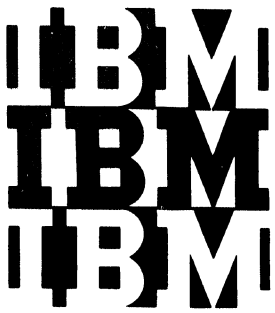


File No. S360-30
Form Y28-6606-1

Program Logic

IBM System/360 Operating System

Catalog Management

Program Number 360S-DM-508

This manual provides detailed information on catalog management routines. These routines record identification of volumes used by data sets by maintaining information in logical records called indexes. The functions and structures of the routines are described, as are their relationships to other portions of IBM System/360 Operating System. This manual also describes the structure of catalog data sets that contain the indexes processed by catalog management routines. It is intended for use by persons involved in program maintenance, and system programmers who are altering the program design. Program logic information is not necessary for the use and operation of the program; therefore, distribution of this publication is limited to those with the aforementioned requirements.

PREFACE

This publication provides customer engineers and other technical personnel with information describing the internal organization and logic of the catalog management routines. Publications that contain external information about the catalog and its use are:

IBM System/360 Operating System Supervisor and Data Management Services, Form C28-6646

IBM System/360 Operating System System Programmer's Guide, Form C28-6550

Some publications describing other aspects of the Operating System are referred to in the text. These are:

IBM System/360 Operating System Direct Access Device Space Management, Form Y28-6607

IBM System/360 Operating System Sequential Access Methods, Form Y28-6604

This manual is divided into eight major sections with three appendixes.

The Introduction describes the catalog management routines and the catalog as they relate to the rest of the Operating System.

The Catalog Data Set section describes the structure and organization of the catalog data set. An understanding of this data set is a prerequisite for an understanding of the routines used to access and modify it.

The Method of Operation section describes the logical functions of the catalog management routines.

The Program Organization section describes each module of the routines in detail, with particular emphasis on the differences between the actual code involved and the logical functions of the routines.

The Directory is a chart that enables the reader to find a section of code, a flowchart, or a text reference, given any one of the three.

The Data Area Layouts section describes in detail each of the catalog entries and also the user's parameter list.

The Diagnostic Aids section contains charts of register usage at various stages in catalog processing and of the factors involved in determining which module gets control when.

The three appendixes contain detailed flowcharts, a diagram of the device type field found in data set pointer entries and CVOL pointer entries, and a description of a CVOL pointer entry which is no longer created by catalog but which may still exist in some installations.

Second Edition (July, 1969)

This publication corresponds to Release 18. It is a major revision of, and obsoletes, Y28-6606-0 and Technical Newsletters Y26-8013 and Y26-8020.

This revision is a complete rewrite of the original publication. The organization has been completely revised, the text has been rewritten, and most of the figures have been changed. This edition also reflects the addition of a new module, IGG0CLC6, to the catalog management routines. This module performs some of the functions previously performed by IGG0CLC2.

Specifications contained herein are subject to change from time to time. Any such change will be reported in subsequent revisions or technical newsletters.

This publication was prepared for production using an IBM computer to update the text and to control the page and line format. Page impressions for photo-offset printing were obtained from an IBM 1403 Printer using a special print chain.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form is provided at the back of this publication for reader's comments. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publications, Department D78, San Jose, California, 95114.

CONTENTS

FIGURES

This page intentionally left blank.

Catalog management is the facility of the Operating System for locating data sets when the user specifies only the data set names. The catalog, itself a data set (DSNAME=SYSCTLG), contains data set names correlated with volume and device type information. The catalog management routines supervise the organization of the catalog, insert, remove, and locate entries in the catalog, and format new catalogs and partitioned data set directories.

ORGANIZATION BY LEVEL OF QUALIFICATION

Operating System data set names may be either simple or qualified. A simple name is a collection of up to eight EBCDIC characters. A qualified name is a collection of simple names separated by periods (.) with a total length of up to 44 bytes.

Catalog management uses the periods in a qualified name as delimiters and uses the simple names (called qualifiers) as index names. The catalog is divided into indexes, each of which represents one level of qualification of a qualified name. Since the catalog management routines can build or update only one index at a time, all levels of a data set name except the lowest one must exist before the data set can be cataloged or before a new index can be built. If the user wishes to catalog a data set called A.B.C, for example, he would have to create index A first, then index A.B, and then he would have to catalog data set A.B.C.

The highest level index, called the volume index, is built automatically the first time a new catalog is used by the catalog management routines.

GENERATION DATA GROUP STRUCUTURE

The same structure is used to maintain generation data groups. A generation data set may be referred to by its absolute name (e.g., A.B.C.G0006V00) for any catalog functions, or by a relative generation number (e.g., A.B.C(-2)) for the locate function. The catalog management routines keep only the specified number of entries in the generation index (index 'C' in this case), deleting older ones and adding new ones when necessary, and emptying the index and deleting the data sets themselves if the user specified the EMPTY or DELETE options when he created the generation index.

For a description of the use of generation data groups, see IBM System/360 Operating System Supervisor and Data Management Services, C28-6646.

CONTROL VOLUMES

Any direct access volume may contain a catalog; any such volume is called a control volume (CVOL). The system residence volume always contains a catalog.

An item in the catalog of a CVOL other than the system residence volume can be made available to the system if the CVOL is "connected" to the system residence volume. To connect a CVOL to the system residence volume, the catalog management routines insert a control volume pointer entry into the volume index of the catalog on the system residence volume. This entry contains, in its name field, the name of a high level index which already exists on the CVOL to be connected. (See Figure 1.)

Any search of the catalog always starts on the system residence volume, but if the catalog management routines encounter a control volume pointer entry attached to the highest level of the name in the volume index, they continue the search for the fully-qualified name on the specified CVOL.

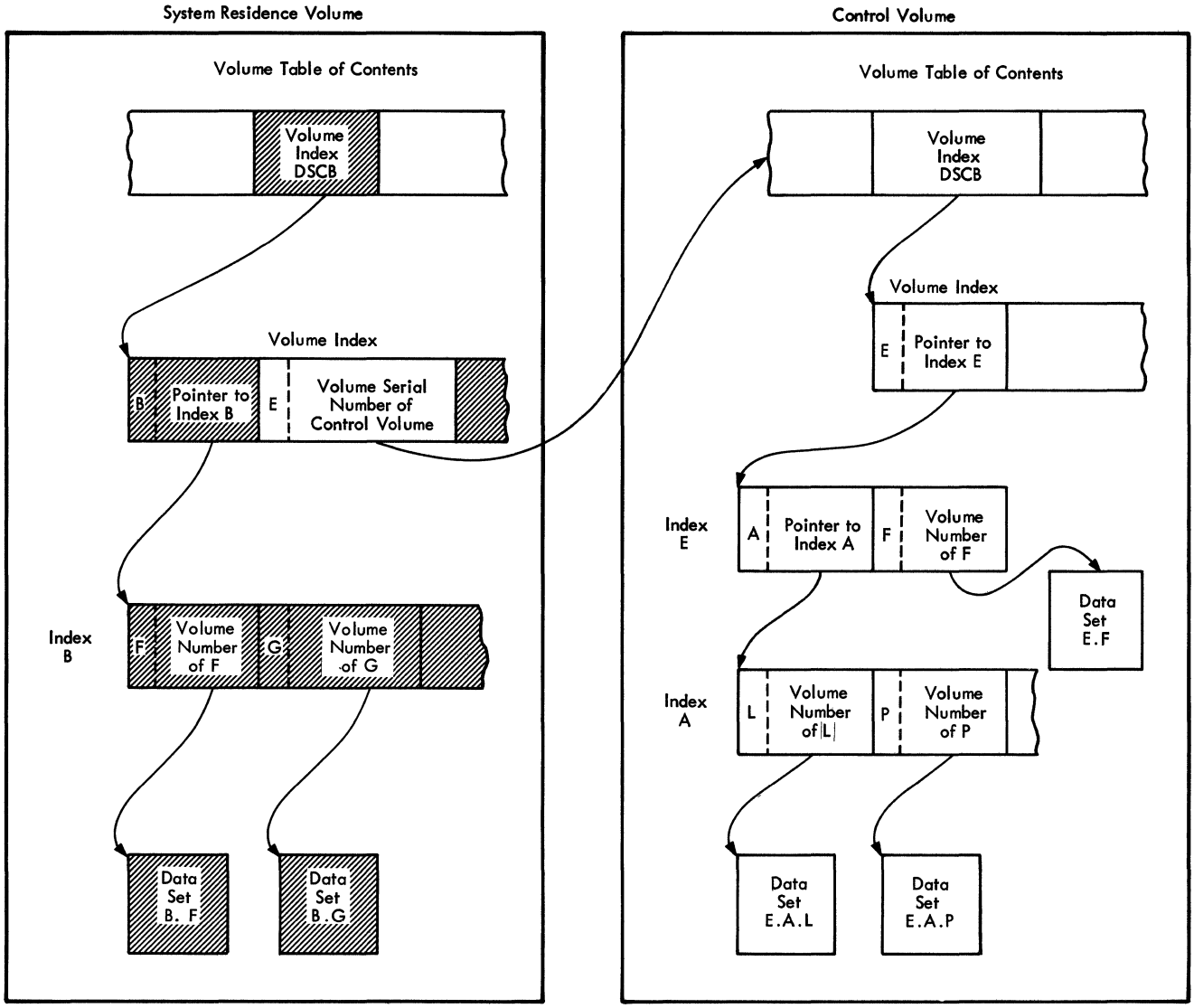


Figure 1. A Control Volume Connected to the System Residence Volume

CALLING THE CATALOG MANAGEMENT ROUTINES

The catalog management routines are accessed through three assembler language macro instructions: LOCATE, INDEX, and CATALOG. The macro instructions generate a reference to a parameter list, which the user must build, and an SVC 26 instruction. The user's parameter list contains a group of flags that indicate what function he is asking the catalog management routines to perform. Figure 2 summarizes these functions, and the section "Data Area Layouts" contains a detailed description of the user's parameter list.

The catalog management macro instructions are most commonly used by the utility IEHPROGM and the job scheduler, although they may be employed by any user of assembler language. IEHPROGM creates and deletes indexes, aliases, and generation indexes, and catalogs and uncatalogs data sets according to specifications supplied by the user of IEHPROGM. The job scheduler calls the catalog management routines when it must locate a data set, given only its name, or when the DISP parameter on a DD card is CATLG or UNCATLG.

FUNCTION		ABBREVIATION*
LOCATE	a data set by name	NAME
	a block in the catalog by TTR	BLOCK
BUILD	a normal index	BLDX
	a generation index	BLDG
	an alias to a high-level index	BLDA
DELETE	an index	DLTX
	an alias	DLTA
CONNECT	two control volumes	LINKX
DISCONNECT	two control volumes	DRPX
CATALOG	a data set	CATALOG
UNCATALOG	a data set	UNCAT
RECATALOG	a data set (change the volume serial number associated with an already cataloged data set)	RECAT

*The abbreviations here are used in the comments of the source code to indicate what operation the user requested

Figure 2. Functions of the Catalog Management Routines

CATALOG DATA SET

Physically, a catalog is arranged in blocks with keys. Logically, it is arranged in index levels. This section will describe the catalog's physical organization briefly and its logical organization in detail.

PHYSICAL BLOCKS

The physical organization of the catalog is identical with that of a partitioned data set directory.

A catalog data set is formatted into 256-byte blocks with 8-byte keys. Each block contains a 2-byte count field, which contains a number indicating how many bytes are used in this block (including this count field).

The keys of catalog blocks are always

X'FFFFFFFFFFFFFFFF', or
X'0000000000000000'.

A high key indicates that the block contains information, and a zero key indicates that the block is available for new entries. The keys are present because the catalog routines use the BLDL routine (IECPBLDL) to read the catalog. The BLDL routine expects to find 256-byte records with 8-byte keys. It ignores blocks with keys of zero.

See Figure 3 for an illustration of a typical block in the catalog.

INDEX LEVELS

The catalog is organized into a series of indexes or levels. The highest level, called the volume index, is initialized by the catalog management routines when the catalog data set is first opened.

Entries in each index are in standard EBCDIC collating sequence by their name fields.

The volume index is all that is required to catalog simple names. It also is the only index that may contain control volume pointer entries (pointers to another catalog) or alias entries. Lower level indexes are required to catalog qualified names, one index for each level of qualification except the last.

To illustrate the organization of indexes, consider the simple data set name, 'DSET' (Figure 4). If this were cataloged, only one entry would be made in the catalog: a data set pointer entry in the volume index. However, a two-level name,

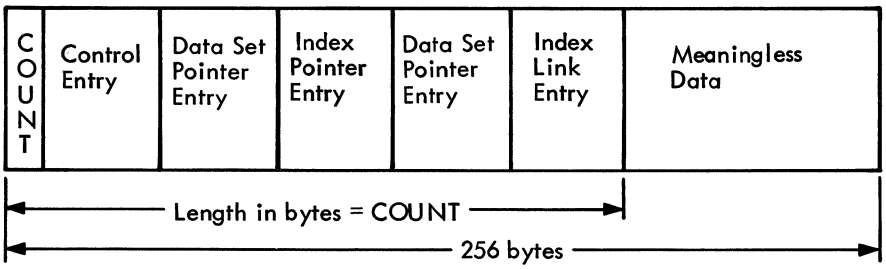


Figure 3. Typical Block in the Catalog

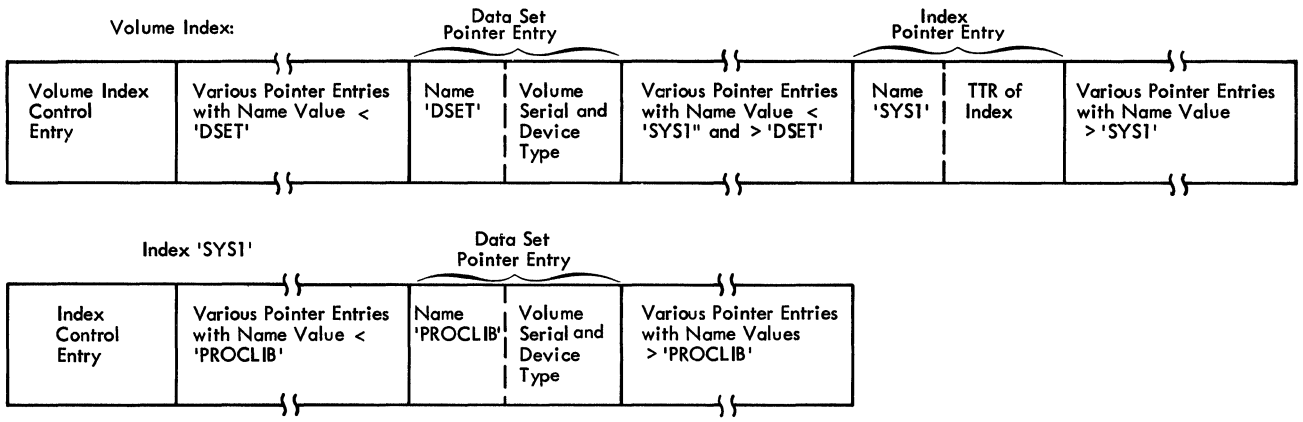


Figure 4. Logical Organization of the Catalog: Normal Index Structure

such as SYS1.PROCLIB requires another index. To catalog this name, two entries would have to be made: an index pointer entry with name 'SYS1' and a data set pointer entry with name 'PROCLIB'.

The periods (.) in a data set name act as level delimiters. The characters to the left of the first period are assumed to indicate a name in the volume index, the next level is assumed to be the name of an entry in the index indicated by the pointer in the volume index, and so on, until the last level is a name in the lowest level index and is associated with a data set pointer entry or volume control block pointer entry.

A data set pointer entry and a volume control block both contain volume serial numbers and device type information for the catalog data set. A data set pointer entry can contain only five volume serial numbers, while a chain of volume control blocks can describe any number of volumes.

A generation data group index contains data set pointer entries and volume control block pointer entries. Figure 5 shows how a catalog containing generation data group indexes and volume control blocks might look. This sample catalog lists generation data sets named "WEEKLY.INVNTRY.GnnnnVxx" to illustrate generation indexes, and a data set named "LOTSA.VOLUMES" to illustrate volume control blocks.

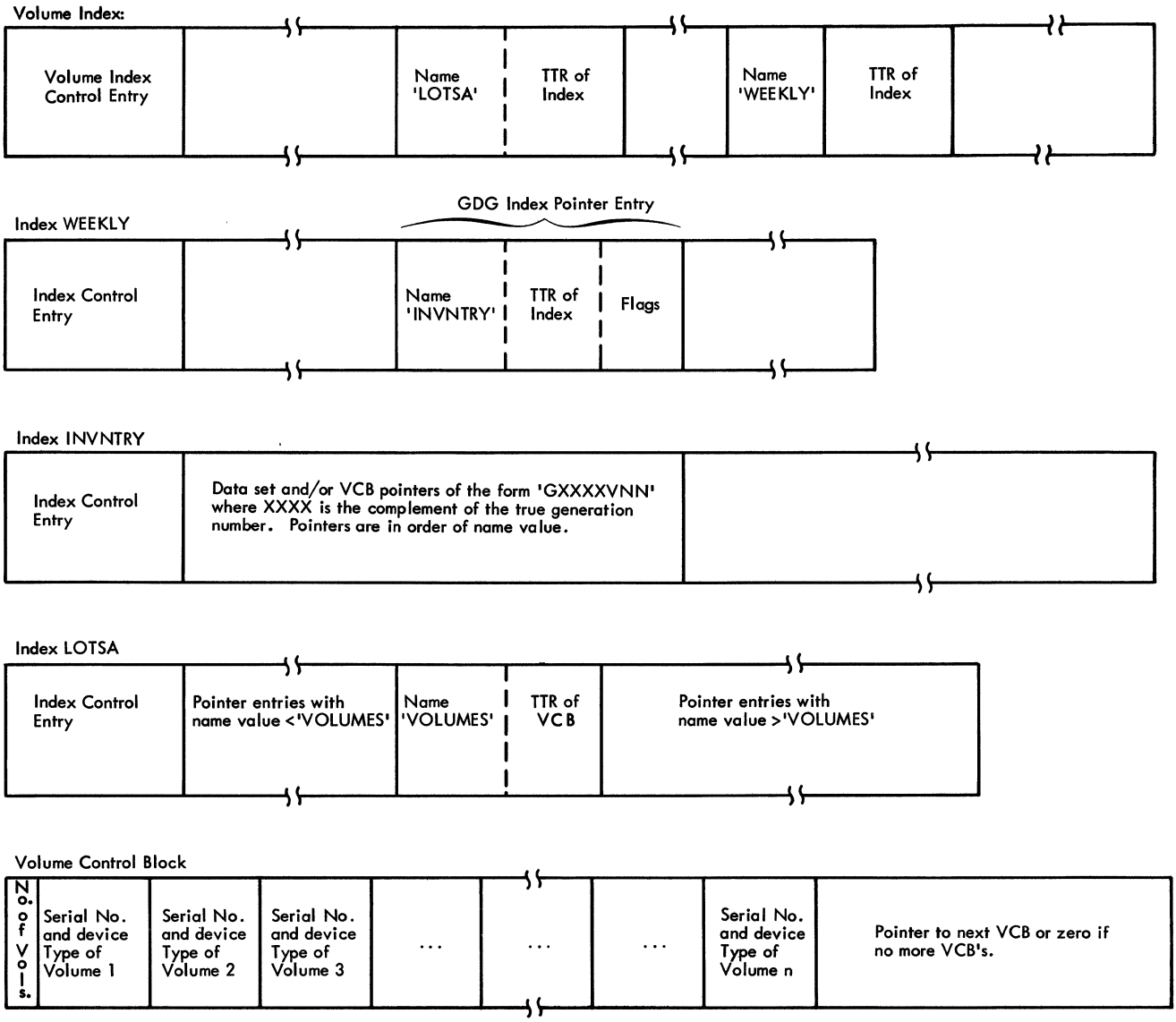


Figure 5. Logical Organization of the Catalog: Generation Indexes and Volume Control Blocks

Indexes may span blocks, but one block may not contain more than one index, or parts of more than one index. The last entry in each index block is called an Index Link Entry. (See Appendix B for specific fields.) If the block is the last one in an index, the pointer field of the link entry contains zeros. If the index is continued in another block, the pointer field of the link entry contains the TTR of the next block in the index. These link entries are present even when the several blocks of an index are contiguous (See Figure 6).

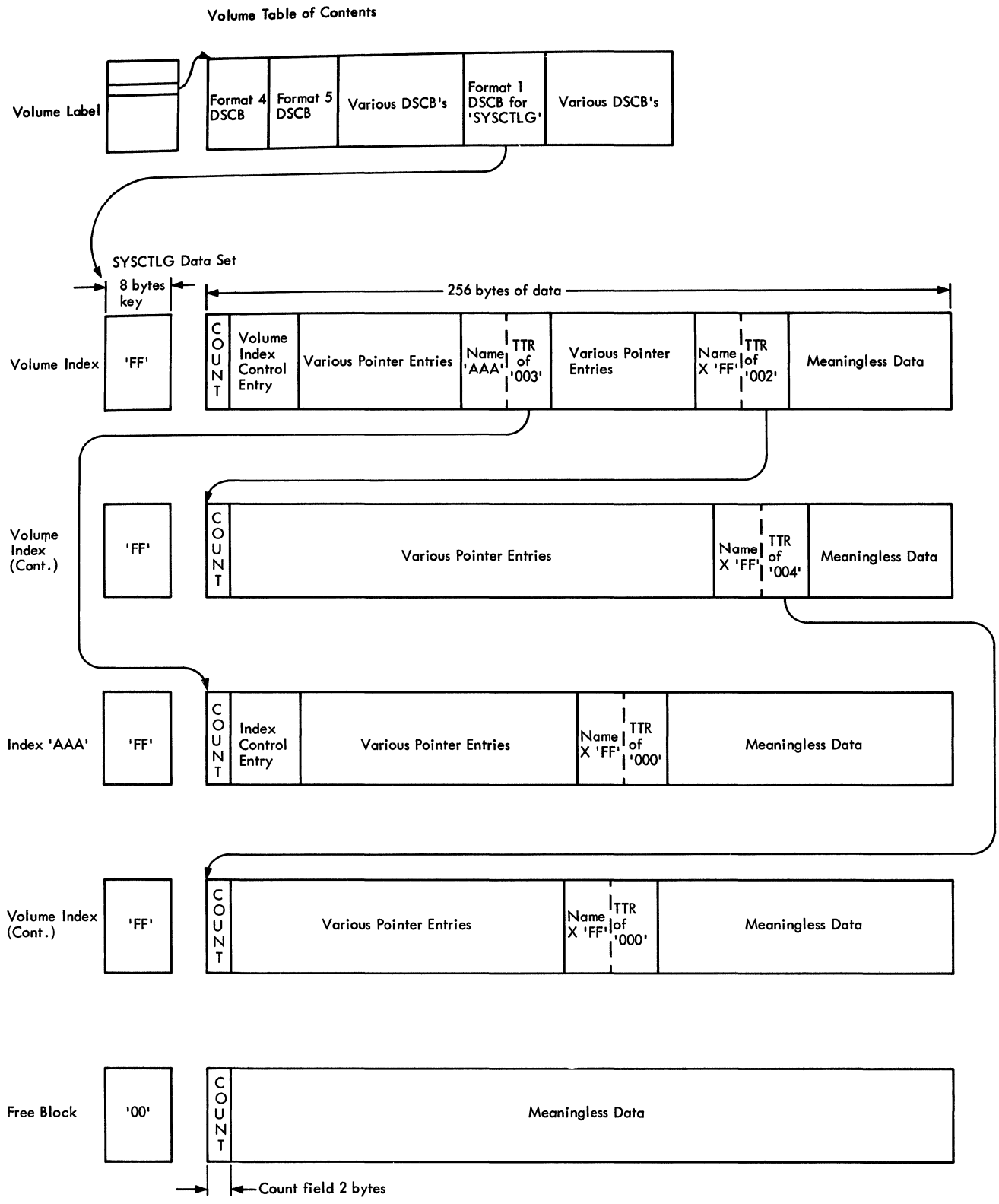


Figure 6. Physical Organization of the Catalog

INDEX ENTRY TYPES

An index always contains one control entry and any number of pointer entries. The control entry is always the first entry in the index (see Figure 6), and its position here is assured by giving it a name field of value X'1'. There are two types of control entries: volume index control entries and normal index control entries. The general information about these entries is given in Figure 7, while specific information about fields and their values is given in the section "Data Area Layouts."

There are several types of pointer entries. A summary of each type and the information it contains is given in Figure 7, while specific information about exact placement of fields, etc., is given in the section "Data Area Layouts."

ENTRY TYPE	CONTENTS
Alias Entry	Contains the name of the alias, a pointer to the next lower level index, and the true name.
CVOL Pointer Entry	Contains the name of a high level index and a pointer to the control volume on which this index may be found.
Data Set Pointer Entry	Contains the lowest level of the data set name and up to five entries specifying volume serial numbers and device codes for the volumes of the data set.
Index Control Entry	Contains the address of the last block in this index, the address of the first block (the address of the block which contains this entry), a count of the number of unused bytes in the last block of this index, and a count of the number of aliases to this index.
Generation Index Pointer Entry	Contains the name of the generation index, the number of entries to be maintained in the index, the number of entries currently in the index, codes for "delete" and "empty" options, and a pointer to the index.
Index Link Entry	Contains a name field of X'FFFFFFFFFFFFFFFF', and a zero to indicate the end of this index, or a pointer to the next block in this index.
Index Pointer Entry	Contains an index name and a pointer to the named index.
Volume Control Block	Contains an indication of the number of volumes named in the block and a list of the volume serials, device type codes, and data set sequence numbers of these volumes, plus a pointer to the next volume control block, or a zero to indicate end of chain.
Volume Control Block Pointer Entry	Contains the lowest level of the data set name and a pointer to the volume control block which describes the volumes of this data set.
Volume Index Control Entry	Contains the address of the last block in the volume index, the address of the last block in the SYSCTLG data set, and the address of the first available block in the SYSCTLG data set. It also contains a count of the number of unused bytes in the last block of the volume index.

Figure 7. Index Entries

METHOD OF OPERATION

This section describes the operation of each logical function of the catalog management routines. Since many of the functions are quite similar to each other, several of these functions have sometimes been combined into one section. The sequence of events described in this section is the actual sequence of events performed by the routines. However the division of the routines into modules does not necessarily correspond to the division of functions used in this section.

HOUSEKEEPING FUNCTIONS

Before actually beginning to search or update the catalog, the catalog management routines must perform some initialization. This initialization does two things:

- It protects the integrity of the catalog.
- It opens the catalog data set.

MAINTAINING CATALOG INTEGRITY

Since the catalog management routines were designed to operate in multiprogramming or multiprocessing environments, they must perform certain functions to ensure the integrity of the catalog with many jobs and CPUs vying for the use of the SYSCTLG data set. The first thing the program must do, therefore, is to protect the catalog from being modified by another user before this particular modification or search is completed.

To do this, the catalog management routines issue the RESERVE and ENQ macro instructions immediately after receiving control. The RESERVE macro reserves the device containing the control volume that the present use of the catalog management routines is searching or modifying. This is necessary in a multiprocessing environment where another CPU might try to access or modify the catalog before the search or update was complete. The ENQ macro instruction informs the Operating System that this use of the routines must complete before another can begin. This prevents other programs under control of the same CPU from accessing the catalog while it is being modified and from attempting to modify while it is being modified.

Since these routines are reenterable, they cannot store within themselves. They issue a GETMAIN macro instruction for some storage area within the user's region. This area is freed when the catalog routines terminate either normally or abnormally. If storage is not available, the calling task is abnormally terminated.

OPENING THE CATALOG DATA SET

To ready the catalog data set for reading and writing, the catalog management routines do not use the data management open routine (SVC 19). Instead they have a special open function called through an SVC 28. This routine builds a data extent block and a data control block so that the catalog routines can use the BLDL and EXCP routines. For a more detailed discussion of the open routine, see the section "The CVOL Routines."

The catalog open routine is called before each search of a catalog. If a search encounters a control volume (CVOL) pointer entry, the old CVOL is closed and the new one is opened.

LOCATE FUNCTION

Regardless of the particular object of one use of the catalog routines - whether the user wishes to modify the catalog or just locate a data set - the program always first tries to locate as much of the user-supplied name as possible.

The locate routine uses the resident BLDL routine (IECPBLDL) to search the catalog for the user-supplied name. This search always begins with the volume index. BLDL returns the entry with the desired name field, locate examines it, and calls BLDL again to find a lower level index or returns to the caller (function requested is locate) or passes control to another phase (function is anything but locate).

The locate portion of the program then passes an error code to other portions to indicate how much of the name was found.

BLDX, LINKX, AND BLDG FUNCTIONS

These functions are quite similar to each other. First, locate finds as much of the user-supplied name as possible and notes how much of the name it found and what kind of entry it found at the lowest level. If anything in the locate process is inconsistent with the function requested, the index/catalog portion of the program frees all its core, dequeues, and passes a nonzero return code to the caller.

For example, assume that a user wished to catalog data set 'A.B.C'. Locate would first search the catalog to find the data set pointer entry, and would pass a zero error code to index/catalog if it found the entry. Index/catalog would immediately return with an error code to the caller because it cannot catalog a name that has already been cataloged. If locate indicated that it had found A.B, but not C, and that it had found an index pointer entry at B, then index/catalog would update the index by inserting the new pointer entry.

If the request is to build an index (BLDX), index/catalog first finds an available block in the catalog and initializes it as an empty index. To do this, it creates an index control entry and an index link entry with a pointer field of zero, and writes a high key (X'FFFFFFFFFFFFFFFF') for the new index block.

A new index pointer entry must then be inserted in the next higher level index. To do this, index/catalog searches the index until it finds an entry which has a name field with value higher than that of the new index pointer entry and which is not an index link entry with a nonzero pointer field. When it finds such an entry, it inserts the pointer to the new index and rewrites the rest of the index.

The index always must be completely rewritten because the insertion of the new entry may cause the chain of index blocks to break differently.

LINKX is just like BLDX, except that a CVOL pointer is created instead of an index pointer.

BLDG is similar, except that the index pointer entry contains the appropriate generation counts and flags.

CATALOG AND RECAT FUNCTIONS

To catalog a data set, the program does very much the same thing as when the function is BLDX or BLDG except that:

- No new index is created. The new data set pointer entry is simply inserted at the appropriate place in the existing index.
- If the data set to be cataloged resides on more than five volumes, one or more volume control blocks (VCBs) must be created. The creation of this block resembles the creation of a new index very closely, except that instead of a new index, a new VCB is created.

To catalog a data set that is part of a generation data group (GDG), the routines must first find the absolute generation number if only the relative generation number was given. First, the latest entry in the index is found. This entry will be the first one in the index even though it has the highest generation number, because the catalog stores generation numbers in complement form. Then the given relative generation number is added to or subtracted from the found generation number to give the desired true generation number.

The given name is now compared with the present entries in the catalog to check for duplications, and the new name is inserted as any other Data Set Pointer Entry or VCB Pointer Entry. The generation count is updated, and, if necessary, the oldest entry in the index is removed. The flags of the generation index pointer entry are checked to see if the index must be emptied or if any data sets must be deleted. If any data sets have to be deleted, the routines transfer control to the Delete routine of DADSM by issuing an SVC 29. (For a discussion of the Delete routine see IBM System/360 Operating System Direct Access Device Space Management, Y28-6607.)

For RECAT, the routines uncatalog the old data set, then catalog the new, as above.

BLDA FUNCTION

The BLDA function is basically similar to the BLDX function, except that BLDA only creates a pointer entry; it builds no new index.

Locate finds the name for which an alias is being built, and checks to be sure it is a high-level name. If it is, the routines read tne block containing the high-level

name, add one to the entry alias count, and rewrite the block.

The routines then create an alias entry and insert it in alphameric order into the volume index. The volume index is reorganized as for BLDG and BLDX.

DLTX, DLTA, DRPX, UNCAT FUNCTIONS

The sequence of operations to delete an index or an alias or to uncatalog a data set or disconnect control volumes is basically similar to the other functions involving reorganization of the catalog:

1. The catalog is searched for the user-supplied name. In this case the entire name must be found.

2. If a pointer entry is deleted, the block it points to must also be deleted. In the case of UNCAT, a VCB may have to be freed. With DLTX, an index block always has to be freed. With DLTA and DRPX, no blocks should have to be freed unless deleting the pointer makes the volume index enough shorter that it takes up fewer blocks than before.

3. To delete a block, the program writes a zero key for that block. The data inside the block remains unchanged. The program recognizes any block with a zero key as a free block.

4. The index from which the entry was deleted is reorganized just as when a new entry is added.

THE CVOL ROUTINES

The CVOL routines open or extend the SYSCTLG data set, format new catalogs or extensions of old catalogs, and format partitioned data set (PDS) directories.

The routines receive from their callers the address of the Unit Control Block (UCB) of the device containing the data set to be opened or extended, and a parameter indicating whether the request is to open a catalog, to extend a catalog, or to format a PDS directory.

OPEN ROUTINE

If the request is to open a catalog, the routines build a data extent block (DEB) and a data control block (DCB) for the SYSCTLG data set using information from the unit control block (UCB) and volume table of contents (VTOC) of the volume being opened. If no space has been allocated for the SYSCTLG data set, an error code is returned to the user.

The Format 1 data set control block (DSCB) for the catalog data set has a format switch which indicates whether this SYSCTLG data set has been previously formatted. If the switch shows that the data set has not been formatted, the open routine passes control to the formatting routine. Otherwise, it returns to the caller.

EXTEND ROUTINE

To extend the data set, the CVOL routine transfers control to the Extend routine of Direct Access Device Space Management. This routine extends the data set by updating the VTOC (provided a secondary allocation quantity was specified when space for SYSCTLG was initially allocated), and transfers control to the formatting routine. The formatting routine formats the extension, but does not initialize a volume index, since there is already one present. It does, however, update the Volume Index Control Entry to show the extra space.

FORMATTING ROUTINE

The formatting routine formats the allocated space into 256-byte records with 8-byte keys, and initializes the volume index with a volume index control entry and an index link entry with a zero pointer field. The key of this block is set to X'FFFFFFFFFFFFFFFF' while the keys of all the other blocks are set to zero. It sets the format switch in the DSCB to indicate that the data set has been formatted and returns to the caller.

To format a partitioned data set (PDS) directory, only the formatting routine is used. The open routine immediately passes control to the formatting routine.

Formatting takes place in the same general way as for SYSCTLG data sets, with 256-byte records and 8-byte keys. Instead of initializing a volume index, however, the routine initializes the first block as an empty PDS directory.

The catalog management modules are designed to fit in the 1024-byte transient areas of the nucleus. They are reenterable. In general, the modules pass control from one to the other through the XCTL macro instruction, although they sometimes use SVCs. The following discussion will enlarge upon the Method of Operation section by discussing the routines module by module. Figure 8 shows the relationships among the catalog management routines and between the catalog management routines and other parts of the Operating System.

NOTE: In this discussion, the term 'write' always refers to the use of an EXCP macro instruction. 'Read' generally refers to the use of the resident routine IECPBLDL, but the modules occasionally use channel programs here, also.

IECPBLDL, the resident BLDL routine, is accessed by the catalog management routines through the Communication Vector Table (CVT). The routines find the address of IECPBLDL in the CVT, put the address of the catalog DCB in register 1 and the address of the BLDL list in register 0, and execute a BALR to the BLDL routine. For the functions of the BLDL routine, see IBM System/360 Operating System Sequential Access Methods, Y28-6604.

INITIALIZATION AND HOUSEKEEPING: MODULE IGC0002F

Entry to the catalog management routines, except the open routine, is through an SVC 26, which gives control to this module. The module issues an ENQ macro instruction on the name 'SYSCTLG' to protect against simultaneous modifications of the catalog in a multiprogramming environment and gets main storage for the open routine. It searches the unit control block (UCB) table to find the UCB of the specified control volume (CVOL) or the system residence device (if no CVOL was specified) to pass on to the open routine, and then reserves the CVOL (if it is not the system residence device) to prevent accesses by another CPU. It then calls the catalog open routine with an SVC 28. It checks the return code from open, and, if no error has occurred, it requests the appropriate amount of storage for locate or index/catalog, via GETMAIN, and transfers control to IGG0CLC1.

IGC0002F may be reentered from IGG0CLC1 if that module finds a control volume pointer entry which it must follow. The only difference this makes in the control path through IGC0002F is that IGC0002F does not issue the ENQ macro instruction if entry was from IGG0CLC1. This is because the ENQ was already issued in the first pass through IGC0002F. IGG0CLC1 passes the address of the serial number of the CVOL to be opened as a parameter to IGC0002F.

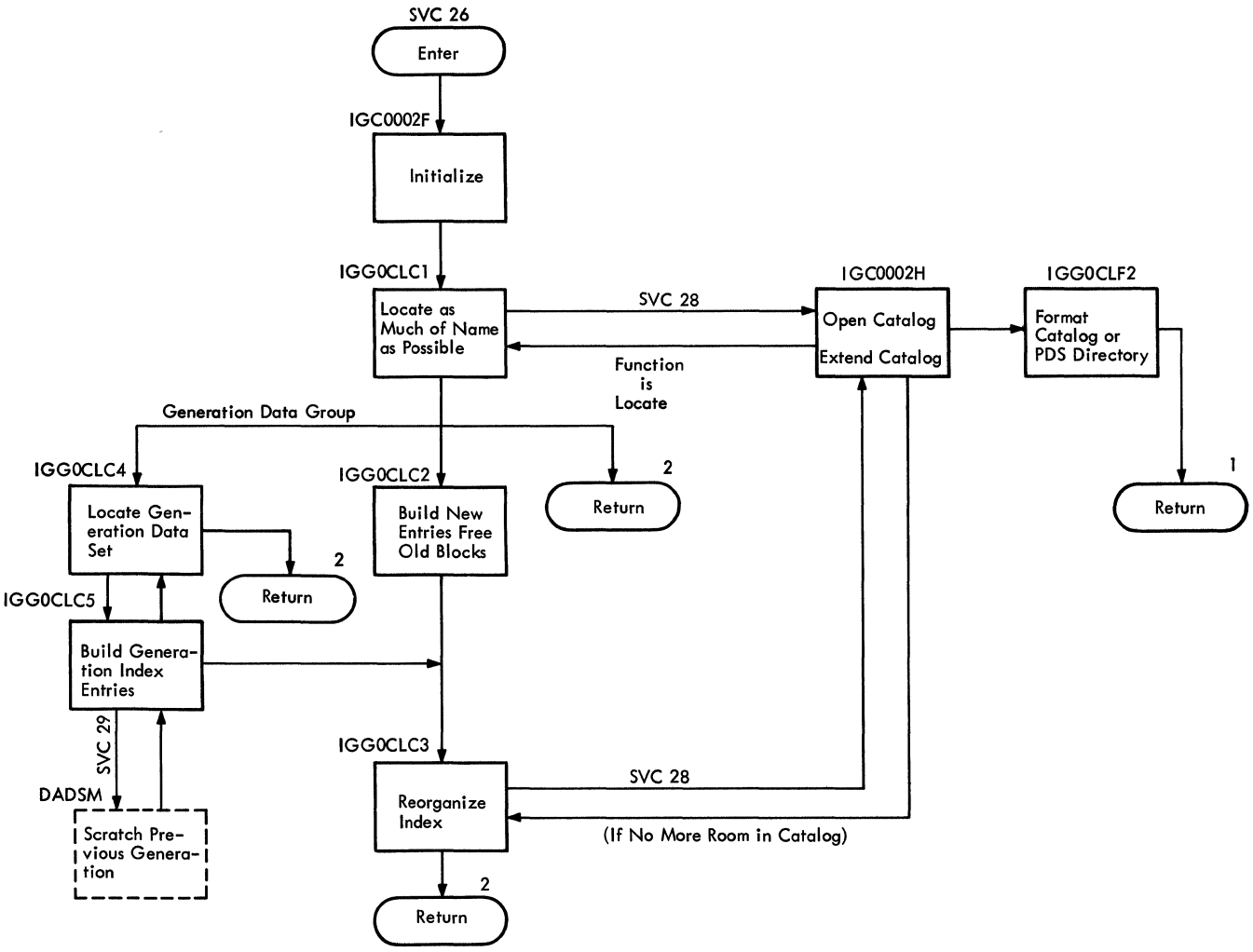
LOCATE: MODULE IGG0CLC1

This module always gets control from IGC0002F. It searches the specified catalog for the supplied name and passes control to one of two other modules, depending on the function requested and the type of entry found at the lowest level. An input parameter indicates whether the user wishes to locate a data set by name or to locate an entry in the catalog by giving the TTR of the block.

If the request is to search for a specified block, the module passes the block's address to the resident routine IECPBLDL. IECPBLDL searches the catalog and returns the correct entry to the caller. The only error possible is that the block might be outside of the SYSCTLG data set, in which case an error code is set and the module returns control to the caller.

If the request is to search for a name or to index or catalog a name, IGG0CLC1 isolates the first level of the name. It uses BLDL to search the volume index for this simple name and analyzes what type of pointer is associated with it. Several different things can happen, depending on what pointer type was found and what function was requested.

In the most typical case, the routines will find an index pointer entry and note that there are more qualifiers left in the name. In this case, the module isolates the next qualifier and searches for that name, specifying to BLDL that the search is to begin at the TTR specified in the found index pointer entry. This process is repeated until either all levels of the name are exhausted or an entry which is not an index pointer entry is found.



[1]This is a return to the issuer of SVC 28: IGG0CLC1 or IGG0CLC3.

[2]This is a return to the issuer of SVC 26: the user.

Figure 8. Catalog Module Flow

When Locate has found all of the pointers it can find, it determines what action to take on the basis of what kind of pointer was the last found, how much of the name could not be found, and what function was requested. It may transfer control to IGG0CLC2 to build new entries in the catalog, it may transfer control to IGG0CLC4 to search generation indexes, or it may return to the caller via an SVC 3 with the appropriate error code.

If control is going anywhere but back to the caller, Locate reads several relevant blocks into main storage:

- Block Containing Volume Index Control Entry - This is necessary to indicate where the first available block in the catalog is. It has to be updated if any new blocks are used or any old ones are freed.
- Block Containing Index Control Entry - This entry is the control entry for the

The catalog management modules are designed to fit in the 1024-byte transient areas of the nucleus. They are reenterable. In general, the modules pass control from one to the other through the XCTL macro instruction, although they sometimes use SVCs. The following discussion will enlarge upon the Method of Operation section by discussing the routines module by module. Figure 8 shows the relationships among the catalog management routines, as well as between the catalog management routines and other parts of the Operating System.

NOTE: In this discussion, the term 'write' always refers to the use of an EXCP macro instruction. 'Read' generally refers to the use of the resident routine IECPBLDL, but the modules occasionally use channel programs here, also.

IECPBLDL, the resident BLDL routine, is accessed by the catalog management routines through the communication vector table (CVT). The routines find the address of IECPBLDL in the CVT, put the address of the catalog DCB in register 1 and the address of the BLDL list in register 0, and execute a BALR to the BLDL routine. For the functions of the BLDL routine, see IBM System/360 Operating System Sequential Access Methods, Y28-6604.

INITIALIZATION AND HOUSEKEEPING: MODULE IGC0002F

Entry to the catalog management routines, except the open routine, is through an SVC26, which gives control to this module.

It searches the unit control block (UCB) table to find the UCB of the specified control volume (CVOL) or the system residence device (if no CVOL was specified) to pass on to the Open routine. It then calls the catalog open routine with an SVC 28. It checks the return code from open, and, if no error has occurred, it requests the appropriate amount of storage for the locate or index/catalog routines, via GETMAIN, and transfers control to IGG0CLC1.

IGC0002F enqueues on the catalog resources and reserves the specified CVOL. The ENQ macro instruction requires two names to be specified: a "qname" and an "rname." The catalog management routines use the following names:

qname	rname
SYSCTLG	SYSCTLG␣00ua

Where "ua" is the two-byte address of the UCB of the CVOL if a CVOL was specified, or two bytes of zeros if no CVOL was specified.

IGC0002F may be reentered from IGG0CLC1 if that module finds a control volume pointer entry which it must follow. The only difference this makes in the control path through IGC0002F is that IGC0002F does not issue the ENQ macro instruction if entry was from IGG0CLC1. This is because the ENQ was already issued in the first pass through IGC0002F. IGG0CLC1 passes the address of the serial number of the CVOL to be opened as a parameter to IGC0002F.

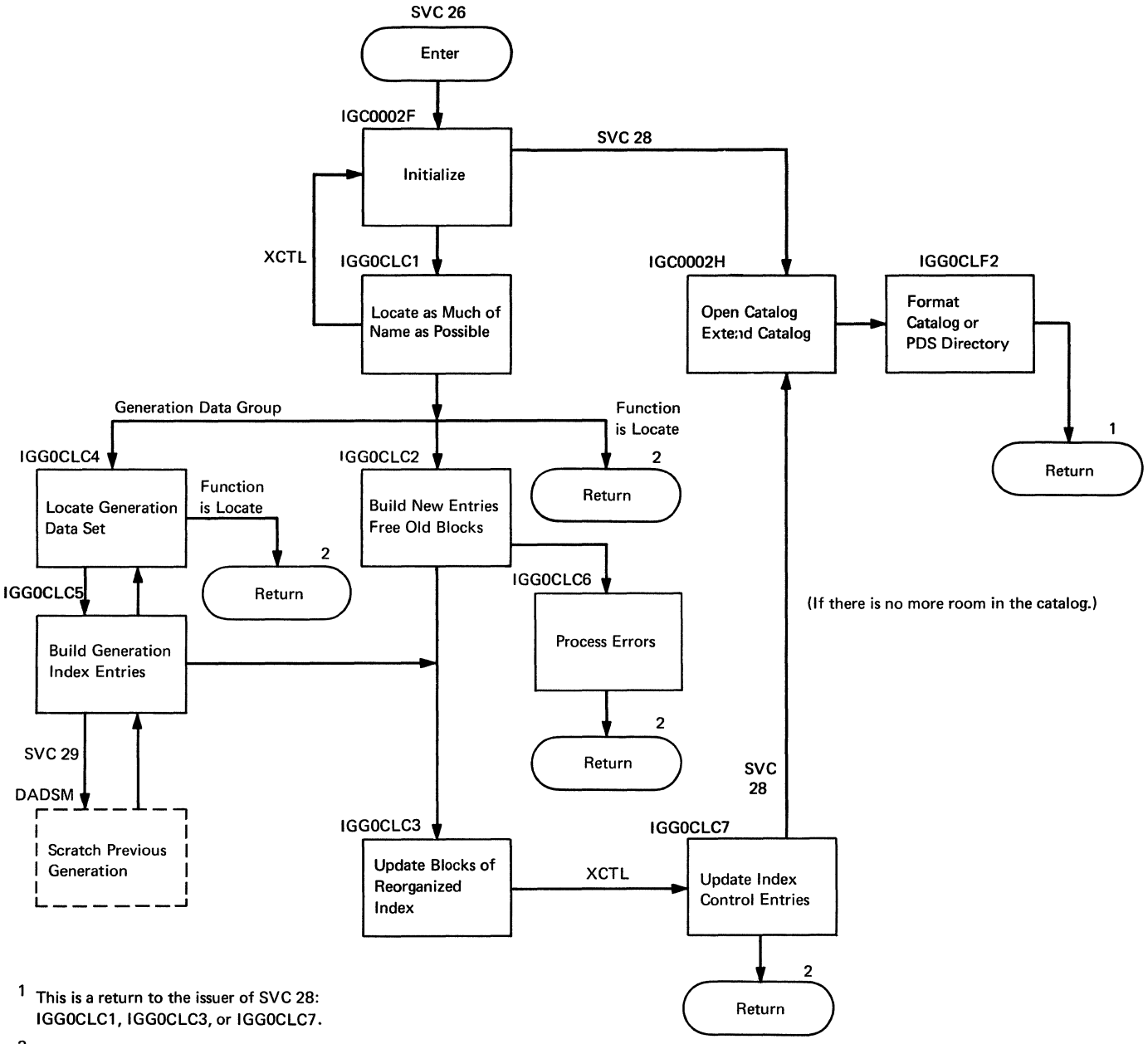
LOCATE: MODULE IGG0CLC1

This module always gets control from IGC0002F. It searches the specified catalog for the supplied name and passes control to one of two other modules, depending on the function requested and the type of entry found at the lowest level. An input parameter indicates whether the user wishes to locate a data set by name or to locate an entry in the catalog by giving the TTR of the block.

If the request is to search for a specified block, the module passes the block's address to the resident routine IECPBLDL. IECPBLDL searches the catalog and returns the correct entry to the caller. The only error possible is that the block might be outside of the SYSCTLG data set, in which case an error code is set and the module returns control to the caller.

If the request is to search for a name or to index or catalog a name, IGG0CLC1 isolates the first level of the name. It uses BLDL to search the volume index for this simple name and analyzes what type of pointer is associated with it. Several different things can happen, depending on what pointer type was found and what function was requested.

In the most typical case, the routines will find an index pointer entry and note that there are more qualifiers left in the name. In this case, the module isolates the next qualifier and searches for that



●Figure 8. Catalog Module Flow

name, specifying to BLDL that the search is to begin at the TTR specified in the found index pointer entry. This process is repeated until either all levels of the name are exhausted or an entry which is not an index pointer entry is found.

When Locate has found all of the pointers it can find, it determines what action to take on the basis of what kind of pointer was the last found, how much of the name could not be found, and what function was requested. It may transfer control to IGG0CLC2 to build new entries in the catalog, it may transfer control to IGG0CLC4 to search generation indexes, or it may return to the caller via an SVC 3 with the appropriate error code.

If control is going anywhere but back to the caller, Locate reads several relevant blocks into main storage:

- Block Containing Volume Index Control Entry - This is necessary to indicate where the first available block in the

If an entry must be removed from the index, IGG0CLC5 removes it and rewrites the index block which contained this entry. If the empty option is indicated by the flags in the generation index pointer entry, the module transfers control back to IGG0CLC4 to empty the index. If the delete option is indicated, the module calls the SCRATCH function of Direct Access Device Space Management (DADSM)* with an SVC 29 to scratch the data set. After the module deletes whatever entries it must delete, it builds any new entries necessary.

When all the counts have been updated, the necessary entries removed from the index, and the specified data sets scratched, IGG0CLC5 reads the index to be updated and transfers control to IGG0CLC3. IGG0CLC3 reorganizes the index just as if it were a normal index.

THE CVOL ROUTINES: MODULES IGC0002H AND IGG0CLF2

These modules together take care of the Open and initialization functions for the catalog management routines. IGC0002H opens or extends the catalog by building or modifying a data control block (DCB) and a data extent block (DEB) for the SYSCTLG data set and IGG0CLF2 formats new catalogs, extensions of the catalog, and new partitioned data set directories.

IGC0002H

This module is entered by an SVC 28, or by XCTL if returning from the Extend routine of DADSM*. If entry is by SVC 28, the module opens or extends the catalog, depending on input parameters. If entry is by XCTL from the DADSM Extend routine, the module finishes extending the catalog.

To open the catalog, the module searches the volume table of contents (VTOC) of the volume whose unit control block (UCB) address was specified by the caller (IGG0CLC1 or 3). If it does not find a format 1 data set control block (DSCB) with name SYSCTLG in the VTOC, it sets a return code of 4 and exits. If it does find the format 1 DSCB, it constructs a DCB and DEB from information in the DSCB and from information contained in the module itself (information common to all SYSCTLG data sets such as blocksize and record format).

There is a switch in the DSCB of a SYSCTLG data set that indicates whether the data set has been formatted or not. If this switch is off, IGC0002H transfers control to IGG0CLF2, the formatting routine, to format the data set. If the switch is on, the module releases any unused DEB or DCB space and exits.

To extend the catalog, the module gets main storage for the Extend routine of DADSM, reads the format 1 DSCB for SYSCTLG, and checks the secondary allocation quantity in the DSCB. If this quantity is zero, the catalog cannot be extended and IGC0002H returns to the caller with an error code of 4. If there is a secondary allocation quantity specified in the DSCB, the module builds a parameter list for the Extend routine and transfers control to module IGG0533A.

The Extend routine of DADSM returns control to the beginning of IGC0002H, which indicates that the data set must be formatted and where the formatting is to begin, and then passes control to the formatting routine (IGG0CLF2). It also builds a new DEB which includes the newly allocated space.

IGG0CLF2

This module formats new catalogs, extensions of existing catalogs, and new partitioned data set (PDS) directories. It does this by filling the available space with 256-byte records with 8-byte keys. If it is formatting a new SYSCTLG data set or a PDS directory it also initializes the first block.

If the request is to format a PDS directory, the module constructs a channel program to write one 256-byte block at a time. The first write operation writes an empty directory, and each subsequent write writes an 8-byte zero key and 256-byte zero record. When it has formatted all the requested blocks, it writes an end of data mark, and returns to the caller via an SVC 3.

If the request is to format a catalog, the module constructs a channel program to write keys and data, a full track at a time. The module uses information from the DSCB to determine how many blocks will fit on a track. It keeps a record of the last relative track formatted to insert it into the volume index control entry.

---

*See IBM System/360 Operating System Direct Access Device Space Management Program Logic Manual, Y28-6607.

When the module has reached the end of the extent assigned to SYSCTLG, it checks to see if it has been formatting a new catalog or an extension. If it has been formatting an extension, it returns directly to the caller. If it has been formatting a new SYSCTLG data set, it builds an empty volume index, containing a volume index control entry and an index link entry with zero TTR field, and sets the format switch in the DSCB to indicate that the data set has been formatted. Before returning to the caller, the module always frees the working storage obtained for it by IGC0002H.

This chart contains information to assist the reader in making the transition from this manual to the assembler language listings of the catalog management modules. It correlates information from three sources:

- The source code
- The executable load modules
- This manual

LOAD MODULE NAME	RESIDENCE NAME	DESCRIPTION	CSECT	FLOWCHART PAGE(S)
IGC0002F	SYS1.SVCLIB	Initialization	IGC026	31
IGG0CLC1	SYS1.SVCLIB	Locate	IGG0CLC1	32-34
IGG0CLC2	SYS1.SVCLIB	Build and free block	IGG0CLC2	35-37
IGG0CLC3	SYS1.SVCLIB	Reorganize index	IGG0CLC3	38,39
IGG0CLC4	SYS1.SVCLIB	Locate generations	IGG0CLC4	40-42
IGG0CLC5	SYS1.SVCLIB	Build generation index entries	IGG0CLC5	43,44
IGG0CLC6	SYS1.SVCLIB	Process errors	IGG0CLC6	45
IGC0002H	SYS1.SVCLIB	Open/extend catalog	IGC028	46,47
IGG0CLF2	SYS1.SVCLIB	Format catalog & PDS directory	IGG0CLF2	48

Figure 9. Directory

DATA AREA LAYOUTS

This section contains illustrations and explanations of the layouts of the various types of catalog entries and of the parameter list which the user supplies to the catalog management routines.

CATALOG ENTRIES

This section describes in detail the format of each of the possible entries in the catalog. Figures 10 and 11 represent each entry pictorially and the following text describes the contents of each field.

The Volume Index Control Entry contains information about the entire catalog and the volume index. It is always the first entry in the catalog. It is 22 bytes long and contains 8 entries.

Field 1: This is the name field. It always contains the value X'0000000000000001' to ensure that this entry is always first in the volume index.

Field 2: This field contains the TTR of the last block in the volume index.

Field 3: This field contains the number 5 to indicate that five halfwords of user data follow. It also serves to identify this entry as a volume index control entry, since this is the only entry that is twenty-two bytes long (total).

Field 4: This field contains the TTR of the last block in the SYSCTLG data set.

Field 5: This is the alias count field in a normal index, but since this is the volume index it will always contain zero.

Field 6: This field contains the TTR of the first unused block in the catalog.

Field 7: This field contains zero.

Field 8: This field contains a count of the number of unused bytes in the last block of the volume index.

An Index Control Entry is quite similar to a volume index control entry, but it only contains information about the index which it begins. It is 18 bytes long and contains six fields.

Field 1: This name field contains X'0000000000000001' to ensure that this entry is first in its index.

Field 2: As in the volume index control entry, this field contains the TTR of the last block in this index.

Field 3: This field contains the number 3 to indicate that three halfwords follow. It identifies this entry as an index control entry.

Field 4: This field contains the TTR of the first block in this index. This address is always the address of the block which contains this entry.

Field 5: This field contains a count of the number of aliases in the catalog that reference this index. This count will be nonzero only for indexes one level removed from the volume index.

Field 6: This field contains a count of the number of unused bytes in the last block of the index.

Index Link Entries and Index Pointer Entries are quite similar. An index link entry is used to chain several blocks of an index together and an index pointer entry is used to chain an index to the next lower level index. An index link entry is always the last entry in any index block. These blocks contain three fields and are 12 bytes long.

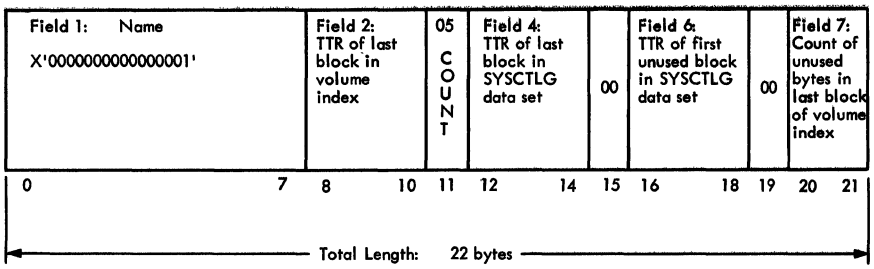
Field 1: This is the name field and contains the name of the index to which this entry points. If the entry is an index link entry, the name field contains X'FFFFFFFFFFFFFFFF'.

Field 2: This is the pointer field and contains either the TTR of the first block of the index, in the case of an index pointer entry, or the TTR of the next block of the index, in the case of an index link entry.

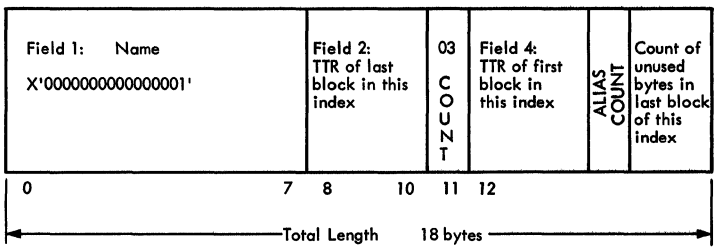
Field 3: This is the count field, and it contains zero to indicate that the entry ends here.

The Data Set Pointer Entry contains the actual information for which the catalog exists: the volume serial number, data set sequence number, and device type code of the data set which the fully qualified name represents. The entry can be from 26 to 74 bytes long, depending on how many volumes the data set occupies.

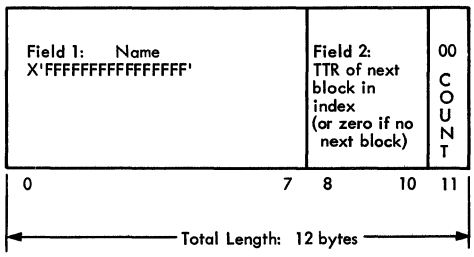
Volume Index Control Entry



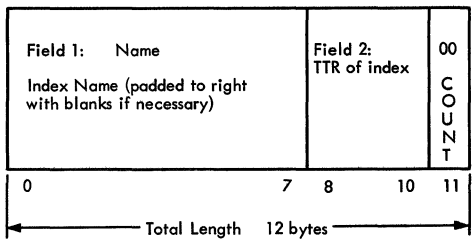
Index Control Entry



Index Link Entry



Index Pointer Entry



Data Set Pointer Entry

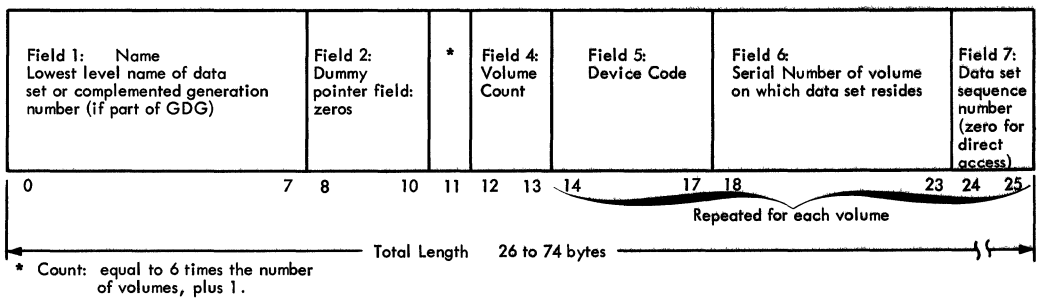
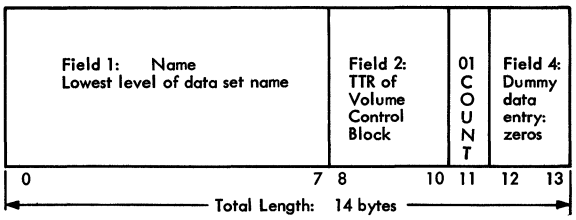
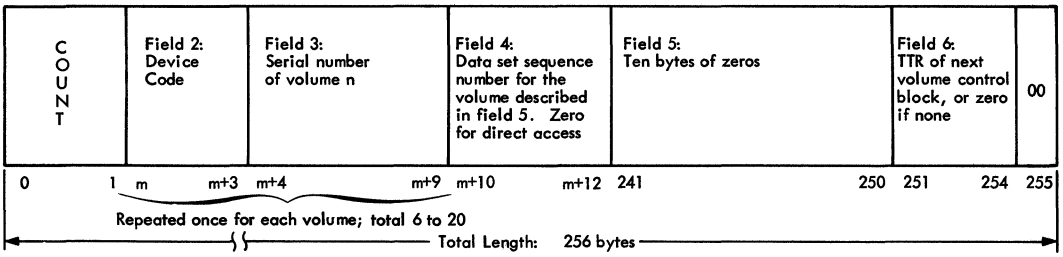


Figure 10. Catalog Entry Formats

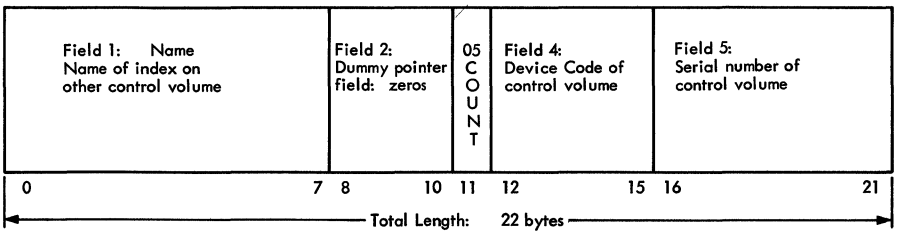
Volume Control Block Pointer Entry



Volume Control Block

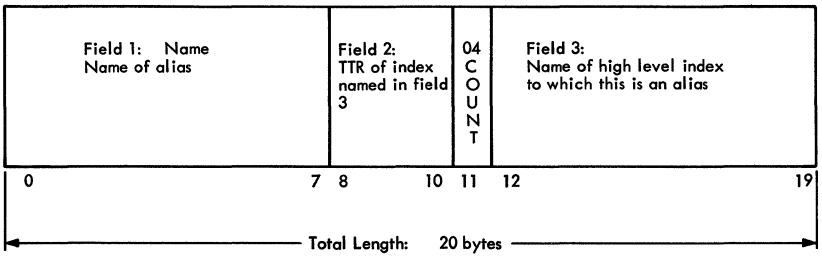


Control Volume Pointer Entry

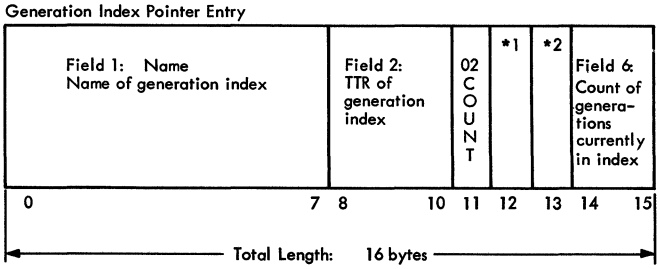


NOTE: Prior to release 17, the Control Volume Pointer Entry contained a count of 03 and did not have a Device Code field (Field 4)

Alias Entry



Generation Index Pointer Entry



*1 Field 4:
Flags:

bits	meaning
0-5	Reserved
6	Delete
7	Empty

*2 Field 5:
Count of maximum gererations to be maintained in index

Figure 11. More Catalog Entry Formats

Fields one through four occur only once while fields five through seven occur once for each volume of the data set.

Field 1: This field contains the lowest level of the data set name.

Field 2: This would normally be the address field, but since a data set pointer entry references no other entries in the catalog, it contains zeros.

Field 3: Count of user data. This field indicates how many halfwords of data follow. The number in here will be six times the number of volumes (there are six halfwords for each volume) plus one (for the volume count).

Field 4: This field contains a count of the volumes following (one to five).

Field 5: This field contains the device type code of the device on which the volume with the following serial can be mounted. (See Appendix C.)

Field 6: This field contains the volume serial number of one of the volumes of the data set.

Field 7: This field contains the sequence number of the data set on a magnetic tape volume. It is zero for any other device.

A Volume Control Block Pointer Entry is used instead of a data set pointer entry when the data set occupies more than five volumes. This entry points to a volume control block, which, in turn, describes the data set. The entry is 14 bytes long.

Field 1: This name field contains the lowest level of the data set name.

Field 2: This field contains the TTR of the first (or only) volume control block for the data set.

Field 3: The count field contains zero to indicate that this is the end of the entry.

A Volume Control Block contains the description of all the volumes of a data set which resides on more than five volumes. One volume control block can describe up to twenty volumes and volume control blocks may be chained together, so that a data set can be cataloged no matter how many volumes it requires. The volume control block is always 256 bytes long, regardless of how many volumes it describes.

Field 1: The first two bytes of a volume control block contain a count of the number of volumes described by this volume control block and any following it. For example the count fields of a series of VCBs for a data set that occupied sixty volumes would show sixty, forty, and twenty as the volume count.

This is the only kind of block in the catalog in which the first two bytes are not used as a count of the number of used bytes in the block.

Field 2: This field can contain up to twenty 12-byte volume descriptions, consisting of device type codes (See Appendix C) and volume serial numbers.

Field 3: This field contains ten bytes of zeros, followed by the TTR of the next volume control block for this data set, followed by one byte of zeros. If there are no more volume control blocks for this data set, the TTR is zero.

A Control Volume Pointer Entry is used to indicate that a particular index resides on a volume other than the system residence volume. Control volume pointer entries can exist only in the volume index. They are 22 bytes long.

Field 1: The name field contains the name of the high level index which resides in the volume described by this entry.

Field 2: The address field contains zeros, because this entry references no others in the catalog.

Field 3: The count field contains the number three to indicate that three halfwords follow.

Field 4: This field contains the device type code of the specified control volume. (See Appendix C.)

Field 5: This field contains the volume serial number of the control volume which has an entry in its volume index of the same name as this entry.

An Alias Entry is used to specify a substitute name for a high level index. Alias entries only appear in the volume index. They are 20 bytes long.

Field 1: The name field contains the alias.

Field 2: The address field contains the TTR of the first block of the index for which this entry specifies an alias.

Field 3: The count field contains the number 3 to indicate that three halfwords of data follow.

Field 4: This field contains the true name of the index for which this entry is an alias.

A Generation Index Pointer Entry points to a generation index. It is basically the same as an Index Pointer Entry, except that it includes the flag and count fields. It is 16 bytes long.

Field 1: The name field contains the lowest level name of the generation data group. That is, a generation data set named WEEKLY.INVNTRY.G0001V00 would have the name "INVNTRY" in the generation index pointer entry name field.

Field 2: The address field contains the TTR of the first block of the generation index.

Field 3: The count field contains the number 2 to indicate that two halfwords follow.

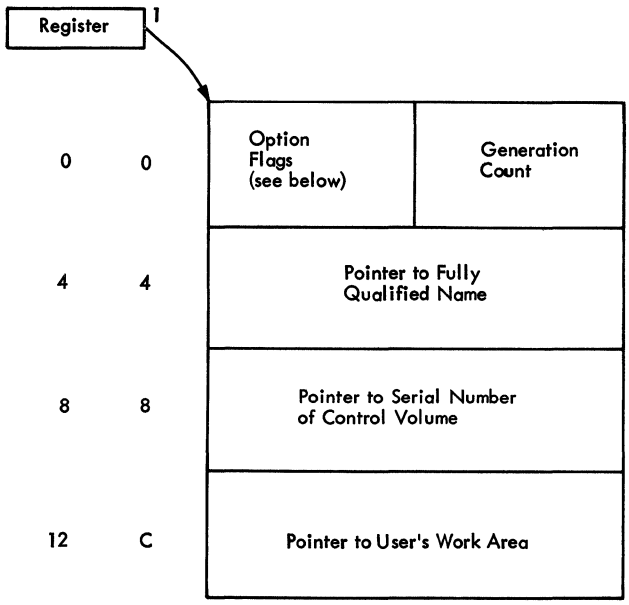
Field 4: This field contains the flags which indicate special handling for generation data sets. Bit 7 indicates the Empty option and bit 6 indicates the Delete option. Bits 0-5 are reserved and are always zero.

Field 5: This field indicates the maximum number of entries to be maintained in the index at one time.

Field 6: This field indicates the number of entries currently in the index.

USER'S PARAMETER LIST

This parameter list must be supplied by the user before he calls the catalog management routines. The CAMLST macro instruction, described in IBM System/360 Operating System Programmer's Guide, form C28-6550, can be used to generate the list.



[1] At entry to IGC0002F, register 1 points to the user's parameter list. At all other times, register 8 points there.

		Option Flags	
Byte 0	1...	Catalog is on System Residence Device	
	.X..	Reserved	
	..1.	CTLG	Catalog a data set
	...1	RECAT	Recatalog a data set
	.... 1...	UNCAT	Uncatalog a data set
	.... .X..	Reserved	
	.... ..1.	BLOCK	Read a block by TTR
	X	Reserved	
Byte 1	X...	Reserved	
	.1..	BLDX	Build normal index structure
	..1.	BLDG	Build generation index
	...1	BLDA	Build an alias to a high-level name
	.... 1...	LINKX	Connect control volumes
	.... .1..	DLTX	Delete an index Structure
	.... ..X.	Reserved	
	1	DLTA	Delete an alias entry
Byte 2	1...	DRPX	Disconnect control volumes
	.1..	DELETE	Scratch generation data sets when they are uncataloged
	..XX	Reserved	
	.... 1...	EMPTY	Remove all entries from the index when the maximum generation count has been reached
	XXX	Reserved	

Note: Function is locate by name if all flags are zero.

Figure 12. User's Parameter List

This section includes miscellaneous charts and tables that might be useful in locating program errors.

MODULE SELECTION CHART

This chart can be used to determine what modules of the catalog management routine will be used to perform a particular function, given the function required and the current status of the catalog.

			1	2	3	4	5	6	7	8
FUNCTION:	LOCATE		Y	Y						
	OTHER				Y	Y	Y	Y	Y	Y
TYPE INDEX FOUND:	NORMAL		Y	Y	Y					
		GENERATION		Y					Y	Y
		NONE					Y	Y		
UNFORMATTED CATALOG					N	Y	N	Y	N	Y
	IGC0002F		X	X	X	X	X	X	X	X
	IGC0002H		X	X	X	X	X	X	X	X
	IGG0CLF2					X		X		X
	IGG0CLC1		X	X	X	X	X	X	X	X
	IGG0CLC2				X	X	X	X		
	IGG0CLC4			X					X	X
	IGG0CLC5								X	X
	IGG0CLC3				X	X	X	X	X	X

Figure 13. Module Selection Chart

REGISTER USAGE

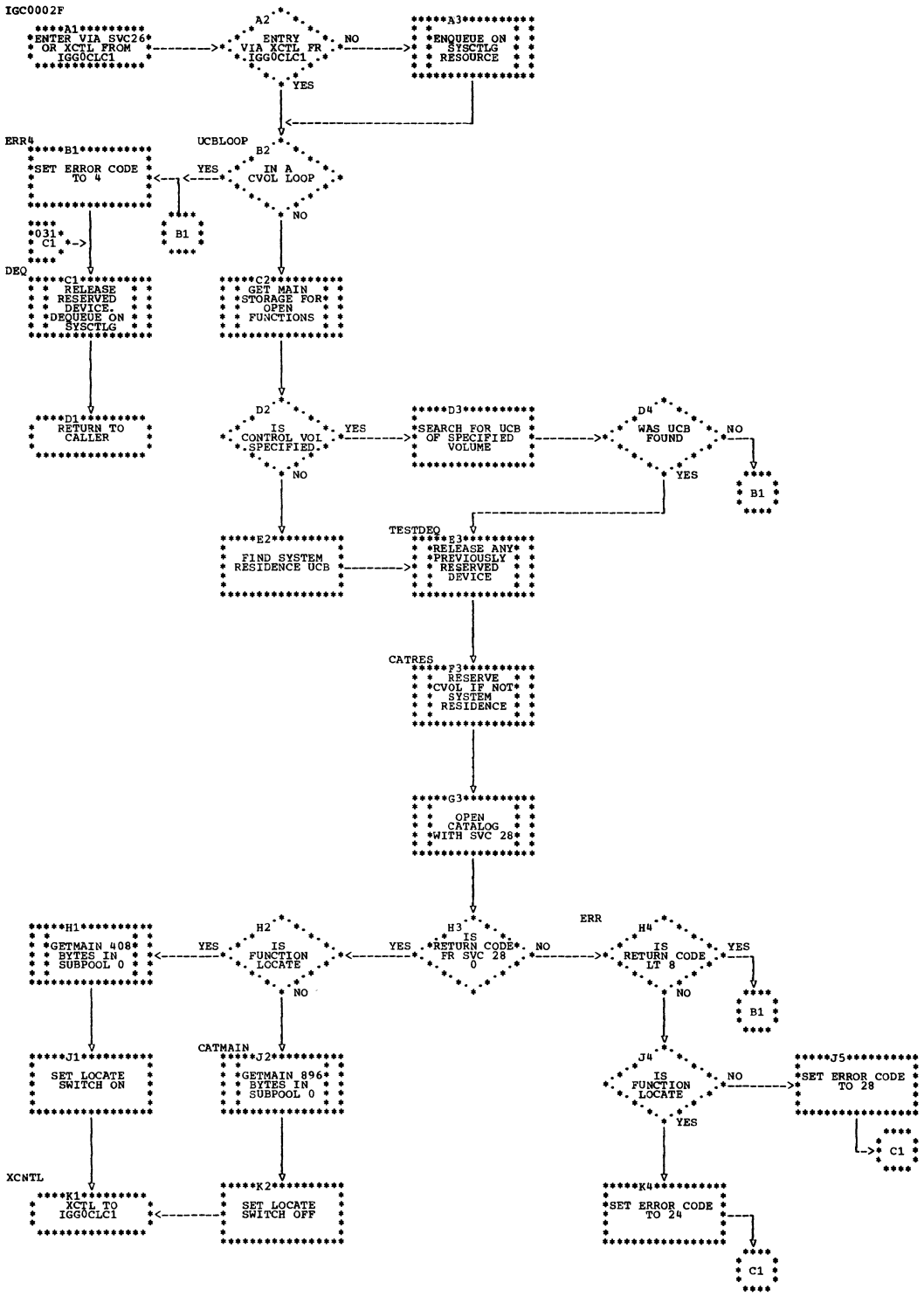
Figure 14 is a register usage chart. In the chart, the contents of certain registers are given as they appear at entry to each module and just before each module loses control. All entries in the table, except those marked "*", are addresses. That is, when the table indicates that at entry to module IGG0CLC1 register 9 is 'DCB', this means that register 9 contains the address of the data control block. When the table indicates that at entry to module IGG0CLC2 register 6 is "No. of Levels Searched *," this means that register 6 contains that number.

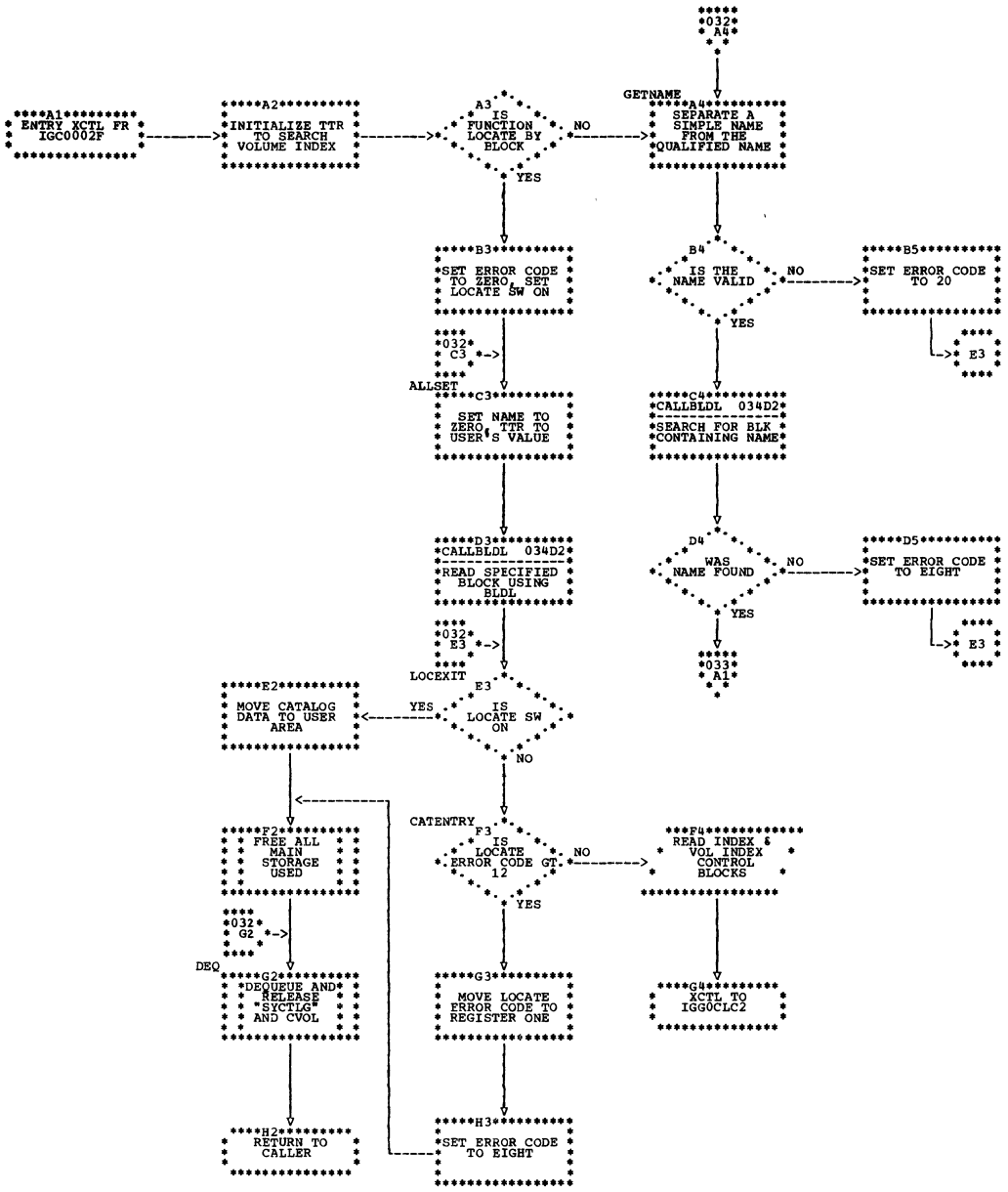
Module Name		Registers													
		0	1	2	3	4	5	6	8	9	10	11	12	13	15
IGC0002F	Entry		User's Parameter List				SVRB								
	Exit						ENQ Parameter List		User's Parameter List	DCB			Work Area	BLDL Work Area	
IGG0CLC1	Entry						ENQ Parameter List		User's Parameter List	DCB			Work Area	BLDL Work Area	
	Exit (To IGG0CLC2 or IGG0CLC4)						ENQ Parameter List	No. of Levels Searched*	User's Parameter List	DCB		Generation Index Block	Work Area	BLDL Work Area	
	Exit (To User)	No. of Levels Searched*	Locate Error Code*												Error Code*
IGG0CLC2	Entry							No. of Levels Searched*	User's Parameter List	DCB	Work Area				
	Exit							No. of Levels Searched*	User's Parameter List	DCB	Work Area				
IGG0CLC3	Entry								User's Parameter List	DCB	Work Area				
	Exit	No. of Levels Searched*	Locate Error Code*												Index Catalog Error Code*
IGG0CLC4	Entry						Entry Indicator*		User's Parameter List	DCB			Work Area	BLDL Work Area	
	Exit						Entry Indicator*		User's Parameter List	DCB		Gen. Index Pointer Entry	Work Area	BLDL Work Area	
IGG0CLC5	Entry						Entry Indicator*		User's Parameter List	DCB		Gen. Index Pointer Entry	Work Area	BLDL Work Area	
	Exit (User)	No. of Levels Searched*	Locate Error Code*												Index Catalog Error Code*
	Exit (IGG0CLC3)								User's Parameter List	DCB			Work Area	BLDL Work Area	
	Exit (IGG0CLC4)						Entry Indicator*		User's Parameter List	DCB			Work Area	BLDL Work Area	
IGG0CLC6	Entry				Index Catalog Error Code*	Locate Error Code*		No. of Levels Searched*		DCB				Locate Work Area	
	Exit	No. of Levels Searched*	Locate Error Code*												Error Code*
IGC0002H	Entry (Via SVC 28)		UCB of CVOL or DCB											Work Area for DEB/DCB	Bin Number if CVOL is on 2321*
	Entry (XCTL from Extend Routine)	A Negative Value*				Extend Work Area			Bin Number if 2321*	DCB	TTR of new Extent*	UCB			
	Exit (To Caller)														Error Code*
	Exit (To DADSM Extend Routine)			DCB		Work Area	DEB				UCB			Non-zero*	
	Exit (To IGG0CLF2)	Zero*	DCB	No. of Blocks/Track *	Subpool ID and Size of Work Area*	Work Area		Begin TTR*							
IGG0CLF2	Entry			DCB		Work Area	DEB				UCB			Non-zero*	
	Exit														Error Code*
		0	1	2	3	4	5	6	8	9	10	11	12	13	15

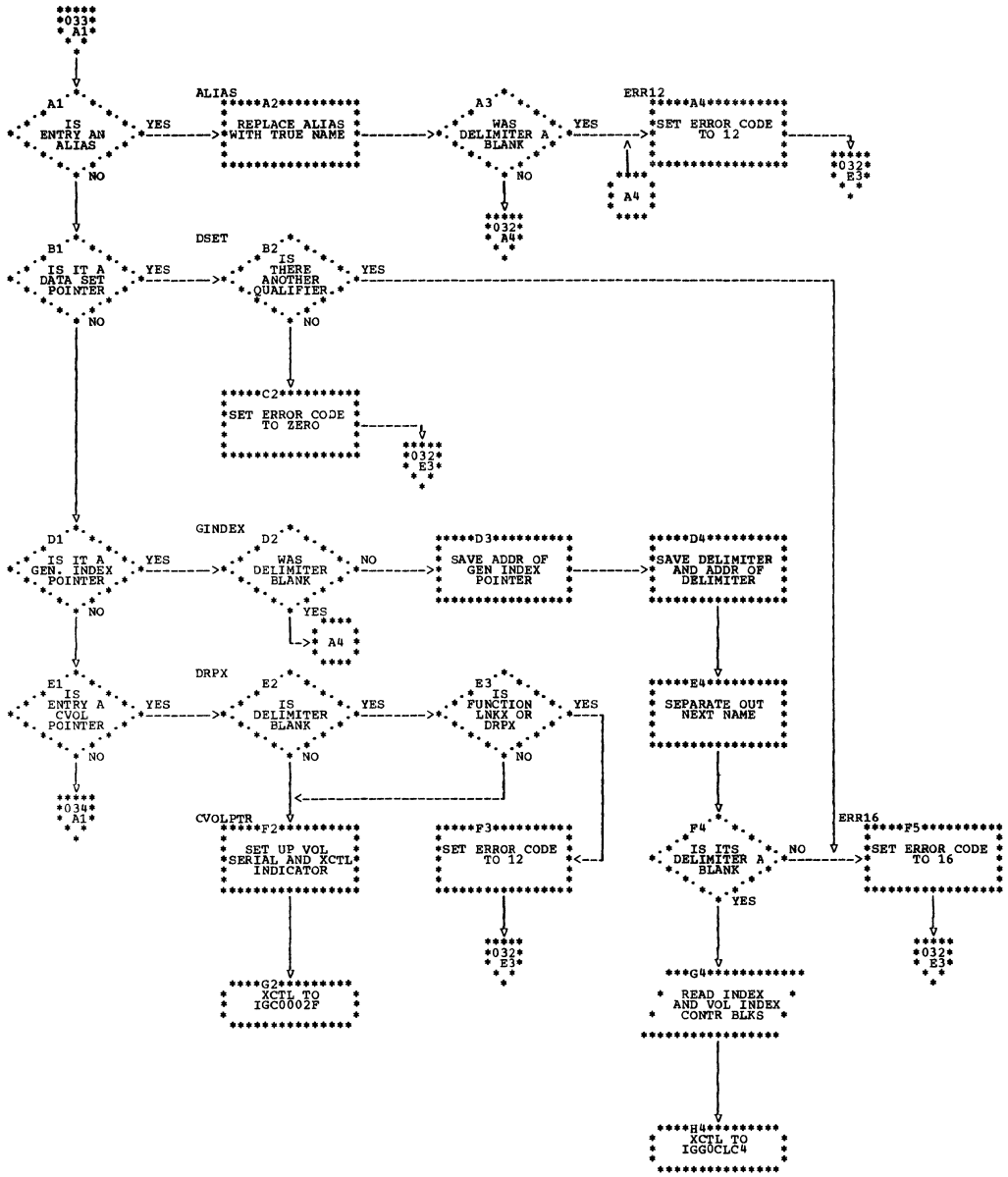
Figure 14. Register Usage

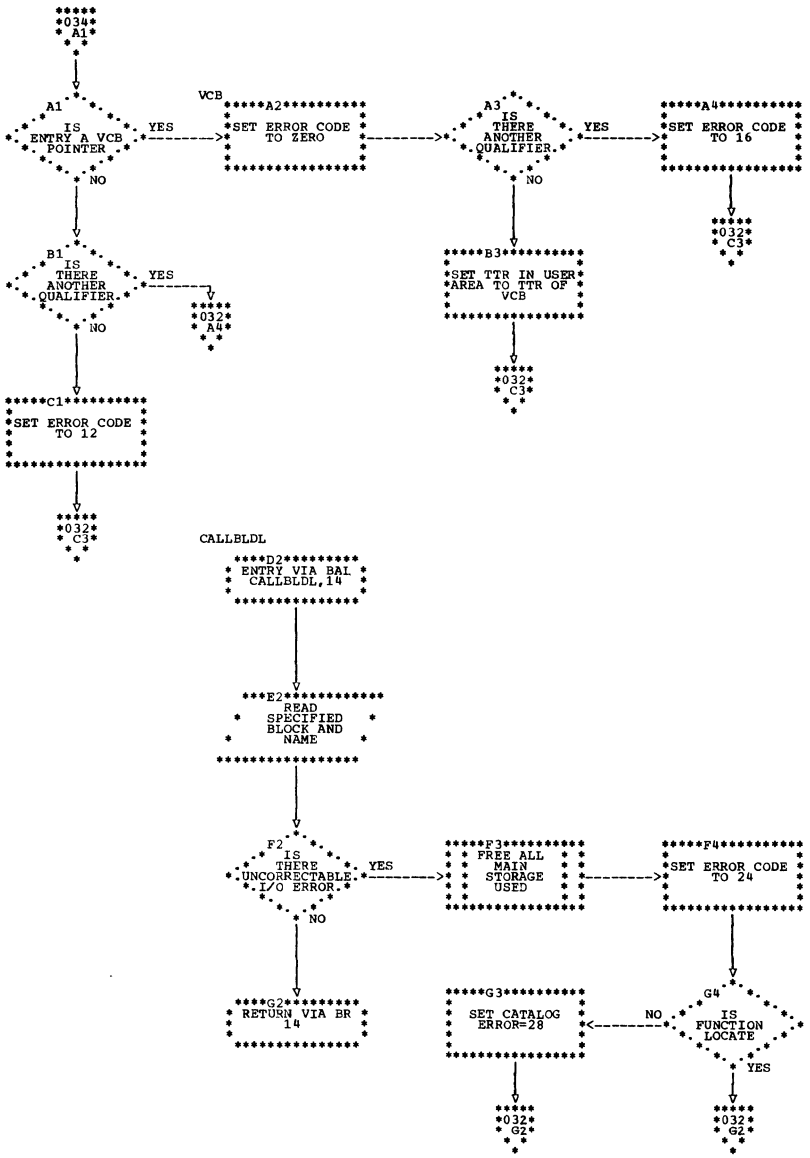
APPENDIX A: FLOWCHARTS

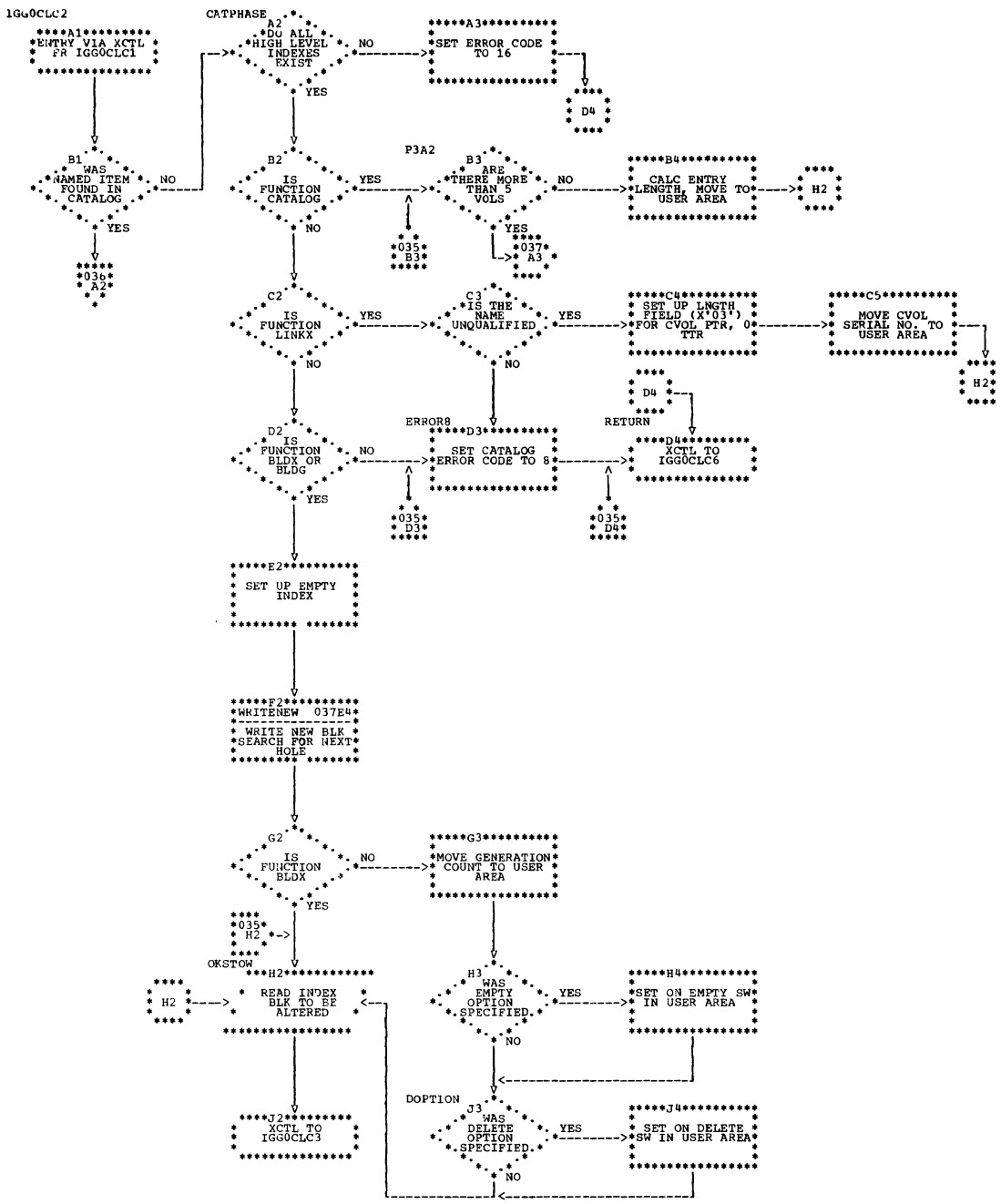
These flowcharts illustrate the operation of the catalog management routines module by module. Each label in the charts is taken directly from the assembler language source code for the module. The charts are intended to bridge the gap between the textual material of this manual and the code itself, so they are best used in conjunction with the code and the text (particularly the Program Organization section).

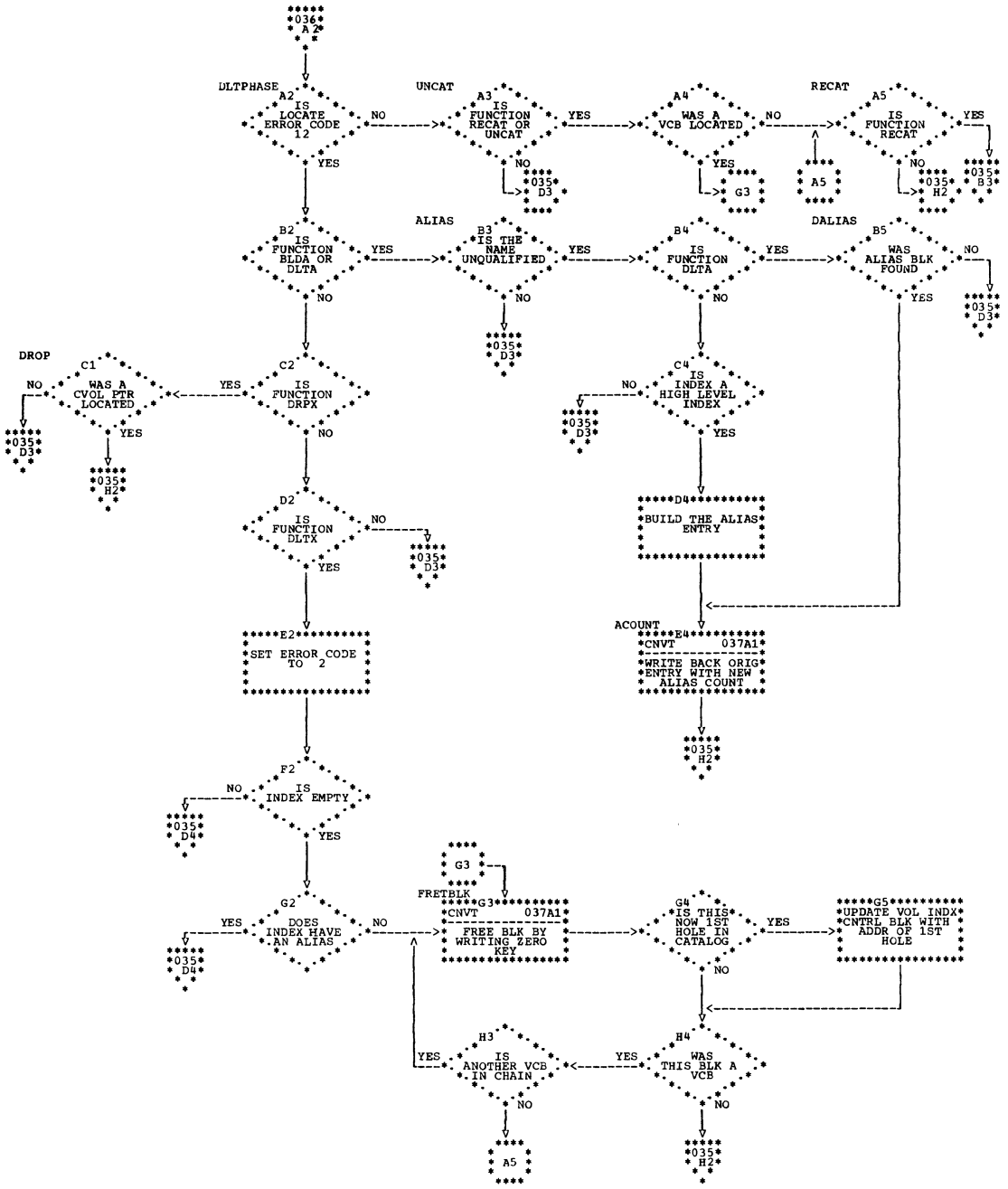

IGC0002F
A1
ENTER VIA SVC26 OR XCTL FROM IGG0CLC1
A2
ENTRY VIA XCTL FR IGG0CLC1
NO
YES
A3
ENQUEUE ON SYSCTLG RESOURCE
ERR4
B1
SET ERROR CODE TO 4
UCBLOOP
B2
IN A CVOL LOOP
YES
NO
B1
031 C1
DEQ
C1
RELEASE RESERVED DEVICE. DEQUEUE ON SYSCTLG
C2
GET MAIN STORAGE FOR OPEN FUNCTIONS
D1
RETURN TO CALLER
D2
IS CONTROL VOL SPECIFIED
YES
NO
D3
SEARCH FOR UCB OF SPECIFIED VOLUME
D4
WAS UCB FOUND
NO
YES
B1
E2
FIND SYSTEM RESIDENCE UCB
TESTDEQ
E3
RELEASE ANY PREVIOUSLY RESERVED DEVICE
CATRES
F3
RESERVE CVOL IF NOT SYSTEM RESIDENCE
G3
OPEN CATALOG WITH SVC 28
H1
GETMAIN 408 BYTES IN SUBPOOL 0
H2
IS FUNCTION LOCATE
YES
NO
H3
IS RETURN CODE FR SVC 28 0
YES
NO
ERR
H4
IS RETURN CODE LT 8
YES
NO
B1
J1
SET LOCATE SWITCH ON
CATMAIN
J2
GETMAIN 896 BYTES IN SUBPOOL 0
J4
IS FUNCTION LOCATE
NO
YES
J5
SET ERROR CODE TO 28
C1
XCNTL
K1
XCTL TO IGG0CLC1
K2
SET LOCATE SWITCH OFF
K4
SET ERROR CODE TO 24
C1

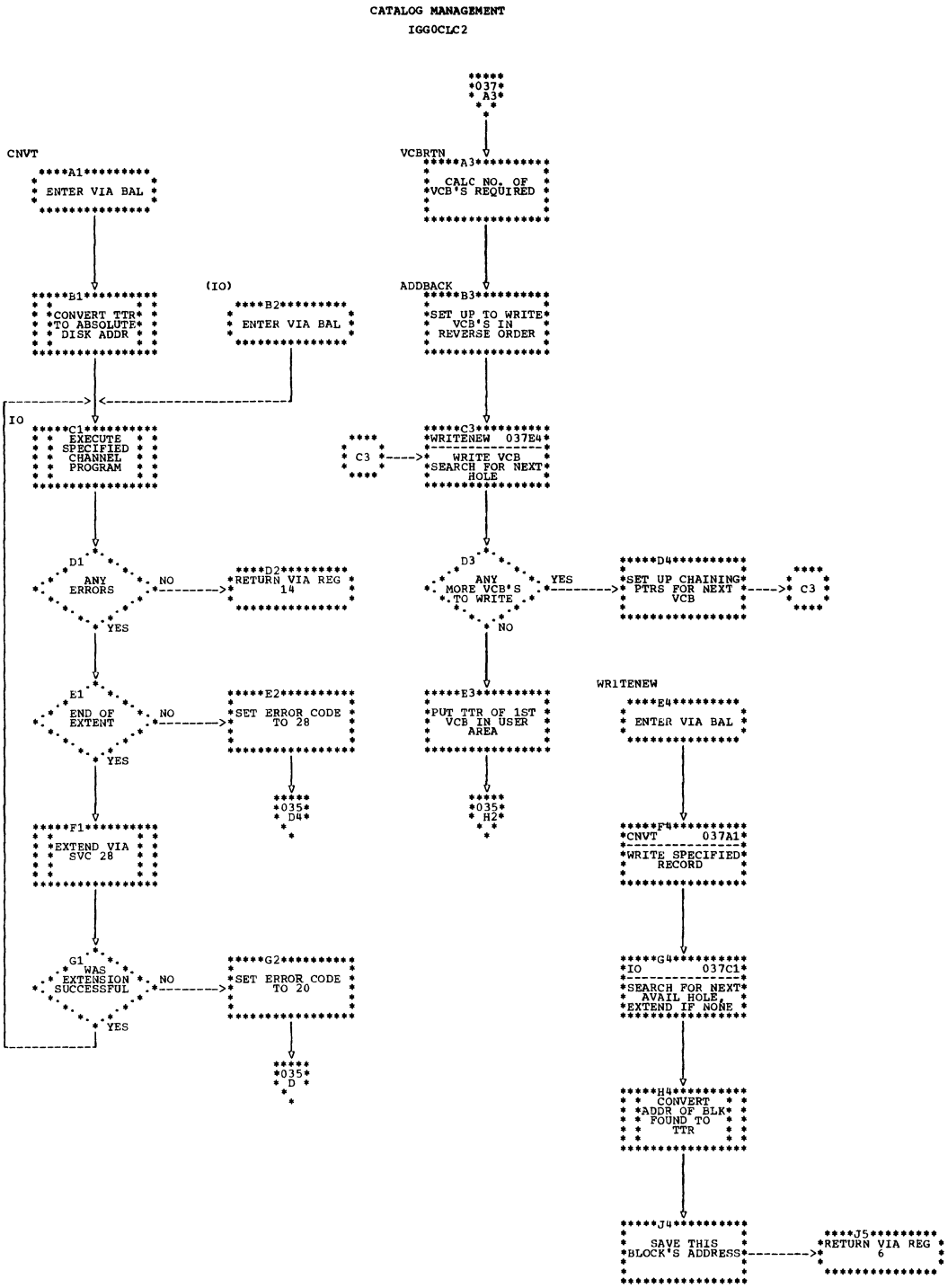

A1
ENTRY XCTL FR IGC0002F
A2
INITIALIZE TTR TO SEARCH VOLUME INDEX
A3
IS FUNCTION LOCATE BY BLOCK
NO
YES
032 A4
GETNAME
A4
SEPARATE A SIMPLE NAME FROM THE QUALIFIED NAME
B3
SET ERROR CODE TO ZERO, SET LOCATE SW ON
B4
IS THE NAME VALID
NO
YES
B5
SET ERROR CODE TO 20
E3
032 C3
ALLSET
C3
SET NAME TO ZERO, TTR TO USER'S VALUE
C4
CALLBLDL 034D2
SEARCH FOR BLK CONTAINING NAME
D3
CALLBLDL 034D2
READ SPECIFIED BLOCK USING BLDL
D4
WAS NAME FOUND
NO
YES
D5
SET ERROR CODE TO EIGHT
E3
032 E3
LOCEXIT
033 A1
E2
MOVE CATALOG DATA TO USER AREA
E3
IS LOCATE SW ON
YES
NO
F2
FREE ALL MAIN STORAGE USED
F3
IS LOCATE ERROR CODE GT. 12
NO
YES
CATENTRY
F4
READ INDEX & VOL INDEX CONTROL BLOCKS
032 G2
DEQ
G2
DEQUEUE AND RELEASE "SYCTLG" AND CVOL
G3
MOVE LOCATE ERROR CODE TO REGISTER ONE
G4
XCTL TO IGG0CLC2
H2
RETURN TO CALLER
H3
SET ERROR CODE TO EIGHT

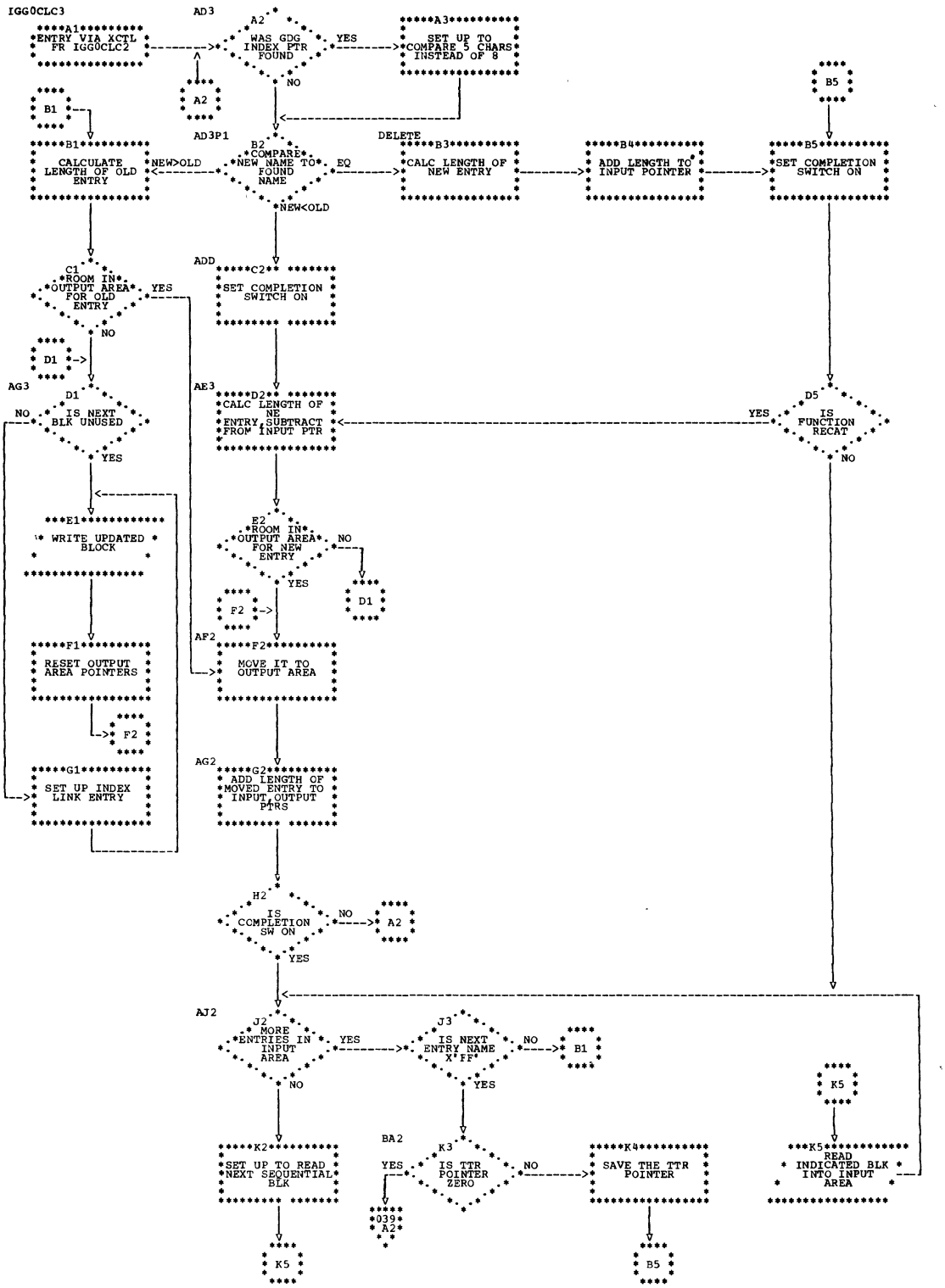

033
A1
A1
IS ENTRY AN ALIAS
YES
NO
ALIAS
A2
REPLACE ALIAS WITH TRUE NAME
A3
WAS DELIMITER A BLANK
YES
NO
032
A4
ERR12
A4
SET ERROR CODE TO 12
A4
032
E3
B1
IS IT A DATA SET POINTER
YES
NO
DSET
B2
IS THERE ANOTHER QUALIFIER
YES
NO
C2
SET ERROR CODE TO ZERO
032
E3
D1
IS IT A GEN. INDEX POINTER
YES
NO
GINDEX
D2
WAS DELIMITER BLANK
NO
YES
A4
D3
SAVE ADDR OF GEN INDEX POINTER
D4
SAVE DELIMITER AND ADDR OF DELIMITER
E1
IS ENTRY A CVOL POINTER
YES
NO
034
A1
DRPX
E2
IS DELIMITER BLANK
YES
NO
E3
IS FUNCTION LNKX OR DRPX
YES
NO
E4
SEPARATE OUT NEXT NAME
CVOLPTR
F2
SET UP VOL SERIAL AND XCTL INDICATOR
F3
SET ERROR CODE TO 12
032
E3
F4
IS ITS DELIMITER A BLANK
NO
YES
ERR16
F5
SET ERROR CODE TO 16
032
E3
G2
XCTL TO IGC0002F
G4
READ INDEX AND VOL INDEX CONTR BLKS
H4
XCTL TO IGG0CLC4

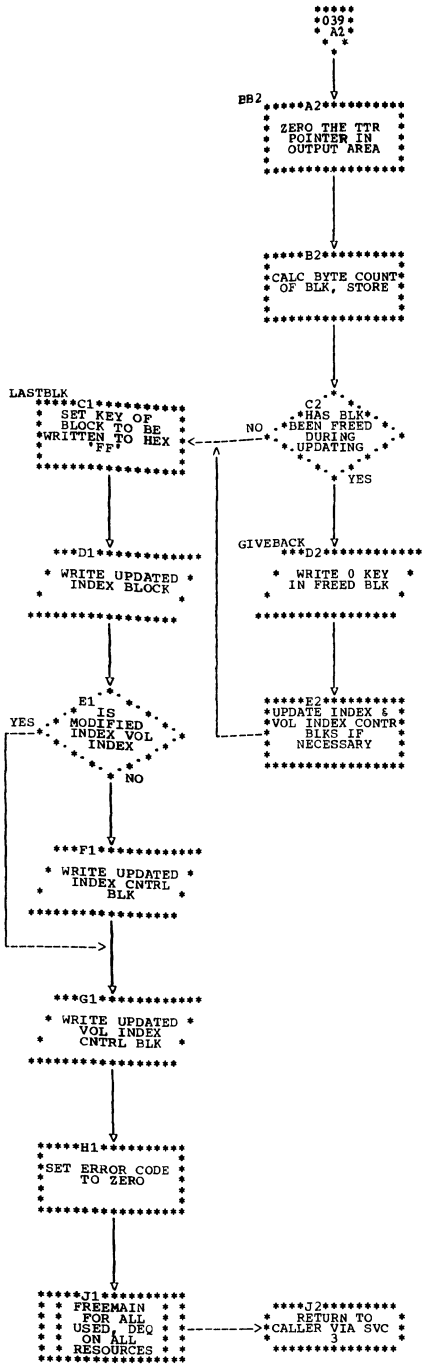

034
A1
A1
IS ENTRY A VCB POINTER
YES
NO
VCB
A2
SET ERROR CODE TO ZERO
A3
IS THERE ANOTHER QUALIFIER
YES
NO
A4
SET ERROR CODE TO 16
032
C3
B1
IS THERE ANOTHER QUALIFIER
YES
NO
032
A4
B3
SET TTR IN USER AREA TO TTR OF VCB
032
C3
C1
SET ERROR CODE TO 12
032
C3
CALLBLDL
D2
ENTRY VIA BAL CALLBLDL,14
E2
READ SPECIFIED BLOCK AND NAME
F2
IS THERE UNCORRECTABLE I/O ERROR
YES
NO
F3
FREE ALL MAIN STORAGE USED
F4
SET ERROR CODE TO 24
G2
RETURN VIA BR 14
G3
SET CATALOG ERROR=28
G4
IS FUNCTION LOCATE
NO
YES
032
G2

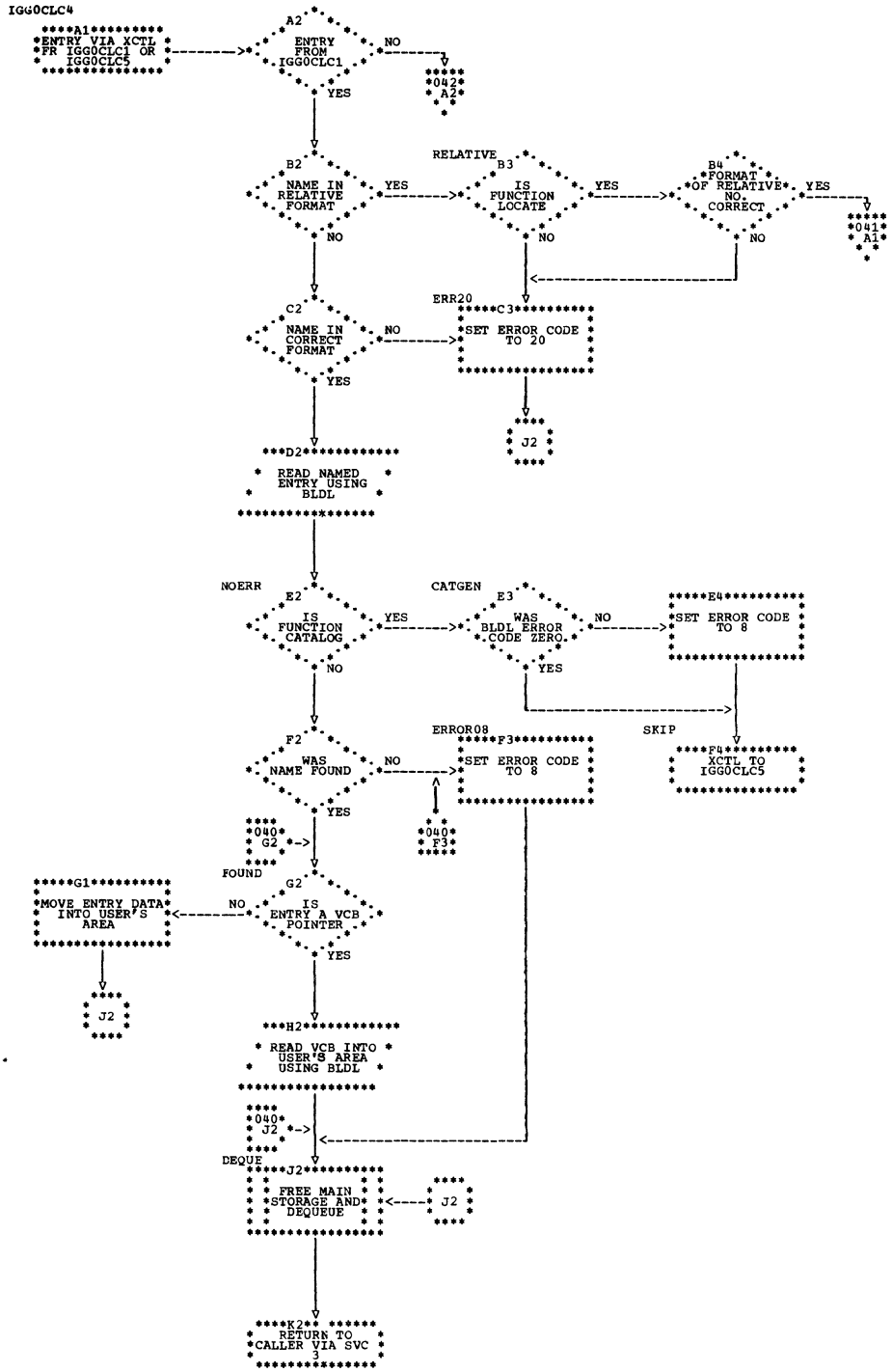

IGG0CLC2
A1
ENTRY VIA XCTL FR IGG0CLC1
B1
WAS NAMED ITEM FOUND IN CATALOG
NO
YES
036 A2
CATPHASE
A2
DO ALL HIGH LEVEL INDEXES EXIST
NO
YES
A3
SET ERROR CODE TO 16
D4
B2
IS FUNCTION CATALOG
YES
NO
035 B3
P3A2
B3
ARE THERE MORE THAN 5 VOLS
NO
YES
037 A3
B4
CALC ENTRY LENGTH, MOVE TO USER AREA
H2
C2
IS FUNCTION LINKX
YES
NO
C3
IS THE NAME UNQUALIFIED
YES
NO
C4
SET UP LNGTH FIELD (X'03') FOR CVOL PTR, 0 TTR
C5
MOVE CVOL SERIAL NO. TO USER AREA
H2
D4
RETURN
D4
XCTL TO IGG0CLC6
D2
IS FUNCTION BLDX OR BLDG
NO
YES
ERROR8
D3
SET CATALOG ERROR CODE TO 8
035 D3
035 D4
E2
SET UP EMPTY INDEX
F2
WRITENEW 037E4
WRITE NEW BLK SEARCH FOR NEXT HOLE
G2
IS FUNCTION BLDX
NO
YES
G3
MOVE GENERATION COUNT TO USER AREA
035 H2
OKSTOW
H2
READ INDEX BLK TO BE ALTERED
H2
H3
WAS EMPTY OPTION SPECIFIED.
YES
NO
H4
SET ON EMPTY SW IN USER AREA
DOPTION
J2
XCTL TO IGG0CLC3
J3
WAS DELETE OPTION SPECIFIED.
YES
NO
J4
SET ON DELETE SW IN USER AREA

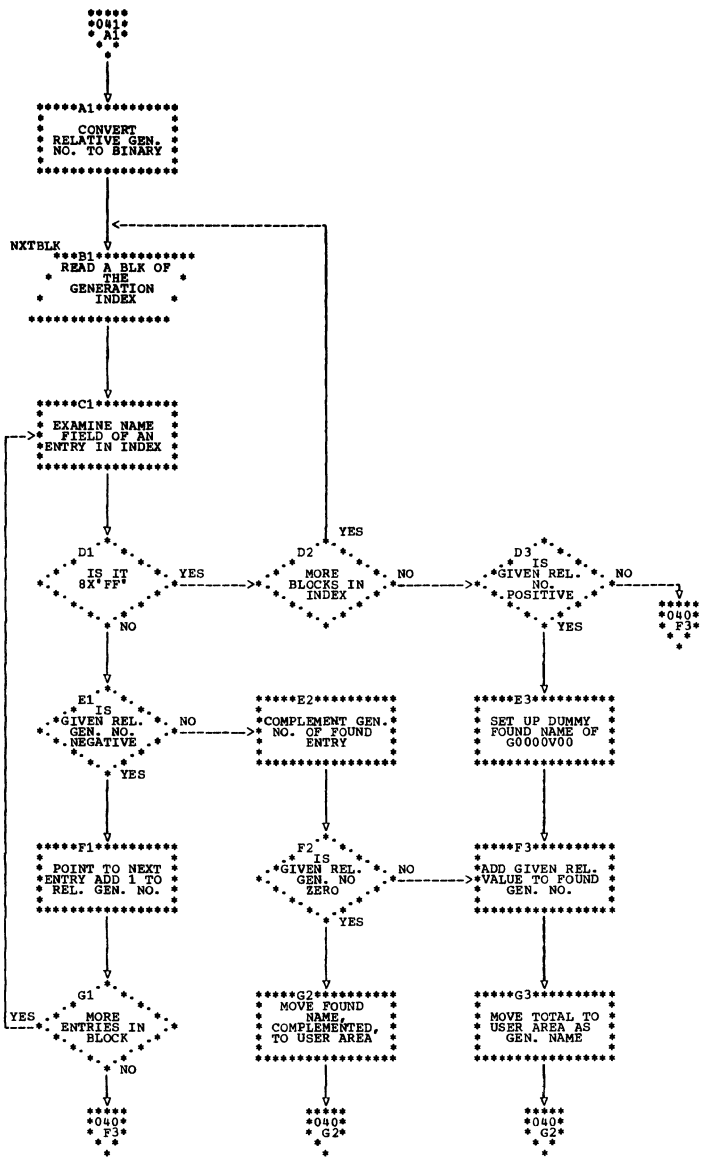

036
A2
DLTPHASE
A2
IS LOCATE ERROR CODE 12
NO
YES
UNCAT
A3
IS FUNCTION RECAT OR UNCAT
YES
NO
035 D3
A4
WAS A VCB LOCATED
NO
YES
G3
RECAT
A5
IS FUNCTION RECAT
YES
NO
A5
035 B3
035 H2
B2
IS FUNCTION BLDA OR DLTA
YES
NO
ALIAS
B3
IS THE NAME UNQUALIFIED
YES
NO
035 D3
B4
IS FUNCTION DLTA
YES
NO
DALIAS
B5
WAS ALIAS BLK FOUND
NO
YES
035 D3
DROP
C1
WAS A CVOL PTR LOCATED
NO
YES
035 D3
035 H2
C2
IS FUNCTION DRPX
YES
NO
C4
IS INDEX A HIGH LEVEL INDEX
NO
YES
035 D3
D2
IS FUNCTION DLTX
NO
YES
035 D3
D4
BUILD THE ALIAS ENTRY
ACOUNT
E4
CNVT 037A1
WRITE BACK ORIG ENTRY WITH NEW ALIAS COUNT
035 H2
E2
SET ERROR CODE TO 2
F2
IS INDEX EMPTY
NO
YES
035 D4
G3
FRETBLK
G3
CNVT 037A1
FREE BLK BY WRITING ZERO KEY
G2
DOES INDEX HAVE AN ALIAS
YES
NO
035 D4
G4
IS THIS NOW 1ST HOLE IN CATALOG
YES
NO
G5
UPDATE VOL INDX CNTRL BLK WITH ADDR OF 1ST HOLE
H3
IS ANOTHER VCB IN CHAIN
YES
NO
A5
H4
WAS THIS BLK A VCB
YES
NO
035 H2

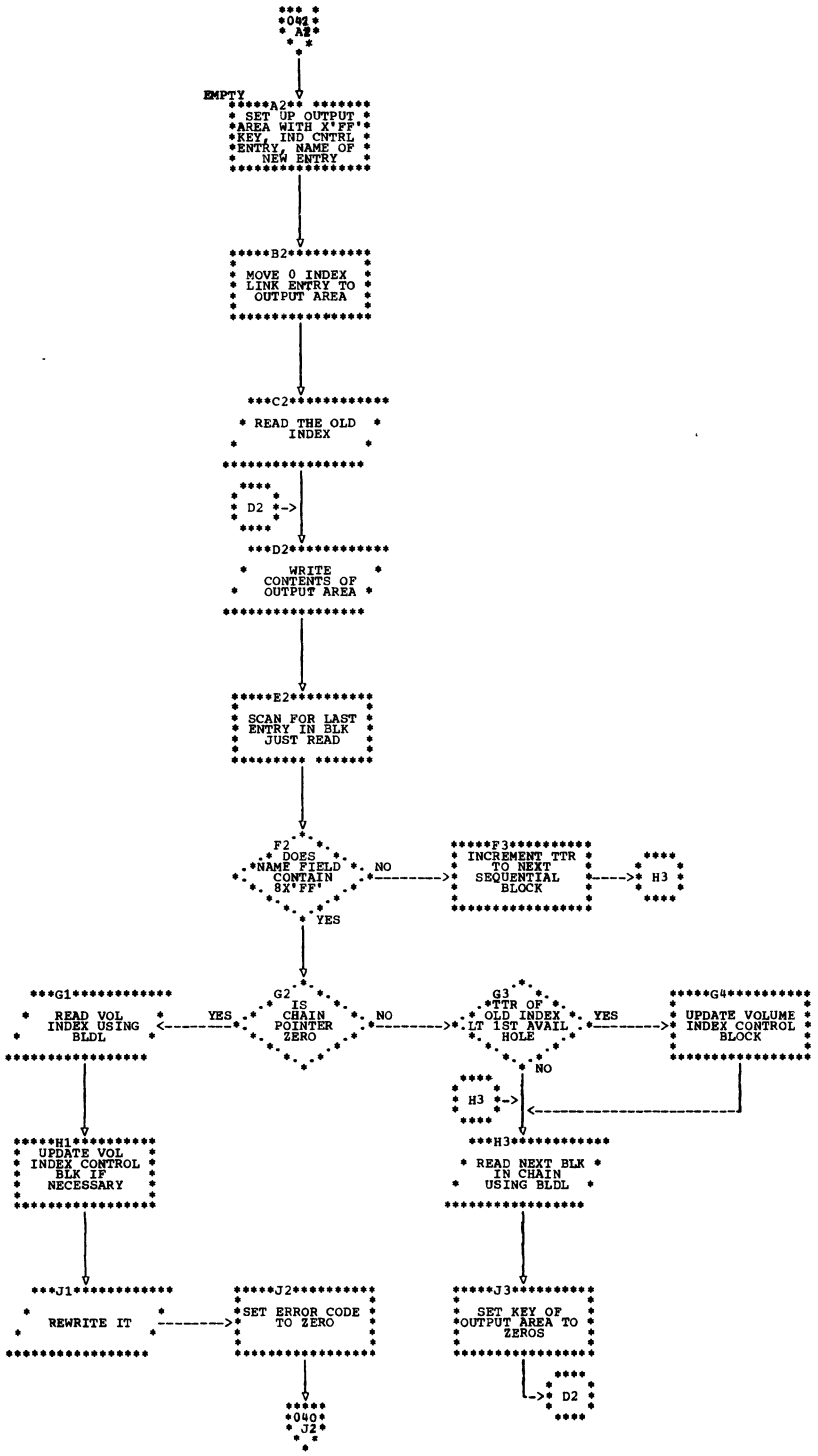

CATALOG MANAGEMENT
IGG0CLC2
037 A3
CNVT
A1 ENTER VIA BAL
VCBRTN
A3 CALC NO. OF VCB'S REQUIRED
B1 CONVERT TTR TO ABSOLUTE DISK ADDR
(IO)
B2 ENTER VIA BAL
ADDBACK
B3 SET UP TO WRITE VCB'S IN REVERSE ORDER
IO
C1 EXECUTE SPECIFIED CHANNEL PROGRAM
C3
WRITENEW 037E4
WRITE VCB SEARCH FOR NEXT HOLE
D1 ANY ERRORS
NO
D2 RETURN VIA REG 14
YES
D3 ANY MORE VCB'S TO WRITE
YES
D4 SET UP CHAINING PTRS FOR NEXT VCB
C3
NO
E1 END OF EXTENT
NO
E2 SET ERROR CODE TO 28
YES
035 D4
E3 PUT TTR OF 1ST VCB IN USER AREA
035 H2
WR1TENEW
E4 ENTER VIA BAL
F1 EXTEND VIA SVC 28
F4 CNVT 037A1 WRITE SPECIFIED RECORD
G1 WAS EXTENSION SUCCESSFUL
NO
G2 SET ERROR CODE TO 20
YES
035 D
G4 IO 037C1 SEARCH FOR NEXT AVAIL HOLE EXTEND IF NONE
H4 CONVERT ADDR OF BLK FOUND TO TTR
J4 SAVE THIS BLOCK'S ADDRESS
J5 RETURN VIA REG 6

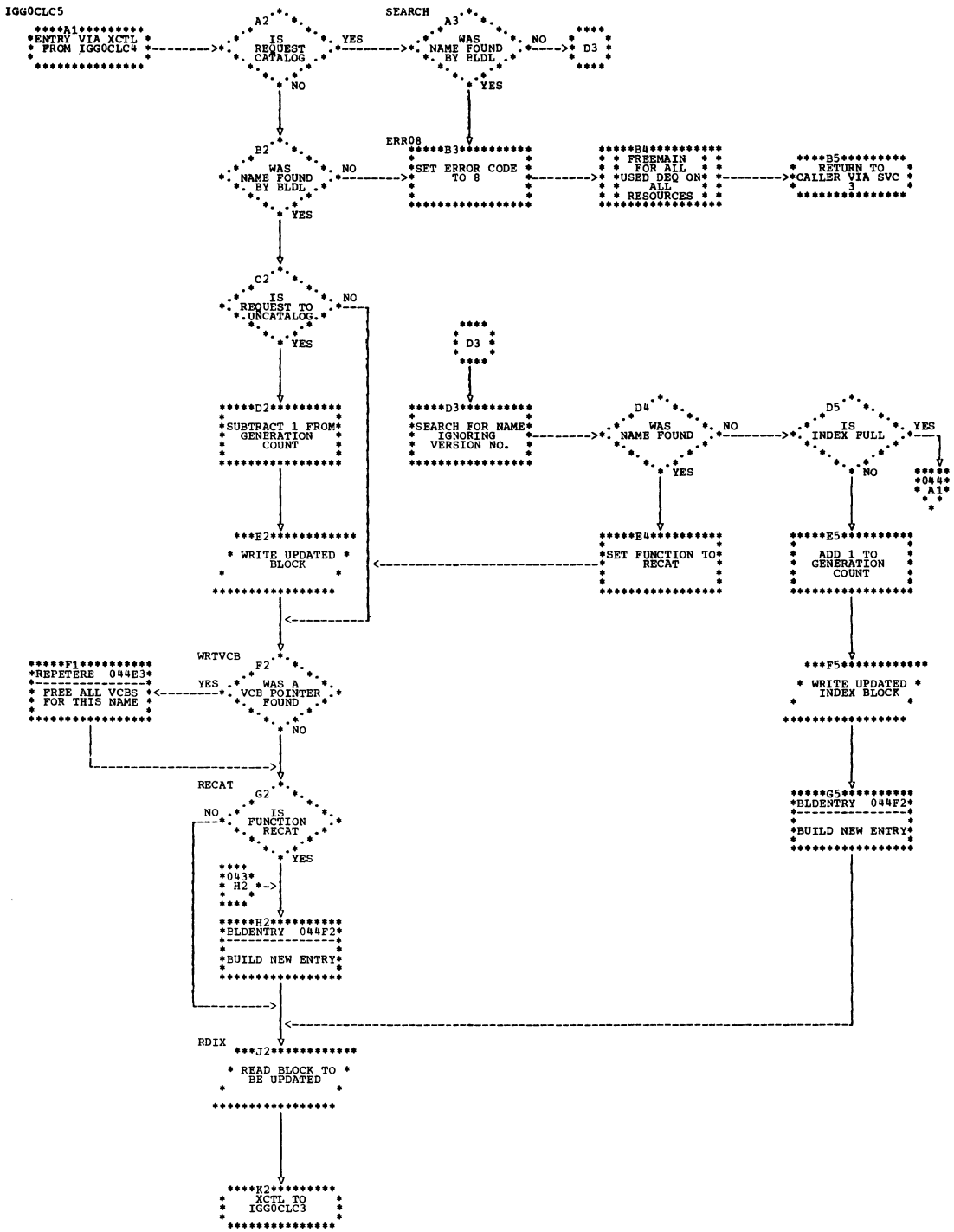

IGG0CLC3
A1
ENTRY VIA XCTL FR IGG0CLC2
AD3
A2
WAS GDG INDEX PTR FOUND
YES
A3
SET UP TO COMPARE 5 CHARS INSTEAD OF 8
NO
A2
B1
AD3P1
B2
COMPARE NEW NAME TO FOUND NAME
DELETE
B3
CALC LENGTH OF NEW ENTRY
B4
ADD LENGTH TO INPUT POINTER
B5
SET COMPLETION SWITCH ON
B1
CALCULATE LENGTH OF OLD ENTRY
NEW>OLD
EQ
NEW<OLD
C1
ROOM IN OUTPUT AREA FOR OLD ENTRY
YES
NO
ADD
C2
SET COMPLETION SWITCH ON
D1
AG3
D1
IS NEXT BLK UNUSED
NO
YES
AE3
D2
CALC LENGTH OF NE ENTRY, SUBTRACT FROM INPUT PTR
D5
IS FUNCTION RECAT
YES
NO
E1
WRITE UPDATED BLOCK
E2
ROOM IN OUTPUT AREA FOR NEW ENTRY
NO
YES
D1
F2
AF2
F1
RESET OUTPUT AREA POINTERS
F2
MOVE IT TO OUTPUT AREA
AG2
G1
SET UP INDEX LINK ENTRY
G2
ADD LENGTH OF MOVED ENTRY TO INPUT, OUTPUT PTRS
H2
IS COMPLETION SW ON
NO
A2
YES
AJ2
J2
MORE ENTRIES IN INPUT AREA
YES
NO
J3
IS NEXT ENTRY NAME X'FF'
NO
B1
YES
K5
K5
READ INDICATED BLK INTO INPUT AREA
K2
SET UP TO READ NEXT SEQUENTIAL BLK
BA2
K3
IS TTR POINTER ZERO
YES
NO
K4
SAVE THE TTR POINTER
039
A2
K5
B5

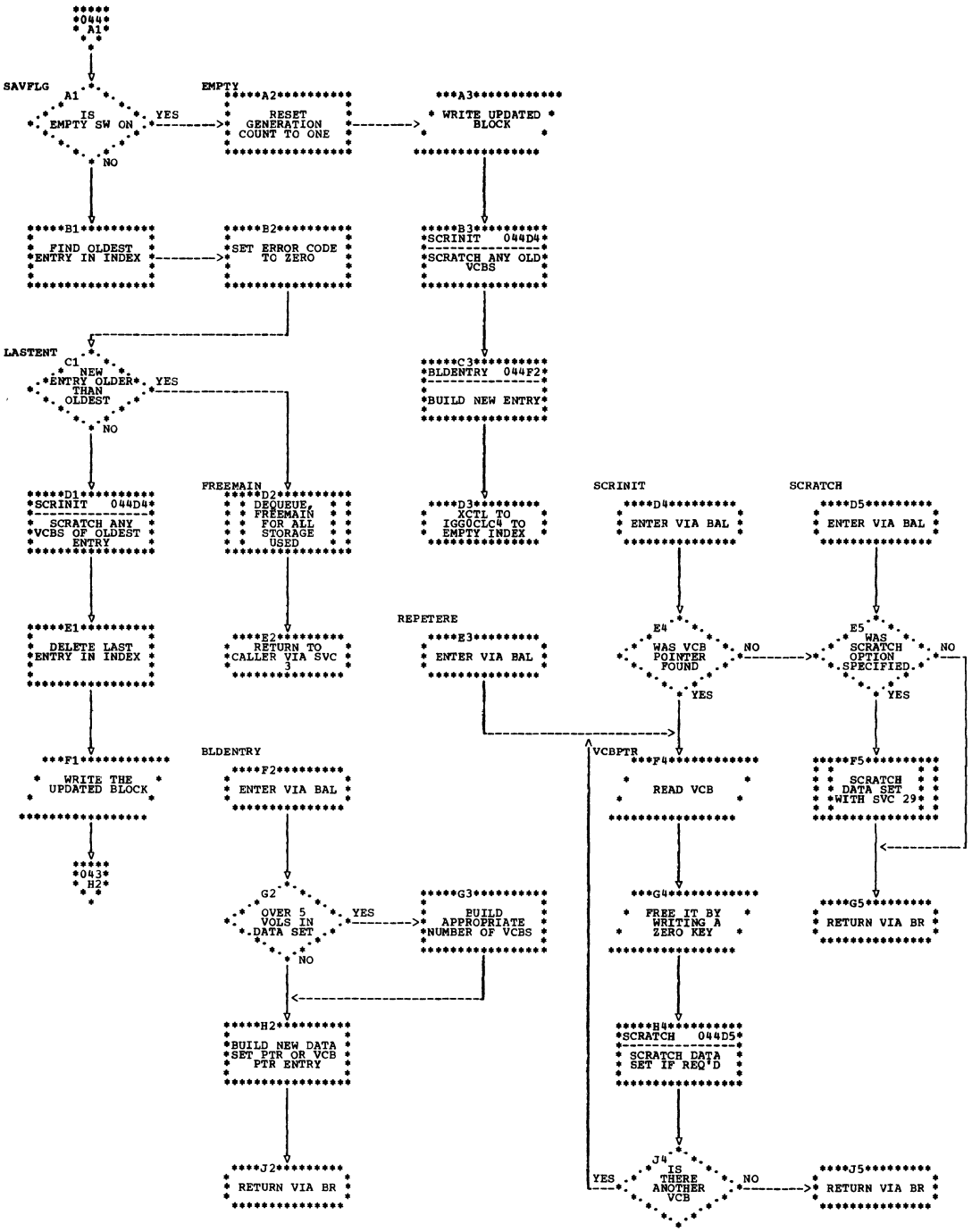

039
A2
BB2
A2
ZERO THE TTR POINTER IN OUTPUT AREA
B2
CALC BYTE COUNT OF BLK, STORE
LASTBLK
C1
SET KEY OF BLOCK TO BE WRITTEN TO HEX 'FF'
C2
HAS BLK BEEN FREED DURING UPDATING
NO
YES
GIVEBACK
D1
WRITE UPDATED INDEX BLOCK
D2
WRITE 0 KEY IN FREED BLK
E1
IS MODIFIED INDEX VOL INDEX
YES
NO
E2
UPDATE INDEX & VOL INDEX CONTR BLKS IF NECESSARY
F1
WRITE UPDATED INDEX CNTRL BLK
G1
WRITE UPDATED VOL INDEX CNTRL BLK
H1
SET ERROR CODE TO ZERO
J1
FREEMAIN FOR ALL USED, DEQ ON ALL RESOURCES
J2
RETURN TO CALLER VIA SVC 3

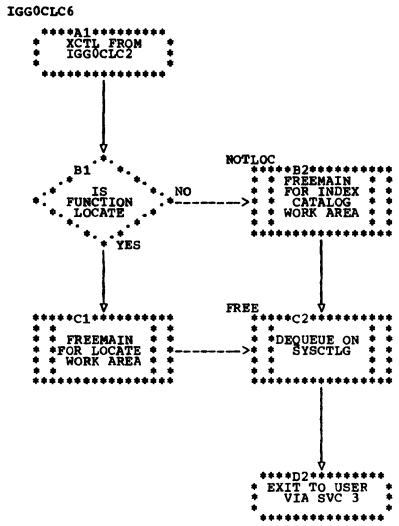

IGG0CLC4
A1
ENTRY VIA XCTL FR IGG0CLC1 OR IGG0CLC5
A2
ENTRY FROM IGG0CLC1
NO
042 A2
YES
B2
NAME IN RELATIVE FORMAT
YES
RELATIVE
B3
IS FUNCTION LOCATE
YES
B4
FORMAT OF RELATIVE NO. CORRECT
YES
041 A1
NO
NO
NO
C2
NAME IN CORRECT FORMAT
NO
ERR20
C3
SET ERROR CODE TO 20
YES
J2
D2
READ NAMED ENTRY USING BLDL
NOERR
E2
IS FUNCTION CATALOG
YES
CATGEN
E3
WAS BLDL ERROR CODE ZERO
NO
E4
SET ERROR CODE TO 8
NO
YES
SKIP
F2
WAS NAME FOUND
NO
ERROR08
F3
SET ERROR CODE TO 8
F4
XCTL TO IGG0CLC5
YES
040 G2
040 F3
FOUND
G2
IS ENTRY A VCB POINTER
NO
G1
MOVE ENTRY DATA INTO USER'S AREA
J2
YES
H2
READ VCB INTO USER'S AREA USING BLDL
040 J2
DEQUE
J2
FREE MAIN STORAGE AND DEQUEUE
J2
K2
RETURN TO CALLER VIA SVC 3

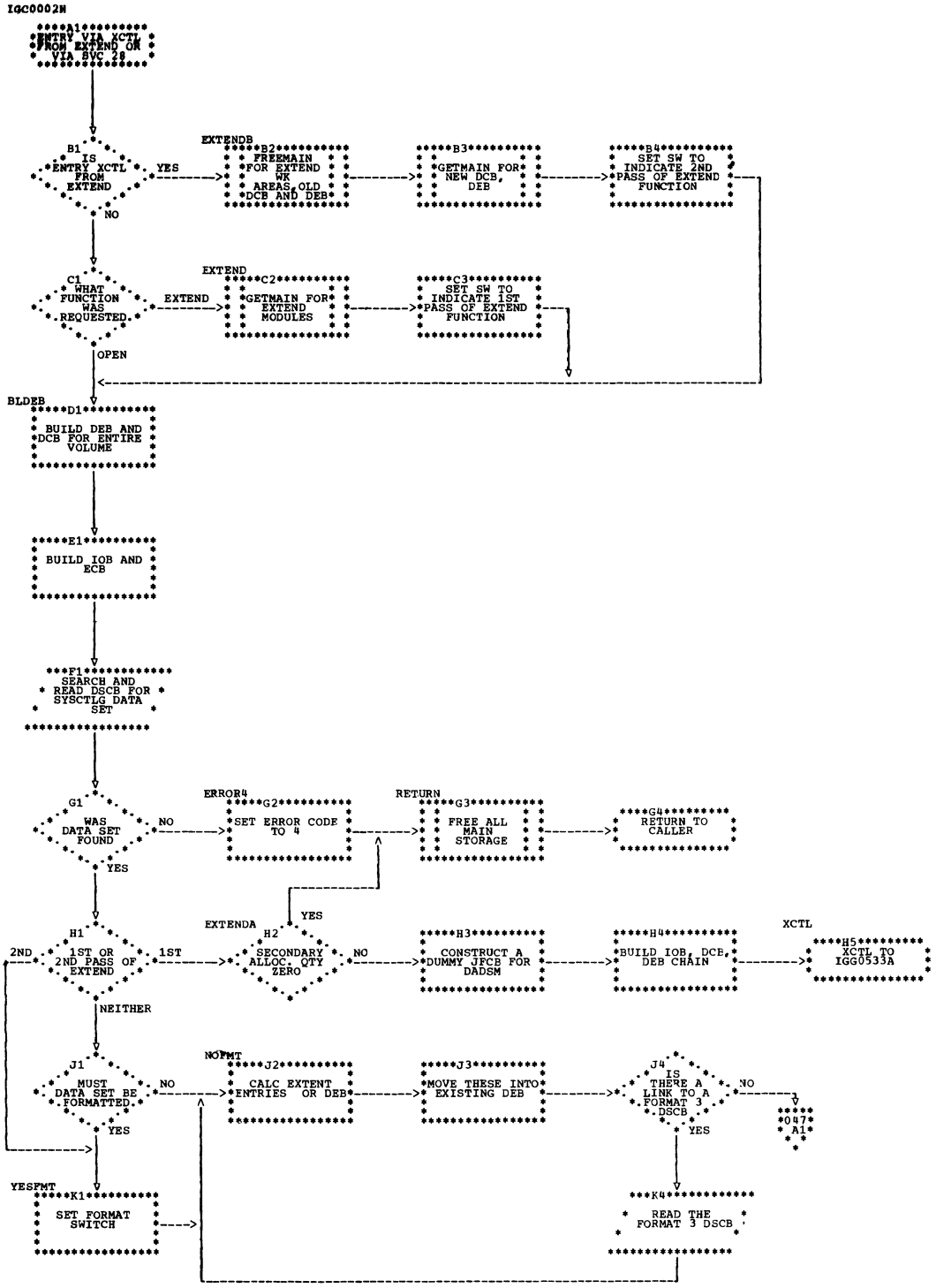

041
A1
A1
CONVERT RELATIVE GEN. NO. TO BINARY
NXTBLK
B1
READ A BLK OF THE GENERATION INDEX
C1
EXAMINE NAME FIELD OF AN ENTRY IN INDEX
D1
IS IT 8X'FF'
YES
NO
D2
MORE BLOCKS IN INDEX
YES
NO
D3
IS GIVEN REL. NO. POSITIVE
NO
YES
040
F3
E1
IS GIVEN REL. GEN. NO. NEGATIVE
NO
YES
E2
COMPLEMENT GEN. NO. OF FOUND ENTRY
E3
SET UP DUMMY FOUND NAME OF G0000V00
F1
POINT TO NEXT ENTRY ADD 1 TO REL. GEN. NO.
F2
IS GIVEN REL. GEN. NO ZERO
NO
YES
F3
ADD GIVEN REL. VALUE TO FOUND GEN. NO.
G1
MORE ENTRIES IN BLOCK
YES
NO
G2
MOVE FOUND NAME, COMPLEMENTED, TO USER AREA
G3
MOVE TOTAL TO USER AREA AS GEN. NAME
040
F3
040
G2
040
G2

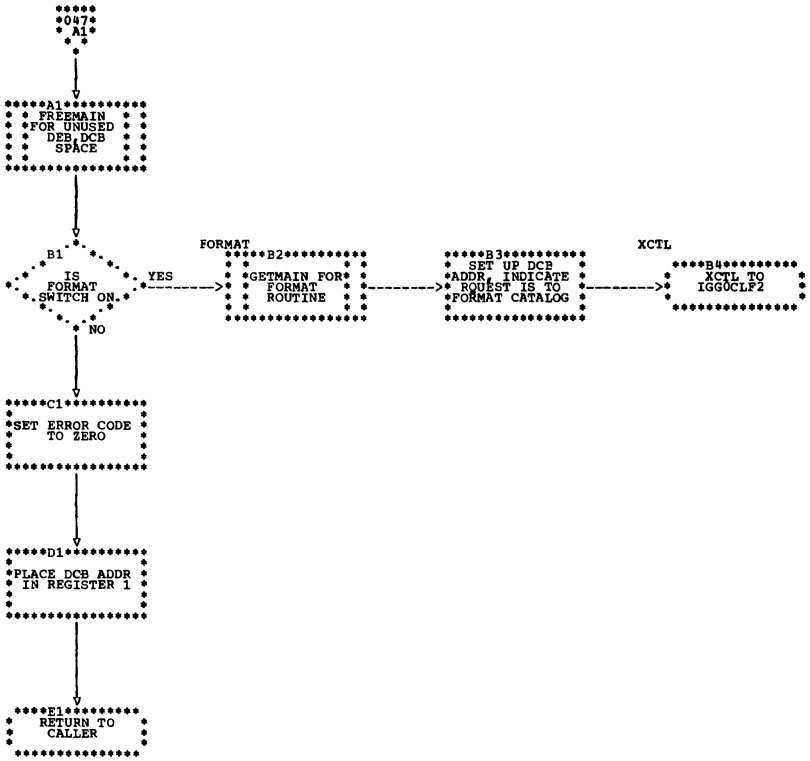

A2
EMPTY
A2
SET UP OUTPUT AREA WITH X'FF' KEY, IND CNTRL ENTRY, NAME OF NEW ENTRY
B2
MOVE 0 INDEX LINK ENTRY TO OUTPUT AREA
C2
READ THE OLD INDEX
D2
D2
WRITE CONTENTS OF OUTPUT AREA
E2
SCAN FOR LAST ENTRY IN BLK JUST READ
F2
DOES NAME FIELD CONTAIN 8X'FF'
NO
F3
INCREMENT TTR TO NEXT SEQUENTIAL BLOCK
H3
YES
G2
IS CHAIN POINTER ZERO
YES
G1
READ VOL INDEX USING BLDL
NO
G3
TTR OF OLD INDEX LT 1ST AVAIL HOLE
YES
G4
UPDATE VOLUME INDEX CONTROL BLOCK
NO
H3
H3
READ NEXT BLK IN CHAIN USING BLDL
H1
UPDATE VOL INDEX CONTROL BLK IF NECESSARY
J1
REWRITE IT
J2
SET ERROR CODE TO ZERO
J3
SET KEY OF OUTPUT AREA TO ZEROS
D2
J2

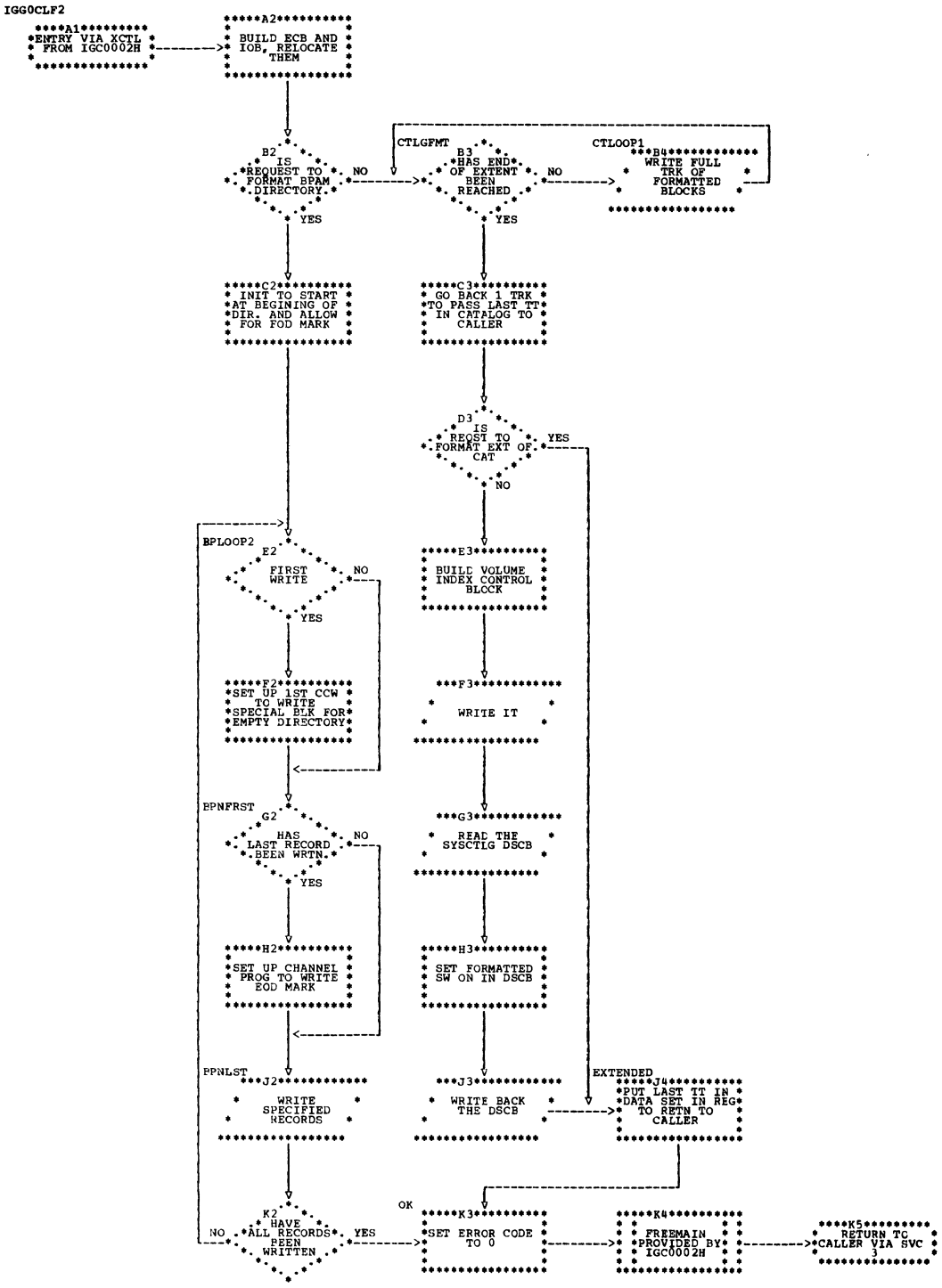

IGG0CLC5
A1
ENTRY VIA XCTL FROM IGG0CLC4
A2
IS REQUEST CATALOG
YES
NO
SEARCH
A3
WAS NAME FOUND BY BLDL
NO
D3
YES
ERR08
B2
WAS NAME FOUND BY BLDL
NO
YES
B3
SET ERROR CODE TO 8
B4
FREEMAIN FOR ALL USED DEQ ON ALL RESOURCES
B5
RETURN TO CALLER VIA SVC 3
C2
IS REQUEST TO UNCATALOG
NO
YES
D3
D2
SUBTRACT 1 FROM GENERATION COUNT
D3
SEARCH FOR NAME IGNORING VERSION NO.
D4
WAS NAME FOUND
NO
YES
D5
IS INDEX FULL
YES
NO
044 A1
E2
WRITE UPDATED BLOCK
E4
SET FUNCTION TO RECAT
E5
ADD 1 TO GENERATION COUNT
WRTVCB
F2
WAS A VCB POINTER FOUND
YES
NO
F1
REPETERE 044E3
FREE ALL VCBS FOR THIS NAME
F5
WRITE UPDATED INDEX BLOCK
RECAT
G2
IS FUNCTION RECAT
NO
YES
G5
BLDENTRY 044F2
BUILD NEW ENTRY
043 H2
H2
BLDENTRY 044F2
BUILD NEW ENTRY
RDIX
J2
READ BLOCK TO BE UPDATED
K2
XCTL TO IGG0CLC3


044
A1
SAVFLG
A1
IS EMPTY SW ON
YES
NO
EMPTY
A2
RESET GENERATION COUNT TO ONE
A3
WRITE UPDATED BLOCK
B1
FIND OLDEST ENTRY IN INDEX
B2
SET ERROR CODE TO ZERO
B3
SCRINIT 044D4
SCRATCH ANY OLD VCBS
LASTENT
C1
NEW ENTRY OLDER THAN OLDEST
YES
NO
C3
BLDENTRY 044F2
BUILD NEW ENTRY
D1
SCRINIT 044D4
SCRATCH ANY VCBS OF OLDEST ENTRY
FREEMAIN
D2
DEQUEUE FREEMAIN FOR ALL STORAGE USED
D3
XCTL TO IGG0CLC4 TO EMPTY INDEX
SCRINIT
D4
ENTER VIA BAL
SCRATCH
D5
ENTER VIA BAL
E1
DELETE LAST ENTRY IN INDEX
E2
RETURN TO CALLER VIA SVC 3
REPETERE
E3
ENTER VIA BAL
E4
WAS VCB POINTER FOUND
NO
YES
E5
WAS SCRATCH OPTION SPECIFIED.
NO
YES
F1
WRITE THE UPDATED BLOCK
043
H2
BLDENTRY
F2
ENTER VIA BAL
VCBPTR
F4
READ VCB
F5
SCRATCH DATA SET WITH SVC 29
G2
OVER 5 VOLS IN DATA SET
YES
NO
G3
BUILD APPROPRIATE NUMBER OF VCBS
G4
FREE IT BY WRITING A ZERO KEY
G5
RETURN VIA BR
H2
BUILD NEW DATA SET PTR OR VCB PTR ENTRY
H4
SCRATCH 044D5
SCRATCH DATA SET IF REQ'D
J2
RETURN VIA BR
J4
IS THERE ANOTHER VCB
YES
NO
J5
RETURN VIA BR

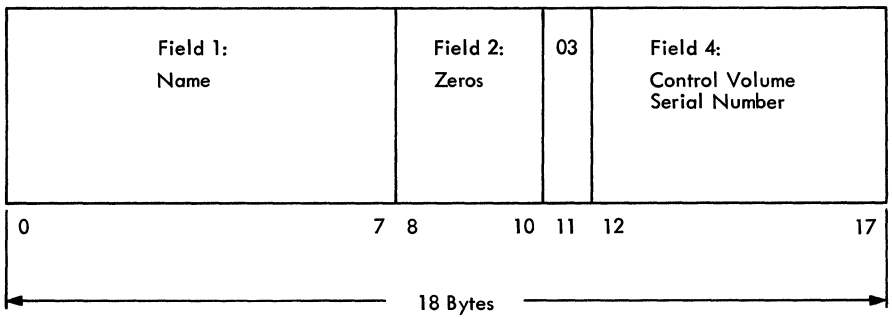

IGG0CLC6
A1
XCTL FROM IGG0CLC2
B1
IS FUNCTION LOCATE
NO
YES
NOTLOC
B2
FREEMAIN FOR INDEX CATALOG WORK AREA
FREE
C1
FREEMAIN FOR LOCATE WORK AREA
C2
DEQUEUE ON SYSCTLG
D2
EXIT TO USER VIA SVC 3


IGC0002H
A1
ENTRY VIA XCTL FROM EXTEND OR VIA SVC 26
B1
IS ENTRY XCTL FROM EXTEND
YES
NO
EXTENDB
B2
FREEMAIN FOR EXTEND WK AREAS, OLD DCB AND DEB
B3
GETMAIN FOR NEW DCB, DEB
B4
SET SW TO INDICATE 2ND PASS OF EXTEND FUNCTION
C1
WHAT FUNCTION WAS REQUESTED.
EXTEND
OPEN
EXTEND
C2
GETMAIN FOR EXTEND MODULES
C3
SET SW TO INDICATE 1ST PASS OF EXTEND FUNCTION
BLDEB
D1
BUILD DEB AND DCB FOR ENTIRE VOLUME
E1
BUILD IOB AND ECB
F1
SEARCH AND READ DSCB FOR SYSCTLG DATA SET
G1
WAS DATA SET FOUND
NO
YES
ERROR4
G2
SET ERROR CODE TO 4
RETURN
G3
FREE ALL MAIN STORAGE
G4
RETURN TO CALLER
H1
1ST OR 2ND PASS OF EXTEND
2ND
1ST
NEITHER
EXTENDA
H2
SECONDARY ALLOC. QTY ZERO
YES
NO
H3
CONSTRUCT A DUMMY JFCB FOR DADSM
H4
BUILD IOB, DCB, DEB CHAIN
XCTL
H5
XCTL TO IGG0533A
J1
MUST DATA SET BE FORMATTED.
NO
YES
NOFMT
J2
CALC EXTENT ENTRIES OR DEB
J3
MOVE THESE INTO EXISTING DEB
J4
IS THERE A LINK TO A FORMAT 3 DSCB
NO
YES
0A7
A1
YESFMT
K1
SET FORMAT SWITCH
K4
READ THE FORMAT 3 DSCB


047
A1
A1
FREEMAIN FOR UNUSED DEB,DCB SPACE
B1
IS FORMAT SWITCH ON
YES
NO
FORMAT
B2
GETMAIN FOR FORMAT ROUTINE
B3
SET UP DCB ADDR, INDICATE ROUEST IS TO FORMAT CATALOG
XCTL
B4
XCTL TO IGG0CLF2
C1
SET ERROR CODE TO ZERO
D1
PLACE DCB ADDR IN REGISTER 1
E1
RETURN TO CALLER

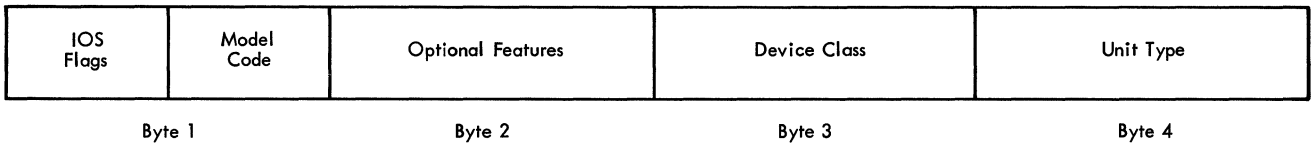

IGG0CLF2
A1
ENTRY VIA XCTL FROM IGC0002H
A2
BUILD ECB AND IOB, RELOCATE THEM
B2
IS REQUEST TO FORMAT BPAM DIRECTORY
NO
YES
CTLGFMT
B3
HAS END OF EXTENT BEEN REACHED
NO
YES
CTLOOP1
B4
WRITE FULL TRK OF FORMATTED BLOCKS
C2
INIT TO START AT BEGINING OF DIR. AND ALLOW FOR EOD MARK
C3
GO BACK 1 TRK TO PASS LAST TT IN CATALOG TO CALLER
D3
IS REQST TO FORMAT EXT OF CAT
YES
NO
BPLOOP2
E2
FIRST WRITE
NO
YES
E3
BUILD VOLUME INDEX CONTROL BLOCK
F2
SET UP 1ST CCW TO WRITE SPECIAL BLK FOR EMPTY DIRECTORY
F3
WRITE IT
BPNFRST
G2
HAS LAST RECORD BEEN WRTN.
NO
YES
G3
READ THE SYSCTLG DSCB
H2
SET UP CHANNEL PROG TO WRITE EOD MARK
H3
SET FORMATTED SW ON IN DSCB
PPNLST
J2
WRITE SPECIFIED RECORDS
J3
WRITE BACK THE DSCB
EXTENDED
J4
PUT LAST TT IN DATA SET IN REG TO RETN TO CALLER
K2
HAVE ALL RECORDS BEEN WRITTEN
NO
YES
OK
K3
SET ERROR CODE TO 0
K4
FREEMAIN PROVIDED BY IGC0002H
K5
RETURN TO CALLER VIA SVC 3

Before Release 17, the control volume pointer entry had no device type code field. Since some control volumes may still contain the old entry, and since the routines still check for it, its format is given here.



APPENDIX C: DEVICE TYPE FIELD

The device code portion of Data Set Pointer Entries, Volume Control Blocks, and Control Volume Pointer Entries is identical to the UCBTYP field of the Unit Control Block. This description is included here for easy reference.



For a complete description of the fields shown above, please refer to IBM System/360 Operating System System Control Blocks, Form C28-6628. A brief description of some of the fields appears below.

Device Class: (Byte 3; values are in hex)

X'80' Magnetic Tape
X'20' Direct Access
X'08' Unit Record
X'10' Graphics
X'40' Communications

When Byte 3 indicates direct access, byte 4 indicates the specific device as follows:

X'01' 2311 Disk Storage Drive
X'02' 2301 Parallel Drum
X'03' 2303 Serial Drum
X'04' 2302 Disk Storage
X'05' 2321 Data Cell Drive
X'08' 2314 Disk Storage Facility

Where more than one page reference is given, the major reference is first.

READER'S COMMENT FORM

IBM System/360 Operating System
Catalog Management
Program Logic Manual

Form Y28-6606-1

- Your comments, accompanied by answers to the following questions, help us produce better publications for your use. If your answer to a question is "No" or requires qualification, please explain in the space provided below. Comments and suggestions become the property of IBM.

	Yes	No
• Does this publication meet your needs?	☐	☐
• Did you find the material:		
Easy to read and understand?	☐	☐
Organized for convenient use?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Written for your technical level?	☐	☐

- What is your occupation? _____
- How do you use this publication?

As an introduction to the subject?	☐	As an instructor in a class?	☐
For advanced knowledge of the subject?	☐	As a student in a class?	☐
For information about operating procedures?	☐	As a reference manual?	☐

Other _____

- Please give specific page and line references with your comments when appropriate.

COMMENTS

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

YOUR COMMENTS, PLEASE...

This publication is one of a series which serves as reference sources for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

fold fold

FIRST CLASS
PERMIT NO. 2078
SAN JOSE, CALIF.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY . . .

IBM Corporation
Monterey & Cottle Rds.
San Jose, California
95114

Attention: Programming Publications, Dept. D78

fold fold

IBM S/360 Printed in U.S.A. Y28-6606-1

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

IBM S/360 Printed in U.S.A. Y28-6606-1

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]