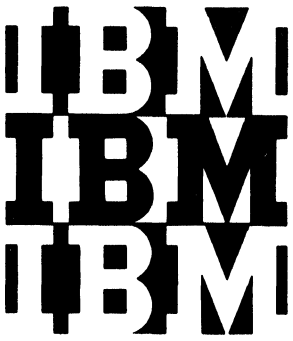


File No. S360-36
Order No. GY28-6745-0

Program Logic

IBM System/360 Operating System:

Time Sharing Option

Catalog Management

Program Logic Manual

Program Number 360S-DM-508

This publication provides customer engineers and other technical personnel with information describing the internal organization and logic of the catalog management routines that are used with the Time Sharing Option has been selected at system generation time. These routines provide the facility of locating data sets when only data set names are specified.

This manual is based on the IBM System/360 Operating System: Catalog Management, Program Logic Manual, GY28-6606. It should be used in place of the above manual only if the Time Sharing Option has been specified at system generation time.

Information in this publication for TSO is for planning purposes until that item is available.

First Edition (March, 1971)

This edition applies to release 20.1, of IBM System/360 Operating System, and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM systems, refer to the latest IBM System/360 SRL Newsletter, Order No. GN20-0360, for the editions that are applicable and current.

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

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Systems Publications, Department D58, PO Box 390, Poughkeepsie, N. Y. 12602

Preface

This publication provides customer engineers and other technical personnel with information describing the internal organization and logic of the catalog management routines that are used when the Time Sharing Option has been selected at system generation time.

This manual is based on the IBM System/360 Operating System Catalog Management Program Logic Manual, GY28-6606. It should be used in place of the above manual only if the Time Sharing Option has been specified at system generation time.

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 management routines.

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 seven 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 the catalog management routines but which may still exist in some installations.

Contents

Illustrations

Figures

Charts

Introduction

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.

The catalog management routines can be used to build or delete a single index or a whole index structure. To catalog a data set called A.B.C, for example, the user may either first create index A, then index A.B, and then catalog A.B.C, or request that catalog management create any missing index levels needed to catalog 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 Structure

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, Form 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 may start 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 CVOL whose serial number is in the control pointer entry. The caller of the catalog management routine may specify what CVOL is used for the search.

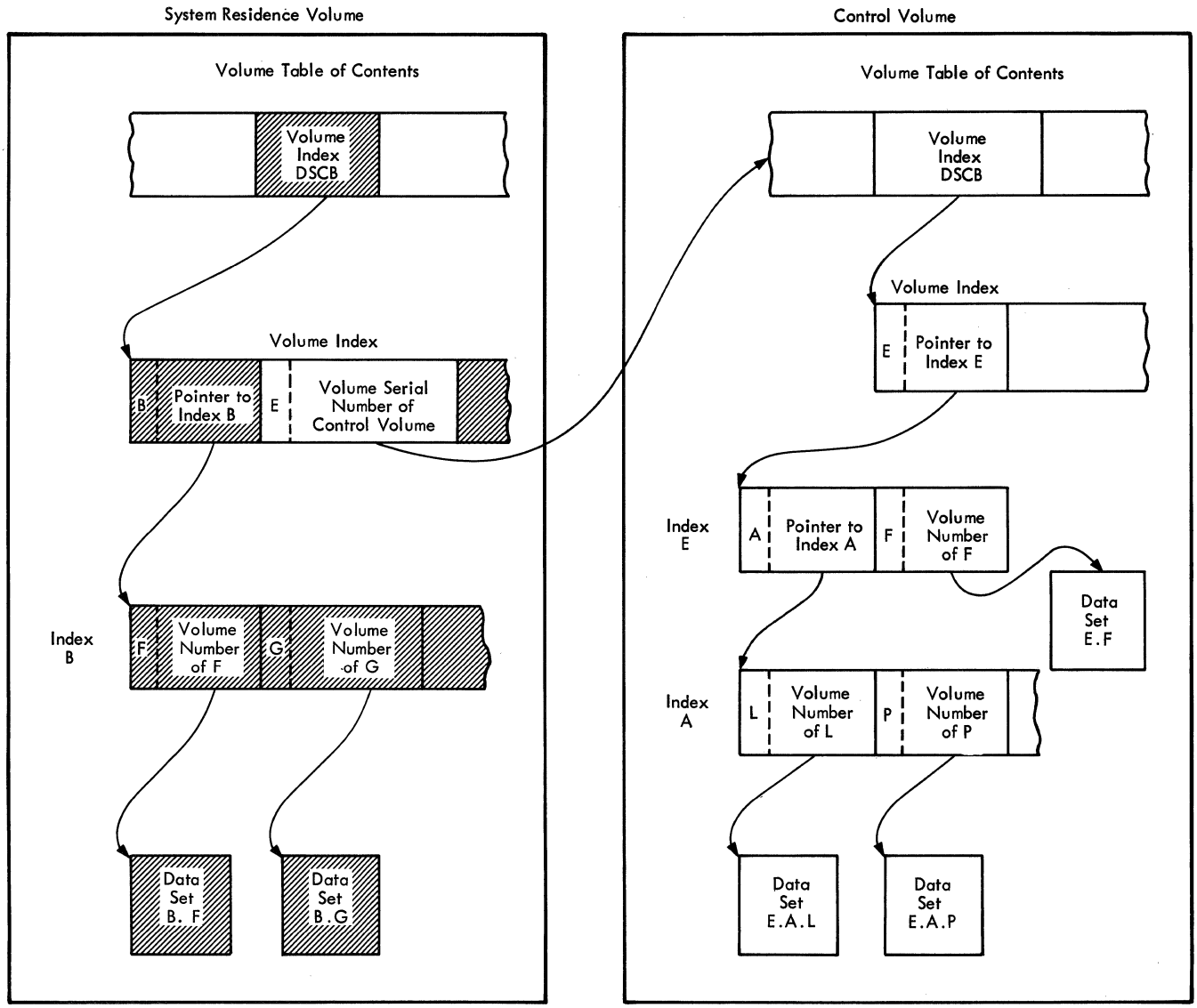


Figure 1. A Control Volume Connected to the System Residence Volume

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, the job scheduler, and TSO, 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.

TSO dynamic allocation locates old data sets and catalogs new data sets. TSO command processors alsc call the catalog management routines to manipulate the catalog.

FUNCTION		ABBREVIATION*
LOCATE	a data set by name a block in the catalog by TTR	NAME BLOCK
BUILD	a normal index a generation index an alias to a high-level index	BLDX BLDG BLDA
DELETE	an index an alias	DLTX DLTA
CONNECT	two control volumes	LINKX
DISCONNECT	two control volumes	DRPX
CATALOG	a data set a data set and build index structure	CATALOG CATBX
UNCATALOG	a data set a data set and delete index structure	UNCAT UCATBX
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, its logical organization, and the way in which its keys are used.

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 the catalog blocks may contain any value from X'0000000000000000' up to, and including, X'FFFFFFFFFFFFFFFF'. A nonzero key indicates a block containing information, while a zero key denotes a block that 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,

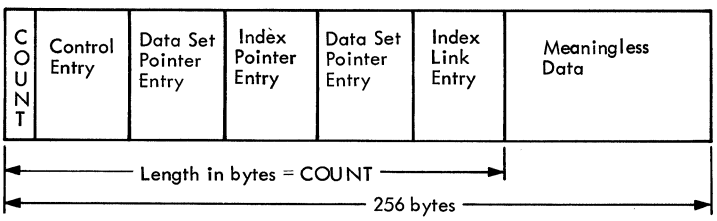


Figure 3. Typical Physical Block in the Catalog

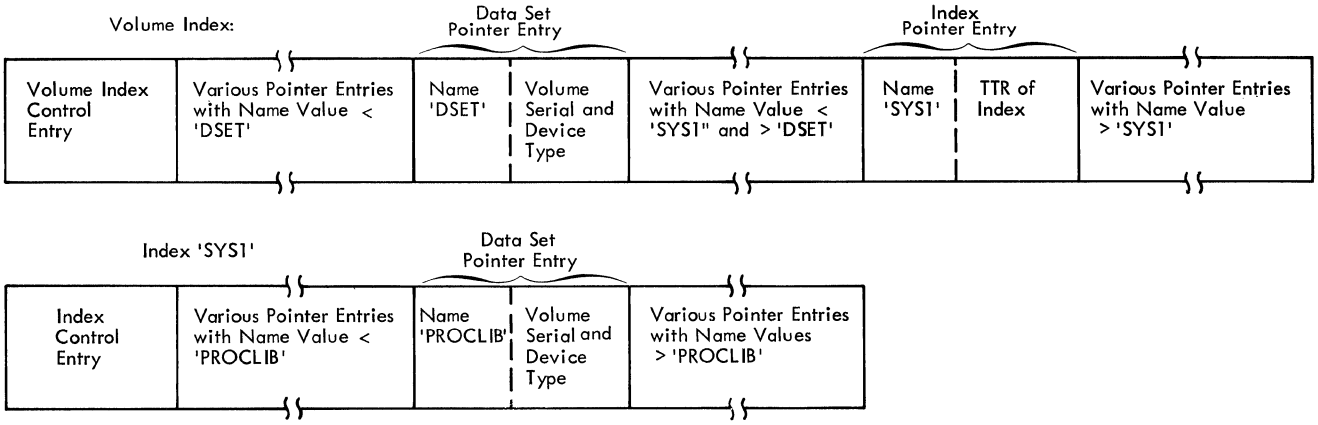


Figure 4. Logical Organization of the Catalog: Normal Index Structure

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.

CHAINING OF 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, but unused, even when the several blocks of an index are contiguous (See Figure 6).

USE OF KEYS

The keys of catalog blocks are designed to allow hardware to perform much of the search with the "search key high or equal" command. The name field of the desired entry is always used as the search argument for this command. Thus, the search is stopped and a block is read into main storage whenever a key with this value or higher is encountered.

The key of a block in the catalog has the value of the name field of the last entry in the block if the next block of the index is not contiguous to this block. This key will always be X'FF ... FF', because the last entry in any block is an index link entry, and the name field of an index link entry is X'FF ... FF'.

The key of a block in the catalog has the value of the name field of the next-to-last entry in the block if the next block in the index is contiguous with this block.

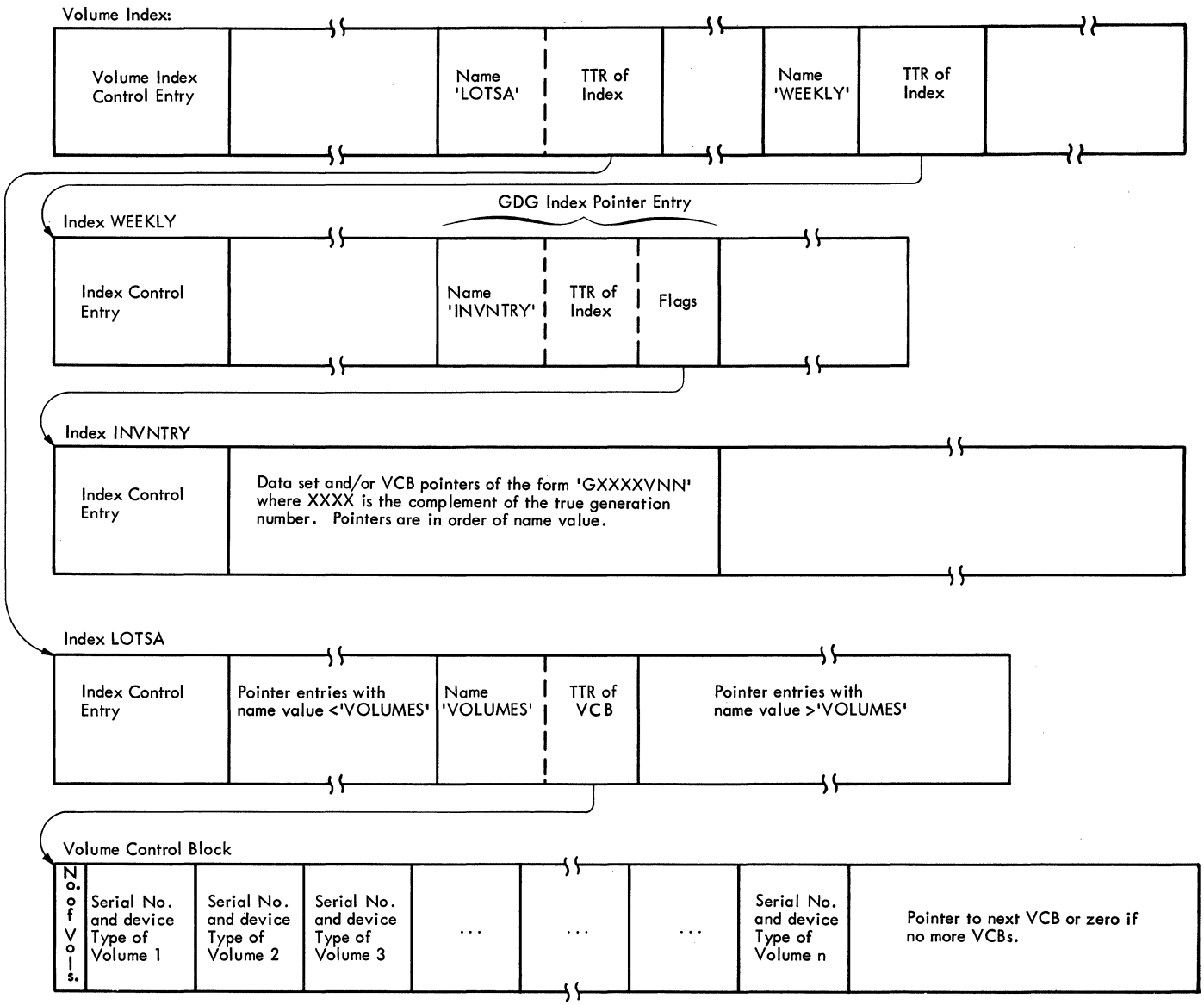


Figure 5. Logical Organization of the Catalog: Generation Indexes and Volume Control Blocks

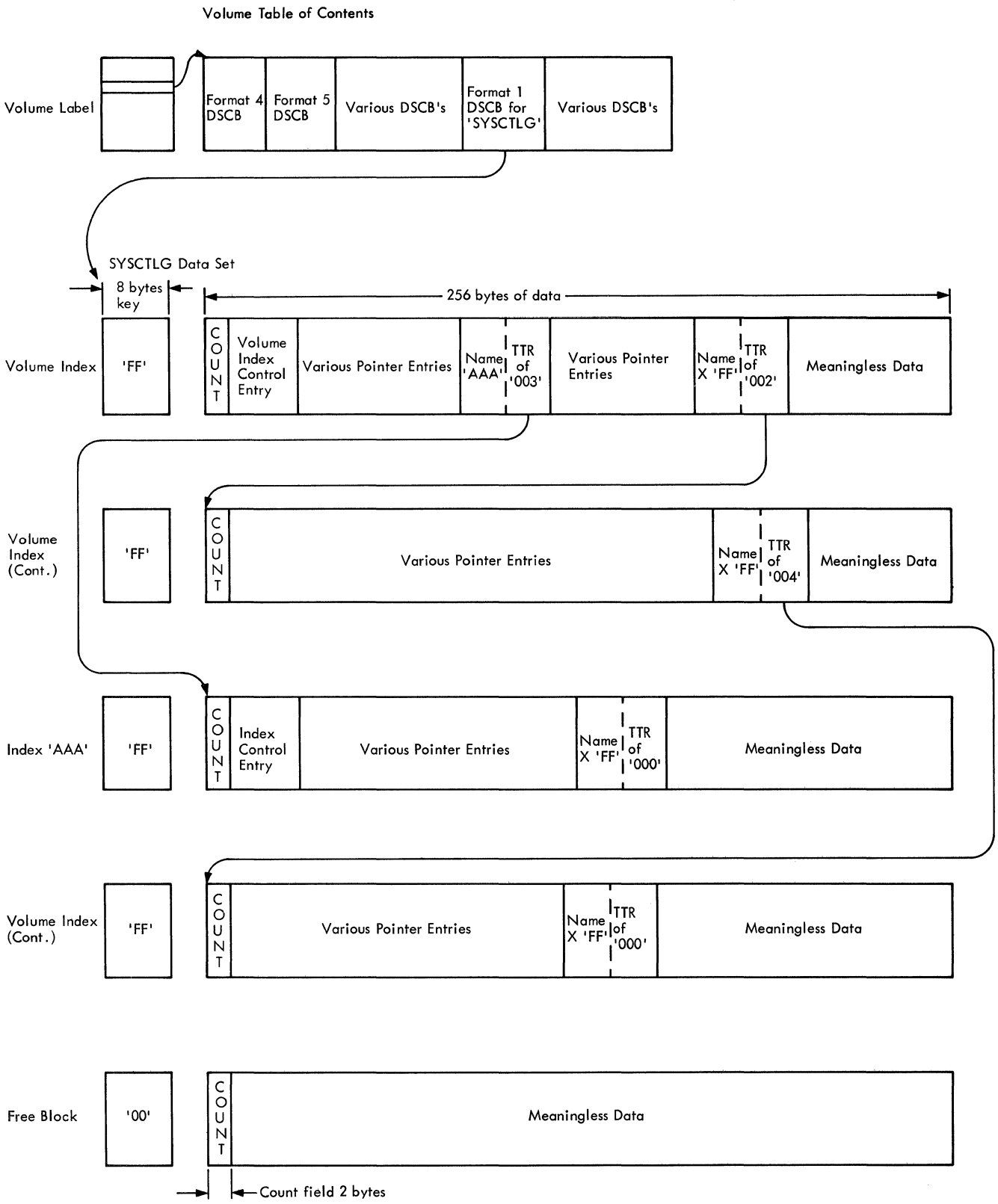


Figure 6. Physical Organization of the Catalog

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 catalog management routines operate in multiprogramming and multiprocessing environments, they must protect the part of the catalog they are manipulating from simultaneous accessing and modification by other programs and CPU's. This protection is afforded by the use of the RESERVE and ENQ supervisor functions.

The RESERVE function protects the device containing the control volume being searched or modified from access by another CPU in a multiprocessing environment.

The ENQ function protects the part of the catalog being manipulated from access by other programs in a multiprogramming environment.

An ENQ function can be either shared or exclusive. A shared ENQ for a catalog resource allows simultaneous access to the resource by other shared ENQ requests. An exclusive ENQ for a catalog resource calls for exclusive control of that resource.

To provide complete protection of the catalog with minimum accessing delays, the catalog resources are divided into three different types:

- A volume index resource represents a complete CVOL. Control of such a resource allows for the accessing of high level names, aliases, and CVOL pointers.
- A high-level name resource represents the complete index tree structure associated with that high-level name even though the tree structure may involve several control volumes.
- A volume index control entry (VICE) resource represents the free space in the catalog and thus the ability to modify the catalog data set.

The catalog management routines issue ENQ requests only for the resources necessary to accomplish a particular function leaving the remaining resources open to access by other users.

For example, to modify a low level index, the routines obtain exclusive control of a high-level name and the VICE, while to perform a locate function, the routines request shared control of a high-level name and (temporarily) the volume index. By separating catalog resources, both operations can be performed concurrently on the same control volume.

Since these routines are reenterable and cannot store within themselves, they obtain a storage area in the user's region by issuing a GETMAIN macro instruction. The area is freed when the catalog routines terminate. 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, the locate routine 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, the locate routine 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 main storage, dequeues, and passes a nonzero return code to the caller.

For example, assume that a user wished to catalog data set 'A.B.C'. The locate routine 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 the locate routine 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 also similar to BLDX, 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 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 Direct Access Device Space Management (DADSM) by issuing an SVC 29. (For a discussion of the Delete routine see IBM System/360 Operating System: Direct Access Device Space Management, Form 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 the 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, and 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 short enough so 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.

CATBX and UCATDX Functions

The CATBX and UCATDX functions are similar to the CATALOG and UNCAT functions. The difference lies with the building and deleting of the index tree structure associated with the cataloged data set. The CATBX function generates any missing index levels needed to catalog the data set, and the UCATDX function deletes any index levels that exist only as qualifications of the data set name in question.

In both cases, if only one level of index is involved, the functions are performed as in CATALOG or UNCAT.

If the function is CATBX and more than one level of index is missing, a BLDX function is performed to insert the highest level missing index entry into the existing catalog structure. Free blocks for each remaining index level are then obtained and chained together, creating an index substructure which is then chained to the entry created by BLDX.

If the function is UCATDX and more than one level of index becomes superfluous when the data set is removed from the catalog, a DLTX function is performed on the highest level index entry to be deleted and the blocks containing all lower level index entries are freed. The highest level index entry eligible for deletion is determined while the LOCATE function is being performed. UCATDX deletes all superfluous index levels except the volume index.

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 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.

Program Organization

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 SVC 26, which gives control to module IGC0002F.

It validates the user's parameter list, gets main storage for an open work area, and 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 is specified. The UCB address is then passed to the catalog open routine (IGC0002H) which is entered with an SVC 28.

Any one of three diagnosed error conditions can cause a return to the user (via SVC 3) with the appropriate error code:

- Invalid user parameter list.
- Control volume UCB not found.
- Open error.

After a successful open, the routine issues a GETMAIN macro instruction to obtain storage for work areas, determines the specified function from user parameters, and then either transfers control to IGG0CLC1 if the function is locate by block or uses the IECPBLDL routine to search for the high-level name specified in the user's parameter area.

If the high-level name search returns a CVOL pointer entry, the new CVOL information is stored in the user's parameter list, and processing is resumed with the UCB search routine after work areas are freed.

Once the correct control volume is found, the routine issues a RESERVE request for the CVOL and passes control to IGG0CLC1.

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.

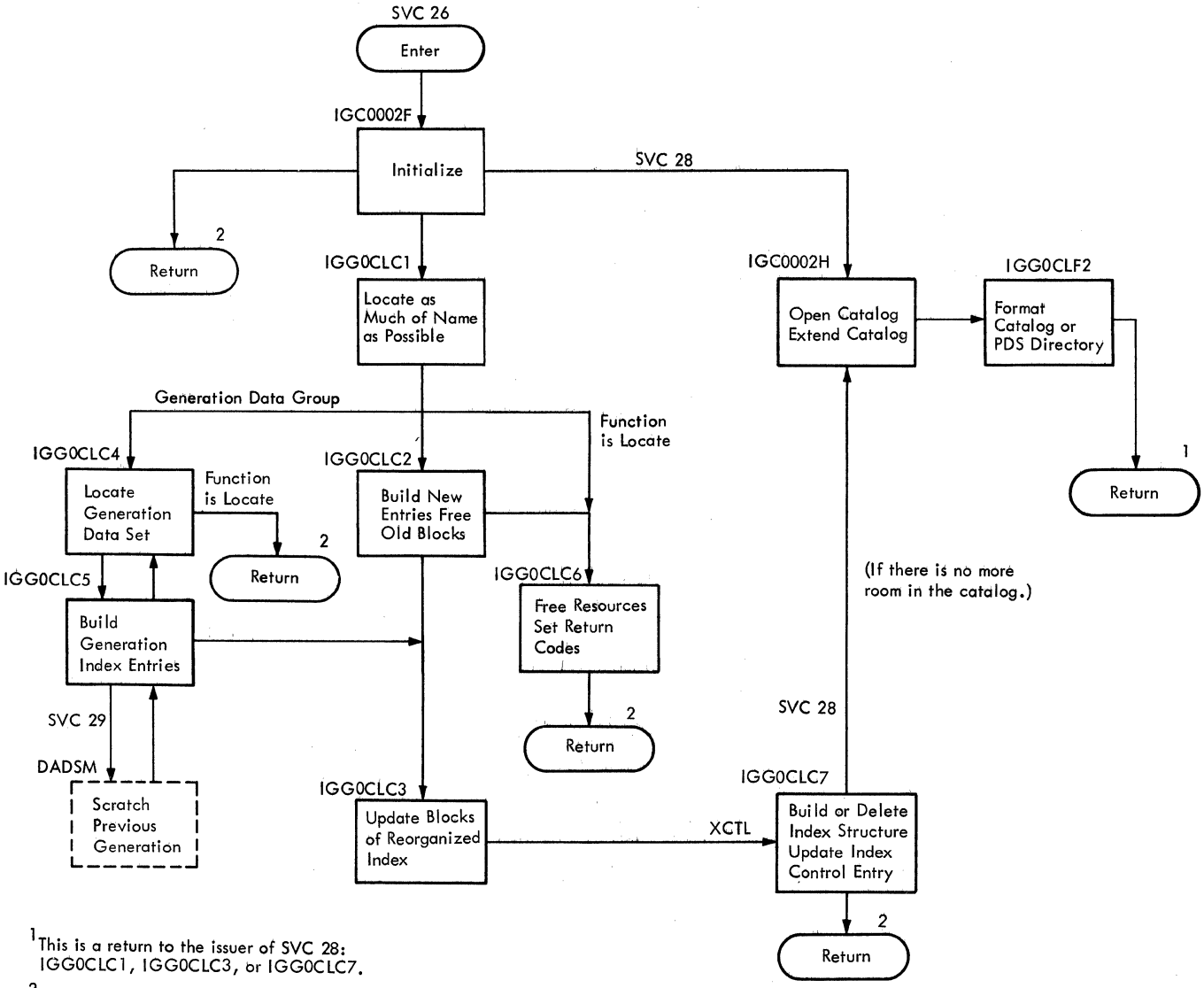
If the request is to search for a name or to index or catalog a name, IGG0CLC1 isolates the first level of the name. It uses BLDL to search the volume index for this simple name and analyzes what type of pointer is associated with it. Several different things can happen, depending on what pointer type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

For CATBX and UCATDX functions, IGG0CLC1 performs initialization by either preparing for a BLDX for the first index entry to be built in a CATBX or preparing for a DLTX for the first index entry to be deleted in a UCATDX and saving the TTR link entry of the deleted index.

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, to IGG0CLC4 to search generation indexes, or to IGG0CLC6 for error



[1]This is a return to the issuer of SVC 28: IGG0CLC1, IGG0CLC3, or IGG0CLC7.

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

Figure 8. Catalog Module Flow

processing or successful completion of a LOCATE.

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 last index searched. It will probably have to be changed.

Index/Catalog, Normal Structure: Modules IGG0CLC2, IGG0CLC3, IGG0CLC6, and IGG0CLC7

These modules together update a normal index structure. They build new indexes and insert pointers to them in old indexes, they delete old pointers and free the associated blocks, they build aliases, and they update control entries.

The catalog is updated in two phases. Phase one, done by IGG0CLC2 (or, in the case of generation indexes, IGG0CLC4 and IGG0CLC5) builds new indexes and pointer entries and deletes old blocks. Phase two, done by IGG0CLC3 and IGG0CLC7, reorganizes the index into which the new pointer will be inserted, or from which a pointer will be deleted. IGG0CLC7 also has the ability to build and delete index structures.

IGG0CLC2

This module constructs all new entries except entries in a generation index and index structures built by IGG0CLC7. It checks to be sure the existing catalog structure is consistent with the new entry and returns to the caller with an error code if it is not. It always receives control from IGG0CLC1 and passes control to IGG0CLC3.

First the module determines from the user's parameters whether a new entry is needed. If it is, the module determines what type of entry, and whether this entry is consistent with what the locate routine found. If, for example, the desired function were catalog and the locate routine had found an entry for every level of the name, this module would set an error code and return. The same name may not be cataloged twice.

The module then determines whether any blocks have to be freed. If the function was RECAT, for example, and the old entry was a VCB pointer, the old VCB would be freed before the new pointer entry was created. To free a block, the module writes zeros as its key and updates the volume index control entry.

If no new entry is to be created, the module only frees unused blocks. This would be the case if the requested function were to delete an index or uncatalog a data set, for example.

IGG0CLC2 also writes certain new blocks when they are required. When a catalog request necessitates VCBs, this module finds available blocks and writes the VCBs in them. If the request is to build an index (either generation or normal), the module builds an empty index and notes its location for the next module.

IGG0CLC6

This module receives control from either IGG0CLC1 or IGG0CLC2. IGG0CLC6 is entered either upon the successful completion of a LOCATE function (from IGG0CLC1) or an error condition (from either module). In either case, IGG0CLC6 frees the main storage acquired by all previous modules, dequeues all resources, and returns the appropriate information to the user.

IGG0CLC3

This module adds or deletes a pointer entry in an index and rewrites the index in such a way that the entries maintain their alphameric order. It receives control either from IGG0CLC2, which constructs new entries for normal indexes, or from IGG0CLC5, which constructs new entries for generation indexes.

First, the module looks at the TTR of the index to be updated and the entry to be added or deleted. The name of this entry becomes the search argument for determining where to update the index. When blocks of the index are contiguous, the search is rapid because each key field of the blocks in the chain contains the name of the highest alphamerically ordered significant entry in the block. The hardware compares the search argument with the key fields of the blocks in the chain, starting with the lowest. When the comparison shows that the search argument is higher than the key field, the search continues on the key

field of the next contiguous block. When the key field is greater than or equal to the search argument the block is read into main storage.

With the block in main storage, the module goes through it entry by entry, each time comparing the name of the current entry with the search argument entry name. When it finds an entry with a name greater than or equal to the search argument name, it performs the update.

Key fields with hexadecimal F's denote index blocks that are either at the end of the index or at the end of a contiguous chain within a single index. If the key denotes the end of a chain, then the index link entry in its block will point to the next block of the index. The search channel program is restarted at the address specified in the pointer and the search is continued as before. If this block is the end of the index, however, the link entry contains zeros, and the module makes the update in this block.

IGG0CLC3 checks the number of bytes in its output buffer continuously, and when the end of a 256-byte block is approaching it builds an index link entry.

IGG0CLC7

This module always receives control from IGG0CLC3 via an XCTL macro instruction. It writes the last block of the updated index, updates and writes control entries, frees the main storage acquired by IGC0002F, dequeues the system resources, and returns to the user with a zero (no error) completion code in register 15.

When IGG0CLC7 receives control, it puts an index link entry with a TTR field of zeros in the last index block, and calculates the number of bytes remaining in the block. If a block has been freed during the updating operation, the module fills its key field with zeros. However, if the index expands into an additional block, the module fills the key field of the new block with hexadecimal F's. In either case, the module updates index control entries and volume index control entries as necessary to record the availability and location of free index blocks. Then it writes the updated entries with the updated index block.

At this point, IGG0CLC7 determines whether CATBX or UCATDX processing is called for. For CATBX processing, an index structure is constructed as follows:

1. A data set entry or VCB chain is created.
2. An index substructure is built beginning with the lowest index level. One block is built at a time and chained to the previous block until all missing index levels are filled.
3. The index substructure is chained to the empty index created by the BLDX part of CATBX processing and the index is rewritten to link the substructure to the catalog.

For UCATDX processing, IGG0CLC7 uses the TTR link entry saved during initialization for the DLTX part of UCATDX processing as a starting point to free all blocks in the index structure. Each block is read into main storage and rewritten with a zero key after its TTR link entry is saved. Each block in the index structure is freed until the complete structure is deleted.

When all the writing functions are complete, the module frees all the main storage and dequeues all the system resources used before returning to the calling routine via an SVC 3.

Catalog Protection

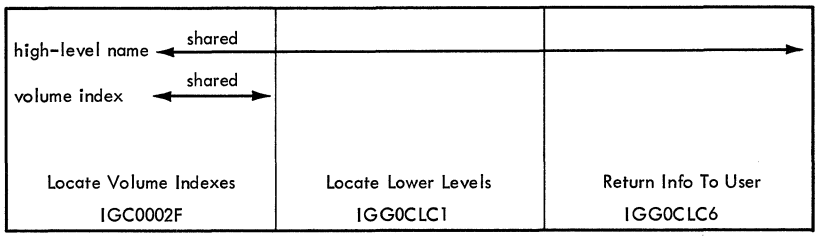
The ENQ supervisor function is used by several job management routines to achieve catalog protection. Figure 9 shows how catalog resources are enqueued and dequeued by the catalog management routines.

The ENQ macro instruction requires the specification of two names: a general name and a resource name. The catalog management routines use the following names for the indicated catalog resources:

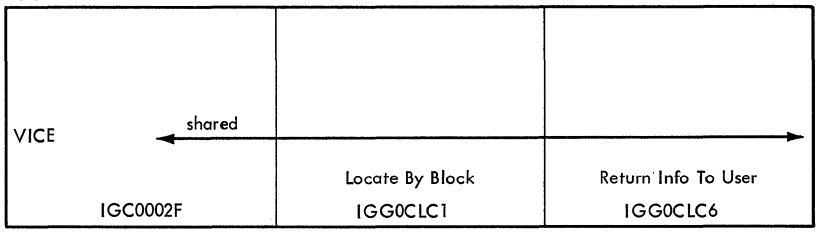
Resource	QNAME	RNAME
volume index	SYSCTLG	SYSCTLGb00ua
VICE	SYSCTLG	bbbbbbbb00ua
high-level name	SYSCTLG	name

Where ua is the two-byte address of the UCB of the CVOL being used, and name is the left-justified high-level name of the data set.

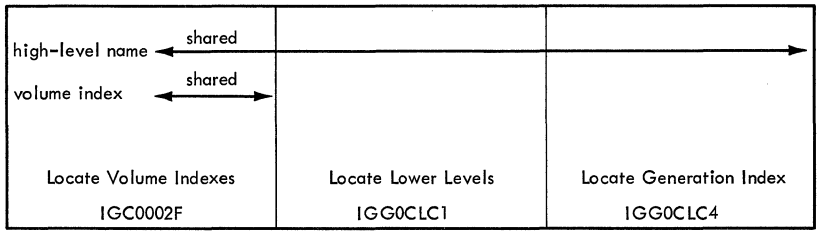
LOCATE



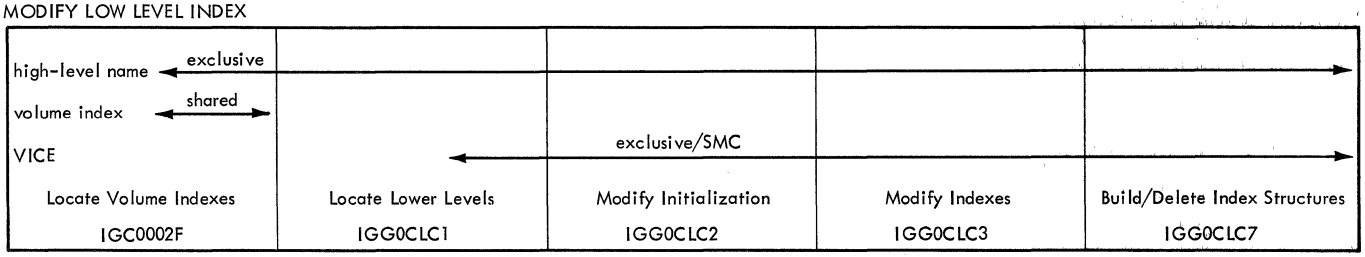
LOCATE BY BLOCK



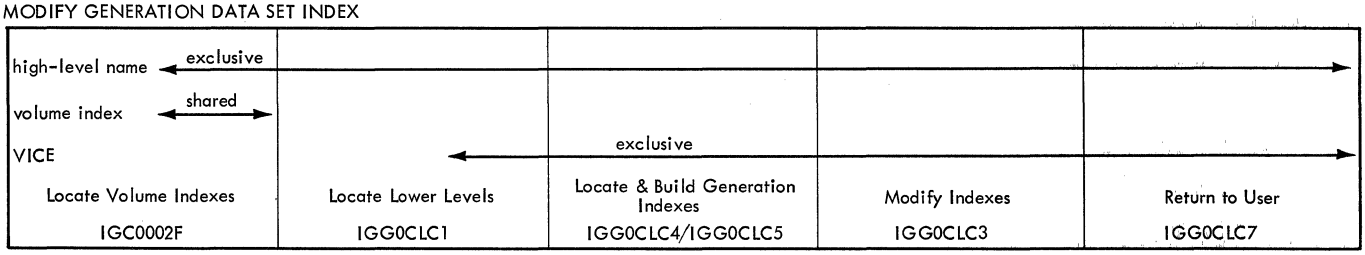
LOCATE GENERATION DATA SET INDEX



MODIFY LOW LEVEL INDEX



MODIFY GENERATION DATA SET INDEX



MODIFY VOLUME INDEX AND LOW LEVEL INDEX

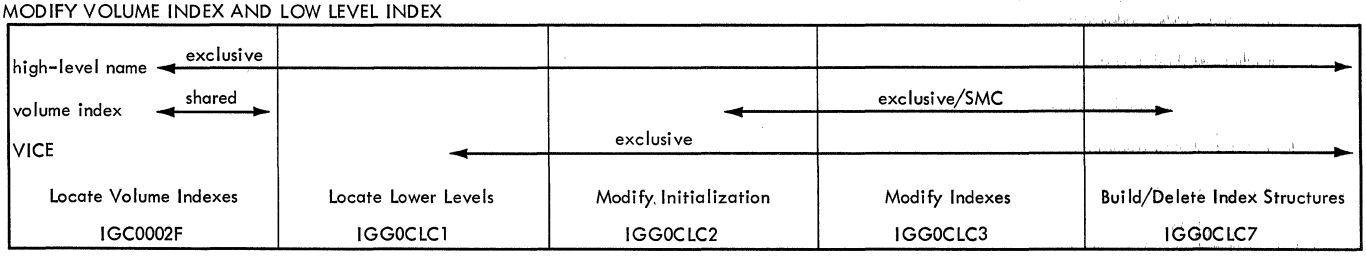


Figure 9. Use of ENQ and DEQ Functions

Locate Generations: Module IGG0CLC4

Generation data groups require significantly different locating and cataloging procedures from other data sets for two reasons:

(1) Generation data groups may be specified by relative generation number (as in GENR(+1)), in which case the absolute generation number must be calculated, and

(2) The absolute generation number is stored in the catalog in hexadecimal complement form, that is, generation G0001V00 would be stored as X'C7 0F 0F 0F 0E E5 F0 F0'. (Note that the version number and the characters 'G' and 'V' are not complemented.) In this way the most recent generation (the one with the highest absolute number) is always the first entry in the index after the index control entry.

In this manual, the term "absolute generation number" refers to the number as it is coded by the user and as it appears in the name field of a data set control block (DSCB). It does not refer to the number as it is stored in the catalog, in complement form.

Module IGG0CLC4 locates the lowest level of a generation name. When IGG0CLC1 finds a generation index pointer entry correlated with the next to last level of a name, it passes control to this module. It may also be entered from IGG0CLC5 when that module finds it must empty an index.

This module first checks to see whether entry is from IGG0CLC5 (empty request) or from IGG0CLC1 (normal locate path). If it was from IGG0CLC5, the module rewrites the generation index, this time with only the highest entry, and frees any blocks no longer needed by the shortened index.

If the path is a normal locate path (entry from IGG0CLC1), IGG0CLC4 checks the format of both relative generation numbers and absolute generation numbers and returns to the user with an error code of 20 if the format of the supplied name is not correct. If the name is in relative format, the only valid function is locate; if any other function has been specified, the module returns with an error code of 20.

If the name is in relative format, the module must calculate its absolute generation number. It does this by adding or subtracting the relative number given and the actual number of the first entry in the index. If the index is empty, the module sets up a dummy 'found' entry called 'G0000V00' as the basis for absolute generation number calculation. If the relative number is negative and exceeds the number of entries in the index, the module returns to the user with an error code of 8.

Once the relative generation number in the user's area has been replaced with the absolute generation number, the module proceeds as though the user had supplied the absolute number in the first place.

With the generation number in absolute format, the module uses BLDL to read the entry associated with the name. If the function is catalog, control is passed to IGG0CLC5 via the XCTL macro instruction. If the function is locate, the module checks BLDL's error code. If the name was found, the module moves the data into the user's area and checks to see if it must read a volume control block to complete the description of the data set. If it does, the volume control block is read into the user's area.

If BLDL cannot find the name, the module returns to the user with an error code.

Catalog Generations: Module IGG0CLC5

This module builds new entries for generation indexes, maintains generation index pointer entries by updating the generation count, and marks entries for deletion or data sets for deletion if the empty or delete option was specified when the generation index was created.

The module first checks the findings of IGG0CLC4 to be sure the current structure of the index is compatible with the function requested. If the requested function is catalog, for example, and the full name of the data set is found, the error code is set to eight and the module returns control to the user. Similarly, if the function is anything but catalog and the name was not found, the module takes an error exit.

If the function requested by the user is consistent with the contents of the index, the module checks the generation count and maximum number of generations to be maintained in this index. This indicates whether the module must delete any entries to add a new one. The module increases or decreases the generation count according to the function requested (increase for catalog, decrease for uncatalog, leave alone for recatalog). It rewrites the index block containing the updated generation index pointer entry.

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 (IGC0002F). 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).

---

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

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.

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.

Directory

This chart, Figure 10, 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	DESCRIPTION	CSECT NAME	FLOWCHART NUMBERS
IGC0002F	SYS1.SVCLIB	Initialize	IGC026	1
IGG0CLC1	SYS1.SVCLIB	Locate	IGG0CLC1	2
IGG0CLC2	SYS1.SVCLIB	Build and free block	IGG0CLC2	3
IGG0CLC3	SYS1.SVCLIB	Update blocks of reorganized index	IGG0CLC3	4
IGG0CLC4	SYS1.SVCLIB	Locate generations	IGG0CLC4	5
IGG0CLC5	SYS1.SVCLIB	Build generation index entries	IGG0CLC5	6
IGG0CLC6	SYS1.SVCLIB	Process errors; Exit for LOCATE processing	IGG0CLC6	7
IGG0CLC7	SYS1.SVCLIB	Update control entries; Release blocks; Build and delete index structures	IGG0CLC7	8
IGC0002H	SYS1.SVCLIB	Open/extend catalog	IGC028	9
IGG0CLF2	SYS1.SVCLIB	Format catalog & PDS directory	IGG0CLF2	10

Figure 10. Directory

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 11 and 12 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.

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.

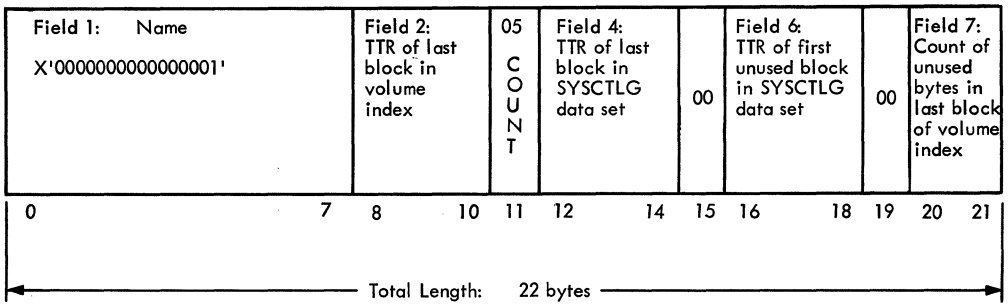
Field 1: This is the name field and contains the name of the index to which this entry points. If the entry is an index link entry, the name field contains X'FFFFFFFFFFFFFFFF'.

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.

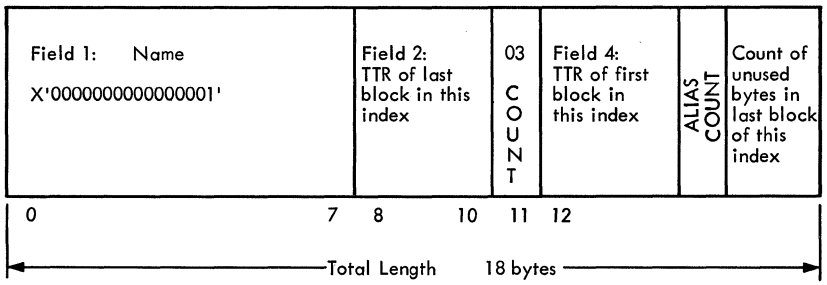
Field 3: This is the count field, and it contains zero to indicate that the entry ends here.

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.

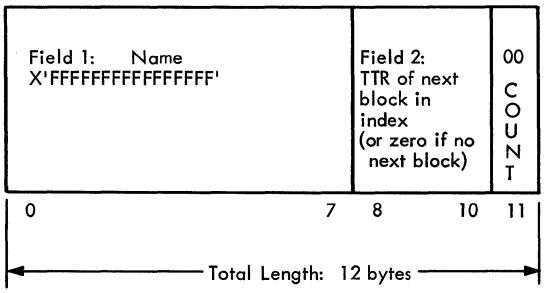
Volume Index Control Entry



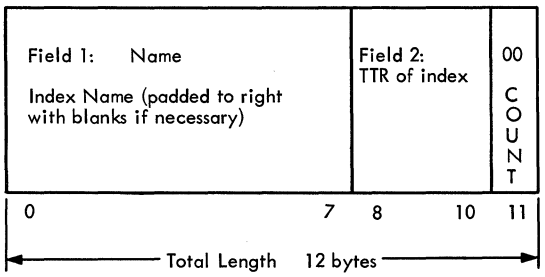
Index Control Entry



Index Link Entry



Index Pointer Entry



Data Set Pointer Entry

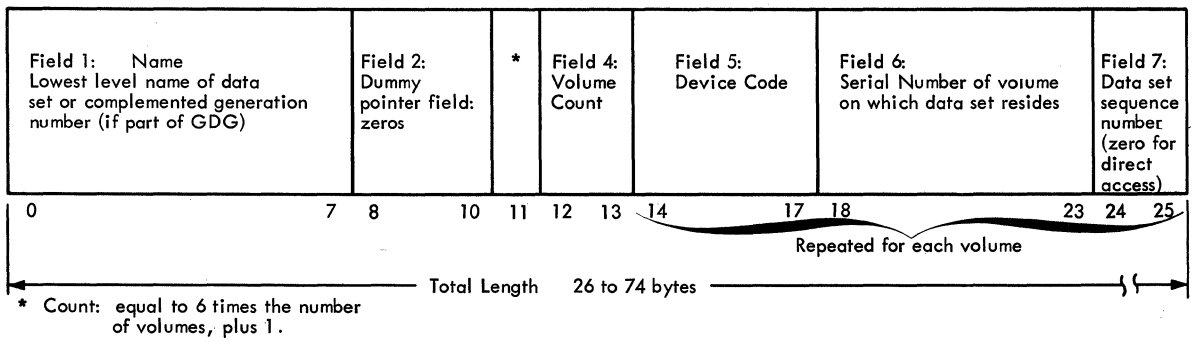
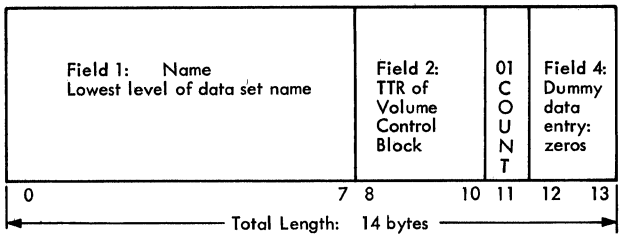
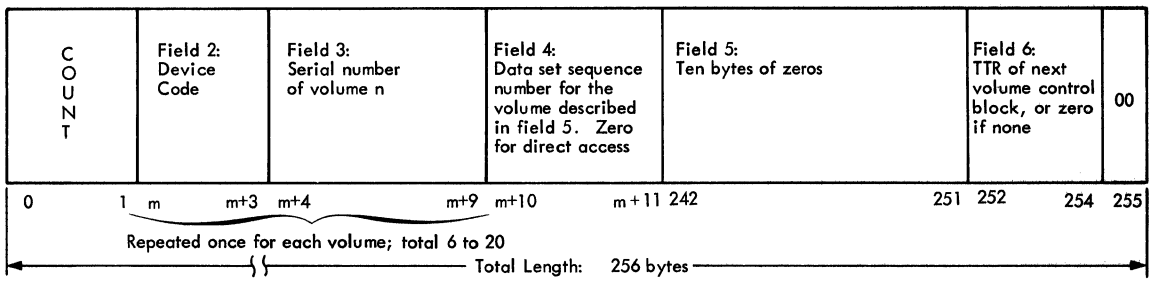


Figure 11. Catalog Entry Formats

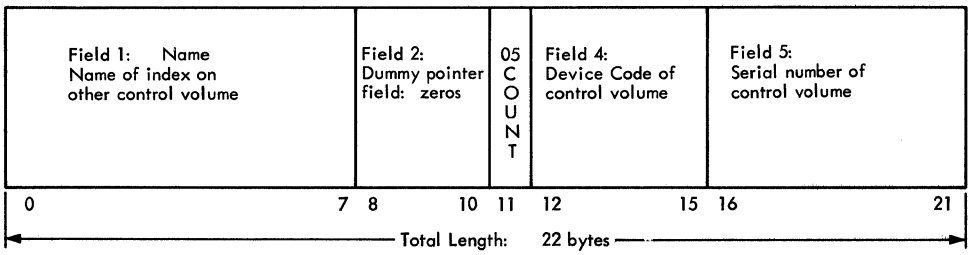
Volume Control Block Pointer Entry



Volume Control Block

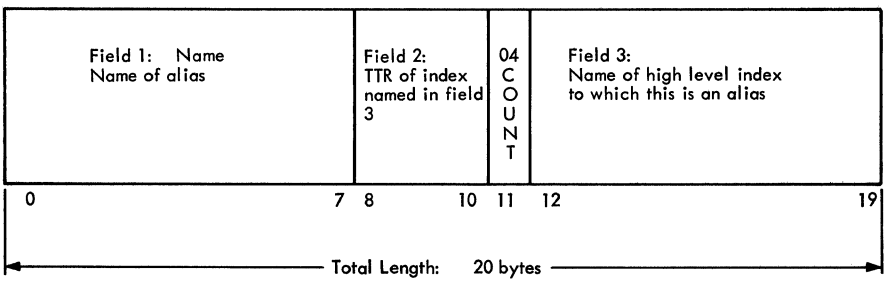


Control Volume Pointer Entry

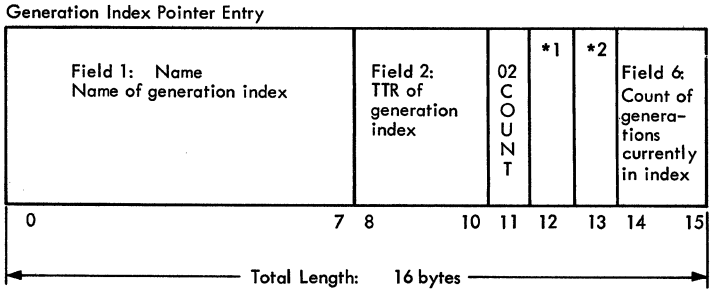


NOTE: Prior to release 17, the Control Volume Pointer Entry contained a count of 03 and did not have a Device Code field (Field 4)

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 12. 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 5 to indicate that five 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 4 to indicate that four 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.

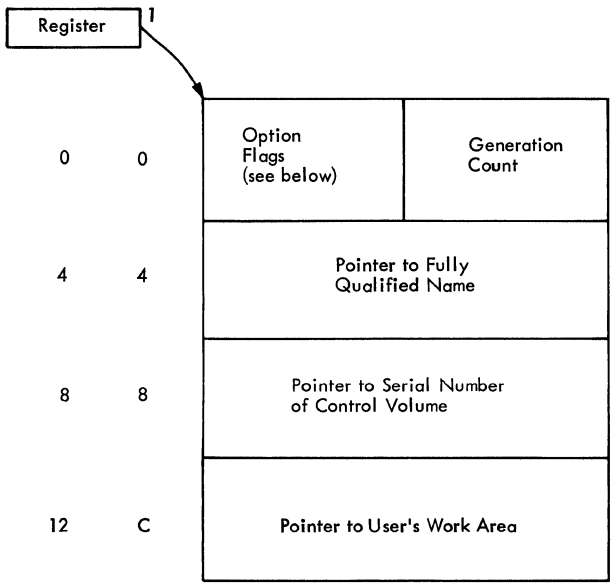
Field 4: This field contains the flags which indicate special handling for generation data sets. Bit 7 indicates the Empty option and bit 6 indicates the Delete option. Bits 0-5 are reserved and are always zero.

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, Figure 13, must be supplied by the user before he calls the catalog management routines. The CAMLST macro instruction, described in IBM System/360 Operating System: 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..	Not used by the Catalog Management routine
	..1.	CTLG Catalog a data set
	...1	RECAT Recatalog a data set
	.... 1...	UNCAT Uncatalog a data set
	.... .X..	Not used by the Catalog Management routine
	.... ..1.	BLOCK Read a block by TTR
	X	Not used by the Catalog Management routine
Byte 1	X...	Not used by the Catalog Management routine
	.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.	Not used by the Catalog Management routine
	1	DLTA Delete an alias entry
Byte 2	1...	DRPX Disconnect control volumes
	.1..	DELETE Scratch generation data sets when they are uncataloged
	..XX	Not used by the Catalog Management routine
	.... 1...	EMPTY Remove all entries from the index when the maximum generation count has been reached
	XXX	Not used by the Catalog Management routine

Note: Function is locate by name if all flags are zero. Function is CATBX if CTLG and BLDX flags are both ones. Function is UCATDX if UNCAT and DLTX flags are both ones.

Figure 13. User's Parameter List

Diagnostic Aids

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

Module Selection Chart

This chart, Figure 14 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
	IGG0CLC7			X	X	X	X	X	X

Figure 14. Module Selection Chart

Register Usage

Figure 15 is a register usage chart. In the chart, the contents of certain registers is given, as it apppears 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				Function Code*		ENQ Parameter List		User's Parameter List	DCB			Work Area	BLDL Work Area	
IGG0CLC1	Entry				Function Code*		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*
IGG0CLC7	Entry					Link Entry Old Index	Area for Updated Link Entry			DCB	Work Area				
	Exit														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 Rtne)	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 Rtne)			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 15. 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).

Chart 1. Catalog Management IGC0002F

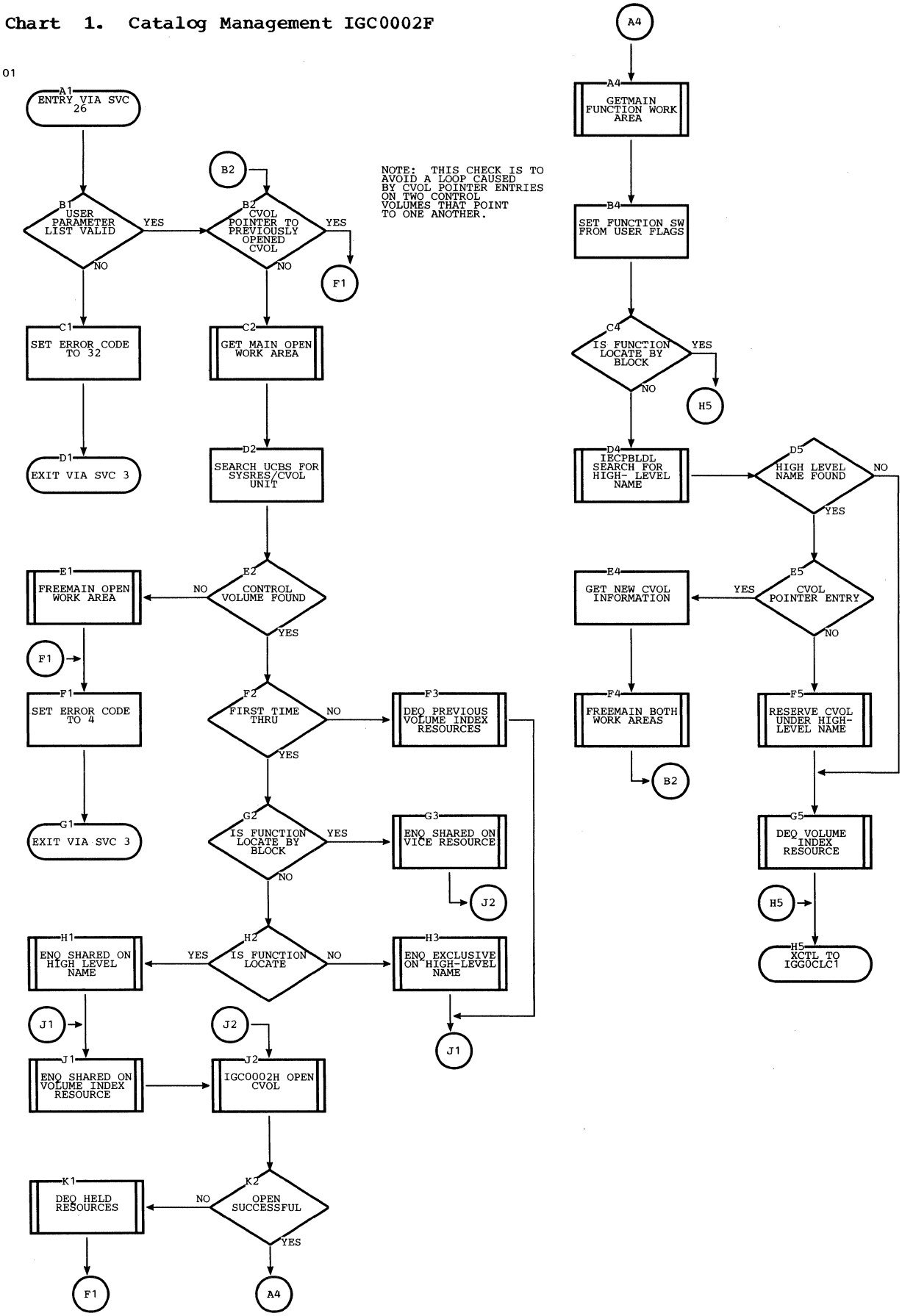


Chart 2. Catalog Management IGC0002H (Part 1 of 2)

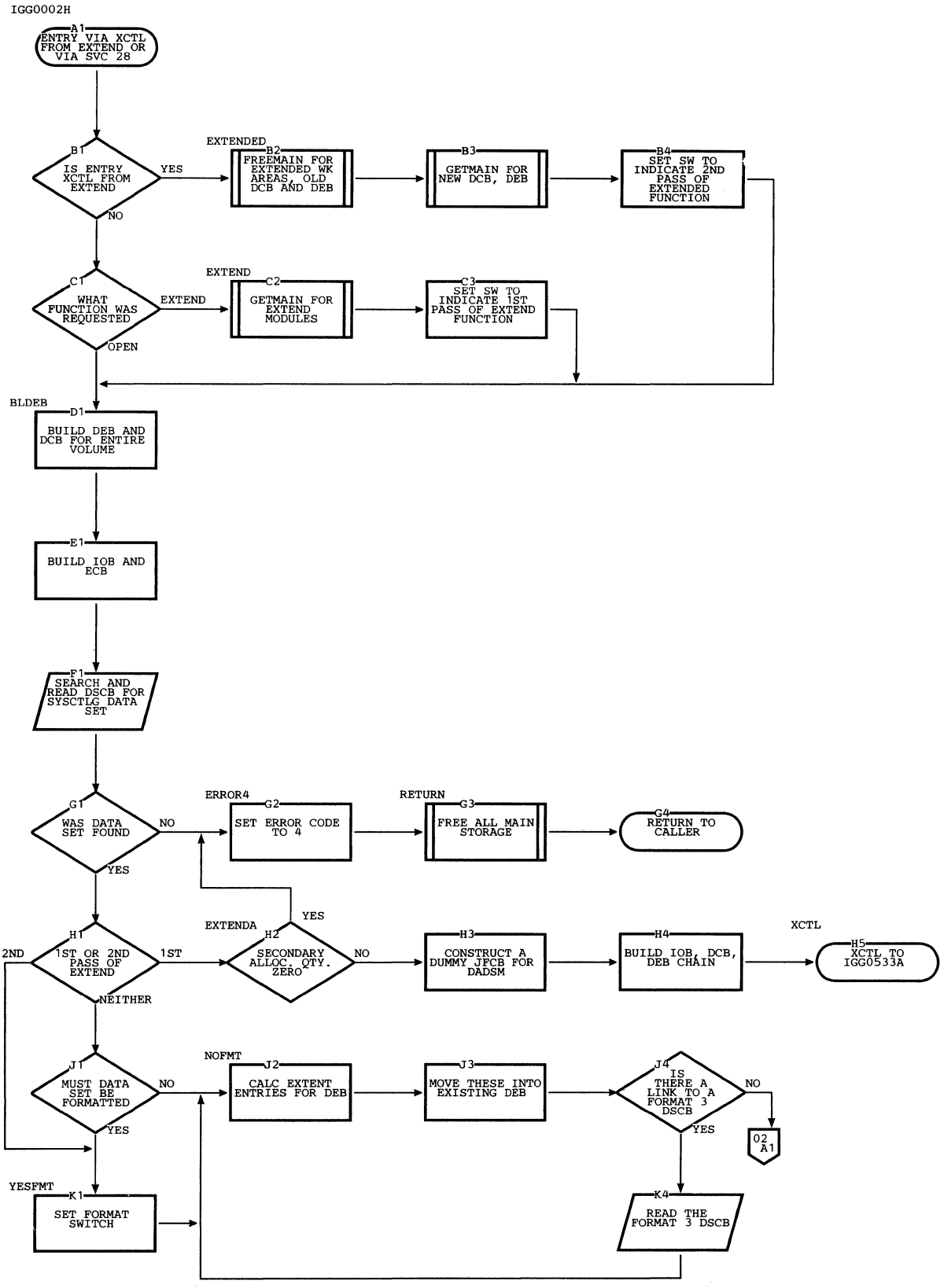


Chart 2. Catalog Management IGC0002H (Part 2 of 2)

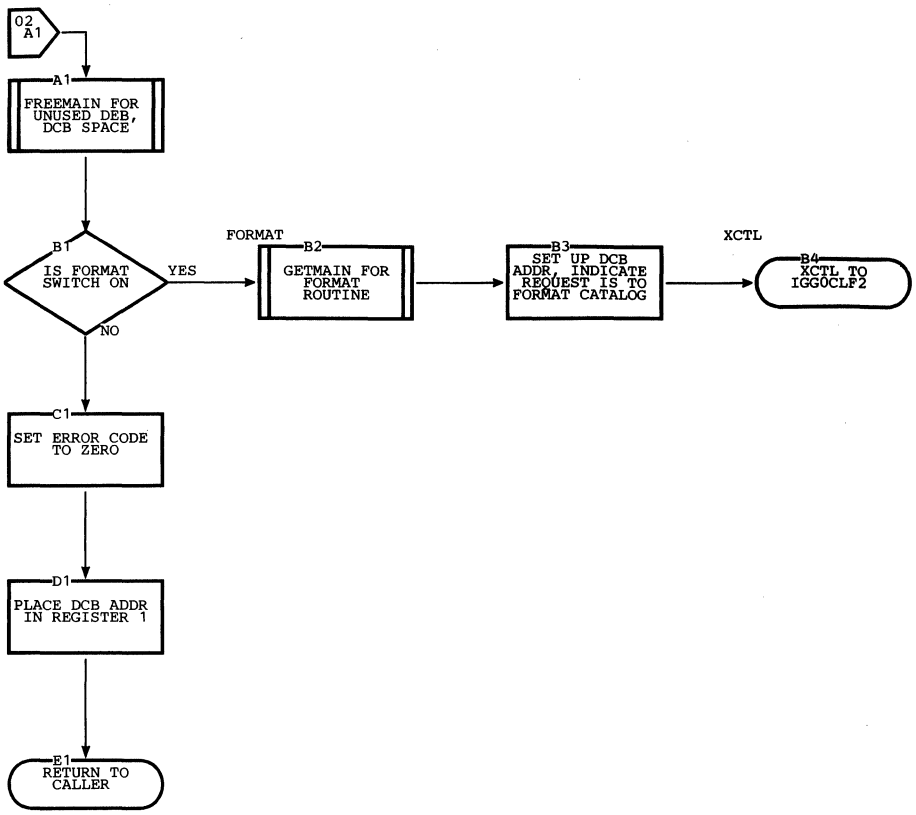


Chart 3. Catalog Management IGG0CLC1 (Part 1 of 2)

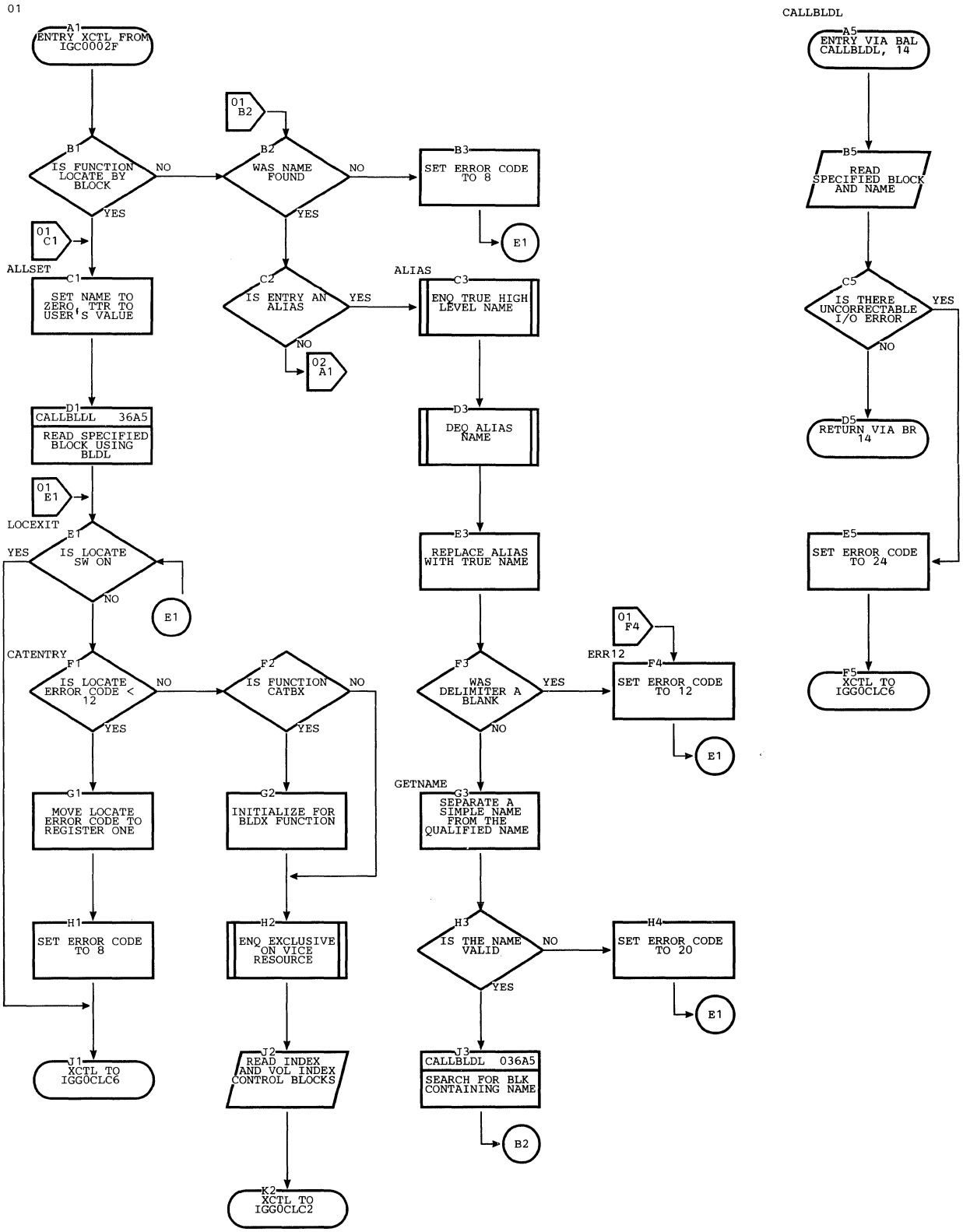


Chart 3. Catalog Management IGG0CLC1 (Part 2 of 2)

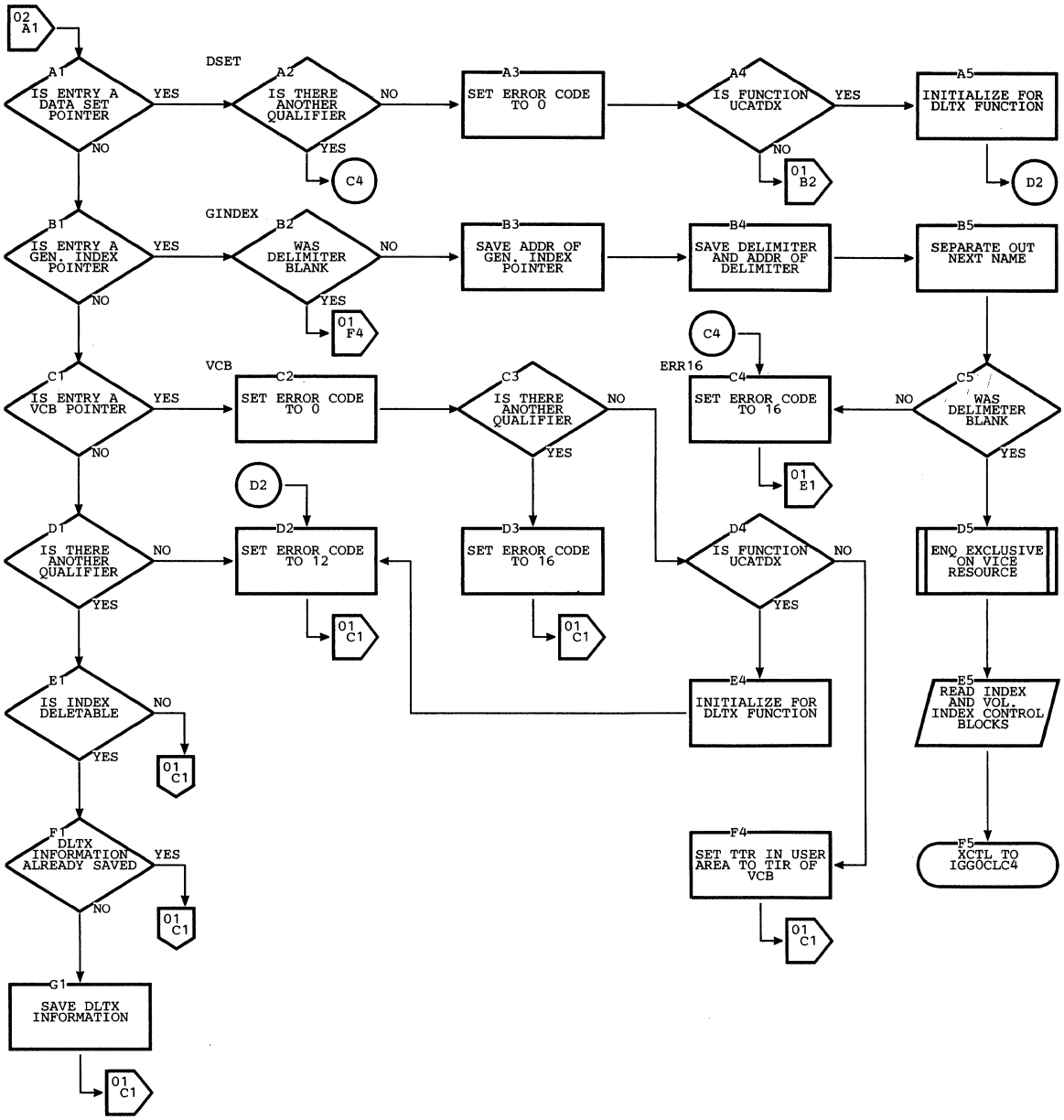


Chart 4. Catalog Management IGG0CLC2 (Part 1 of 3)

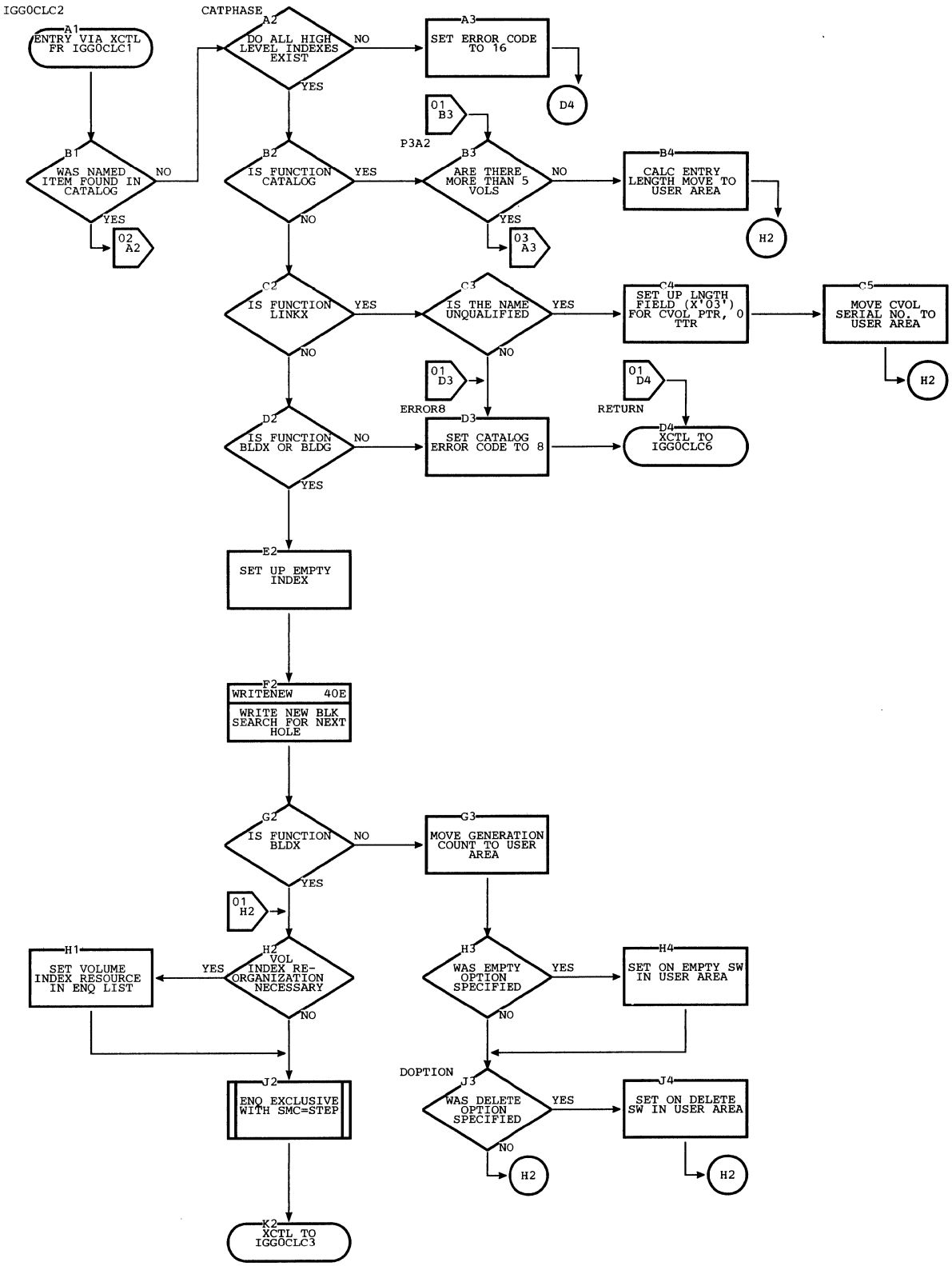


Chart 4. Catalog Management IGG0CLC2 (Part 2 of 3)

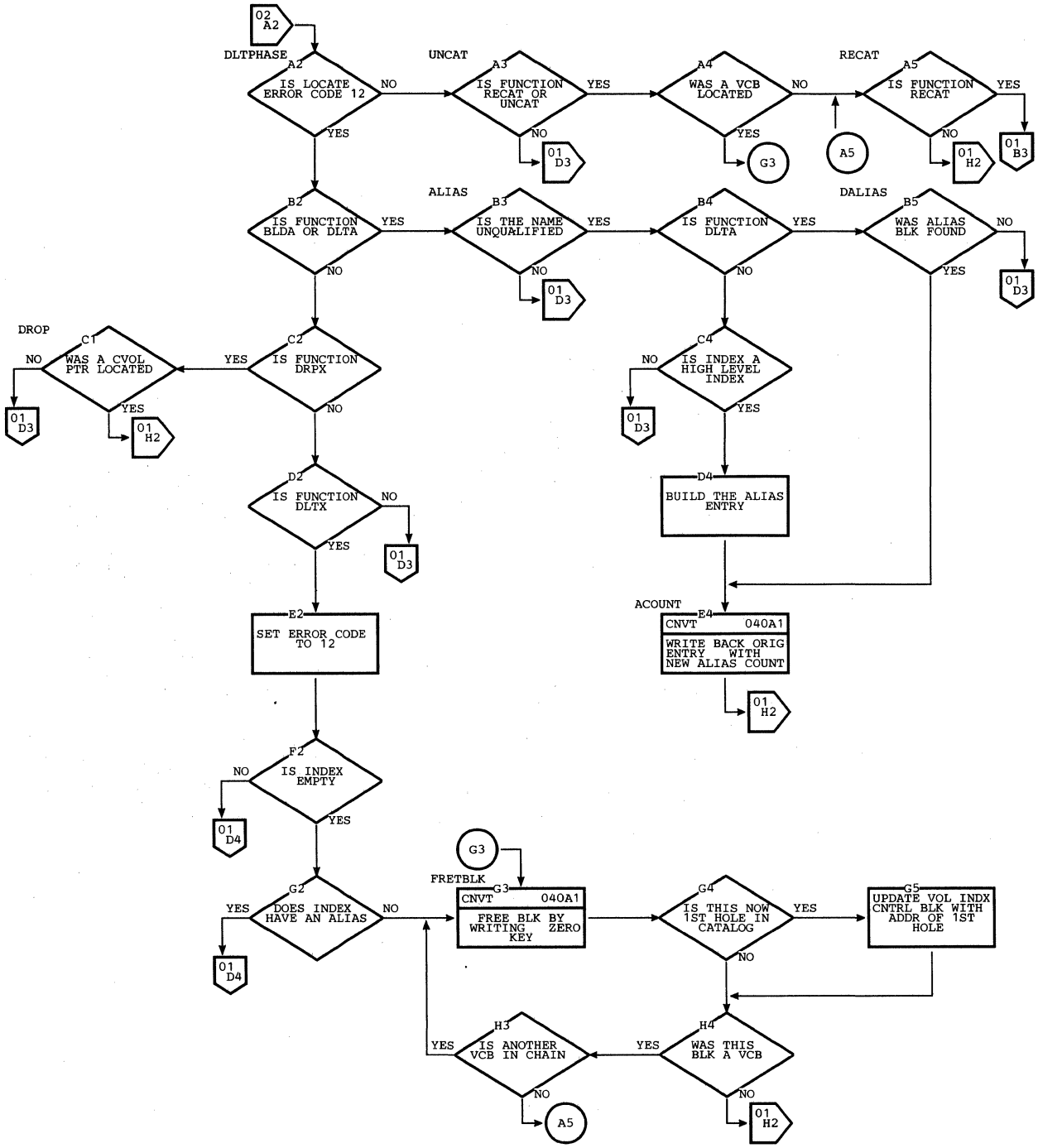


Chart 4. Catalog Management IGG0CLC2 (Part 3 of 3)

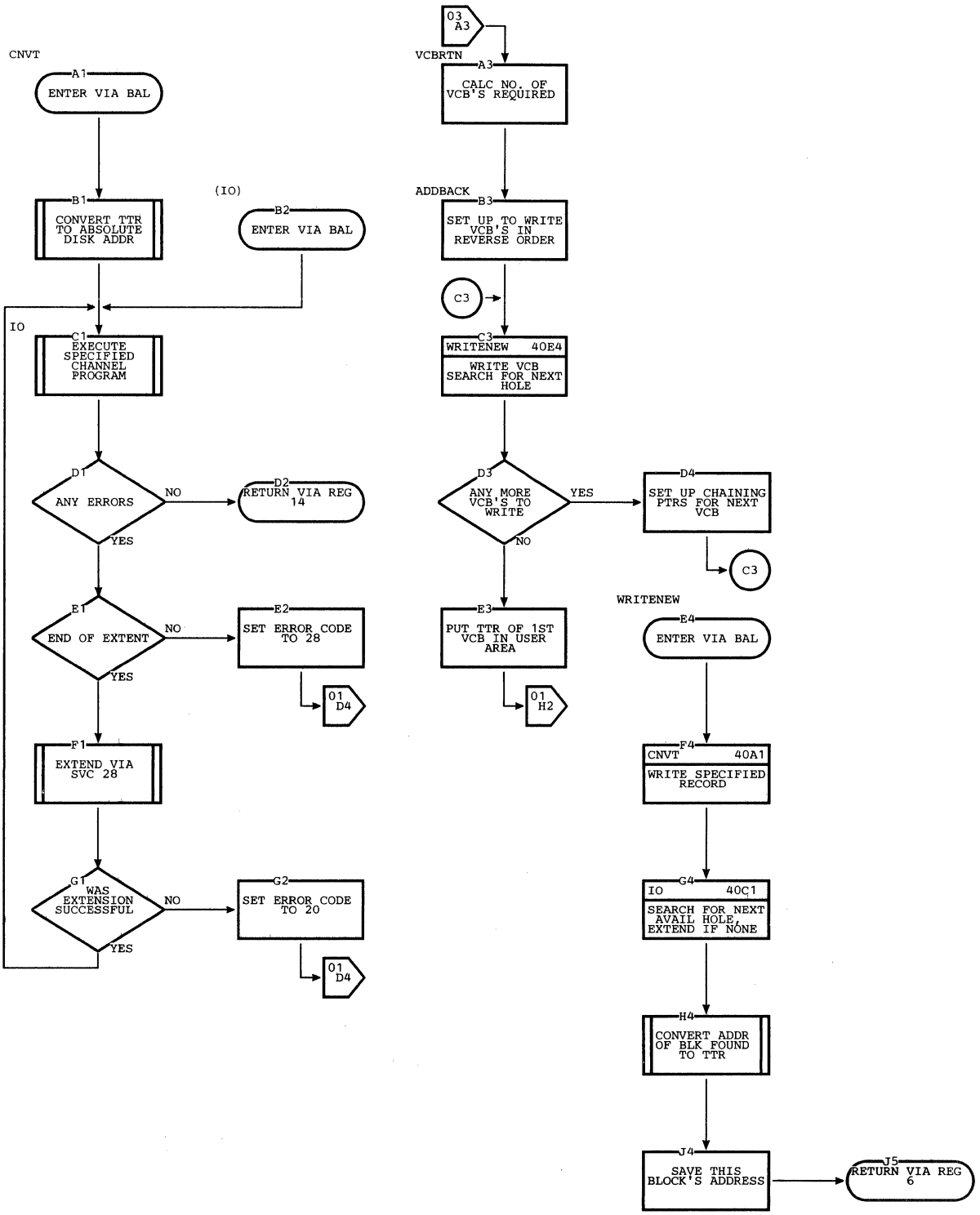


Chart 5. Catalog Management IGG0CLC3 (Part 1 of 2)

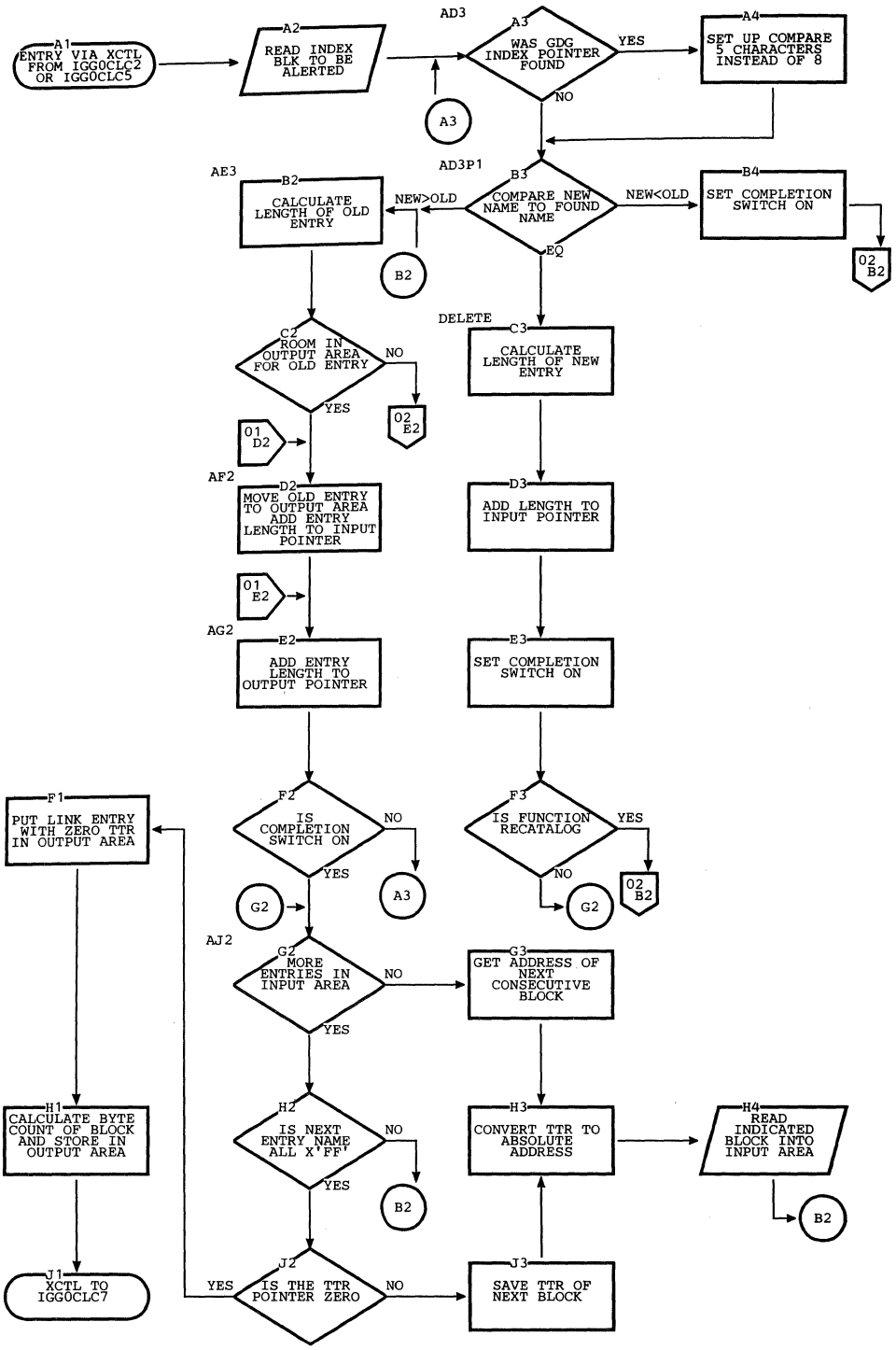


Chart 5. Catalog Management IGG0CLC3 (Part 2 of 2)

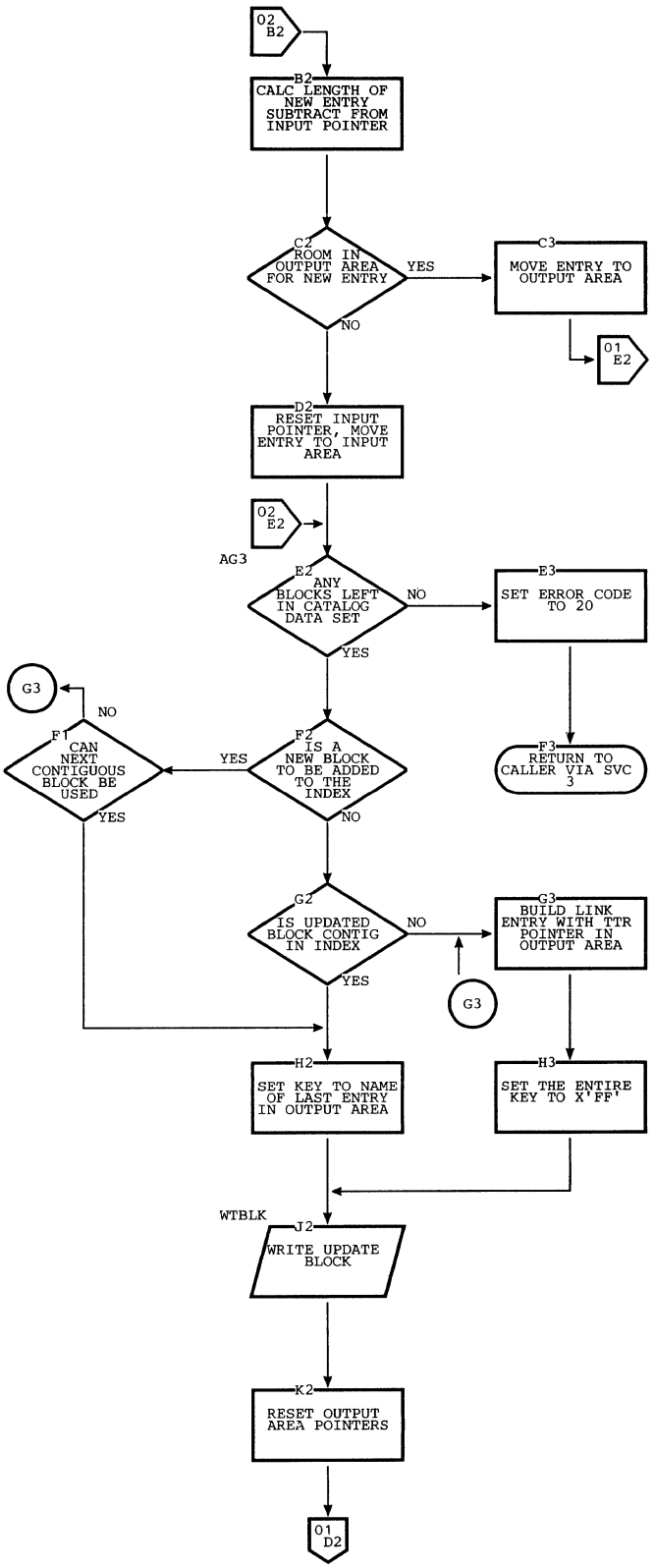


Chart 6. Catalog Management IGG0CLC4 (Part 1 of 3)

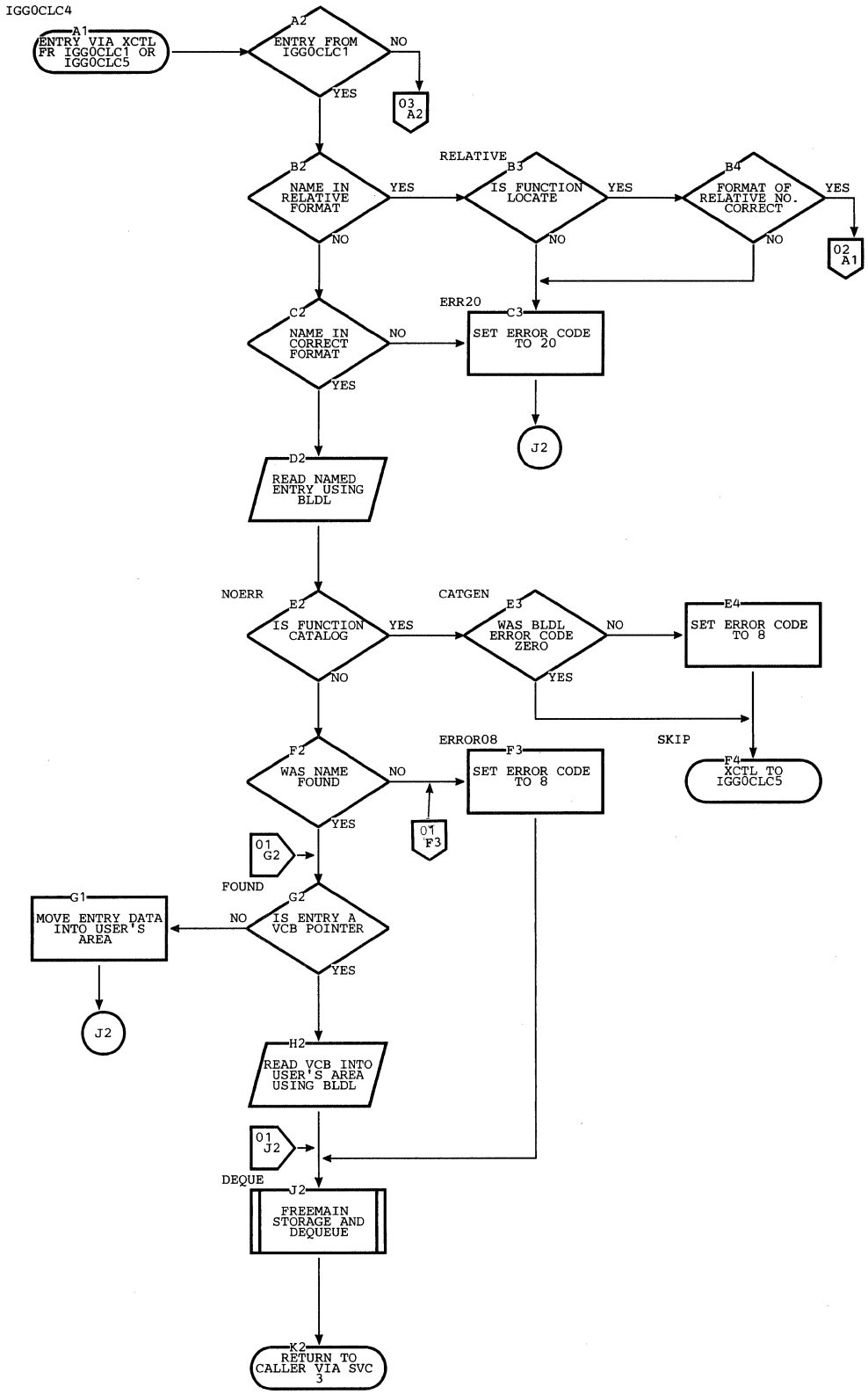


Chart 6. Catalog Management IGG0CLC4 (Part 2 of 3)

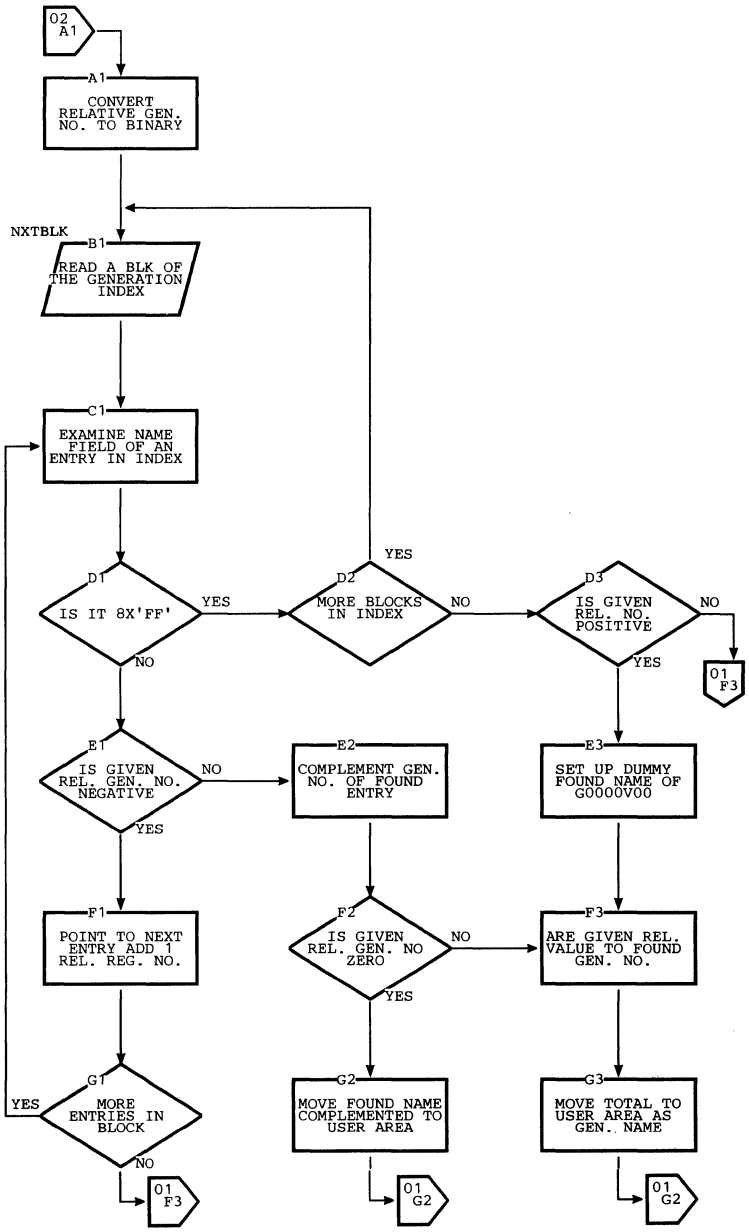


Chart 6. Catalog Management IGG0CLC4 (Part 3 of 3)

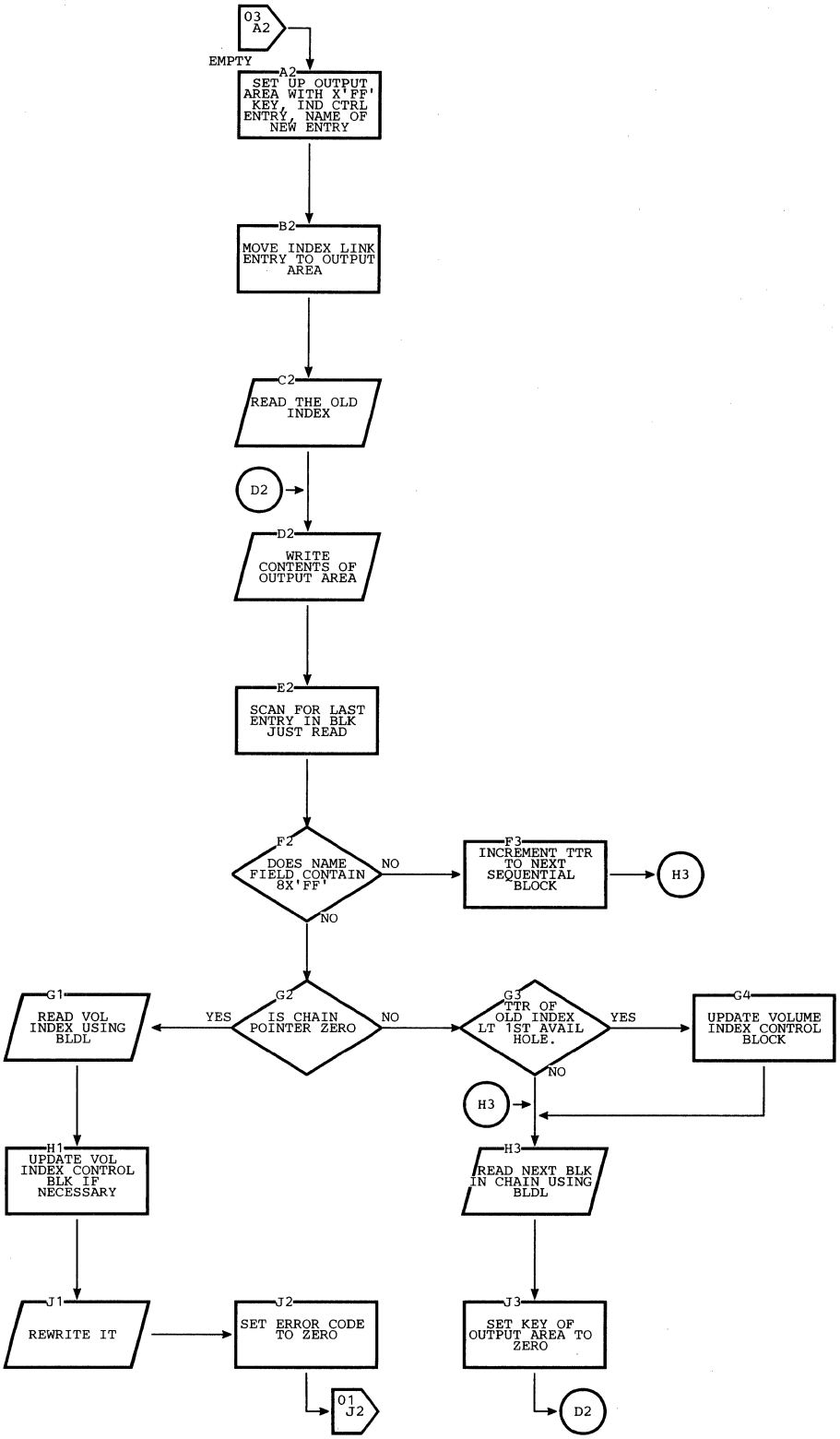


Chart 7. Catalog Management IGG0CLC5 (Part 1 of 2)

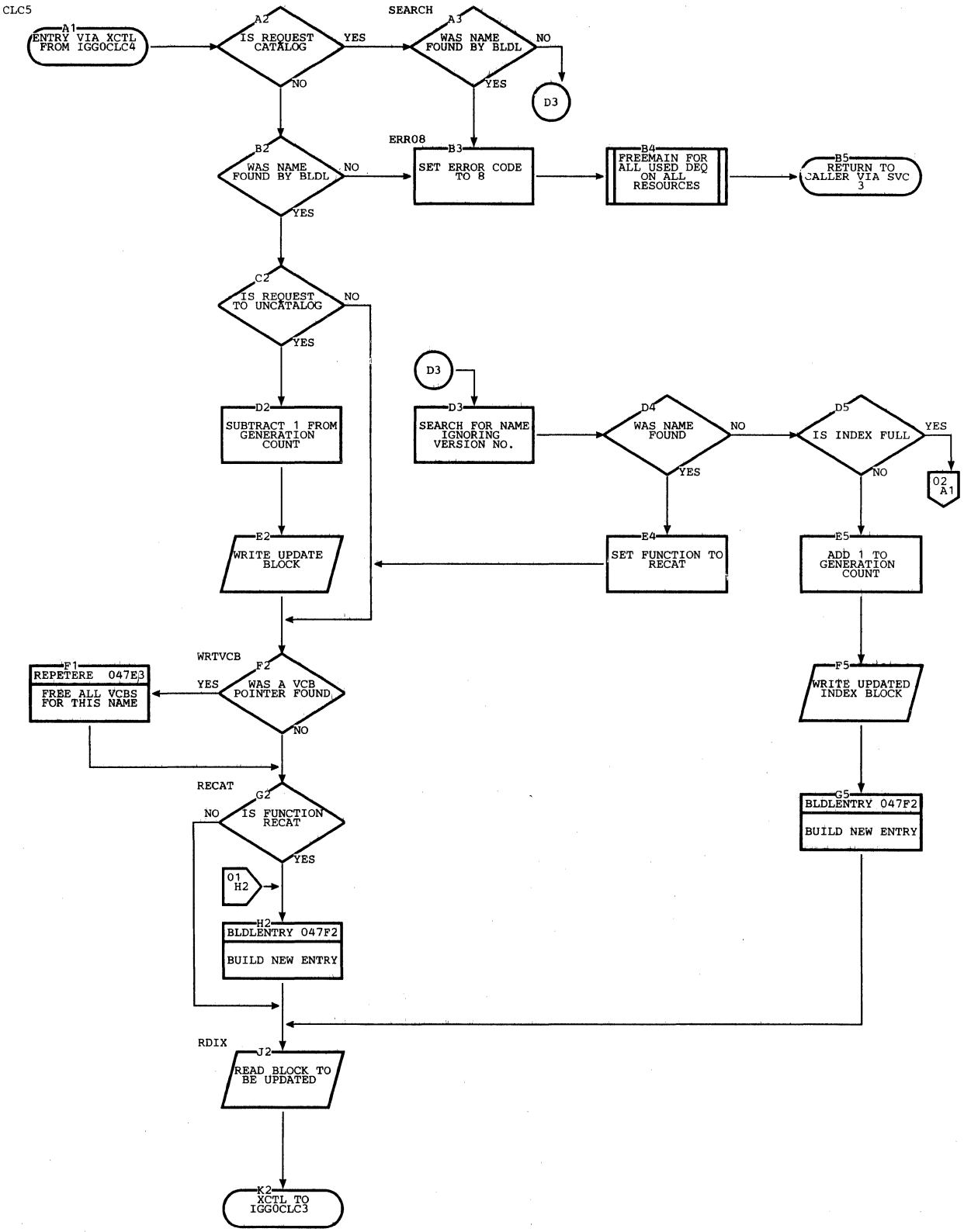


Chart 7. Catalog Management IGG0CLC5 (Part 2 of 2)

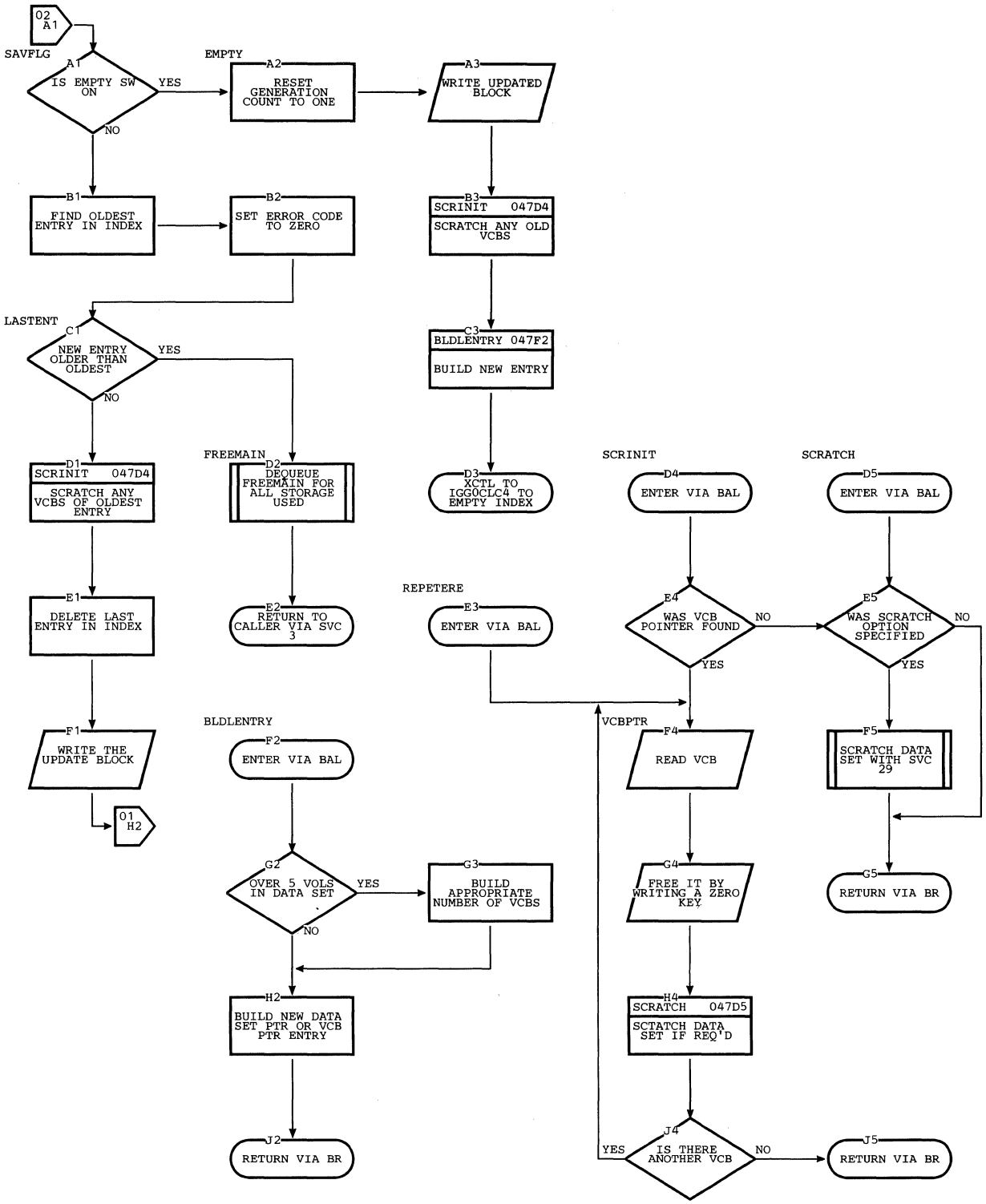


Chart 8. Catalog Management IGG0CLC6

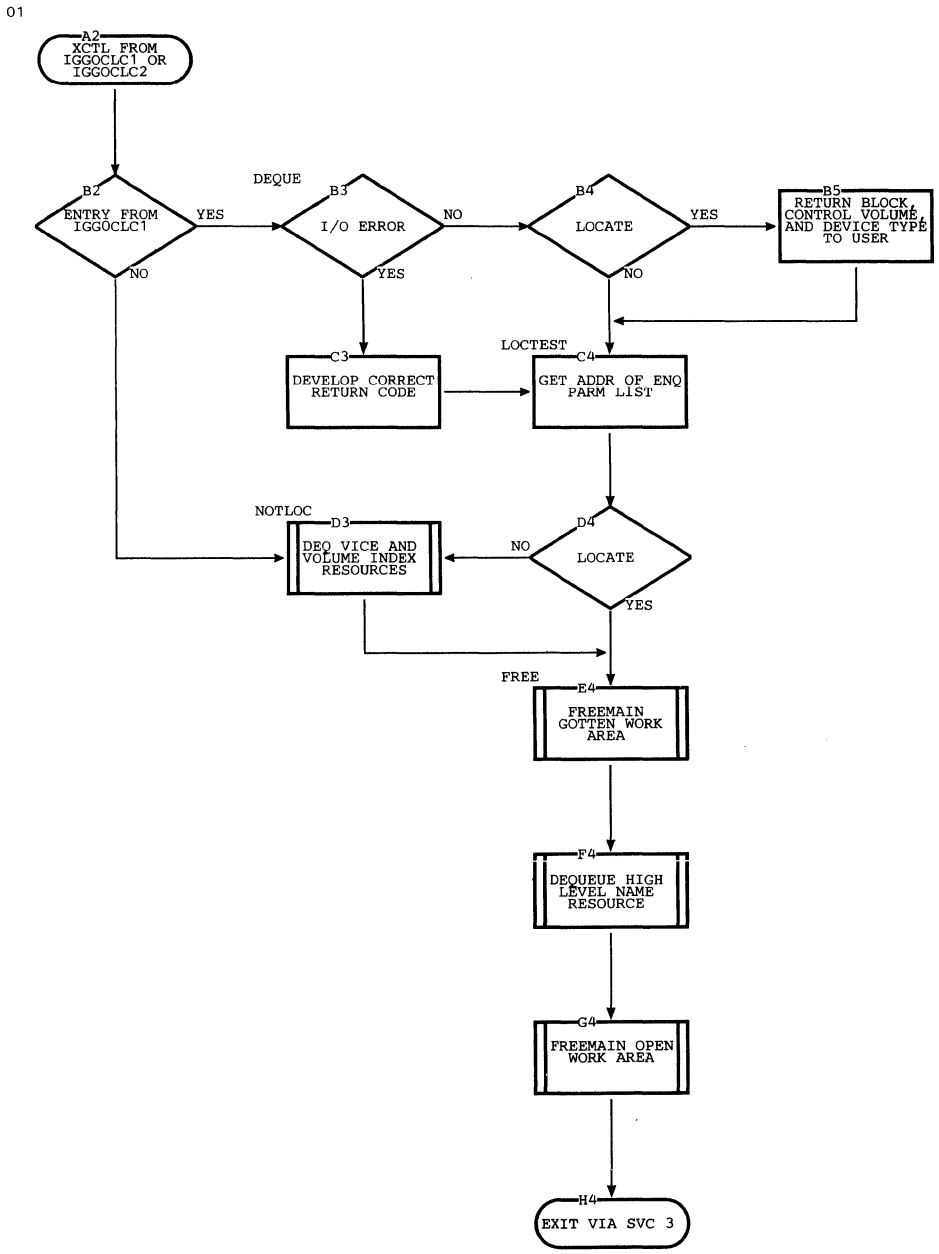


Chart 9. Catalog Management IGG0CLC7

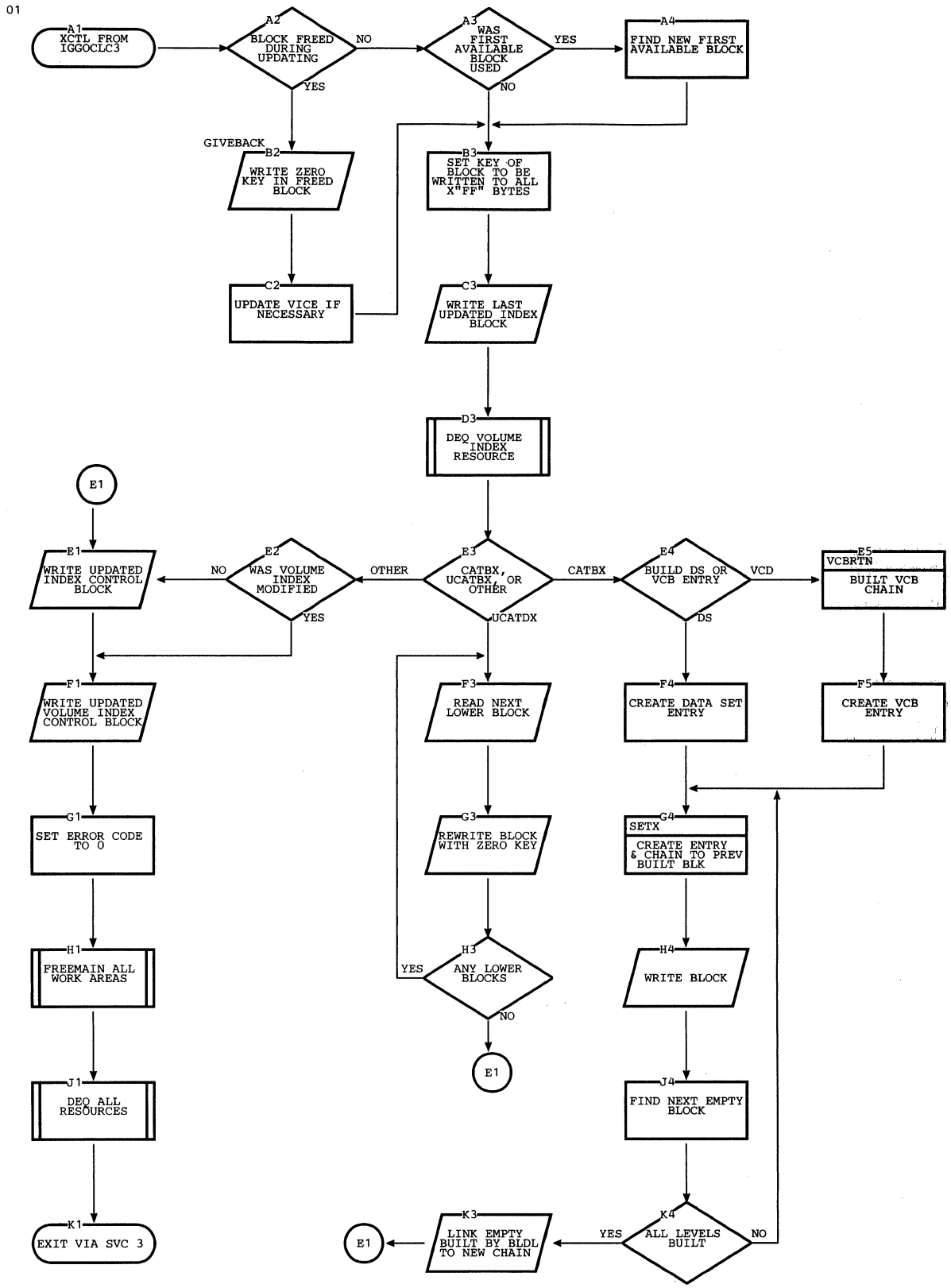
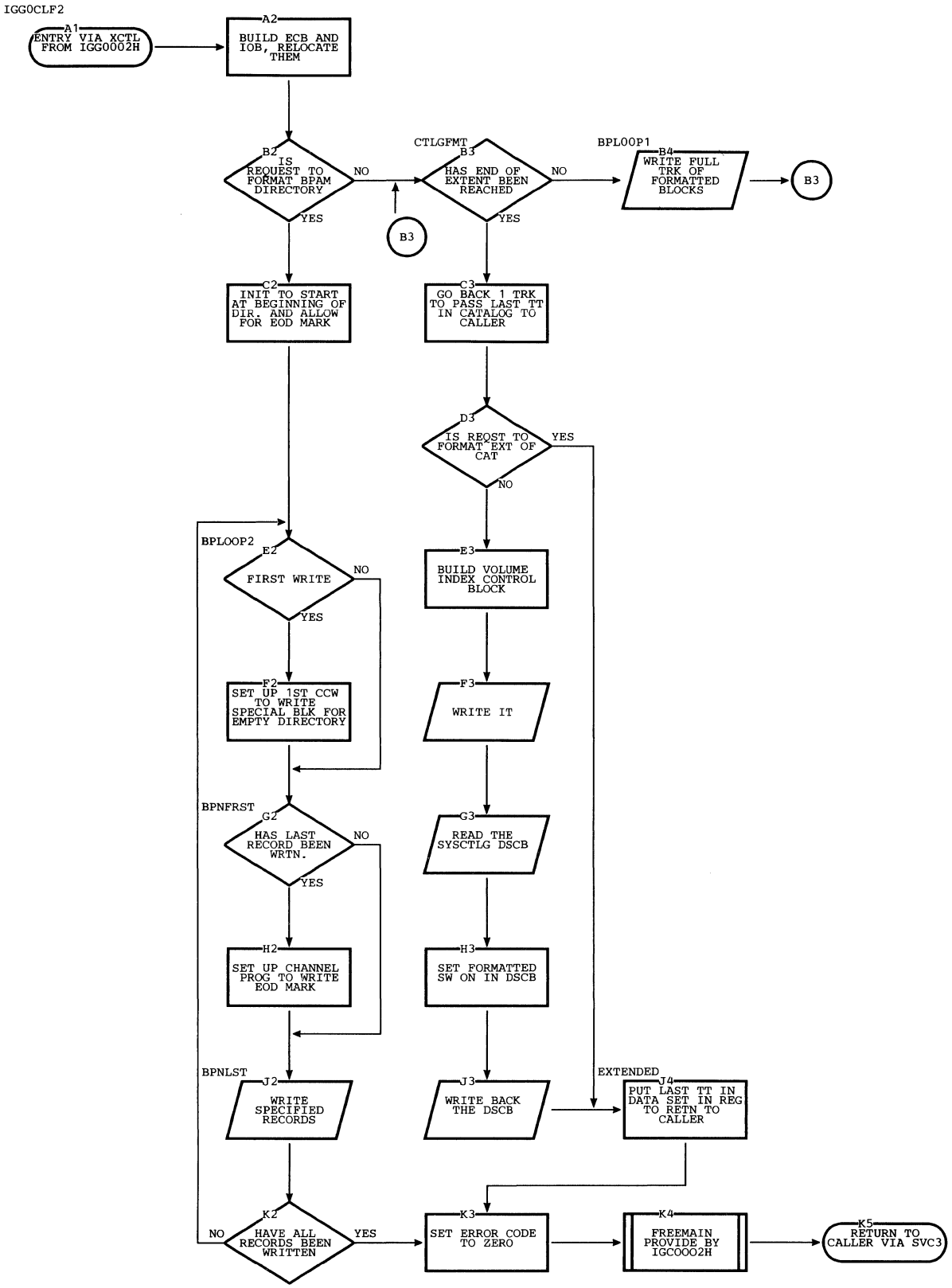
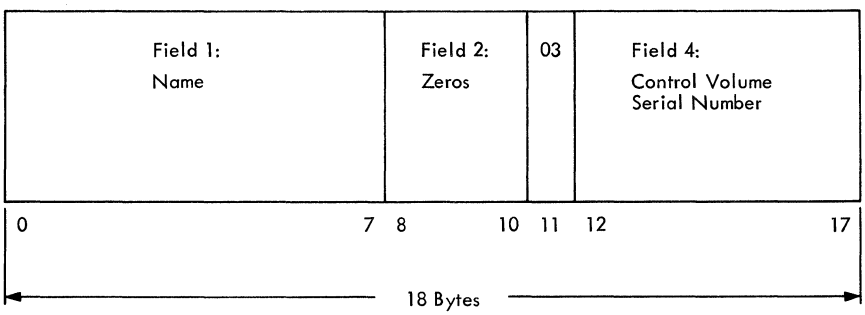


Chart 10. Catalog Management IGG0CLF2



Appendix B: Old CVOL Pointer

Before Release 17, the control volume pointer entry had nc 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.

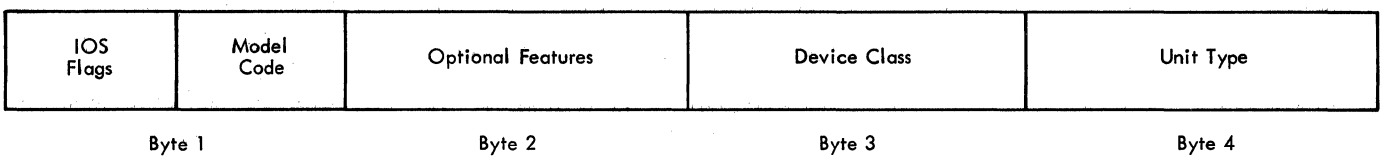
For a complete description of the fields above, please refer to IBM System/360 Operating System: System Control Blocks, Form C28-6628. A brief description of some of the fields appears below.

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



Index

Indexes to program logic manuals are consolidated in the publication IBM System/360 Operating System: Program Logic Manual Master Index, Form Y28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

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

System/360 OS TSO Catalog Management PLM (S360-36) Printed in U.S.A. GY28-6745-0

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

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

READER'S COMMENT FORM

IBM System/360 Operating System:
TSO Catalog Management
Program Logic Manual

Order No. GY28-6745-0

Please use this form to express your opinion of this publication. We are interested in your comments about its technical accuracy, organization, and completeness. All suggestions and comments become the property of IBM.

Please do not use this form to request technical information or additional copies of publications. All such requests should be directed to your IBM representative or to the IBM Branch Office serving your locality.

- Please indicate your occupation: _____
- How did you use this publication?
  - ☐ Frequently for reference in my work.
  - ☐ As an introduction to the subject.
  - ☐ As a textbook in a course.
  - ☐ For specific information on one or two subjects.
- Comments (Please include page numbers and give examples.):

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

YOUR COMMENTS, PLEASE . . .

This manual is part of a library that serves as a reference source 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.

Note: Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Fold Fold

FIRST CLASS
PERMIT NO. 81
POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY ...

IBM Corporation
P.O. Box 390
Poughkeepsie, N.Y. 12602

Attention: Programming Systems Publications
Department D58

Fold Fold

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

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

Cut Along Line

System/360 OS TSO Catalog Management PLM (S360-36) Printed in U.S.A. GY28-6745-0

READER'S COMMENT FORM

IBM System/360 Operating System:
TSO Catalog Management
Program Logic Manual

Order No. GY28-6745-0

Please use this form to express your opinion of this publication. We are interested in your comments about its technical accuracy, organization, and completeness. All suggestions and comments become the property of IBM.

Please do not use this form to request technical information or additional copies of publications. All such requests should be directed to your IBM representative or to the IBM Branch Office serving your locality.

- Please indicate your occupation: _____
- How did you use this publication?
  - ☐ Frequently for reference in my work.
  - ☐ As an introduction to the subject.
  - ☐ As a textbook in a course.
  - ☐ For specific information on one or two subjects.
- Comments (Please include page numbers and give examples.):

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

YOUR COMMENTS, PLEASE . . .

This manual is part of a library that serves as a reference source 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.

Note: Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Fold Fold

FIRST CLASS
PERMIT NO. 81
POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY ...

IBM Corporation
P.O. Box 390
Poughkeepsie, N.Y. 12602

Attention: Programming Systems Publications
Department D58

Fold Fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

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

Cut Along Line

System/360 OS TSO Catalog Management PLM (S360-36) Printed in U.S.A. GY28-6745-0

IBM Technical Newsletter

File Number S360-20 (OS Rel. 20.1)
Re: Order No. GY28-6745-0
This Newsletter No. GN28-2481
Date June 1, 1971
Previous Newsletter Nos. None

IBM SYSTEM/360 OPERATING SYSTEM:
TSO CATALOG MANAGEMENT

This Technical Newsletter, a part of release 20.1 of IBM System/360 Operating System, provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent releases unless specifically altered. Pages to be inserted and/or removed are:

Cover,Edition Notice
6.1
7,8
25-28
35,36
41,42
57-62

A change to the text or a small change to an illustration is indicated by a vertical line to the left of the change.

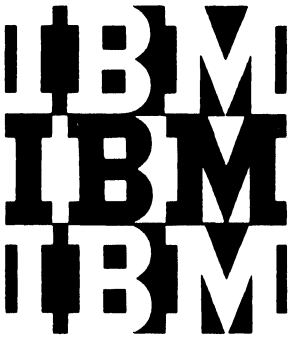
Summary of Amendments

This Technical Newsletter adds information on the Catalog Support for Rotational Position Sensing.

Note: Please file this cover at the back of the manual to provide a record of changes.

IBM Corporation, Programming Systems Publications, P.O. Box 390, Poughkeepsie, N.Y. 12602

PRINTED IN U.S.A.



File No. S360-36
Order No. GY28-6745-0

Program Logic

IBM System/360 Operating System:

Time Sharing Option

Catalog Management

Program Logic Manual

Program Number 360S-DM-508

This publication provides customer engineers and other technical personnel with information describing the internal organization and logic of the catalog management routines that are used when the Time Sharing Option has been selected at system generation time. These routines provide the facility of locating data sets when only data set names are specified.

This manual is based on the IBM System/360 Operating System: Catalog Management, Program Logic Manual, GY28-6606. It should be used in place of the above manual only if the Time Sharing Option has been specified at system generation time.

First Edition (March, 1971)

This edition with Technical Newsletter GN28-2481 applies to Release 20.1, of IBM System/360 Operating System, and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM systems, refer to the latest IBM System/360 SRL Newsletter, Order No. GN20-0360, for the editions that are applicable and current.

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

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Systems Publications, Department D58, PO Box 390, Poughkeepsie, N. Y. 12602

Summary of Major Changes

Release 20.1 (GY28-6745-0)

Item	Description	See Pages:
Rotational Position Sensing (RPS) for 3330 and 2305	An I/O feature that permits channel use during seek time and record search operations.	7,26,27,35,41, 57,59

Introduction

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. For further information concerning Rotational Position Sensing (RPS), which is mentioned throughout this manual, refer to the section concerning this feature in IBM System/360 Operating System: Direct Access Device Space Management Program Logic Manual, GY28-6607.

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.

The catalog management routines can be used to build or delete a single index or a whole index structure. To catalog a data set called A.B.C, for example, the user may either first create index A, then index A.B, and then catalog A.B.C, or request that catalog management create any missing index levels needed to catalog 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 Structure

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, Form 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 may start 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 CVOL whose serial number is in the control pointer entry. The caller of the catalog management routine may specify what CVOL is used for the search.

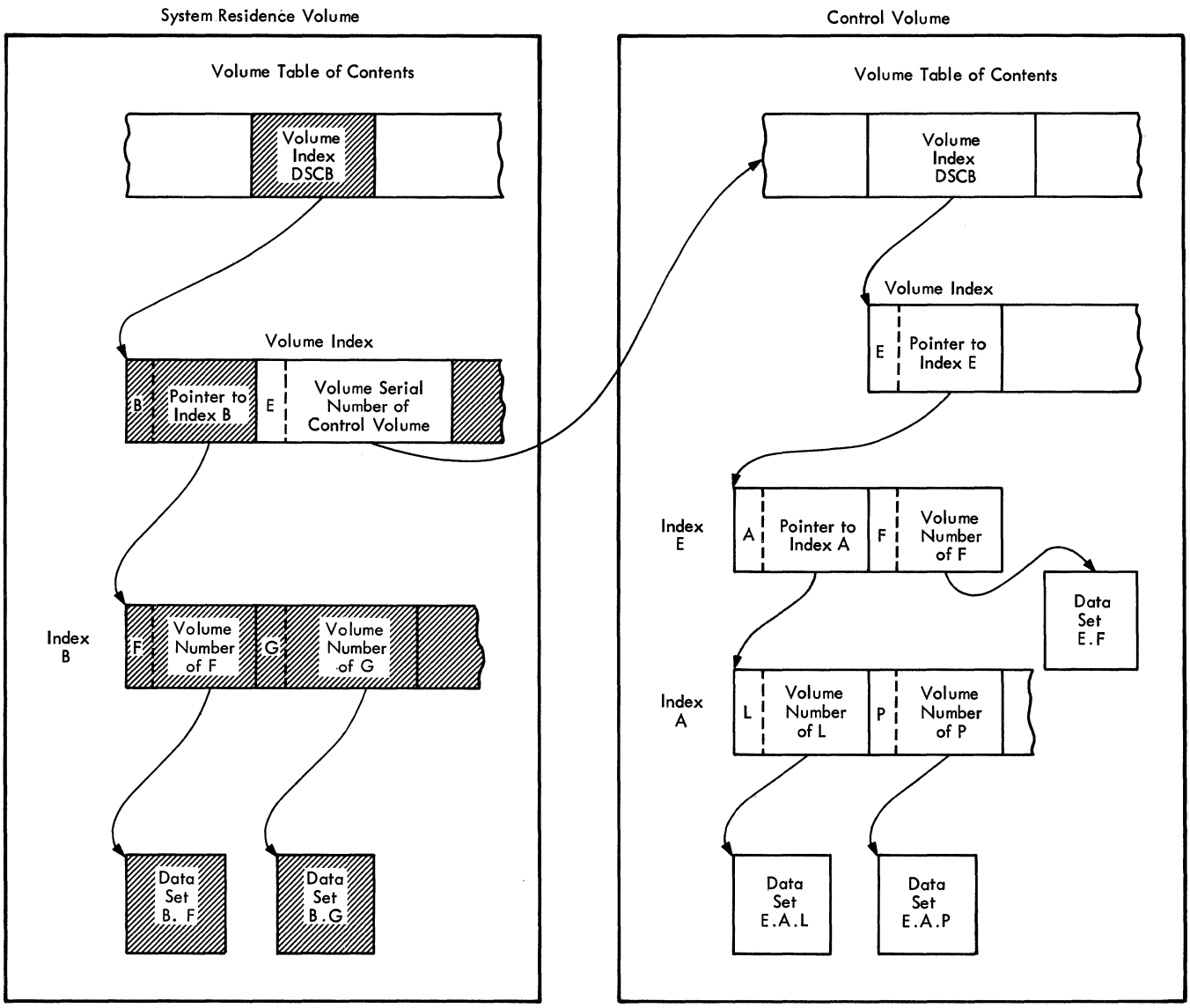


Figure 1. A Control Volume Connected to the System Residence Volume

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, the job scheduler, and TSO, 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.

TSO dynamic allocation locates old data sets and catalogs new data sets. TSO command processors also call the catalog management routines to manipulate the catalog.

Locate Generations: Module IGG0CLC4

Generation data groups require significantly different locating and cataloging procedures from other data sets for two reasons:

(1) Generation data groups may be specified by relative generation number (as in GENR(+1)), in which case the absolute generation number must be calculated, and

(2) The absolute generation number is stored in the catalog in hexadecimal complement form, that is, generation G0001V00 would be stored as X'C7 0F 0F 0F 0E E5 F0 F0'. (Note that the version number and the characters 'G' and 'V' are not complemented.) In this way the most recent generation (the one with the highest absolute number) is always the first entry in the index after the index control entry.

In this manual, the term "absolute generation number" refers to the number as it is coded by the user and as it appears in the name field of a data set control block (DSCB). It does not refer to the number as it is stored in the catalog, in complement form.

Module IGG0CLC4 locates the lowest level of a generation name. When IGG0CLC1 finds a generation index pointer entry correlated with the next to last level of a name, it passes control to this module. It may also be entered from IGG0CLC5 when that module finds it must empty an index.

This module first checks to see whether entry is from IGG0CLC5 (empty request) or from IGG0CLC1 (normal locate path). If it was from IGG0CLC5, the module rewrites the generation index, this time with only the highest entry, and frees any blocks no longer needed by the shortened index.

If the path is a normal locate path (entry from IGG0CLC1), IGG0CLC4 checks the format of both relative generation numbers and absolute generation numbers and returns to the user with an error code of 20 if the format of the supplied name is not correct. If the name is in relative format, the only valid function is locate; if any other function has been specified, the module returns with an error code of 20.

If the name is in relative format, the module must calculate its absolute generation number. It does this by adding or subtracting the relative number given and the actual number of the first entry in the index. If the index is empty, the module sets up a dummy 'found' entry called 'G0000V00' as the basis for absolute generation number calculation. If the relative number is negative and exceeds the number of entries in the index, the module returns to the user with an error code of 8.

Once the relative generation number in the user's area has been replaced with the absolute generation number, the module proceeds as though the user had supplied the absolute number in the first place.

With the generation number in absolute format, the module uses BLDL to read the entry associated with the name. If the function is catalog, control is passed to IGG0CLC5 via the XCTL macro instruction. If the function is locate, the module checks BLDL's error code. If the name was found, the module moves the data into the user's area and checks to see if it must read a volume control block to complete the description of the data set. If it does, the volume control block is read into the user's area.

If BLDL cannot find the name, the module returns to the user with an error code.

Catalog Generations: Module IGG0CLC5

This module builds new entries for generation indexes, maintains generation index pointer entries by updating the generation count, and marks entries for deletion or data sets for deletion if the empty or delete option was specified when the generation index was created.

The module first checks the findings of IGG0CLC4 to be sure the current structure of the index is compatible with the function requested. If the requested function is catalog, for example, and the full name of the data set is found, the error code is set to eight and the module returns control to the user. Similarly, if the function is anything but catalog and the name was not found, the module takes an error exit.

If the function requested by the user is consistent with the contents of the index, the module checks the generation count and maximum number of generations to be maintained in this index. This indicates whether the module must delete any entries to add a new one. The module increases or decreases the generation count according to the function requested (increase for catalog, decrease for uncatalog, leave alone for recatalog). It rewrites the index block containing the updated generation index pointer entry.

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 (IGC0002F). 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).

For RPS devices, IGC0002H obtains an RPS work area (and frees it when it frees the DCB and DEB area.) When the DCB and DEB are constructed initially for an RPS device, control is transfered by XCTL to the RPS setup module, IGG019EK. Upon return from IGG019EK, normal DEB construction continues.

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 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.

---

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

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.

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 tests for an RPS device. If the device has the RPS feature, the RPS work area is freed and the RPS appendage module, IGG019EK, is deleted. Then the working storage obtained by IGC0002H is freed.

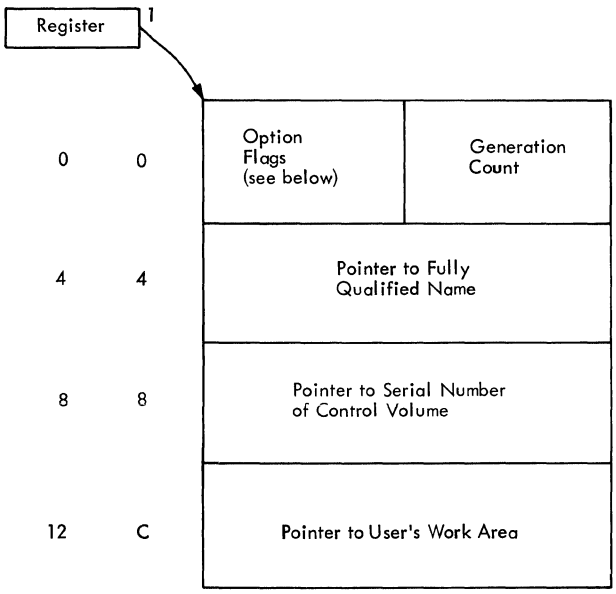
Directory

This chart, Figure 10, 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	DESCRIPTION	CSECT NAME	FLOWCHART NUMBERS
IGC0002F	SYS1.SVCLIB	Initialize	IGC026	1
IGG0CLC1	SYS1.SVCLIB	Locate	IGG0CLC1	2
IGG0CLC2	SYS1.SVCLIB	Build and free block	IGG0CLC2	3
IGG0CLC3	SYS1.SVCLIB	Update blocks of reorganized index	IGG0CLC3	4
IGG0CLC4	SYS1.SVCLIB	Locate generations	IGG0CLC4	5
IGG0CLC5	SYS1.SVCLIB	Build generation index entries	IGG0CLC5	6
IGG0CLC6	SYS1.SVCLIB	Process errors; Exit for LOCATE processing	IGG0CLC6	7
IGG0CLC7	SYS1.SVCLIB	Update control entries; Release blocks; Build and delete index structures	IGG0CLC7	8
IGC0002H	SYS1.SVCLIB	Open/extend catalog	IGC028	9
IGG0CLF2	SYS1.SVCLIB	Format catalog & PDS directory	IGG0CLF2	10

Figure 10. Directory



[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 CVOL	
	.X..	Not used by the Catalog Management routine	
	..1.	CTLG	Catalog a data set
	...1	RECAT	Recatalog a data set
	.... 1...	UNCAT	Uncatalog a data set
	.... .X..	Not used by the Catalog Management routine	
	.... ..1.	BLOCK	Read a block by TTR
	X	Not used by the Catalog Management routine	
Byte 1	X...	Not used by the Catalog Management routine	
	.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.	Not used by the Catalog Management routine	
	1	DLTA	Delete an alias entry
Byte 2	1...	DRPX	Disconnect control volumes
	.1..	DELETE	Scratch generation data sets when they are uncataloged
	..XX	Not used by the Catalog Management routine	
	.... 1...	EMPTY	Remove all entries from the index when the maximum generation count has been reached
	XXX	Not used by the Catalog Management routine	

Note: Function is locate by name if all flags are zero. Function is CATBX if CTLG and BLDX flags are both ones. Function is UCATDX if UNCAT and DLTX flags are both ones.

Figure 13. User's Parameter List

Diagnostic Aids

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

Module Selection Chart

This chart, Figure 14 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
	IGG0CLC7				X	X	X	X	X	X

Figure 14. Module Selection Chart

Chart 2. Catalog Management IGC0002H (Part 1 of 2)

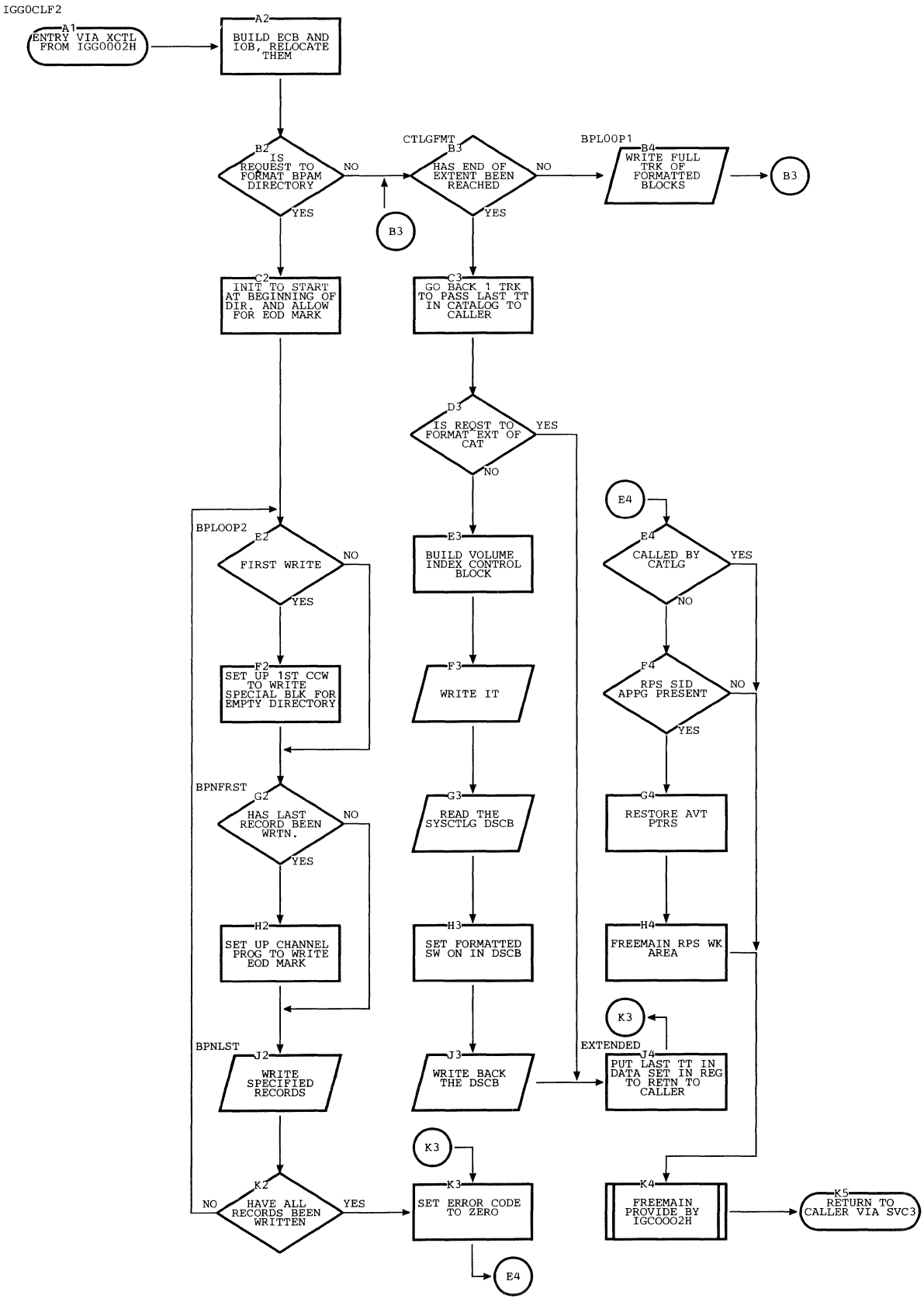


Chart 2. Catalog Management IGC0002H (Part 2 of 2)

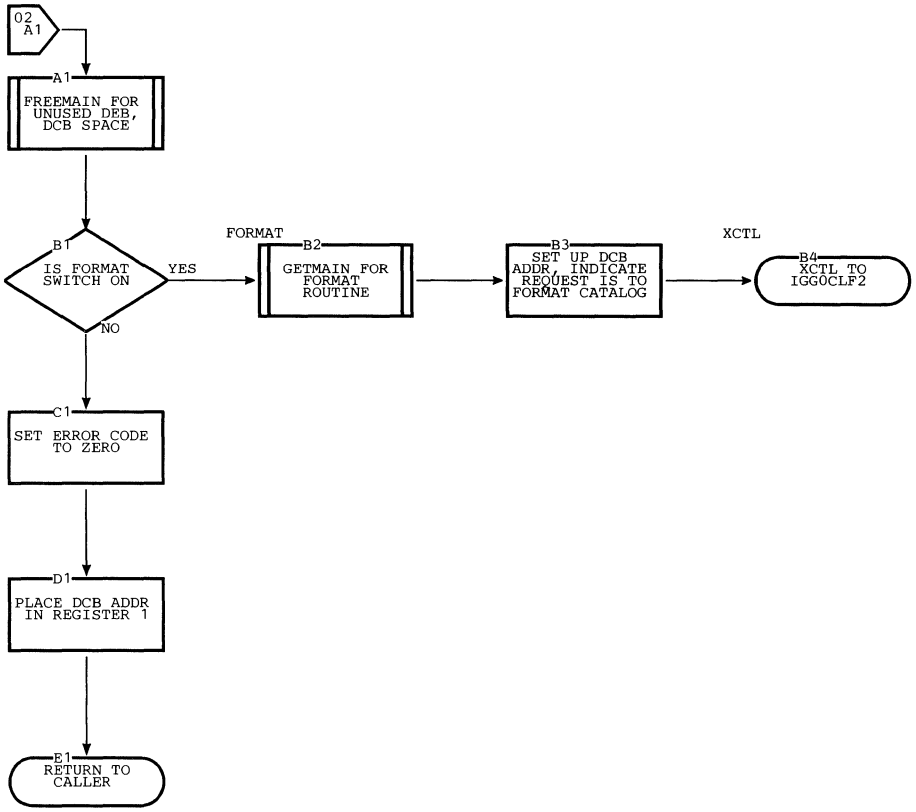
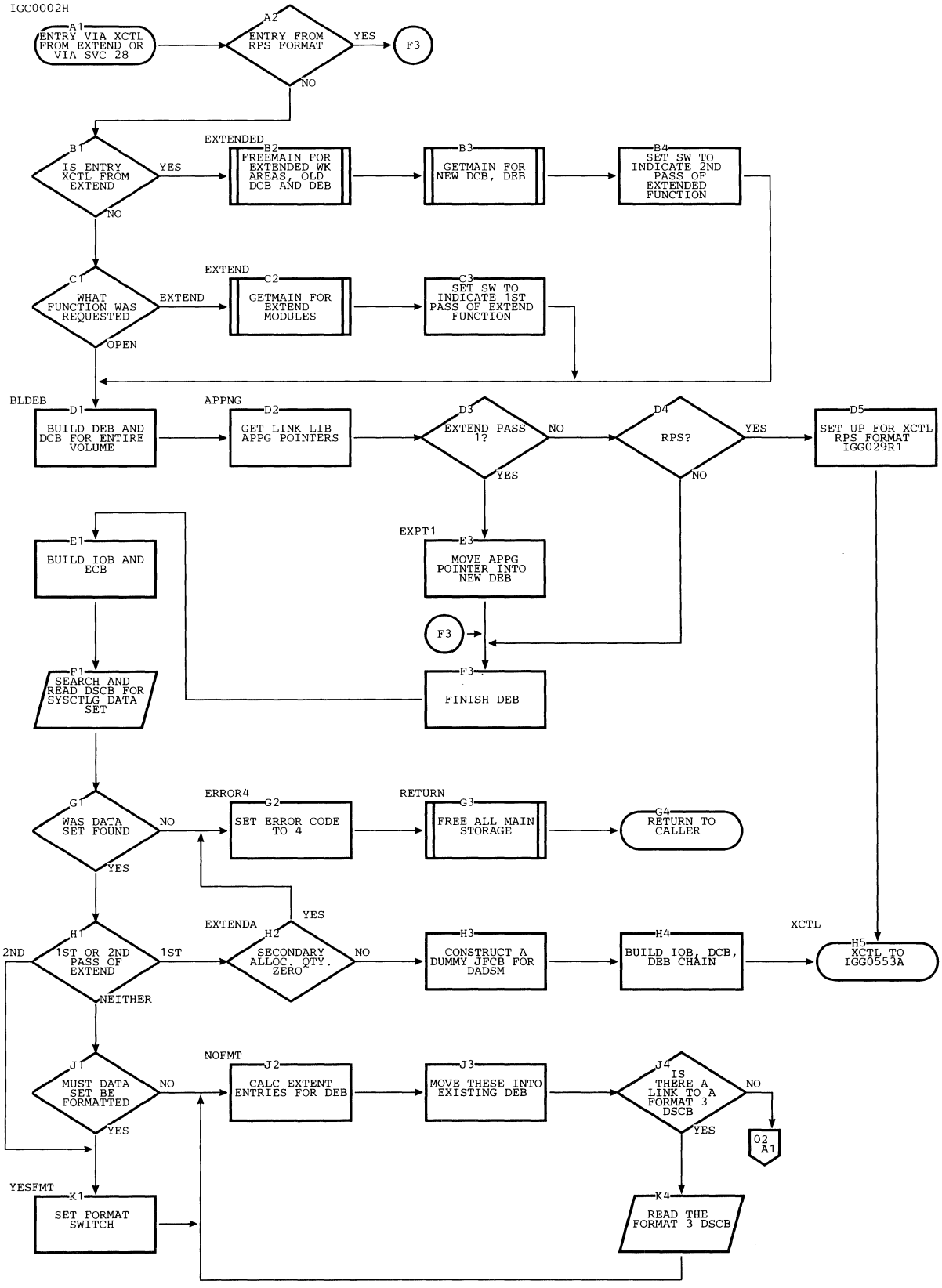


Chart 10. Catalog Management IGG0CLF2



Appendix B: Old CVOL Pointer

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 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.

Optional Features: (Byte 2; values are in hex)

X'10' Rotational Position Sensing (RPS)

Device Class: (Byte 3)

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'06' 2305 Model 1 FHSF
X'07' 2305 Model 2 FHSF
X'08' 2314 Disk Storage Facility
X'09' 3330 Disk Storage Facility

Index

Indexes to program logic manuals are consolidated in the publication IBM System/360 Operating System: Program Logic Manual Master Index, Form Y28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject in the Master Index.

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

READER'S COMMENT FORM

IBM System/360 Operating System:
TSO Catalog Management
Program Logic Manual

Order No. GY28-6745-0

Please use this form to express your opinion of this publication. We are interested in your comments about its technical accuracy, organization, and completeness. All suggestions and comments become the property of IBM.

Please do not use this form to request technical information or additional copies of publications. All such requests should be directed to your IBM representative or to the IBM Branch Office serving your locality.

- Please indicate your occupation: _____
- How did you use this publication?
  - ☐ Frequently for reference in my work.
  - ☐ As an introduction to the subject.
  - ☐ As a textbook in a course.
  - ☐ For specific information on one or two subjects.
- Comments (Please include page numbers and give examples.):

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

YOUR COMMENTS, PLEASE . . .

This manual is part of a library that serves as a reference source 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.

Note: Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Fold Fold

FIRST CLASS
PERMIT NO. 81
POUGHKEEPSIE, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY ...

IBM Corporation
P.O. Box 390
Poughkeepsie, N.Y. 12602

Attention: Programming Systems Publications
Department D58

Fold Fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

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

Cut Along Line

System/360 OS TSO Catalog Management PLM (S360-36) Printed in U.S.A. GY28-6745-0