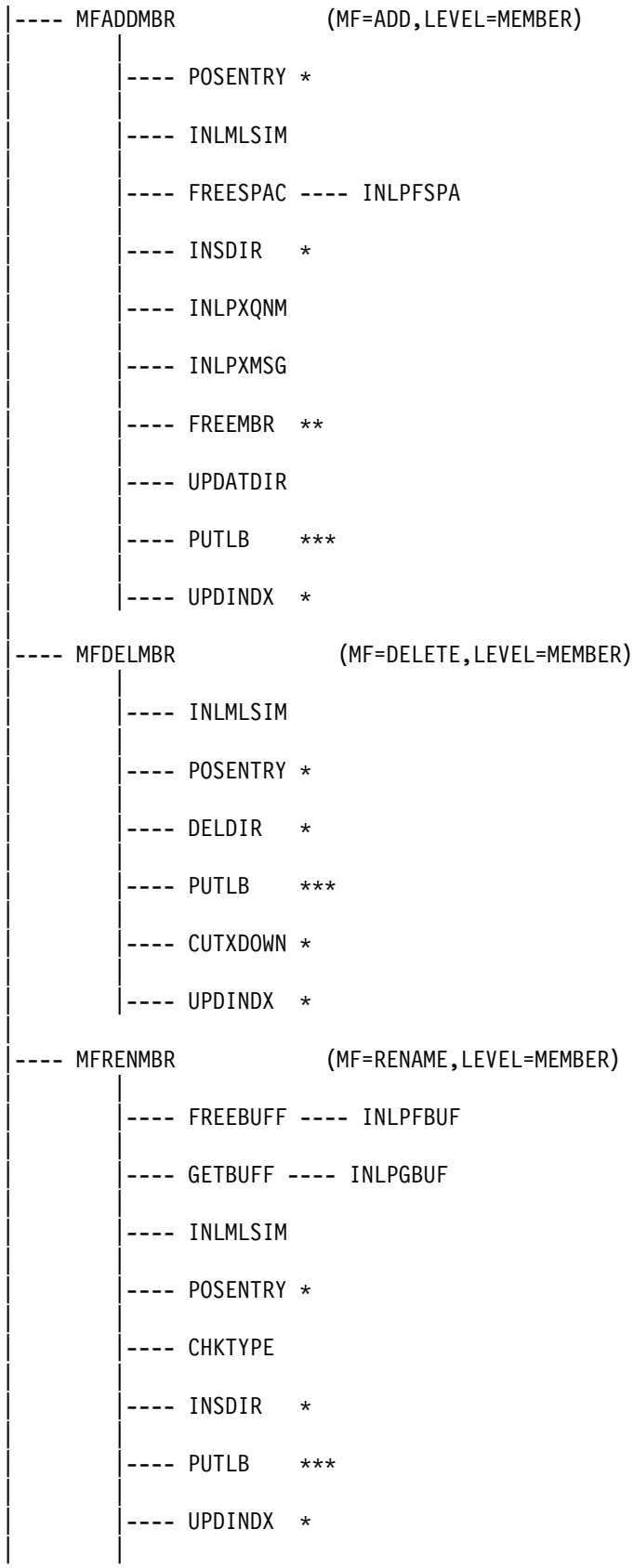
SEARCH IN LIST OF LIBRARIES OR ALL OPEN LIBRARIES

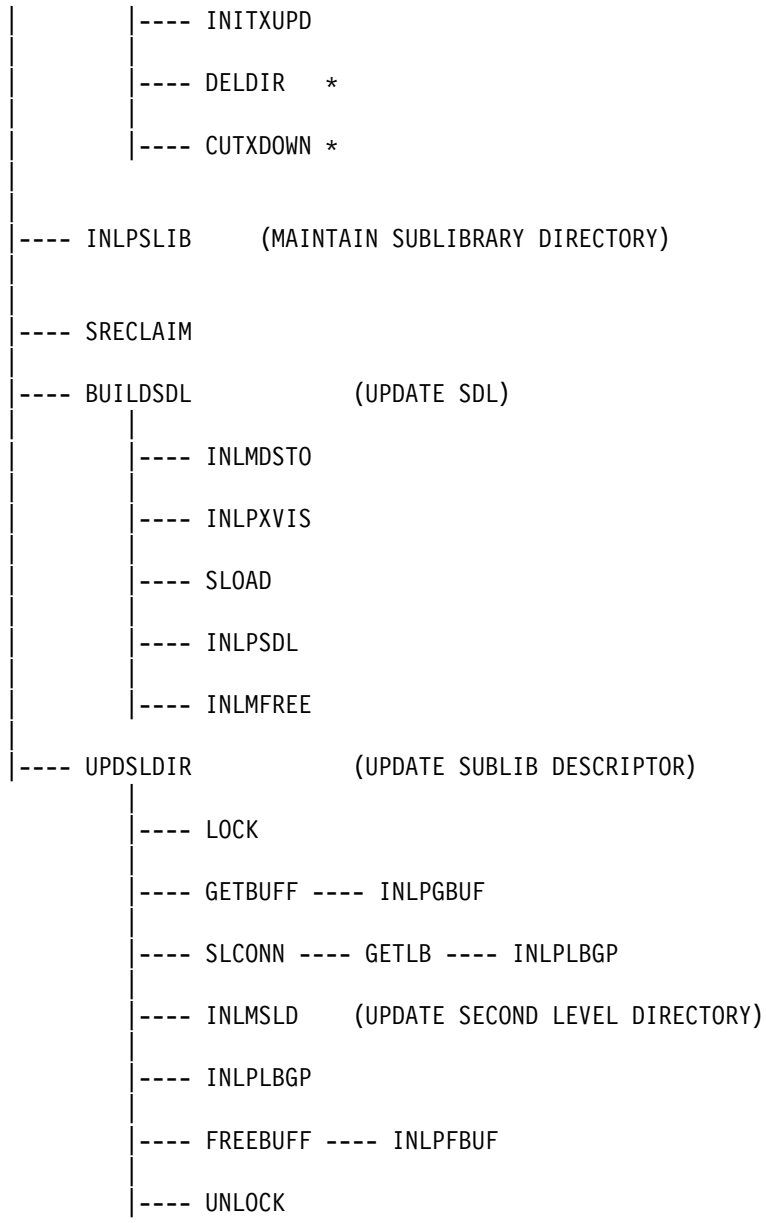


```
SEARCH IN LIST OF SUBLIBRARIES

INLPSRCH
   |
   |
   |--- PROLI
          |
          |
          |--- REDOUB
          |
          |--- LISTFIND
          |       |
          |       |
          |       |---FINDM
          |
          |--- PRINTL
```

INLPSTOW

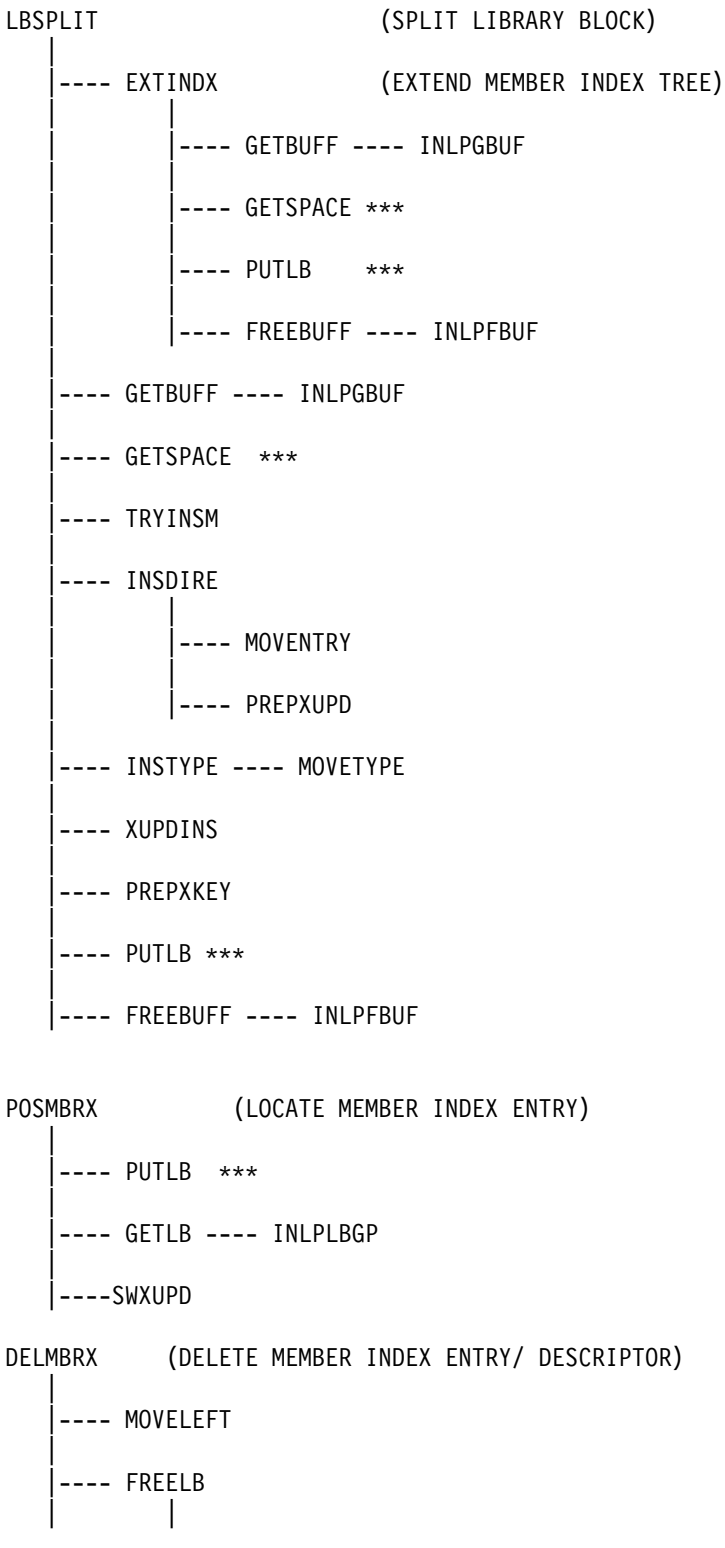

INLPSTOW (MAINTAIN LIBRARY INDEX)
PARMCHCK ---- INLPXPRM (PARAMETER CHECK)
SORT
INITXLST
FREESPAC ---- INLPFSPA
BUILDDTL
INLPXPRM
INLPPBUF
GETBUFF ---- INLPGBUF
FREEBUFF ---- INLPFBUF
SLCONN ---- GETLB ---- INLPLBGP (GET SUBLIB DESCR.)
INLPXLIB
GETVIS
FREEVIS
LOCK
UNLOCK
MFPURMBR (MF=PURGE,LEVEL=MEMBER)
INLMLSIM
FREEMBR **


MFADDMBR (MF=ADD,LEVEL=MEMBER)
POSENTRY *
INLMLSIM
FREESPAC ---- INLPFSPA
INSDIR *
INLPXQNM
INLPXMSG
FREEMBR **
UPDATDIR
PUTLB ***
UPDINDX *
MFDELMBR (MF=DELETE,LEVEL=MEMBER)
INLMLSIM
POSENTRY *
DELDIR *
PUTLB ***
CUTXDOWN *
UPDINDX *
MFRENMBR (MF=RENAME,LEVEL=MEMBER)
FREEBUFF ---- INLPFBUF
GETBUFF ---- INLPGBUF
INLMLSIM
POSENTRY *
CHKTYPE
INSDIR *
PUTLB ***
UPDINDX *

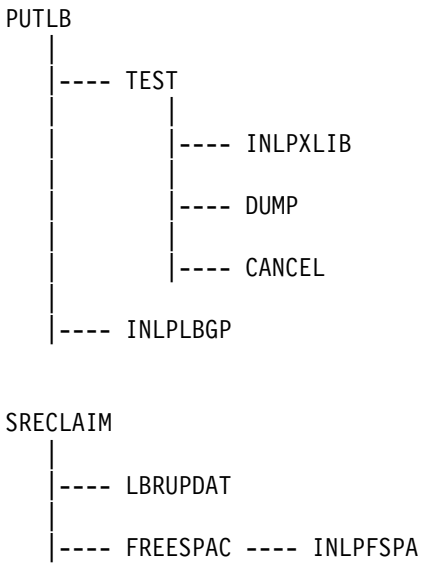

|---- INITXUPD
|---- DELDIR *
|---- CUTXDOWN *
|---- INLPSLIB (MAINTAIN SUBLIBRARY DIRECTORY)
|---- SRECLAIM
|---- BUILDSDL (UPDATE SDL)
|---- INLMDSTO
|---- INLPXVIS
|---- SLOAD
|---- INLPSDL
|---- INLMFREE
|---- UPDSLDIR (UPDATE SUBLIB DESCRIPTOR)
|---- LOCK
|---- GETBUFF ---- INLPGBUF
|---- SLCONN ---- GETLB ---- INLPLBGP
|---- INLMSLD (UPDATE SECOND LEVEL DIRECTORY)
|---- INLPLBGP
|---- FREEBUFF ---- INLPFBUF
|---- UNLOCK

```
  *):

INSDIR             (INSERT MEMBER DESCRIPTOR)
   |
   |---- INSDIRE
   |         |
   |         |---- MOVENTRY
   |         |
   |         |---- PREPXUPD
   |
   |---- LBSPLIT  **
   |
   |---- INSTYPE ---- MOVENTRY


POSENTRY           (LOCATE MEMBER DESCRIPTOR)
   |
   |---- GETLB ---- INLPLBGP
   |
   |---- PUTLB   ***
   |
   |---- SWXUPD


DELDIR             (DELETE MEMBER)
   |
   |---- FREEMBR  **
   |
   |---- DELTYPE ---- MOVELEFT
   |
   |---- DELMBRX  **


UPDINDX            (UPDATE MEMBER INDEX)
   |
   |---- GETLB ---- INLPLBGP
   |
   |---- PREPXKEY
   |
   |---- LEVSRCH --- GETLB ---- INLPLBGP
   |
   |---- INITXUPD
   |
   |---- EXECXUPD  (CHANGE MEMBER INDEX ENTRY)
   |         |
   |         |---- POSMBRX **
   |         |
   |         |---- UPDMBRX
   |         |
   |         |---- PREPXUPD
   |         |
   |         |---- INSXTYPE
   |         |
   |         |---- DELMBRX **
   |         |
```

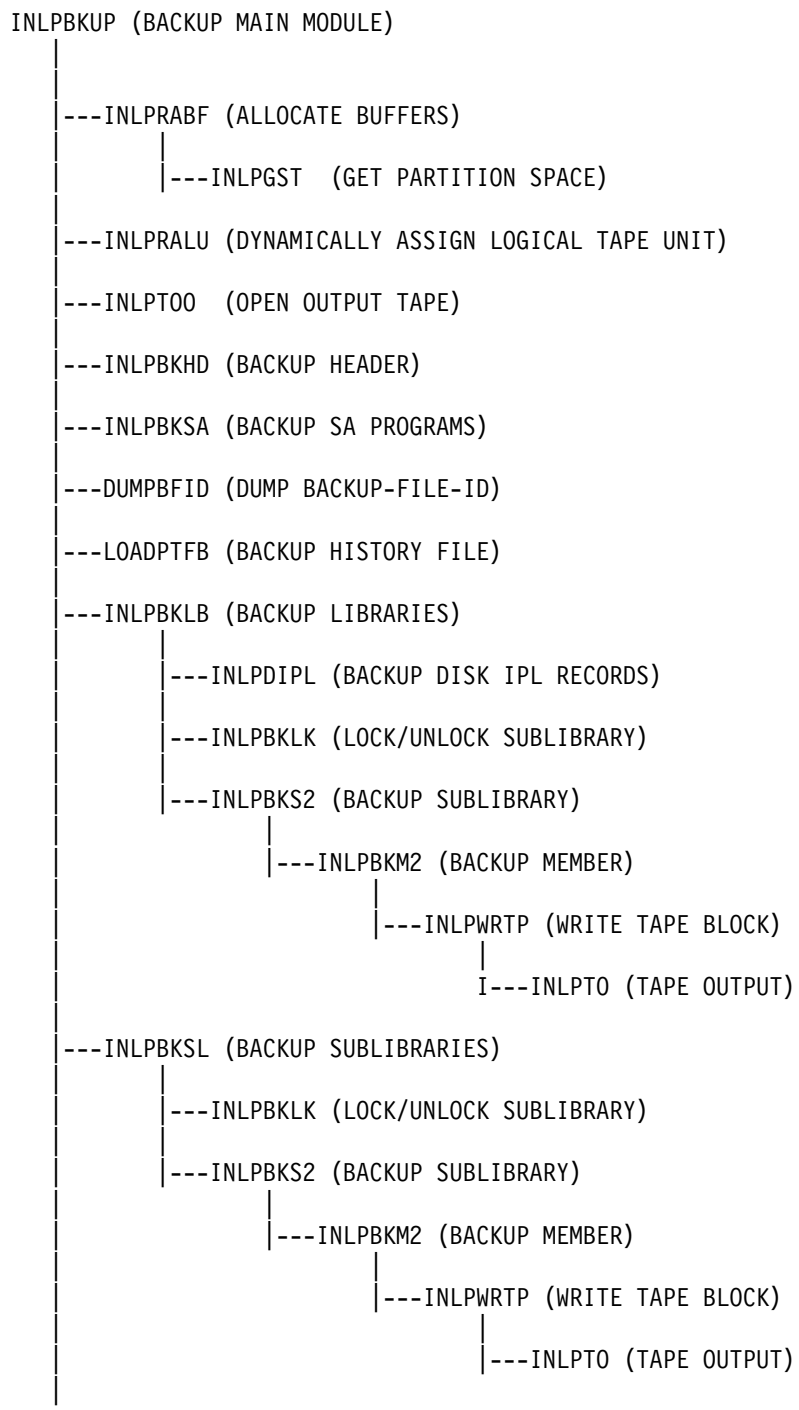
```
   |        |---- LBSPLIT **
   |        |
   |        |---- MOVENTRY
   |
   |---- EXECXINS           (INSERT MEMBER INDEX ENTRY)
   |        |
   |        |---- POSMBRX  **
   |        |
   |        |---- INSXTYPE
   |        |
   |        |---- INSMBRX
   |        |
   |        |---- PREPXUPD
   |        |
   |        |---- LBSPLIT  **
   |        |
   |        |---- MOVENTRY
   |
   |---- EXECXDEL           (DELETE MEMBER INDEX ENTRY)
            |
            |---- POSMBRX  **
            |
            |---- DELTYPE
            |
            |---- DELMBRX  **

CUTXDOWN                   (REDUCE MEMBER INDEX TREE)
   |
   |---- GETBUFF ---- INLPGBUF
   |
   |---- PUTLB  ***
   |
   |---- GETLB ---- INLPLBGP
   |
   |---- FREEBUFF ---- INLPFBUF

**):

FREEMBR            (FREE MEMBER SPACE)
   |
   |---- GETBUFF ---- INLPGBUF
   |
   |---- INLPLBGP
   |
   |---- FREEBUFF ---- INLPFBUF
```

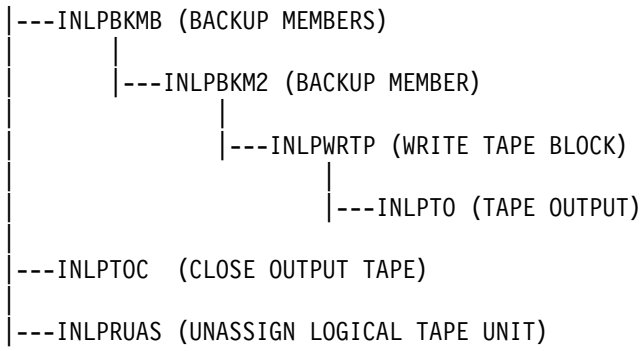

LBSPLIT (SPLIT LIBRARY BLOCK)
EXTINDX (EXTEND MEMBER INDEX TREE)
GETBUFF ---- INLPGBUF
GETSPACE ***
PUTLB ***
FREEBUFF ---- INLPFBUF
GETBUFF ---- INLPGBUF
GETSPACE ***
TRYINSM
INSDIRE
MOVENTRY
PREPXUPD
INSTYPE ---- MOVETYPE
XUPDINS
PREPXKEY
PUTLB ***
FREEBUFF ---- INLPFBUF
POSMBRX (LOCATE MEMBER INDEX ENTRY)
PUTLB ***
GETLB ---- INLPLBGP
SWXUPD
DELMBRX (DELETE MEMBER INDEX ENTRY/ DESCRIPTOR)
MOVELEFT
FREELB

```
    |         |---- GETBUFF ---- INLPGBUF
    |         |
    |         |---- GETLB ---- INLPLBGP
    |         |
    |         |---- PUTLB  ***
    |         |
    |         |---- RECLAIM ---- INLPLBGP
    |         |
    |         |---- PREPXKEY
    |         |
    |         |---- FREEBUFF ---- INLPFBUF
    |
    |---- PREPXKEY
    |
    |---- LBMERGE  ***

***):

LBMERGE         (MERGE TWO LIBRARY BLOCKS)
    |
    |---- GETBUFF ---- INLPGBUF
    |
    |---- PREPXKEY
    |
    |---- PUTLB  ***
    |
    |---- RECLAIM ---- INLPLBGP
    |
    |---- FREEBUFF ---- INLPFBUF


GETSPACE
    |
    |---- INLPGSPA
    |
    |---- FREEBUFF ---- INLPFBUF
    |
    |---- LIBFULL
    |         |
    |         |---- SRECLAIM  ****
    |         |
    |         |---- FREESPAC ---- INLPFSPA
    |
    |---- GETBUFF ---- INLPGBUF


****):
```


PUTLB
TEST
INLPXLIB
DUMP
CANCEL
INLPLBGP
SRECLAIM
LBRUPDAT
FREESPAC
INLPFSPA

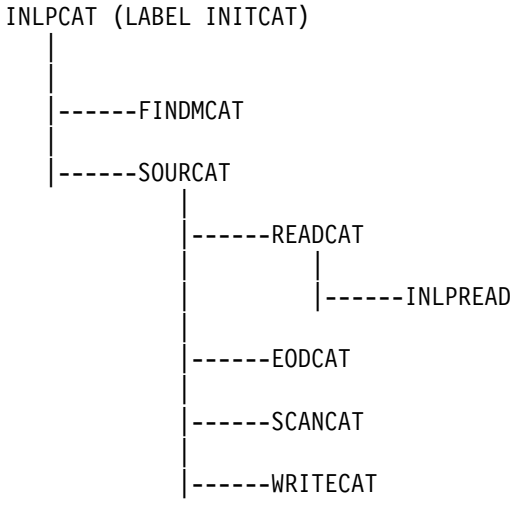
INLPBKUP


INLPBKUP (BACKUP MAIN MODULE)
INLPRABF (ALLOCATE BUFFERS)
INLPGST (GET PARTITION SPACE)
INLPRALU (DYNAMICALLY ASSIGN LOGICAL TAPE UNIT)
INLPTOO (OPEN OUTPUT TAPE)
INLPBKHD (BACKUP HEADER)
INLPBKSA (BACKUP SA PROGRAMS)
DUMPBFID (DUMP BACKUP-FILE-ID)
LOADPTFB (BACKUP HISTORY FILE)
INLPBKLB (BACKUP LIBRARIES)
INLPDIPL (BACKUP DISK IPL RECORDS)
INLPBKLK (LOCK/UNLOCK SUBLIBRARY)
INLPBKS2 (BACKUP SUBLIBRARY)
INLPBKM2 (BACKUP MEMBER)
INLPWRTP (WRITE TAPE BLOCK)
INLPTO (TAPE OUTPUT)
INLPBKSL (BACKUP SUBLIBRARIES)
INLPBKLK (LOCK/UNLOCK SUBLIBRARY)
INLPBKS2 (BACKUP SUBLIBRARY)
INLPBKM2 (BACKUP MEMBER)
INLPWRTP (WRITE TAPE BLOCK)
INLPTO (TAPE OUTPUT)

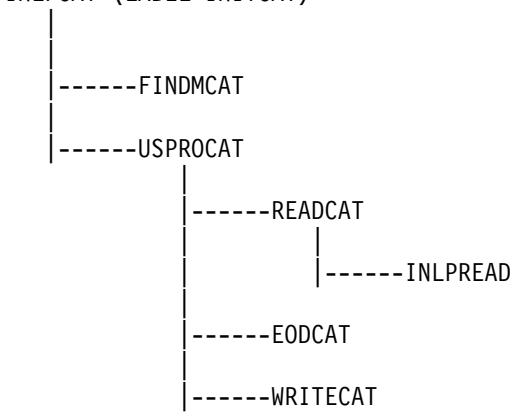

|---INLPBKMB (BACKUP MEMBERS)
|---INLPBKM2 (BACKUP MEMBER)
|---INLPWRTP (WRITE TAPE BLOCK)
|---INLPTO (TAPE OUTPUT)
|---INLPTOC (CLOSE OUTPUT TAPE)
|---INLPRUAS (UNASSIGN LOGICAL TAPE UNIT)

INLPCAT

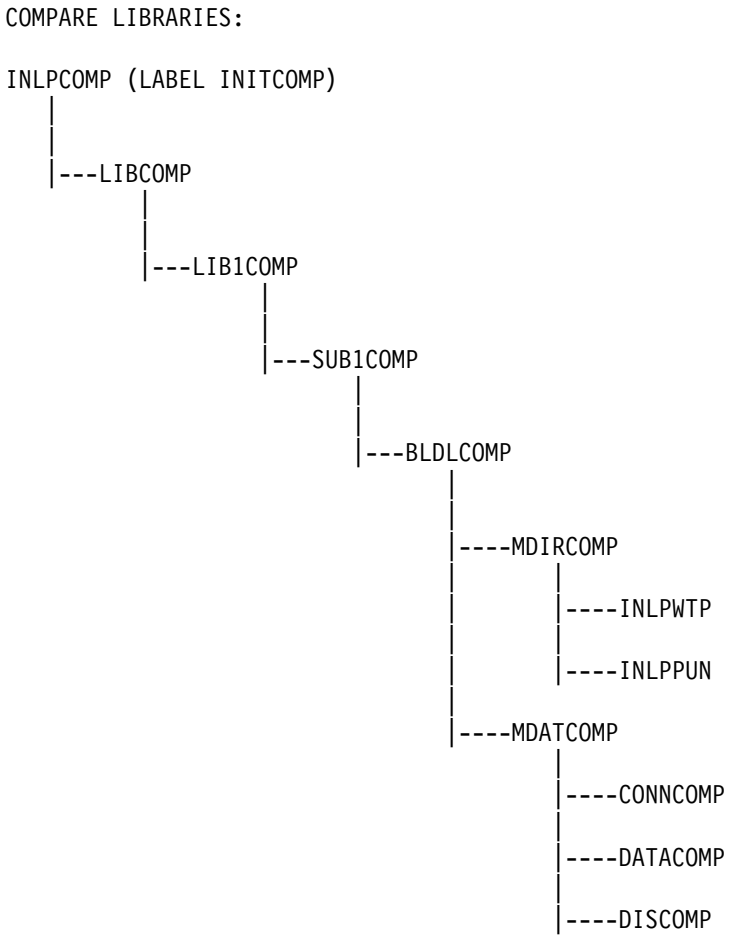
CATALOG MEMBERS OF A SOURCE-TYPE:



CATALOG MEMBERS OF TYPE PROC, OBJ OR USER-TYPE

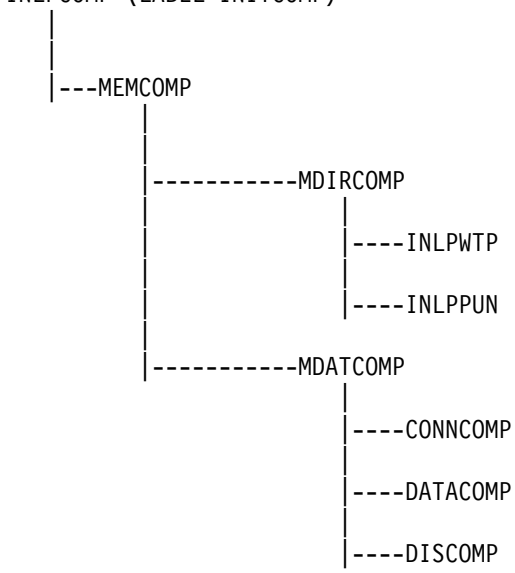


INLPCOMP

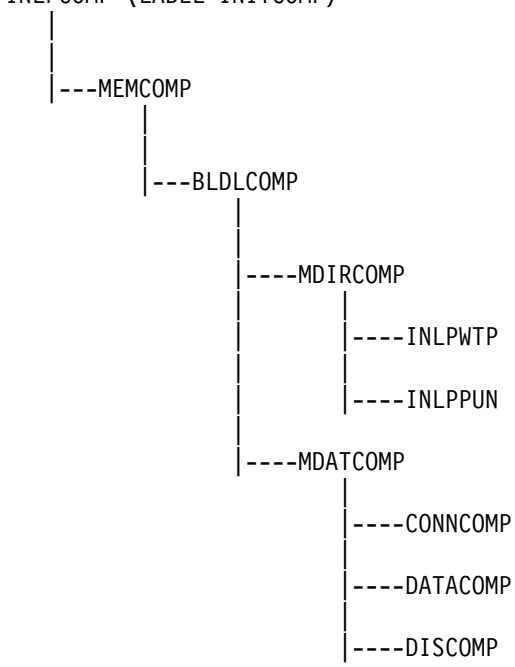

COMPARE LIBRARIES:
INLPCOMP (LABEL INITCOMP)
LIBCOMP
LIB1COMP
SUB1COMP
BLDLCOMP
MDIRCOMP
INLPWTP
INLPPUN
MDATCOMP
CONNCOMP
DATACOMP
DISCOMP

```
COMPARE SUBLIBRARIES:

INLPCOMP (LABEL INITCOMP)
   |
   |
   |---SUBCOMP
          |
          |
          |---SUB1COMP
                 |
                 |
                 |---BLDLCOMP
                        |
                        |
                        |----MDIRCOMP
                        |       |
                        |       |----INLPWTP
                        |       |
                        |       |----INLPPUN
                        |
                        |----MDATCOMP
                                |
                                |----CONNCOMP
                                |
                                |----DATACOMP
                                |
                                |----DISCOMP
```

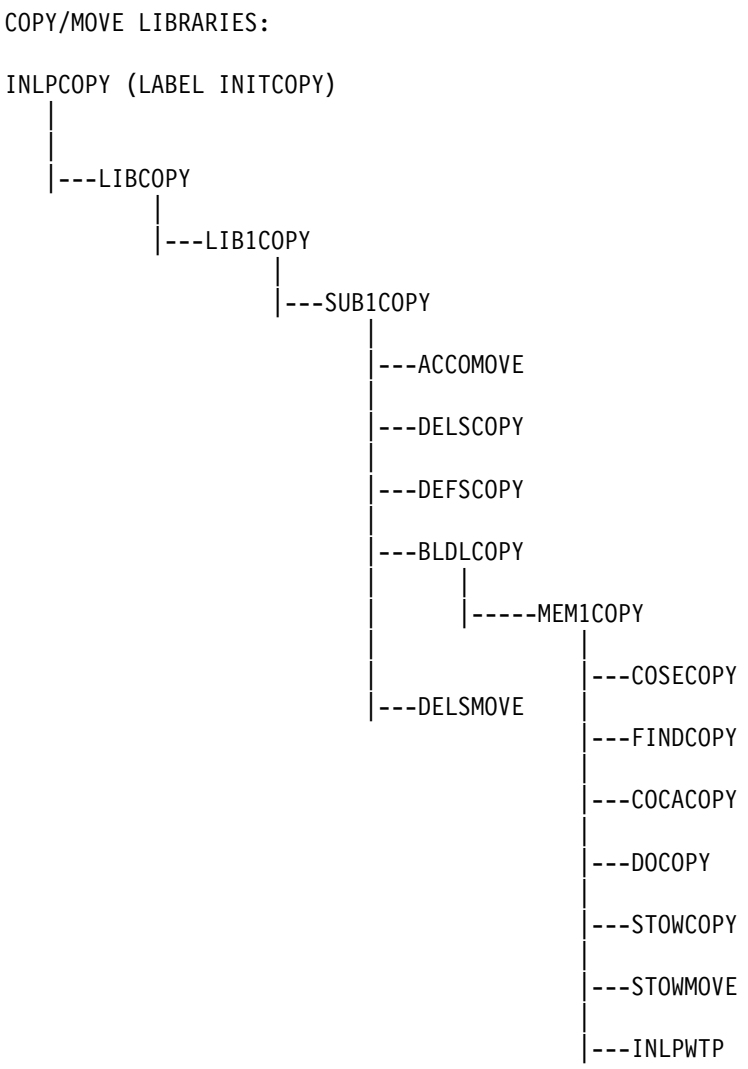
COMPARE MEMBERS (NONGENERIC MEMBER.TYPE SPECIFICATION):



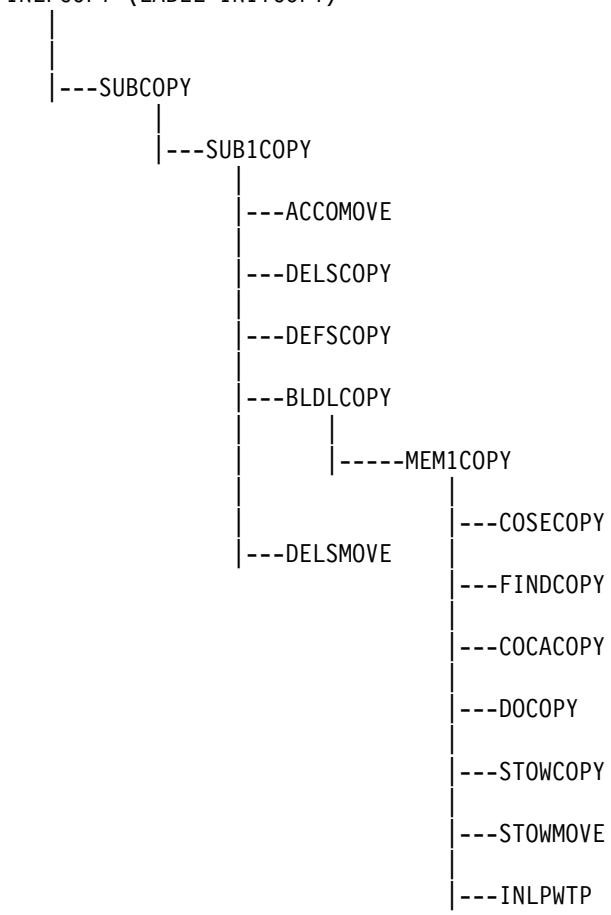
COMPARE MEMBERS (GENERIC MEMBER.TYPE SPECIFICATION):



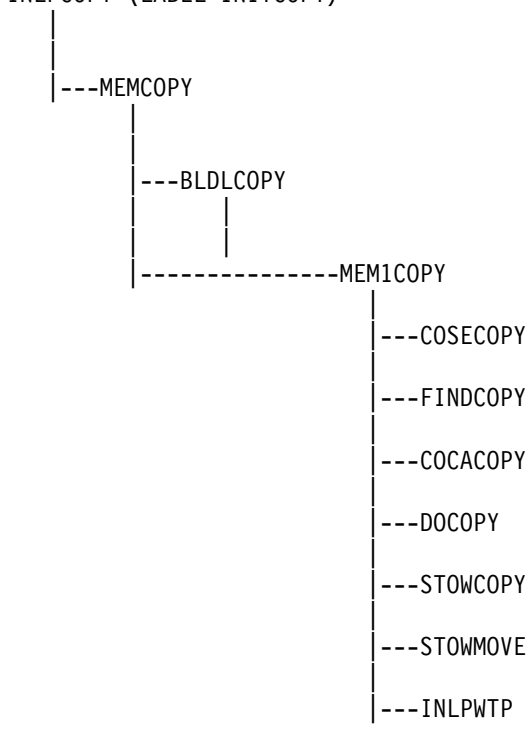
INLPCOPY


COPY/MOVE LIBRARIES:
INLPCOPY (LABEL INITCOPY)
LIBCOPY
LIB1COPY
SUB1COPY
ACCOMOVE
DELSCOPY
DEFSCOPY
BLDLCOPY
MEM1COPY
COSECOPY
DELSMOVE
FINDCOPY
COCACOPY
DOCOPY
STOWCOPY
STOWMOVE
INLPWTP

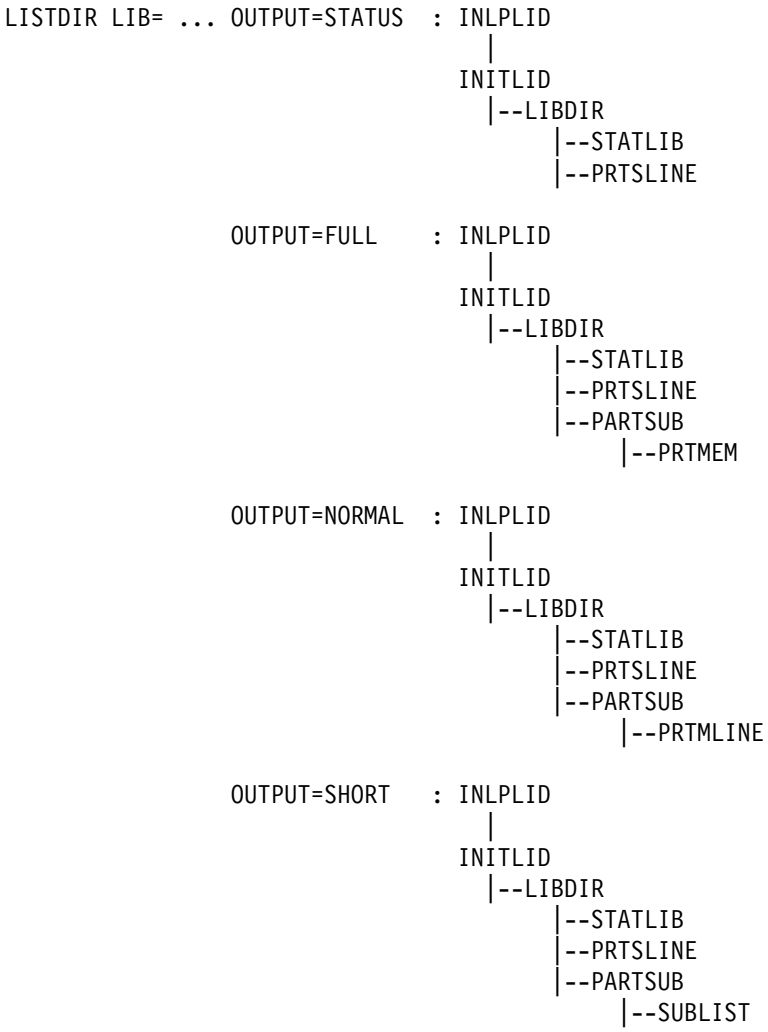
COPY/MOVE SUBLIBRARIES:



COPY/MOVE MEMBERS:

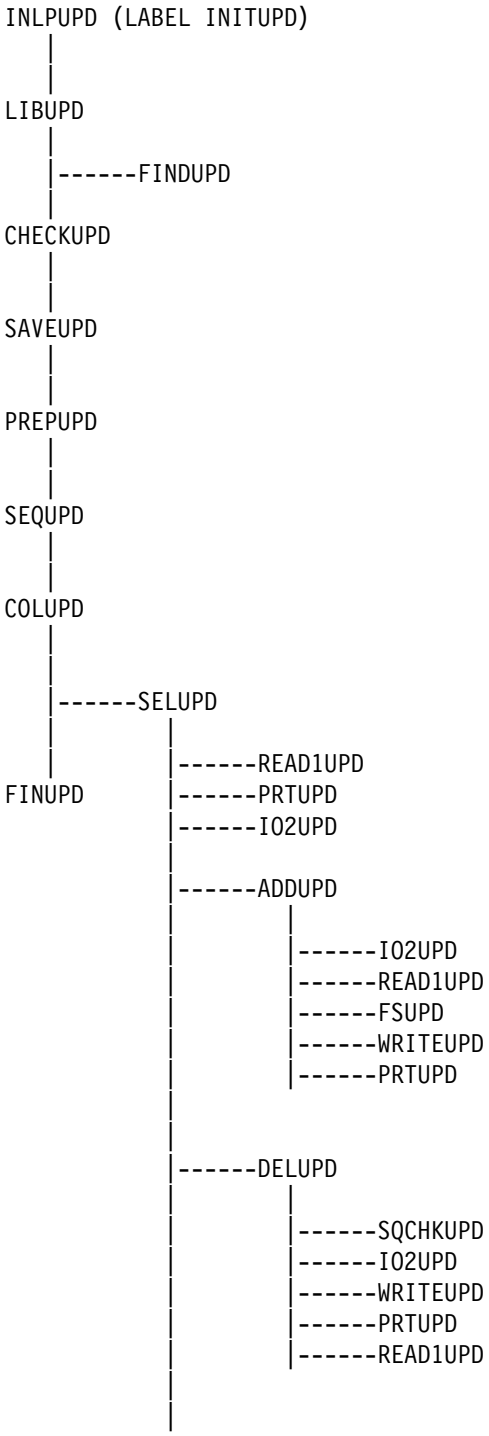


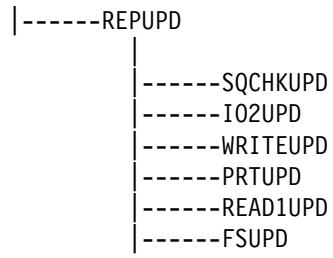
INLPLID


LISTDIR LIB= ... OUTPUT=STATUS : INLPLID
INITLID
--LIBDIR
--STATLIB
--PRTSLINE
OUTPUT=FULL : INLPLID
INITLID
--LIBDIR
--STATLIB
--PRTSLINE
--PARTSUB
--PRTMEM
OUTPUT=NORMAL : INLPLID
INITLID
--LIBDIR
--STATLIB
--PRTSLINE
--PARTSUB
--PRTMLINE
OUTPUT=SHORT : INLPLID
INITLID
--LIBDIR
--STATLIB
--PRTSLINE
--PARTSUB
--SUBLIST

```
LISTDIR SUB= ... OUTPUT=STATUS  : INLPLID
                                    |
                                  INITLID
                                    |--SUBDIR
                                         |--STATSUB

                 OUTPUT=FULL    : INLPLID
                                    |
                                  INITLID
                                    |--SUBDIR
                                         |--STATSUB
                                         |--PARTSUB
                                              |--PRTMEM

                 OUTPUT=NORMAL  : INLPLID
                                    |
                                  INITLID
                                    |--SUBDIR
                                         |--STATSUB
                                         |--PARTSUB
                                              |--PRTMLINE

                 OUTPUT=SHORT   : INLPLID
                                    |
                                  INITLID
                                    |--SUBDIR
                                         |--STATSUB
                                         |--PARTSUB
                                              |--SUBLIST

LISTDIR ...      OUTPUT=FULL    : INLPLID
                                    |
                                  INITLID
                                    |--MEMDIR
                                         |--PARTMEM
                                              |--PRTMEM

                 OUTPUT=NORMAL  : INLPLID
                                    |
                                  INITLID
                                    |--MEMDIR
                                         |--PARTMEM
                                              |--PRTMLINE

                 OUTPUT=SHORT   : INLPLID
                                    |
                                  INITLID
                                    |--MEMDIR
                                         |--PARTMEM
                                              |--SUBLIST
```

INLPUPD


INLPUPD (LABEL INITUPD)
LIBUPD
FINDUPD
CHECKUPD
SAVEUPD
PREPUPD
SEQUPD
COLUPD
SELUPD
READ1UPD
PRTUPD
IO2UPD
ADDUPD
IO2UPD
READ1UPD
FSUPD
WRITEUPD
PRTUPD
DELUPD
SQCHKUPD
IO2UPD
WRITEUPD
PRTUPD
READ1UPD
FINUPD


REPUPD
SQCHKUPD
IO2UPD
WRITEUPD
PRTUPD
READ1UPD
FSUPD

Cross-Reference List

Librarian Module-Module Interrelations

Sorted by Calling Modules

```
IJBCTUPD calls     INLPGDIR INLPXMSG INLPXPRM INLPXQNM INLPXSYS
                   INLPXVIS
IJBLBBRN calls     INLPXMSG INLPXPRM INLPXSYS
IJBLBHLS calls     INLPXPRM INLPXSYS INLPXTRC INLPXVIS
IJBLBULT calls     INLPRESN INLPXMSG INLPXPRM INLPXQNM INLPXSYS
INLPACC  calls     INLPEXIT INLPGST  INLPGSTI INLPMSG  INLPQNAM
INLPBBUF calls     INLPXPRM
INLPBKLB calls     INLPBKHD INLPBKLK INLPBKSA INLPBKS2 INLPDIPL
                   INLPMSG  INLPQNAM INLPRESN INLPTOL  INLPTO1
                   INLPWRTP
INLPBKM2 calls     INLPMSG  INLPQNAM INLPWRTP
INLPBKSA calls     INLPMSG  INLPQNAM INLPTOCH INLPTOL  INLPTO1
                   INLPWRTP
INLPBKSL calls     INLPBKHD INLPBKLK INLPBKS2 INLPMSG  INLPQNAM
                   INLPRESN INLPTOL  INLPTO1  INLPWRTP
INLPBKS2 calls     INLPBKM2 INLPMSG  INLPQNAM INLPWRTP
INLPBKUP calls     INLPBKLB INLPBKSL INLPEXIT INLPGST  INLPGSTH
                   INLPMSG  INLPRABF INLPRALU INLPRUAS INLPTOC
                   INLPTOL  INLPTOO  INLPTO1  INLPWRTP
INLPBLDL calls     INLPFIND INLPGDIR INLPOPEN INLPRESN INLPXPRM
                   INLPXTRC
INLPBSPA calls     INLPFBUF INLPGBUF INLPLBGP
INLPCAT  calls     INLPEXIT INLPGST  INLPGSTI INLPMSG  INLPQNAM
                   INLPREAD
INLPCHAN calls     INLPEXIT INLPGST  INLPGSTI INLPMSG  INLPQNAM
INLPCOMM calls     INLPWTP  INLPXMSG INLPXRCD INLPXSYS
INLPCOMP calls     INLPEXIT INLPGST  INLPGSTI INLPMSG  INLPPUN
                   INLPQNAM INLPWTP
INLPCONL calls     INLPBBUF INLPBSPA INLPFBUF INLPGBUF INLPLBGP
                   INLPXLIB INLPXPRM
INLPCONN calls     INLPEXIT INLPGST  INLPGSTI INLPMSG  INLPQNAM
INLPCONS calls     INLPFBUF INLPGBUF INLPGSPA INLPLBGP INLPXPRM
INLPCOPY calls     INLPEXIT INLPGST  INLPGSTI INLPMSG  INLPQNAM
                   INLPWTP
INLPDEF  calls     INLPEXIT INLPGST  INLPGSTI INLPMSG  INLPQNAM
                   INLPRIPL
INLPDEL  calls     INLPEXIT INLPGST  INLPGSTI INLPMSG  INLPQNAM
INLPDIAG calls     INLPWOR  INLPWTO  INLPWTP
INLPEXIT calls     INLPEND  INLPMSG  INLPREAD INLPSYNA
INLPFBUF calls     INLPLBGP INLPXTRC
INLPFIND calls     INLPMCON INLPOPEN INLPXPRM INLPXTRC
INLPGBUF calls     INLPLBGP INLPXPRM INLPXTRC
INLPGDIR calls     INLPFBUF INLPGBUF INLPLBGP INLPXPRM INLPXQNM
                   INLPXSLB INLPXTRC
```

```
INLPGSPA calls     INLPBSPA INLPFBUF INLPGBUF INLPLBGP INLPPBUF
                   INLPXLIB INLPXPRM INLPXTRC
INLPLBGP calls     INLPXPRM INLPXTRC
INLPLCON calls     INLPCONL INLPXPRM INLPXSYS INLPXTRC INLPXVIS
INLPLDIS calls     INLPPBUF INLPPUTR INLPXPRM INLPXSYS INLPXTRC
INLPLEV3 calls     INLPACC  INLPCAT  INLPCHAN INLPCOMP INLPCONN
                   INLPCOPY INLPDEF  INLPDEL  INLPGSTH INLPLID
                   INLPLIPU INLPMIGR INLPREL  INLPREN  INLPTD
                   INLPUPD
INLPLID  calls     INLPEXIT INLPGST  INLPGSTI INLPMSG  INLPQNAM
                   INLPRESN INLPWTO  INLPWTP
INLPLIPU calls     INLPEXIT INLPGST  INLPGSTI INLPMSG  INLPPUN
                   INLPQNAM INLPWTO  INLPWTP
INLPLLSR calls     INLPOUT  INLPRDTP INLPWTO  INLPWTP
INLPLSIM calls     INLPXMSG INLPXQNM INLPXSYS
INLPMAIN calls     INLPMSG  INLPPUN  INLPSYNA
INLPMCON calls     INLPBBUF INLPCONL INLPCONS INLPFBUF INLPGBUF
                   INLPLBGP INLPXPRM INLPXQNM INLPXSLB INLPXSYS
                   INLPXTRC INLPXVIS
INLPMDIS calls     INLPPUTR INLPXPRM INLPXSYS INLPXTRC
INLPMIBK calls     INLPWTP
INLPMIGR calls     INLPCAT  INLPCOPY INLPDEL  INLPEXIT INLPGSTH
                   INLPLID  INLPLIPU INLPMSG  INLPQNAM INLPREN
INLPMIRE calls     INLPWTP
INLPMSG  calls     INLPDIAG
INLPNOTE calls     INLPXPRM INLPXTRC
INLPOPEN calls     INLPXPRM INLPXSYS INLPXVIS
INLPPIDT calls     INLPXPRM
INLPPOIN calls     INLPBBUF INLPOPEN INLPXPRM INLPXTRC INLPXVIS
INLPPUTR calls     INLPFBUF INLPFSPA INLPGBUF INLPGSPA INLPLBGP
                   INLPXMSG INLPXPRM INLPXQNM INLPXTRC INLPXVIS
INLPRABF calls     INLPGST  INLPGSTI INLPMSG
INLPRALU calls     INLPMSG
INLPRCLM calls     INLPFBUF INLPFSPA INLPGBUF INLPLBGP INLPXPRM
INLPRDTP calls     INLPMSG  INLPTI   INLPTIL  INLPTIR  INLPTI1
INLPREAD calls     INLPMSG  INLPWTP
INLPREL  calls     INLPEXIT INLPGST  INLPGSTI INLPMSG  INLPQNAM
INLPRELB calls     INLPLLBS INLPLLSR INLPMSG  INLPQNAM INLPRDTP
                   INLPRES2 INLPRIPL INLPRNAC INLPSNAC
INLPREMB calls     INLPMSG  INLPMSTW INLPQNAM INLPRDTP INLPREM2
                   INLPRNAC INLPSNAC
INLPREM2 calls     INLPMSG  INLPQNAM INLPRDTP
INLPREN  calls     INLPEXIT INLPGST  INLPGSTI INLPMSG  INLPQNAM
INLPRESL calls     INLPLSSR INLPMSG  INLPQNAM INLPRDTP INLPRES2
                   INLPRNAC
INLPRESN calls     INLPXPRM
INLPREST calls     INLPEXIT INLPGSTH INLPINFR INLPMSG  INLPOUT
                   INLPQNAM INLPRABF INLPRAB1 INLPRALU INLPRCAN
                   INLPRDTP INLPRDT1 INLPRDT2 INLPRDT3 INLPRELB
                   INLPREMB INLPRESL INLPROLD INLPRSHF INLPRUAS
                   INLPTIC  INLPTIL  INLPTIO
```

```
INLXREST calls     INLPGSTH INLPMSG  INLPQNAM INLPRABF INLPRAB1
                   INLPRDTP INLPRDT1 INLPRES2 INLPRIPL INLPTIC
                   INLPTIL  INLPTIO
INLPRES2 calls     INLPMSG  INLPMSTW INLPQNAM INLPRDTP INLPREM2
                   INLPRNAC INLPSNAC
INLPRIPL calls     INLPMSG
INLPROLD calls     INLPDBLO INLPEXIT INLPGST  INLPGSTH INLPMSG
                   INLPQNAM INLPRAB2 INLPRALU INLPRTI  INLPRTIC
                   INLPRTIL INLPRTIO INLPRTIR INLPRTI1 INLPRUAS
INLPRTI  calls     INLPGST  INLPMSG
INLPSCON calls     INLPCONL INLPCONS INLPXPRM INLPXSYS INLPXTRC
                   INLPXVIS
INLPSDIS calls     INLPPUTR INLPXPRM INLPXSYS INLPXTRC
INLPSLD  calls     INLPFBUF INLPGBUF INLPLBGP INLPXQNM INLPXMSG
                   INLPXSYS
INLPSLIB calls     INLPFBUF INLPFSPA INLPGBUF INLPLBGP INLPRESN
                   INLPXLIB INLPXPRM INLPXVIS
INLPSTOW calls     INLPFBUF INLPFSPA INLPGBUF INLPGSPA INLPLBGP
                   INLPPBUF INLPRESN INLPSLIB INLPUSDL INLPXLIB
                   INLPXMSG INLPXPRM INLPXQNM INLPXTRC INLPXVIS
INLPSYNA calls     INLPEND  INLPEXIT INLPMSG  INLPREAD INLPSYPA
                   INLPWTP
INLPSYNX calls     INLPEXIT INLPMSG  INLPREAD INLPWTP
INLPSYPA calls     INLPEND  INLPEXIT INLPMSG
INLPTD   calls     INLPEXIT INLPFBUF INLPGBUF INLPLBGP INLPRESN
                   INLPTDLH INLPWTO  INLPWTP  INLPXMSG INLPXQNM
                   INLPXVIS INLPTDX
INLPTDLH calls     INLPFBUF INLPGBUF INLPLBGP INLPRESN INLPWTO
                   INLPWTP
INLPTDX  calls     INLPEXIT
                   INLPFBUF INLPLBGP INLPRESN INLPTDLH INLPWTO
                   INLPXMSG INLPXQNM INLPXVIS
                   INLPWTP
INLPTI   calls     INLPMSG
INLPUPD  calls     INLPEXIT INLPGST  INLPGSTI INLPMSG  INLPQNAM
                   INLPREAD INLPWTP
INLPUSDL calls     INLPXVIS
INLPWOR  calls     INLPWTO
INLPWRTP calls     INLPTO
```

Sorted by Called Modules

```
INLPACC  is called by     INLPLEV3
INLPBBUF is called by     INLPCONL INLPMCON INLPPOIN
INLPBKHD is called by     INLPBKLB INLPBKSL
INLPBKLB is called by     INLPBKUP
INLPBKLK is called by     INLPBKLB INLPBKSL
INLPBKM2 is called by     INLPBKS2
INLPBKSA is called by     INLPBKLB
INLPBKSL is called by     INLPBKUP
INLPBKS2 is called by     INLPBKLB INLPBKSL
INLPBSPA is called by     INLPCONL INLPGSPA
INLPCAT  is called by     INLPLEV3 INLPMIGR
INLPCHAN is called by     INLPLEV3
INLPCOMP is called by     INLPLEV3
INLPCONL is called by     INLPLCON INLPMCON INLPSCON
INLPCONN is called by     INLPLEV3
INLPCONS is called by     INLPMCON INLPSCON
INLPCOPY is called by     INLPLEV3 INLPMIGR
INLPDBLO is called by     INLPROLD
INLPDEF  is called by     INLPLEV3
INLPDEL  is called by     INLPLEV3 INLPMIGR
INLPDIAG is called by     INLPMSG
INLPDIPL is called by     INLPBKLB
INLPEND  is called by     INLPEXIT INLPSYNA INLPSYPA
INLPEXIT is called by     INLPACC  INLPBKUP INLPCAT  INLPCHAN
                          INLPCOMP INLPCONN INLPCOPY INLPDEF
                          INLPDEL  INLPLID  INLPLIPU INLPMIGR
                          INLPREL  INLPREN  INLPREST INLPROLD
                          INLPSYNA INLPSYNX INLPSYPA INLPTD
                          INLPTDX  INLPUPD
INLPFBUF is called by     INLPBSPA INLPCONL INLPCONS INLPGDIR
                          INLPGSPA INLPMCON INLPPUTR INLPRCLM
                          INLPSLD  INLPSLIB INLPSTOW INLPTD
                          INLPTDLH INLPTDX
INLPFIND is called by     INLPBLDL
INLPFSPA is called by     INLPPUTR INLPRCLM INLPSLIB INLPSTOW
INLPGBUF is called by     INLPBSPA INLPCONL INLPCONS INLPGDIR
                          INLPGSPA INLPMCON INLPPUTR INLPRCLM
                          INLPSLD  INLPSLIB INLPSTOW INLPTD
                          INLPTDLH
INLPGDIR is called by     IJBCTUPD INLPBLDL
INLPGSPA is called by     INLPCONS INLPPUTR INLPSTOW
INLPGST  is called by     INLPACC  INLPBKUP INLPCAT  INLPCHAN
                          INLPCOMP INLPCONN INLPCOPY INLPDEF
                          INLPDEL  INLPLID  INLPLIPU INLPRABF
                          INLPREL  INLPREN  INLPREST INLPROLD
                          INLPRTI  INLPUPD
INLPGSTH is called by     INLPBKUP INLPLEV3 INLPMIGR INLPREST
                          INLPROLD INLXREST
INLPGSTI is called by     INLPACC  INLPCAT  INLPCHAN INLPCOMP
                          INLPCONN INLPCOPY INLPDEF  INLPDEL
                          INLPLID  INLPLIPU INLPRABF INLPREL
                          INLPREN  INLPUPD
```

```
INLPLBGP is called by     INLPBSPA INLPCONL INLPCONS INLPFBUF
                          INLPGBUF INLPGDIR INLPGSPA INLPMCON
                          INLPPUTR INLPRCLM INLPSLD  INLPSLIB
                          INLPSTOW INLPTD   INLPTDLH INLPTDX
INLPLID  is called by     INLPLEV3 INLPMIGR
INLPLIPU is called by     INLPLEV3 INLPMIGR
INLPLLBS is called by     INLPRELB
INLPLLSR is called by     INLPRELB
INLPLSSR is called by     INLPRESL
INLPMCON is called by     INLPFIND
INLPMIGR is called by     INLPLEV3
INLPMSG  is called by     INLPACC  INLPBKLB INLPBKM2 INLPBKSA
                          INLPBKSL INLPBKS2 INLPBKUP INLPCAT
                          INLPCHAN INLPCOMP INLPCONN INLPCOPY
                          INLPDEF  INLPDEL  INLPEXIT INLPLID
                          INLPLIPU INLPMAIN INLPMIGR INLPRABF
                          INLPRALU INLPRDTP INLPREAD INLPREL
                          INLPRELB INLPREMB INLPREM2 INLPREN
                          INLPRESL INLPREST INLPRES2 INLPRIPL
                          INLPROLD INLPRTI  INLPSYNA INLPSYNX
                          INLPSYPA INLPTI   INLPUPD  INLXREST
INLPMSTW is called by     INLPREMB INLPRES2
INLPOPEN is called by     INLPBLDL INLPFIND INLPPOIN
INLPOUT  is called by     INLPLLSR INLPREST
INLPPBUF is called by     INLPGSPA INLPLDIS INLPSTOW
INLPPUN  is called by     INLPCOMP INLPLIPU INLPMAIN
INLPPUTR is called by     INLPLDIS INLPMDIS INLPSDIS
INLPQNAM is called by     INLPACC  INLPBKLB INLPBKM2 INLPBKSA
                          INLPBKSL INLPBKS2 INLPCAT  INLPCHAN
                          INLPCOMP INLPCONN INLPCOPY INLPDEF
                          INLPDEL  INLPLID  INLPLIPU INLPMIGR
                          INLPREL  INLPRELB INLPREMB INLPREM2
                          INLPREN  INLPRESL INLPREST INLPRES2
                          INLPROLD INLPUPD  INLXREST
INLPRABF is called by     INLPBKUP INLPREST INLXREST
INLPRAB1 is called by     INLPREST INLXREST
INLPRAB2 is called by     INLPROLD
INLPRALU is called by     INLPBKUP INLPREST INLPROLD
INLPRDTP is called by     INLPLLSR INLPRELB INLPREMB INLPREM2
                          INLPRESL INLPREST INLPRES2 INLXREST
INLPRDT1 is called by     INLPREST INLXREST
INLPRDT2 is called by     INLPREST
INLPREAD is called by     INLPCAT  INLPEXIT INLPSYNA INLPSYNX
                          INLPUPD
INLPREL  is called by     INLPLEV3
INLPRELB is called by     INLPREST
INLPREMB is called by     INLPREST
INLPREM2 is called by     INLPREMB INLPRES2
INLPREN  is called by     INLPLEV3 INLPMIGR
INLPRESL is called by     INLPREST
INLPRESN is called by     IJBLBULT INLPBKLB INLPBKSL INLPBLDL
                          INLPLID  INLPSLIB INLPSTOW INLPTD
                          INLPTDLH INLPTDX
INLPRES2 is called by     INLPRELB INLPRESL INLXREST
```

```
INLPRIPL is called by     INLPDEF  INLPRELB INLXREST
INLPRNAC is called by     INLPRELB INLPREMB INLPRESL INLPREST INLPRES2
INLPROLD is called by     INLPREST
INLPRTI  is called by     INLPROLD
INLPRTIC is called by     INLPROLD
INLPRTIL is called by     INLPROLD
INLPRTIO is called by     INLPROLD
INLPRTIR is called by     INLPROLD
INLPRTI1 is called by     INLPROLD
INLPRUAS is called by     INLPBKUP INLPREST INLPROLD
INLPSLIB is called by     INLPSTOW
INLPSNAC is called by     INLPRELB INLPREMB INLPRES2
INLPSYNA is called by     INLPEXIT INLPMAIN
INLPSYPA is called by     INLPSYNA
INLPTD   is called by     INLPLEV3
INLPTDLH is called by     INLPTD   INLPTDX
INLPTDX  is called by     INLPTD
INLPTI   is called by     INLPRDTP
INLPTIC  is called by     INLPREST INLXREST
INLPTIL  is called by     INLPRDTP INLPREST INLXREST
INLPTIO  is called by     INLPREST INLXREST
INLPTIR  is called by     INLPRDTP
INLPTI1  is called by     INLPRDTP
INLPTO   is called by     INLPWRTP
INLPTOC  is called by     INLPBKUP
INLPTOCH is called by     INLPBKSA
INLPTOL  is called by     INLPBKLB INLPBKSA INLPBKSL INLPBKUP
INLPTOO  is called by     INLPBKUP
INLPTO1  is called by     INLPBKLB INLPBKSA INLPBKSL INLPBKUP
INLPUPD  is called by     INLPLEV3
INLPUSDL is called by     INLPSTOW
INLPWOR  is called by     INLPDIAG
INLPWRTP is called by     INLPBKLB INLPBKM2 INLPBKSA INLPBKSL
                          INLPBKS2 INLPBKUP
INLPWTO  is called by     INLPDIAG INLPLID  INLPLIPU INLPLLSR
                          INLPTD   INLPTDLH INLPTDX  INLPWOR
INLPWTP  is called by     INLPCOMM INLPCOMP INLPCOPY INLPDIAG
                          INLPLID  INLPLIPU INLPLLSR INLPMIBK
                          INLPMIRE INLPREAD INLPSYNA INLPSYNX
                          INLPTD   INLPTDLH INLPTDX  INLPUPD
INLPXLIB is called by     INLPCONL INLPGSPA INLPSLIB INLPSTOW
INLPXMSG is called by     IJBCTUPD IJBLBBRN IJBLBULT INLPCOMM
                          INLPLSIM INLPPUTR INLPSLD  INLPSTOW
                          INLPTD   INLPTDX
INLPXPRM is called by     IJBCTUPD IJBLBBRN IJBLBHLS IJBLBULT
                          INLPBBUF INLPBLDL INLPCONL INLPCONS
                          INLPFIND INLPGBUF INLPGDIR INLPGSPA
                          INLPLBGP INLPLCON INLPLDIS INLPMCON
                          INLPMDIS INLPNOTE INLPOPEN INLPPIDT
                          INLPPOIN INLPPUTR INLPRCLM INLPRESN
                          INLPSCON INLPSDISINLPSLIBINLPSTOW
```

```
INLPXQNM is called by     IJBCTUPD IJBLBULT INLPGDIR INLPLSIM
                          INLPMCON INLPPUTR INLPSLD  INLPSTOW
                          INLPTD   INLPTDX
INLPXRCD is called by     INLPCOMM
INLPXSLB is called by     INLPGDIR INLPMCON
INLPXSYS is called by     IJBCTUPD IJBLBBRN IJBLBHLS IJBLBULT
                          INLPCOMM INLPLCON INLPLDIS INLPLSIM
                          INLPMCON INLPMDIS INLPOPEN INLPSCON
                          INLPSDIS INLPSLD
INLPXTRC is called by     IJBLBHLS INLPBLDL INLPFBUF INLPFIND
                          INLPGBUF INLPGDIR INLPGSPA INLPLBGP
                          INLPLCON INLPLDIS INLPMCON INLPMDIS
                          INLPNOTE INLPPOIN INLPPUTR INLPSCON
                          INLPSDIS INLPSTOW
INLPXVIS is called by     IJBCTUPD IJBLBHLS INLPLCON INLPMCON
                          INLPOPEN INLPPOIN INLPPUTR INLPSCON
                          INLPSLIB INLPSTOW INLPTD   INLPUSDL
```

Librarian Module-Macro Interrelations

Sorted by Invoking Modules

```
IJBCTUPD invokes   INLMLAMB INLMLCON INLMLDIS INLMLRPL INLMLSIM
                   INLMPIDT INLMRCLM INLMSCON INLMSLD  INLCDDTE
                   INLCEDTE INLCLDTE INLCLOT  INLCLPT  INLCLSIM
                   INLCSDTE INLCSLDE INLCSLXE
IJBLBBRN invokes   INLMPIDT LBRBLDCB INLCDDTE INLCEDTE INLCEXTD
                   INLCLAMB INLCLBCF INLCLDES INLCLDTE INLCLPT
                   INLCLRPL INLCPIDT
IJBLBHLS invokes   LBRACCCB LBRBLDCB LBRBLDRN LBRCTUPD LBRLCTAC
                   LBRLCTCB INLCDDTE INLCEDTE INLCLAMB INLCLPT
                   INLCLRPL
IJBLBIPL invokes   INLMGDIR INLMLAMB INLMLCON INLMLDIS INLMLRPL
                   INLMRCLM INLMSCON INLMSLD  INLCDDTE INLCDENT
                   INLCEDTE INLCEXTD INLCIDTE INLCLACB INLCLDES
                   INLCLDTE INLCLOT  INLCLPT  INLCRESN INLCSACB
                   INLCSDTE INLCSLXE
IJBLBLLS invokes   LBRLCTCB INLCLDTE INLCLOT  INLCLPT  INLCLRPL
                   INLCSDTE
IJBLBULT invokes   LBRUPDAT INLCLAMB INLCLDTE INLCLOT  INLCLPT
                   INLCLRPL INLCRESN INLCSDTE
INLPACC  invokes   INLMLRPL INLMPROC LBRACCCB LBRACCES INLCCMDP
                   INLCCOMR
INLPAREA invokes   INLMPROC INLCCMDP INLCCOMR INLCSCAN
INLPBBUF invokes   INLMPROC INLCARPA INLCBHDR INLCBUCB INLCLACB
                   INLCMACB INLCSACB
INLPBKLB invokes   INLMBLDL INLMGDIR INLMLAMB INLMLCON INLMLDIS
                   INLMLRPL INLMPROC LBRACCCB LBRACCES INLCBDB
                   INLCBRCR INLCCMDP INLCCOMR INLCDDTE INLCDENT
                   INLCDIO  INLCEDTE INLCLDES INLCRESN INLCSLXE
                   INLCTDCL
INLPBKM2 invokes   INLMGETR INLMLAMB INLMLRPL INLMMCON INLMMDIS
                   INLMPROC LBRACCCB INLCBDB  INLCBRCR INLCCOMR
                   INLCDENT INLCDIO  INLCSLXE INLCTDCL
INLPBKSA invokes   INLMGETR INLMLAMB INLMLDIS INLMLRPL INLMMCON
                   INLMMDIS INLMPROC INLMSCON INLMSDIS LBRACCCB
                   LBRACCES INLCBDB  INLCBRCR INLCCMDP INLCCOMR
                   INLCDENT INLCDIO  INLCTDCL
INLPBKSL invokes   INLMBLDL INLMLAMB INLMLDIS INLMLRPL INLMPROC
                   LBRACCCB LBRACCES INLCBDB  INLCBRCR INLCCMDP
                   INLCCOMR INLCDENT INLCDIO  INLCLDES INLCSLXE
                   INLCTDCL
INLPBKS2 invokes   INLMGDIR INLMLAMB INLMLRPL INLMPROC INLMSCON
                   INLMSDIS LBRACCCB INLCBDB  INLCBRCR INLCCOMR
                   INLCDENT INLCDIO  INLCLDES INLCRESN INLCSLXE
                   INLCTDCL
```

```
INLPBKUP invokes   INLMLAMB INLMLRPL INLMPROC LBRACCCB INLCBDB
                   INLCBRCR INLCCMDP INLCCOMR INLCTDCL
INLPBLDL invokes   INLMFIND INLMGDIR INLMLCON INLMLDIS INLMLRPL
                   INLMLSIM INLMMDIS INLCDDTE INLCDENT INLCEDTE
                   INLCLACB INLCLAMB INLCLBTB INLCLDTE INLCLPT
                   INLCLRPL INLCLSIM INLCMACB INLCRESN INLCSACB
                   INLCSLXE
INLPBSPA invokes   INLMPROC INLCARPA INLCBHDR INLCBUCB INLCEXTD
                   INLCLACB INLCLBCF INLCLDES INLCMACB INLCSACB
INLPCAT  invokes   INLMFIND INLMLAMB INLMLDIS INLMLRPL INLMMCON
                   INLMMDIS INLMPROC INLMPUTR INLMSCON INLMSTOW
                   LBRACCCB LBRACCES INLCCMDP INLCCOMR INLCDENT
                   INLCPARB
INLPCHAN invokes   INLMLAMB INLMLCON INLMLDIS INLMLRPL INLMPROC
                   INLMSTOW LBRACCCB LBRACCES INLCCMDP INLCCOMR
                   INLCDENT
INLPCMPR invokes   INLMPROC
INLPCOMM invokes   INLMLAMB INLCLSIM
INLPCOMP invokes   INLMBLDL INLMGETR INLMLAMB INLMLDIS INLMLRPL
                   INLMLSIM INLMMCON INLMMDIS INLMPROC INLMSDIS
                   LBRACCCB LBRACCES INLCCMDP INLCCOMR INLCDENT
                   INLCLSIM
INLPCONL invokes   INLCARPA INLCBHDR INLCLACB INLCLAMB INLCLBCF
                   INLCLDES INLCLDTE INLCLRPL
INLPCONN invokes   INLMLAMB INLMLRPL INLMPROC LBRACCCB LBRACCES
                   INLCCMDP INLCCOMR
INLPCONS invokes   INLCARPA INLCBHDR INLCEDTE INLCLACB INLCLAMB
                   INLCLBCF INLCLDES INLCLDTE INLCLRPL INLCSACB
                   INLCSDTE INLCSLXE
INLPCOPY invokes   INLMBLDL INLMFIND INLMGETR INLMLAMB INLMLCON
                   INLMLDIS INLMLRPL INLMLSIM INLMMCON INLMMDIS
                   INLMPROC INLMPUTR INLMSCON INLMSTOW LBRACCCB
                   LBRACCES LBRUPDAT INLCCMDP INLCCOMR INLCDENT
                   INLCLSIM
INLPDBLO invokes   INLMPROC INLCCMDP INLCCOMR
INLPDEF  invokes   INLMLAMB INLMLCON INLMLDIS INLMLRPL INLMPROC
                   INLMSCON INLMSTOW LBRACCCB LBRACCES INLCCMDP
                   INLCCOMR INLCDENT
INLPDEL  invokes   INLMBLDL INLMLAMB INLMLCON INLMLDIS INLMLRPL
                   INLMPROC INLMSCON INLMSTOW LBRACCCB LBRACCES
                   INLCCMDP INLCCOMR INLCDDTE INLCDENT INLCEDTE
INLPDIAG invokes   INLMLAMB
INLPDOIO invokes   INLMPROC INLCBUCB
INLPEXIT invokes   INLMLAMB INLMPROC INLCCMDP INLCCOMR INLCSCAN
INLPFBUF invokes   INLML1TD INLMPROC INLCARPA INLCBHDR INLCBUCB
                   INLCEDTE INLCLACB INLCLPT  INLCMACB INLCSACB
INLPFIND invokes   INLMMCON INLCDENT INLCLACB INLCLAMB INLCLBTB
                   INLCLPT  INLCLRPL INLCMACB INLCSACB
INLPGBUF invokes   INLML1TD INLMPROC INLCARPA INLCBHDR INLCBUCB
                   INLCLACB INLCLPT  INLCMACB INLCSACB
INLPGDIR invokes   INLMLSIM LBRACCCB INLCARPA INLCBHDR INLCDENT
                   INLCDESC INLCLACB INLCLAMB INLCLBCF INLCLDES
                   INLCLDTE INLCLPT  INLCLRPL INLCLSIM INLCSACB
                   INLCSDTE INLCSLXE INLCTYPE
```

```
INLPGSPA invokes   INLMPROC LBRACCES INLCARPA INLCBHDR INLCBUCB
                   INLCEXTD INLCLACB INLCLBCF INLCLPT  INLCMACB
                   INLCSACB
INLPGST  invokes   INLMPROC
INLPLBGP invokes   INLML1TD INLMPROC INLCARPA INLCBHDR INLCBUCB
                   INLCLACB INLCLPT  INLCMACB INLCSACB
INLPLCON invokes   INLCDENT INLCEDTE INLCLACB INLCLAMB INLCLBTB
                   INLCLDTE INLCLPT  INLCLRPL INLCMACB INLCSACB
INLPLDIS invokes   INLCARPA INLCBUCB INLCDENT INLCLACB INLCLAMB
                   INLCLBTB INLCLPT  INLCLRPL INLCMACB INLCSACB
INLPLEV3 invokes   INLMLAMB INLMPROC INLCCMDP INLCCOMR
INLPLID  invokes   INLMBLDL INLMGDIR INLMLAMB INLMLCON INLMLDIS
                   INLMLRPL INLMPROC LBRACCCB LBRACCES INLCCMDP
                   INLCCOMR INLCDDTE INLCDENT INLCEDTE INLCLDES
                   INLCRESN INLCSLXE
INLPLIPU invokes   INLMBLDL INLMFIND INLMGETR INLMLAMB INLMLDIS
                   INLMLRPL INLMMCON INLMMDIS INLMPROC LBRACCCB
                   LBRACCES INLCCMDP INLCCOMR INLCDENT
INLPLLSR invokes   INLMPROC INLCBRCR INLCCMDP INLCCOMR INLCDDTE
                   INLCDEVT INLCTDCL
INLPLSIM invokes   INLCDENT INLCEDTE INLCLAMB INLCLDTE INLCLOT
                   INLCLRPL INLCLSIM INLCSDTE INLCSLXE
INLPMAIN invokes   INLMPROC LBRACCCB LBRACCES INLCCMDP INLCCOMR
                   INLCPARB
INLPMCON invokes   INLMLSIM INLCARPA INLCBHDR INLCDENT INLCDESC
                   INLCEDTE INLCLACB INLCLAMB INLCLBCF INLCLDES
                   INLCLDTE INLCLPT  INLCLRPL INLCLSIM INLCMACB
                   INLCMBRX INLCSACB INLCSDTE INLCTYPE
INLPMDIS invokes   INLCARPA INLCBHDR INLCBUCB INLCDENT INLCLACB
                   INLCLAMB INLCLPT  INLCLRPL INLCMACB INLCSACB
INLPMIBK invokes   INLMPROC
INLPMICO invokes   INLMPROC
INLPMICS invokes   INLMPROC
INLPMIDS invokes   INLMPROC
INLPMIGR invokes   INLMLAMB INLMLRPL INLMPROC LBRACCCB LBRACCES
                   INLCCMDP INLCCOMR
INLPMIMA invokes   INLMPROC
INLPMIPS invokes   INLMPROC
INLPMIRE invokes   INLMPROC
INLPMIRS invokes   INLMPROC
INLPMISS invokes   INLMPROC
INLPMSG  invokes   INLMLAMB INLMPROC INLCCOMR
INLPNOTE invokes   INLMPROC INLCBHDR INLCBUCB INLCDENT INLCLBTB
                   INLCLPT  INLCMACB INLCNTPT
INLPOPEN invokes   INLMLAMB INLMLRPL LBRACCCB LBRACCES INLCDENT
                   INLCLACB INLCLBTB INLCMACB INLCSACB
INLPOUT  invokes   INLMLAMB INLMPROC INLCCOMR INLCPARB
INLPPIDT invokes   INLMLAMB INLMLRPL INLCDDTE INLCEDTE INLCIDTE
                   INLCLDTE INLCLPT  INLCPIDT
INLPPOIN invokes   INLMPROC LBRACCDS INLCARPA INLCBHDR INLCBUCB
                   INLCDENT INLCLACB INLCLBTB INLCLPT  INLCMACB
                   INLCNTPT INLCSACB
INLPPUN  invokes   INLMLAMB INLMPROC INLCCOMR INLCPARB
INLPPUTR invokes   INLMPROC INLCARPA INLCBHDR INLCBUCB INLCDENT
                   INLCLBCF INLCLBTB INLCLPT  INLCMACB INLCNTPT
```

```
INLPQNAM invokes   INLMPROC
INLPRABF invokes   INLMLAMB INLMPROC INLCBDB  INLCBRCR INLCCMDP
                   INLCCOMR INLCDENT INLCDIO  INLCLDES INLCSLXE
                   INLCTDCL
INLPRALU invokes   INLMLAMB INLMPROC INLCBRCR INLCCMDP INLCCOMR
INLPRCLM invokes   INLMLCON INLMLDIS INLMLRPL INLMSCON INLMSDIS
                   INLMSLD  INLCARPA INLCBHDR INLCLACB INLCLAMB
                   INLCLBCF INLCLDTE INLCLRPL INLCSDTE INLCSLXE
INLPRDTP invokes   INLMLAMB INLMPROC INLCBDB  INLCBRCR INLCCMDP
                   INLCCOMR INLCDIO  INLCTDCL
INLPREAD invokes   INLMLAMB INLMPROC INLCCMDP INLCCOMR INLCPARB
INLPREL  invokes   INLMBLDL INLMLAMB INLMLRPL INLMLSIM INLMPROC
                   INLMRCLM LBRACCCB LBRACCES INLCCMDP INLCCOMR
                   INLCDENT INLCLSIM INLCSLXE
INLPRELB invokes   INLMLAMB INLMLCON INLMLDIS INLMLRPL INLMPROC
                   LBRACCCB LBRACCES INLCBDB  INLCBRCR INLCCMDP
                   INLCCOMR INLCDDTE INLCDIO  INLCEDTE INLCLDES
                   INLCSLXE INLCTDCL
INLPREMB invokes   INLMLAMB INLMLDIS INLMLRPL INLMLSIM INLMPROC
                   INLMSCON LBRACCCB LBRACCES INLCBDB  INLCBRCR
                   INLCCMDP INLCCOMR INLCDENT INLCDIO  INLCLDES
                   INLCLSIM INLCSLXE INLCTDCL
INLPREM2 invokes   INLMLAMB INLMLRPL INLMMCON INLMMDIS INLMPROC
                   INLMPUTR INLMSTOW LBRACCCB LBRUPDAT INLCBDB
                   INLCBRCR INLCCMDP INLCCOMR INLCDENT INLCTDCL
INLPREN  invokes   INLMBLDL INLMLAMB INLMLCON INLMLDIS INLMLRPL
                   INLMPROC INLMSCON INLMSTOW LBRACCCB LBRACCES
                   INLCCMDP INLCCOMR INLCDENT
INLPRESL invokes   INLMLAMB INLMLDIS INLMLRPL INLMPROC LBRACCCB
                   LBRACCES INLCBDB  INLCBRCR INLCCMDP INLCCOMR
                   INLCDENT INLCDIO  INLCLDES INLCSLXE INLCTDCL
INLPRESN invokes   INLCDDTE INLCEDTE INLCLAMB INLCLDTE INLCLRPL
                   INLCRESN
INLPREST invokes   INLMLAMB INLMLRPL INLMPROC LBRACCCB INLCBDB
                   INLCBRCR INLCCMDP INLCCOMR INLCDIO  INLCTDCL
INLPRES2 invokes   INLMLAMB INLMLCON INLMLDIS INLMLRPL INLMPROC
                   INLMSCON INLMSDIS INLMSTOW LBRACCCB LBRACCES
                   LBRUPDAT INLCBDB  INLCBRCR INLCCMDP INLCCOMR
                   INLCDENT INLCDIO  INLCLDES INLCSLXE INLCTDCL
INLPRIPL invokes   INLMPROC INLCDDTE INLCEDTE INLCLDTE
INLPROLD invokes   INLMLAMB INLMLDIS INLMLRPL INLMMCON INLMMDIS
                   INLMPROC INLMPUTR INLMSCON INLMSTOW LBRACCCB
                   LBRACCES LBRUPDAT INLCBRCR INLCCMDP INLCCOMR
                   INLCDENT INLCLDES INLCSLXE
INLPRTI  invokes   INLMLAMB INLMPROC INLCCOMR
INLPSCON invokes   INLCDENT INLCEDTE INLCLACB INLCLAMB INLCLBTB
                   INLCLDTE INLCLPT  INLCLRPL INLCMACB INLCSACB
                   INLCSDTE
INLPSDIS invokes   INLCARPA INLCBHDR INLCBUCB INLCLACB INLCLAMB
                   INLCLPT  INLCLRPL INLCMACB INLCSACB
INLPSDL  invokes   INLCDENT INLCEDTE INLCLPT  INLCSLDE
```

```
INLPSLD  invokes   INLCARPA INLCBHDR INLCDENT INLCDESC INLCLACB
                   INLCLAMB INLCLBCF INLCLDTE INLCLPT  INLCLRPL
                   INLCMBRX INLCSACB INLCSDTE INLCSLDE INLCSLXE
                   INLCTYPE
INLPSLIB invokes   INLMLSIM LBRUPDAT INLCARPA INLCBHDR INLCDDTE
                   INLCDENT INLCDESC INLCEDTE INLCLACB INLCLAMB
                   INLCLBCF INLCLDES INLCLDTE INLCLPT  INLCLRPL
                   INLCLSIM INLCMBRX INLCRESN INLCSACB INLCSDTE
                   INLCSLXE INLCTYPE
INLPSTOW invokes   INLMLSIM INLMSLD  LBRUPDAT INLCARPA INLCBHDR
                   INLCDDTE INLCDENT INLCDESC INLCEDTE INLCLACB
                   INLCLAMB INLCLBCF INLCLDES INLCLDTE INLCLPT
                   INLCLRPL INLCLSIM INLCMBRX INLCRESN INLCSACB
                   INLCSDTE INLCSLDE INLCSLXE INLCTYPE
INLPSYNA invokes   INLMLAMB INLMPROC INLCCMDP INLCCOMR INLCSCAN
INLPSYNX invokes   INLMPROC INLCCOMR INLCSCAN
INLPSYPA invokes   INLMPROC INLCCMDP INLCCOMR INLCSCAN
INLPTD   invokes   INLMBLDL INLMGDIR INLMLAMB INLMLCON INLMLDIS
                   INLMLRPL INLMLSIM INLMMCON INLMMDIS INLMPROC
                   INLMSCON LBRACCCB LBRACCES LBRUPDAT INLCARPA
                   INLCBHDR INLCCMDP INLCCOMR INLCDDTE INLCDENT
                   INLCDESC INLCEDTE INLCEXTD INLCLACB INLCLBCF
                   INLCLDES INLCLDTE INLCLSIM INLCMACB INLCMBRX
                   INLCRESN INLCSACB INLCSDTE INLCSLXE
INLPTDLH invokes   INLMLAMB INLMLRPL INLMPROC INLCARPA INLCBHDR
                   INLCCMDP INLCCOMR INLCDDTE INLCEDTE INLCEXTD
                   INLCLACB INLCLBCF INLCLDES INLCLDTE INLCLPT
                   INLCLSIM INLCRESN
INLPTDX  invokes   INLMPRAS INLCCMDO INLCCOMR
                   INLCDDTE INLCDENT INLCEDTE
                   INLCRESN INLDDENT INLDLINE INLMLRPL LBRACCCB
INLPTI   invokes   INLMLAMB INLMPROC INLCBDB  INLCCOMR INLCDIO
                   INLCTDCL
INLPTO   invokes   INLMPROC INLCBDB  INLCDIO  INLCTDCL
INLPUPD  invokes   INLMFIND INLMGETR INLMLAMB INLMLDIS INLMLRPL
                   INLMMCON INLMMDIS INLMPROC INLMPUTR INLMSTOW
                   LBRACCCB LBRACCES INLCCMDP INLCCOMR INLCDENT
INLPUSDL invokes   INLCARPA INLCBHDR INLCDDTE INLCDENT INLCDESC
                   INLCEDTE INLCLACB INLCLAMB INLCLBCF INLCLDES
                   INLCLDTE INLCLPT  INLCLRPL INLCMBRX INLCSACB
                   INLCSDTE INLCSLXE INLCTYPE
INLPWOR  invokes   INLMLAMB INLMPROC INLCPARB
INLPWRTP invokes   INLMPROC INLCBDB  INLCBRCR INLCTDCL
INLPWTO  invokes   INLMLAMB INLMPROC INLCPARB
INLPWTP  invokes   INLMLAMB INLMPROC INLCPARB
INLXREST invokes   INLMLAMB INLMLCON INLMLDIS INLMLRPL INLMPROC
                   LBRACCCB INLCBDB  INLCBRCR INLCCMDP INLCCOMR
                   INLCDDTE INLCDENT INLCDIO  INLCEDTE INLCLDES
                   INLCLDTE INLCLPT  INLCSDTE INLCSLXE INLCTDCL
```

Sorted by Invoked Macros

```
LBRACCCB is invoked by    IJBLBHLS INLPACC  INLPBKLB INLPBKM2
                          INLPBKSA INLPBKSL INLPBKS2 INLPBKUP
                          INLPCAT  INLPCHAN INLPCOMP INLPCONN
                          INLPCOPY INLPDEF  INLPDEL  INLPGDIR
                          INLPLID  INLPLIPU INLPMAIN INLPMIGR
                          INLPOPEN INLPREL  INLPRELB INLPREMB
                          INLPREM2 INLPREN  INLPRESL INLPREST
                          INLPRES2 INLPROLD INLPTD   INLPTDX
                          INLPUPD  INLXREST
LBRACCDS is invoked by    INLPPOIN
LBRACCES is invoked by    INLPACC  INLPBKLB INLPBKSA INLPBKSL
                          INLPCAT  INLPCHAN INLPCOMP INLPCONN
                          INLPCOPY INLPDEF  INLPDEL  INLPGSPA
                          INLPLID  INLPLIPU INLPMAIN INLPMIGR
                          INLPOPEN INLPREL  INLPRELB INLPREMB
                          INLPREN  INLPRESL INLPRES2 INLPROLD
                          INLPTD   INLPUPD
LBRBLDCB is invoked by    IJBLBBRN IJBLBHLS
LBRBLDRN is invoked by    IJBLBHLS
INLCARPA is invoked by    INLPBBUF INLPBSPA INLPCONL INLPCONS
                          INLPFBUF INLPGBUF INLPGDIR INLPGSPA
                          INLPLBGP INLPLDIS INLPMCON INLPMDIS
                          INLPPOIN INLPPUTR INLPRCLM INLPSDIS
                          INLPSLD  INLPSLIB INLPSTOW INLPTD
                          INLPTDLH INLPUSDL
INLCBDB  is invoked by    INLPBKLB INLPBKM2 INLPBKSA INLPBKSL
                          INLPBKS2 INLPBKUP INLPRABF INLPRDTP
                          INLPRELB INLPREMB INLPREM2 INLPRESL
                          INLPREST INLPRES2 INLPTI   INLPTO
                          INLPWRTP INLXREST
INLCBHDR is invoked by    INLPBBUF INLPBSPA INLPCONL INLPCONS
                          INLPFBUF INLPGBUF INLPGDIR INLPGSPA
                          INLPLBGP INLPMCON INLPMDIS INLPNOTE
                          INLPPOIN INLPPUTR INLPRCLM INLPSDIS
                          INLPSLD  INLPSLIB INLPSTOW INLPTD
                          INLPTDLH INLPUSDL
INLCBRCR is invoked by    INLPBKLB INLPBKM2 INLPBKSA INLPBKSL
                          INLPBKS2 INLPBKUP INLPLLSR INLPRABF
                          INLPRALU INLPRDTP INLPRELB INLPREMB
                          INLPREM2 INLPRESL INLPREST INLPRES2
                          INLPROLD INLPWRTP INLXREST
INLCBUCB is invoked by    INLPBBUF INLPBSPA INLPDOIO INLPFBUF
                          INLPGBUF INLPGSPA INLPLBGP INLPLDIS
                          INLPMDIS INLPNOTE INLPPOIN INLPPUTR
                          INLPSDIS
```

```
INLCCMDP is invoked by    INLPACC  INLPAREA INLPBKLB INLPBKSA
                          INLPBKSL INLPBKUP INLPCAT  INLPCHAN
                          INLPCOMP INLPCONN INLPCOPY INLPDBLO
                          INLPDEF  INLPDEL  INLPEXIT INLPLEV3
                          INLPLID  INLPLIPU INLPLLSR INLPMAIN
                          INLPMIGR INLPRABF INLPRALU INLPRDTP
                          INLPREAD INLPREL  INLPRELB INLPREMB
                          INLPREM2 INLPREN  INLPRESL INLPREST
                          INLPRES2 INLPROLD INLPSYNA INLPSYPA
                          INLPTD   INLPTDLH INLPTDX
                          INLPUPD  INLXREST
INLCCOMR is invoked by    INLPACC  INLPAREA INLPBKLB INLPBKM2
                          INLPBKSA INLPBKSL INLPBKS2 INLPBKUP
                          INLPCAT  INLPCHAN INLPCOMP INLPCONN
                          INLPCOPY INLPDBLO INLPDEF  INLPDEL
                          INLPEXIT INLPLEV3 INLPLID  INLPLIPU
                          INLPLLSR INLPMAIN INLPMIGR INLPMSG
                          INLPOUT  INLPPUN  INLPRABF INLPRALU
                          INLPRDTP INLPREAD INLPREL  INLPRELB
                          INLPREMB INLPREM2 INLPREN  INLPRESL
                          INLPREST INLPRES2 INLPROLD INLPRTI
                          INLPSYNA INLPSYNX INLPSYPA INLPTD
                          INLPTDLH INLPTX
                          INLPTI   INLPUPD  INLXREST
INLCDDTE is invoked by    IJBCTUPD IJBLBBRN IJBLBHLS IJBLBIPL
                          INLPBKLB INLPBLDL INLPDEL  INLPLID
                          INLPLLSR INLPRELB INLPRESN INLPRIPL
                          INLPSLIB INLPSTOW INLPTD   INLPTDLH
                          INLPTDX  INLPUSDL INLXREST
INLCDENT is invoked by    IJBLBIPL INLPBKLB INLPBKM2 INLPBKSA
                          INLPBKSL INLPBKS2 INLPBLDL INLPCAT
                          INLPCHAN INLPCOMP INLPCOPY INLPDEF
                          INLPDEL  INLPFIND INLPGDIR INLPLCON
                          INLPLDIS INLPLID  INLPLIPU INLPLSIM
                          INLPMCON INLPMDIS INLPNOTE INLPOPEN
                          INLPPOIN INLPPUTR INLPRABF INLPREL
                          INLPREMB INLPREM2 INLPREN  INLPRESL
                          INLPREST INLPRES2 INLPROLD INLPSCON
                          INLPSDL  INLPSLD  INLPSLIB INLPSTOW
                          INLPTD   INLPTDX
                          INLPUPD  INLPUSDL INLXREST
INLCDESC is invoked by    INLPGDIR INLPMCON INLPSLD  INLPSLIB
                          INLPSTOW INLPTD   INLPUSDL
INLCDEVT is invoked by    INLPLLSR
INLCDIO  is invoked by    INLPBKLB INLPBKM2 INLPBKSA INLPBKSL
                          INLPBKS2 INLPRABF INLPRDTP INLPRELB
                          INLPREMB INLPRESL INLPREST INLPRES2
                          INLPTI   INLPTO   INLXREST
INLCEDTE is invoked by    IJBCTUPD IJBLBBRN IJBLBHLS IJBLBIPL
                          INLPBKLB INLPBLDL INLPCONS INLPDEL
                          INLPFBUF INLPLCON INLPLID  INLPLSIM
                          INLPMCON INLPPIDT INLPRELB INLPRESN
                          INLPRIPL INLPSCON INLPSDL
                          INLPSLIB INLPSTOW INLPTD   INLPTDLH
                          INLPTDX  INLPUSDL INLXREST
```

```
INLCEXTD is invoked by    IJBLBBRN IJBLBIPL INLPBSPA INLPGSPA
                          INLPTD   INLPTDLH
INLCIDTE is invoked by    IJBLBIPL INLCPIDT
INLCLACB is invoked by    IJBLBIPL INLPBBUF INLPBLDL INLPBSPA
                          INLPCONL INLPCONS INLPFBUF INLPFIND
                          INLPGBUF INLPGDIR INLPGSPA INLPLBGP
                          INLPLCON INLPLDIS INLPMCON INLPMDIS
                          INLPOPEN INLPPOIN INLPRCLM INLPSCON
                          INLPSDIS INLPSLD  INLPSLIB INLPSTOW
                          INLPTD   INLPTDLH INLPUSDL
INLCLAMB is invoked by    IJBLBBRN IJBLBHLS IJBLBULT INLPBLDL
                          INLPCONL INLPCONS INLPFIND INLPGDIR
                          INLPLCON INLPLDIS INLPLSIM INLPMCON
                          INLPMDIS INLPRCLM INLPRESN INLPSCON
                          INLPSDIS INLPSLD  INLPSLIB INLPSTOW
                          INLPUSDL
INLCLBCF is invoked by    IJBLBBRN INLPBSPA INLPCONL INLPCONS
                          INLPGDIR INLPGSPA INLPMCON INLPPUTR
                          INLPRCLM INLPSLD  INLPSLIB INLPSTOW
                          INLPTD   INLPTDLH INLPUSDL
INLCLBTB is invoked by    INLPBLDL INLPFIND INLPLCON INLPLDIS
                          INLPNOTE INLPOPEN INLPPOIN INLPPUTR
                          INLPSCON
INLCLDES is invoked by    IJBLBBRN IJBLBIPL INLPBKLB INLPBKSL
                          INLPBKS2 INLPBSPA INLPCONL INLPCONS
                          INLPLID  INLPMCON INLPRABF INLPRELB
                          INLPREMB INLPRESL INLPRES2
                          INLPROLD INLPSLIB INLPSTOW INLPTD
                          INLPTDLH INLPUSDL INLXREST
INLCLDTE is invoked by    IJBCTUPD IJBLBBRN IJBLBIPL IJBLBLLS
                          IJBLBULT INLPBLDL INLPCONL INLPCONS
                          INLPGDIR INLPLCON INLPLSIM INLPMCON
                          INLPPIDT INLPRCLM INLPRESN
                          INLPRIPL INLPSCON INLPSLD  INLPSLIB
                          INLPSTOW INLPTD   INLPTDLH INLPUSDL
                          INLXREST
INLCLOT  is invoked by    IJBCTUPD IJBLBIPL IJBLBLLS IJBLBULT
                          INLPLSIM
INLCLPT  is invoked by    IJBCTUPD IJBLBBRN IJBLBHLS IJBLBIPL
                          IJBLBLLS IJBLBULT INLPBLDL INLPFBUF
                          INLPFIND INLPGBUF INLPGDIR INLPGSPA
                          INLPLBGP INLPLCON INLPLDIS INLPMCON
                          INLPMDIS INLPNOTE INLPPIDT INLPPOIN
                          INLPPUTR INLPSCON INLPSDIS
                          INLPSDL  INLPSLD  INLPSLIB INLPSTOW
                          INLPTDLH INLPUSDL INLXREST
```

```
INLCLRPL is invoked by    IJBLBBRN IJBLBHLS IJBLBLLS IJBLBULT
                          INLPBLDL INLPCONL INLPCONS INLPFIND
                          INLPLCON INLPLDIS INLPLSIM INLPMCON
                          INLPMDIS INLPRCLM INLPRESN INLPSCON
                          INLPSDIS INLPSLD  INLPSLIB INLPSTOW
                          INLPUSDL
INLCLSIM is invoked by    IJBCTUPD INLPBLDL INLPCOMM INLPCOMP
                          INLPCOPY INLPGDIR INLPLSIM INLPMCON
                          INLPREL  INLPREMB INLPSLIB INLPSTOW
                          INLPTD   INLPTDLH
INLCMACB is invoked by    INLPBBUF INLPBLDL INLPBSPA INLPFBUF
                          INLPFIND INLPGBUF INLPGSPA INLPLBGP
                          INLPLCON INLPLDIS INLPMCON INLPMDIS
                          INLPNOTE INLPOPEN INLPPOIN INLPPUTR
                          INLPSCON INLPSDIS INLPTD
INLCMBRX is invoked by    INLPMCON INLPSLD  INLPSLIB INLPSTOW
                          INLPTD   INLPUSDL
INLCNTPT is invoked by    INLPNOTE INLPPOIN INLPPUTR
INLCPARB is invoked by    INLPCAT  INLPMAIN INLPOUT  INLPPUN
                          INLPREAD INLPWOR  INLPWTO  INLPWTP
INLCPIDT is invoked by    IJBLBBRN INLPPIDT
INLCRESN is invoked by    IJBLBIPL IJBLBULT INLPBKLB INLPBKS2
                          INLPBLDL INLPLID  INLPRESN INLPSLIB
                          INLPSTOW INLPTD   INLPTDLH INLPTDX
INLCSACB is invoked by    IJBLBIPL INLPBBUF INLPBLDL INLPBSPA
                          INLPCONS INLPFBUF INLPFIND INLPGBUF
                          INLPGDIR INLPGSPA INLPLBGP INLPLCON
                          INLPLDIS INLPMCON INLPMDIS INLPOPEN
                          INLPPOIN INLPSCON INLPSDIS INLPSLD
                          INLPSLIB INLPSTOW INLPTD   INLPUSDL
INLCSCAN is invoked by    INLPAREA INLPEXIT INLPSYNA INLPSYNX
                          INLPSYPA
INLCSDTE is invoked by    IJBCTUPD IJBLBIPL IJBLBLLS IJBLBULT
                          INLPCONS INLPGDIR INLPLSIM INLPMCON
                          INLPRCLM INLPREST INLPSCON INLPSLD
                          INLPSLIB INLPSTOW INLPTD   INLPUSDL
INLCSLDE is invoked by    IJBCTUPD INLPSDL  INLPSLD  INLPSTOW
INLCSLXE is invoked by    IJBCTUPD IJBLBIPL INLPBKLB INLPBKM2
                          INLPBKSL INLPBKS2 INLPBLDL INLPCONS
                          INLPGDIR INLPLID  INLPLSIM INLPRABF
                          INLPRCLM INLPREL  INLPRELB INLPREMB
                          INLPRESL INLPRES2 INLPROLD
                          INLPSLD  INLPSLIB INLPSTOW INLPTD
                          INLPUSDL INLXREST
INLCTDCL is invoked by    INLPBKLB INLPBKM2 INLPBKSA INLPBKSL
                          INLPBKS2 INLPBKUP INLPLLSR INLPRABF
                          INLPRDTP INLPRELB INLPREMB INLPREM2
                          INLPRESL INLPREST INLPRES2 INLPTI
                          INLPTO   INLPWRTP INLXREST
LBRCTUPD is invoked by    IJBLBHLS
INLCTYPE is invoked by    INLPGDIR INLPMCON INLPSLD  INLPSLIB
                          INLPSTOW INLPUSDL
```

```
LBRLCTAC is invoked by    IJBLBHLS
LBRLCTCB is invoked by    IJBLBHLS IJBLBLLS
INLMBLDL is invoked by    INLPBKLB INLPBKSL INLPCOMP INLPCOPY
                          INLPDEL  INLPLID  INLPLIPU INLPREL
                          INLPREN  INLPTD
INLMFIND is invoked by    INLPBLDL INLPCAT  INLPCOPY INLPLIPU
                          INLPUPD
INLMGDIR is invoked by    IJBLBIPL INLPBKLB INLPBKS2 INLPBLDL
                          INLPLID  INLPTD
INLMGETR is invoked by    INLPBKM2 INLPBKSA INLPCOMP INLPCOPY
                          INLPLIPU INLPUPD
INLMLAMB is invoked by    IJBCTUPD IJBLBIPL INLPBKLB INLPBKM2
                          INLPBKSA INLPBKSL INLPBKS2 INLPBKUP
                          INLPCAT  INLPCHAN INLPCOMM INLPCOMP
                          INLPCONN INLPCOPY INLPDEF  INLPDEL
                          INLPDIAG INLPEXIT INLPLEV3 INLPLID
                          INLPLIPU INLPMIGR INLPMSG  INLPOPEN
                          INLPOUT  INLPPIDT INLPPUN  INLPRABF
                          INLPRALU INLPRDTP INLPREAD INLPREL
                          INLPRELB INLPREMB INLPREM2 INLPREN
                          INLPRESL INLPREST INLPRES2 INLPROLD
                          INLPRTI  INLPSYNA INLPTD   INLPTDLH
                          INLPTDX
                          INLPTI   INLPUPD  INLPWOR  INLPWTO
                          INLPWTP  INLXREST
INLMLCON is invoked by    IJBCTUPD IJBLBIPL INLPBKLB INLPBLDL
                          INLPCHAN INLPCOPY INLPDEF  INLPDEL
                          INLPLID  INLPRCLM INLPRELB INLPREN
                          INLPRES2 INLPTD   INLXREST
INLMLDIS is invoked by    IJBCTUPD IJBLBIPL INLPBKLB INLPBKSA
                          INLPBKSL INLPBLDL INLPCAT  INLPCHAN
                          INLPCOMP INLPCOPY INLPDEF  INLPDEL
                          INLPLID  INLPLIPU INLPRCLM INLPRELB
                          INLPREMB INLPREN  INLPRESL INLPRES2
                          INLPROLD INLPTD   INLPUPD  INLXREST
INLMLRPL is invoked by    IJBCTUPD IJBLBIPL INLPACC  INLPBKLB
                          INLPBKM2 INLPBKSA INLPBKSL INLPBKS2
                          INLPBKUP INLPBLDL INLPCAT  INLPCHAN
                          INLPCOMP INLPCONN INLPCOPY INLPDEF
                          INLPDEL  INLPLID  INLPLIPU INLPMIGR
                          INLPOPEN INLPRCLM INLPREL  INLPRELB
                          INLPREMB INLPREM2 INLPREN  INLPRESL
                          INLPREST INLPRES2 INLPROLD INLPTD
                          INLPTDLH INLPTX   INLPUPD  INLXREST
INLMLSIM is invoked by    IJBCTUPD INLPBLDL INLPCOMP INLPCOPY
                          INLPGDIR INLPMCON INLPREL  INLPREMB
                          INLPSLIB INLPSTOW INLPTD
INLML1TD is invoked by    INLPFBUF INLPGBUF INLPLBGP
INLMMCON is invoked by    INLPBKM2 INLPBKSA INLPCAT  INLPCOMP
                          INLPCOPY INLPFIND INLPLIPU INLPREM2
                          INLPROLD INLPTD   INLPUPD
```

```
INLMMDIS is invoked by    INLPBKM2 INLPBKSA INLPBLDL INLPCAT
                          INLPCOMP INLPCOPY INLPLIPU INLPREM2
                          INLPROLD INLPTD   INLPUPD
INLMPIDT is invoked by    IJBCTUPD IJBLBBRN
INLMPROC is invoked by    INLPACC  INLPAREA INLPBBUF INLPBKLB
                          INLPBKM2 INLPBKSA INLPBKSL INLPBKS2
                          INLPBKUP INLPBSPA INLPCAT  INLPCHAN
                          INLPCMPR INLPCOMP INLPCONN INLPCOPY
                          INLPDBLO INLPDEF  INLPDEL  INLPDOIO
                          INLPEXIT INLPFBUF INLPGBUF INLPGSPA
                          INLPGST  INLPLBGP INLPLEV3 INLPLID
                          INLPLIPU INLPLLSR INLPMAIN INLPMIBK
                          INLPMICO INLPMICS INLPMIDS INLPMIGR
                          INLPMIMA INLPMIPS INLPMIRE INLPMIRS
                          INLPMISS INLPMSG  INLPNOTE INLPOUT
                          INLPPOIN INLPPUN  INLPPUTR INLPQNAM
                          INLPRABF INLPRALU INLPRDTP INLPREAD
                          INLPREL  INLPRELB INLPREMB INLPREM2
                          INLPREN  INLPRESL INLPREST INLPRES2
                          INLPRIPL INLPROLD INLPRTI  INLPSYNA
                          INLPSYNX INLPSYPA INLPTD   INLPTDLH
                          INLPTDX
                          INLPTI   INLPTO   INLPUPD  INLPWOR
                          INLPWRTP INLPWTO  INLPWTP  INLXREST
```

Note: In newer modules, INLMPRAS is used instead of INLMPROC, because of its PLAS compatibility.

```
INLMPUTR is invoked by    INLPCAT  INLPCOPY INLPREM2 INLPROLD
                          INLPUPD
INLMRCLM is invoked by    IJBCTUPD IJBLBIPL INLPREL
INLMSCON is invoked by    IJBCTUPD IJBLBIPL INLPBKSA INLPBKS2
                          INLPCAT  INLPCOPY INLPDEF  INLPDEL
                          INLPRCLM INLPREMB INLPREN  INLPRES2
                          INLPROLD INLPTD
INLMSDIS is invoked by    INLPBKSA INLPBKS2 INLPCOMP INLPRCLM
                          INLPRES2
INLMSLD  is invoked by    IJBCTUPD IJBLBIPL INLPRCLM INLPSTOW
INLMSTOW is invoked by    INLPCAT  INLPCHAN INLPCOPY INLPDEF
                          INLPDEL  INLPREM2 INLPREN  INLPRES2
                          INLPROLD INLPUPD
LBRUPDAT is invoked by    IJBLBULT INLPCOPY INLPREM2 INLPRES2
                          INLPROLD INLPSLIB INLPSTOW INLPTD
```

Librarian Module-Message Interrelations

Sorted by Modules

```
IJBCTUPD causes  MSG101   MSG159   MSG160   MSG162   MSG258
IJBLBBRN causes  MSG101   MSG151   MSG152   MSG251   MSG252
                 MSG253   MSG254   MSG255   MSG256   MSG257
                 MSG259   MSG260   MSG261   MSG262   MSG263
                 MSG264   MSG265   MSG266   MSG267   MSG268
                 MSG269   MSG270   MSG271   MSG272   MSG273
                 MSG274   MSG275   MSG276   MSG277   MSG278
IJBLBULT causes  MSG151   MSG152   MSG280   MSG281   MSG282
                 MSG283
INLPACC  causes  MSG022   MSG025   MSG027   MSG028   MSG042
                 MSG101   MSG159
INLPBKLB causes  MSG022   MSG071   MSG072   MSG073   MSG098
                 MSG101   MSG137   MSG151   MSG159   MSG252
                 MSG258
INLPBKM2 causes  MSG035   MSG081   MSG151
INLPBKSA causes  MSG042   MSG056   MSG078   MSG082   MSG083
                 MSG118   MSG150   MSG155   MSG159
INLPBKSL causes  MSG042   MSG073   MSG159
INLPBKS2 causes  MSG079   MSG080   MSG121   MSG151
INLPBKUP causes  MSG022   MSG027   MSG061   MSG070   MSG075
INLPCAT  causes  MSG020   MSG022   MSG027   MSG028   MSG029
                 MSG030   MSG031   MSG032   MSG033   MSG034
                 MSG035   MSG036   MSG039   MSG201
INLPCHAN causes  MSG022   MSG027   MSG042   MSG101   MSG105
                 MSG159
INLPCOMM causes  MSG151   MSG152   MSG157   MSG158   MSG257
                 MSG258
INLPCOMP causes  MSG007   MSG008   MSG022   MSG027   MSG028
                 MSG042   MSG049   MSG072   MSG073   MSG082
                 MSG101   MSG104   MSG112   MSG136   MSG137
                 MSG139   MSG141   MSG143   MSG149   MSG159
INLPCONN causes  MSG022   MSG027   MSG028   MSG042   MSG101
                 MSG130   MSG139   MSG159
INLPCOPY causes  MSG022   MSG027   MSG028   MSG031   MSG032
                 MSG038   MSG042   MSG072   MSG073   MSG082
                 MSG101   MSG131   MSG136   MSG137   MSG139
                 MSG148   MSG149   MSG159   MSG201
INLPDEF  causes  MSG022   MSG027   MSG037   MSG038   MSG041
                 MSG101   MSG128   MSG131   MSG133   MSG148
                 MSG159   MSG201
INLPDEL  causes  MSG022   MSG027   MSG028   MSG037   MSG038
                 MSG040   MSG042   MSG082   MSG101   MSG149
                 MSG151   MSG159   MSG256   MSG258
INLPEXIT causes  MSG113
INLPLID  causes  MSG022   MSG027   MSG028   MSG042   MSG082
                 MSG101   MSG102   MSG103   MSG151   MSG159
                 MSG252   MSG258
INLPLIPU causes  MSG022   MSG027   MSG028   MSG029   MSG035
                 MSG082   MSG129   MSG134   MSG138   MSG142
```

```
INLPLSIM causes  MSG163   MSG252
INLPMAIN causes  MSG115   MSG140   MSG152
INLPMIGR causes  MSG010   MSG023   MSG024   MSG027   MSG042
                 MSG101   MSG146   MSG159
INLPPUTR causes  MSG135
INLPRABF causes  MSG022
INLPRALU causes  MSG060   MSG062   MSG063   MSG064   MSG065
                 MSG151   MSG256   MSG258
INLPRDTP causes  MSG050   MSG054   MSG057   MSG124   MSG125
INLPREAD causes  MSG001   MSG002   MSG003   MSG004
INLPREL  causes  MSG022   MSG027   MSG042   MSG047   MSG048
                 MSG101   MSG137   MSG144   MSG145   MSG159
INLPRELB causes  MSG037   MSG041   MSG050   MSG066   MSG068
                 MSG069   MSG072   MSG076   MSG090   MSG099
                 MSG128   MSG159
INLPREMB causes  MSG028   MSG042   MSG074   MSG079   MSG092
                 MSG101   MSG150   MSG159
INLPREM2 causes  MSG031   MSG032   MSG150   MSG201
INLPREN  causes  MSG022   MSG027   MSG028   MSG038   MSG042
                 MSG043   MSG044   MSG045   MSG046   MSG049
                 MSG082   MSG101   MSG132   MSG149   MSG159
                 MSG201
INLPRESL causes  MSG038   MSG091   MSG101   MSG159
INLPREST causes  MSG027   MSG050   MSG052   MSG059   MSG061
                 MSG075   MSG085   MSG086   MSG087   MSG088
                 MSG111   MSG123   MSG126
INLPRES2 causes  MSG073   MSG077   MSG101   MSG131   MSG148
                 MSG150   MSG159   MSG201
INLPRIPL causes  MSG022   MSG120   MSG151
INLPROLD causes  MSG022   MSG031   MSG038   MSG050   MSG053
                 MSG058   MSG072   MSG076   MSG081   MSG085
                 MSG090   MSG093   MSG094   MSG095   MSG096
                 MSG097   MSG101   MSG119   MSG126   MSG131
                 MSG159   MSG201
INLPRTI  causes  MSG051   MSG055   MSG351   MSG355
INLPSDL  causes  MSG169   MSG170   MSG171   MSG172   MSG173
                 MSG174   MSG175   MSG177   MSG178   MSG179
INLPSLD  causes  MSG176   MSG325
INLPSTOW causes  MSG032   MSG164
INLPSYNA causes  MSG009   MSG010   MSG011   MSG110   MSG147
INLPSYNX causes  MSG005   MSG006   MSG114
INLPSYPA causes  MSG011   MSG013   MSG015   MSG016   MSG021
                 MSG026   MSG111   MSG113   MSG152
```

```
INLPTD   causes  MSG027   MSG028   MSG042   MSG082   MSG101
                 MSG106   MSG107   MSG108   MSG109
INLPTI   causes  MSG051   MSG055   MSG351   MSG355
INLPUPD  causes  MSG012   MSG014   MSG017   MSG018   MSG019
                 MSG020   MSG022   MSG027   MSG028   MSG029
                 MSG032   MSG034   MSG044   MSG049   MSG082
                 MSG201
INLXREST causes  MSG027   MSG050   MSG052   MSG300   MSG301
                 MSG302   MSG303   MSG304   MSG305   MSG306
                 MSG307   MSG308   MSG309   MSG310   MSG311
                 MSG312   MSG313   MSG314   MSG315   MSG316
                 MSG317   MSG318   MSG319   MSG320   MSG321
                 MSG322   MSG323   MSG324   MSG325   MSG326
                 MSG327   MSG328   MSG329   MSG330   MSG331
                 MSG332   MSG333   MSG334   MSG335   MSG336
                 MSG337   MSG338   MSG339   MSG340   MSG341
                 MSG342   MSG343   MSG344   MSG345   MSG346
```

Sorted by Messages

```
MSG001 is caused by       INLPREAD
MSG002 is caused by       INLPREAD
MSG003 is caused by       INLPREAD
MSG004 is caused by       INLPREAD
MSG005 is caused by       INLPSYNX
MSG006 is caused by       INLPSYNX
MSG007 is caused by       INLPCOMP
MSG008 is caused by       INLPCOMP
MSG009 is caused by       INLPSYNA
MSG010 is caused by       INLPMIGR INLPSYNA
MSG011 is caused by       INLPSYNA INLPSYPA
MSG012 is caused by       INLPUPD
MSG013 is caused by       INLPSYPA
MSG014 is caused by       INLPUPD
MSG015 is caused by       INLPSYPA
MSG016 is caused by       INLPSYPA
MSG017 is caused by       INLPUPD
MSG018 is caused by       INLPUPD
MSG019 is caused by       INLPUPD
MSG020 is caused by       INLPCAT  INLPUPD
MSG021 is caused by       INLPSYPA
MSG022 is caused by       INLPACC  INLPBKLB INLPBKUP INLPCAT
                          INLPCHAN INLPCOMP INLPCONN INLPCOPY
                          INLPDEF  INLPDEL  INLPLID  INLPLIPU
                          INLPRABF INLPREL  INLPREN  INLPRIPL
                          INLPROLD INLPUPD
MSG023 is caused by       INLPMIGR
MSG024 is caused by       INLPMIGR
MSG025 is caused by       INLPACC
MSG026 is caused by       INLPSYPA
MSG027 is caused by       INLPACC  INLPBKUP INLPCAT  INLPCHAN
                          INLPCOMP INLPCONN INLPCOPY INLPDEF
                          INLPDEL  INLPLID  INLPLIPU INLPMIGR
                          INLPREL  INLPREN  INLPREST INLPTD
                          INLPUPD
MSG028 is caused by       INLPACC  INLPCAT  INLPCOMP INLPCONN
                          INLPCOPY INLPDEL  INLPLID  INLPLIPU
                          INLPREMB INLPREN  INLPTD   INLPUPD
MSG029 is caused by       INLPCAT  INLPLIPU INLPUPD
MSG030 is caused by       INLPCAT
MSG031 is caused by       INLPCAT  INLPCOPY INLPREM2 INLPROLD
MSG032 is caused by       INLPCAT  INLPCOPY INLPREM2 INLPSTOW
                          INLPUPD
MSG033 is caused by       INLPCAT
MSG034 is caused by       INLPCAT  INLPUPD
MSG035 is caused by       INLPBKM2 INLPCAT  INLPLIPU
MSG036 is caused by       INLPCAT
MSG037 is caused by       INLPDEF  INLPDEL  INLPRELB
MSG038 is caused by       INLPCOPY INLPDEF  INLPDEL  INLPREN
                          INLPRESL INLPROLD
```

```
MSG039 is caused by       INLPCAT
MSG040 is caused by       INLPDEL
MSG041 is caused by       INLPDEF  INLPRELB
MSG042 is caused by       INLPACC  INLPBKSA INLPBKSL INLPCHAN
                          INLPCOMP INLPCONN INLPCOPY INLPDEL
                          INLPLID  INLPMIGR INLPREL  INLPREMB
                          INLPREN  INLPTD
MSG043 is caused by       INLPREN
MSG044 is caused by       INLPREN  INLPUPD
MSG045 is caused by       INLPREN
MSG046 is caused by       INLPREN
MSG047 is caused by       INLPREL
MSG048 is caused by       INLPREL
MSG049 is caused by       INLPCOMP INLPREN  INLPUPD
MSG050 is caused by       INLPRDTP INLPRELB INLPREST INLPROLD
MSG051 is caused by       INLPRTI  INLPTI
MSG052 is caused by       INLPREST
MSG053 is caused by       INLPROLD
MSG054 is caused by       INLPRDTP
MSG055 is caused by       INLPRTI  INLPTI
MSG056 is caused by       INLPBKSA
MSG057 is caused by       INLPRDTP
MSG058 is caused by       INLPROLD
MSG059 is caused by       INLPREST
MSG060 is caused by       INLPRALU
MSG061 is caused by       INLPBKUP INLPREST
MSG062 is caused by       INLPRALU
MSG063 is caused by       INLPRALU
MSG064 is caused by       INLPRALU
MSG065 is caused by       INLPRALU
MSG066 is caused by       INLPRELB
MSG067 is caused by       INLPREST
MSG068 is caused by       INLPRELB
MSG069 is caused by       INLPRELB
MSG071 is caused by       INLPBKLB
MSG072 is caused by       INLPBKLB INLPCOMP INLPCOPY INLPRELB
                          INLPROLD
MSG073 is caused by       INLPBKLB INLPBKSL INLPCOMP INLPCOPY
                          INLPRES2
MSG074 is caused by       INLPREMB
MSG075 is caused by       INLPBKUP INLPREST
MSG076 is caused by       INLPRELB INLPROLD
MSG077 is caused by       INLPRES2
MSG078 is caused by       INLPBKSA
MSG079 is caused by       INLPBKS2 INLPREMB
MSG080 is caused by       INLPBKS2
MSG081 is caused by       INLPBKM2 INLPROLD
MSG082 is caused by       INLPBKSA INLPCOMP INLPCOPY INLPDEL
                          INLPLID  INLPLIPU INLPREN  INLPTD
                          INLPUPD
MSG083 is caused by       INLPBKSA
MSG084 is caused by       INLPBKSA
MSG085 is caused by       INLPREST INLPROLD
MSG086 is caused by       INLPREST
```

```
MSG087 is caused by       INLPREST
MSG088 is caused by       INLPREST
MSG089 is caused by       INLPBKUP
MSG090 is caused by       INLPRELB INLPROLD
MSG091 is caused by       INLPRESL
MSG092 is caused by       INLPREMB
MSG093 is caused by       INLPROLD
MSG094 is caused by       INLPROLD
MSG095 is caused by       INLPROLD
MSG096 is caused by       INLPROLD
MSG097 is caused by       INLPROLD
MSG098 is caused by       INLPBKLB
MSG099 is caused by       INLPRELB
MSG101 is caused by       IJBCTUPD IJBLBBRN INLPACC  INLPBKLB
                          INLPCHAN INLPCOMP INLPCONN INLPCOPY
                          INLPDEF  INLPDEL  INLPLID  INLPMIGR
                          INLPREL  INLPREMB INLPREN  INLPRESL
                          INLPRES2 INLPROLD INLPTD
MSG102 is caused by       INLPLID
MSG103 is caused by       INLPLID
MSG104 is caused by       INLPCOMP
MSG105 is caused by       INLPCHAN
MSG106 is caused by       INLPTD
MSG107 is caused by       INLPTD
MSG108 is caused by       INLPTD
MSG109 is caused by       INLPTD
MSG110 is caused by       INLPSYNA
MSG111 is caused by       INLPREST INLPSYPA
MSG112 is caused by       INLPCOMP
MSG113 is caused by       INLPEXIT INLPSYPA
MSG114 is caused by       INLPSYNX
MSG115 is caused by       INLPMAIN
MSG116 is caused by       INLPDEF
MSG118 is caused by       INLPBKSA
MSG119 is caused by       INLPROLD
MSG120 is caused by       INLPRIPL
MSG121 is caused by       INLPBKS2
MSG123 is caused by       INLPREST
MSG124 is caused by       INLPRDTP
MSG125 is caused by       INLPRDTP
MSG126 is caused by       INLPREST INLPROLD
MSG127 is caused by       INLPBKUP
MSG128 is caused by       INLPDEF  INLPRELB
MSG129 is caused by       INLPLIPU
MSG130 is caused by       INLPCONN
MSG131 is caused by       INLPCOPY INLPDEF  INLPRES2 INLPROLD
MSG132 is caused by       INLPREN
MSG133 is caused by       INLPDEF
MSG134 is caused by       INLPLIPU
MSG135 is caused by       INLPPUTR
MSG136 is caused by       INLPCOMP INLPCOPY
MSG137 is caused by       INLPBKLB INLPCOMP INLPCOPY INLPREL
MSG138 is caused by       INLPLIPU
MSG139 is caused by       INLPCOMP INLPCONN INLPCOPY
MSG140 is caused by       INLPMAIN
MSG141 is caused by       INLPCOMP
MSG142 is caused by       INLPLIPU
MSG143 is caused by       INLPCOMP
```

```
MSG144 is caused by       INLPREL
MSG145 is caused by       INLPREL
MSG146 is caused by       INLPMIGR
MSG147 is caused by       INLPSYNA
MSG148 is caused by       INLPCOPY INLPDEF  INLPRES2
MSG149 is caused by       INLPCOMP INLPCOPY INLPDEL  INLPREN
MSG150 is caused by       INLPBKSA INLPREMB INLPREM2 INLPRES2
MSG151 is caused by       IJBLBBRN IJBLBULT INLPBKLB INLPBKM2
                          INLPBKS2 INLPCOMM INLPDEL  INLPLID
                          INLPRALU INLPRIPL
MSG152 is caused by       IJBLBBRN IJBLBULT INLPCOMM INLPMAIN
                          INLPSYPA
MSG153 is caused by       IJBCTUPD
MSG154 is caused by       IJBLBLDL
MSG155 is caused by       INLPBKSA
MSG157 is caused by       INLPCOMM
MSG158 is caused by       INLPCOMM
MSG159 is caused by       IJBCTUPD INLPACC  INLPBKLB INLPBKSA
                          INLPBKSL INLPCHAN INLPCOMP INLPCONN
                          INLPCOPY INLPDEF  INLPDEL  INLPLID
                          INLPMIGR INLPREL  INLPRELB INLPREMB
                          INLPREN  INLPRESL INLPRES2 INLPROLD
MSG160 is caused by       IJBCTUPD
MSG162 is caused by       IJBCTUPD
MSG163 is caused by       INLPLSIM
MSG164 is caused by       INLPSTOW
MSG165 is caused by       INLPIASV
MSG169 is caused by       INLPSDL
MSG170 is caused by       INLPSDL
MSG171 is caused by       INLPSDL
MSG172 is caused by       INLPSDL
MSG173 is caused by       INLPSDL
MSG174 is caused by       INLPSDL
MSG175 is caused by       INLPSDL
MSG176 is caused by       INLPSDL
MSG177 is caused by       INLPSLD
MSG178 is caused by       INLPSLD
MSG179 is caused by       INLPSLD
MSG180 is caused by       INLPSRCH
MSG181 is caused by       INLPSRCH
MSG182 is caused by       INLPSRCH
MSG183 is caused by       INLPSRCH
MSG184 is caused by       INLPSRCH
MSG185 is caused by       INLPSRCH
MSG186 is caused by       INLPSRCH
MSG187 is caused by       INLPLID
MSG188 is caused by       INLPLID
MSG189 is caused by       INLPLOCK
MSG190 is caused by       INLPTDLH
MSG191 is caused by       INLPTDLH
MSG192 is caused by       INLPTDLH
MSG193 is caused by       INLPCOMP
MSG201 is caused by       INLPCAT  INLPCOPY INLPDEF  INLPREM2
                          INLPREN  INLPRES2 INLPROLD INLPUPD
MSG251 is caused by       IJBLBBRN
MSG252 is caused by       IJBLBBRN INLPBKLB INLPLID  INLPLSIM
MSG253 is caused by       IJBLBBRN
MSG254 is caused by       IJBLBBRN
```

```
MSG255 is caused by       IJBLBBRN
MSG256 is caused by       IJBLBBRN INLPDEL  INLPRALU
MSG257 is caused by       IJBLBBRN INLPCOMM INLPGVFV
MSG258 is caused by       IJBCTUPD INLPBKLB INLPCOMM INLPDEL
                          INLPLID  INLPRALU
MSG259 is caused by       IJBLBBRN
MSG260 is caused by       IJBLBBRN
MSG261 is caused by       IJBLBBRN
MSG262 is caused by       IJBLBBRN
MSG263 is caused by       IJBLBBRN
MSG264 is caused by       IJBLBBRN
MSG265 is caused by       IJBLBBRN
MSG266 is caused by       IJBLBBRN
MSG267 is caused by       IJBLBBRN
MSG268 is caused by       IJBLBBRN
MSG269 is caused by       IJBLBBRN
MSG270 is caused by       IJBLBBRN
MSG271 is caused by       IJBLBBRN
MSG272 is caused by       IJBLBBRN
MSG273 is caused by       IJBLBBRN
MSG274 is caused by       IJBLBBRN
MSG275 is caused by       IJBLBBRN
MSG276 is caused by       IJBLBBRN
MSG277 is caused by       IJBLBBRN
MSG278 is caused by       IJBLBBRN
MSG280 is caused by       IJBLBULT
MSG281 is caused by       IJBLBULT
MSG282 is caused by       IJBLBULT
MSG283 is caused by       IJBLBULT
MSG284 is caused by       INLPLOCK
MSG285 is caused by       INLPLOCK
MSG286 is caused by       INLPCOPY
MSG287 is caused by       INLPLOCK
MSG288 is caused by       INLPLOCK
MSG289 is caused by       INLPSTOX
MSG290 is caused by       INLPTD
MSG291 is caused by       INLPTD
MSG292 is caused by       INLPSLIB
MSG293 is caused by       INLPIALC
MSG294 is caused by       INLPCOPY
MSG300 is caused by       INLXREST
MSG301 is caused by       INLXREST
MSG302 is caused by       INLXREST
MSG303 is caused by       INLXREST
MSG304 is caused by       INLXREST
MSG305 is caused by       INLXREST
MSG306 is caused by       INLXREST
MSG307 is caused by       INLXREST
MSG308 is caused by       INLXREST
MSG309 is caused by       INLXREST
MSG310 is caused by       INLXREST
MSG311 is caused by       INLXREST
MSG312 is caused by       INLXREST
MSG313 is caused by       INLXREST
MSG314 is caused by       INLXREST
MSG315 is caused by       INLXREST
MSG316 is caused by       INLXREST
MSG317 is caused by       INLXREST
```

```
MSG318 is caused by       INLXREST
MSG319 is caused by       INLXREST
MSG320 is caused by       INLXREST
MSG321 is caused by       INLXREST
MSG322 is caused by       INLXREST
MSG323 is caused by       INLXREST
MSG324 is caused by       INLXREST
MSG325 is caused by       INLXREST
MSG326 is caused by       INLXREST
MSG327 is caused by       INLXREST
MSG328 is caused by       INLXREST
MSG329 is caused by       INLXREST
MSG330 is caused by       INLXREST
MSG331 is caused by       INLXREST
MSG332 is caused by       INLXREST
MSG333 is caused by       INLXREST
MSG334 is caused by       INLXREST
MSG335 is caused by       INLXREST
MSG336 is caused by       INLXREST
MSG337 is caused by       INLXREST
MSG338 is caused by       INLXREST
MSG339 is caused by       INLXREST
MSG340 is caused by       INLXREST
MSG341 is caused by       INLXREST
MSG342 is caused by       INLXREST
MSG343 is caused by       INLXREST
MSG344 is caused by       INLXREST
MSG345 is caused by       INLXREST
MSG346 is caused by       INLXREST
MSG351 is caused by       INLPTI   INLPRTI
MSG355 is caused by       INLPTI   INLPRTI
```

Librarian Module-INLCCOMR Interrelations

Sorted by Using Modules

```
INLPACC  uses      CRGCMDP  CRGLAMB
INLPBKLB uses      CRGCMDP  CRGRETCD CRGCAREQ
INLPBKM2 uses      CRGRETCD CRGCAREQ
INLPBKSA uses      CRGCMDP  CRGRETCD CRGCAREQ
INLPBKSL uses      CRGCMDP  CRGCAREQ
INLPBKS2 uses      CRGRETCD CRGCAREQ
INLPBKUP uses      CRGCMDP  CRGLAMB  CRGLOG   CRGRETCD CRGCAREQ
                   CRGSAVRC
INLPCAT  uses      CRGCMDP  CRGIPTIO CRGLAMB  CRGMSHP  CRGMSHPB
                   CRGLOG   CRGEOF   CRGEODCK CRGMSG03 CRGRETCD
                   CRGCAREQ
INLPCHAN uses      CRGCMDP  CRGLAMB  CRGLOG   CRGRETCD CRGCAREQ
INLPCOMP uses      CRGCMDP  CRGLAMB  CRGLOG   CRGRETCD CRGCAREQ
INLPCONN uses      CRGCMDP  CRGLAMB  CRGCONN
INLPCOPY uses      CRGCMDP  CRGLAMB  CRGMSHP  CRGMSHPB CRGLOG
                   CRGCONN  CRGRETCD CRGSRALL CRGCAREQ
INLPDBLO uses      CRGCMDP
INLPDEF  uses      CRGCMDP  CRGLAMB  CRGLOG   CRGCONN  CRGRETCD
                   CRGCAREQ
INLPDEL  uses      CRGCMDP  CRGLAMB  CRGLOG   CRGCONN  CRGRETCD
                   CRGSRALL CRGCAREQ
INLPEXIT uses      CRGCMDP  CRGNMLIS CRGLAMB  CRGMSHP  CRGLOG
                   CRGRETCD CRGGTVIS CRGSYNER CRGNMLEN CRGSAVRC
INLPLEV3 uses      CRGCMDP  CRGLAMB  CRGLOG   CRGRETCD CRGMIGR
                   CRGCAREQ CRGSAVRC
INLPLID  uses      CRGCMDP  CRGLAMB  CRGLOG   CRGRETCD CRGSRALL
                   CRGCAREQ
INLPLIPU uses      CRGCMDP  CRGLAMB  CRGLOG   CRGRETCD CRGMIGR
                   CRGSRALL CRGPUNSA CRGCAREQ
INLPLLSR uses      CRGCMDP  CRGLOG
INLPMAIN uses      CRGCMDL  CRGLOAD  CRGCMDP  CRGNMLIS CRGNMLGT
                   CRGDTFIN CRGIPTL  CRGIPTIO CRGDTFPC CRGPCHL
                   CRGPCHIO CRGLAMB  CRGOPIN  CRGOPPCH CRG81BYT
                   CRGOPLOG CRGOPLST CRGACC   CRGMSHP  CRGMSHPB
                   CRGLOG   CRGCONN  CRGRETCD CRGMIGR  CRGMAINT
                   CRGCORGZ CRGCSERV CRGDSERV CRGPSERV CRGSSERV
                   CRGRSERV CRGPUNSA CRGCAREQ CRGSAVRC
INLPMIGR uses      CRGCMDP  CRGLAMB  CRGRETCD CRGMAINT CRGCORGZ
                   CRGCSERV CRGDSERV CRGPSERV CRGSSERV CRGRSERV
                   CRGSRALL
INLPMSG  uses      CRGLAMB
INLPOUT  uses      CRGLAMB
INLPPUN  uses      CRGDTFPC CRGPCHL  CRGPCHIO CRGLAMB  CRGOPPCH
INLPRALU uses      CRGCMDP
INLPRDTP uses      CRGCMDP  CRGCAREQ CRGPIDV2
INLPREAD uses      CRGCMDP  CRGDTFIN CRGIPTL  CRGLAMB  CRGOPIN
                   CRG81BYT CRGLOG   CRGEOF   CRGMSG01 CRGMSG03
                   CRGCOMM  CRGRQ1ST CRGMSG04
```

```
INLPREL  uses      CRGCMDP  CRGLAMB  CRGLOG   CRGRETCD CRGCAREQ
INLPRELB uses      CRGCMDP  CRGCONN  CRGRETCD CRGCAREQ
INLPREMB uses      CRGCMDP  CRGLAMB  CRGRETCD CRGCAREQ
INLPREM2 uses      CRGCMDP  CRGMSHP  CRGMSHPB CRGRETCD CRGCAREQ
INLPREN  uses      CRGCMDP  CRGLAMB  CRGLOG   CRGCONN  CRGRETCD
                   CRGCAREQ
INLPRESL uses      CRGCMDP  CRGRETCD CRGCAREQ
INLPREST uses      CRGCMDP  CRGLAMB  CRGMSHP  CRGLOG   CRGRETCD
                   CRGCAREQ CRGSAVRC
INLPRES2 uses      CRGCMDP  CRGLAMB  CRGRETCD CRGCAREQ
INLPROLD uses      CRGCMDP  CRGLAMB  CRGMSHP  CRGMSHPB CRGLOG
                   CRGRETCD CRGCAREQ
INLPRTI  uses      CRGLAMB
INLPSYNA uses      CRGCMDL  CRGCMDP  CRGIPTIO CRGLAMB  CRGACC
                   CRGLOG   CRGEOF   CRGEODCK CRGMSG01 CRGMSG03
                   CRGRETCD CRGSYNER CRGMIGR  CRGMAINT CRGCORGZ
                   CRGONFLG CRGSAVRC CRGONIND
INLPSYNX uses      CRGMSG01 CRGCOMM  CRGSYNER
INLPSYPA uses      CRGLOAD  CRGCMDP  CRGNMLIS CRGPHPTR CRGPHN
                   CRGEOF   CRGRETCD CRGGTVIS CRGSYNER CRGNMLEN
                   CRGSAVRC
INLPTD   uses      CRGCMDP  CRGLAMB  CRGLOG   CRGRETCD CRGCAREQ
                   CRGSAVRC
INLPTDLH uses      CRGCMDP  CRGLOG   CRGCAREQ
INLPTI   uses      CRGLAMB
INLPUPD  uses      CRGCMDP  CRGIPTIO CRGLAMB  CRGMSHP  CRGMSHPB
                   CRGLOG   CRGEOF   CRGMSG04 CRGCAREQ
INLXREST uses      CRGCMDP  CRGLAMB  CRGOPLOG CRGOPLST
```

Sorted by Used Variable

```
CRGACC   is used by       INLPMAIN INLPSYNA
CRGCAREQ is used by       INLPBKLB INLPBKM2 INLPBKSA INLPBKSL
                          INLPBKS2 INLPBKUP INLPCAT  INLPCHAN
                          INLPCOMP INLPCOPY INLPDEF  INLPDEL
                          INLPLEV3 INLPLID  INLPLIPU INLPMAIN
                          INLPRDTP INLPREL  INLPRELB INLPREMB
                          INLPREM2 INLPREN  INLPRESL INLPREST
                          INLPRES2 INLPROLD INLPTD   INLPTDLH
                          INLPUPD
CRGCMDL  is used by       INLPMAIN INLPSYNA
CRGCMDP  is used by       INLPACC  INLPBKLB INLPBKSA INLPBKSL
                          INLPBKUP INLPCAT  INLPCHAN INLPCOMP
                          INLPCONN INLPCOPY INLPDBLO INLPDEF
                          INLPDEL  INLPEXIT INLPLEV3 INLPLID
                          INLPLIPU INLPLLSR INLPMAIN INLPMIGR
                          INLPRALU INLPRDTP INLPREAD INLPREL
                          INLPRELB INLPREMB INLPREM2 INLPREN
                          INLPRESL INLPREST INLPRES2 INLPROLD
                          INLPSYNA INLPSYPA INLPTD   INLPTDLH
                          INLPUPD  INLXREST
CRGCOMM  is used by       INLPREAD INLPSYNX
CRGCONN  is used by       INLPCONN INLPCOPY INLPDEF  INLPDEL
                          INLPMAIN INLPRELB INLPREN
CRGCORGZ is used by       INLPMAIN INLPMIGR INLPSYNA
CRGCSERV is used by       INLPMAIN INLPMIGR
CRGDSERV is used by       INLPMAIN INLPMIGR
CRGDTFIN is used by       INLPMAIN INLPREAD
CRGDTFPC is used by       INLPMAIN INLPPUN
CRGEODCK is used by       INLPCAT  INLPSYNA
CRGEOF   is used by       INLPCAT  INLPREAD INLPSYNA INLPSYPA
                          INLPUPD
CRGGTVIS is used by       INLPEXIT INLPSYPA
CRGIPTIO is used by       INLPCAT  INLPMAIN INLPSYNA INLPUPD
CRGIPTL  is used by       INLPMAIN INLPREAD
CRGLAMB  is used by       INLPACC  INLPBKUP INLPCAT  INLPCHAN
                          INLPCOMP INLPCONN INLPCOPY INLPDEF
                          INLPDEL  INLPEXIT INLPLEV3 INLPLID
                          INLPLIPU INLPMAIN INLPMIGR INLPMSG
                          INLPOUT  INLPPUN  INLPREAD INLPREL
                          INLPREMB INLPREN  INLPREST INLPRES2
                          INLPROLD INLPRTI  INLPSYNA INLPTD
                          INLPTI   INLPUPD  INLXREST
CRGLOAD  is used by       INLPMAIN INLPSYPA
CRGLOG   is used by       INLPBKUP INLPCAT  INLPCHAN INLPCOMP
                          INLPCOPY INLPDEF  INLPDEL  INLPEXIT
                          INLPLEV3 INLPLID  INLPLIPU INLPLLSR
                          INLPMAIN INLPREAD INLPREL  INLPREN
                          INLPREST INLPROLD INLPSYNA INLPTD
                          INLPTDLH INLPUPD
CRGMAINT is used by       INLPMAIN INLPMIGR INLPSYNA
```

```
CRGMIGR  is used by       INLPLEV3 INLPLIPU INLPMAIN INLPSYNA
CRGMSG01 is used by       INLPREAD INLPSYNA INLPSYNX
CRGMSG03 is used by       INLPCAT  INLPREAD INLPSYNA
CRGMSG04 is used by       INLPREAD INLPUPD
CRGMSHP  is used by       INLPCAT  INLPCOPY INLPEXIT INLPMAIN
                          INLPREM2 INLPREST INLPROLD INLPUPD
CRGMSHPB is used by       INLPCAT  INLPCOPY INLPMAIN INLPREM2
                          INLPROLD INLPUPD
CRGNMLEN is used by       INLPEXIT INLPSYPA
CRGNMLGT is used by       INLPMAIN
CRGNMLIS is used by       INLPEXIT INLPMAIN INLPSYPA
CRGONFLG is used by       INLPSYNA
CRGONIND is used by       INLPSYNA
CRGOPIN  is used by       INLPMAIN INLPREAD
CRGOPLOG is used by       INLPMAIN INLXREST
CRGOPLST is used by       INLPMAIN INLXREST
CRGOPPCH is used by       INLPMAIN INLPPUN
CRGPCHIO is used by       INLPMAIN INLPPUN
CRGPCHL  is used by       INLPMAIN INLPPUN
CRGPHN   is used by       INLPSYPA
CRGPHPTR is used by       INLPSYPA
CRGPIDV2 is used by       INLPRDTP
CRGPSERV is used by       INLPMAIN INLPMIGR
CRGPUNSA is used by       INLPLIPU INLPMAIN
CRGRETCD is used by       INLPBKLB INLPBKM2 INLPBKSA INLPBKS2
                          INLPBKUP INLPCAT  INLPCHAN INLPCOMP
                          INLPCOPY INLPDEF  INLPDEL  INLPEXIT
                          INLPLEV3 INLPLID  INLPLIPU INLPMAIN
                          INLPMIGR INLPREL  INLPRELB INLPREMB
                          INLPREM2 INLPREN  INLPRESL INLPREST
                          INLPRES2 INLPROLD INLPSYNA INLPSYPA
                          INLPTD
CRGRQ1ST is used by       INLPREAD
CRGRSERV is used by       INLPMAIN INLPMIGR
CRGSAVRC is used by       INLPBKUP INLPEXIT INLPLEV3 INLPMAIN
                          INLPREST INLPSYNA INLPSYPA INLPTD
CRGSRALL is used by       INLPCOPY INLPDEL  INLPLID  INLPLIPU
                          INLPMIGR
CRGSSERV is used by       INLPMAIN INLPMIGR
CRGSYNER is used by       INLPEXIT INLPSYNA INLPSYNX INLPSYPA
CRG81BYT is used by       INLPMAIN INLPREAD
```

Data Area-Data Area Interrelations

Sorted by Addressing Areas

```
INLCLACB points to     INCLBHDR via     LACBHDR
                       INLCEDTE via     LACBAEDT
INLCLDTE points to     INLCEDTE via     LDTEEDTX
                       INLCSDTE via     LDTESDTX
INLCLPT  points to     INLCDDTE via     LPTDDT
                       INLCEDTE via     LPTEDT
                       INLCLDTE via     LPTLDT
                       INLCLOTP via     LPTLLOTA
                       INLCSDTE via     LPTSDT
INLCLRPL points to     INLCDENT via     LRPLDENT
                       INLCLAMB via     LRPLLAMB
                       INLCLARG via     LRPLARG
                       LIBINFO  via     LRPLINFO
INLCMACB points to     INLCSACB via     MACBSACB
INLCSACB points to     INLCLACB via     SACBLACB
INLCSDTE points to     INLCSLDE via     SDTESLDA
LBRACCDS points to     INLCLAMB via     LBRLAMB
                       LIBINFO  via     LBRINFO
LIBINFO  points to     INLCLDTE via     LBRINFOC
                       INLCLDTE via     LBRINFOP
                       INLCLOT  via     LBRINFOO
                       INLCSDTE via     LBRINFOS
```

Sorted by Referenced Areas

```
INLCBHDR is addressed by     INLCLACB via     LACBHDR
INLCDDTE is addressed by     INLCLPT  via     LPTDDT
INLCDENT is addressed by     INLCLRPL via     LRPLDENT
INLCEDTE is addressed by     INLCLACB via     LACBAEDT
                             INLCLDTE via     LDTEEDTX
                             INLCLPT  via     LPTEDT
INLCLACB is addressed by     INLCSACB via     SACBLACB
INLCLAMB is addressed by     INLCLRPL via     LRPLLAMB
                             LBRACCDS via     LBRLAMB
INLCLARG is addressed by     INLCLRPL via     LRPLARG
INLCLDTE is addressed by     INLCLPT  via     LPTLDT
                             LIBINFO  via     LBRINFOC
                             LIBINFO  via     LBRINFOP
INLCLOT  is addressed by     LIBINFO  via     LBRINFOO
INLCLOTP is addressed by     INLCLPT  via     LPTLLOTA
INLCSACB is addressed by     INLCMACB via     MACBSACB
INLCSDTE is addressed by     INLCLDTE via     LDTESDTX
                             INLCLPT  via     LPTSDT
                             LIBINFO  via     LBRINFOS
INLCSLDE is addressed by     INLCSDTE via     SDTESLDA
LIBINFO  is addressed by     INLCLRPL via     LRPLINFO
                             LBRACCDS via     LBRINFO
```

Data Areas

This chapter provides data area maps. The data areas can be divided into the following categories:

1. Data areas which describe the contents of a library and which are located on disk:

```
INLCDENT ... member descriptor (member directory entry)
             How to locate: entry in member index level 1 of
                            a sublibrary pointed to by SLXELXRA;
                            when in storage address is contained
                            in LRPLDENT, LARGDEPT.
INLCLCKV ... Lock VIF (variable information field, part of DENT)
             How to locate: at the end of corresponding DENT
                            entry exists a VIF list
INLCBIGM ... VIF for gigabyte members (variable information field)
             How to locate: at the end of corresponding DENT
                            entry exists a VIF list
INLCDESC ... member index entry descriptor
             How to locate: common start area of data areas
                            INLCTYPE, INLCMBRX, INLCDENT.
INLCEXTD ... extent descriptor
             How to locate: for first library extent at LB-# 1,
                            offset 0; for further extents, first
                            LB of extent, offset 0.
INLCLBCF ... library block control field
             How to locate: in each LB, at offset X'3DC'.
INLCLDES ... library descriptor
             How to locate: in LB-# 0, offset 0.
INLCMBRX ... (higher level) member index entry
             How to locate: entry in member index LBs of higher
                            level than 1.
INLCSLXE ... sublibrary descriptor (sublibrary index entry)
             How to locate: entry in sublibrary index starting
                            at LB-# 0, offset X'D8' (LDESPRBA).
INLCSPAD ... space descriptor
             How to locate: at LB-# 1, offset X'40'.
INLCTYPE ... member type entry
             How to locate: in each LB of the member index, e.g.
                            at offset 0.
INLCUPHA ... variable information field for members of type PHASE
             How to locate: in each descriptor for a phase, starting
                            at offset X'48'.
```

2. Data areas which describe the contents of a library and which are located on tape:

```
BFHDR ...... backup file header
BKUPFID .... backup file identification
DIPLPHDR ... disk IPL phase header
EOBKFID .... end-of-backup-file identification
EXTENTR .... extent information record
LIBRHDR .... library header
MEMBDAT .... member data descriptor
MEMBHDR .... member header
SUBLHDR .... sublibrary header
TAPEITEM ... tape item description

             How to locate: the currently processed tape item is
                            pointed to by BUFPTR which is located
                            at displacement X'08' in INLCBRCR in
                            INLPRABF of root phase LIBR.
                            The address BDBBFR at offset 4 in each
                            BDB points to the start of the
                            associated buffer in storage.
                            The first tape item in each buffer
                            starts at offset X'10' after the block
                            header BLKHDR.
```

3. Data areas which describe the contents of the "Stand-alone IPL file" and the "Stand-alone Utility file" located on a stand-alone tape:

```
INLCSABL.... Tape block header
INLCSAPH.... Header for stand-alone phase records
```

4. Data areas which describe the library search chain and all information which are necessary in order to access library data (These areas are called Library Control Tables (LCT's) and are located in the system GETVIS area.):

```
INLCDDTE ... device definition table entry
             How to locate: entry in DDT, pointed to by LPTDDT;
                            address also contained in EDTEDDTX.
INLCEDTE ... extent definition table entry
             How to locate: entry in EDT, pointed to by LPTEDT;
                            address also contained in LDTEEDTX,
                            EDTENEXT, LACBAEDT.
INLCIDTE ... OPEN identification table entry
             How to locate: entry in IDT, pointed to by LPTIDT.
INLCLANC ... library anchor table
             How to locate: LPTLANC pointer in INLCLPT.
INLCLDTE ... library definition table entry
             How to locate: entry in LDT, pointed to by LPTLDT;
                            address also contained in LIBINFOC,
                            LIBINFOP.
INLCLOT .... library offset table on partition basis
             How to locate: row of a LOT, pointed to by pointers
                            in the LPT for types PHASE, OBJ, SOURCE,
                            PROC, DUMP, LBR.
INLCLOTP ... library offset table on task basis
             How to locate: row in IALOT pointed to by LPTIALOT.
INLCLPT .... library pointer table
             How to locate: offset X'12C' (IJBPLCT) in SYSCOM.
INLCSDLE ... System Directory List (SDL) entry
             How to locate: entry in SDL.
INLCSDTE ... sublibrary definition table entry
             How to locate: entry in SDT pointed to by LPTSDT;
                            address is also contained in LDTESDTX,
                            LIBINFOS.
INLCSLDE ... Second Level Directory (SLD) entry
             How to locate: entry in SLD pointed to by SDTESLDA.
LOTENTRY ... Library Offset Table (LOT) entry
             How to locate: entry in a LOT row (data area INLCLOT).
LPTROW ..... Library Pointer Table (LPT) row
             How to locate: pointer triple (begin, end, next-free
                            address) in LPT for LDT, IDT, SDT, EDT,
                            DDT, and IALOT; pointer triple (address
                            of temporary, permanent, and system LOT)
                            for LOTs of type PHASE, OBJ, SOURCE,
                            PROC, DUMP, and LBR.
```

5. Data areas which describe the status of a library access (Such control blocks are built in the (system or partition) GETVIS area by Level 2 services):

```
INLCLACB ... library access control block
             How to locate: address contained in LRPLRQCB, if
                            LRPLLIB=TRUE; address also contained
                            in SACBLACB.
INLCLBTB ... library search table
             How to locate: address contained in LRPLLBTB.
INLCMACB ... member access control block
             How to locate: address contained in LRPLRQCB, if
                            LRPLMBR=TRUE.
INLCSACB ... sublibrary access control block
             How to locate: address contained in LRPLRQCB, if
                            LRPLSLIB=TRUE; address also contained
                            in MACBSACB.
```

6. Data areas used as parameter lists for Level 2 services which are provided by the caller (usually in the partition space):

```
BLDCBMAP ... parameter list for module IJBLBBRN
             How to locate: at entry of IJBLBBRN, address
                            is contained in Register 1.
INLCLAMB ... library access method block
             How to locate: address contained in LRPLLAMB(INLCLRPL),
                            TOLCLAMB (TOLCRQL), LBRLAMB (LBRACCDS),
                            PIDTLAMB (INLCPIDT), LSIMLAMB (INLCLSIM)
                            CRGLAMB (INLCCOMR).
INLCLARG ... library entry argument
             How to locate: address contained in LRPLARG for
                            request on member level; entry in
                            stow table (pointed to by LRPLSTTB)
                            starting after stow table header
                            INLCSTOH.
INLCLRPL ... library request parameter list
             How to locate: at entry of Level-2 modules, address
                            is contained in Register 1.
INLCLSIM ... parameter list for module INLPLSIM
             How to locate: at entry of INLPLSIM, address
                            is contained in Register 1.
INLCNTPT ... note/point information record
             How to locate: address is contained in LRPLARG for
                            INLMNOTE/INLMPOIN requests.
INLCPIDT ... parameter list for module INLPPIDT
             How to locate: at entry of INLPPIDT, address
                            is contained in Register 1.
INLCRESN ... resource name argument
             How to locate: at entry of INLPRESN, address
                            is contained in Register 1.
INLCSTOH ... stow table header
             How to locate: address is contained in LRPLSTTB.
LBRACCDS ... parameter list for module IJBLBHLS
             How to locate: at entry of IJBLBHLS, address
                            is contained in Register 1.
LBRLCTDS ... parameter list for module IJBLBLLS
             How to locate: at entry of IJBLBLLS, address
                            is contained in Register 1.
LIBINFO .... library information pointers
             How to locate: address is contained in LRPLINFO
                            (of data area INLCLRPL), LBRLINFO
                            (LBRLCTDS), LBRINFO (LBRACCDS); also
                            LACBINFO.
TOLCRQL .... parameter list for module IJBCTUPD
             How to locate: at entry of IJBCTUPD, address
                            is contained in Register 1.
UPDATMAP.... parameter list for module IJBLBULT
             How to locate: at entry of IJBLBULT, address
                            is contained in Register 1.
```

7. Data areas used as buffer control blocks (usually in the partition space):

```
BDB ........ tape buffer definition block
             How to locate: the address of the current BDB is
                            stored in BDBPTR which is located
                            immediately after data area INLCBRCR in
                            storage.  All BDBs are contained in a
                            circular chain, the address of the next
                            BDB is contained in BDBBDB (see data
                            area BDB).
BLKHDR ..... tape buffer block header
             How to locate: its address is contained in BDBBFR
                            (see data area BDB).
CCB ........ channel command block
             How to locate: pointed to by BIORB (data area INLCBUCB)
                            or pointed to by BDBCCB (data area BDB)
                            for tape I/O.
CCW ........ channel command word
             How to locate: entry of CCW-list, addressed by BCCWL
                            (data area INLCBUCB) of pointed to by
                            BDBCCW (data area BDB) for tape I/O.
INLCBHDR ... disk buffer header
             How to locate: entries in buffer queues with anchors
                            BRQUEANC, BIQUEANC, BSQUEANC, BFQUEANC,
                            BUQUEANC, BPQUEANC (all in data area
                            INLCBUCB), chaining pointers BIQUE,
                            BSQUE, BRQUE, BFQUE, BUQUE, BPQUE (all
                            in data area (INLCBHDR).
INLCBUCB ... disk buffer control block
             How to locate: address contained in LAMBSBUF,
                            LRPLPBUF.
```

8. Communication regions for Level 3 services:

```
INLCBRCR ... Backup/Restore communication region
             How to locate: data area is contained in module
                            INLPRABF of phase LIBR. Its start is
                            indicated by the string 'BRCR'.
INLCCMDP ... command parameter block
             How to locate: string 'CMDP' in module INLPAREA
                            (about X'1000' after partition begin)
                            indicates start of INLCCMDP. It is
                            immediately preceded by data area
                            INLCCOMR.
INLCCOMR ... Level 3 communication region
             How to locate: string 'COMR' in module INLPAREA
                            (about X'1000' after partition begin)
                            indicates start of INLCCOMR.
INLCPARB ... call interface parameter block
             How to locate: when using Librarian Call I/F at entry
                            of module INLPMAIN Register 1 contains
                            a pointer to the address of INLCPARB.
                            The address is also stored in LAMBEXTL
                            (see data area INLCLAMB).
INLCSCAN ... scan item description (parser)
             How to locate: string 'SCAN' in module INLPAREA
                            (about X'1000' after partition begin)
                            indicates start of INLCSCAN.  It is
                            immediately preceded by data area
                            INLCCMDP.
INLTCENT ... command table entry (parser)
INLTKEY .... keyword parameter description (parser)
INLTNINF ... positional parameter value description (parser)
INLTPOS .... positional parameter description (parser)
INLTSEP .... separator description (parser)
             How to locate: the parser tables INLTCENT, INLTKEY,
                            INLTNINF, INLTPOS, AND INLTSEP are
                            located at partition start + X'C508'.
```

BDB

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	56	BDB	BUFFER DEFINITION BLOCK
0	(0)	CHARACTER	4	BDBID	HEADER IDENTIFIER
4	(4)	A-ADDRESS	4	BDBBDB	POINTER TO NEXT BDB: THE BDBBDB OF THE LAST BDB FOR WHICH A BUFFER IS ALLOCATED POINTS TO THE FIRST BDB
8	(8)	A-ADDRESS	4	BDBBFR	ADDR(ASSOCIATED BUFFER)
12	(C)	A-ADDRESS	4	BDBBLEN	LENGTH(ASSOCIATED BUFFER)
16	(10)	A-ADDRESS	4	BDBCCB	ADDR(ASSOCIATED CCB)
20	(14)	A-ADDRESS	4	BDBCCW	ADDR(ASSOCIATED CCW)
24	(18)	A-ADDRESS	4		RESERVED
28	(1C)	A-ADDRESS	4		RESERVED
32	(20)	CHARACTER	24	BDBIOT	I/O REQUEST BLOCK(IORB) NOTE: THIS REQUEST BLOCK MUST BE ON DWORD BDY SO THAT THE FOLLOWING CCW IS ALSO ON DWORD BOUNDARY

Cross Reference:

Name	Hex Offset	Hex Value	Level
BDB	0	(0)	
BDBBDB	4	(4)	
BDBBFR	8	(8)	
BDBBLEN	12	(C)	
BDBCCB	16	(10)	
BDBCCW	20	(14)	
BDBID	0	(0)	
BDBIOT	32	(20)	

BFHDR

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	12	BFHDR	BACKUP FILE HEADER
0	(0)	SIGNED	2	BFHDRLEN	LENGTH OF BACKUP FILE HEADER
2	(2)	CHARACTER	4	BFDESCR	BACKUP FILE TYPE ID: = 'LIBR' IF LIBRARY BACKUP FILE = 'SUBL' IF SUBLIBRARY BACKUP FILE = 'MEMB' IF MEMBER BACKUP FILE
6	(6)	CHARACTER	6		RESERVED

Cross Reference:

Name	Hex Offset	Hex Value	Level
BFDESCR	2	(2)	
BFHDR	0	(0)	
BFHDRLEN	0	(0)	

BKUPFID

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	48	BKUPFID	BACKUP FILE ID
0	(0)	SIGNED	2	BFIDLEN	LENGTH OF BACKUP FILE ID
2	(2)	CHARACTER	1		RESERVED
3	(3)	UNSIGNED	1	BFIDLEVL	B/R LEVEL
4	(4)	SIGNED	4	BFIDBLNO	BLOCK NUMBER
8	(8)	CHARACTER	4	BFIDDESR	BACKUP FILE ID 'BFID'
12	(C)	CHARACTER	16	BFID	BACKUP FILE IDENTIFICATION (OPTIONAL)
28	(1C)	SIGNED	4	BFIDN	MAX # OF BUFFERS WHICH MAY BE USED BY THE TAPE INPUT ROUTINE
32	(20)	SIGNED	4	BFIDLNO	# OF ELEMENTS OF THE SPECIFICATION LIST DURING BACKUP
36	(24)	CHARACTER	12		RESERVED

Cross Reference:

Name	Hex Offset	Hex Value	Level
BFID	12	(C)	
BFIDBLNO	4	(4)	
BFIDDESR	8	(8)	
BFIDLEN	0	(0)	
BFIDLEVL	3	(3)	
BFIDLNO	32	(20)	
BFIDN	28	(1C)	
BKUPFID	0	(0)	

BLDCBMAP

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	24	BLDCBMAP	MAP OF CONTROL BLOCK LBRBLDCB
0	(0)	A-ADDRESS	4	BLDCBLDT	LDTFLD
4	(4)	A-ADDRESS	4	BLDCBEXT	EXTENTS
8	(8)	A-ADDRESS	4	BLDCBDEV	DEVCHAR
12	(C)	A-ADDRESS	4	BLDCBLMB	LAMB
16	(10)	A-ADDRESS	4	BLDCBFLG	FLAGS
		11..		BLDCBTYP	TYPEFLE: 10-INPUT 01-OUTPUT
		..11 1...		BLDCBMF	MF: 100-NEWLIB 010-EXTEND 001-EXTUPD
		.... .1..		BLDCBREP	REPLACE: 1-YES 0-NO
		11			RESERVED
17	(11)	UNSIGNED	1		RESERVED
18	(12)	BITSTRING	2		RESERVED
20	(14)	A-ADDRESS	4	BLDCBRIN	RETURN INFO.IN CASE OF FAILURE
20	(14)	UNSIGNED	2	BLDCBRI1	
20	(14)	UNSIGNED	1	BLDCBRCO	ORIGIN OF RETURN CODE
21	(15)	UNSIGNED	1	BLDCBRCV	VALUE OF RETURN CODE
22	(16)	UNSIGNED	2		RESERVED

Cross Reference:

Name	Hex Offset	Hex Value	Level
BLDCBDEV	8	(8)	
BLDCBEXT	4	(4)	
BLDCBFLG	16	(10)	
BLDCBLDT	0	(0)	
BLDCBLMB	12	(C)	
BLDCBMAP	0	(0)	
BLDCBMF	16	X'38'	
BLDCBRCO	20	(14)	
BLDCBRCV	21	(15)	
BLDCBREP	16	X'04'	
BLDCBRIN	20	(14)	
BLDCBRI1	20	(14)	
BLDCBTYP	16	X'C0'	

BLKHDR

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	16	BLKHDR	TAPE BUFFER BLOCK HEADER
0	(0)	SIGNED	2	BLKLEN	BLOCK LENGTH
2	(2)	UNSIGNED	1		RESERVED
3	(3)	UNSIGNED	1	TLAST	INDICATES LAST BLK OF LIB (OLDLIB)
4	(4)	CHARACTER	8	BLKID	
4	(4)	SIGNED	4	BLKNO	BLOCK NUMBER
8	(8)	CHARACTER	4	DESCR	TAPE BLOCK DESCRIPTOR: = 'FHDR' IF BACKUP FILE HEADER = 'DATA' IF REGULAR BACKUP FILE BLOCK = 'DRID' IF DUMMY TAPE RECORD = 'ERR ' IF ERROR RECORD (IF ABNORMAL END OF BACKUP) = 'EOB ' IF END-OF-BACKUP RECORD = 'EOV ' IF END-OF-VOLUME RECORD
12	(C)	CHARACTER	4	INLID	USED FOR EOV RECORD

Cross Reference:

Name	Hex Offset	Hex Value	Level
BLKHDR	0	(0)	
BLKID	4	(4)	
BLKLEN	0	(0)	
BLKNO	4	(4)	
DESCR	8	(8)	
INLID	12	(C)	
TLAST	3	(3)	

DIPLPHDR

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	18	DIPLPHDR	DISK IPL PHASE HEADER ON BACKUP TAPE
0	(0)	SIGNED	2	DIPLLEN	LENGTH OF DISK IPL PHASE HEADER PLUS LENGTH OF IPL PHASE
2	(2)	CHARACTER	4	DIPLDESR	DISK IPL PHASE HEADER ID: = 'IPLK' IF CKD DISK IPL PHASE = 'IPLF' IF FBA DISK IPL PHASE
6	(6)	CHARACTER	12	DIPLBUFF	RESERVED FOR INLPRIPL INTERFACE
18	(12)	CHARACTER	0	DIPLPHSE	DISK IPL PHASE

Cross Reference:

Name	Hex Offset	Hex Value	Level
DIPLBUFF	6	(6)	
DIPLDESR	2	(2)	
DIPLLEN	0	(0)	
DIPLPHDR	0	(0)	
DIPLPHSE	18	(12)	

EOBKFID

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	6	EOBKFID	END-OF-BACKUP-FILE ID
0	(0)	SIGNED	2	EOBFLEN	LENGTH OF END-OF-BACKUP-FILE ID
2	(2)	CHARACTER	4	EOBFDESR	END-OF-BACKUP FILE ID = 'EOBF'

Cross Reference:

Name	Hex Offset	Hex Value	Level
EOBFDESR	2	(2)	
EOBFLEN	0	(0)	
EOBKFID	0	(0)	

EXTENTR

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	72	EXTENTR	EXTENT INFORMATION RECORD
0	(0)	SIGNED	2	XTNTLEN	LENGTH OF EXTENT INFORMATION RECORD
2	(2)	CHARACTER	4	XTNTDESR	EXTENT INFORMATION ID: 'XTNT'
6	(6)	CHARACTER	10		RESERVED
16	(10)	CHARACTER	3		RESERVED
19	(13)	CHARACTER	1	XTNTFLAG	EXTENT FLAG BYTE
20	(14)	UNSIGNED	4	XTNTSTRX	START OF EXTENT
20	(14)	UNSIGNED	2	XTNTSTRC	START OF EXTENT CC
22	(16)	UNSIGNED	2	XTNTSTRH	START OF EXTENT HH
24	(18)	UNSIGNED	4	XTNTENDX	END OF EXTENT
24	(18)	UNSIGNED	2	XTNTENDC	END OF EXTENT CC
26	(1A)	UNSIGNED	2	XTNTENDH	END OF EXTENT HH
28	(1C)	SIGNED	4	XTNTNTRK	# OF TRACKS/EXTENT OR # OF PBN/EXTENT
32	(20)	SIGNED	4		RESERVED
36	(24)	CHARACTER	6	XTNTVLID	VOLID
42	(2A)	CHARACTER	2		RESERVED
44	(2C)	UNSIGNED	4	XTNTNEXT	X'FFFFFFFF' IF LAST EXTENT
48	(30)	CHARACTER	24	XTNTDDTX	DEVICE DEFINITION INFORMATION
48	(30)	BITSTRING	1	XTNTTYP	RPS DEVICE TYPE
49	(31)	BITSTRING	1	XTNTDEV	DEVICE INDICATOR
50	(32)	BITSTRING	1		RESERVED
51	(33)	BITSTRING	1		RESERVED
52	(34)	SIGNED	4	XTNTBLZ	FBA BLOCKSIZE
52	(34)	SIGNED	2	XTNTTRCY	# OF TRACKS/CYLINDER
54	(36)	SIGNED	2	XTNTBLTR	# OF LBS/TRACK
56	(38)	CHARACTER	12		RESERVED
68	(44)	SIGNED	4	XTNTIDEN	RESERVED

Cross Reference:

Name	Hex Offset	Hex Value	Level
EXTENTR	0	(0)	
XTNTBLTR	54	(36)	
XTNTBLZ	52	(34)	
XTNTDDTX	48	(30)	
XTNTDESR	2	(2)	
XTNTDEV	49	(31)	
XTNTENDC	24	(18)	
XTNTENDH	26	(1A)	
XTNTENDX	24	(18)	
XTNTFLAG	19	(13)	
XTNTIDEN	68	(44)	
XTNTLEN	0	(0)	
XTNTNEXT	44	(2C)	
XTNTNTRK	28	(1C)	
XTNTSTRC	20	(14)	
XTNTSTRH	22	(16)	
XTNTSTRX	20	(14)	
XTNTTRCY	52	(34)	
XTNTTYP	48	(30)	
XTNTVLID	36	(24)	

INLCBHDR

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	64	INLCBHDR	BUFFER HEADER
0	(0)	CHARACTER	4	BHDRID	CONTROL BLOCK ID
4	(4)	SIGNED	4	BHDRSQ	SEQUENCE NUMBER IN LIST
8	(8)	CHARACTER	1	BIOCB	IO CONTROL INFORMATION
9	(9)	BITSTRING	1		
		1...		BIOFBA	FBA DEVICE
		.1..		BIOCKD	CKD DEVICE
		..1.		BIORPS	CKD DEVICE WITH RPS
		...1		BIOECKD	ECKD DEVICE
10	(A)	BITSTRING	2		
		1...		BUPDT	CONTENT OF BUFFER CHANGED
		.1..		B1THINEX	1TH LB IN EXTENT
		..1.		FORCEWRT	FORCE WRITE ON MAKE PREEMPT.
		...1		DELAYWRT	DELAY WRITE ON MAKE PREEMPT.
12	(C)	CHARACTER	2	BCUU	CUU OF DEVICE
14	(E)	CHARACTER	8	BDSKAD	DISK ADDRESS
14	(E)	UNSIGNED	2	BBB	BB
16	(10)	UNSIGNED	2	BCC	CC
18	(12)	UNSIGNED	2	BHH	HH
20	(14)	UNSIGNED	1	BRR	R
22	(16)	CHARACTER	6	BPRBA	PRBA 1TH RECORD IN BUFFER
22	(16)	SIGNED	2	BPRBAO	OFFSET IN LB
24	(18)	SIGNED	4	BPRBAP	REL.BLOCK NUMBER IN LIBRARY
28	(1C)	A-ADDRESS	4	BLRBA	LRBA BEGIN OF DATA
32	(20)	A-ADDRESS	4	BEXT	EXTENT POINTER
36	(24)	A-ADDRESS	4	BIQUE	I/O REQUEST LIST
40	(28)	A-ADDRESS	4	BSQUE	SORTED I/O REQUEST LIST
44	(2C)	A-ADDRESS	4	BRQUE	REQUEST LIST PTR
48	(30)	A-ADDRESS	4	BFQUE	FREE QUEUE PTR
52	(34)	A-ADDRESS	4	BUQUE	IN-USE QUEUE PTR
56	(38)	A-ADDRESS	4	BPQUE	PRE-EMPTABLE QUEUE PTR
60	(3C)	A-ADDRESS	4	BDATA	DATA AREA

Cross Reference:

Name	Hex Offset	Hex Value	Level
BBB	14	(E)	
BCC	16	(10)	
BCUU	12	(C)	
BDATA	60	(3C)	
BDSKAD	14	(E)	
BEXT	32	(20)	
BFQUE	48	(30)	
BHDRID	0	(0)	
BHDRSQ	4	(4)	
BHH	18	(12)	
BIOCB	8	(8)	
BIOCKD	9	X'40'	
BIOECKD	9	X'10'	
BIOFBA	9	X'80'	
BIORPS	9	X'20'	
BIQUE	36	(24)	
BLRBA	28	(1C)	
BPQUE	56	(38)	
BPRBA	22	(16)	
BPRBAO	22	(16)	
BPRBAP	24	(18)	
BRQUE	44	(2C)	
BRR	20	(14)	
BSQUE	40	(28)	
BUPDT	10	X'80'	
BUQUE	52	(34)	
B1THINEX	10	X'40'	
DELAYWRT	10	X'10'	
FORCEWRT	10	X'20'	
INLCBHDR	0	(0)	

INLCBIGM

VIF FOR BIGM

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	20	INLCBIGM	
0	(0)	SIGNED	2	BIGMELEN	LENGTH OF V.I.F.
2	(2)	BITSTRING	2	BIGMEATR	RESERVED
		1...		BIGMVIF	VIF FOLLOWS
4	(4)	CHARACTER	4	BIGMEID	V.I.F. IDENTIFIER
8	(8)	SIGNED	4	BIGMBLK	# OF LB
12	(C)	CHARACTER	8	*	RESERVED

INLCBRCR

BACKUP/RESTORE COMMUNICATION REGION

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	244	INLCBRCR	
0	(0)	CHARACTER	244	BRCOMRG	B/R COMREG AREA
0	(0)	CHARACTER	4	BRCRID	B/R COMREG IDENTIFIER
4	(4)	ADDRESS	4	BDBPTR	ADDRESS OF BDB
					THE BDBPTR AND THE BDB-LOOP IS NOT USED BY RESTORE OLDLIB. FOR RESTORE OLDLIB ONLY 2 BUFFERS ARE USED POINTED TO BY ABUF1 AND ABUF2. (SEE MODULE INLPROLD)
8	(8)	ADDRESS	4	BUFPTR	PTR INTO CURRENT BUFFER
12	(C)	ADDRESS	4	BUFEND	END ADDR OF CURRENT BUFFER
16	(10)	SIGNED	4	TBUFLEN	TAPE BUFFER LENGTH
20	(14)	ADDRESS	4	BDBSTPTR	PTR TO FIRST BDB ALLOCATED
24	(18)	ADDRESS	4	STORBEG	BEGIN OF WORK AREA STORAGE
28	(1C)	ADDRESS	4	N	# OF TAPE BUFFERS ALLOCATED
32	(20)	ADDRESS	4	NMAX	MAX # OF TAPE BUFFERS WHICH MAY BE ALLOCATED
36	(24)	ADDRESS	4	LAMBPTR	ADDRESS OF LAMB
40	(28)	ADDRESS	4	LDTPTR	POINTER TO LDT
44	(2C)	ADDRESS	4	WRDRCP	ADDR(WRITE DUMMY REC'S CP)
48	(30)	ADDRESS	4	*	RESERVED
52	(34)	ADDRESS	4	SHRBFPTR	PTR TO SHARED DISK BUFFERS
56	(38)	ADDRESS	4	PRVBFPTR	PTR TO PRIVATE DISK BUFFERS
60	(3C)	ADDRESS	4	SHRBFLEN	SHARD DISK BUFFER AREA LENGTH
64	(40)	ADDRESS	4	PRVBFLEN	PRIV DISK BUFFER AREA LENGTH
68	(44)	ADDRESS	4	WAPTR	PTR TO WORK AREA FOR LBRACCES
72	(48)	ADDRESS	4	*	RESERVED
76	(4C)	ADDRESS	4	BRWAPTR	PTR TO B/R WORK AREA
80	(50)	ADDRESS	4	BRWALEN	B/R WORK AREA LENGTH
84	(54)	ADDRESS	4	LISTPTR	POINTER TO NAME LIST
88	(58)	ADDRESS	4	ILISTPTR	PTR TO SPEC INDEX LIST (REST)
92	(5C)	SIGNED	4	BLKNBR	SEQ. BLOCK NR ON TAPE FOR A BACKUP FILE ACROSS VOLUMES
96	(60)	ADDRESS	4	ILISTLEN	LEN(SPEC. INDEX LIST)
100	(64)	ADDRESS	4	LDESPTR	PTR TO LIBR DESCRIPTOR RECORD
104	(68)	ADDRESS	4	SLXEPTR	PTR TO SUBLIB INDEX ENTRY
108	(6C)	ADDRESS	4	ARGPTR	POINTER TO MEMBER ARGUMENT
112	(70)	ADDRESS	4	DEPTR	POINTER TO MEMBER DIR ENTRY
116	(74)	ADDRESS	4	UPHAPTR	POINTER TO V.I.F.
120	(78)	ADDRESS	4	SVASAPTR	PTR TO AREA FOR LOADING SA LOAD BOOK $SVASA
124	(7C)	ADDRESS	4	SLTABPTR	PTR TO SL INDEX ENTRY TABLE
128	(80)	ADDRESS	4	SLTABLEN	LENGTH(SL INDEX ENTRY TABLE)
132	(84)	ADDRESS	4	STOHPTR	PTR TO STOW TABLE HDR
136	(88)	ADDRESS	4	STOARLEN	STOW TABLE AREA LENGTH
140	(8C)	ADDRESS	4	STOENPTR	PTR TO STOW TAB ENTRY
144	(90)	ADDRESS	4	NSTOW	NUMBER OF ENTRIES IN STOW TAB
148	(94)	ADDRESS	4	NPROCD	NUMBER OF LIB'S,SUBLIB'S OR MEMBERS OF NAMELIST PROCESSED
152	(98)	ADDRESS	4	CRLRPL	B/R RPL ADDR
156	(9C)	SIGNED	4	CRRETC	B/R RETURN CODE
160	(A0)	ADDRESS	4	*	RESERVED
164	(A4)	CHARACTER	7	ACLIBNAM	LIB NAME FROM ACCESS FOR RESTORE MEMBER OR LIBRARY CURRENTLY IN PROCESS
171	(AB)	CHARACTER	8	ACSUBNAM	SUBLIB NAME FROM ACCESS FOR RESTORE MEMBER OR SUBLIB CURRENTLY IN PROCESS
179	(B3)	BITSTRING	1	BRSW1	SWITCH BYTE 1
		1...		RSALIBR	LIBRARY RESTORE IF RESTORE

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
		.1..		RSASUBL	SUBLIBR RESTORE IF RESTORE
		..1.		RSAMEMB	MBR REST IF RESTORE
		...1		EOBF	END OF BACKUP FILE INDICATOR
		.... 1...		EOLIB	END OF LIBRARY INDICATOR
		.... .1..		EOSUBL	END OF SUBLIB INDICATOR
		.... ..1.		EOMBR	END OF MEMBER INDICATOR
		1		EODE	END OF DIR ENTRY INDICATOR
180	(B4)	BITSTRING	1	BRSW2	SWITCH BYTE 2
		1...		NOACCFLG	'NO ACCESS ALLOWED' SET IND.
		.1..		SKMBRFLG	SKIP MEMBER INDICATOR - USED FOR RESTORE OLDLIB ONLY
		..1.		EOSCAN	END OF SCAN FOR ID=
		...1		EMPTYMBR	EMPTY MEMBER SWITCH
		.... 1...		LIBFOUND	LIBRARY FOUND ON TAPE TO REST
		.... .1..		SLFOUND	SUBLIB TO RESTORE
		.... ..1.		MBRFOUND	MEMBER TO RESTORE
		1		HFFOUND	HISTORY FILE FOUND ON TAPE
181	(B5)	BITSTRING	1	BRSW3	SWITCH BYTE 3
		1...		PIDV2SW	PID-V2-STACKED TAPE DETECTED DURING RESTORE
		.1..		TDEV9346	9346 TAPE DEVICE
		..1.		*	RESERVED
		...1		GENERIC	CURRENT MEMBER SPECIFICATION GENERIC
		.... 1...		SGENERIC	MEMBER SPECIFICATION PROCESSED LAST GENERIC
		.... .1..		MBRTAPE	RESTORE FUNCTION: BACKUP FILE CURRENTLY PROCESSED CONTAINS ONLY MEMBERS
		11		*	RESERVED
182	(B6)	CHARACTER	2	*	RESERVED
184	(B8)	CHARACTER	1	BRIND	R: RESTORE B: BACKUP
185	(B9)	CHARACTER	1	MSHPLU	START OF PARM FIELD FOR PTFBKUP = 'X' (+ LOG UNIT)
186	(BA)	CHARACTER	2	BRLU	LOG UNIT USED FOR TAPE I/O
188	(BC)	CHARACTER	7	TOLIBNAM	RESTORE: TARGET LIBRARY NAME
195	(C3)	CHARACTER	1	*	RESERVED
196	(C4)	CHARACTER	8	TOSUBNAM	RESTORE: TARGET SUBLIB NAME
204	(CC)	CHARACTER	32	*	RESERVED
236	(EC)	CHARACTER	8	INSERT	MESSAGE INSERTION INFO
236	(EC)	UNSIGNED	1	INSLNGTH	LENGTH OF TEXT
237	(ED)	CHARACTER	7	INSTEXT	INSERT COMMAND NAME

Cross Reference

Name	Hex Offset	Hex Value	Level
ACLIBNAM	A4		3
ACSUBNAM	AB		3
ARGPTR	6C		3
BDBPTR	4		3
BDBSTPTR	14		3
BLKNBR	5C		3
BRCOMRG	0		2
BRCRID	0		3
BRIND	B8		3
BRLU	BA		3
BRSW1	B3		3
BRSW2	B4		3
BRSW3	B5		3
BRWALEN	50		3
BRWAPTR	4C		3
BUFEND	C		3
BUFPTR	8		3
CRLRPL	98		3
CRRETC	9C		3

Name	Hex Offset	Hex Value	Level
DEPTR	70		3
EMPTYMBR	B4	10	4
EOBF	B3	10	4
EODE	B3	01	4
EOLIB	B3	08	4
EOMBR	B3	02	4
EOSCAN	B4	20	4
EOSUBL	B3	04	4
GENERIC	B5	10	4
HFFOUND	B4	01	4
ILISTLEN	60		3
ILISTPTR	58		3
INLCBRCR	0		1
INSERT	EC		3
INSLNGTH	EC		4
INSTEXT	ED		4
LAMBPTR	24		3
LDESPTR	64		3
LDTPTR	28		3
LIBFOUND	B4	08	4
LISTPTR	54		3
MBRFOUND	B4	02	4
MBRTAPE	B5	04	4
MSHPLU	B9		3
N	1C		3
NMAX	20		3
NOACCFLG	B4	80	4
NPROCD	94		3
NSTOW	90		3
PIDV2SW	B5	80	4
PRVBFLEN	40		3
PRVBFPTR	38		3
RSALIBR	B3	80	4
RSAMEMB	B3	20	4
RSASUBL	B3	40	4
SGENERIC	B5	08	4
SHRBFLEN	3C		3
SHRBFPTR	34		3
SKMBRFLG	B4	40	4
SLFOUND	B4	04	4
SLTABLEN	80		3
SLTABPTR	7C		3
SLXEPTR	68		3
STOARLEN	88		3
STOENPTR	8C		3
STOHPTR	84		3
STORBEG	18		3
SVASAPTR	78		3
TBUFLEN	10		3
TDEV9346	B5	40	4
TOLIBNAM	BC		3
TOSUBNAM	C4		3
UPHAPTR	74		3
WAPTR	44		3
WRDRCP	2C		3

INLCBUCB

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	168	INLCBUCB	BUFFERS CONTROL
0	(0)	CHARACTER	4	BUCBID	CONTROL BLOCK ID
4	(4)	ADDRESS	4	BCCWL	CCW-LIST ANCHOR
8	(8)	SIGNED	4	BCCELM	CCW-LIST HI INDEX
12	(C)	SIGNED	4	BNO	NO.OF ALLOCATED BUFFERS
16	(10)	SIGNED	4	BLEN	LENGTH OF EACH BUFFER
20	(14)	SIGNED	4	LBLRBA	LOWEST LRBA IN BUFFER
24	(18)	SIGNED	4	HBLRBA	HIGHEST LRBA IN BUFFER
28	(1C)	ADDRESS	4	BRQUEANC	BUF HEADER_LIST ANCHOR
32	(20)	ADDRESS	4	BIQUEANC	I/O LIST ANCHOR
36	(24)	ADDRESS	4	BSQUEANC	I/O SORTED LIST ANCHOR
40	(28)	ADDRESS	4	BFQUEANC	FREE QUEUE ANCHOR
44	(2C)	ADDRESS	4	BUQUEANC	IN-USE QUEUE ANCHOR
48	(30)	ADDRESS	4	BPQUEANC	PRE-EMPT QUEUE ANCHOR
52	(34)	BITSTRING	4	BECB	E C B
56	(38)	CHARACTER	20	BIORB	I O R B
56	(38)	UNSIGNED	2	BIOCNT	RES. COUNT
58	(3A)	CHARACTER	2	BIOXMIT	TRANSMISSION BYTE
58	(3A)	BITSTRING	1	*	
59	(3B)	.... .1..		BIONRCFD	NO REC. FOUND COND.@D14LDFB
		11		*	
60	(3C)	CHARACTER	2	*	
62	(3E)	BITSTRING	1	BIOTYP	TYPE
		1111		*	
		.... 1...		BIOPHYS	PHYSICAL ADDRESSING
		.... .111		*	
63	(3F)	CHARACTER	1	BIOPUB	ADDRESSING BYTE
64	(40)	CHARACTER	1	*	
65	(41)	ADDRESS	3	BIOCCWA	CCW ADDRESS
68	(44)	CHARACTER	1	*	
69	(45)	ADDRESS	3	*	
72	(48)	BITSTRING	1	BIOFF	FIX FLAG
73	(49)	ADDRESS	3	BIOFLST	FIX LIST ADDR.
76	(4C)	SIGNED	2	BCMPRL	DECMPR AREA LEN
78	(4E)	ADDRESS	4	BCMPRP	ECMPR AREA PTR
82	(52)	CHARACTER	6	*	RESERVED
88	(58)	CHARACTER	8	BSEEKARG	
88	(58)	UNSIGNED	2	BSEEKBB	ARGUMENT FOR 1TH SEEK
90	(5A)	UNSIGNED	2	BSEEKCC	
92	(5C)	UNSIGNED	2	BSEEKHH	
94	(5E)	UNSIGNED	1	BSEEKRR	
95	(5F)	CHARACTER	1	*	
96	(60)	CHARACTER	32	INLCFBAH	FBA AND ECKD AREA FOR DEF EXTENT AND LOCATE
96	(60)	CHARACTER	16	FBAHDEFE	DEFINE EXTENT
96	(60)	CHARACTER	1	DEFEMASK	OPERATION MASK
97	(61)	CHARACTER	1	DEFEATTR	ECKD: GLOBAL ATTR. FBA : RESERVED
98	(62)	UNSIGNED	2	DEFEBLKS	ECKD: RECORD SIZE FBA : BLKSIZE
100	(64)	UNSIGNED	4	DEFE1OFF	OFFSET EXT BEGIN
100	(64)	CHARACTER	3	*	ECKD: ZERO
103	(67)	CHARACTER	1	DEFEATTE	ECKD: GLOBAL ATTR. EXTENDED
104	(68)	UNSIGNED	4	DEFELODA	EXTENT BEGIN IN DATASET
108	(6C)	UNSIGNED	4	DEFEHIDA	EXTENT END IN DATASET
112	(70)	CHARACTER	16	LOCA	LOCATE
112	(70)	CHARACTER	8	FBAHLOCA	
112	(70)	CHARACTER	1	LOCAMDOP	MODIFIER/OPERATION
113	(71)	CHARACTER	1	LOCAAUX	ECKD: AUXIL. BYTE FBA : RESERVED
114	(72)	UNSIGNED	2	LOCACNT	ECKD: RECORD COUNT FBA : BLOCK COUNT
116	(74)	UNSIGNED	4	LOCADSOF	ECKD: SEEK ADDRESS FBA : OFFSET IN DS
120	(78)	CHARACTER	5	LOCASRCH	SEARCH ADDRESS
125	(7D)	UNSIGNED	1	LOCASVAL	SECTOR VALUE
126	(7E)	UNSIGNED	2	LOCATLF	TRANSFER LENGTH FACTOR (ALWAYS ZERO)

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
128	(80)	SIGNED	4	NLRBA	NEXT LRBA FOR BUFFER FILL
132	(84)	CHARACTER	6	NPRBA	CORR. NEXT PRBA
132	(84)	UNSIGNED	2	NPRBO	
134	(86)	SIGNED	4	NPRBN	
138	(8A)	SIGNED	2	NCONT	CORR. CONTIG. INDICATOR
140	(8C)	SIGNED	4	PREVLRBA	LRBA OF LAST ACCESS
144	(90)	CHARACTER	6	PREVPRBA	
144	(90)	UNSIGNED	2	PREVPRBO	CORR. PRBA
146	(92)	SIGNED	4	PREVPRBP	
152	(98)	CHARACTER	8	BUCBBREQ	REQ.CONTROL BLK
152	(98)	ADDRESS	4	BUCBBPTR	REQUESTS FOR RQUEU
156	(9C)	ADDRESS	4	BUCBBANC	
160	(A0)	CHARACTER	8	BUCBIREQ	REQ.CONTROL BLK
160	(A0)	ADDRESS	4	BUCBIUCB	REQUESTS FOR RQUEU
164	(A4)	ADDRESS	4	BUCBIANC	

Cross Reference

Name	Hex Offset	Hex Value	Level
BCCELM	8		2
BCCWL	4		2
BCMPRL	4C		2
BCMPRP	4E		2
BECB	34		2
BFQUEANC	28		2
BIOCCWA	41		3
BIOCNT	38		3
BIOFF	48		3
BIOFLST	49		3
BIONRCFD	3B	04	4
BIOPHYS	3E	08	4
BIOPUB	3F		3
BIORB	38		2
BIOTYP	3E		3
BIOXMIT	3A		3
BIQUEANC	20		2
BLEN	10		2
BNO	C		2
BPQUEANC	30		2
BRQUEANC	1C		2
BSEEKARG	58		2
BSEEKBB	58		3
BSEEKCC	5A		3
BSEEKHH	5C		3
BSEEKRR	5E		3
BSQUEANC	24		2
BUCBBANC	9C		3
BUCBBPTR	98		3
BUCBBREQ	98		2
BUCBIANC	A4		3
BUCBID	0		2
BUCBIREQ	A0		2
BUCBIUCB	A0		3
BUQUEANC	2C		2
DEFEATTE	67		5
DEFEATTR	61		4
DEFEBLKS	62		4
DEFEHIDA	6C		4
DEFELODA	68		4
DEFEMASK	60		4
DEFE1OFF	64		4
FBAHDEFE	60		3
FBAHLOCA	70		4

Name	Hex Offset	Hex Value	Level
HBLRBA	18		2
INLCBUCB	0		1
INLCFBAH	60		2
LBLRBA	14		2
LOCA	70		3
LOCAAUX	71		5
LOCACNT	72		5
LOCADSOF	74		5
LOCAMDOP	70		5
LOCASRCH	78		4
LOCASVAL	7D		4
LOCATLF	7E		4
NCONT	8A		2
NLRBA	80		2
NPRBA	84		2
NPRBN	86		3
NPRBO	84		3
PREVLRBA	8C		2
PREVPRBA	90		2
PREVPRBO	90		3
PREVPRBP	92		3

INLCCMDP

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	247	INLCCMDP	
INTERNAL REPRESENTATION OF AN EXTERNAL COMMAND					
0	(0)	CHARACTER	4	CMDPID	HEADER IDENTIFIER
4	(4)	CHARACTER	1	INLCVER	VERSION/MODIFICATION LEVEL
5	(5)	CHARACTER	7	INLCCMD	FULL COMMAND NAME
12	(C)	ADDRESS	4	INLCALST	POINTER TO NAME LIST
16	(10)	SIGNED	2	INLCNO	NO. OF NAME LIST ENTRIES
18	(12)	CHARACTER	10	INLCPARM	PARAMETER FIELDS -
18	(12)	CHARACTER	2	INLCEODV	CATALOG EOD PARAMETER
20	(14)	UNSIGNED	2	INLCSEQV	UPDATE SEQ PARAMETER
22	(16)	CHARACTER	2	INLCUNV	UNIT PARAMETER
24	(18)	CHARACTER	4	INLCCOLV	UPDATE COLUMN PARAMETER
24	(18)	UNSIGNED	2	INLCCOLS	COLUMN START
26	(1A)	UNSIGNED	2	INLCCOLE	COLUMN END
28	(1C)	CHARACTER	1	INLCGENP	GENERAL PURPOSE BYTE
29	(1D)	CHARACTER	3	*	RESERVED
COMMAND PARAMETER FLAGBYTES					
32	(20)	BITSTRING	1	INLCACC	ACCESS FLAGBYTE
		1...		INLCADIS	DISPLAY OPTION (1=YES,0=NO)
		.111 1111		*	RESERVED
33	(21)	BITSTRING	1	INLCCAT	CATALOG FLAGBYTE
		1...		INLCDATA	DATA OPTION (1=YES,0=NO)
		.1..		INLCREP	REPLACE OPTION (1=YES,0=NO)
		..1.		INLCEOD	EOD PARAMETER (1=YES,0=NO)
		...1 1111		*	RESERVED
34	(22)	BITSTRING	1	INLCCOP	COPY/COMPARE/MOVE FLAGBYTE
		1...		INLCCOLB	LIB FUNCTION (1=YES,0=NO)
		.1..		INLCCOSB	SUBLIB FUNCT. (1=YES,0=NO)
		..1.		INLCCOMB	MEMBER FUNCT. (1=YES,0=NO)
		...1		INLCCORP	REPLACE OPTION (1=YES,0=NO)
		.... 1...		INLCCOLI	LIST OPTION (1=YES,0=NO)
		.... .1..		INLCCODA	DATA OPTION (1=YES,0=NO)
		.... ..1.		INLCCOPU	PUNCH OPTION (1=YES,0=NO)
		1		*	RESERVED
35	(23)	BITSTRING	1	INLCDEF	DEFINE FLAGBYTE
		1...		INLCDFL	DEFINE LIBRARY (1=YES,0=NO)
		.1..		INLCDFS	DEFINE SUBLIB (1=YES,0=NO)
		..1.		INLCDFRA	REUSE AUTOMATIC(1=YES,0=NO)
		...1		INLCDFRI	REUSE IMMEDIATE(1=YES,0=NO)
		.... 1...		INLCDFRP	REPLACE PARAM. (1=YES,0=NO)
		.... .111		*	RESERVED
36	(24)	BITSTRING	1	INLCDEL	DELETE FLAGBYTE
		1...		INLCDLL	DELETE LIBRARY (1=YES,0=NO)
		.1..		INLCDLS	DELETE SUBLIB (1=YES,0=NO)
		..1.		INLCDLM	DELETE MEMBER (1=YES,0=NO)
		...1 1111		*	RESERVED
37	(25)	BITSTRING	1	INLCLIP	LIST/PUNCH FLAGBYTE 1
		1...		INLCLI	LIST (1=YES,0=NO)
		.1..		INLCLIX	LISTX (1=YES,0=NO)
		..1.		INLCPUN	PUNCH (1=YES,0=NO)
		...1		INLCFORM	FORMAT OPTION (1=OLD,0=NEW)
		.... 1...		INLCLPLG	UNIT=SYSLOG (1=YES,0=NO)
		.... .1..		INLCLPLS	UNIT=SYSLST (1=YES,0=NO)
		.... ..1.		INLCPEOF	EOF OPTION (1=SUPPRESS, 0=PUNCH EOF
		1		INLCPNOH	FORMAT=NOHEADER(1=YES,0=NO)
38	(26)	BITSTRING	1	INLCCONN	CONNECT FLAGBYTE
		1...		INLCMDIS	DISPLAY OPTION (1=YES,0=NO)
		.111 1111		*	RESERVED
39	(27)	BITSTRING	2	INLCLID	LISTDIR/TEST FLAGBYTES
		1...		INLCLDL	LD/TEST LIBR. (1=YES,0=NO)

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
		.1..		INLCLDS	LD/TEST SUBLIB (1=YES,0=NO)
		..1.		INLCLDM	LD/TEST MEMBER (1=YES,0=NO)
		...1		INLCSDL	LISTDIR SDL (1=YES,0=NO)
		.... 1...		INLCLDLG	UNIT=SYSLOG (1=YES,0=NO)
		.... .1..		INLCLDLS	UNIT=SYSLST (1=YES,0=NO)
		.... ..1.		INLCLDFN	OUTPUT PARAM. (1=FULL 0=NORMAL) AREA PARAM. (1=SPACE 0=ALL)
		1		INLCLDSH	OUTPUT PARAM. (1=SHORT)
40	(28)	1...		INLCLDST	OUTPUT PARAM. (1=STATUS)
		.1..		INLCTSRP	TEST-REPAIR PARAM. (1=YES 0=NO)
		..1.		INLCLDPH	PHASE PARAM. (1=YES 0=NO)
		...1		INLCBIGL	BIGLIB PARAM. (1=YES 0=NO)
		 1111		*	RESERVED
41	(29)	BITSTRING	1	INLCREN	RENAME FLAGBYTE
		1...		INLCRNS	RENAME SUBLIB (1=YES,0=NO)
		.1..		INLCRNM	RENAME MEMBER (1=YES,0=NO)
		..11 1111		*	RESERVED
42	(2A)	BITSTRING	1	INLCBKP	BACKUP FLAGBYTE 1
		1...		INLCBKL	BACKUP LIBR. (1=YES,0=NO)
		.1..		INLCBKS	BACKUP SUBLIB (1=YES,0=NO)
		..1.		INLCBKHD	HEADER PARAM. (1=YES,0=NO)
		...1		INLCBKOF	RESTORE OPTION (1=STANDALONE 0=ONLINE)
		.... 1...		INLCBKLU	LOG.UNIT PARM. (1=YES,0=NO)
		.... .1..		INLCBKPU	PHYS.UNIT PARM (1=YES,0=NO)
		.... ..1.		INLCBKHF	HIST.FILE OPT. (1=YES,0=NO)
		1		INLCBKID	ID PARAMETER (1=YES,0=NO)
43	(2B)	BITSTRING	2	INLCRES	RESTORE FLAGBYTES
		1...		INLCRSL	RESTORE LIBR. (1=YES,0=NO)
		.1..		INLCRSS	RESTORE SUBLIB (1=YES,0=NO)
		..1.		INLCRSM	RESTORE MEMBER (1=YES,0=NO)
		...1		INLCRSO	RESTORE OLDLIB (1=YES,0=NO)
		.... 1...		INLCRSA	RESTORE ALL (1=YES,0=NO)
		.... .1..		INLCRSCN	SCAN OPTION (1=YES,0=NO)
		.... ..1.		INLCRSLU	LOG.UNIT PARM. (1=YES,0=NO)
		1		INLCRSPU	PHYS.UNIT PARM (1=YES,0=NO)
44	(2C)	1...		INLCRSID	ID=STRING PARM (1=YES,0=NO)
		.1..		INLCRSIA	ID= PARAMETER (1=YES,0=NO)
		..1.		INLCRSLY	1 - LIST=YES SPECIFIED
		...1		INLCRSLN	1 - LIST=NO SPECIFIED
		.... 1...		INLCRSRP	REPLACE PARAM. (1=YES,0=NO)
		.... .1..		INLCRSTL	TAPE LABEL (1=YES)
		.... ..1.		INLCRSOD	1 - DATE=OLD
		1		*	RESERVED
45	(2D)	BITSTRING	1	INLCUPD	UPDATE FLAGBYTE
		1...		INLCSAV	SAVE PARAMETER (1=YES,0=NO)
		.1..		INLCSEQ	SEQ PARAMETER (1=YES,0=NO)
		..1.		INLCCOL	COL PARAMETER (1=YES,0=NO)
		...1		INLCSQFS	SEQUENCE=FS (1=YES,0=NO)
		.... 1...		INLCSQNO	SEQUENCE=NO (1=YES,0=NO)
		.... .111		*	RESERVED
46	(2E)	BITSTRING	1	INLCINPU	INPUT FLAGBYTE
		1...		INLCIPT	SYSIPT OPTION (1=YES,0=NO)
		.111 1111		*	RESERVED
47	(2F)	BITSTRING	1	INLCREL	RELEASE FLAGBYTE
		1...		INLCRLLB	LIB FUNCTION (1=YES,0=NO)
		.1..		INLCRLSB	SUBLIB FUNCT. (1=YES,0=NO)
		..11 1111		*	RESERVED
48	(30)	BITSTRING	2	INLCCHAN	CHANGE FLAGBYTES
		1...		*	RESERVED
		.1..		INLCCHS	CHANGE SUBLIB (1=YES,0=NO)
		..1.		*	RESERVED
		...1		INLCCHRA	REUSE AUTOMATIC(1=YES,0=NO)
		.... 1...		INLCCHRI	REUSE IMMEDIATE(1=YES,0=NO)
48	(30)	BITSTRING	1	*	RESERVED
50	(32)	CHARACTER	16	INLCIDV	BACKUP/RESTORE ID-NAME

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
66	(42)	CHARACTER	31	INLCSTR	SINGLE QUALIFIED NAME
66	(42)	CHARACTER	7	INLCLIB	LIBRARY NAME
73	(49)	CHARACTER	8	INLCSUB	SUBLIBRARY NAME
81	(51)	CHARACTER	8	INLCMEM	MEMBER NAME
89	(59)	CHARACTER	8	INLCTYP	MEMBER TYPE
97	(61)	BITSTRING	3	INLCTRAC	TEST-TRACE FLAGBYTES
		1...		INLCTROF	1 - TRACE=OFF
		.1..		INLCTRAL	1 - TRACE=ALL
		..1.		INLCTRL1	1 - TRACE=LEVEL1
		...1		INLCTRL2	1 - TRACE=LEVEL2
		.... 1...		INLCTRSP	1 - TRACE=SPACE
		.... .1..		INLCTRIO	1 - TRACE=IO
		.... ..1.		INLCTRBF	1 - TRACE=BUFFER
		1		INLCTRPA	1 - PART = XX
98	(62)	1...		INLCTRPL	1 - PART = ALL
		.1..		INLCTRTK	1 - TASK = YY
98	(62)	BITSTRING	1	*	RESERVED
100	(64)	SIGNED	4	*	RESERVED
104	(68)	BITSTRING	1	INLCBKP2	BACKUP FLAGBYTE 2
		1...		INLCBKTL	TAPE LABEL (1=YES)
		.1..		INLCBKCT	CUSTTABLE (1=YES)
		..1.		INLCBKDS	1 - DSF OP. SPECFD
		...1		*	RESERVED
		.... 1...		INLCBKM	BACKUP MBR (1=YES)
		.... .1..		INLCBKLY	1 - LIST=YES
		.... ..1.		INLCBKLN	1 - LIST=NO SPECFD
		1		*	RESERVED
105	(69)	BITSTRING	1	INLCLIP2	LIST/PUNCH FLAGBYTE 2
		1...		INLCIEBU	FORMAT=IEBUPDTE(1=YES,0=NO)
		.111 1111		*	RESERVED
106	(6A)	BITSTRING	2	*	RESERVED
108	(6C)	CHARACTER	7	INLCTLNM	FILENAME OF TLBL
115	(73)	CHARACTER	31	INLCCUST	QUALIFIED NAME OF CUSTOMIZATION TABLE
115	(73)	CHARACTER	7	INLCCTL	LIBRARY NAME
122	(7A)	CHARACTER	8	INLCCTS	SUBLIBRARY NAME
130	(82)	CHARACTER	8	INLCCTMN	MEMBER NAME
138	(8A)	CHARACTER	8	INLCCTMT	MEMBER TYPE
146	(92)	CHARACTER	4	*	RESERVED
150	(96)	CHARACTER	9	INLCLKUL	LOCK UNLOCK CMD. & GENERAL LCKG. PAR.
		1...		INLCLKL	LIST OF LIBRARIES
		.1..		INLCLKS	LIST OF SUBLIBS
		..1.		INLCLKFO	1 - IF FORCE = YES
		...1		INLCLKIF	LKID SP. GEN. PAR.
		.... 11..		INLCLKCF	LOCK SP. GEN. PAR.
		.... 1...		INLCLKCC	00 - LOCK = NORMAL
		.... .1..		INLCLKCI	10 - NOT ALLOWED 01 - LOCK = RESET 11 - LOCK = COPY
		11		INLCLKRL	RESETLOCK GEN. PAR
		.... ..1.		INLCLKRO	ON
		1		INLCLKRF	OFF
151	(97)	CHARACTER	8	INLCLKID	LOCKID
159	(9F)	CHARACTER	1	*	RESERVED
160	(A0)	CHARACTER	13	INLCSRCH	SEARCH MEMBER
		1...		INLCSRL	LIST OF LIBRARIES
		.1..		INLCSRS	LIST OF SUBLIBS
		..1.		INLCSRAC	LBR ACCESS CHAIN
		...1		INLCSRCO	LBR CONNECT CHAINS
		.... 1...		INLCSRPH	PHASE
		.... .1..		INLCSROB	OBJECT
		.... ..1.		INLCSRSO	SOURCE
		1		INLCSRPR	PROC
161	(A1)	1...		INLCSRDU	DUMP
		.1..		INLCSRSR	SEARCH CHAIN
		..1.		INLCSRCA	CATALOG CHAIN
		...1		INLCSRAN	L=

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
		.... 1...		INLCSRLG	UNIT = SYSLOG
		.... .1..		INLCSRLS	UNIT = SYSLIST
161	(A1)	BITSTRING	1	*	RESERVED
163	(A3)	CHARACTER	2	INLCSRPA	PARTITION ID
165	(A5)	CHARACTER	8	*	RESERVED
173	(AD)	CHARACTER	31	INLCNAM	MEMBERSECIFICATION
173	(AD)	CHARACTER	8	INLCMNAM	
181	(B5)	CHARACTER	8	INLCMTYP	
189	(BD)	CHARACTER	7	INLCLIBN	LIBRARY NAME
196	(C4)	CHARACTER	8	INLCSUBN	SUBLIBRARYNAME
204	(CC)	CHARACTER	33	INLCTIME	CONTAINS TIME SPEC.
237	(ED)	CHARACTER	2	INLCTASK	CONTAINS TASK SPEC.
239	(EF)	CHARACTER	8	*	RESERVED

Cross Reference

Name	Hex Offset	Hex Value	Level
CMDPID	0		2
INLCACC	20		2
INLCADIS	20	80	3
INLCALST	C		2
INLCBIGL	28	10	3
INLCBKCT	68	40	3
INLCBKDS	68	20	3
INLCBKHD	2A	20	3
INLCBKHF	2A	02	3
INLCBKID	2A	01	3
INLCBKL	2A	80	3
INLCBKLN	68	02	3
INLCBKLU	2A	08	3
INLCBKLY	68	04	3
INLCBKM	68	08	3
INLCBKOF	2A	10	3
INLCBKP	2A		2
INLCBKPU	2A	04	3
INLCBKP2	68		2
INLCBKS	2A	40	3
INLCBKTL	68	80	3
INLCCAT	21		2
INLCCHAN	30		2
INLCCHRA	30	10	3
INLCCHRI	30	08	3
INLCCHS	30	40	3
INLCCMD	5		2
INLCCMDP	0		1
INLCCODA	22	04	3
INLCCOL	2D	20	3
INLCCOLB	22	80	3
INLCCOLE	1A		4
INLCCOLI	22	08	3
INLCCOLS	18		4
INLCCOLV	18		3
INLCCOMB	22	20	3
INLCCONN	26		2
INLCCOP	22		2
INLCCOPU	22	02	3
INLCCORP	22	10	3
INLCCOSB	22	40	3
INLCCTL	73		3
INLCCTMN	82		3
INLCCTMT	8A		3
INLCCTS	7A		3
INLCCUST	73		2

Name	Hex Offset	Hex Value	Level
INLCDATA	21	80	3
INLCDEF	23		2
INLCDEL	24		2
INLCDFL	23	80	3
INLCDFRA	23	20	3
INLCDFRI	23	10	3
INLCDFRP	23	08	3
INLCDFS	23	40	3
INLCDLL	24	80	3
INLCDLM	24	20	3
INLCDLS	24	40	3
INLCEOD	21	20	3
INLCEODV	12		3
INLCFORM	25	10	3
INLCGENP	1C		2
INLCIDV	32		2
INLCIEBU	69	80	3
INLCINPU	2E		2
INLCIPT	2E	80	3
INLCLDFN	27	02	3
INLCLDL	27	80	3
INLCLDLG	27	08	3
INLCLDLS	27	04	3
INLCLDM	27	20	3
INLCLDPH	28	20	3
INLCLDS	27	40	3
INLCLDSH	27	01	3
INLCLDST	28	80	3
INLCLI	25	80	3
INLCLIB	42		3
INLCLIBN	BD		3
INLCLID	27		2
INLCLIP	25		2
INLCLIP2	69		2
INLCLIX	25	40	3
INLCLKCC	96	08	4
INLCLKCF	96	0C	3
INLCLKCI	96	04	4
INLCLKFO	96	20	3
INLCLKID	97		3
INLCLKIF	96	10	3
INLCLKL	96	80	3
INLCLKRF	96	01	4
INLCLKRL	96	03	3
INLCLKRO	96	02	4
INLCLKS	96	40	3
INLCLKUL	96		2
INLCLPLG	25	08	3
INLCLPLS	25	04	3
INLCMDIS	26	80	3
INLCMEM	51		3
INLCMNAM	AD		3
INLCMTYP	B5		3
INLCNAM	AD		2
INLCNO	10		2
INLCPARM	12		2
INLCPEOF	25	02	3
INLCPNOH	25	01	3
INLCPUN	25	20	3
INLCREL	2F		2
INLCREN	29		2
INLCREP	21	40	3
INLCRES	2B		2
INLCRLLB	2F	80	3
INLCRLSB	2F	40	3

Name	Hex Offset	Hex Value	Level
INLCRNM	29	40	3
INLCRNS	29	80	3
INLCRSA	2B	08	3
INLCRSCN	2B	04	3
INLCRSIA	2C	40	3
INLCRSID	2C	80	3
INLCRSL	2B	80	3
INLCRSLN	2C	10	3
INLCRSLU	2B	02	3
INLCRSLY	2C	20	3
INLCRSM	2B	20	3
INLCRSO	2B	10	3
INLCRSOD	2C	02	3
INLCRSPU	2B	01	3
INLCRSRP	2C	08	3
INLCRSS	2B	40	3
INLCRSTL	2C	04	3
INLCSAV	2D	80	3
INLCSDL	27	10	3
INLCSEQ	2D	40	3
INLCSEQV	14		3
INLCSQFS	2D	10	3
INLCSQNO	2D	08	3
INLCSRAC	A0	20	3
INLCSRAN	A1	10	3
INLCSRCA	A1	20	3
INLCSRCH	A0		2
INLCSRCO	A0	10	3
INLCSRDU	A1	80	3
INLCSRL	A0	80	3
INLCSRLG	A1	08	3
INLCSRLS	A1	04	3
INLCSROB	A0	04	3
INLCSRPA	A3		3
INLCSRPH	A0	08	3
INLCSRPR	A0	01	3
INLCSRS	A0	40	3
INLCSRSO	A0	02	3
INLCSRSR	A1	40	3
INLCSTR	42		2
INLCSUB	49		3
INLCSUBN	C4		3
INLCTASK	ED		2
INLCTIME	CC		2
INLCTLNM	6C		2
INLCTRAC	61		2
INLCTRAL	61	40	3
INLCTRBF	61	02	3
INLCTRIO	61	04	3
INLCTRL1	61	20	3
INLCTRL2	61	10	3
INLCTROF	61	80	3
INLCTRPA	61	01	3
INLCTRPL	62	80	3
INLCTRSP	61	08	3
INLCTRTK	62	40	3
INLCTSRP	28	40	3
INLCTYP	59		3
INLCUNV	16		3
INLCUPD	2D		2
INLCVER	4		2

INLCCOMR

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	80	INLCCOMR	
0	(0)	CHARACTER	4	COMRID	HEADER IDENTIFIER
4	(4)	CHARACTER	3	CRGVERS	VERSION/MODIFICATION LEVEL
7	(7)	UNSIGNED	1	CRGCMDL	LENGTH OF COMMAND BLOCK
8	(8)	SIGNED	4	CRGLOAD	LOAD ADDRESS FOR LEVEL-3
12	(C)	SIGNED	4	CRGCMDP	ADDRESS OF COMMAND BLOCK
16	(10)	SIGNED	4	CRGNMLIS	ADDRESS OF NAME LIST AREA
20	(14)	SIGNED	4	CRGNMLGT	LENGTH OF NAME LIST AREA
24	(18)	SIGNED	4	CRGDTFIN	ADDRESS OF SYSIN DTF
28	(1C)	UNSIGNED	1	CRGIPTL	LENGTH OF SYSIPT I/O-AREA
29	(1D)	A-ADDRESS	3	CRGIPTIO	ADDRESS OF SYSIPT I/O-AREA
32	(20)	SIGNED	4	CRGDTFPC	ADDRESS OF SYSPCH DTF
36	(24)	UNSIGNED	1	CRGPCHL	LENGTH OF SYSPCH I/O-AREA
37	(25)	A-ADDRESS	3	CRGPCHIO	ADDRESS OF SYSPCH I/O-AREA
40	(28)	A-ADDRESS	4	CRGLAMB	ADDRESS OF LAMB
44	(2C)	SIGNED	4	CRGPHPTR	ACTUAL BRANCH ADDRESS TO THE LEV EL-3 SUB-PHASE
48	(30)	CHARACTER	8	CRGPHN	NAME OF LEVEL-3 PHASE WHICH IS ACTUALLY LOADED
56	(38)	BITSTRING	1	CRGFLFLG	SYSTEM FILE FLAGBYTE
		1...		CRGOPIN	SYSIN DTF IS OPENED
		.1..		CRGOPPCH	SYSPCH DTF IS OPENED
		..1.		CRG81BYT	SYSIPT 81-BYTE RECORDS
		...1		CRGOPLOG	SYSLOG DTF IS OPENED
		.... 1...		CRGOPLST	SYSLST DTF IS OPENED
		.... .111			RESERVED
57	(39)	BITSTRING	2	CRGFLAG	FLAGBYTES
		1...		CRGACC	PARMINF FROM EXEC LIBR
		.1..		CRGMSHP	RUNNING UNDER MSHP CONTROL
		..1.		CRGMSHPB	MSHP BYPASS REQUESTED
		...1		CRGLOG	LIBR STARTED FROM SYSLOG
		.... 1...		CRGEOF	EOF ON SYSIPT/SYSLOG
		.... .1..		CRGPIDV2	RESTORE INPUT TAPE IS OF FORMAT: PID-V2-STACKED
		.... ..1.		CRGEODCK	CHECK FOR EOD STATEMENT WHEN READING NEXT COMMAND
		1		CRGMSG01	WRITE MESSAGE 001
		1...		CRGMSG03	WRITE MESSAGE 003
		.1..		CRGMSG04	WRITE MESSAGE 004
		..1.		CRGCOMM	READING WITHIN A COMMENT
		...1		CRGCONN	CONNECT COMMAND ACTIVE
		.... 1...		CRGRQ1ST	1ST SYSIPT REQUEST COMPLETE
		.... .1..		CRGGTVIS	NAMELIST IN GETVIS AREA
		.... ..1.		CRGICCF	ICCF ACTIVE
		1		CRGSYNER	SYNTAX ERROR OCCURRED
59	(3B)	CHARACTER	2	CRGEOD	EOD STRING TO BE CHECKED
61	(3D)	UNSIGNED	1	CRGRETCD	CURRENT RETURN CODE OF EXECUTION MODULE
62	(3E)	BITSTRING	2	CRGMIGFL	FLAGBYTE FOR MIGRATION
		1...		CRGMIGR	MIGRATION IN PROCESS
		.1..		CRGMAINT	MAINT-MIGRATION IN PROCESS
		..1.		CRGCORGZ	CORGZ-MIGRATION IN PROCESS
		...1		CRGCSERV	CSERV-MIGRATION IN PROCESS
		.... 1...		CRGDSERV	DSERV-MIGRATION IN PROCESS
		.... .1..		CRGPSERV	PSERV-MIGRATION IN PROCESS
		.... ..1.		CRGRSERV	RSERV-MIGRATION IN PROCESS
		1		CRGSSERV	SSERV-MIGRATION IN PROCESS
		1...		CRGSRALL	REQUEST FOR ALL SOURCE TYPES
		.1..		CRGPUNSA	PUNCH SLASH-ASTERISK
		..11 1111			RESERVED
64	(40)	SIGNED	4	CRGNMLEN	LENGTH OF NAME LIST AREA IN GETVIS SPACE
68	(44)	BITSTRING	1	CRGONFL	FLAGBYTE FOR 'ON' CONDIT. AND DELAYED AR-CANCEL

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
		1...		CRGONFLG	'ON'-CONDITION ACTIVE
		.1..			RESERVED
		..1.		CRGCAREQ	1= CANCEL REQUESTED
		...1 1111			RESERVED
69	(45)	UNSIGNED	1	CRGSAVRC	MAX. RC PER JOB STEP
70	(46)	BITSTRING	2	CRGFLAGB	FURTHER FLAGBYTES
		1...		CRGIEBUP	PUNCH WITH IEBUPDTE
		.1..		CRGNOIEB	PUNCH W/O IEBUPDTE
		..1.		CRGNOASA	SYSPUNCH TO TAPE OR DASD
		...1 1111			RESERVED
		1111 1111			RESERVED
72	(48)	SIGNED	4	CRGONIND	INDEX OF 'ON' CONDITION STACK
76	(4C)	UNSIGNED	1	CRGFUNC	CALL-INTERFACE FUNCTION
77	(4D)	UNSIGNED	3	CRGPARMS	START OF EXEC PARM LIBR COMMAND LINE

Cross Reference:

Name	Hex Offset	Hex Value	Level
COMRID	0	(0)	
CRGACC	57	X'80'	
CRGCAREQ	68	X'20'	
CRGCMDL	7	(7)	
CRGCMDP	12	(C)	
CRGCOMM	58	X'20'	
CRGCONN	58	X'10'	
CRGCORGZ	62	X'20'	
CRGCSERV	62	X'10'	
CRGDSERV	62	X'08'	
CRGDTFIN	24	(18)	
CRGDTFPC	32	(20)	
CRGEOD	59	(3B)	
CRGEODCK	57	X'02'	
CRGEOF	57	X'08'	
CRGFLAG	57	(39)	
CRGFLFLG	56	(38)	
CRGFUNC	76	(4C)	
CRGGTVIS	58	X'04'	
CRGICCF	58	X'02'	
CRGIPTIO	29	(1D)	
CRGIPTL	28	(1C)	
CRGLAMB	40	(28)	
CRGLOAD	8	(8)	
CRGLOG	57	X'10'	
CRGMAINT	62	X'40'	
CRGIEBUP	70	X'80'	
CRGMIGFL	62	(3E)	
CRGMIGR	62	X'80'	
CRGMSG01	57	X'01'	
CRGMSG03	58	X'80'	
CRGMSG04	58	X'40'	
CRGMSHP	57	X'40'	
CRGMSHPB	57	X'20'	
CRGNMLEN	64	(40)	
CRGNMLGT	20	(14)	
CRGNMLIS	16	(10)	
CRGNOASA	70	X'20'	
CRGNOIEB	70	X'40'	
CRGONFL	68	(44)	
CRGONFLG	68	X'80'	
CRGONIND	72	(48)	
CRGOPIN	56	X'80'	
CRGOPLOG	56	X'10'	
CRGOPLST	56	X'08'	
CRGOPPCH	56	X'40'	
CRGPARMS	77	(4D)	
CRGPCHIO	37	(25)	
CRGPCHL	36	(24)	
CRGPHN	48	(30)	
CRGPHPTR	44	(2C)	
CRGPIDV2	57	X'04'	
CRGPSERV	62	X'04'	
CRGPUNSA	63	X'40'	
CRGRETCD	61	(3D)	
CRGRQ1ST	58	X'08'	
CRGRSERV	62	X'02'	
CRGSAVRC	69	(45)	
CRGSRALL	63	X'80'	
CRGSSERV	62	X'01'	
CRGSYNER	58	X'01'	
CRGVERS	4	(4)	

Name	Hex Offset	Hex Value	Level
CRG81BYT	56	X'20'	
INLCCOMR	0	(0)	

INLCDDTE

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	8	INLCDDTE	DEVICE DEFINITION TABLE
0	(0)	BITSTRING	1	DDTETYP	RPS DEVICE TYPE
1	(1)	BITSTRING	1	DDTEDEV	DEVICE INDICATOR
2	(2)	BITSTRING	1	DDTESTAT	STATUS INFORMATION
3	(3)	BITSTRING	1		RESERVED
4	(4)	SIGNED	4	DDTEBLZ	FBA BLOCKSIZE
4	(4)	SIGNED	2	DDTETRCY	# TRACKS PER CYLINDER
6	(6)	SIGNED	2	DDTEBLTR	# LBS PER TRACK

Cross Reference:

Name	Hex Offset	Hex Value	Level
DDTEBLTR	6	(6)	
DDTEBLZ	4	(4)	
DDTEDEV	1	(1)	
DDTESTAT	2	(2)	
DDTETRCY	4	(4)	
DDTETYP	0	(0)	
INLCDDTE	0	(0)	

MEMBER DIRECTORY ENTRY

INLCDENT

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	72	INLCDENT	DIRECTORY ENTRY
0	(0)	CHARACTER	38	DENTSEGM	D.E. FIRST SEGMENT
0	(0)	CHARACTER	8	DENTNAM	MEMBER NAME
8	(8)	CHARACTER	1	DENTGEN	RESERVED FOR GENERATION G.
9	(9)	CHARACTER	1	DENTDEF1	ATTRIBUTES FOR DIR.ENTRY
		11..		*	RESERVED
		..1.		DENTEDIR	TYPE OF ENTRY = DIRECTORY
		...1 1111		*	RESERVED
10	(A)	CHARACTER	6	DENTPRBA	MEMBER BEGIN PRBA
10	(A)	UNSIGNED	2	DENTRBAO	OFFSET IN LB
12	(C)	UNSIGNED	4	DENTRBAP	REL.BLOCK ADDRESS
16	(10)	SIGNED	2	DENTVIFL	LENGTH OF DE COMMON SEGMNT
18	(12)	CHARACTER	6	DENTLSTA	PRBA OF LAST LB OF MEMBER
18	(12)	UNSIGNED	2	DENTLSTO	OFFSET IN LB
20	(14)	UNSIGNED	4	DENTLSTP	REL.BLOCK ADDRESS
24	(18)	BITSTRING	1	DENTDEF2	FLAGS
		1...		DENTMSH1	MODULE UNDER MSHP CONT.
		.1..		DENTMSH2	CHANGED WITH MSHP BYPASS
		..1.		DENTAPI	BUILD WITH API
		...1		*	RESERVED
		.... 1...		DENTRTF	REC.TYPE=FIXED
		.... .1..		DENTRTU	REC.TYPE=UNDEFINED
		.... ..1.		DENTRTV	REC.TYPE=VARIABLE
		1		DENTSIPT	SYSIPT DATA IN MEMBER
25	(19)	BITSTRING	1	DENTDEF3	
		1...		DENTVIF	ONE OR MORE VIFS IN DE
		.1..		DENTCMPR	DATA IN COMPRESSED FORM
		..11 1111		*	RESERVED
26	(1A)	SIGNED	2	DENTMBLK	NO. OF LBS FOR MEMBER
28	(1C)	SIGNED	4	DENTNORL	NO OF LOGICAL RECORDS
32	(20)	SIGNED	4	DENTRLEN	LOGICAL REC. LENGTH
36	(24)	SIGNED	2	DENTCONT	CONTIGUITY COUNTER
38	(26)	CHARACTER	10	DENTDORI	TIME STAMP ORIGINATION
48	(30)	CHARACTER	10	DENTDUPD	TIME STAMP LAST UPDATE
58	(3A)	CHARACTER	2	DENTVEMO	VERSION MODIFICATION
60	(3C)	CHARACTER	2	DENTEODA	SYSIPT END OF DATA
62	(3E)	CHARACTER	2	*	RESERVED
64	(40)	CHARACTER	4	DENTIDEN	RESERVED
68	(44)	CHARACTER	4	*	RESERVED

Cross Reference:

Name	Hex Offset	Hex Value	Level
DENTAPI	18		4
DENTCMPR	19		4
DENTCONT	24		3
DENTDEF1	9		3
DENTDEF2	18		3
DENTDEF3	19		3
DENTDORI	26		2
DENTDUPD	30		2
DENTEDIR	9		4
DENTEODA	3C		2
DENTGEN	8		3
DENTIDEN	40		2
DENTLSTA	12		3
DENTLSTO	12		4
DENTLSTP	14		4
DENTMBLK	1A		3
DENTMSH1	18		4
DENTMSH2	18		4
DENTNAM	0		3
DENTNORL	1C		3
DENTPRBA	A		3
DENTRBAO	A		4
DENTRBAP	C		4
DENTRLEN	20		3
DENTRTF	18		4
DENTRTU	18		4
DENTRTV	18		4
DENTSEGM	0		2
DENTSIPT	18		4
DENTVEMO	3A		2
DENTVIF	19		4
DENTVIFL	10		3
INLCDENT	0		1

INLCDESC

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	10	INLCDESC	ENTRY DESCRIPTOR
0	(0)	CHARACTER	8	DESCNAME	ENTRY NAME
8	(8)	CHARACTER	1	DESCGEN	RESERVED
9	(9)	BITSTRING	1	DESCFLAG	FLAG BYTE

Cross Reference:

Name	Hex Offset	Hex Value	Level
DESCFLAG	9	(9)	
DESCGEN	8	(8)	
DESCNAME	0	(0)	
INLCDESC	0	(0)	

INLCEDTE

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	44	INLCEDTE	EXTENT DEFINITION TABLE
0	(0)	SIGNED	4	EDTELODA	LOWEST PRBA OF EXTENT
4	(4)	SIGNED	4	EDTEHIDA	HIGHEST PRBA OF EXTENT
8	(8)	UNSIGNED	2	EDTELUPU	LU OR PUB INDEX
8	(8)	UNSIGNED	2	EDTEPUB	PUB INDEX (TWO BYTE)
8	(8)	BITSTRING	1		INTERNALLY USED
9	(9)	UNSIGNED	1	EDTELU	LOGICAL UNIT
10	(A)	UNSIGNED	1	EDTEERR	RESERVED
11	(B)	BITSTRING	1	EDTEFLAG	EXTENT FLAG BYTE
12	(C)	CHARACTER	4	EDTESTRX	START OF EXTENT
12	(C)	UNSIGNED	2	EDTESTRC	START OF EXTENT CC
14	(E)	UNSIGNED	2	EDTESTRH	START OF EXTENT HH
16	(10)	CHARACTER	4	EDTEENDX	END OF EXTENT
16	(10)	UNSIGNED	2	EDTEENDC	END OF EXTENT CC
18	(12)	UNSIGNED	2	EDTEENDH	END OF EXTENT HH
20	(14)	SIGNED	4	EDTENTRK	# TRACKS OR # PBN
24	(18)	SIGNED	4		RESERVED
28	(1C)	CHARACTER	6	EDTEVLID	VOLID
34	(22)	CHARACTER	2	EDTECUU	CHANNEL/UNIT
36	(24)	A-ADDRESS	4	EDTEDDTX	ADDR OF DEVICE DEFINITION
40	(28)	A-ADDRESS	4	EDTENEXT	ADDR OF NEXT EDTE

Cross Reference:

Name	Hex Offset	Hex Value	Level
EDTECUU	34	(22)	
EDTEDDTX	36	(24)	
EDTEENDC	16	(10)	
EDTEENDH	18	(12)	
EDTEENDX	16	(10)	
EDTEERR	10	(A)	
EDTEFLAG	11	(B)	
EDTEHIDA	4	(4)	
EDTELODA	0	(0)	
EDTELU	9	(9)	
EDTELUPU	8	(8)	
EDTENEXT	40	(28)	
EDTENTRK	20	(14)	
EDTEPUB	8	(8)	
EDTEPUBX	8	(8)	
EDTESTRC	12	(C)	
EDTESTRH	14	(E)	
EDTESTRX	12	(C)	
EDTEVLID	28	(1C)	
INLCEDTE	0	(0)	

INLCEXTD

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	64	INLCEXTD	DISK EXTENT DESCRIPTOR
0	(0)	CHARACTER	12		CONTROL BLOCK ID
0	(0)	CHARACTER	8	EXTDID	CONTROL BLOCK ID
8	(8)	CHARACTER	4	EXTDNO	CONTROL BLOCK SEQ. NUMBER
12	(C)	SIGNED	4	EXTDLODA	LOWEST PRBA IN EXTENT
16	(10)	SIGNED	4	EXTDHIDA	HIGHEST PRBA IN EXTENT
20	(14)	CHARACTER	4	EXTDSTRX	LOWEST DISK ADDR IN EXTENT
24	(18)	CHARACTER	4	EXTDENDX	HIGHEST DISK ADDR IN EXTENT
28	(1C)	SIGNED	4	EBMBLK	LENGTH OF BITMAP IN LBS
32	(20)	SIGNED	4	EBMBYT	LENGTH OF BITMAP IN BYTES
36	(24)	SIGNED	4	EBMBIT	LENGTH OF BITMAP IN BITS
40	(28)	SIGNED	4	EXTDFSP	1TH ALLOCATABLE PRBA IN EXTENT
44	(2C)	UNSIGNED	2	EXTDLK	LENGTH OF CONTROL BLOCK
46	(2E)	SIGNED	2		RESERVED
48	(30)	CHARACTER	12		RESERVED
60	(3C)	SIGNED	4	EXTDNTRK	NO. OF TRACKS IN EXTENT

Cross Reference:

Name	Hex Offset	Hex Value	Level
EBMBIT	36	(24)	
EBMBLK	28	(1C)	
EBMBYT	32	(20)	
EXTDENDX	24	(18)	
EXTDFSP	40	(28)	
EXTDHIDA	16	(10)	
EXTDID	0	(0)	
EXTDLK	44	(2C)	
EXTDLODA	12	(C)	
EXTDNO	8	(8)	
EXTDNTRK	60	(3C)	
EXTDSTRX	20	(14)	
INLCEXTD	0	(0)	

INLCFSRL

FORMATTED SPACE REQUIREMENT LIST ELEMENT FOR
PRODUCTS WHICH ARE BACKED UP AS SUBLIBRARIES

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	120	INLCFSRL	
0	(0)	SIGNED	2	FSRLLEN	LENGTH OF FSRL ELEMENT
2	(2)	CHARACTER	2	FSRLID	FSRL-ID
4	(4)	CHARACTER	16	FSRLBFID	BACKUP FILE ID
20	(14)	SIGNED	4	FSRLLBS	MIN LIBRARY BLOCKS REQUIRED
24	(18)	SIGNED	4	*	RESERVED
28	(1C)	SIGNED	4	*	RESERVED
32	(20)	SIGNED	4	*	RESERVED
36	(24)	SIGNED	4	*	RESERVED
40	(28)	SIGNED	4	CYL3350	REQUIRED CYLINDERS FOR 3350
44	(2C)	SIGNED	4	TRK3350	REQUIRED TRACKS FOR 3350
48	(30)	SIGNED	4	CYL3375	REQUIRED CYLINDERS FOR 3375
52	(34)	SIGNED	4	TRK3375	REQUIRED TRACKS FOR 3375
56	(38)	SIGNED	4	CYL3380	REQUIRED CYLINDERS FOR 3380
60	(3C)	SIGNED	4	TRK3380	REQUIRED TRACKS FOR 3380
64	(40)	SIGNED	4	*	RESERVED
68	(44)	BITSTRING	1	FSRLSW	SWITCH BYTE
		1...		FSRLPID2	PID-V2-STACKED TAPE
69	(45)	CHARACTER	3	*	RESERVED
72	(48)	SIGNED	4	FSRLFBA	FBA BLOCKS REQUIRED
76	(4C)	SIGNED	4	*	RESERVED
80	(50)	SIGNED	4	CYL3390	REQUIRED CYLS 3390
84	(54)	SIGNED	4	TRK3390	REQUIRED TRACKS 3390
88	(58)	SIGNED	4	CYL9345	REQUIRED CYLS 9345
92	(5C)	SIGNED	4	TRK9345	REQUIRED TRACKS 9345
96	(60)	CHARACTER	24	*	RESERVED

Cross Reference

Name	Hex Offset	Hex Value	Level
CYL3350	28		2
CYL3375	30		2
CYL3380	38		2
CYL3390	50		2
CYL9345	58		2
FSRLBFID	4		2
FSRLFBA	48		2
FSRLID	2		2
FSRLLBS	14		2
FSRLLEN	0		2
FSRLPID2	44	80	3
FSRLSW	44		2
INLCFSRL	0		1
TRK3350	2C		2
TRK3375	34		2
TRK3380	3C		2
TRK3390	54		2
TRK9345	5C		2

INLCIDTE

OPEN IDENTIFICATION TABLE

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	64	INLCIDTE	OPEN IDENTIFICATION TABLE
0	(0)	ADDRESS	4	IDTEMEMB	VSAM CATALOG MEMBER CHAIN
4	(4)	CHARACTER	44	IDTEFID	FILE-ID
4	(4)	CHARACTER	44	IDTECID	VSAM CATALOG ID
48	(30)	BITSTRING	1	*	RESERVED
49	(31)	BITSTRING	1	IDTEFLAG	FLAG BYTE
		1...		IDTEVSAM	- 0: BAM , 1: VSAM
		.1..		IDTECTLG	- 0: FILE-ID, 1: CATALOG
		..11 111.		*	RESERVED
		1		IDTEACTV	- 0: INACTIVE, 1: ACTIVE
50	(32)	CHARACTER	6	IDTEVLID	VOLUME IDENTIFIER
56	(38)	CHARACTER	4	IDTEADDR	LIBRARY START ADDRESS
60	(3C)	CHARACTER	2	*	RESERVED
62	(3E)	CHARACTER	2	IDTECUU	PHYSICAL UNIT

Cross Reference

Name	Hex Offset	Hex Value	Level
IDTEACTV	31	01	3
IDTEADDR	38		2
IDTECID	4		3
IDTECTLG	31	40	3
IDTECUU	3E		2
IDTEFID	4		2
IDTEFLAG	31		2
IDTEMEMB	0		2
IDTEVLID	32		2
IDTEVSAM	31	80	3
INLCIDTE	0		1

INLCLACB

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	148	INLCLACB	
0	(0)	CHARACTER	66	LACBSAVE	RESERVED
66	(42)	CHARACTER	6	LACBGDIR	NEXT RECORD PTR FOR GDIR
72	(48)	CHARACTER	8	LACBHDR	LACB HEADER "INLCLACB"
80	(50)	CHARACTER	16	LACBINFO	LIBINFO FIELD
96	(60)	A-ADDRESS	4	LACBLAMB	ADDRESS OF LAMB
100	(64)	CHARACTER	6	LACBSXST	PRBA OF SUBLIB INDEX START
100	(64)	UNSIGNED	2	LACBSXOS	OFFSET IN LB
102	(66)	SIGNED	4	LACBSXBN	LIBRARY BLOCK #
106	(6A)	CHARACTER	6	LACBCSXE	PRBA OF CURRENT INDEX ENTRY
106	(6A)	UNSIGNED	2	LACBCXOS	OFFSET IN LB
108	(6C)	SIGNED	4	LACBCXBN	LIBRARY BLOCK #
112	(70)	CHARACTER	8	LACBBREQ	REQUEST LIST FOR BUFFER MGT
120	(78)	A-ADDRESS	4	LACBCBUF	ADDRESS OF CURRENT X BUFFER
124	(7C)	A-ADDRESS	4	LACBAEDT	ADDRESS OF EXTENT TABLE
128	(80)	BITSTRING	4	LACBFLAG	CONTROL FLAGS
		1...		LACBRCLM	IMMEDIATE SPACE RECLAM.
132	(84)	CHARACTER	8	LACBSREQ	REQUEST LIST FOR BUFFER MGT
140	(8C)	A-ADDRESS	4	LACBSBUF	ADDRESS OF ALTERNATE BUFFER
144	(90)	SIGNED	4	LACBIDEN	NEXT IDENTIFICATION VALUE

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCLACB	0	(0)	
LACBAEDT	124	(7C)	
LACBBREQ	112	(70)	
LACBCBUF	120	(78)	
LACBCSXE	106	(6A)	
LACBCXBN	108	(6C)	
LACBCXOS	106	(6A)	
LACBFLAG	128	(80)	
LACBGDIR	66	(42)	
LACBHDR	72	(48)	
LACBIDEN	144	(90)	
LACBINFO	80	(50)	
LACBLAMB	96	(60)	
LACBRCLM	128	X'80'	
LACBSAVE	0	(0)	
LACBSBUF	140	(8C)	
LACBSREQ	132	(84)	
LACBSXBN	102	(66)	
LACBSXOS	100	(64)	
LACBSXST	100	(64)	

INLCLAMB

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	176	INLCLAMB	
0	(0)	CHARACTER	72	LAMBSAVE	SAVE AREA
72	(48)	CHARACTER	8	LAMBHDR	HEADER FOR INLCLAMB
72	(48)	CHARACTER	4	LAMBID	LAMB-ID
76	(4C)	BITSTRING	1	LAMBLVL	VERSION
77	(4D)	BITSTRING	1		RESERVED
78	(4E)	SIGNED	2	LAMBLEN	LENGTH OF CONTROL BLOCK
80	(50)	A-ADDRESS	4	LAMBAMOD	ADDRESS OF SVA SERVICES
84	(54)	A-ADDRESS	4	LAMBSBUF	ADDRESS OF SHARED BUFFER AREA
88	(58)	SIGNED	4	LAMBLBUF	LENGTH OF SHARED BUFFER AREA
92	(5C)	A-ADDRESS	4	LAMBLST	ADDRESS OF DTF FOR SYSLST
96	(60)	A-ADDRESS	4	LAMBLOG	ADDRESS OF DTF FOR SYSLOG
100	(64)	A-ADDRESS	4	LAMBLSTA	ADDRESS OF LIST AREA
104	(68)	SIGNED	4	LAMBLSTL	LENGTH OF LIST AREA
108	(6C)	A-ADDRESS	4	LAMBLOGA	ADDRESS OF LOG AREA
112	(70)	SIGNED	4	LAMBLOGL	LENGTH OF LOG AREA
116	(74)	SIGNED	4	LAMBMID	MESSAGE ID #
120	(78)	SIGNED	4	LAMBMSG1	ADDRESS OF MESSAGE PART 1
124	(7C)	SIGNED	4	LAMBMSG2	ADDRESS OF MESSAGE PART 2
128	(80)	SIGNED	4	LAMBMSG3	ADDRESS OF MESSAGE PART 3
132	(84)	SIGNED	4	LAMBMSG4	ADDRESS OF MESSAGE PART 4
136	(88)	A-ADDRESS	4	LAMBEXTL	ADDRESS OF EXIT LIST
140	(8C)	BITSTRING	1	LAMBSW1	SWITCH BYTE 1
		1...		LAMBINT	INTERACTIVE MODE
		.1..		LAMBERR	0: ERROR=RETURN, 1: ERROR=CANCEL
		..1.		LAMBCALL	1: CALL I/F ACTIVE
		...1 1...			RESERVED
		.... .1..		LAMBLOCK	FAILURE: 0=WAIT, 1=RETURN
		.... ..1.		LAMBXTND	VSAM EXTENSION: 0=YES, 1=NO
		1		LAMBSVIS	SYSTEM GETVIS SPACE REQUIRED
141	(8D)	BITSTRING	1	LAMBSW2	SWITCH BYTE 2
		1...		LAMBSBIN	BUFFER INITIALIZED
		.1..		LAMBNOCA	SUPPRESS CANCEL
		.. 1 1...			RESERVED
		.... .1..		LAMBCINH	CACHE INHIBIT LOAD
		.... ..1.		LAMBCBYP	CACHE BYPASS
		1		LAMBGVIN	GETVIS STORAGE KEPT
142	(8E)	SIGNED	2	LAMBMSGL	LENGTH OF MESSAGE
144	(90)	SIGNED	2	LAMBCLCT	CURRENT LINE COUNT FOR SYSLST
146	(92)	SIGNED	2	LAMBMLCT	MAXIMUM LINE COUNT FOR SYSLST
148	(94)	CHARACTER	8	LAMBSPID	SUBPOOL ID FOR AUTOMATIC STORAGE
156	(9C)	A-ADDRESS	4	LAMBGVS	START ADDRESS OF GETVIS STORAGE
160	(A0)	A-ADDRESS	4	LAMBGVE	END ADDRESS OF GETVIS STORAGE
164	(A4)	A-ADDRESS	4	LAMBGVU	POINTER TO UNUSED GETVIS STORAGE
168	(A8)	CHARACTER	8		RESERVED

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCLAMB	0	(0)	
LAMBAMOD	80	(50)	
LAMBCALL	140	X'20'	
LAMBCBYP	141	X'02'	
LAMBCINH	141	X'04'	
LAMBCLCT	144	(90)	
LAMBERR	140	X'40'	
LAMBEXTL	136	(88)	
LAMBGVE	160	(A0)	
LAMBGVIN	141	X'01'	
LAMBGVS	156	(9C)	
LAMBGVU	164	(A4)	
LAMBHDR	72	(48)	
LAMBID	72	(48)	
LAMBINT	140	X'80'	
LAMBLBUF	88	(58)	
LAMBLEN	78	(4E)	
LAMBLOCK	140	X'04'	
LAMBLOG	96	(60)	
LAMBLOGA	108	(6C)	
LAMBLOGL	112	(70)	
LAMBLST	92	(5C)	
LAMBLSTA	100	(64)	
LAMBLSTL	104	(68)	
LAMBLVL	76	(4C)	
LAMBMID	116	(74)	
LAMBMLCT	146	(92)	
LAMBMSGL	142	(8E)	
LAMBMSG1	120	(78)	
LAMBMSG2	124	(7C)	
LAMBMSG3	128	(80)	
LAMBMSG4	132	(84)	
LAMBNOCA	141	X'40'	
LAMBSAVE	0	(0)	
LAMBSBIN	141	X'80'	
LAMBSBUF	84	(54)	
LAMBSPID	148	(94)	
LAMBSVIS	140	X'01'	
LAMBSW1	140	(8C)	
LAMBSW2	141	(8D)	
LAMBXTND	140	X'02'	

INLCLANC

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	20	INLCLANC	
0	(0)	BITSTRING	2		ENTRY INDICATION
2	(2)	SIGNED	2	LANCID	PIK OF ACTIVE PARTITION
4	(4)	A-ADDRESS	4	LANCTASK	TASK CHAIN ANCHOR
8	(8)	A-ADDRESS	4	LANCJOB	TEMP LIBDEF ANCHOR
12	(C)	A-ADDRESS	4	LANCPART	PERM LIBDEF ANCHOR
16	(10)	A-ADDRESS	4	LANCSYST	SYSTEM LIBDEF ANCHOR

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCLANC	0	(0)	
LANCID	2	(2)	
LANCJOB	8	(8)	
LANCPART	12	(C)	
LANCSYST	16	(10)	
LANCTASK	4	(4)	

INLCLARG

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	28	INLCLARG (*)	ARGUMENT
0	(0)	ADDRESS	4	LARGDEPT	DIR.ENTRY POINTER
4	(4)	SIGNED	2	LARGELNG	LENGTH OF STOWL ENTRY
6	(6)	UNSIGNED	1	LARGLSEQ	LIB/SUBLIB SEQ.NUMBER
7	(7)	BITSTRING	1	LARGSW	SWITCH BYTE
		1...		LARGFND	MEMBER FOUND
		.1..		LARGCAT	MEMBER CATALOGUED
		..1.		LARGDEL	MEMBER DELETED
		...1		LARGREN	MEMBER RENAMED
		.... 1...		LARGIPT	DATA ON SYSIPT
		.... .1..		LARGSYS	MEMBER IN SYSTEM LIBRARY
		.... ..1.		LARGSEC	SECURITY
		1		*	RESERVED
8	(8)	CHARACTER	17	LARGKEY	MEMBER KEY (TYPE.NAME)
8	(8)	CHARACTER	8	LARGMTYP	MEMBER TYPE
16	(10)	CHARACTER	8	LARGNAM	MEMBER NAME
24	(18)	CHARACTER	1	LARGGEN	RESERVED FOR GENERATION G.
25	(19)	CHARACTER	1	LARGDEF1	ATTRIBUTES FOR DIR.ENTRY
		11..		*	RESERVED
		..1.		LARGEDIR	TYPE OF ENTRY = DIRECTORY
		...1 1111		*	RESERVED
26	(1A)	CHARACTER	1	LARGLOCK	RESERVED
		1...		LARGDLCK	DIRECTORY LOCKED
		.1..		LARGDLID	LOCKID EQUAL
		..1.		LARGSLCK	SUPERVISOR LOCKED
		...1 1111		*	RESERVED
27	(1B)	CHARACTER	1	*	RESERVED

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCLARG	0		1
LARGCAT	7		3
LARGDEF1	19		2
LARGDEL	7		3
LARGDEPT	0		2
LARGDLCK	1A		3
LARGDLID	1A		3
LARGEDIR	19		3
LARGELNG	4		2
LARGFND	7		3
LARGGEN	18		3
LARGIPT	7		3
LARGKEY	8		2
LARGLOCK	1A		2
LARGLSEQ	6		2
LARGMTYP	8		3
LARGNAM	10		3
LARGREN	7		3
LARGSEC	7		3
LARGSLCK	1A		3
LARGSW	7		2
LARGSYS	7		3

INLCLBCF

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	36	INLCLBCF	
0	(0)	CHARACTER	8		RESERVED
8	(8)	SIGNED	4	LBCFIDEN	RESERVED
12	(C)	CHARACTER	2	LBCFFLGS	CONTROL FLAGS
12	(C)	BITSTRING	1	LBCFFLG1	RECORD FLAG BYTE
		1...		LBCFVARL	0: FIXED, 1: VARIABLE LENGTH
		.1..		LBCFCOMP	0: UNCOMPRESSED, 1: COMPRSD.
		..1.		LBCFSPND	0: NONSPANNED, 1: SPANNED
		...1 1111			RESERVED
13	(D)	BITSTRING	1	LBCFFLG2	LB FLAG BYTE
		1...		LBCFSRLI	STORED RECORD LENGTH FIELD IS 0: 1 BYTE, 1: 2 BYTES
		.111			RESERVED
		.... 1...		LBCFMDEL	1: MEMBER IS DELETED (IMM.)
		.... .1..		LBCFIDEL	1: MEMBER IS DELETED (DEL.)
		.... ..1.			RESERVED
		1		LBCFCCFI	1: CHAINING INF. INCOMPLETE
14	(E)	SIGNED	2	LBCFNREC	# OF LOGICAL RECORDS IN LB
16	(10)	SIGNED	2	LBCFRECL	(MAXIMUM) LOGICAL RECORD LEN.
18	(12)	SIGNED	2	LBCFOFFS	FREESPACE OFFSET
20	(14)	SIGNED	2	LBCFLNFS	FREESPACE LENGTH
22	(16)	SIGNED	2	LBCFCONT	# OF CONTIGUOUS BLOCKS FOLLOW.
24	(18)	CHARACTER	6	LBCFRBAB	BACKWARD CHAINING PRBA
24	(18)	SIGNED	2	LBCFLBOB	OFFSET IN LB
26	(1A)	CHARACTER	4	LBCFLBNB	LB NUMBER OF PREVIOUS LB
30	(1E)	CHARACTER	6	LBCFRBAF	FORWARD CHAINING PRBA
30	(1E)	SIGNED	2	LBCFLBOF	OFFSET IN LB
32	(20)	CHARACTER	4	LBCFLBNF	LB NUMBER OF NEXT LB

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCLBCF	0	(0)	
LBCFCCFI	13	X'01'	
LBCFCOMP	12	X'40'	
LBCFCONT	22	(16)	
LBCFFLGS	12	(C)	
LBCFFLG1	12	(C)	
LBCFFLG2	13	(D)	
LBCFIDEL	13	X'04'	
LBCFIDEN	8	(8)	
LBCFLBNB	26	(1A)	
LBCFLBNF	32	(20)	
LBCFLBOB	24	(18)	
LBCFLBOF	30	(1E)	
LBCFLNFS	20	(14)	
LBCFMDEL	13	X'08'	
LBCFNREC	14	(E)	
LBCFOFFS	18	(12)	
LBCFRBAB	24	(18)	
LBCFRBAF	30	(1E)	
LBCFRECL	16	(10)	
LBCFSPND	12	X'20'	
LBCFSRLI	13	X'80'	
LBCFVARL	12	X'80'	

INLCLBTB

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	428	INLCLBTB	LIB TABLE
0	(0)	CHARACTER	8	LBTBHDR	LIBRARY TABLE HDR
0	(0)	CHARACTER	4	LBTBID	LIBRARY TABLE ID
4	(4)	CHARACTER	2	LBTBVM	VERS./MODIFICATION
6	(6)	CHARACTER	2		RESERVED
8	(8)	UNSIGNED	2	LBTBINDX	INDEX TO ACC. LIB.
10	(A)	BITSTRING	1	LBTBSWT	GENERAL SWITCH
11	(B)	BITSTRING	1		RESERVED
12	(C)	CHARACTER	1552	LBTBLIBS	ARRAY OF LIBINFO
1564	(61C)	CHARACTER	3104	LBTBNTPT	ARRAY OF NOTE INFO
1564	(61C)	CHARACTER	28	LBTBLARG	
1592	(638)	SIGNED	4	LBTBIDEN	

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCLBTB	0	(0)	
LBTBHDR	0	(0)	
LBTBID	0	(0)	
LBTBIDEN	64	(40)	
LBTBINDX	8	(8)	
LBTBLARG	60	(3C)	
LBTBLIBS	12	(C)	
LBTBNTPT	60	(3C)	
LBTBSWT	10	(A)	
LBTBVM	4	(4)	

INLCLCKV

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	32	INLCLCKV	
0	(0)	SIGNED	2	LCKVELEN	LENGTH OF V.I.F.
2	(2)	BITSTRING	2	LCKVEATR	RESERVED
		1...		LCKVVIF	VIF FOLLOWS
4	(4)	CHARACTER	4	LCKVEID	V.I.F. IDENTIFIER
8	(8)	CHARACTER	8	LCKVLID	LOCKID
16	(10)	CHARACTER	16	*	RESERVED

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCLCKV	0		1
LCKVEATR	2		2
LCKVEID	4		2
LCKVELEN	0		2
LCKVLID	8		2
LCKVVIF	2		3

INLCLDES

LIBRARYDESCRIPTOR

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	216	INLCLDES	
0	(0)	CHARACTER	8	LDESLABL	LIBRARY IDENTIFIER "LIBRARY"
8	(8)	CHARACTER	8	LDESNAME	LIBRARY NAME
16	(10)	CHARACTER	8	LDESCRID	LIBRARY CREATOR ID
24	(18)	CHARACTER	10	LDESCRDT	TIME-STAMP OF CREATION DATE
34	(22)	CHARACTER	1	*	RESERVED
35	(23)	BITSTRING	1	LDESFLAG	LIBRARY ATTRIBUTES
		1...		LDESRCLM	- DEFAULT SPACE REUSAGE: 1=IMMEDIATE
		.1..		LDESALT	IPL EXCHANGED SYSLIB-ALTLIB
		..1.		LDESALTR	IPL FROM ALTLIB REQUESTED
		...1		LDESBIGL	BIG LIBRARY, MAX. 32 VSAM EXTS
36	(24)	CHARACTER	4	LDESLEVL	VERSION LEVEL
40	(28)	SIGNED	2	LDESLNG	LENGTH OF LIBRARY DESCRIPTOR
42	(2A)	SIGNED	2	LDESLBCF	LENGTH OF LB CONTROL INFORMATION FIELD
44	(2C)	UNSIGNED	1	LDESLBSZ	LIBRARY BLOCK SIZE
45	(2D)	CHARACTER	3	*	RESERVED
48	(30)	CHARACTER	8	LDESCMPR	NAME OF COMPRESS ROUTINE
56	(38)	SIGNED	2	LDESNOSL	# OF SUBLIBRARIES
58	(3A)	CHARACTER	6	LDESPRBA	PRBA OF SUBLIBRARY INDEX
58	(3A)	SIGNED	2	LDESSXOS	- OFFSET OF SUBLIBRARY INDEX START
60	(3C)	SIGNED	4	LDESSXBN	- LB-# OF SUBLIBRARY INDEX START
64	(40)	SIGNED	2	LDESNOLB	# OF LBS FOR SUBLIBRARY DIRECTORY
66	(42)	CHARACTER	2	*	RESERVED
68	(44)	SIGNED	4	LDESIDEN	RESERVED
72	(48)	SIGNED	4	LDESNLCK	# OF LOCKED MEMBERS
76	(4C)	CHARACTER	140	*	RESERVED

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCLDES	0		1
LDESALT	23		3
LDESALTR	23		3
LDESBIGL	23	10	3
LDESCMPR	30		2
LDESCRDT	18		2
LDESCRID	10		2
LDESFLAG	23		2
LDESIDEN	44		2
LDESLABL	0		2
LDESLBCF	2A		2
LDESLBSZ	2C		2
LDESLEVL	24		2
LDESLNG	28		2
LDESNAME	8		2
LDESNLCK	48		2
LDESNOLB	40		2
LDESNOSL	38		2
LDESPRBA	3A		2
LDESRCLM	23		3
LDESSXBN	3C		3
LDESSXOS	3A		3

INLCLDTE

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	36	INLCLDTE	LIBRARY DEFINITION TABLE
0	(0)	CHARACTER	7	LDTEFNME	LIBRARY FILE NAME
7	(7)	CHARACTER	12	LDTERNME	RESOURCE NAME
7	(7)	CHARACTER	1	LDTENTID	ENTRY ID
8	(8)	CHARACTER	6	LDTEVOLS	PACK VOLID
8	(8)	A-ADDRESS	4	LDTEPADR	PTR TO ANOTHER LDT ENTRY
12	(C)	CHARACTER	2		RESERVED
14	(E)	A-ADDRESS	4	LDTELADR	LIBRARY START ADDRESS
18	(12)	CHARACTER	1		RESERVED
19	(13)	UNSIGNED	1	LDTELBSZ	LB SIZE
20	(14)	SIGNED	2	LDTELBCF	LENGTH OF LB CNT.FIELD
22	(16)	BITSTRING	1		RESERVED
23	(17)	BITSTRING	1	LDTEUACC	RESERVED
24	(18)	BITSTRING	2	LDTEPART	PARTITION FLAGS
26	(1A)	BITSTRING	1	LDTEINFO	LIBR I N F O B Y T E
		1...		LDTESYST	SYSTEM LIBRARY
		.1..		LDTESHRD	LIBR ON SHARED DASD AND SV WITH DASD SHARING SUPP.
		..1.		LDTERCLM	IMMED. SPACE RECL.@D14LDFB
		...1 1...			RESERVED
		.... .1..		LDTEPROT	LIBRARY PROTECTED
		.... ..1.			
		1		LDTEVSAM	LIB. VSAM MANAG.
27	(1B)	BITSTRING	1	LDTEFLAG	LIBR F L A G B Y T E
		1...		LDTENOAC	NO ACCESS ALLOWED
		.1..		LDTEDELT	LIB IS DELETED
		..11 11..			RESERVED
		.... ..1.		LDTEBIGL	BIG LIBRARY
		1		LDTENEWL	NEW LIBRARY
28	(1C)	A-ADDRESS	4	LDTESDTX	ADDR OF SDT
32	(20)	A-ADDRESS	4	LDTEEDTX	ADDR OF EDT

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCLDTE	0	(0)	
LDTEBIGL	1A	X'02'	
LDTEDELT	27	X'40'	
LDTEEDTX	32	(20)	
LDTEFLAG	27	(1B)	
LDTEFNME	0	(0)	
LDTEINFO	26	(1A)	
LDTELADR	14	(E)	
LDTELBCF	20	(14)	
LDTELBSZ	19	(13)	
LDTENEWL	27	X'01'	
LDTENOAC	27	X'80'	
LDTENTID	7	(7)	
LDTEPADR	8	(8)	
LDTEPART	24	(18)	
LDTEPROT	26	X'04'	
LDTERCLM	26	X'20'	
LDTERNME	7	(7)	
LDTESDTX	28	(1C)	
LDTESHRD	26	X'40'	
LDTESYST	26	X'80'	
LDTEUACC	23	(17)	
LDTEVOLS	8	(8)	
LDTEVSAM	26	X'01'	

INLCLOT

```
LIBRARIAN - INLCLOT - 5666-301-06-H07 - VERSION 82-01-05
     L I B R A R Y  O F F S E T  T A B L E
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
     DO NOT USE FOR TASK LOT ROW
     (USE INLCLOTP INSTEAD)
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	36	INLCLOT	LIBRARY OFFSET TABLE
0	(0)	CHARACTER	6	LOTROW	BEGIN OF OFFSET ROW
0	(0)	CHARACTER	4	LOTBEGIN	ROW START INDICATOR
4	(4)	BITSTRING	1	LOTFLG1	RESERVED
5	(5)	BITSTRING	1	LOTFLG2	ROW INFORMATION BITS
6	(6)	CHARACTER	6	LOTCAT	LOT ENTRY "CATALOG"
6	(6)	BITSTRING	2	LOTCATLO	OFFSET INTO LDT
8	(8)	BITSTRING	2	LOTCATSO	OFFSET INTO SDT
10	(A)	BITSTRING	2	LOTCATFL	RESERVED
12	(C)	CHARACTER	6	LOTNEW	RESERVED
12	(C)	BITSTRING	2	LOTNEWLO	OFFSET INTO LDT
14	(E)	BITSTRING	2	LOTNEWSO	OFFSET INTO SDT
16	(10)	BITSTRING	2	LOTNEWFL	RESERVED
18	(12)	CHARACTER	6	LOTFROM	LOT ENTRY FOR "FROM"
18	(12)	BITSTRING	2	LOTFRMLO	OFFSET INTO LDT
20	(14)	BITSTRING	2	LOTFRMSO	OFFSET INTO SDT
22	(16)	BITSTRING	2	LOTFRMFL	RESERVED
24	(18)	CHARACTER	6	LOTTO	LOT ENTRY FOR "TO"
24	(18)	BITSTRING	2	LOTTOLO	OFFSET INTO LDT
26	(1A)	BITSTRING	2	LOTTOSO	OFFSET INTO SDT
28	(1C)	BITSTRING	2	LOTTOFL	RESERVED
30	(1E)	CHARACTER	6	LOTSRCH	LOT ENTRY "SEARCH"
30	(1E)	BITSTRING	2	LOTSRCLO	OFFSET INTO LDT
32	(20)	BITSTRING	2	LOTSRCSO	OFFSET INTO SDT
34	(22)	BITSTRING	2	LOTSRCFL	RESERVED

INLCLOTP

```
!!! INLCLOTP: USE FOR TASK LOT ROW ONLY !!!!!!!!!!!!!!!
```

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	42	INLCLOTP	L B R OFFSET TABLE - TASK
0	(0)	CHARACTER	12	LOTPHEAD	TASK CHAIN ENTRY HDR
0	(0)	ADDRESS	4	LOTPNEXT	NEXT IN CHAIN
4	(4)	CHARACTER	8	LOTPCHID	TASK LOT IDENTIFIER
12	(C)	CHARACTER	18	LOTPUSER	TASK LOT ROW
12	(C)	CHARACTER	6	LOTPROW	LOT ROW START ENTRY
12	(C)	CHARACTER	4	LOTPBEGN	ROW START INDICATOR
16	(10)	BITSTRING	1	LOTPFLG1	RESERVED
17	(11)	BITSTRING	1	LOTPFLG2	ROW INFORMATION BITS
18	(12)	CHARACTER	6	LOTPTMPE	LOT ENTRY DYNAMIC "USER"
18	(12)	BITSTRING	2	LOTPTMPL	OFFSET INTO LDT
20	(14)	BITSTRING	2	LOTPTMPS	OFFSET INTO SDT
22	(16)	BITSTRING	2	LOTPTMPF	RESERVED
24	(18)	CHARACTER	6	LOTPAUXE	LOT ENTRY INTERMED. "USER"
24	(18)	BITSTRING	2	LOTPAUXL	OFFSET INTO LDT
26	(1A)	BITSTRING	2	LOTPAUXS	OFFSET INTO SDT

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
28	(1C)	BITSTRING	2	LOTPAUXF	RESERVED
30	(1E)	CHARACTER	6	*	RESERVED
36	(24)	CHARACTER	6	*	RESERVED
36	(24)	CHARACTER	6	LOTPPRME	LOT ENTRY CHAIN "USER"
36	(24)	BITSTRING	2	LOTPPRML	OFFSET INTO LDT
38	(26)	BITSTRING	2	LOTPPRMS	OFFSET INTO SDT
40	(28)	BITSTRING	2	LOTPPRMF	RESERVED

INLCLOTA

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	36	INLCLOTA	L B R OFFSET TABLE - ACCESS
0	(0)	CHARACTER	6	LOTROWA	LOT ROW START ENTRY
0	(0)	CHARACTER	4	LOTBEGNA	ROW START INDICATOR
4	(4)	BITSTRING	1	LOTFLG1A	RESERVED
5	(5)	BITSTRING	1	LOTFLG2A	ROW INFORMATION BITS
6	(6)	CHARACTER	6	LOTACCD	LOT ENTRY DYNAMIC "ACCESS"
6	(6)	BITSTRING	2	LOTACDLO	OFFSET INTO LDT
8	(8)	BITSTRING	2	LOTACDSO	OFFSET INTO SDT
10	(A)	BITSTRING	2	LOTACDFL	RESERVED
12	(C)	CHARACTER	6	*	RESERVED
18	(12)	CHARACTER	6	*	RESERVED
24	(18)	CHARACTER	6	*	RESERVED
30	(1E)	CHARACTER	6	LOTACC	LOT ENTRY "ACCESS"
30	(1E)	BITSTRING	2	LOTACCLO	OFFSET INTO LDT
32	(20)	BITSTRING	2	LOTACCSO	OFFSET INTO SDT
34	(22)	BITSTRING	2	LOTACCFL	RESERVED

INLCLOTS

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	36	INLCLOTS	L B R OFFSET TABLE - SEARCH
0	(0)	CHARACTER	6	LOTROWS	LOT ROW START ENTRY
0	(0)	CHARACTER	4	LOTBEGNS	ROW START INDICATOR
4	(4)	BITSTRING	1	LOTFLG1S	RESERVED
5	(5)	BITSTRING	1	LOTFLG2S	ROW INFORMATION BITS
6	(6)	CHARACTER	6	LOTSRCD	LOT ENTRY DYNAMIC "SEARCH"
6	(6)	BITSTRING	2	LOTSRDLO	OFFSET INTO LDT
8	(8)	BITSTRING	2	LOTSRDSO	OFFSET INTO SDT
10	(A)	BITSTRING	2	LOTSRDFL	RESERVED
12	(C)	CHARACTER	6	*	RESERVED
18	(12)	CHARACTER	6	*	RESERVED
24	(18)	CHARACTER	6	*	RESERVED
30	(1E)	CHARACTER	6	LOTSRCL	LOT ENTRY "SEARCH"
30	(1E)	BITSTRING	2	LOTSRCLL	OFFSET INTO LDT
32	(20)	BITSTRING	2	LOTSRCLS	OFFSET INTO SDT
34	(22)	BITSTRING	2	LOTSRCLF	RESERVED

INLCLOTC

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	36	INLCLOTC	L B R OFFSET TABLE - CATALOG
0	(0)	CHARACTER	6	LOTROWC	LOT ROW START ENTRY
0	(0)	CHARACTER	4	LOTBEGNC	ROW START INDICATOR
4	(4)	BITSTRING	1	LOTFLG1C	RESERVED
5	(5)	BITSTRING	1	LOTFLG2C	ROW INFORMATION BITS
6	(6)	CHARACTER	6	LOTCATD	LOT ENTRY DYNAMIC "CATALOG"
6	(6)	BITSTRING	2	LOTCADLO	OFFSET INTO LDT
8	(8)	BITSTRING	2	LOTCADSO	OFFSET INTO SDT

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
10	(A)	BITSTRING	2	LOTCADFL	RESERVED
12	(C)	CHARACTER	6	*	RESERVED
18	(12)	CHARACTER	6	*	RESERVED
24	(18)	CHARACTER	6	*	RESERVED
30	(1E)	CHARACTER	6	LOTCATL	LOT ENTRY "SEARCH"
30	(1E)	BITSTRING	2	LOTCATLL	OFFSET INTO LDT
32	(20)	BITSTRING	2	LOTCATLS	OFFSET INTO SDT
34	(22)	BITSTRING	2	LOTCATLF	RESERVED

LIBRARIAN - INLCLOT - 5666-301-06-H07 - VERSION 03/19/1981
L I B R A R Y O F F S E T E N T R Y

LOTENTRY

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	6	LOTENTRY	ENTRY OF LIBR. OFFSET TABLE
0	(0)	CHARACTER	6	LOTENTR	ENTRY LOT
0	(0)	SIGNED	4	LOTETLPT	TASK LOT POINTER
0	(0)	BITSTRING	2	LOTEOLDT	OFFSET IN LDT
2	(2)	BITSTRING	2	LOTEOSDT	OFFSET IN SDT
4	(4)	SIGNED	2	LOTETIK	TASK LOT POINTER
4	(4)	BITSTRING	2	LOTEFLAG	RESERVED
4	(4)	BITSTRING	1	LOTIREQ	INDIVIDUAL REQUEST
		1...		LOTIFREE	IMMEDIATE SPACE RECL.@D14LDFB
5	(5)	BITSTRING	1	LOTIACC	INDIV. ACCESS RIGHTS
		1...		LOTICCF	RESERVED
		.111		*	RESERVED
		.... 1...		LOTMIGRT	MIGRATION "LIBDEF"
		.... .1..		LOTRACT	ACTIVE ENTRIES IN ROW
		.... ..1.		LOTASSGN	ENTRIES INSERTED DUE TO ASSGN
		1		LOTCHPR	CHAIN IN PROCESS

Cross Reference

Name	Hex Offset	Hex Value	Level
INLCLOT	0		1
INLCLOTA	0		1
INLCLOTC	0		1
INLCLOTP	0		1
INLCLOTS	0		1
LOTACC	1E		2
LOTACCD	6		2
LOTACCFL	22		3
LOTACCLO	1E		3
LOTACCSO	20		3
LOTACDFL	A		3
LOTACDLO	6		3
LOTACDSO	8		3
LOTASSGN	5	02	6
LOTBEGIN	0		3
LOTBEGNA	0		3
LOTBEGNC	0		3
LOTBEGNS	0		3
LOTCADFL	A		3
LOTCADLO	6		3
LOTCADSO	8		3
LOTCAT	6		2
LOTCATD	6		2
LOTCATFL	A		3
LOTCATL	1E		2
LOTCATLF	22		3

Name	Hex Offset	Hex Value	Level
LOTCATLL	1E		3
LOTCATLO	6		3
LOTCATLS	20		3
LOTCATSO	8		3
LOTCHPR	5	01	6
LOTEFLAG	4		4
LOTENTR	0		2
LOTENTRY	0		1
LOTEOLDT	0		4
LOTEOSDT	2		4
LOTETIK	4		3
LOTETLPT	0		3
LOTFLG1	4		3
LOTFLG1A	4		3
LOTFLG1C	4		3
LOTFLG1S	4		3
LOTFLG2	5		3
LOTFLG2A	5		3
LOTFLG2C	5		3
LOTFLG2S	5		3
LOTFRMFL	16		3
LOTFRMLO	12		3
LOTFRMSO	14		3
LOTFROM	12		2
LOTIACC	5		5
LOTICCF	5	80	6
LOTIFREE	4	80	6
LOTIREQ	4		5
LOTMIGRT	5	08	6
LOTNEW	C		2
LOTNEWFL	10		3
LOTNEWLO	C		3
LOTNEWSO	E		3
LOTPAUXE	18		3
LOTPAUXF	1C		4
LOTPAUXL	18		4
LOTPAUXS	1A		4
LOTPBEGN	C		4
LOTPCHID	4		3
LOTPFLG1	10		4
LOTPFLG2	11		4
LOTPHEAD	0		2
LOTPNEXT	0		3
LOTPPRME	24		3
LOTPPRMF	28		4
LOTPPRML	24		4
LOTPPRMS	26		4
LOTPROW	C		3
LOTPTMPE	12		3
LOTPTMPF	16		4
LOTPTMPL	12		4
LOTPTMPS	14		4
LOTPUSER	C		2
LOTRACT	5	04	6
LOTROW	0		2
LOTROWA	0		2
LOTROWC	0		2
LOTROWS	0		2
LOTSRCD	6		2
LOTSRCFL	22		3
LOTSRCH	1E		2
LOTSRCL	1E		2
LOTSRCLF	22		3
LOTSRCLL	1E		3
LOTSRCLO	1E		3

Name	Hex Offset	Hex Value	Level
LOTSRCLS	20		3
LOTSRCSO	20		3
LOTSRDFL	A		3
LOTSRDLO	6		3
LOTSRDSO	8		3
LOTTO	18		2
LOTTOFL	1C		3
LOTTOLO	18		3
LOTTOSO	1A		3

INLCLPT

LIBRARY POINTER TABLE

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	303	INLCLPT	LIBRARY POINTER TABLE
0	(0)	CHARACTER	12	LPTHEADR	HEADER POINTER TABLE
0	(0)	ADDRESS	4	LPT$ADDR	ADDR CONCAT SERVICES
4	(4)	SIGNED	2	LPTNSRCH	NR OF SEARCH LIBRARIES
6	(6)	SIGNED	2	LPTNENTR	NR OF LOT ENTRIES / ROW
8	(8)	SIGNED	2	LPTNSUB	NR OF SUBLIBS
10	(A)	SIGNED	2	LPTNTASK	NR OF TASK ENTR.
12	(C)	CHARACTER	12	LPTLDT	PTR TO LDT
12	(C)	ADDRESS	4	LPTBLDT	BEGIN ADDR OF LDT
16	(10)	ADDRESS	4	LPTELDT	END ADDR OF LDT
20	(14)	ADDRESS	4	LPTFLDT	NEXT FREE ENTRY
24	(18)	CHARACTER	12	LPTSDT	PTR TO SDT
24	(18)	ADDRESS	4	LPTBSDT	BEGIN ADDR OF SDT
28	(1C)	ADDRESS	4	LPTESDT	END ADDR OF SDT
32	(20)	ADDRESS	4	LPTFSDT	NEXT FREE ENTRY
36	(24)	CHARACTER	12	LPTIDT	PTR TO IDT
36	(24)	ADDRESS	4	LPTBIDT	BEGIN ADDR OF IDT
40	(28)	ADDRESS	4	LPTEIDT	END ADDR OF IDT
44	(2C)	ADDRESS	4	LPTFIDT	NEXT FREE ENTRY
48	(30)	CHARACTER	12	LPTEDT	PTR TO EDT
48	(30)	ADDRESS	4	LPTBEDT	BEGIN ADDR OF EDT
52	(34)	ADDRESS	4	LPTEEDT	END ADDR OF EDT
56	(38)	ADDRESS	4	LPTFEDT	NEXT FREE ENTRY
60	(3C)	CHARACTER	12	LPTDDT	PTR TO DDT
60	(3C)	ADDRESS	4	LPTBDDT	BEGIN ADDR OF DDT
64	(40)	ADDRESS	4	LPTEDDT	END ADDR OF DDT
68	(44)	ADDRESS	4	LPTFDDT	NEXT FREE ENTRY
72	(48)	CHARACTER	12	LPTROW1	RESERVED
72	(48)	CHARACTER	12	*	
84	(54)	CHARACTER	12	LPTROW2	RESERVED
84	(54)	CHARACTER	12	*	
96	(60)	CHARACTER	12	LPTROW3	RESERVED
96	(60)	CHARACTER	12	*	
108	(6C)	CHARACTER	12	LPTROW4	RESERVED
108	(6C)	CHARACTER	12	*	
120	(78)	CHARACTER	12	LPTLBR	LIBRARIAN LOT
120	(78)	ADDRESS	4	LPTLLOTA	ACCESS CHAIN
124	(7C)	ADDRESS	4	LPTLLOTS	SEARCH CHAIN
128	(80)	ADDRESS	4	LPTLLOTC	CATALOG CHAIN
132	(84)	CHARACTER	12	LPTROW5	RESERVED
132	(84)	CHARACTER	12	*	
144	(90)	CHARACTER	12	LPTIALOT	I/A LOT POOL
144	(90)	ADDRESS	4	LPTBILOT	BEGIN LOT POOL
148	(94)	ADDRESS	4	LPTEILOT	END LOT POOL
152	(98)	ADDRESS	4	LPTFILOT	FREE LOT POOL ENT.@D14CDFB
156	(9C)	ADDRESS	4	LPTLANC	ADDRESS OF LANC
160	(A0)	CHARACTER	8	*	RESERVED
168	(A8)	CHARACTER	12	*	RESERVED
180	(B4)	ADDRESS	4	LPTVIOTB	VIO POINTER TABLE
184	(B8)	CHARACTER	8	*	RESERVED
192	(C0)	CHARACTER	8	LPTPOLID	SLD - SUBPOOL ID.
200	(C8)	CHARACTER	2	*	RESERVED
202	(CA)	BITSTRING	2	LPTTRACE	TRACE CONTROL
202	(CA)	BITSTRING	1	LPTTRLV1	TRACE LEVEL 1 REQ
		1...		LPTALLV1	TRACE ALL LEVEL 1
		.111		*	RESERVED
		.... 1...		LPTTRBUF	TRACE BUFFER REQ.

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
		.... .1..		LPTTRSPA	TRACE SPACE REQ.
		.... ..1.		LPTTRIO	TRACE I/O REQ.
		1		*	RESERVED
203	(CB)	BITSTRING	1	LPTTRLV2	TRACE LEVEL 2 REQ
		1...		LPTALLV2	TRACE ALL LEVEL 2
		.111 111.		*	RESERVED
		1		LPTTRUNT	TRACE UNIT:SYSLST
204	(CC)	CHARACTER	9	SCALEL (11)	

Cross Reference

Name	Hex Offset	Hex Value	Level
INLCLPT	0		1
LPT$ADDR	0		3
LPTALLV1	CA	80	4
LPTALLV2	CB	80	4
LPTBDDT	3C		3
LPTBEDT	30		3
LPTBIDT	24		3
LPTBILOT	90		3
LPTBLDT	C		3
LPTBSDT	18		3
LPTDDT	3C		2
LPTEDDT	40		3
LPTEDT	30		2
LPTEEDT	34		3
LPTEIDT	28		3
LPTEILOT	94		3
LPTELDT	10		3
LPTESDT	1C		3
LPTFDDT	44		3
LPTFEDT	38		3
LPTFIDT	2C		3
LPTFILOT	98		3
LPTFLDT	14		3
LPTFSDT	20		3
LPTHEADR	0		2
LPTIALOT	90		2
LPTIDT	24		2
LPTLANC	9C		2
LPTLBR	78		2
LPTLDT	C		2
LPTLLOTA	78		3
LPTLLOTC	80		3
LPTLLOTS	7C		3
LPTNENTR	6		3
LPTNSRCH	4		3
LPTNSUB	8		3
LPTNTASK	A		3
LPTPOLID	C0		2
LPTROW1	48		2
LPTROW2	54		2
LPTROW3	60		2
LPTROW4	6C		2
LPTROW5	84		2
LPTSDT	18		2
LPTTRACE	CA		2
LPTTRBUF	CA	08	4
LPTTRIO	CA	02	4
LPTTRLV1	CA		3
LPTTRLV2	CB		3
LPTTRSPA	CA	04	4
LPTTRUNT	CB	01	4

Name	Hex Offset	Hex Value	Level
LPTVIOTB	B4		2
SCALEL	CC		2

INLCLRPL

LIBRARIANREQUESTPARMLIST

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	124	INLCLRPL	
0	(0)	CHARACTER	8	LRPLHDR	HEADER FOR INLCLRPL
0	(0)	CHARACTER	4	LRPLID	- LRPL-ID
4	(4)	BITSTRING	1	LRPLLVL	- VERSION
5	(5)	BITSTRING	1	*	- RESERVED
6	(6)	SIGNED	2	LRPLLEN	- LENGTH OF CONTROL BLOCK
8	(8)	ADDRESS	4	LRPLINFO	ADDRESS OF LIBINFO
12	(C)	ADDRESS	4	LRPLARG	ADDRESS OF ARGUMENT
16	(10)	ADDRESS	4	LRPLWAEA	ADDRESS OF WORKAREA
20	(14)	SIGNED	4	LRPLLNWA	LENGTH OF WORKAREA
24	(18)	ADDRESS	4	LRPLLAMB	ADDRESS OF LAMB
28	(1C)	ADDRESS	4	LRPLPBUF	ADDRESS OF PRIVATE BUFFER
32	(20)	SIGNED	4	LRPLLBUF	LENGTH OF PRIVATE BUFFER AREA
36	(24)	ADDRESS	4	LRPLLBTB	ADDRESS OF LIBRARY CHAIN TABLE
40	(28)	ADDRESS	4	LRPLSTTB	ADDRESS OF STOWTABLE
44	(2C)	SIGNED	4	LRPLSTLN	LENGTH OF STOWAREA
48	(30)	ADDRESS	4	LRPLRQCB	ADDRESS OF ACCESS CONTROL BLOCK
52	(34)	BITSTRING	1	LRPLROBJ	ACCESSED LIBRARY OBJECT
		1...		LRPLLIB	- LIBRARY LEVEL
		.1..		LRPLSLIB	- SUBLIBRARY LEVEL
		..1.		LRPLMBR	- MEMBER LEVEL
53	(35)	BITSTRING	3	LRPLRQTY	REQUEST TYPE
		1...		LRPLGETR	- GET MBR RECORD
		.1..		LRPLPUTR	- PUT MBR RECORD
		..1.		LRPLNOTE	- NOTE MBR RECORD
		...1		LRPLPOIN	- POINT MBR RECORD
		.... 1...		LRPLGDIR	- GET DIRECTORY RECORD
		.... .1..		LRPLSTOW	- STOW DIRECTORY RECORD
		.... ..1.		LRPLLOCK	- LOCK/UNLOCK MEMBER
		1		*	- RESERVED
		1...		LRPLLCON	- CONNECT TO LIBRARY
		.1..		LRPLSCON	- CONNECT TO SUBLIBRARY
		..1.		LRPLMCON	- CONNECT TO MEMBER
		...1		*	- RESERVED
		.... 1...		LRPLLDIS	- DISCONNECT FROM LIBRARY
		.... .1..		LRPLSDIS	- DISCONNECT FROM SUBLIBRARY
		.... ..1.		LRPLMDIS	- DISCONNECT FROM MEMBER
		1		*	- RESERVED
		1...		LRPLBLDL	- BUILD LIST
		.1..		LRPLFIND	- FIND MBR
		..11		*	- RESERVED
		.... 1...		LRPLBSLD	- BUILD SLD
		.... .1..		LRPLRCLM	- RECLAIM SPACE
		11		*	- RESERVED
56	(38)	BITSTRING	1	LRPLLUSE	LIBRARY USAGE
		1...		LRPLSRCH	- SEARCH CHAIN
		.1..		LRPLTO	- TO CHAIN
		..1.		LRPLFROM	- FROM CHAIN
		...1		LRPLACC	- ACCESS CHAIN
		.... 1...		LRPLCAT	- CATALOG CHAIN
		.... .11.		*	- RESERVED
		1		LRPLUSER	- USER-DEFINED CHAIN
57	(39)	BITSTRING	3	LRPLOPCD	OPTION CODE
		1...		LRPLLEVL	- MEMBER/SUBLIBRARY LEVEL
		.1..		LRPLENTR	- ENTRY INFORMATION
		..1.		LRPLRESO	- RESOURCE NAME INFORMATION
		...1		LRPLPURP	- UPDATE/RENAME PURPOSE

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
		.... 1...		LRPLNEW	- OLD/NEW CONNECTION
		.... .1..		LRPLREPL	- REPLACE/NOREPLACE OLD ENTRY
		.... ..1.		LRPLGEN	- NONGENERIC/GENERIC REQUEST
		1		LRPLCONT	- NONCONT./CONTINUOUS REQUEST
		1...		LRPLADDR	- SEQUENTIAL/ADDRESSED ACCESS
		.1..		LRPLMODE	- MOVE/LOCATE MODE
		..1.		LRPLCOMP	- DECOMPRESSED/COMPRESSED STATUS
		...1		LRPLDESC	- LIBRARY LEVEL
		.... 1...		LRPLCNTD	- CONTINUED REQUEST
		.... .1..		LRPLCHCK	- CHECK REQUEST
		.... ..1.		LRPLPREV	- NOTE PREVIOUS RECORD
		1		LRPLCMPR	- RECORD STATE: COMPRESSED
		1...		LRPLRCSP	- RECLAIM SPACE IMMEDIATE
		.1..		LRPLRCFM	- RECORD FORMAT: VARIABLE
		..1.		LRPLUPD	- CONNECT=UPDATE
		...1		*	- RESERVED
		.... 1...		LRPLDATE	- DATE: 0 = NEW, 1 = OLD
		.... .1..		LRPLBIGL	- BIGL: 0 = NO, 1 = YES
		11		*	- RESERVED
60	(3C)	ADDRESS	4	LRPLLRBA	LOGICAL RELATIVE BYTE ADDRESS
64	(40)	SIGNED	2	LRPLLRNO	# OF LOGICAL RECORDS
66	(42)	SIGNED	2	LRPLRECL	LOGICAL RECORD LENGTH
68	(44)	UNSIGNED	1	LRPLFDBC	FEEDBACK CODES
69	(45)	BITSTRING	1	LRPLTYPE	LIBRARY TYPE
70	(46)	BITSTRING	1	*	RESERVED
71	(47)	BITSTRING	1	LRPLFLAG	FLAG BYTE
		1...		LRPLPBIN	- BUFFER INITIALIZED
		.1..		LRPLDELW	- DELAYED WRITE REQUEST
		..1.		LRPLAPIC	- API REQUEST LRPL
		...1 111.		*	- RESERVED
		1		LRPLNOLK	- NO LOCK MUST BE GIVEN
72	(48)	SIGNED	4	LRPLMVLN	MOVED LENGTH
76	(4C)	ADDRESS	4	LRPLDENT	RECEIVER OF DIRECTORY ENTRY
80	(50)	ADDRESS	4	LRPLCHLS	CHECK LIST ADDRESS
84	(54)	SIGNED	4	LRPLSAVE	INTERMEDIATE DIR. ENTRY
88	(58)	CHARACTER	8	LRPLSPID	SUBPOOL ID FOR CONTROL BLOCKS
96	(60)	ADDRESS	4	LRPLCHID	ADDRESS OF CHAIN IDENTIFIER
100	(64)	ADDRESS	4	LRPLSLXE	ADDRESS OF SUBLIB DESCRIPTOR
104	(68)	ADDRESS	4	LRPLTIMP	ADDRESS OF DATEVALUE
108	(6C)	ADDRESS	4	LRPLLKID	ADDRESS OF LOCKID
112	(70)	CHARACTER	1	LRPLLCKF	LOCKINGFLAGS
		1...		LRPLLVID	LRPLLKID VALID
		.1..		LRPLLCPY	1 IF LOCKINF. COPIED
		..1.		LRPLLFRC	RESERVED
		...1		LRPLLUL	1 -> LOCK,0 -> UNLOCK
		 1111		*	RESERVED
113	(71)	CHARACTER	1	LRPLSEC	RESERVED
		1...		LRPLKLIK	CLICK ACCESS
		.111 1111		*	RESERVED
114	(72)	CHARACTER	2	*	RESERVED
116	(74)	CHARACTER	8	*	RESERVED

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	12	INLCCHKL (*)	
0	(0)	SIGNED	4	CHKLLRBA	LRBA
4	(4)	CHARACTER	6	CHKLPRBA	PRBA
4	(4)	SIGNED	2	CHKLRBO	- OFFSET IN LB
6	(6)	SIGNED	4	CHKLRBN	- LB NUMBER
10	(A)	SIGNED	2	*	RESERVED

Cross Reference:

Name	Hex Offset	Hex Value	Level
CHKLLRBA	0		2
CHKLPRBA	4		2
CHKLRBN	6		3
CHKLRBO	4		3
INLCCHKL	0		1
INLCLRPL	0		1
LRPLACC	38		3
LRPLADDR	3A		3
LRPLAPIC	47		3
LRPLARG	C		2
LRPLBIGL	3B	04	3
LRPLBLDL	37		3
LRPLBSLD	37		3
LRPLCAT	38		3
LRPLCHCK	3A		3
LRPLCHID	60		2
LRPLCHLS	50		2
LRPLCMPR	3A		3
LRPLCNTD	3A		3
LRPLCOMP	3A		3
LRPLCONT	39		3
LRPLDATE	3B		3
LRPLDELW	47		3
LRPLDENT	4C		2
LRPLDESC	3A		3
LRPLENTR	39		3
LRPLFDBC	44		2
LRPLFIND	37		3
LRPLFLAG	47		2
LRPLFROM	38		3
LRPLGDIR	35		3
LRPLGEN	39		3
LRPLGETR	35		3
LRPLHDR	0		2
LRPLID	0		3
LRPLINFO	8		2
LRPLKLIK	71		3
LRPLLAMB	18		2
LRPLLBTB	24		2
LRPLLBUF	20		2
LRPLLCKF	70		2
LRPLLCON	36		3
LRPLLCPY	70		3
LRPLLDIS	36		3
LRPLLEN	6		3
LRPLLEVL	39		3
LRPLLFRC	70		3
LRPLLIB	34		3
LRPLLKID	6C		2
LRPLLNWA	14		2
LRPLLOCK	35		3
LRPLLRBA	3C		2
LRPLLRNO	40		2
LRPLLUL	70		3
LRPLLUSE	38		2
LRPLLVID	70		3
LRPLLVL	4		3
LRPLMBR	34		3
LRPLMCON	36		3
LRPLMDIS	36		3
LRPLMODE	3A		3
LRPLMVLN	48		2

Name	Hex Offset	Hex Value	Level
LRPLNEW	39		3
LRPLNOLK	47		3
LRPLNOTE	35		3
LRPLOPCD	39		2
LRPLPBIN	47		3
LRPLPBUF	1C		2
LRPLPOIN	35		3
LRPLPREV	3A		3
LRPLPURP	39		3
LRPLPUTR	35		3
LRPLRCFM	3B		3
LRPLRCLM	37		3
LRPLRCSP	3B		3
LRPLRECL	42		2
LRPLREPL	39		3
LRPLRESO	39		3
LRPLROBJ	34		2
LRPLRQCB	30		2
LRPLRQTY	35		2
LRPLSAVE	54		2
LRPLSCON	36		3
LRPLSDIS	36		3
LRPLSEC	71		2
LRPLSLIB	34		3
LRPLSLXE	64		2
LRPLSPID	58		2
LRPLSRCH	38		3
LRPLSTLN	2C		2
LRPLSTOW	35		3
LRPLSTTB	28		2
LRPLTIMP	68		2
LRPLTO	38		3
LRPLTYPE	45		2
LRPLUPD	3B		3
LRPLUSER	38		3
LRPLWAEA	10		2

INLCLSIM

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	20	INLCLSIM	
0	(0)	CHARACTER	8	LSIMID	CONTROL BLOCK PASSED
8	(8)	A-ADDRESS	4	LSIMCB	ADDRESS OF CONTROL BLOCK
12	(C)	A-ADDRESS	4	LSIMLAMB	ADDRESS OF LAMB
16	(10)	BITSTRING	4	LSIMRQL	ACCESS REQUEST
16	(10)	BITSTRING	1	LSIMAUT	REQUIRED AUTHORIZATION
17	(11)	BITSTRING	1	LSIMTYP	OBJECT TYPE
18	(12)	BITSTRING	1	LSIMOAT	OWNED ACCESS RIGHT

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCLSIM	0	(0)	
LSIMAUT	16	(10)	
LSIMCB	8	(8)	
LSIMID	0	(0)	
LSIMLAMB	12	(C)	
LSIMOAT	18	(12)	
LSIMRQL	16	(10)	
LSIMTYP	17	(11)	

INLCMACB

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	112	INLCMACB	MEMBER ACCESS CONTROL BLOCK
0	(0)	CHARACTER	8	MACBHDR	MACB HEADER "INLCMACB"
8	(8)	ADDRESS	4	MACBSACB	ADDR OF SACB
12	(C)	SIGNED	4	MACBLRBA	LRBA OF MEMBER RECORD
16	(10)	CHARACTER	6	MACBPRBA	PRBA OF MEMBER RECORD
16	(10)	UNSIGNED	2	MACBPROS	- OFFSET IN LB
18	(12)	SIGNED	4	MACBPRBN	- LIBRARY BLOCK #
22	(16)	SIGNED	2	MACBCREM	CONTIGOUS REMAINDER
24	(18)	SIGNED	4	MACBPLRA	LRBA OF PREVIOUS LB
28	(1C)	CHARACTER	6	MACBPPRA	PRBA OF PREVIOUS LB
28	(1C)	UNSIGNED	2	MACBPPOS	- OFFSET IN LB
30	(1E)	SIGNED	4	MACBPPBN	- LIBRARY BLOCK #
34	(22)	CHARACTER	6	MACBHRBA	PRBA OF HIGHEST LB
34	(22)	UNSIGNED	2	MACBHPOS	- OFFSET IN LB
36	(24)	SIGNED	4	MACBHPBN	- LIBRARY BLOCK #
40	(28)	SIGNED	4	MACBHLRA	LRBA OF HIGHEST LB
44	(2C)	CHARACTER	6	MACBMRBA	PRBA OF MEMBER START LB
44	(2C)	UNSIGNED	2	MACBMPOS	- OFFSET IN LB
46	(2E)	SIGNED	4	MACBMPBN	- LIBRARY BLOCK #
50	(32)	SIGNED	2	MACBCWAA	OFFSET IN CALLER WORKAREA
52	(34)	CHARACTER	8	MACBBREQ	REQUEST BLOCK FOR BUF-MGMT
52	(34)	ADDRESS	4	MACBBUCB	-ADDR BUFFER CONTROL BLOCK
56	(38)	ADDRESS	4	MACBBANC	- BUFFER LIST ANCHOR
60	(3C)	ADDRESS	4	MACBCBUF	ADDR OF CURRENT BUFFER HDR
64	(40)	CHARACTER	8	MACBIREQ	REQUEST BLOCK FOR I/O PROC.
64	(40)	ADDRESS	4	MACBIUCB	- ADDR BUFFER CONTROL BLOCK
68	(44)	ADDRESS	4	MACBIANC	- I/O LIST ANCHOR
72	(48)	ADDRESS	4	MACBMDIR	ADDRESS OF DIRECTORY ENTRY
76	(4C)	BITSTRING	1	MACBFLAG	CONTROL FLAGS
		1...		MACB1THT	- 1ST TIME MARK
		.1..		MACBCHGD	- MBR HAS BEEN CHANGED
		..1.		MACBFWRT	- FORCE WRITE REQUEST PUTR
		...1		MACBRCON	- CONTINUOUS FROM ROOT
		.... 1...		MACBKEEP	- KEEP MACB
		.... .1..		MACBPOIN	- POINT GIVEN FOR MEMBER
		.... ..1.		MACBNOWR	- NO WRITE ALLOWED
		1		MACBNEWC	- CONNECT NEW MEMBER
77	(4D)	UNSIGNED	1	MACBLSEQ	LIBRARY SEQUENCE NUMBER
78	(4E)	UNSIGNED	1	MACBXSEQ	ARGUMENT SEQUENCE NUMBER
79	(4F)	UNSIGNED	1	*	RESERVED
80	(50)	BITSTRING	2	MACBCONT	CONTIGUITY COUNTER
82	(52)	SIGNED	2	MACBMROS	OFFSET OF RECORD IN LB
84	(54)	BITSTRING	1	MACBDEF2	FLAGS
		1...		MACBCMPR	- RECORDS ARE COMPRESSED
		.1..		MACBRCLM	- IMMEDIATE SPACE RECLAM.
		..11		*	- RESERVED
		.... 1...		MACBRTF	- FIXED RECORD TYPE
		.... .1..		MACBRTU	- UNDEFINED RECORD TYPE
		.... ..1.		MACBRTV	- VARIABLE RECORD TYPE
		1		*	- RESERVED
85	(55)	BITSTRING	1	MACBDEF3	RESERVED
		1...		MACBCCUP	UPDATE OF HIGH CONT. COUNT
86	(56)	SIGNED	2	*	RESERVED
88	(58)	SIGNED	4	MACBMBLK	# OF LBS IN MEMBER
92	(5C)	SIGNED	4	MACBNORL	# OF LOGICAL RECORDS
96	(60)	SIGNED	4	MACBRLEN	LOGICAL RECORD LENGTH
100	(64)	SIGNED	4	MACBIDEN	IDENTIFICATION FOR MEMBER
104	(68)	BITSTRING	2	MACBSCNT	CONTIGUITY COUNTER
106	(6A)	CHARACTER	6	MACBCNTA	PRBA OF CONTIGUOUS PART
106	(6A)	UNSIGNED	2	MACBCNTO	- OFFSET IN LB
108	(6C)	SIGNED	4	MACBCNTP	- LIBRARY BLOCK #

Cross Reference

Name	Hex Offset	Hex Value	Level
INLCMACB	0		1
MACBBANC	38		3
MACBBREQ	34		2
MACBBUCB	34		3
MACBCBUF	3C		2
MACBCCUP	55	80	3
MACBCHGD	4C	40	3
MACBCMPR	54	80	3
MACBCNTA	6A		2
MACBCNTO	6A		3
MACBCNTP	6C		3
MACBCONT	50		2
MACBCREM	16		2
MACBCWAA	32		2
MACBDEF2	54		2
MACBDEF3	55		2
MACBFLAG	4C		2
MACBFWRT	4C	20	3
MACBHDR	0		2
MACBHLRA	28		2
MACBHPBN	24		3
MACBHPOS	22		3
MACBHRBA	22		2
MACBIANC	44		3
MACBIDEN	64		2
MACBIREQ	40		2
MACBIUCB	40		3
MACBKEEP	4C	08	3
MACBLRBA	C		2
MACBLSEQ	4D		2
MACBMBLK	58		2
MACBMDIR	48		2
MACBMPBN	2E		3
MACBMPOS	2C		3
MACBMRBA	2C		2
MACBMROS	52		2
MACBNEWC	4C	01	3
MACBNORL	5C		2
MACBNOWR	4C	02	3
MACBPLRA	18		2
MACBPOIN	4C	04	3
MACBPPBN	1E		3
MACBPPOS	1C		3
MACBPPRA	1C		2
MACBPRBA	10		2
MACBPRBN	12		3
MACBPROS	10		3
MACBRCLM	54	40	3
MACBRCON	4C	10	3
MACBRLEN	60		2
MACBRTF	54	08	3
MACBRTU	54	04	3
MACBRTV	54	02	3
MACBSACB	8		2
MACBSCNT	68		2
MACBXSEQ	4E		2
MACB1THT	4C	80	3

INLCMBRX

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	16	INLCMBRX	
0	(0)	CHARACTER	8	MBRXHKEY	HIGHEST KEY IN LOWER LEVEL LB
8	(8)	CHARACTER	1	MBRXGG	RESERVED
9	(9)	BITSTRING	1	MBRXFLAG	FLAG BYTE BIT 1 IS ALWAYS ON
10	(A)	CHARACTER	6	MBRXPRBA	PRBA OF LOWER LEVEL LB
10	(A)	CHARACTER	2	MBRXOFFS	PRBA OF LOWER LEVEL LB
12	(C)	CHARACTER	4	MBRXLBBN	PRBA OF LOWER LEVEL LB

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCMBRX	0	(0)	
MBRXFLAG	9	(9)	
MBRXGG	8	(8)	
MBRXHKEY	0	(0)	
MBRXLBBN	12	(C)	
MBRXOFFS	10	(A)	
MBRXPRBA	10	(A)	

INLCNTPT

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	32	INLCNTPT	ARGUMENT FOR NOTE/POINT
0	(0)	SIGNED	4	NTPTLRBA	LRBA OF 1TH REC. IN LB
4	(4)	CHARACTER	6	NTPTPRBA	PRBA OF LB
4	(4)	SIGNED	2	NTPTPRBO	REC.NO. IN LB
6	(6)	SIGNED	4	NTPTPRBN	REL. LB NUMBER
10	(A)	UNSIGNED	1	NTPTLSEQ	LIBR.SEQ.NUMBER
11	(B)	UNSIGNED	1	NTPTXSEQ	RESERVED
12	(C)	SIGNED	4	NTPTIDEN	RESERVED
16	(10)	CHARACTER	16		RESERVED

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCNTPT	0	(0)	
NTPTIDEN	12	(C)	
NTPTLRBA	0	(0)	
NTPTLSEQ	10	(A)	
NTPTPRBA	4	(4)	
NTPTPRBN	6	(6)	
NTPTPRBO	4	(4)	
NTPTXSEQ	11	(B)	

INLCPARB

CALL INTERFACE PARAMETER CONTROL BLOCK

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	64	INLCPARB	
0	(0)	SIGNED	2	PARMLGTH	LENGTH OF CONTROL BLOCK
2	(2)	CHARACTER	2	*	RESERVED
4	(4)	SIGNED	4	PARMID	IDENTIFICATION
8	(8)	ADDRESS	4	PARMIPTE	ADDRESS OF SYSIPT EXIT ROUTINE
12	(C)	ADDRESS	4	PARMIPTA	ADDRESS OF SYSIPT INPUT AREA
16	(10)	ADDRESS	4	PARMLOGE	ADDRESS OF SYSLOG EXIT ROUTINE
20	(14)	ADDRESS	4	PARMLOGA	ADDRESS OF SYSLOG OUTPUT AREA
24	(18)	ADDRESS	4	PARMLSTE	ADDRESS OF SYSLST EXIT ROUTINE
28	(1C)	ADDRESS	4	PARMLSTA	ADDRESS OF SYSLST OUTPUT AREA
32	(20)	ADDRESS	4	PARMPCHE	ADDRESS OF SYSPCH EXIT ROUTINE
36	(24)	ADDRESS	4	PARMPCHA	ADDRESS OF SYSPCH OUTPUT AREA
40	(28)	UNSIGNED	1	PARMCLP	CURRENT LINE POSITION SYSLST
41	(29)	UNSIGNED	1	PARMFUNC	INTERNAL FUNCTION REQUEST #
42	(2A)	BITSTRING	1	PARMFLAG	FLAGBYTE
		1...		PARMPDV2	1 = PID V2 STACKED BACKUP TAPE
		.1..		PARMEOC	1 = END OF CATALOGING A MEMBER
		..1.		PARMMSHP	1 = MSHP BYPASS ON
		...1 1111		*	RESERVED
43	(2B)	BITSTRING	1	PARMFLG2	FLAGBYTE FOR LNKEDT (NOT USED BY THE LIBRARIAN)
		11..		*	RESERVED
		..1.		PARMCATL	1 = OPTION CATAL REQUESTED VIA CALL I/F
		...1		PARMNORP	1 = NO REPL OF PHASES
		 1111		*	RESERVED
44	(2C)	CHARACTER	4	PARMUSER	USER AREA (USED BY THE LIBRARIAN ONLY IF CALLED BY MSHP)
		POINTERS FOR FORMATTED OUTPUT FUNCTIONS (ONLY USED IF PARMFUNC IS SPECIFIED):			
48	(30)	ADDRESS	4	PARMOUTE	ADDRESS OF OUTPUT EXIT ROUTINE
52	(34)	ADDRESS	4	PARMOUTA	ADDRESS OF OUTPUT AREA
56	(38)	CHARACTER	2	PARMEOD	EOD STRING OF CATALOG FOR MSHP
58	(3A)	BITSTRING	1	PARMFLG3	2ND FLAGBYTE FOR LNKEDT (NOT USED BY THE LIBRARIAN)
		1...		PARMRMOD	1 = RMODE SPECIFIED VIA CALL I/F
		.1..		PARMAMOD	1 = AMODE SPECIFIED VIA CALL I/F
		..11 1...		*	RESERVED
		.... .1..		PARMRANY	1 = RMODE=ANY, 0 = RMODE=24
		11		PARMAANY	11 = AMODE=ANY, 10 = AMODE=31, 01,00 = AMODE=24
		.... ..1.		PARMA31	
		1		PARMA24	
59	(3B)	CHARACTER	1	PARMRC	RETURN CODE FOR SYSIPT EXIT OF LAST LIBRARIAN MODULE EXECUTION
60	(3C)	CHARACTER	4	*	RESERVED

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCPARB	0		1
PARMAANY	3A		3
PARMAMOD	3A		3
PARMA24	3A		4
PARMA31	3A		4
PARMCATL	2B		3
PARMCLP	28		2
PARMEOC	2A		3
PARMEOD	38		2
PARMFLAG	2A		2
PARMFLG2	2B		2
PARMFLG3	3A		2
PARMFUNC	29		2
PARMID	4		2
PARMIPTA	C		2
PARMIPTE	8		2
PARMLGTH	0		2
PARMLOGA	14		2
PARMLOGE	10		2
PARMLSTA	1C		2
PARMLSTE	18		2
PARMMSHP	2A		3
PARMNORP	2B		3
PARMOUTA	34		2
PARMOUTE	30		2
PARMPCHA	24		2
PARMPCHE	20		2
PARMPDV2	2A		3
PARMRANY	3A		3
PARMRC	3B		2
PARMRMOD	3A		3
PARMUSER	2C		2

INLCPIDT

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	40	INLCPIDT	
0	(0)	A-ADDRESS	4	PIDTLAMB	ADDRESS OF LAMB
4	(4)	A-ADDRESS	4	PIDTFID	ADDRESS OF FILE-ID
8	(8)	A-ADDRESS	4	PIDTCID	ADDRESS OF VSAM CATALOG ID
12	(C)	A-ADDRESS	4	PIDTPUNT	ADDRESS OF PHYSICAL UNIT
16	(10)	A-ADDRESS	4	PIDTVLID	ADDRESS OF VOLID
20	(14)	A-ADDRESS	4	PIDTSTAD	ADDRESS OF LIBRARY START ADDRESS
24	(18)	A-ADDRESS	4	PIDTLDTE	ADDRESS OF LDT C-ENTRY
28	(1C)	A-ADDRESS	4	PIDTEXTS	ADDRESS OF LIBRARY EXTENTS
32	(20)	A-ADDRESS	4	PIDTDEVC	ADDRESS OF EXTENT DEVICE ENTRIES
36	(24)	BITSTRING	1	PIDTFDBC	FEEDBACK CODE
37	(25)	BITSTRING	1	PIDTFTYP	SPACE MANGEMENT TYPE
		1...		PIDTVSAM	0: BAM , 1: VSAM MGMT SPACE
		.111 1111			RESERVED
38	(26)	BITSTRING	1	PIDTMF	MAIN FUNCTION
		1...		PIDTLKUP	1: LOOKUP
		.1..		PIDTINS	1: INSERT
		..1.		PIDTDEL	1: DELETE
39	(27)	CHARACTER	1		RESERVED

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCPIDT	0	(0)	
PIDTCID	8	(8)	
PIDTDEL	38	X'20'	
PIDTDEVC	32	(20)	
PIDTEXTS	28	(1C)	
PIDTFDBC	36	(24)	
PIDTFID	4	(4)	
PIDTFTYP	37	(25)	
PIDTINS	38	X'40'	
PIDTLAMB	0	(0)	
PIDTLDTE	24	(18)	
PIDTLKUP	38	X'80'	
PIDTMF	38	(26)	
PIDTPUNT	12	(C)	
PIDTSTAD	20	(14)	
PIDTVLID	16	(10)	
PIDTVSAM	37	X'80'	

INLCRESN

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	20	INLCRESN	
0	(0)	CHARACTER	20	RESARG	RESOUCE NAME ARGUMENT
0	(0)	CHARACTER	8	RESUBNAM	SUBLIBRARY NAME
8	(8)	CHARACTER	12	RESNAME	RESOURCE NAME
8	(8)	CHARACTER	1	RESID	RESOURCE ID
		1111 11..			RESERVED
		.... ..1.		RESNSEC	SECURITY VIOLATION
		1		RESNFND	SUBLIB NOT FOUND
9	(9)	CHARACTER	6	RESVOLID	VOLID
15	(F)	CHARACTER	5	RESDSKAD	START ADDRESS
15	(F)	CHARACTER	4	RESCCHH	CCHH OR PBN
15	(F)	SIGNED	2	RESCC	CC
17	(11)	SIGNED	2	RESHH	HH
19	(13)	UNSIGNED	1	RESRECNO	RECORD NUMBER

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCRESN	0	(0)	
RESARG	0	(0)	
RESCC	15	(F)	
RESCCHH	15	(F)	
RESDSKAD	15	(F)	
RESHH	17	(11)	
RESID	8	(8)	
RESNAME	8	(8)	
RESNFND	8	X'01'	
RESNSEC	8	X'02'	
RESRECNO	19	(13)	
RESUBNAM	0	(0)	
RESVOLID	9	(9)	

INLCSABL

TAPE BLOCK HEADER FOR SA IPL AND SA UTILITY FILE

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	8	INLCSABL	
0	(0)	UNSIGNED	2	SABLKLEN	LENGTH OF TAPE BLOCK
2	(2)	CHARACTER	2	SABLDESR	TAPE BLOCK DESCRIPTOR: "SA"
4	(4)	SIGNED	4	SABLKNO	BLOCK NUMBER

INLCSACB

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	88	INLCSACB	SUBLIB ACCESS CONTROL BLOCK
0	(0)	CHARACTER	8	SACBHDR	SACB HEADER "INLCSACB"
8	(8)	A-ADDRESS	4	SACBLACB	ADDR OF LACB
12	(C)	CHARACTER	6	SACBHXAD	PRBA OF HIGEST INDEX LEVEL
12	(C)	UNSIGNED	2	SACBHXOS	OFFSET IN LB
14	(E)	SIGNED	4	SACBHXBN	LIBRARY BLOCK #
18	(12)	CHARACTER	6	SACBLXAD	PRBA OF LOWEST INDEX LEVEL
18	(12)	UNSIGNED	2	SACBLXOS	OFFSET IN LB
20	(14)	SIGNED	4	SACBLXBN	LIBRARY BLOCK #
24	(18)	CHARACTER	6	SACBCXAD	PRBA OF CURRENT INDEX ENTRY
24	(18)	UNSIGNED	2	SACBCXOS	OFFSET IN LB
26	(1A)	SIGNED	4	SACBCXBN	LIBRARY BLOCK #
30	(1E)	UNSIGNED	1	SACBXLNO	# OF INDEX LEVELS
31	(1F)	UNSIGNED	1	SACBCXLN	# OF CURRENT INDEX LEVEL
32	(20)	A-ADDRESS	4	SACBICXA	ADDRESS OF INCORE INDEX
36	(24)	CHARACTER	8	SACBCREQ	REQUEST LIST FOR BUFFER MGT
44	(2C)	A-ADDRESS	4	SACBCBUF	ADDRESS OF CURRENT X BUFFER
48	(30)	BITSTRING	4	SACBFLAG	CONTROL FLAGS
		1...		SACBRCLM	IMMEDIATE SPACE RECLAM.
52	(34)	CHARACTER	32	SACBXSPC	INDEX LB SPLIT CONTROL
52	(34)	CHARACTER	6	SACBGDIR	NEXT RECORD PTR FOR GDIR
58	(3A)	CHARACTER	6	SACBKOID	RESERVED
64	(40)	A-ADDRESS	4	SACBOLDX	LB-# OF 2ND HIGHEST LEVEL
68	(44)	A-ADDRESS	4	SACBSBNO	# OF SPLIT LB
72	(48)	CHARACTER	8	SACBSREQ	REQUEST LIST FOR B-MGMT
80	(50)	A-ADDRESS	4	SACBSBUF	ADDRESS OF SPLIT BUFFER
84	(54)	SIGNED	4	SACBIDEN	IDENTIFICATION OF INDEX

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCSACB	0	(0)	
SACBCBUF	44	(2C)	
SACBCREQ	36	(24)	
SACBCXAD	24	(18)	
SACBCXBN	26	(1A)	
SACBCXLN	31	(1F)	
SACBCXOS	24	(18)	
SACBFLAG	48	(30)	
SACBGDIR	52	(34)	
SACBHDR	0	(0)	
SACBHXAD	12	(C)	
SACBHXBN	14	(E)	
SACBHXOS	12	(C)	
SACBICXA	32	(20)	
SACBIDEN	84	(54)	
SACBKOID	58	(3A)	
SACBLACB	8	(8)	
SACBLXAD	18	(12)	
SACBLXBN	20	(14)	
SACBLXOS	18	(12)	
SACBOLDX	64	(40)	
SACBRCLM	48	X'80'	
SACBSBNO	68	(44)	
SACBSBUF	80	(50)	
SACBSREQ	72	(48)	
SACBXLNO	30	(1E)	
SACBXSPC	52	(34)	

INLCSAPH

TAPE STAND ALONE PHASE RECORD HEADER

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	100	INLCSAPH	
0	(0)	UNSIGNED	2	SAPHSHDL	LENGTH OF PHASE RECORD HEADER
2	(2)	CHARACTER	4	SAPHDESR	PHASE RECORD DESCRIPTOR: "SAPH"
6	(6)	UNSIGNED	2	SAPHNXTR	OFFSET TO NEXT RECORD IN BLOCK OR TO END OF BLOCK
8	(8)	SIGNED	2	SAPHSREC	PHASE RECORD NUMBER ON TAPE
10	(A)	UNSIGNED	1	SAPHSEND	X"FF" - LAST PHASE RECORD IND.
11	(B)	CHARACTER	1	*	RESERVED
12	(C)	SIGNED	4	SAPHSRLD	OFFSET TO RLD INFO IN PHASE
16	(10)	SIGNED	4	SAPHSLEN	LENGTH OF PHASE INCL RLD INFO
20	(14)	CHARACTER	8	*	RESERVED
28	(1C)	CHARACTER	72	SAPHSDLE	PHASE SDL ENTRY

Cross Reference

Name	Hex Offset	Hex Value	Level
INLCSAPH	0		1
SAPHDESR	2		2
SAPHNXTR	6		2
SAPHSDLE	1C		2
SAPHSEND	A		2
SAPHSHDL	0		2
SAPHSLEN	10		2
SAPHSREC	8		2
SAPHSRLD	C		2

INLCSCAN

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	25	INLCSCAN	
0	(0)	CHARACTER	4	SCANID	HEADER IDENTIFIER
4	(4)	A-ADDRESS	4	TEXTBEG	START OF STRING TO BE SCANNED
8	(8)	A-ADDRESS	4	TEXTEND	END OF STRING TO BE SCANNED
12	(C)	A-ADDRESS	4	PRESTART	LOCAL SAVE FIELD
16	(10)	A-ADDRESS	4	START	PTR TO RECOGNIZED ITEM
20	(14)	SIGNED	4	LEN	LENGTH OF RECOGNIZED ITEM
24	(18)	CHARACTER	1	OPTNS	OPTIONS BYTE
		1...		SKPBLNK	TRUE SKIP BLANKS BEFORE ITEM
		.1..		ENDSTR	TRUE END OF STRING
		..1.		NEWCMD	NEW COMMAND IS STARTING
		...1		CONTIO	I/O DUE TO CONTINUATION
		.... 1...		CONTSUPP	SUPPRESS CONTINUAT. I/O
		.... .1..		FLUSHING	FLUSHING IN PROCESS
		.... ..1.			RESERVED, SHOULD BE FALSE
		1		FOUNDIND	FLAG TO INDICATE SUCCESS OF SCAN

Cross Reference:

Name	Hex Offset	Hex Value	Level
CONTIO	24	X'10'	
CONTSUPP	24	X'08'	
ENDSTR	24	X'40'	
FLUSHING	24	X'04'	
FOUNDIND	24	X'01'	
INLCSCAN	0	(0)	
LEN	20	(14)	
NEWCMD	24	X'20'	
OPTNS	24	(18)	
PRESTART	12	(C)	
SCANID	0	(0)	
SKPBLNK	24	X'80'	
START	16	(10)	
TEXTBEG	4	(4)	
TEXTEND	8	(8)	

INLCSDLE

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	72	INLCSDLE (*)	SYSTEM DIRECTORY ENTRY
0	(0)	CHARACTER	18	SDLESEGM	D.E. FIRST SEGMENT
0	(0)	CHARACTER	8	SDLENAM	MEMBER NAME
8	(8)	CHARACTER	1	SDLEGEN	RESERVED FOR GENERATION G.
9	(9)	CHARACTER	1	SDLEDEF1	ATTRIBUTES FOR DIR.ENTRY
		11..		*	RESERVED
		..1.		SDLEEDIR	TYPE OF ENTRY = DIRECTORY
		...1 1111		*	RESERVED
10	(A)	CHARACTER	6	SDLEPRBA	MEMBER BEGIN PRBA
10	(A)	UNSIGNED	2	SDLERBAO	OFFSET IN LB
12	(C)	UNSIGNED	4	SDLERBAP	REL.BLOCK ADDRESS
16	(10)	SIGNED	2	SDLECONT	CONTIGUITY COUNTER
18	(12)	BITSTRING	2	*	RESERVED
20	(14)	BITSTRING	4	SDLEPFL	ATTRIBUTE FIELDS
20	(14)	BITSTRING	1	SDLEFLG	FLAGS
		1...		SDLEBSR	SELF RELOCATING PHASE
		.1..		SDLEBRL	RELOCATING PHASE
		..1.		SDLEBSE	SVA ELIGIBLE
		...1		SDLEBSV	PHASE IN SVA
		.... 1...		SDLEBPC	PCIL FLAG FOR INCORE DIR.
		.... .1..		SDLEBNF	NOT FOUND FLAG (INCORE D)
		.... ..1.		SDLEBAC	ENTRY ACTIVE (INCORE DIR)
		1		SDLESVPF	SVA ELI. + PFIX
21	(15)	BITSTRING	1	SDLESWT	MODE SWITCHES
		1...		SDLECLM	SET SDL:MOVE MODE PHASE
		.1..		SDLECLS	SET SDL:SVA ELIGIBLE
		..1.		SDLERMOD	RMODE 1=ANY 0=24
		...1 1...		SDLEAMOD	AMODE 00,01=24 10=31 , 11=ANY
		...1		SDLEAM31	AMODE 31 OR ANY
		.... 1...		SDLEAM24	AMODE 24 OR ANY
		.... .1..		SDLEMARK	
		.... ..1.		SDLEM33I	
		1		SDLEBMSG	
22	(16)	UNSIGNED	1	SDLELDRT	LOAD RETURN CODE
23	(17)	BITSTRING	1	SDLEAT2	
		1...		SDLERM	RMODE FROM MODE OR PARM
		.1..		SDLEAM	AMODE FROM MODE OR PARM
		..1.		SDLEMDR	RMODE PARM OR MODE
		...1 1...		SDLEMDA	RMODE PARM OR MODE
		...1		SDLEMD31	AMODE/MOD 31 | ANY
		.... 1...		SDLEMD24	AMODE/MOD 24 | ANY
		.... .1..		SDLESDR	RMODE/ESD
		11		SDLESDA	AMODE/ESD
		.... ..1.		SDLESD31	AMODE 31 | ANY
		1		SDLESD24	AMODE 24 | ANY
24	(18)	SIGNED	4	SDLEPLN	LENGTH OF PHASE IN BYTES
28	(1C)	ADDRESS	4	SDLELPT	LOAD POINT AT L.E. TIME
32	(20)	ADDRESS	4	SDLEENP	ENTRY POINT AT L.E. TIME
36	(24)	ADDRESS	4	SDLESTR	PARTITION BEG AT L.E. TIME
40	(28)	UNSIGNED	2	SDLERLD	NO. OF RLD ITEMS
42	(2A)	CHARACTER	6	SDLERLDA	PRBA OF 1TH RLD ITEM
42	(2A)	UNSIGNED	2	SDLERLDO	OFFSET IN BLOCK
44	(2C)	UNSIGNED	4	SDLERLDP	REL. BLOCK ADDRESS
48	(30)	BITSTRING	1	SDLEBYTE	SWITCH
		1...		SDLE23BA	2/3 BYTE A-CONST
		.111 1111		*	RESERVED
49	(31)	BITSTRING	1	*	RESERVED
50	(32)	CHARACTER	6	SDLESMPA	RESERVED
50	(32)	UNSIGNED	2	SDLESMPO	RESERVED
52	(34)	UNSIGNED	4	SDLESMPP	RESERVED
56	(38)	ADDRESS	4	SDLESVAP	ADDRESS OF PHASE IN SVA
60	(3C)	UNSIGNED	4	SDLEIDEN	LIBRARY BLOCK IDENTIFIER

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
64	(40)	ADDRESS	4	SDLEALIB	ADDRESS OF LDTE
68	(44)	ADDRESS	4	SDLEASLB	ADDRESS OF SDTE

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCSDLE	0		1
SDLEALIB	40		2
SDLEAM	17		4
SDLEAMOD	15		4
SDLEAM24	15		5
SDLEAM31	15		5
SDLEASLB	44		2
SDLEAT2	17		3
SDLEBAC	14		4
SDLEBMSG	15		4
SDLEBNF	14		4
SDLEBPC	14		4
SDLEBRL	14		4
SDLEBSE	14		4
SDLEBSR	14		4
SDLEBSV	14		4
SDLEBYTE	30		2
SDLECLM	15		4
SDLECLS	15		4
SDLECONT	10		3
SDLEDEF1	9		3
SDLEEDIR	9		4
SDLEENP	20		2
SDLEFLG	14		3
SDLEGEN	8		3
SDLEIDEN	3C		2
SDLELDRT	16		3
SDLELPT	1C		2
SDLEMARK	15		4
SDLEMDA	17		4
SDLEMDR	17		4
SDLEMD24	17		5
SDLEMD31	17		5
SDLEM33I	15		4
SDLENAM	0		3
SDLEPFL	14		2
SDLEPLN	18		2
SDLEPRBA	A		3
SDLERBAO	A		4
SDLERBAP	C		4
SDLERLD	28		2
SDLERLDA	2A		2
SDLERLDO	2A		3
SDLERLDP	2C		3
SDLERM	17		4
SDLERMOD	15		4
SDLESDA	17		4
SDLESDR	17		4
SDLESD24	17		5
SDLESD31	17		5
SDLESEGM	0		2
SDLESMPA	32		2
SDLESMPO	32		3
SDLESMPP	34		3
SDLESTR	24		2
SDLESVAP	38		2
SDLESVPF	14		4
SDLESWT	15		3
SDLE23BA	30		3

INLCSDLH

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	16	INLCSDLH	TABLE HEADER
0	(0)	SIGNED	2	SDLHLEN	LENGTH OF ENTRY IF SDLTAB
2	(2)	CHARACTER	6	SDLHDIRA	PRBA OF MEMBER DIR. LIBRAR
2	(2)	UNSIGNED	2	SDLHOFFS	OFFSET IN BLOCK
4	(4)	UNSIGNED	4	SDLHLBN	REL. BLOCK ADDRESS
8	(8)	BITSTRING	1	SDLHCOMM	COMMUNICATION BYTE
		1...		SDLHMSHP	MSHP BIT
		.1..		SDLHSCIL	SYSLIB1
		..1.		SDLHMSH1	MSHP IS ACTIVE
		...1		SDLHMSH2	MSHP-BYPASS IS ACTIVE
		 1111		*	
9	(9)	CHARACTER	1	SDLHTYPE	TYPE OF REQUEST
10	(A)	SIGNED	2	SDLHNOEN	NO. OF ENTRIES IN LIST
12	(C)	SIGNED	4	SDLHEXTA	EXTENT TABLE ADDRESS

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCSDLH	0		1
SDLHCOMM	8		2
SDLHDIRA	2		2
SDLHEXTA	C		2
SDLHLBN	4		3
SDLHLEN	0		2
SDLHMSHP	8		3
SDLHMSH1	8		3
SDLHMSH2	8		3
SDLHNOEN	A		2
SDLHOFFS	2		3
SDLHSCIL	8		3
SDLHTYPE	9		2

INLCSDTE

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	44	INLCSDTE	SUBLIB DEF. TABLE ENTRY
0	(0)	CHARACTER	8	SDTENAME	SUBLIBRARY NAME
8	(8)	A-ADDRESS	4	SDTESLDA	ADDRESS OF SLD
12	(C)	SIGNED	2	SDTENSLD	# SLD ENTRIES
14	(E)	UNSIGNED	1	SDTEXLEV	# INDEX LEVELS IN SUBLIB
15	(F)	BITSTRING	1	SDTESTAT	RESERVED
		1...		SDTELBER	NO SLD UPDATE POSSIBLE
		.1..		SDTEITER	INTERFACE ERROR
		..1.		SDTEIOER	I/O ERROR
16	(10)	CHARACTER	6	SDTEXRBA	PRBA OF HIGHEST INDEX LVL
16	(10)	SIGNED	2	SDTEXRBO	OFFSET WITHIN LB
18	(12)	CHARACTER	4	SDTEXRBP	BLOCK-#
22	(16)	CHARACTER	6	SDTEDRBA	PRBA OF LOWEST INDEX LVL
22	(16)	SIGNED	2	SDTEDRBO	OFFSET WITHIN LB
24	(18)	CHARACTER	4	SDTEDRBP	BLOCK-#
28	(1C)	BITSTRING	2	SDTEPART	PARTITION FLAGS
30	(1E)	BITSTRING	1	SDTEFLAG	FLAG BYTE
		1...		SDTENOAC	NO ACCESS POSSIBLE
		.1..		SDTEDELT	SUBLIB IS DELETED
		..1.		SDTERCLM	IMMEDIATE SPACE RECLAM.
		...1 1...			RESERVED
		.... .1..		SDTEPROT	SUBLIB IS PROTECTED
		.... ..1.		SDTEINCX	INCOMPLETE SLD
		1		SDTENEWS	NEW SUBLIBRARY
31	(1F)	UNSIGNED	1	SDTEUCNT	SUBLIB USER COUNT
32	(20)	CHARACTER	3		RESERVED
35	(23)	BITSTRING	1	SDTEUACC	UNIVERSAL ACCESS RIGHT
36	(24)	SIGNED	4	SDTEIDEN	IDENTIFICATION OF INDEX
40	(28)	A-ADDRESS	4	SDTENEXT	ADDR OF NEXT SUBLIB

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCSDTE	0	(0)	
SDTEDELT	30	X'40'	
SDTEDRBA	22	(16)	
SDTEDRBO	22	(16)	
SDTEDRBP	24	(18)	
SDTEFLAG	30	(1E)	
SDTEIDEN	36	(24)	
SDTEINCX	30	X'02'	
SDTEIOER	15	X'20'	
SDTEITER	15	X'40'	
SDTELBER	15	X'80'	
SDTENAME	0	(0)	
SDTENEWS	30	X'01'	
SDTENEXT	40	(28)	
SDTENOAC	30	X'80'	
SDTENSLD	12	(C)	
SDTEPART	28	(1C)	
SDTEPROT	30	X'04'	
SDTERCLM	30	X'20'	
SDTESLDA	8	(8)	
SDTESTAT	15	(F)	
SDTEUACC	35	(23)	
SDTEUCNT	31	(1F)	
SDTEXLEV	14	(E)	
SDTEXRBA	16	(10)	
SDTEXRBO	16	(10)	
SDTEXRBP	18	(12)	

INLCSLDE

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	12	INLCSLDE	SECOND LEVEL DIRECTORY ENTRY
0	(0)	CHARACTER	8	SLDEKEY	LAST MEMBER NAME IN LB
8	(8)	CHARACTER	4	SLDEPRBA	PRBA OF DIRECTORY LB

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCSLDE	0	(0)	
SLDEKEY	0	(0)	
SLDEPRBA	8	(8)	

INLCSLXE

SUBLIBRARYINDEXENTRY

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	72	INLCSLXE	
0	(0)	CHARACTER	8	SLXENAME	SUBLIBRARY NAME
8	(8)	CHARACTER	10	SLXECRDT	TIME-STAMP OF SUBLIBRARY CREATION
18	(12)	UNSIGNED	1	SLXESEQN	RESERVED (SEQUENCE NUMBER)
19	(13)	UNSIGNED	1	SLXENXLV	# OF INDEX LEVELS
20	(14)	CHARACTER	6	SLXEHXRA	PRBA OF HIGHEST MBR INDEX LEVEL
20	(14)	CHARACTER	2	SLXEHXOS	- OFFSET WITHIN BLOCK
22	(16)	CHARACTER	4	SLXEHXBN	- BLOCK-#
26	(1A)	CHARACTER	6	SLXELXRA	PRBA OF MBR DIRECTORY START LB
26	(1A)	CHARACTER	2	SLXELXOS	- OFFSET WITHIN BLOCK
28	(1C)	CHARACTER	4	SLXELXBN	- BLOCK-#
32	(20)	CHARACTER	6	SLXEDELA	PRBA OF RECLAMATION CHAIN
32	(20)	UNSIGNED	2	SLXEDEOS	PRBA OF RECLAMATION CHAIN
34	(22)	CHARACTER	4	SLXEDEBN	PRBA OF RECLAMATION CHAIN
38	(26)	BITSTRING	1	SLXEFLAG	FLAGS
		1...		SLXEDELM	- SUBLIBRARY CONTAINS DELETED MEMBERS
		.1..		SLXERCLM	- IMMEDIATE SPACE REUSAGE FOR MEMBERS
		..11 11..		*	- RESERVED
		.... ..1.		SLXENSEC	- RESERVED (SECURITY VIOLATION)
		1		SLXENFND	- RESERVED (NOT FOUND)
39	(27)	UNSIGNED	1	*	RESERVED
40	(28)	SIGNED	4	SLXEMBRN	# OF MEMBERS IN SUBLIBRARY
44	(2C)	SIGNED	4	SLXERESV	# OF USED LBS
48	(30)	SIGNED	4	SLXERECL	# OF RECLAIMED LBS
52	(34)	SIGNED	4	SLXEIDEN	RESERVED
56	(38)	SIGNED	4	SLXENLCK	# OF LOCKED MEMBERS
60	(3C)	CHARACTER	12	*	RESERVED

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCSLXE	0		1
SLXECRDT	8		2
SLXEDEBN	22		3
SLXEDELA	20		2
SLXEDELM	26		3
SLXEDEOS	20		3
SLXEFLAG	26		2
SLXEHXBN	16		3
SLXEHXOS	14		3
SLXEHXRA	14		2
SLXEIDEN	34		2
SLXELXBN	1C		3
SLXELXOS	1A		3
SLXELXRA	1A		2
SLXEMBRN	28		2
SLXENAME	0		2
SLXENFND	26		3
SLXENLCK	38		2
SLXENSEC	26		3
SLXENXLV	13		2
SLXERCLM	26		3
SLXERECL	30		2
SLXERESV	2C		2
SLXESEQN	12		2

INLCSPAD

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	72	INLCSPAD	FREE SPACE DESCRIPTOR
0	(0)	CHARACTER	4	SPADID	CONTROL BLOCK ID.
4	(4)	SIGNED	4	SPADLK	LENGTH OF CONTROL BLOCK
8	(8)	SIGNED	4	SPADBMCN	BMLB COUNTER
12	(C)	SIGNED	4	SPADALLC	ALLOCATED LBS IN LIBRARY
16	(10)	SIGNED	4	SPADAVAL	AVAILABLE LBS IN LIBRARY
20	(14)	SIGNED	4	SPADBFRE	FREE SPACE BEGIN
24	(18)	SIGNED	4	SPADHIWA	NEXT FREE POSITION
28	(1C)	BITSTRING	2		
		1...		SPADHIFG	IF HI END REACHED
		.111 1111			
		1111 1111			
30	(1E)	CHARACTER	4		RESERVED
34	(22)	SIGNED	2		RESERVED
36	(24)	CHARACTER	32		RESERVED
68	(44)	CHARACTER	4	SPADEND	MARK BEGIN OF BIT STRING

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCSPAD	0	(0)	
SPADALLC	12	(C)	
SPADAVAL	16	(10)	
SPADBFRE	20	(14)	
SPADBMCN	8	(8)	
SPADEND	68	(44)	
SPADHIFG	28	X'80'	
SPADHIWA	24	(18)	
SPADID	0	(0)	
SPADLK	4	(4)	

INLCSTOH

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	16	INLCSTOH	TABLE HEADER
0	(0)	SIGNED	2	STOHLEN	LENGTH OF ENTRY IF SDLTAB
2	(2)	CHARACTER	6	STOHDIRA	PRBA OF MEMBER DIR. LIBRAR
2	(2)	UNSIGNED	2	STOHOFFS	OFFSET IN BLOCK
4	(4)	UNSIGNED	4	STOHLBN	REL. BLOCK ADDRESS
8	(8)	BITSTRING	1	STOHCOMM	COMMUNICATION BYTE
		1...		STOHMSHP	MSHP BIT
		.1..		STOHSCIL	SYSLIB1
		..1.		STOHMSH1	MSHP IS ACTIVE
		...1		STOHMSH2	MSHP-BYPASS IS ACTIVE
		 1111		*	
9	(9)	CHARACTER	1	STOHTYPE	TYPE OF REQUEST
10	(A)	SIGNED	2	STOHNOEN	NO. OF ENTRIES IN LIST
12	(C)	SIGNED	4	STOHSLST	SORTED LIST ANCHOR

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCSTOH	0		1
STOHCOMM	8		2
STOHDIRA	2		2
STOHLBN	4		3
STOHLEN	0		2
STOHMSHP	8		3
STOHMSH1	8		3
STOHMSH2	8		3
STOHNOEN	A		2
STOHOFFS	2		3
STOHSCIL	8		3
STOHSLST	C		2
STOHTYPE	9		2

INLCSTOL

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	88	INLCSTOL (*)	STOW LIST
0	(0)	ADDRESS	4	STOLDEPT	DIR.ENTRY POINTER
4	(4)	SIGNED	2	STOLELNG	LENGTH OF STOWL ENTRY
6	(6)	UNSIGNED	1	STOLLSEQ	LIB/SUBLIB SEQ.NUMBER
7	(7)	BITSTRING	1	STOLSW	SWITCH BYTE
		1...		STOLFND	MEMBER FOUND
		.1..		STOLCAT	MEMBER CATALOGUED
		..1.		STOLDEL	MEMBER DELETED
		...1		STOLREN	MEMBER RENAMED
		.... 1...		STOLIPT	DATA ON SYSIPT
		.... .1..		STOLSYS	MEMBER IN SYSTEM LIBRARY
		.... ..1.		STOLSEC	SECURITY
		1		*	RESERVED
8	(8)	CHARACTER	17	STOLKEY	MEMBER KEY (TYPE.NAME)
8	(8)	CHARACTER	8	STOLMTYP	MEMBER TYPE
16	(10)	CHARACTER	8	STOLNAM	MEMBER NAME
24	(18)	CHARACTER	1	STOLGEN	RESERVED FOR GENERATION G.
25	(19)	CHARACTER	1	STOLDEF1	ATTRIBUTES FOR DIR.ENTRY
		11..		*	RESERVED
		..1.		STOLEDIR	TYPE OF ENTRY = DIRECTORY
		...1 1111		*	RESERVED
26	(1A)	CHARACTER	6	STOLPRBA	MEMBER BEGIN PRBA
26	(1A)	UNSIGNED	2	STOLRBAO	OFFSET IN LB
28	(1C)	UNSIGNED	4	STOLRBAP	REL.BLOCK ADDRESS
32	(20)	SIGNED	2	STOLVIFL	LENGTH OF DE COMMON SEGMNT
34	(22)	CHARACTER	6	STOLLSTA	PRBA OF LAST LB OF MEMBER
34	(22)	UNSIGNED	2	STOLLSTO	OFFSET IN LB
36	(24)	UNSIGNED	4	STOLLSTP	REL.BLOCK ADDRESS
40	(28)	BITSTRING	1	STOLDEF2	FLAGS
		1...		STOLMSH1	MODULE UNDER MSHP CONT.
		.1..		STOLMSH2	CHANGED WITH MSHP BYPASS
		..1.		STOLAPI	BUILD WITH API
		...1		*	RESERVED
		.... 1...		STOLRTF	REC.TYPE=FIXED
		.... .1..		STOLRTU	REC.TYPE=UNDEFINED
		.... ..1.		STOLRTV	REC.TYPE=VARIABLE
		1		STOLSIPT	SYSIPT DATA IN MEMBER
41	(29)	BITSTRING	1	STOLDEF3	
		1...		STOLVIF	ONE OR MORE VIFS IN DE
		.1..		STOLCMPR	DATA IN COMPRESSED FORM
		..11 1111		*	RESERVED
42	(2A)	SIGNED	2	STOLMBLK	NO. OF LBS FOR MEMBER
44	(2C)	SIGNED	4	STOLNORL	NO OF LOGICAL RECORDS
48	(30)	SIGNED	4	STOLRLEN	LOGICAL REC. LENGTH
52	(34)	SIGNED	2	STOLCONT	CONTIGUITY COUNTER
54	(36)	CHARACTER	10	STOLDORI	TIME STAMP ORIGINATION
64	(40)	CHARACTER	10	STOLDUPD	TIME STAMP LAST UPDATE
74	(4A)	CHARACTER	2	STOLVEMO	VERSION MODIFICATION
76	(4C)	CHARACTER	2	STOLEODA	SYSIPT END OF DATA
78	(4E)	CHARACTER	2	*	RESERVED
80	(50)	CHARACTER	4	STOLIDEN	RESERVED
84	(54)	CHARACTER	4	*	RESERVED

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	17	STOLNWKY	NEW KEY FOR RENAME
0	(0)	CHARACTER	8	STOLNTYP	NEW TYPE
8	(8)	CHARACTER	8	STOLNNAM	NEW NAME
16	(10)	CHARACTER	1	*	RESERVED

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCSTOL	0		1
STOLAPI	28		3
STOLCAT	7		3
STOLCMPR	29		3
STOLCONT	34		2
STOLDEF1	19		2
STOLDEF2	28		2
STOLDEF3	29		2
STOLDEL	7		3
STOLDEPT	0		2
STOLDORI	36		2
STOLDUPD	40		2
STOLEDIR	19		3
STOLELNG	4		2
STOLEODA	4C		2
STOLFND	7		3
STOLGEN	18		3
STOLIDEN	50		2
STOLIPT	7		3
STOLKEY	8		2
STOLLSEQ	6		2
STOLLSTA	22		2
STOLLSTO	22		3
STOLLSTP	24		3
STOLMBLK	2A		2
STOLMSH1	28		3
STOLMSH2	28		3
STOLMTYP	8		3
STOLNAM	10		3
STOLNNAM	8		2
STOLNORL	2C		2
STOLNTYP	0		2
STOLNWKY	0		1
STOLPRBA	1A		2
STOLRBAO	1A		3
STOLRBAP	1C		3
STOLREN	7		3
STOLRLEN	30		2
STOLRTF	28		3
STOLRTU	28		3
STOLRTV	28		3
STOLSEC	7		3
STOLSIPT	28		3
STOLSW	7		2
STOLSYS	7		3
STOLVEMO	4A		2
STOLVIF	29		3
STOLVIFL	20		2

INLCTYPE

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	10	INLCTYPE	
0	(0)	CHARACTER	8	TYPENAME	MEMBER TYPE NAME
8	(8)	CHARACTER	1	TYPEGG	RESERVED
9	(9)	BITSTRING	1	TYPEFLAG	FLAG BYTE BIT 0 IS ALWAYS ON

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCTYPE	0	(0)	
TYPEFLAG	9	(9)	
TYPEGG	8	(8)	
TYPENAME	0	(0)	

INLCUPHA

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	44	INLCUPHA	
0	(0)	SIGNED	2	UPHAELEN	LENGTH OF V.I.F.
2	(2)	BITSTRING	2	UPHAEATR	RESERVED
		1...		UPHAVIF	VIF FOLLOWS
4	(4)	CHARACTER	4	UPHAEID	V.I.F. IDENTIFIER
8	(8)	BITSTRING	4	UPHAPFL	ATTRIBUTE FIELDS
8	(8)	BITSTRING	1	UPHAFLG	FLAGS
		1...		UPHABSR	SELF RELOCATING PHASE
		.1..		UPHABRL	RELOCATING PHASE
		..1.		UPHABSE	SVA ELIGIBLE
		...1		UPHABSV	PHASE IN SVA
		.... 1...		UPHABPC	PCIL FLAG FOR INCORE DIR.
		.... .1..		UPHABNF	NOT FOUND FLAG (INCORE D)
		.... ..1.		UPHABAC	ENTRY ACTIVE (INCORE DIR)
		1		UPHASVPF	SVA ELI. + PFIX
9	(9)	BITSTRING	1	UPHASWT	MODE SWITCHES
		1...		UPHACLM	SET SDL:MOVE MODE PHASE
		.1..		UPHACLS	SET SDL:SVA ELIGIBLE
		..1.		UPHARMOD	RMODE 1=ANY 0=24
		...1 1...		UPHAAMOD	AMODE 00,01=24 10=31 , 11=ANY
		...1		UPHAAM31	AMODE 31 OR ANY
		.... 1...		UPHAAM24	AMODE 24 OR ANY
		.... .1..		UPHAMARK	
		.... ..1.		UPHAM33I	
		1		UPHABMSG	
10	(A)	UNSIGNED	1	UPHALDRT	LOAD RETURN CODE
11	(B)	BITSTRING	1	UPHAAT2	
		1...		UPHARM	RMODE FROM MODE OR PARM
		.1..		UPHAAM	AMODE FROM MODE OR PARM
		..1.		UPHAMDR	RMODE PARM OR MODE
		...1 1...		UPHAMDA	AMODE PARM OR MODE
		...1		UPHAMD31	AMODE/MOD 31 I ANY
		.... 1...		UPHAMD24	AMODE/MOD 24 I ANY
		.... .1..		UPHASDR	RMODE/ESD
		11		UPHASDA	AMODE/ESD: 00,01=24 10=31, 11=ANY
		.... ..1.		UPHASD31	AMODE 31 I ANY
		1		UPHASD24	AMODE 24 I ANY
12	(C)	SIGNED	4	UPHAPLN	LENGTH OF PHASE IN BYTES
16	(10)	ADDRESS	4	UPHALPT	LOAD POINT AT L.E. TIME
20	(14)	ADDRESS	4	UPHAENP	ENTRY POINT AT L.E. TIME
24	(18)	ADDRESS	4	UPHASTR	PARTITION BEG AT L.E. TIME
28	(1C)	UNSIGNED	2	UPHARLD	NO. OF RLD ITEMS
30	(1E)	CHARACTER	6	UPHARLDA	PRBA OF 1TH RLD ITEM
30	(1E)	UNSIGNED	2	UPHARLDO	OFFSET IN BLOCK
32	(20)	UNSIGNED	4	UPHARLDP	REL. BLOCK ADDRESS
36	(24)	BITSTRING	1	UPHABYTE	SWITCH
		1...		UPHA23BA	2/3 BYTE A-CONST
		.111 1111		*	RESERVED
37	(25)	BITSTRING	1	*	RESERVED
38	(26)	CHARACTER	6	UPHASMPA	RESERVED
38	(26)	UNSIGNED	2	UPHASMPO	RESERVED
40	(28)	UNSIGNED	4	UPHASMPP	RESERVED

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCUPHA	0		1
UPHAAM	B		4
UPHAAMOD	9		4
UPHAAM24	9		5
UPHAAM31	9		5
UPHAAT2	B		3
UPHABAC	8		4
UPHABMSG	9		4
UPHABNF	8		4
UPHABPC	8		4
UPHABRL	8		4
UPHABSE	8		4
UPHABSR	8		4
UPHABSV	8		4
UPHABYTE	24		2
UPHACLM	9		4
UPHACLS	9		4
UPHAEATR	2		2
UPHAEID	4		2
UPHAELEN	0		2
UPHAENP	14		2
UPHAFLG	8		3
UPHALDRT	A		3
UPHALPT	10		2
UPHAMARK	9		4
UPHAMDA	B		4
UPHAMDR	B		4
UPHAMD24	B		5
UPHAMD31	B		5
UPHAM33I	9		4
UPHAPFL	8		2
UPHAPLN	C		2
UPHARLD	1C		2
UPHARLDA	1E		2
UPHARLDO	1E		3
UPHARLDP	20		3
UPHARM	B		4
UPHARMOD	9		4
UPHASDA	B		4
UPHASDR	B		4
UPHASD24	B		5
UPHASD31	B		5
UPHASMPA	26		2
UPHASMPO	26		3
UPHASMPP	28		3
UPHASTR	18		2
UPHASVPF	8		4
UPHASWT	9		3
UPHAVIF	2		3
UPHA23BA	24		3

INLCVIFD

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	8	INLCVIFD	
0	(0)	SIGNED	2	VIFDELEN	LENGTH OF V.I.F.
2	(2)	BITSTRING	2	VIFDEATR	RESERVED
		1...		VIFDVIF	VIF FOLLOWS
4	(4)	CHARACTER	4	VIFDEID	V.I.F. IDENTIFIER

Cross Reference:

Name	Hex Offset	Hex Value	Level
INLCVIFD	0		1
VIFDEATR	2		2
VIFDEID	4		2
VIFDELEN	0		2
VIFDVIF	2		3

LBRACCDS

LIBRARYACCESSCONTROLBLOCK

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	156	LBRACCDS	
0	(0)	ADDRESS	4	LBRINFO	ADDRESS OF LOT, SDT, LDT PRTS
4	(4)	ADDRESS	4	LBRCHID	ADDRESS USER CHAIN ID
8	(8)	ADDRESS	4	LBRIOAR	RESERVED
12	(C)	ADDRESS	4	LBRDTL	ADDRESS OF LOCK TABLE ENTRY
16	(10)	ADDRESS	4	LBRLAMB	ADDRESS OF LAMB
20	(14)	BITSTRING	1	LBRMF	MAIN FUNCTION
21	(15)	BITSTRING	1	LBRTYPE	LIBRARY TYPE
22	(16)	BITSTRING	1	LBRUSE	USAGE OF LIBRARY
23	(17)	BITSTRING	1	LBRCHAIN	SEARCH ORDER
24	(18)	BITSTRING	1	LBRERROR	FAILURE HANDLING
25	(19)	BITSTRING	1	LBRDFLT	DEFAULT HANDLING
26	(1A)	BITSTRING	2	LBRLOGUN	LOGICAL UNIT
28	(1C)	ADDRESS	4	LBREXUL	START ADDR OF EDT
32	(20)	BITSTRING	1	LBRRET	FOR INTERNAL USE ONLY
33	(21)	BITSTRING	1	LBRRET1	FOR INTERNAL USE ONLY
34	(22)	BITSTRING	1	LBRTUSE	FOR INTERNAL USE ONLY
35	(23)	BITSTRING	1	LBRTSTAT	RESERVED
36	(24)	CHARACTER	4	LBRSAVE2	FOR INTERNAL USE ONLY
40	(28)	BITSTRING	1	LBRDEFNE	DEFINITION STATUS
41	(29)	BITSTRING	1	LBRLEVEL	LEVEL OF ENTRY
42	(2A)	BITSTRING	1	LBRREPLC	REPLACE
43	(2B)	BITSTRING	1	LBRAPIOP	LIBR API OPTION
		1...		LBRAPIRQ	LIBR API FIND REQUEST
44	(2C)	SIGNED	4	LBRMOD	MODIFICATION LEVEL
48	(30)	CHARACTER	36	LBRLDT	ENTRY OF INLCLDTE
84	(54)	CHARACTER	44	LBRSDT	ENTRY OF INLCSDTE
128	(80)	ADDRESS	4	LBRWRK	POINTER TO WORKAREA
132	(84)	SIGNED	4	LBRLEN	LENGTH OF WORKAREA
136	(88)	CHARACTER	2	LBRPID	PARTITION ID
138	(8A)	CHARACTER	18	*	FOR FURTHER USE

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	16	LIBINFO	
0	(0)	ADDRESS	4	LBRINFOO	LOT PTR
4	(4)	ADDRESS	4	LBRINFOC	LDT PTR C LIBRARY
8	(8)	ADDRESS	4	LBRINFOP	LDT PTR P LIBRARY
12	(C)	ADDRESS	4	LBRINFOS	SDT PTR
MAIN FUNCTION					
1		HEX	80	LBRGET	GET LIBRARY/SUBLIB
1		HEX	40	LBRADLIB	ADD LIBRARY/SUBLIB
1		HEX	10	LBRUPDEX	UPDATE BACKLEVEL EDT
1		HEX	08	LBRREMOV	DELETE LIBRARY/SUBLIB
1		HEX	04	LBRLOCK	LOCK LIBRARY
1		HEX	02	LBRUNLOC	UNLOCK LIBRARY
1		HEX	01	LBREXTND	EXTEND LIBRARY

Constants:

Len	Type	Value	Name	Description
	LIBTYPE			
1	HEX	80	LBRCL	PHASE
1	HEX	40	LBRRL	OBJECT MODULE
1	HEX	20	LBRSL	SOURCE BOOK
1	HEX	10	LBRPL	PROCEDURE
1	HEX	08	LBRSCL	SYSTEM PHASE
1	HEX	04	LBRSRL	SYSTEM OBJECT MODULE
1	HEX	02	LBRSSL	SYSTEM SOURCE BOOK
1	HEX	01	LBRSPL	SYSTEM PROCEDURE
1	HEX	0F	LBRLBRTY	LBR ACCESS
1	HEX	88	LBRDUMP	DUMP
1	HEX	F0	LBRALLTY	ALL TYPES
1	HEX	FF	LBRALL	ALL OPEN LIBRARIES
	LIBUSE			
1	HEX	80	LBRSEARC	SEARCH LIBRARY/SUBLIB
1	HEX	40	LBRTO	TO LIBRARY/SUBLIB
1	HEX	20	LBRFROM	FROM LIBRARY/SUBLIB
1	HEX	10	LBRACC	ACCESS LIBRARY
1	HEX	F0	LBRUSER	USER LOT ID
1	HEX	08	LBRFIRST	FIRST LIB/SUBLIB
1	HEX	04	LBRNEXT	NEXT LIB/SUBLIB
1	HEX	02	LBRANY	ANY LIBDEF GIVEN
1	HEX	01	LBRCAT	CATALOG LIB/SUBLIB
	CHAIN			
1	HEX	80	LBRTEMP	TEMPORARY LIB DEFINITION
1	HEX	40	LBRPERM	PERMANENT LIB DEFINITION
1	HEX	20	LBRBOTH	PERMANENT LIB DEFINITION
1	HEX	10	LBRSYS	SYSTEM LIBRARY
1	HEX	01	LBRDYNC	DYNAMIC CHAIN
	DEFINE			
1	HEX	80	LBRDNOLD	OLD STATUS
1	HEX	40	LBRDNNEW	NEW STATUS
1	HEX	20	LBRDNDUM	DUMMY STATUS
	LEVEL			
1	HEX	80	LBRLIBLV	LIB LEVEL
1	HEX	40	LBRSUBLV	SUBLIB LEVEL
1	HEX	20	LBRBTHLV	BOTH LEVEL
	REPLACE			
1	HEX	80	LBRRPYES	REPLACE = YES
1	HEX	40	LBRRPNO	REPLACE = NO
	ERROR			
1	HEX	80	LBRERRRT	RETURN TO CALLER
1	HEX	40	LBRERRCA	CANCEL
	DEFAULT			
1	HEX	80	LBRDFLTY	CREATE LDT FOR OLD LIB
1	HEX	40	LBRDFLTN	DONT CREATE LDT OLD LIB
	RETURN INFORMATION			
1	HEX	80	LBRASSGN	LIBDEF FOR ASSGN SYSXLB
1	HEX	40	LBRMIGRT	OLD LIBDEF

Cross Reference:

Name	Hex Offset	Hex Value	Level
LBRACCDS	0		1
LBRAPIOP	2B		2
LBRAPIRQ	2B		3
LBRCHAIN	17		2
LBRCHID	4		2
LBRDEFNE	28		2
LBRDFLT	19		2
LBRDTL	C		2
LBRERROR	18		2
LBREXUL	1C		2
LBRINFO	0		2
LBRINFOC	4		2
LBRINFOO	0		2
LBRINFOP	8		2
LBRINFOS	C		2
LBRIOAR	8		2
LBRLAMB	10		2
LBRLDT	30		2
LBRLEN	84		2
LBRLEVEL	29		2
LBRLOGUN	1A		2
LBRMF	14		2
LBRMOD	2C		2
LBRPID	88		2
LBRREPLC	2A		2
LBRRET	20		2
LBRRET1	21		2
LBRSAVE2	24		2
LBRSDT	54		2
LBRTSTAT	23		2
LBRTUSE	22		2
LBRTYPE	15		2
LBRUSE	16		2
LBRWRK	80		2
LIBINFO	0		1

LBRLCTDS

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	20	LBRLCTDS	
0	(0)	A-ADDRESS	4	LBRLINFO	ADDRESS OF LOT/LDT/SDT PTRS
4	(4)	A-ADDRESS	4	LBRLPID	ADDRESS OF PARTITION ID
8	(8)	BITSTRING	1	LBRLTYPE	LIBRARY TYPE
9	(9)	BITSTRING	1	LBRLUSE	USAGE OF LIBRARY
10	(A)	BITSTRING	1	LBRLCHN	SEARCH ORDER
11	(B)	BITSTRING	1	LBRLRET	RETURN INFORMATION
12	(C)	UNSIGNED	2	LBRLTIK	TEMP. SAVE OF TIK
14	(E)	BITSTRING	2		RESERVED
16	(10)	A-ADDRESS	4	LBRLCHID	USER-DEFINED LOT ID

Cross Reference:

Name	Hex Offset	Hex Value	Level
LBRLCHID	16	(10)	
LBRLCHN	10	(A)	
LBRLCTDS	0	(0)	
LBRLINFO	0	(0)	
LBRLPID	4	(4)	
LBRLRET	11	(B)	
LBRLTIK	12	(C)	
LBRLTYPE	8	(8)	
LBRLUSE	9	(9)	

LIBINFO

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	16	LIBINFO	
0	(0)	A-ADDRESS	4	LIBINFOO	LOT PTR
4	(4)	A-ADDRESS	4	LIBINFOC	LDT PTR C LIBRARY
8	(8)	A-ADDRESS	4	LIBINFOP	LDT PTR P LIBRARY
12	(C)	A-ADDRESS	4	LIBINFOS	SDT PTR

Cross Reference:

Name	Hex Offset	Hex Value	Level
LIBINFO	0	(0)	
LIBINFOC	4	(4)	
LIBINFOO	0	(0)	
LIBINFOP	8	(8)	
LIBINFOS	12	(C)	

LIBRHDR

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	216	LIBRHDR	LIBRARY HEADER ON BACKUP TAPE
0	(0)	SIGNED	2	LHDRLEN	LENGTH OF LIBRARY HEADER INCL. EXTENT INFO RECORDS
2	(2)	CHARACTER	4	LHDRDESR	LIBRARY HEADER ID: 'LHDR'
6	(6)	CHARACTER	7	LHDRLNAM	LIBRARY NAME
13	(D)	CHARACTER	1		RESERVED
14	(E)	CHARACTER	6	LHDRVLID	VOLID OF FIRST EXTENT
20	(14)	CHARACTER	44	LHDRFLID	ORIGINAL FILE-ID OF LIBRARY
64	(40)	SIGNED	4	LHDRBLKS	# OF LB BLOCKS TOTALLY OCCUPIED
68	(44)	CHARACTER	4		RESERVED
72	(48)	CHARACTER	144	LHDLDES	INFO FROM LIBRARY DESCRIPTOR RECORD
72	(48)	CHARACTER	8	LHDRCRID	LIBRARY CREATOR ID
80	(50)	CHARACTER	10	LHDRCRDT	TIME-STAMP OF CREATION DATE
90	(5A)	CHARACTER	2		RESERVED
92	(5C)	CHARACTER	4	LHDRLEVL	VERSION LEVEL
96	(60)	CHARACTER	2		RESERVED
98	(62)	SIGNED	2	LHDRLBCF	LENGTH OF LB CONTROL INFORMATION FIELD
100	(64)	UNSIGNED	1	LHDRLBSZ	LIBRARY BLOCK SIZE
101	(65)	CHARACTER	3		RESERVED
104	(68)	CHARACTER	8	LHDRCMPR	NAME OF COMPRESS ROUTINE
112	(70)	SIGNED	2	LHDRNOSL	# OF SUBLIBRARIES
114	(72)	CHARACTER	6		RESERVED
120	(78)	SIGNED	4	LHDRIDEN	RESERVED
124	(7C)	CHARACTER	92		RESERVED

Cross Reference:

Name	Hex Offset	Hex Value	Level
LHDLDES	72	(48)	
LHDRBLKS	64	(40)	
LHDRCMPR	104	(68)	
LHDRCRDT	80	(50)	
LHDRCRID	72	(48)	
LHDRDESR	2	(2)	
LHDRFLID	20	(14)	
LHDRIDEN	120	(78)	
LHDRLBCF	98	(62)	
LHDRLBSZ	100	(64)	
LHDRLEN	0	(0)	
LHDRLEVL	92	(5C)	
LHDRLNAM	6	(6)	
LHDRNOSL	112	(70)	
LHDRVLID	14	(E)	
LIBRHDR	0	(0)	

LOTENTRY

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	6	LOTENTRY	ENTRY OF LIBR. OFFSET TABLE
0	(0)	CHARACTER	6	LOTENTR	ENTRY LOT
0	(0)	SIGNED	4	LOTETLPT	TASK LOT POINTER
0	(0)	BITSTRING	2	LOTEOLDT	OFFSET IN LDT
2	(2)	BITSTRING	2	LOTEOSDT	OFFSET IN SDT
4	(4)	SIGNED	2	LOTETIK	TASK LOT POINTER
4	(4)	BITSTRING	2	LOTEFLAG	RESERVED
4	(4)	BITSTRING	1	LOTIREQ	INDIVIDUAL REQUEST
		1...		LOTIFREE	IMMEDIATE SPACE RECL.@D14LDFB
5	(5)	BITSTRING	1	LOTIACC	INDIV. ACCESS RIGHTS
		1...		LOTICCF	RESERVED
		.111			RESERVED
		.... 1...		LOTMIGRT	MIGRATION "LIBDEF"
		.... .1..		LOTRACT	ACTIVE ENTRIES IN ROW
		.... ..1.		LOTASSGN	ENTRIES INSERTED DUE TO ASSGN
		1		LOTCHPR	CHAIN IN PROCESS

Cross Reference:

Name	Hex Offset	Hex Value	Level
LOTASSGN	5	X'02'	
LOTCHPR	5	X'01'	
LOTEFLAG	4	(4)	
LOTENTR	0	(0)	
LOTENTRY	0	(0)	
LOTEOLDT	0	(0)	
LOTEOSDT	2	(2)	
LOTETIK	4	(4)	
LOTETLPT	0	(0)	
LOTIACC	5	(5)	
LOTICCF	5	X'80'	
LOTIFREE	4	X'80'	
LOTIREQ	4	(4)	
LOTMIGRT	5	X'08'	
LOTRACT	5	X'04'	

LPTROW

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	12	LPTROW	LIBRARY POINTER TABLE
0	(0)	A-ADDRESS	4	LPTXLOTT	TEMP LOT
0	(0)	A-ADDRESS	4	LPTBXDT	BEGIN ADDR
4	(4)	A-ADDRESS	4	LPTXLOTP	PERM LOT
4	(4)	A-ADDRESS	4	LPTEXDT	END ADDR
8	(8)	A-ADDRESS	4	LPTXLOTS	SYSTEM LOT
8	(8)	A-ADDRESS	4	LPTFXDT	NEXT FREE ENTRY

Cross Reference:

Name	Hex Offset	Hex Value	Level
LPTBXDT	0	(0)	
LPTEXDT	4	(4)	
LPTFXDT	8	(8)	
LPTROW	0	(0)	
LPTXLOTP	4	(4)	
LPTXLOTS	8	(8)	
LPTXLOTT	0	(0)	

MEMBHDR

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	104	MEMBHDR	MEMBER HEADER ON BACKUP TAPE
0	(0)	SIGNED	2	MHDRLEN	LENGTH OF MEMBER HEADER
2	(2)	CHARACTER	4	MHDRDESR	MEMBER HEADER ID: 'MHDR'
6	(6)	CHARACTER	2		RESERVED
8	(8)	CHARACTER	7	MHDRLNAM	LIBRARY NAME
15	(F)	CHARACTER	1		RESERVED
16	(10)	CHARACTER	8	MHDRSLNM	SUBLIBRARY NAME
24	(18)	CHARACTER	80	MHDRDENT	INFO FROM MEMBER DIRECTORY RECORD
24	(18)	CHARACTER	38	MHDRSEGM	MHDR FIRST SEGMENT
24	(18)	CHARACTER	8	MHDRNAM	MEMBER NAME
32	(20)	CHARACTER	8	MHDRTYPE	MEMBER TYPE
40	(28)	CHARACTER	1	MHDRGEN	RESERVED FOR GENERATION G.
41	(29)	CHARACTER	1	MHDRDEF1	ATTRIBUTES FOR DIR.ENTRY
		11..			RESERVED
		..1.		MHDREDIR	TYPE OF ENTRY = DIRECTORY
		...1 1111			RESERVED
42	(2A)	CHARACTER	2		RESERVED
44	(2C)	SIGNED	2	MHDRVIFL	LENGTH OF MHDR COMMON SEGMENT
46	(2E)	CHARACTER	2		RESERVED
48	(30)	BITSTRING	1	MHDRDEF2	FLAGS
		1...		MHDRMSH1	MODULE UNDER MSHP CONT.
		.1..		MHDRMSH2	CHANGED WITH MSHP BYPASS
		..11			RESERVED
		.... 1...		MHDRRTF	REC.TYPE=FIXED
		.... .1..		MHDRRTU	REC.TYPE=UNDEFINED
		.... ..1.		MHDRRTV	REC.TYPE=VARIABLE
		1		MHDRSIPT	SYSIPT DATA IN MEMBER
49	(31)	BITSTRING	1	MHDRDEF3	RESERVED
		1...		MHDRVIF	ONE OR MORE VIFS IN DE
		.1..		MHDRCMPR	MEMBER IS COMPRESSED
		..11 1111			RESERVED
50	(32)	SIGNED	2	MHDRMBLK	NO. OF LIBRARY BLKS FOR MEMBER
52	(34)	SIGNED	4	MHDRNORL	NO OF LOGICAL RECORDS
56	(38)	SIGNED	4	MHDRRLEN	LOGICAL REC. LENGTH
60	(3C)	SIGNED	2		RESERVED
62	(3E)	CHARACTER	10	MHDRDORI	TIME STAMP ORIGINATION
72	(48)	CHARACTER	10	MHDRDUPD	TIME STAMP LAST UPDATE
82	(52)	CHARACTER	2	MHDRVEMO	VERSION MODIFICATION
84	(54)	CHARACTER	2	MHDREODA	SYSIPT END OF DATA
86	(56)	CHARACTER	4	MHDRIDEN	RESERVED
90	(5A)	CHARACTER	2		RESERVED
92	(5C)	SIGNED	4	MHDRSTRT	START BYTE OF RLD IN PHASE
96	(60)	CHARACTER	8		RESERVED
104	(68)	CHARACTER	0	MHDRAREA	BEGIN OF USER AREA

Cross Reference:

Name	Hex Offset	Hex Value	Level
MEMBHDR	0	(0)	
MHDRAREA	104	(68)	
MHDRCMPR	49	X'40'	
MHDRDEF1	41	(29)	
MHDRDEF2	48	(30)	
MHDRDEF3	49	(31)	
MHDRDENT	24	(18)	
MHDRDESR	2	(2)	
MHDRDORI	62	(3E)	
MHDRDUPD	72	(48)	
MHDREDIR	41	X'20'	
MHDREODA	84	(54)	
MHDRGEN	40	(28)	
MHDRIDEN	86	(56)	
MHDRLEN	0	(0)	
MHDRLNAM	8	(8)	
MHDRMBLK	50	(32)	
MHDRMSH1	48	X'80'	
MHDRMSH2	48	X'40'	
MHDRNAM	24	(18)	
MHDRNORL	52	(34)	
MHDRRLEN	56	(38)	
MHDRRTF	48	X'08'	
MHDRRTU	48	X'04'	
MHDRRTV	48	X'02'	
MHDRSEGM	24	(18)	
MHDRSIPT	48	X'01'	
MHDRSLNM	16	(10)	
MHDRSTRT	92	(5C)	
MHDRTYPE	32	(20)	
MHDRVEMO	82	(52)	
MHDRVIF	49	X'80'	
MHDRVIFL	44	(2C)	

SUBLHDR

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	72	SUBLHDR	SUBLIBRARY HEADER ON BACKUP TAPE
0	(0)	SIGNED	2	SHDRLEN	LENGTH OF SUBLIBRARY HEADER
2	(2)	CHARACTER	4	SHDRDESR	SUBLIBRARY HEADER ID: 'SHDR'
6	(6)	CHARACTER	7	SHDRLNAM	LIBRARY NAME
13	(D)	CHARACTER	1		RESERVED
14	(E)	CHARACTER	58	SHDRSLXE	INFO FROM SUBLIBRARY INDEX ENTRY
14	(E)	CHARACTER	8	SHDRNAME	SUBLIBRARY NAME
22	(16)	CHARACTER	10	SHDRCRDT	TIME-STAMP OF SUBLIBRARY CREATION
32	(20)	UNSIGNED	1		RESERVED
33	(21)	UNSIGNED	1		RESERVED
34	(22)	CHARACTER	4		RESERVED
38	(26)	BITSTRING	1	SHDRFLAG	FLAGS
		1...			
		.1..		SHDRRCLM	IMMEDIATE SPACE REUSAGE FOR MEMBERS
		..11 1111			
39	(27)	UNSIGNED	1		RESERVED
40	(28)	SIGNED	4	SHDRMBRN	# OF MEMBERS IN SUBLIBRARY
44	(2C)	SIGNED	4	SHDRRESV	# OF USED LBS
48	(30)	SIGNED	4		RESERVED
52	(34)	SIGNED	4	SHDRIDEN	RESERVED
56	(38)	CHARACTER	4		RESERVED
60	(3C)	SIGNED	4	SHDRLBSZ	LIBRARY BLOCK SIZE
64	(40)	CHARACTER	8		RESERVED

Cross Reference:

Name	Hex Offset	Hex Value	Level
SHDRCRDT	22	(16)	
SHDRDESR	2	(2)	
SHDRFLAG	38	(26)	
SHDRIDEN	52	(34)	
SHDRLBSZ	60	(3C)	
SHDRLEN	0	(0)	
SHDRLNAM	6	(6)	
SHDRMBRN	40	(28)	
SHDRNAME	14	(E)	
SHDRRCLM	38	X'40'	
SHDRRESV	44	(2C)	
SHDRSLXE	14	(E)	
SUBLHDR	0	(0)	

TAPEITEM

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	6	TAPEITEM	GENERAL TAPE ITEM LAYOUT
0	(0)	SIGNED	2	TITEMLEN	LENGTH OF TAPE ITEM
2	(2)	CHARACTER	4	TAPEDESR	TAPE ITEM DESCRIPTOR

Cross Reference:

Name	Hex Offset	Hex Value	Level
TAPEDESR	2	(2)	
TAPEITEM	0	(0)	
TITEMLEN	0	(0)	

TOLCRQL

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	74	TOLCRQL	
0	(0)	A-ADDRESS	4	TOLCRQP	RESERVED
4	(4)	UNSIGNED	1	TOLCFDBC	FEEDBACK CODE
5	(5)	BITSTRING	1		UNUSED
6	(6)	SIGNED	2	TOLCRTP	ERROR CODE
8	(8)	SIGNED	4	TOLCRC	RETURN CODE
12	(C)	A-ADDRESS	4	TOLCSAP	RESERVED
16	(10)	A-ADDRESS	4	TOLCACBP	ADDRESS OF ACCESS BLOCK
20	(14)	A-ADDRESS	4	TOLCLOT	ADDRESS OF LOT ENTRY
24	(18)	A-ADDRESS	4	TOLCLDTC	ADDRESS OF LDT C-ENTRY
28	(1C)	A-ADDRESS	4	TOLCLDTP	ADDRESS OF LDT P-ENTRY
32	(20)	A-ADDRESS	4	TOLCSDT	ADDRESS OF SDT ENTRY
36	(24)	A-ADDRESS	4	TOLCLNM	ADDRESS OF LIBRARY ENTRY
40	(28)	A-ADDRESS	4	TOLCSLU	ADDRESS OF SUBLIB ENTRY
44	(2C)	A-ADDRESS	4	TOLCEXT	ADDRESS OF EXTENT LIST
48	(30)	A-ADDRESS	4	TOLCDEV	ADDRESS OF DEVICE ENTRY
52	(34)	A-ADDRESS	4	TOLCCHID	ADDRESS OF USER LOT ID
56	(38)	A-ADDRESS	4	TOLCIOA	RESERVED
60	(3C)	A-ADDRESS	4	TOLCLAMB	ADDRESS OF LAMB
64	(40)	UNSIGNED	2	TOLCPIK	PARTITION IDENTIFICATION KEY
66	(42)	BITSTRING	1	TOLCLTYP	LIBRARY TYPE
		1...		TOLCCL	PHASE
		.1..		TOLCRL	OBJECT MODULE
		..1.		TOLCSL	SOURCE BOOK
		...1		TOLCPL	PROCEDURE
		.... 1...		TOLCALL	ALL TYPES
		.... .1..		TOLCDUMP	DUMP
		.... ..1.		TOLCJCL	JCL-TYPE CHAIN
		1		TOLCLBR	LIBRARIAN
67	(43)	BITSTRING	1	TOLCSTAT	STATUS FLAGS
		1...		TOLCFRST	FIRST CALL
		.111			RESERVED
		.... 1...		TOLCUSCL	CHAIN CLEARED
		.... .1..		TOLCCTCL	CHAIN CLEARED
		.... ..1.		TOLCACCL	CHAIN CLEARED
		1		TOLCSRCL	CHAIN CLEARED
68	(44)	BITSTRING	1	TOLCUSE	LIBRARY USAGE
		1...		TOLCCAT	CATALOG LIB/SUBLIB
		.1..		TOLCFROM	FROM LIB/SUBLIB
		..1.		TOLCTO	TO LIB/SUBLIB
		...1		TOLCSRCH	SEARCH LIB/SUBLIB
		.... 1...		TOLCACC	ACCESS LIB/SUBLIB
		.... .1..		TOLCUSER	USER LIB/SUBLIB
		11			RESERVED
69	(45)	BITSTRING	1	TOLCCHTP	CHAIN TYPE
		1...		TOLCPERM	PERMANENT CHAIN
		.111 1111			RESERVED
70	(46)	BITSTRING	1	TOLCLOTF	REQUEST FOR LOT FLAG
		1...		TOLCEOST	END-OF-STEP PROCESSING
		.1..		TOLCMIGT	MIGRATION
		..1.		TOLCEOJ	END-OF-JOB PROCESSING
		...1 1111			RESERVED
71	(47)	BITSTRING	1	TOLCLEVL	DEFINITION LEVEL
		1...		TOLCLIB	LIBRARY ENTRY
		.1..		TOLCSUBL	SUBLIBRARY ENTRY
		..11 1111			RESERVED
72	(48)	BITSTRING	1	TOLCDEFN	DEFINITION STATUS
		1...		TOLCDNEW	NEW DEFINITION
		.1..		TOLCDDUM	DUMMY DEFINITION
		..11 1111			RESERVED

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
73	(49)	BITSTRING	1	TOLCOP	REQUEST OPERATION
		1...			RESERVED
		.1..		TOLCXTND	EXTEND REQUEST
		..1.		TOLCADD	ADD REQUEST
		...1		TOLCLEAR	CLEAR REQUEST
		.... 1...		TOLCLIST	LIST REQUEST
		.... .111			RESERVED

Cross Reference:

Name	Hex Offset	Hex Value	Level
TOLCACBP	16	(10)	
TOLCACC	68	X'08'	
TOLCACCL	67	X'02'	
TOLCADD	73	X'20'	
TOLCALL	66	X'08'	
TOLCCAT	68	X'80'	
TOLCCHID	52	(34)	
TOLCCHTP	69	(45)	
TOLCCL	66	X'80'	
TOLCCTCL	67	X'04'	
TOLCDDUM	72	X'40'	
TOLCDEFN	72	(48)	
TOLCDEV	48	(30)	
TOLCDNEW	72	X'80'	
TOLCDUMP	66	X'04'	
TOLCEOJ	70	X'20'	
TOLCEOST	70	X'80'	
TOLCEXT	44	(2C)	
TOLCFDBC	4	(4)	
TOLCFROM	68	X'40'	
TOLCFRST	67	X'80'	
TOLCIOA	56	(38)	
TOLCJCL	66	X'02'	
TOLCLAMB	60	(3C)	
TOLCLBR	66	X'01'	
TOLCLDTC	24	(18)	
TOLCLDTP	28	(1C)	
TOLCLEAR	73	X'10'	
TOLCLEVL	71	(47)	
TOLCLIB	71	X'80'	
TOLCLIST	73	X'08'	
TOLCLNM	36	(24)	
TOLCLOT	20	(14)	
TOLCLOTF	70	(46)	
TOLCLTYP	66	(42)	
TOLCMIGT	70	X'40'	
TOLCOP	73	(49)	
TOLCPERM	69	X'80'	
TOLCPIK	64	(40)	
TOLCPL	66	X'10'	
TOLCRC	8	(8)	
TOLCRL	66	X'40'	
TOLCRQL	0	(0)	
TOLCRQP	0	(0)	
TOLCRTP	6	(6)	
TOLCSAP	12	(C)	
TOLCSDT	32	(20)	
TOLCSL	66	X'20'	
TOLCSLU	40	(28)	
TOLCSRCH	68	X'10'	
TOLCSRCL	67	X'01'	
TOLCSTAT	67	(43)	
TOLCSUBL	71	X'40'	
TOLCTO	68	X'20'	
TOLCUSCL	67	X'08'	
TOLCUSE	68	(44)	
TOLCUSER	68	X'04'	
TOLCXTND	73	X'40'	

UPDATCBL

Offsets Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	20	UPDATCBL	CONTROL BLOCK LBRUPDAT
0	(0)	A-ADDRESS	4	UPDATLIN	LIBINFO
4	(4)	A-ADDRESS	4	UPDATLMB	LAMB
8	(8)	CHARACTER	6	UPDATLTY	LEVEL
14	(E)	A-ADDRESS	1	UPDATFLG	FLAGS
		11..		UPDATSET	SET: 00-ACC/10-NOACC/01-DEL
		..11 1111			RESERVED
15	(F)	A-ADDRESS	1		RESERVED
16	(10)	A-ADDRESS	4	UPDATRIN	RETURN INFO.IN CASE OF FAILURE
16	(10)	UNSIGNED	2	UPDATRI1	
16	(10)	UNSIGNED	1	UPDATRCO	ORIGIN OF RETURN CODE
17	(11)	UNSIGNED	1	UPDATRCV	VALUE OF RETURN CODE
18	(12)	UNSIGNED	2		RESERVED

Cross Reference:

Name	Hex Offset	Hex Value	Level
UPDATCBL	0	(0)	
UPDATFLG	14	(E)	
UPDATLIN	0	(0)	
UPDATLMB	4	(4)	
UPDATLTY	8	(8)	
UPDATRCO	16	(10)	
UPDATRCV	17	(11)	
UPDATRIN	16	(10)	
UPDATRI1	16	(10)	
UPDATSET	14	X'C0'	

Diagnostic Aids

Two features are provided to ease the task of problem determination in the areas of Librarian services and library data structure: the notification of internal error situations via messages and the Librarian command TEST.
The messages identify the module in which the internal error condition is detected and provide additional information, like library or sublibrary name, if appropriate.
Additionally, the messages contain so-called FEEDBACK codes, which uniquely describe internal error situations. The FEEDBACK codes are also inserted into field LRPLFDBC of the requestor LRPL, so that they are available in a main storage dump.
The TEST command provides a tool to check the library contents for consistency and to trace the sequence of Librarian services on different levels of detail.

Feedback Codes

Feedback codes are 1-byte values between 1 and 255. They are listed as part of messages L150, L152, L157, and L158 to describe unexpected conditions detected by Librarian modules in general as a result of system internal errors.

Value	Dec	Hex	Description
* PARAMETER CHECKING			
FDBCCONN	1	1	ANY CONNECTION ACTIVE A service was called without any connection to the req. library object.
FDBCMBRC	2	2	CONNECTION TO MBR ACTIVE A service which is not applicable while connection-to-member holds was issued.
FDBCNOVC	3	3	NO VALID CONNECTION The indicated connection in the LRPL is not defined or incompatible to the current request.
FDBCNSUB	4	4	NO CONNECTION TO SUBLIB ACTIVE The used service needs a connection-to-sublibrary which does not exist.
FDBCNLIB	5	5	NO CONNECTION TO LIBRARY ACTIVE The used service needs a connection-to-library which does not exist.
FDBCNEWC	6	6	NEW CONNECTION W/O LBRACCES NEW A CONNECT=NEW request was issued without having done a previous LBRACCES DEFINE=NEW.
FDBCSIPT	7	7	invalid SYSIPT EXIT specification in call-interface block INLCPARB
FDBCSLOG	8	8	invalid SYSLOG EXIT specification in call-interface block INLCPARB
FDBCSLST	9	9	invalid SYSLST EXIT specification in call-interface block INLCPARB
FDBCSPCH	10	A	invalid SYSPCH EXIT specification in call-interface block INLCPARB
FDBCLTYP	11	B	LIBRARY TYPE MISSING The passed parameter LIBTYPE contains an invalid value or is incompatible to other parameters.
FDBCLUSE	12	C	LIBRARY USAGE MISSING The passed parameter LIBUSE contains an invalid value or is incompatible to other parameters.
FDBCAM31	13	D	API CALLED WITH AMODE 31 (API)
FDBCSTOR	15	F	MINIMUM BUFFER STORAGE NOT AVAIL. The area allocated for the buffer pool is too small. At least one buffer and the corresponding control blocks must be accommodated.
FDBCFREE	16	10	NO FREE BUFFER AVAILABLE A request for the allocation of one or more buffers cannot be satisfied.

FDBCPRBA	17	11	LB-# OUTSIDE RANGE The PRBA presented for an I/O request is not within the limits of the library extent(s) as given by OPEN.
FDBCINBF	18	12	INVALID BUFFER REQUEST
FDBCNGEN	21	15	GENERIC REQUEST W/O VALID ARGUMENT
FDBCSTLN	22	16	LENGTH FOR STOW TABLE TOO SMALL
FDBCSTNS	23	17	INVALID # OF STOW ENTRIES The number of stow entries passed to the library service has an invalid value.
FDBCGKEY	24	18	BLDL-SUBLIB REQ.,INF=KEY
FDBCISTP	25	19	INVALID STOW TYPE The STOW function was requested with a not defined stow type (main function).
FDBCRLEN	28	1C	invalid record length.
FDBCGDIR	29	1D	INCONSISTENT GET DIRECTORY INFORM. The position of the directory pointer for an INLMGDIR request is invalid.
FDBCLDSW	30	1E	Insufficient Workspace for INLPLDS
FDBCLRPL	31	1F	LRPL MISSING
FDBCLAMB	32	20	LAMB MISSING The control block LAMB was not passed to the requested service.
FDBCSTOW	33	21	STOW TABLE MISSING A service was requested which needs a stow table.
FDBCLACB	34	22	LACB MISSING A library is connected but the library access control block is missing.
FDBCSACB	35	23	SACB MISSING A sublibrary is connected but the sublibrary access control block is missing.
FDBCMACB	36	24	MACB MISSING A member is connected but the member access control block is missing.
FDBCLINF	37	25	LIBINFO MISSING The connection to a library/sublibrary cannot be done because the LIBINFO is missing or the addressed library/sub-library chain does not exist.
FDBCNOTE	38	26	INVALID NOTE WORD
FDBCWRKA	39	27	Workarea/Buffer missing
* PROCESSING ERROR			
FDBCSERV	40	28	USED SERVICE FAILED A used system service returned with a not expected return code.
FDBCXUPD	41	29	XUPDLIST IN ERROR
FDBCBFCH	42	2A	INCONSISTENT BUFFER CHAINS
FDBCIOER	43	2B	I/O ERROR

FDBCLOCK	44	2C	RESOURCE ALREADY LOCKED The request was specified with LOCK= RETURN but the resource was not available or the LOCK service failed.
FDBCOPFA	45	2D	OPEN FAILURE
FDBCCLFA	46	2E	CLOSE FAILURE
FDBCLNID	47	2F	LIBRARY NOT IMPLICITLY DELETED
FDBCMSLK	48	30	Member supervisor locked.
* LIBRARY VERIFICATION			
FDBCNOLB	51	33	DATA SET IS NOT A VSE LIBRARY The accessed data set is no valid library.
FDBCINTR	52	34	NO VALID INDEX ENTRY TYPE
FDBCDTYP	53	35	DUPLICATE TYPE ENTRY
FDBCNSTP	54	36	NO STARTING TYPE ENTRY IN LB
FDBCMINX	55	37	INCONSISTENCY IN MEMBER INDEX The consistency check of a library block of the member index failed.
FDBCLBEM	56	38	EMPTY LB
FDBCNXLV	57	39	INVALID # OF MEMBER INDEX LEVELS
FDBCRECL	58	3A	INVALID DASD SPACE MGMT REQUEST The PRBA presented for a free space request is outside library limits or it cannot be freed (library control blocks)
FDBCTPMX	59	3B	NO MBRX ENTRY AFTER TYPE ENTRY
FDBCLBIV	60	3C	INVALID LB DATA INVARIANT
FDBCMBX0	61	3D	ZERO PRBA IN MBRX ENTRY
FDBCIDEN	62	3E	LIBRARY BLOCK IDENTIFIER WRONG The library block which has been read has an LBID value which does not match the value contained in the descriptor.
FDBCDIRL	63	3F	DIRECTORY ENTRY IS TOO LONG
FDBCLKSC	64	40	SPECIFIED LOCKID IS NOT ALLOWED
FDBCFULL	65	41	LIBRARY IS FULL
FDBCSLCK	66	42	LIBRARY/SUBLIB CONTAINS LOCKED MEMBERS
FDBCMLCK	67	43	Member is locked (API)
FDBCLKID	68	44	Lockid is invalid
FDBCDMDI	69	45	duplicate member directory exists A request for allocation of one or more LB(s) cannot be satisfied.
FDBCNXMB	70	46	MEMBER DOES NOT EXIST
FDBCNXSB	71	47	SUBLIB DOES NOT EXIST A service was issued for a sublibrary which does not exist.
FDBCNXLB	72	48	LIBRARY DOES NOT EXIST A service was issued for a library which does not exist (or is being defined or deleted).
FDBCVIFL	73	49	VIFS INCONSISTENT
FDBCMLEN	74	4A	MEMBER EXCEEDS MAXIMAL LENGTH

```
*  ENVIRONMENT

FDBCGVIS    100      64         GETVIS SPACE EXHAUSTED
                                A GETVIS or INLMDSTO request failed.
FDBCLCTO    105      69         LIBRARY CONTROL TABLES OVERFLOW
                                An insertion in a library control
                                table fails because of overflow.
FDBCUNUN    106      6A         LIB/SUBLIB NOT UNIQUELY ASSIGNED
FDBCLUEX    107      6B         LOGICAL UNITS EXHAUSTED
FDBCVNMO    108      6C         VOLUME NOT MOUNTED
FDBCSPIB    109      6D         ERROR DURING SCANNING PIB

*  YEAR 2000 SUPPORT

FDBCCENT    112      70         INCONSISTENT DATE

*  PARAMETER CHECKING (LCT SERVICES)

FDBCDFLT    151      97         INVALID DEFAULT PARAMETER
                                The parameter DEFAULT of the LBRACCES
                                macro contains an invalid value.
FDBCMFUN    152      98         INVALID MAIN FUNCTION
                                The requested main function for the
                                called service is not defined.
FDBCFNME    153      99         MISSING/INVALID FILE NAME
                                The parameter FILENAM for an ADD or
                                EXTEND request of the LBRACCES or
                                LBRCTUPD macros is missing.
FDBCHAIN    154      9A         INVALID CHAIN PARAMETER
                                The CHAIN parameter of the LCT service
                                contains an invalid value.
FDBCNEXT    155      9B         INVALID CHAINING REQUEST
FDBCSLIB    156      9C         MISSING SUBLIB NAME
                                The parameter SUBLIB for an ADD
                                request of the LBRACCES or LBRCTUPD
                                macros is missing.
FDBCEDTE    157      9D         MISSING EXTENT ENTRIES
                                The parameter EXTENTS for an ADD or
                                EXTEND request for the LBRCTUPD macro
                                is missing.
FDBCDDTE    158      9E         MISSING DEVICE CHAR. ENTRIES
                                The DEVCHAR parameter of the
                                LBRCTUPD macro is missing.
FDBCUSET    159      9F         INCORRECT LBRUPDAT-SET
FDBCULEV    160      A0         INCORRECT LBRUPDAT-LEVEL
FDBCULIB    161      A1         INCORRECT LBRUPDAT-LIBINFO
FDBCEDTM    162      A2         EDT MISSING
FDBCDDTM    163      A3         DDT MISSING
FDBCTYPF    164      A4         INVALID TYPFLE
FDBCDDTY    165      A5         EXTENT ON DIFFERENT DEVICE TYPE
FDBCTYPE    166      A6         INVALID/ missing Type(API)
FDBCMEMB    167      A7         INVALID/ missing Member (API)
```

```
*  INPUT CHECKING (LCT SERVICES)

FDBCUNAL    174      AE         INCONSISTENCY: LIB/NOACC-LDTFLAGS
FDBCUNAS    175      AF                        SUBLIB/NOACC-SDTFL.
FDBCUACL    176      B0                        LIB/ACC-LDTFLAGS
FDBCUACS    177      B1                        SUBLIB/ACC-SDTFLAGS
FDBCUDEL    178      B2                        LIB/DEL-LDTFLAGS
FDBCUDES    179      B3                        SUBLIB/DEL-SDTFLAGS
FDBCEXST    180      B4         MORE THAN 16 EXTENT STATEMENTS
FDBCNOEX    181      B5         LIBRARY IS NOT EXTENDABLE
FDBCNOVS    182      B6         LABEL FOR LIB EXTENSION NOT VSAM
FDBCLAMI    183      B7         DLBL/EXTENT STATEMENT MISSING
FDBCINEX    184      B8         INCORRECT EXTENT STATEMENT
FDBCINDL    185      B9         INCORRECT DLBL STATEMENT
FDBCMIEX    186      BA         EXTENT STATEMENT MISSING
FDBCUNPA    187      BB         LDT ENTRY WITHOUT PARTITION FLAGS
FDBCLDTM    188      BC         REFERENCED LDT ENTRY MISSING
                                The LDT entry referenced by LIBINFO
                                is missing.

*  INPUT CHECKING (LEVEL 3 SERVICES)

FDBCL3P1    201      C9         MORE THAN 32 PARAMETERS PER COMMAND
                                IN PARSER TABLE
FDBCL3P2    202      CA         MORE THAN 10 EXCLUDING ALTERNATIVE
                                PARAMETERS PER COMMAND IN PARSER TABLE
FDBCFOUT    203      CB         INVALID FORMATTED OUTPUT EXIT
                                The specification of the formatted
                                output exit in the call interface
                                control block INLCPARB is invalid.
FDBCSAOV    211      D3         STAND-ALONE PHASE >= 64K
                                A stand-alone phase must be smaller
                                than 64k because the maximum tape
                                block size is 64k.
FDBCINVT    212      D4         TAPE BUFFER CONTAINS INVALID DATA
                                the contents of a tape buffer has
                                been destroyed.
FDBCINVR    213      D5         INCORRECT RLD POINTER
                                the RLD pointer points to an address
                                outside the range of the phase.
FDBCIPOV    214      D6         IPL BOOTST> PHASE > 16K
FDBCORDR    215      D7         $SVASA NOT IN REQUIRED ORDER
FDBCALFA    231      E7         NO ALPHANUMERIC STRING (API)
FDBCMTYP    232      E8         RENAME WITHOUT TARGET MEMBER SPEC. (API)
FDBCNTST    233      E9         NOTE-STACK OVERFLOW/UNDERFLOW(API)
FDBCMOPN    234      EA         MEMBER NOT OPENED (API)
FDBCRFMI    235      EB         RECORD FORMAT MISMATCH (API)
FDBCVIFE    236      EC         ERROR IN PHASE VIF     (API)
FDBCMCHN    238      EE         MISSING CHAIN AREA     (API)
FDBCCHID    242      F2         CHAIN ID MISSING       (API)
FDBCMLIB    246      F6         LIB/SUBLIB MISSING     (API)
FDBCINSQ    248      F8         INVALID MACRO SEQUENCE (API)
FDBCNSTG    249      F9         INVALID OPEN NESTING   (API)
FDBCIALC    250      FA         INVALID IALC REQUEST(API)
FDBCSTSK    255      FF         NOT FOR SYSTEM TASKS
```

```
*  ADDITIONAL CODES IN IJBLBHLS

GETVISFL     50      32         GETVIS failed, Return Code in LBRRET1
GETVCEFL     61      3D         GETVCE failed, Return Code in LBRRET1
CTUPDFL      69      45         Update LCT's failed in Procedure ADDLIB
LCTACFLF     76      4C         IBJLBLLS failed in Procedure GETFILIB
                                ( GET first Library )
LCTACFLX     81      51         IBJLBLLS failed in Procedure GETNXLIB
                                ( GET next Library )
LCTACFLT     86      56         IBJLBLLS failed in Procedure GETTOLIB
                                ( GET TO  Library )
LCTACFLR     91      5B         IBJLBLLS failed in Procedure GETFRLIB
                                ( GET FROM Library )
LCTACFLN     95      5F         IBJLBLLS failed in Procedure GETNWLIB
                                ( GET NEW Library )
```

TEST Command

Syntax Description

The TEST command has the syntax:

```
Test  {Library = libname ... [Area={Space}]  [Repair={Yes}]     }
      {                      [     {All}  ]  [       {No} ]     }
      {                                                         }
      {Sublibrary = libname.sublibname ...                      }
      {                                                         }
      {membername.typename ...                                  }
      {                                                         }
      {Trace = {Buffer}                                          }
      {        {Space }                                         }
      {        {Io    }                                        }
      {        {LEVEL1}                                          }
      {        {LEVEL2}                                          }
      {        {All   }                                         }
      {        {Off   }
                                 {PARTition={xx|ALL} | TASK=yy}

      }
      [Unit=SYSLOG|SYSLST]
```

Command name and operand keywords may be shortened by leaving off one or more letters from right to left, so long as the name remains unique. The part of the name which must be coded is in capitals, the rest in lower case. For a detailed description of the syntax rules refer to *VSE/ESA System Control Statements*.

libname — is the name of the library as in the DLBL statement. The Test command for Lib has two different functions which are specified with the Area operand:

Area=Space — displays or lists the descriptors of the library DASD space (space and extent descriptors) together with the allocation status of each library block (bit map).

Area=All — performs an integrity and consistency check on all library items (members, bit map, library and sublibrary descriptors, index and directory entries). Detected incompatibilities are displayed or listed.
Area=All is the default value.

Repair=Yes — If the bit map shows error #402, that is, a library block which does not occur within any chain is indicated as reserved, the affected library block is freed. Also, a wrong value for the free space in the space descriptor (error #056) is corrected.
This option is only relevant for Area=All and a uniquely assigned library, in all other cases it will be ignored.
Repair=NO is the default value.

The Area and Repair operands are only supported for the Test Lib function. In all other cases, they are ignored.

sublibname specifies the name of a sublibrary. An integrity test is performed on index and directory entries of the specified sublibrary. Additionally, index and directory entries are displayed or listed.
The AREA and REPAIR options are ignored.

membername is the name of a member (can be generic)

typename is the name of a member type (can be generic)
In this format the Test command displays or prints the member descriptor and the chain of the library blocks which contain the member data. Any inconsistencies are reported.
The AREA and REPAIR options are ignored.

Trace specifies which trace information shall be provided. The trace is not only active for the task which has given the TEST command but for all tasks requesting Librarian services in all VSE partitions within one CPU. It can be changed by any task. The AREA and REPAIR options are ignored. The trace information consists of the task identification key, an address of the main control block or the library block under consideration, the function identification, and, optionally, further information dependent on the called function.
A trace can be requested for different levels of detail:

Buffer specifies that all I/O buffer requests shall be traced.

Space specifies that all requests for reserving or releasing library blocks shall be traced.

Io specifies that all read and write request for library blocks shall be logged (exception: the formatting is not logged).

LEVEL1 requests a trace for the Level 1 services (buffer, space and I/O requests).

LEVEL2 requests a trace for the Level 2 services.

All requests that Level 1 and Level 2 services be traced.

Off deactivates all trace specifications.

Unit The output of the Test command can be routed either to SYSLOG or to SYSLST. Default is SYSLOG if the command was entered from SYSLOG, otherwise it is SYSLST. If the trace function is active, output on SYSLST is only possible when the program which requests Librarian Level 1 and Level 2 services has specified a DTF in the LAMB.

PART/TASK The PARTition and TASK operand are optional.
Maximal one parameter (PARTition or TASK) can be specified. If both are specified, the PARTition parameter will be ignored.
The PARTition and TASK operand are ignored for TEST TRACE=OFF.
If PARTition=xx is specified, the TRACE will be done for the partition specified.
for PARTition=ALL the trace will be done for all partitions.
If TASK=yy is specified, the TRACE will be done for the task specified only.
xx represents a two character field for the specification of a partition (e.g. BG, F1, C4, ...).
yy represents a hexadecimal number for the specification of the taskid (corresponds to the TIK value displayed by the trace messages).
If the PARTition and TASK operand are omitted, the trace will only be done for the partition where the TEST commsnd had been entered for all tasks of the partition.
Maximal 10 TRACE commands without PARTition=ALL can be active in the system.
If two subsequent TRACE commands with the same PARTition or TASK specification are given, the first TRACE command will be overwritten. If PARTition=ALL is specified, this does not overwrite any trace specification for explicit tpartition specifications (e.g. PART=F1) but only for PARTition=ALL.

Examples

In the following examples, the TEST command was applied to a library called SERVLIB, in which two sublibraries, S1$XE8 and S2$XE8, are defined. Several members are catalogued into the sublibraries, and due to forcing cancelation of a catalog job during execution (operator CANCEL with option FORCE or abnormal termination because of a system malfunction), some library blocks are marked as allocated in the disk storage map, although they do not belong to any catalogued member.

Command entered:

```
          TEST L=SERVLIB AREA=SPACE
```

Output listing:

```
TEST  REPORT             LIBRARY = SERVLIB              DATE: 1997-07-01
                                                        TIME: 12:14
------------------------------------------------------------------------
FILE-ID  =  SERVICE-LIBRARY-REL4

EXTENT  VOLID        DISK-ADDRESS-RANGE    PRBA-RANGE
001     SYSRES         30.00 -   39.18     00000000 - 00000687

 LIBRARY = SERVLIB
 LIBRARY DESCRIPTOR:
     LABEL                = LIBRARY
     LIBRARY NAME         = SERVLIB
     LIBRARY ID           = 00000000
     DESCRIPTOR LENGTH    = D8
     NUMBER OF SUBLIBS    = 002
     SUBLIB INDEX START   = 00D8 00000000
     LIBRARY BLOCK SIZE   = 0A
     CONTROL FIELD LENGTH = 24
     COMPRESS MODULE      = INLPCMPR
     CREATION DATE        = 19960723
```

```
FREE SPACE DESCRIPTOR:
    CONTROL BLOCK ID     = BITM
    DESCRIPTOR LENGTH    = 48
    LIB. HEADER BLOCKS   = 02
    ALLOCATED BLOCKS     = 000688
    AVAILABLE BLOCKS     = 0000EB
    FREE SPACE BEGIN     = 000000
    HIGH WATER MARK:
      NEXT FREE POSITION = 0000067F
      HIGH END REACHED   = YES
    BITMAP START MARK    = --->
EXTENT DESCRIPTOR:
    EXTENT LABEL         = LIB-EXT-0001
    LOWEST BLOCK NUMBER  = 00000000
    HIGHEST BLOCK NUMBER = 00000687
    LOWEST DISK ADDRESS  = 001E0000
    HIGHEST DISK ADDRESS = 00250012
    BLOCKS FOR BITMAP    = 01
    BYTES FOR BITMAP     = 0000D1
    BITS FOR BITMAP      = 000688
    1. ALLOCATABLE BLOCK = 000001
    DESCRIPTOR LENGTH    = 40
    NUMBER OF TRACKS     = 0098

PRBA                  B I T  -  M A P                   DISK ADDRESS
000000    1111 1111 1111 1111    1111 1111 1111 1111         30.00.01
000020    1111 1111 1111 1111    1111 1111 1111 1111         30.02.11
000040    1111 1111 1111 1111    1111 1111 1111 1111         30.05.10
000060    1111 1111 1111 1111    1111 1111 1111 1111         30.08.09
000080    1111 1111 1111 1111    1111 1111 1111 1111         30.11.08
0000A0    1111 1111 1111 1111    1111 1111 1111 1111         30.14.07
0000C0    1111 1111 1111 1111    1111 1111 1111 1111         30.17.06
0000E0    1111 1111 11.............................................
.................................................................
0003A0    ........................... 1111 1111 1111         34.08.05
000620    1111 1111 1111 1111    1111 1111 1111 1111         37.09.07
000640    1111 1111 1111 1111    1111 1111 1111 1111         37.12.06
000660    1111 1111 1111 1111    1111 1111 1111 1110         37.15.05
000680    0000 0000                                          37.18.04
```

Note: In the Bit Map if a bit is set to 1, the corresponding library block is allocated; if a bit is set to 0, the corresponding library block belongs to the free space of the library. The disk address indicates the position of the first library block of each row.

Command entered:

```
          TEST L=SERVLIB  AREA=ALL
```

Output listing:

```
TEST  REPORT             LIBRARY = SERVLIB              DATE: 1997-07-01
                                                        TIME: 12:14
------------------------------------------------------------------------
FILE-ID  =  SERVICE-LIBRARY-REL4

EXTENT  VOLID        DISK-ADDRESS-RANGE    PRBA-RANGE
001     SYSRES         30.00 -   39.18     00000000 - 00000687

 *   VALIDATION OF LIBRARY SERVLIB  IN PROGRESS.
 **  VALIDATION OF SUBLIB  S1$XE8   IN PROGRESS.
 **  VALIDATION OF SUBLIB  S2$XE8   IN PROGRESS.
 *** VALIDATION OF SPACE/STATISTICS IN PROGRESS.
 ------------------------------------------------------------------------
 ERROR  MSG  DSECT NAME_OF           MISMATCH_OF       BLOCK     DISK
 MSG    NO.  NAME  LIBRARY_OBJECT    COMPARED_VALUES   NUMBER    ADDRESS
 ------------------------------------------------------------------------
 ERR==> #402 BITM: SERVLIB           &PRBA   |&BITON   00000598  36.16.03
 ERR==> #402 BITM: SERVLIB           &PRBA   |&BITON   00000599  36.16.04
 ERR==> #402 BITM: SERVLIB           &PRBA   |&BITON   0000059A  36.16.05
 ERR==> #402 BITM: SERVLIB           &PRBA   |&BITON   0000059B  36.16.06
 ERR==> #402 BITM: SERVLIB           &PRBA   |&BITON   0000059C  36.16.07
 ERR==> #402 BITM: ..................................................8
 ...........( lines skipped   ).....................................
 .....................VLIB           &PRBA   |&BITON   000005EA  37.04.08
 ERR==> #402 BITM: SERVLIB           &PRBA   |&BITON   000005EB  37.04.09
 ERR==> #056 BITM: SERVLIB           SPADAVAL|#BITS    00000001  30.00.02
 ****     85 DEFECTS DETECTED IN LIBRARY SERVLIB
```

Notes:

1. For a detailed description of values used for integrity checking see under ERROR MESSAGES OF TEST COMMAND later in this section.
2. If the TEST command would have been entered with the Repair=Yes option then the line '**** DEFECTS #402, #056 HAVE BEEN REPAIRED' would have occurred following the error report. A following TEST command would have shown zero defects.
3. If the TEST command is executed while the addressed library is used by another task whenever error messages occur they will be followed by the line
   '**** ERROR MESSAGES MAY HAVE BEEN CAUSED BY UPDATES CONCURRENT TO THIS RUN'.
   In this case, the REPAIR option is ignored. Because the space allocation and index update is not an indivisible operation the library will temporarily be in an inconsistent state which can cause the previous error messages. To get a reliable statement about the status of the library it is necessary to repeat the TEST command while no update is running on the library (e.g. while the library is uniquely assigned).

Command entered:

```
          TEST SUBLIB=SERVLIB.S2$XE8
```

Output listing:

```
 ----------------------------------------------------------------
 INDEX BLOCK  HIGHEST_KEY    BLOCK_NUMBER_OF            DISK
 LEVEL NUMBER NAME.....TYPE  LOWER_LEVEL_INDEX          ADDRESS
 ----------------------------------------------------------------
   2   000561         .PROC                       CYL  36.11.03
              $IPLMINA.            00052F
              $IPL370 .            000562
              STATUS  .            00053E
 OFFS=03A, LNFS=39E, NREC=004 ========>     OKAY
```

Note: This is a listing of the second index level. In this case it consists of one library block (block number 561) which contains three entries of type PROC. They are the highest entries in the level one index blocks, the PRBAs of which are given under the header Block_number_of_lower_level_index. For instance, $IPLMINA.PROC is the (alphanumerically) highest entry contained in block 52F (see below), which is the first block of the level 1 index.

Each library block is checked for consistency, e.g. whether the information in the library block control field states the actual reservation of the block. A block which is consistent gets the remark OKAY, otherwise NOT OKAY. Additionally the used data space, the free space and the number of records within the block are displayed. In this example, the library block shows the following reservation:

```
OFFS = offset to freespace = number of used data bytes = X'03A'
LNFS = length of freespace = number of unused bytes    = X'39E'
NREC = number of records within the library block      = X'004'
```

```
-----------------------------------------------------------------
INDEX BLOCK  MEMBER   MEMBER   MEMBER_BLOCK_CHAIN LOGICAL_RECORDS
LEVEL NUMBER NAME.....TYPE     START  END  NUMBER TYPE LENGTH NO.
-----------------------------------------------------------------
  1   00052F         .PHASE                      CYL  36.06.08
 36.07.11    $$BATTF0.         00053D 00053D 0001  U   0001E4 0001
 36.08.02    $$BATTF1.         00053F 00053F 0001  U   000223 0001
 36.08.03    $$BATTF2.         000540 000541 0002  U   0004B7 0001
 37.05.04    $IJBLBR .         0005F1 00067E 008E  U   0221A8 0001
 36.08.05    $LNKEDT .         000542 000560 001F  U   007478 0001
                   .PROC
 36.06.09    $ASIPROC.         000530 000530 0001  F   000050 0007
 36.06.10    $IPLMINA.         000531 000531 0001  F   000050 0010
OFFS=2E8, LNFS=0EB, NREC=009 ========>     OKAY
      000562         .PROC                       CYL  36.11.04
 36.06.11    $IPLMINB.         000532 000532 0001  F   000050 0010
 36.07.01    $IPLMINC.         000533 000533 0001  F   000050 0011
 36.07.02    $IPLMIND.         000534 000534 0001  F   000050 0011
 36.07.03    $IPLMINE.         000535 000535 0001  F   000050 0010
 36.07.04    $IPL370 .         000536 000536 0001  F   000050 0012
OFFS=172, LNFS=264, NREC=006 ========>     OKAY
      00053E         .PROC                       CYL  36.08.01
 36.07.05    $0JCLMIN.         000537 000537 0001  F   000050 000C
 36.07.06    $0JCL370.         000538 000539 0002  F   000050 0032
 36.07.08    $1JCLMIN.         00053A 00053A 0001  F   000050 0003
 36.07.09    LABELS  .         00053B 00053B 0001  F   000050 0013
 36.07.10    STATUS  .         00053C 00053C 0001  F   000050 0004
OFFS=172, LNFS=264, NREC=006 ========>     OKAY
NUMBER OF INDEX LEVELS:          2
NUMBER OF MEMBERS:            00011     OKAY
NUMBER OF USED BLOCKS:       0000C2     OKAY
NUMBER OF RECLAIMED BLOCKS:  00000     OKAY
```

Note: The level 1 index contains an entry for each member catalogued in the library. The TEST command listing above provides following information:
PRBA of index block and corresponding disk address for each member: member name, PRBA of first library block of member and disk address, PRBA of last library block of member, number of library blocks occupied by member, record type (F for fixed length, U for undefined), record length in bytes and number of records.
For instance, the last index block listed above occupies PRBA 53E which is at disk address 36.08.01. It contains five members all of type PROC. The first member listed is thus $0JCLMIN.PROC, starts at PRBA 537 at disk address 36.07.05. Since it only occupies 1 library block, the PRBA of its last block is also 537. It contains 12 (000C) records each of them has 80 bytes (000050).

Command entered:

```
          TEST SUBLIB=SERVLIB.S1$XE8
```

Output listing:

```
 ----------------------------------------------------------------
 INDEX BLOCK  HIGHEST_KEY    BLOCK_NUMBER_OF            DISK
 LEVEL NUMBER NAME.....TYPE  LOWER_LEVEL_INDEX          ADDRESS
 ----------------------------------------------------------------
   2   0000F7         .OBJ                         CYL  31.03.06
               IJBJCLCN.            000002
               IJBJC7  .            0000E1
               IJBLBULT.            0000F8
               IJJTEOF .            00032C
               IJJXCVH0.            000478
               INLPBBUF.            0003AD
               INLPBLDL.            00032D
               INLPCONN.            00032E
               INLPEXIT.            00032F
               INLPGST .            000330
               INLPLIPO.            000331
               INLPMDIS.            000332
               INLPMIRS.            000333
               INLPPUN .            000334
               INLPREAD.            000335
               INLPREST.            000336
               INLPSDL .            000337
               INLPTD  .            000338
               INLTACC .            000377
               INLTCONN.            000378
               INLTDEF .            000379
               INLTDSP .            00037A
               INLTMOVE.            00037B
               INLTRENR.            00037C
               INLXRDTP.            0003CD
               INLXWTO .            0003E8
 OFFS=1AA, LNFS=217, NREC=01B ========>     OKAY
 ----------------------------------------------------------------
 INDEX BLOCK  MEMBER   MEMBER   MEMBER_BLOCK_CHAIN LOGICAL_RECORDS
 LEVEL NUMBER NAME.....TYPE     START  END  NUMBER TYPE LENGTH NO.
 ----------------------------------------------------------------
   1   000002         .OBJ                         CYL  30.00.03
  35.11.10    DTSECTAB.         000497 000499 0003  F   000142 0008
  30.00.10    IJBCTUPD.         000009 00001C 0014  F   000142 003C
  30.02.08    IJBJC   .         00001D 000021 0005  F   000142 000E
  30.03.02    IJBJCB  .         000022 000029 0008  F   000142 0016
  30.03.10    IJBJCC  .         00002A 00002C 0003  F   000142 0008
  30.04.02    IJBJCH  .         00002D 000040 0014  F   000142 003C
  30.05.11    IJBJCI  .         000041 000048 0008  F   000142 0016
  30.06.08    IJBJCLCN.         000049 00004A 0002  F   000142 0005
```

```
OFFS=24A, LNFS=189, NREC=009 ========>     OKAY
      0000E1         .OBJ                          CYL  31.01.06
 30.06.10    IJBJC1  .         00004B 000063 0019  F   000142 004A
 30.09.02    IJBJC2  .         000064 00007B 0018  F   000142 0046
 34.03.01    INLTMOVE.         000365 000366 0002  F   000142 0004
OFFS=202, LNFS=1D2, NREC=008 ========>     OKAY
      00037C         .OBJ                          CYL  34.05.02
 34.03.03    INLTPARS.         000367 00036A 0004  F   000142 000C
 34.03.07    INLTPUN .         00036B 00036B 0001  F   000142 0002
 34.03.08    INLTPUNM.         00036C 00036C 0001  F   000142 0002
 34.03.09    INLTREN .         00036D 00036D 0001  F   000142 0002
 34.03.10    INLTRENC.         00036E 00036E 0001  F   000142 0002
 34.03.11    INLTRENP.         00036F 00036F 0001  F   000142 0002
 ..............................................................
```

Command entered:

```
          TEST MEMBER1.OBJ U=SYSLST
```

Output listing:

```
TEST  REPORT          SUBLIBRARY=R4LIB01.SL1          DATE: 1997-07-01
                                                      TIME: 13:26
------------------------------------------------------------------

EXTENT   VOLID   DISK-ADDRESS-RANGE   PRBA-RANGE
001      PAGEDS    29.00 -   32.13    00000000 - 0000030C

MEMBER DESCRIPTOR:

MEMBER NAME                 = MEMBER1
MEMBER TYPE                 = OBJ
GENERATION GROUP            =
ENTRY TYPE                  = 20
BEGIN PRBA                  = 0000 00000008       CYL  29.00.09
END PRBA                    = 0000 00000008       CYL  29.00.09
MSHP CONTROL                = NO
MSHP BYPASS                 = NO
RECORD TYPE                 = FIXED
SYSIPT DATA                 = NO
STATE OF DATA               = COMPRESSED
CONTIGUITY COUNTER          = 0001
NO. OF LIBRARY BLOCKS       = 0001
NO. OF LOGICAL RECORDS      = 00000007
LOGICAL RECORD LENGTH       = 00000050
ORIGINATION TIME STAMP      = 1996-11-04  14:42.32
LAST UPDATE TIME STAMP      =
VERSION/MODIFICATION        = 0000
SYSIPT END-OF-DATA          =
MEMBER IDENTIFICATION       = 00000009
LENGTH OF COMMON SEGMENT    = 0048

  PRBA    DISK-ADDR  BACKWARD PRBA  FORWARD PRBA   CONT  LREC (DEC)

00000008   29.00.09  0000 FFFFFFFF  0000 FFFFFFFF  0000     0
```

Note: CONT gives the contiguity counter of the library block.

Command entered:

```
          TEST $IJBLBSL.PHASE U=SYSLST
```

Output listing:

```
TEST  REPORT          SUBLIBRARY=IJSYSRS.SYSLIB       DATE: 1997-07-01
                                                      TIME: 13:36
------------------------------------------------------------------

EXTENT   VOLID   DISK-ADDRESS-RANGE   PRBA-RANGE
001      SYSRES     0.02 -   29.18    00000000 - 00001867

MEMBER DESCRIPTOR:

MEMBER NAME                 = $IJBLBSL
MEMBER TYPE                 = PHASE
GENERATION GROUP            =
ENTRY TYPE                  = 20
BEGIN PRBA                  = 0000 000008BD       CYL  10.15.05
END PRBA                    = 0000 000008BE       CYL  10.15.06
MSHP CONTROL                = YES
MSHP BYPASS                 = NO
RECORD TYPE                 = UNDEFINED
SYSIPT DATA                 = NO
STATE OF DATA               = UNCOMPRESSED
CONTIGUITY COUNTER          = 0002
NO. OF LIBRARY BLOCKS       = 0002
NO. OF LOGICAL RECORDS      = 00000001
LOGICAL RECORD LENGTH       = 00000460
ORIGINATION TIME STAMP      = 1996-11-04  14:42.32
LAST UPDATE TIME STAMP      =
VERSION/MODIFICATION        = 0000
SYSIPT END-OF-DATA          =
MEMBER IDENTIFICATION       = 000001AC
LENGTH OF COMMON SEGMENT    = 0048

V.I.F. IDENTIFIER           = INL1
V.I.F. LENGTH               = 002C
ATTRIBUTE FLAGS             = 60
SWITCH FLAGS                = 00
PHASE LENGTH                = 00000460
LOAD POINT AT LNKEDT TIME   = 00039078
ENTRY POINT AT LNKEDT TIME  = 000390D8
PART. START AT LNKEDT TIME  = 00039078
NO. OF RLD ITEMS            = 0000
PRBA OF RLD INFORMATION     = 0000 00000000

  PRBA    DISK-ADDR  BACKWARD PRBA  FORWARD PRBA   CONT  LRBA (HEX)

000008BD   10.15.05  0000 FFFFFFFF  0000 000008BE  0001  00000000
000008BE   10.15.06  0000 000008BD  0000 FFFFFFFF  0000  000003DC
```

Error Messages of the TEST Command

An error message of the TEST command has the following general format:

```
ERR==>msg-no dsect-name: name-of-library-object
             mismatch-of-compared-values block-number disk-address
```

where

msg-no
: is the number of the error message

dsect-name
: is the name of a descriptor or area in the library where the error has been detected. The following names can occur:

  LDES for the library descriptor (DSECT: INLCLDES)
  SPAD for the space descriptor (DSECT: INLCSPAD)
  EXTD for the extent descriptor (DSECT: INLCEXTD)
  SLXE for the sublib. descriptor (DSECT: INLCSLXE)
  INDX for a sublibrary index
  DENT for the member descriptor (DSECT: INLCDENT)
  MBR for a member
  LB for a library block
  BITM for the library block bit map

name-of-library-object
: is the name of the library, sublibrary or member, where the error has been detected.

mismatch-of-compared-values
: gives the identifiers which are checked against each other and which are inconsistent. If a value is taken out of a descriptor or control block, the name which describes the value in the corresponding DSECT is given. Otherwise, symbols are used which are explained below.

block-number and disk-address
: list the PRBA and the physical disk address (cylinder-track-record for CKD, physical block number for FBA devices), respectively, of that library block, where the mismatch was detected.

The following list contains the message number and the mismatching identifiers together with the error cause:

MESSAGE-NO	MISMATCH-OF-COMPARED-VALUES	ERROR CAUSE	
LDES			
001	LDESLABL	LIBRARY	The first library block does not contain the identification 'LIBRARY'.
003	LDESLNG	&CBLEN	The length of the library descriptor DSECT differs from the value given in the library descriptor.
004	LDESLBCF	&CBLEN	The length of the library block control field DSECT differs from the value given in the library descriptor.
005	LDESLBCF	LDTELBCF	The value for the length of the library block control field in the library descriptor and in the LDT-entry are different from each other.
006	LDESLBSZ	LDTELBSZ	The value for the library block size in the library descriptor differs from that in the LDT entry.
007	LDESLBSZ	X0A	The value for the library block size is unequal to X'0A'.
008	LDESCMPR	INLPCMPR	The name of the compress routine in the library descriptor is not 'INLPCMPR'.
009	LDESSXBN	0	The library block number of the sublibrary index start in the library descriptor is not 0.
011	LDESLNG	&LEN	The length of the library descriptor as given in the length field of the first library block differs from the value given in the library descriptor.
012	LDESSXOS	&LEN	The sublibrary index does not start beginning from the end of the library descriptor.
013	LDESNOSL	&SLIB	The library actually contains more sublibraries than is stated in the library descriptor.

014	LDESNOLB	#LB	The library header consists of more library blocks than is stated in the library descriptor.
015	LDESNLCK	#MLOCKED	The number of locked members in the Library is different from LDESMLCK

SPAD

051	SPADID	BITM	The space descriptor does not contain the identification 'BITM'
052	SPADLK	&CBLEN	The length of the space descriptor DSECT differs from the value given in the space descriptor.
053	SPADEND	--->	The end of the space descriptor is not marked by --->.
054	SPADBMCN	#LB	The actual number of library blocks used for the bit map is greater than the value given in the space descriptor.
055	SPADALLC	#BITS	The number of library blocks allocated to a library as indicated in the EDT differs from the value in the space descriptor.
056	SPADAVAL	#BITS	The actual number of free library blocks differs from the value in the space descriptor.
057	SPADHIWA	&PRBA	A used library block was found beyond the high water mark given in the space descriptor.

EXTD

101	&LEN	&CBLEN	The length of the extent descriptor in the library block differs from the corresponding DSECT length.
102	#LB	EBMBLK	The number of library blocks used for the bit map for this extent is unequal to the information in the extent descriptor.
103	EXTDID	LIB-EXT-	The extent descriptor does not start with the identification 'LIB-EXT-'.
104	EXTDLODA	EDTELODA	The extent descriptor contains another value for the lowest PRBA in this extent than the extent definition table entry.
105	EXTDHIDA	EDTEHIDA	The extent descriptor contains another value for the highest in this extent than the extent definition table entry.
108	EBMBIT	EBMBYT	In the extent descriptor the value for the number of bits used for the bit map in this is inconsistent to the value for the bytes used for the bit map.
109	EBMBIT	&LENSUM	The number of bits used for the bit map in this extent differs from the corresponding value in the extent descriptor.
110	EBMBIT	EDTEHIDA	In the extent descriptor the value for the number of bits used for the bit map in this extent is inconsistent to the information give in the EDT-entry.
111	EXTDLK	&CBLEN	The value of the length in the extent descriptor is different from its DSECT value.
112	EXTDNTRK	EDTENTRK	The number of tracks (CKD) or physical block (FBA) of this extent is different in the extent descriptor from the EDT entry.

113	EXTDNO	1-16	The extent identification number is different from '0001','0002',...,'0016'
114	#EXTD	#EDTE	The number of library extents is less than the number of EDT entries.
115	#EXTD	#EDTE	The number of library extents is greater than the number of EDT entries.

SLXE

151	SLXELXBNA	&PRBA	The address of the first library block of the sublibrary index mismatches to the value in the sublibrary descriptor.
152	SLXEMBRN	#MBR	The number of members contained in the sublibrary is different from the value in the sublibrary descriptor.
153	SLXERESV	#USED	The number of used library blocks for the sublibrary is different from the value in the sublibrary descriptor.
154	SLXERECL	#LB	The number of reclaimed library blocks for the sublibrary is different from the value in the sublibrary descriptor.
155	SLXEDEBN	SLXERECL	In the sublibrary descriptor the value for the number of reclaimed library blocks and for the start of the reclamation chain are contradictory.
158	LBCFLBOB	&CONT	In the reclamation chain the contiguity value is in error.
159	SLXEMLCK	#MLOCKED	The number of locked members in the sublibrary is different from SLXEMLCK.

INDX

201	&TYPEPOS	&TYPEEND	A type entry occurs as last entry in a library block of the member index.
202	&TYPEPOS	&TYPEBEG	A library block of the member index does not start with a type entry.
203	&TYPEPOS	&MBRXSUC	In the member index, the type entry is not followed by a higher member index entry (INLCMBRX) or by a directory entry (INLCDENT).
204	DESCTYPE	&ENTKIND	A library block in the member index contains an entry with undefined type.
205	TYPENAME	&TYPEORD	A type entry exist which does not fit in the alphameric ordering of the type entries.
206	MBRXHKEY	&MBRXORD	An index entry exist twice, in the number index or does not fit into its alphameric ordering.
207	#ENTRY	#LB	In the member index, the number of library blocks on one level does not correspond to the number of index entries in the next higher level.
208	MBRXHKEY	&ENTRY	In the member index, the last entry in a library block is not the one which is given in the index level above this level.

DENT

251	&LEN	&DLEN	The length of the member descriptor as it is given in the descriptor is unequal to the value of its length field.
252	DENTLSTP	&LSTPRBA	The last library block of a member has a different address from the address shown in its descriptor.
253	DENTDEF1	X20	The entry is not a member descriptor (X'20').
254	DENTRTF	&UNIQUE	The record type of a member is FIXED, but not unique.
255	DENTRTU	&UNIQUE	The record type of a member is UNDEFINED, but not unique.
256	DENTRV	&UNIQUE	The record type of a member is VARIABLE but not unique.
258	DENTDEF2	&UNIQUE	The record type of a member is not specified.
259	DENTRU	&TYPE	A member with type 'PHASE' or type 'DUMP' has not the record type UNDEFINED.
260	DENTVIF	PHASE	A member with type 'PHASE' does not contain phase information.
261	DENTVIFL	&CBLEN	The length of the basic part of a member descriptor is different from its DSECT-length (INLCDENT).
262	VIFDEID	INL	The identification part of the variable information field in a member descriptor does not start with the string 'INL'.
263	DENTCMPR	DENTRTF	A member is indicated as compressed but has not the record type FIXED.
264	DENTRLEN	DENTRTF	A member has the record type FIXED but a record length other than 80 bytes.
265	DENTCONT	DENTMBLK	The number of contiguous library block as indicated in the member descriptor exceeds the number of library blocks for the member.

267	DENTMBLK	#LB	The number of library blocks in a member exceeds the value given in its descriptor.
268	DENTNORL	#LEN	The number of records stored in a member is different from the value given in its descriptor.
269	DENTRLEN	DENTMBLK	For a member with record type UNDEFINED, the value in the descriptor for member length and for the number of used library blocks are contradictory.
270	UPHARLDP	&INVALID	The variable information field of a phase shows RLD items but an invalid address for them.
271	VIFDELEN	0	The vifs are chained incorrectly

LB

351	&LEN	&CBLEN	The length of the first record in the first library block for the bit map is different from the DSECT length.
352	&PRBA	EDTEHIDA	A library block has a PRBA which exceeds the value in the last extent definition table entry for this library.
353	LBCFLBNB	&LSTPRBA	The backward chain does not correspond to the forward chain.
354	INLCLBCF	&LBSIZE	The data invariant of a library block length of library block = length(used space)+length(free space)+length(control information) does not hold.
355	LBCFNREC	#LEN	The number of records within the library block is unequal to the value given in its control field.
356	LBCFOFFS	&LENSUM	The used space of a library block is unequal to the value given in its control field.
357	LBCFIDEN	0	The identification number of library block is zero.
358	LBCFCONT	&CONT	The number of contiguous library blocks is unequal to the value given in the control field.
359	LBCFLBNF	&CONT	The last library block of a chain is reached but the contiguity value is greater than zero.
360	LBCFLBNF	&CONT	A library block is indicated as contiguous to a previous one but it is not contiguous.
361	LBCFLBNF	&CONT	The number of contiguous library blocks given in the member descriptor is contradictory to the real value.
362	LBCFLBOF	0	The offset of the forward pointer is not zero.
363	LBCFLBOB	0	The offset of the backward pointer is not zero.

BITM

401	&PRBA	&BITOFF	A library block exists within a chain which is not indicated as used in the library block bit map.
402	&PRBA	&BITON	A library block is indicated as used in the library block bit map but does not occur within the chain of any member or index level.

Trace Entries

Trace entries of the TEST command have the following general layout:

```
TIK=task-id  address  FUNC=function-id  < parameter-values >
```

where

task-id gives the task identification key of the executing task.

address contains the address of the most important data area for the given request, i.e. for BUFFER traces the address of the buffer header (BHDR), for SPACE and I/O traces the PRBA of the corresponding library block, for LEVEL2 traces the address of the LRPL (exception: for the LBRACCES macro the pointer to the parameter block LBRACCDS).

function-id gives an abbreviation for the name of the macro (for Level 2 services) or module (for Level 1 services) which was invoked for the requested function; the abbreviation mostly consists of the last four characters of the corresponding name.

parameter-values (optional) contain the parameter values of the given request and depend on the function.

Trace Entries for Level 1 Services

Buffer Requests:

Get Buffer Request (INLPGBUF):

```
TIK=21 BUFF=08A800 FUNC=GBUF {S} {O}
                             {P} {N}

where: S=shared buffer, P=private buffer, O=old buffer, N=new buffer.
```

Free Buffer Request (INLPFBUF):

```
TIK=21 BUFF=08A800 FUNC=FBUF {P}
                             {F}

where: P=preempt queue, F=free queue.
```

Purge Buffer Request (INLPPBUF):

```
TIK=21 BUFF=07AC90 FUNC=PBUF {S}
                             {P}
```

where: S=shared buffers, P=private buffers.

Space Requests:

Get Space Request (INLPGSPA):

```
TIK=21   LB=00004F FUNC=GSPA R4LIB01
```

where: R4LIB01 is the library name.

Free Space Request (INLPFSPA):

```
TIK=21   LB=00004F FUNC=FSPA R4LIB01
```

where: R4LIB01 is the library name.

I/O Requests:

Read Library Block (INLPLBGP):

```
TIK=21   LB=00004A FUNC=LBRD IJSYSRS
```

where: IJSYSRS is the library name.

Write Library Block (INLPLBGP):

```
TIK=21   LB=000001 FUNC=LBWR R4LIB05
```

where: R4LIB05 is the library name.

Real SIO entries

```
TIK=21  CCB=0FE038 FUNC=SIO  IJSYSRS 00008A {S} {C}
                                            {P} {E}
                                                {F}
where: IJSYSRS is the library name.
       S = shared buffer, P private buffer
       C = CKD device, E = ECKD device, F = FBA device
       CCB = Address of CCB
```

Trace Entries for Level 2 Services:

Access library/sublibrary (LBRACCES):

```
TIK=21 ADDR=07CAC0 FUNC=ACCS {ADD} R4LIB01 SL1      {LBR} {ACC} {PRM}
                             {GET}                  {PHS} {CAT} {TMP}
                             {REM}                  {OBJ} {SRH}
                                                    {SRC} {USR}
                                                    {PRC}
                                                    {DMP}

where: ADD, GET, REM denote the MF values ADDLIB, GET, REMOVE;
       R4LIB01 is the library name, SL1 the sublibrary name;
       the LIBTYPE parameter values LBR, PHASE, OBJ, SOURCE, PROC,
       DUMP are given by LBR, PHS, OBJ, SRC, PRC, DMP, respectively;
       ACC, CAT, SRH, USR denote the LIBUSE values ACCESS, CATALOG,
       SEARCH, and USER, respectively;
       the CHAIN parameter values PERM and TEMP are indicated by
       PRM and TMP.
```

Connect to library (INLMLCON):

```
TIK=21 LRPL=09410C FUNC=LCON {OLD} R4LIB01
                             {NEW}

where: OLD, NEW give the value of the CONNECT parameter;
       R4LIB01 is the library name.
```

Connect to sublibrary (INLMSCON):

```
TIK=21 LRPL=09410C FUNC=SCON {OLD} R4LIB01 SL1
                             {NEW}

where: OLD, NEW give the value of the CONNECT parameter;
       R4LIB01 is the library name, SL1 the sublibrary name.
```

Connect to member (INLMMCON):

```
TIK=21 LRPL=09410C FUNC=MCON {OLD} MEMBER1  OBJ
                             {NEW}
                             {UPD}

where: OLD, NEW, UPD (for UPDATE) give the value of the
       CONNECT parameter;
       MEMBER1 is the member name, OBJ is the type name.
```

Disconnect from member (INLMMDIS):

```
TIK=21 LRPL=09410C FUNC=MDIS
```

Disconnect from sublibrary (INLMSDIS):

```
TIK=21 LRPL=09410C FUNC=SDIS
```

Disconnect from library (INLMLDIS):

```
TIK=21 LRPL=09410C FUNC=LDIS
```

Get member record(s) (INLMGETR):

```
TIK=21 LRPL=09410C FUNC=GETR
```

Put member record(s) (INLMPUTR):

```
TIK=21 LRPL=09410C FUNC=PUTR
```

Update directory entry(-ies) (INLMSTOW):

```
TIK=21 LRPL=09410C FUNC=STOW {ADD} R4LIB01 SL1
                             {DEL}
                             {REN}
                             {CHN}
                             {EMP}
                             {PUR}

where: the MF values ADD, DELETE, RENAME, CHANGE, EMPTY, PURGE are
       given by ADD, DEL, REN, CHN, EMP, and PUR, respectively;
       R4LIB01 is the library name, SL1 the sublibrary name; for
       LEVEL=SUBLIB, the sublibrary name is missing.
```

Get next directory entry (INLMGDIR):

```
TIK=21 LRPL=09410C FUNC=GDIR {MBR} R4LIB01 SL1
                             {SLB}
                             {LIB}

where: the LEVEL values are given by MBR (MEMBER), SLB (SUBLIB), and
       LIB (LIB);
       R4LIB01 is the library name, SL1 the sublibrary name; for
       LEVEL=SUBLIB|LIB the sublibrary name is missing.
```

Find member (INLMFIND):

```
TIK=21 LRPL=09410C FUNC=FIND {NOG} MEMBER1  OBJ
                             {GEN}

where: the REQUEST values are given by NOG (NOGENERIC) and
       GEN (GENERIC);
       MEMBER1 is the member name, OBJ the type name.
```

Build list of directory entries (INLMBLDL):

```
TIK=21 LRPL=09410C FUNC=BLDL {NOG}
                             {GEN}

where: the REQUEST values are given by NOG (NOGENERIC) and
       GEN (GENERIC).
```

Problem Determination Hints

Problems Detected in the SVA Phase

When a problem is detected in the SVA phase $IJBLBR, it is usually possible to follow the save areas chain established by the PL/S calling conventions.

Reg 13 Points to beginning of save area

Save Area Is allocated every time a level 2 or level 1 module is called. Standard calling conventions are used. The allocated area is not cleared before usage, so that no assumption can be made on initial values, e.g. for chain pointers and saved registers values.

Offset 0 Not used

Offset 4 Backward pointer to previous save area.

Offset 8 Forward pointer to next save area.

Offset 12 Caller Reg 14, contains the return to caller address.

Offset 16 Caller Reg 15, contains the entry point

address of module currently being executed.

Offset 20 Caller Reg 0

Offset 24 Caller Reg 1. On entry to a module, register 1 points to the parameter list pointer. For level 2 modules, the only parameter passed is the INLCLRPL pointer. For level 1 modules, the INLCLRPL pointer is part of the parameter list passed.

The INLCLRPL pointer is at offset 0 in the parameter list passed to modules INLPBBUF and INLPBSPA, at offset 4 for modules INLPLBGP, INLPGBUF, INLPFBUF, INLPGSPA.

It is important to be able to locate the current INLCLRPL, because the complete control block structure can be addressed out of anchor pointers in the LRPL (see data area INLCLRPL).

Most problems in level 1 modules are caused by improperly setup control blocks. It is advisable to start problem determination by verifying the existence and correctness of the primary control blocks and data area such as MACB, SACB, LACB, LAMB, LIBINFO, LDTE, EXTE, DDTE etc., as well as buffer pools for shared and/or private buffers.

Not all error conditions are checked during execution and signalled by use of messages L150, L151, L152 and additional feedback codes. In case of invalid I/O requests, command rejects etc. one should verify the correctness of both BUCB and BHDR(s) involved. CCB and CCW lists are part of the buffer pool BUCB.

Problems Detected in a Library

When you want to analyze a problem detected in the library, for example, one of the messages L150 - L158 occurred, it is helpful to have the TEST command output listings available (TEST L=lib,AREA=FULL and TEST L=lib,AREA=SPACE and TEST S=lib.sublib). In order to localize the program causing the error, the TEST command can be interspersed among job statements starting from a library which shows zero defects with the TEST command.
A TEST trace (I/O, space, etc) may be helpful to support this task.

Index

Special Characters

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

File Number: S370/S390-37
Program Number: 5686-066

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC33-6330-01