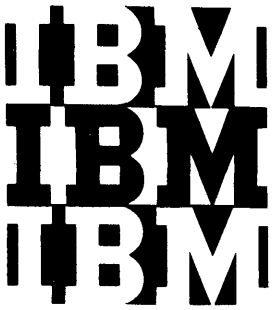


Y28-6606-0

Program Logic

IBM System/360 Operating System

Catalog Management

Program Number 360S-DM-508

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

PREFACE

The information contained in this manual is intended for programmers engaged in maintenance of catalog management routines. Because of the close relationship between catalog management and other portions of the control programs of IBM System/360 Operating System, many details in this manual are helpful in understanding overall system operations.

This manual is arranged in two sections. The first describes the structure of a catalog data set and how indexes and volume control blocks are recorded in such a data set. The second is the description of the catalog management functions and the internal logic of the routines that perform the functions.

The manual is a detailed and comprehensive guide to the internal structure and functions of the catalog management routines. It is designed to be used in conjunction with assembly listing and, consequently, does not discuss program structure at the machine instruction level.

PREREQUISITE PUBLICATIONS

IBM System/360 Operating System: Concepts and Facilities, Form C28-6535

IBM System/360 Operating System: Data Management, Form C28-6537

IBM System/360 Operating System: Introduction to Control Program Logic, Program Logic Manual, Form Z28-6605

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

Copies of this and other IBM publications can be obtained through IBM Branch Offices.

A form for readers' comments appears at the back of this publication. It may be mailed directly to IBM. Address any additional comments concerning this publication to the IBM Corporation, Programming Systems Publications, Department D58, PO Box 390, Poughkeepsie, N. Y. 12602

CONTENTS

INTRODUCTION 5

The Catalog. 5

Catalog Management Routines. 5

CATALOG STRUCTURE. 6

Index Contents 7
Control Entries 9
Pointer Entries 10

The Volume Control Block Contents. . . . 11
A Sample Catalog. 12

CATALOG MANAGEMENT ROUTINES. 14

The Locate Function (Modules IGC0002F and IGG0CLC4) 15

The Index Function (Modules IGC0002F, IGG0CLC2, and IGG0CLC3) 16

The Catalog Function (Modules IGC0002F, IGG0CLC2, IGG0CLC3, IGG0CLC4, and IGG0CLC5) 16

The CVOL Routine 17
The CVOL Routine Modules (Modules IGC0002H and IGG0CLF2) 17

INDEX. 19

ILLUSTRATIONS

FIGURES

Figure 1. Catalog Index Chaining. . . . 7
Figure 2. A Sample Catalog. 13
Figure 3. The Catalog Routine Modules. . 14
Figure 4. CVOL Routine Modules. 15

TABLES

Table 1. Summary of Index Entry Formats 8

The system for identifying the volumes on which data sets are recorded consists of two parts: the catalog and the catalog management routines. The catalog is one or more data sets each of which is recorded on a separate direct-access volume. The catalog contains the information necessary to identify the volumes on which other data sets reside. Catalog management routines provide access to the catalog and update the catalog as necessary.

THE CATALOG

A direct-access volume that contains a catalog data set is called a control volume. In any operating system, the system residence volume is always a control volume because a catalog data set is created on it at system generation time. Each catalog data set is named SYSCTLG. One such data set can serve as the catalog of an operating system. If there is more than one catalog data set in an operating system, each data set can be separate and distinct, or several data sets can be logically connected.

Catalog data sets contain two types of unit: indexes and volume control blocks. Both are variable-length records recorded in one or more fixed-length blocks. An index is a series of variable-length entries that identify other indexes, control volumes, or volumes of data sets by name and physical location. A volume control block contains volume identification for a data set when the data set extends over more volumes (five) than can be easily identified in an index. Volume control blocks, therefore, supplement information in indexes and are identified by entries in indexes.

There are three types of index: the volume index, the generation index, and the normal index. Every control volume is initialized by catalog management routines with a basic index structure called the volume index. This is all the structure that is required to catalog data sets with simple names. Generation indexes contain the identification of data sets of generation data groups. They allow the automatic cataloging functions provided for generation data groups as described in the publication IBM System/360 Operating System: Data Management. Normal indexes facilitate data separation among several functions or departments by providing names that are used to qualify simple data set names. For example, two departments, A and B, can create two indexes named A and B, respectively. Both departments can give the name DATA to one of their data sets and refer to it without ambiguity by using their indexes as qualifying names (i.e., A.DATA, and B.DATA).

CATALOG MANAGEMENT ROUTINES

The catalog management routines provide the functions for the LOCATE, INDEX, and CATALOG macro-instructions described in the publication IBM System/360 Operating System: System Programmer's Notebook, Form C28-6550. For the locate function, catalog management finds volume information of a cataloged data set by searching the index structure for the qualified name of the data set. It can also read an index block, if provided with the address of the block. For the index function, catalog management inserts or deletes indexes in a catalog data set. For the catalog function, catalog management can enter or delete the identification of the volumes of a data set.

Components of the operating system use the catalog management routines when cataloging operations are requested in data definition (DD) statements of the job control language. Utility programs also use catalog management routines to build a catalog in an operating system. The locate function of catalog management is used by job management to find a cataloged data set when the data set is requested as input to a program. The locate operation includes calculation of new generation numbers of generation data sets. The catalog function is used by job management to catalog a data set after it has been created. The index function of catalog management is used by utility programs to build indexes in a catalog data set.

An index is a record of indefinite length consisting of a series of fields or entries. The record is open-ended to accommodate new entries. The entries are in ascending sequence of the binary value of their name fields, and they are contiguous to one another at the beginning of the index.

To be recorded on a direct-access volume, the records are divided into fixed-length blocks. Each block consists of an eight-byte key and a 256-byte data portion. Index entries are placed in the data portion of the block, and the key is used to identify the block. An index may extend over several blocks, but two indexes can not share a block. Blocks of an index are chained together in the proper sequence by addresses at the end of each block. When processing an index, the catalog management routines begin with the first block of the index and, through chain addresses, continue to the last block.

The first field in the data portion of an index block is two bytes long and contains the count of the number of used bytes in the data portion. That field is necessary because the lengths of index entries vary, and an index block will not always be filled (i.e., there may be unused bytes at the end of the block's data portion). Index entries are placed in the data portion following the byte count field.

Each entry in an index is either a control entry or a pointer entry. Control entries contain information about the index in which they appear, such as the address of the first and last blocks assigned to the index or the chain address to the next block. Pointer entries contain the names of other index levels and the identification of the entity (index address, control volume, volumes of a data set, or a volume control block) to which each name applies.

The first entry in the first block of an index is a control entry. The entry indicates the address of the last block of the index and the number of unused bytes at the end of that block. The special control entry at the beginning of the volume index indicates the first available block in the catalog data set not assigned to any index. The last entry in an index block is usually a control entry that contains the address (the chain address) of the next block in the index. If two blocks are adjacent and if the blocks are in the proper logical sequence in an index, the control entry can be eliminated from the end of the first of the two blocks. Pointer entries are placed in the data portion of an index block following the control entry at the beginning of the index and up to the control entry at the end of each block. The last entry in the last block of any index is a control entry with a chain address of zero.

The key of each block contains the name that appears in the last entry (control or pointer) in the data portion of the block. Since entries are placed in the index in ascending sequence of their name values, the name in the key will always be the one of highest binary value in the block.

A volume control block, like an index, is recorded in one or more blocks with 8-byte keys and 256-byte data portions. If a volume control block extends over more than one block, the blocks are chained together. One block of a volume control block contains up to 20 volume serial numbers for one data set. The key of each block contains the hexadecimal number FF.

Figure 1 shows a method of chaining blocks that provides ease of access to catalog entries. Blocks A, B, C, D, and E are all of one index. The entries are placed in the index in the sequence of the binary value of the name fields of the entries, from name A to name Z. The key of each block contains the eight-byte name that is the last entry in the data portion of the block.

Blocks A, B, and C are contiguous. The last entry in block C is an index link entry that indicates block D as the next block of the index. Blocks C and D are separated by blocks of another index or by volume control blocks, and the name in the key of block C is, therefore, hexadecimal FF. Blocks D and E are contiguous, and the last entry in block E is the index link entry indicating that no block follows block E. The name in the key of block E is hexadecimal FF, and the address in the last entry of block E is zero. This system of recording indexes allows the catalog management routines to use a channel program to search the keys for a name greater than or equal to a name to be found.

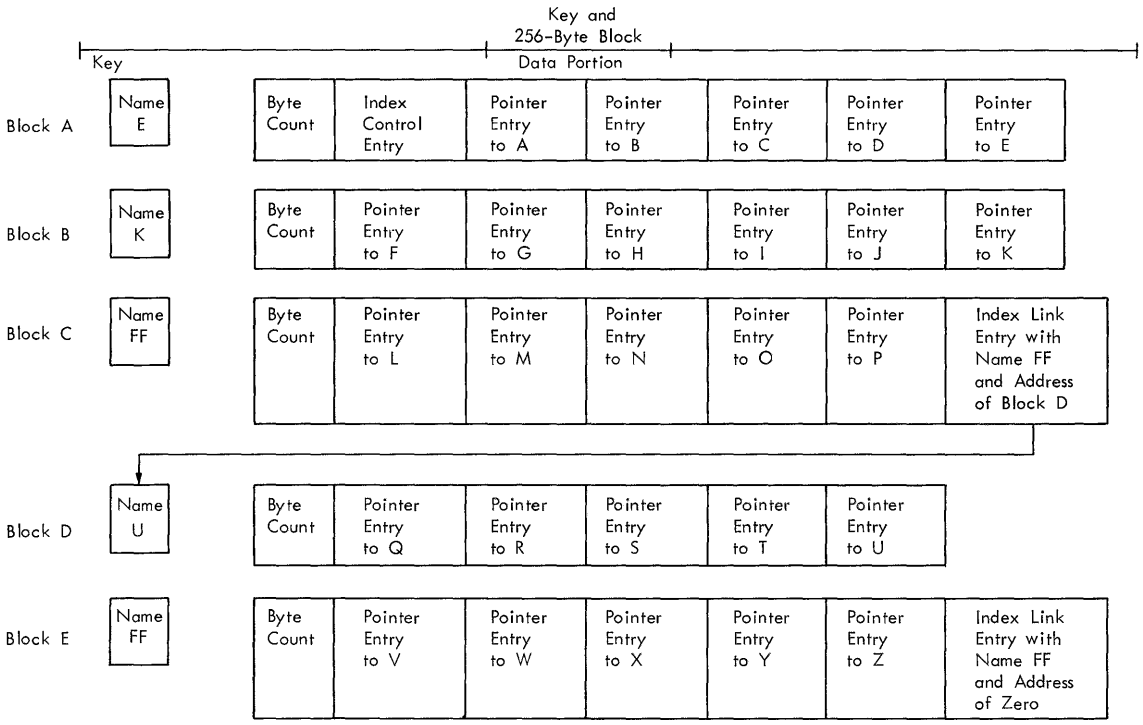


Figure 1. Catalog Index Chaining

Except for the first volume index block, index blocks or blocks of volume control blocks have no assigned position in a catalog data set. Blocks of the same index or volume control block are not necessarily contiguous; they may be chained together. The address of the first available block in a catalog data set is recorded in its volume index. When an index or volume control block overflows, the first available block is assigned to that index or volume control block. When an index or volume control block is reduced so that one of its blocks becomes empty, binary zeros are written in the empty block's key, and the block is returned to the pool of available blocks. If the block is closer to the beginning of the data set than any other available block, it is the first to be used when an index overflows or when a block is needed to record a volume control block. Thus, unused blocks will tend to accumulate close to the end of the data set.

INDEX CONTENTS

An index contains one or more variable-length entries of which there are two types: control and pointer. Control entries identify the beginning and end of each index and the end of each block. Control entries are used to chain blocks of the same index. Pointer entries identify one of the following:

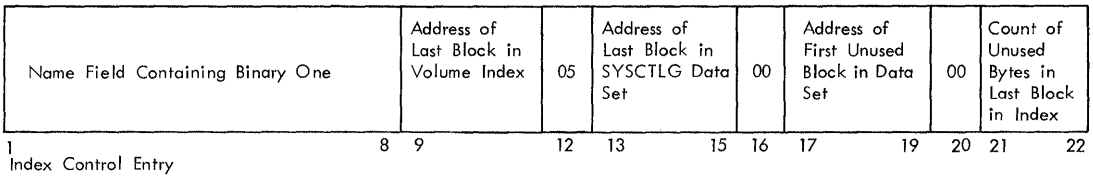
- Another index.
- A data set's volumes.
- A volume control block.
- A control volume.
- A generation index.
- An alias of an index.

All indexes do not contain all types of pointer entries. Normal indexes can not contain entries that identify control volumes or aliases of indexes. Generation indexes can contain only those entries that identify data set volumes or volume control blocks. Volume indexes can contain any type of pointer entry. Names appearing in entries in volume indexes are called high level names, and indexes identified by such entries are called high level indexes. Table 1 is a graphic summary of all the entries. The entries are described in the following paragraphs.

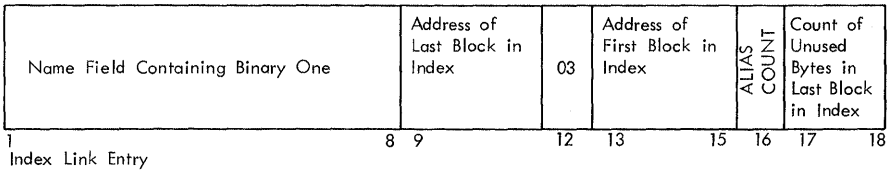
Table 1. Summary of Index Entry Formats

CONTROL ENTRIES

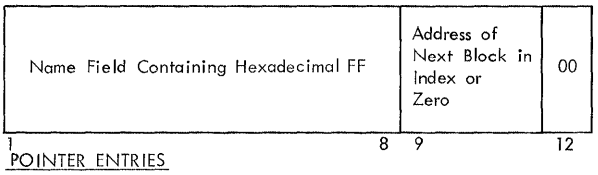
Volume Index Control Entry



Index Control Entry

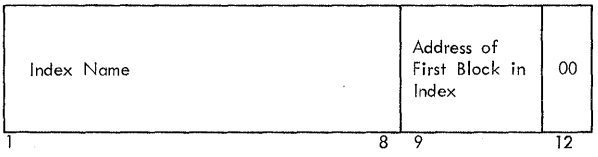


Index Link Entry

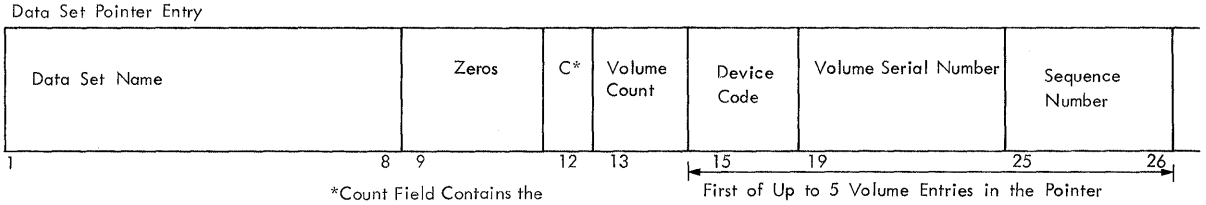


POINTER ENTRIES

Index Pointer Entry

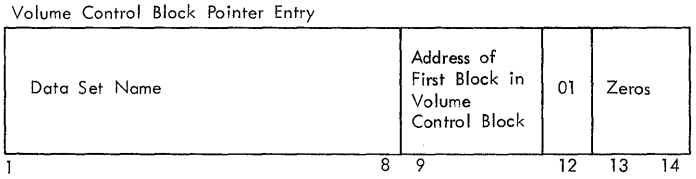


Data Set Pointer Entry

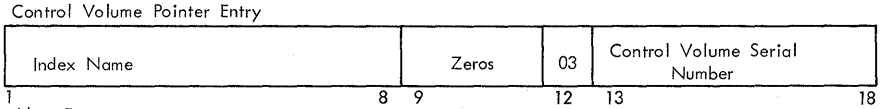


*Count Field Contains the Number = 6 (number of volume entries) +1

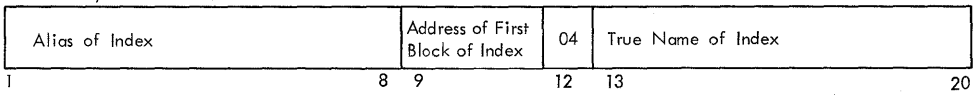
Volume Control Block Pointer Entry



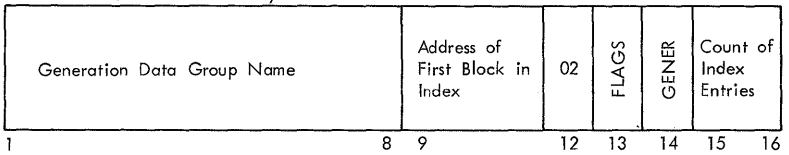
Control Volume Pointer Entry



Alias Entry



Generation Index Pointer Entry



FLAGS= 01 for EMPTY, 02 for DELETE
03 for EMPTY and DELETE
GENER= Maximum number of entries allowed in index.

CONTROL ENTRIES

A volume index control entry is always the first entry in a volume index. It consists of the ending address of the volume index, the amount of available space in the volume index, and the address of the first available block in the catalog data set. It appears only in volume indexes. The entry is 22 bytes long and contains 8 fields.

Field 1: Name Field (8 bytes long) -- This field contains the binary number one to ensure that the entry will appear as the first entry in the first block of the index.

Field 2: Last Block Address (3 bytes long) -- This field contains the relative track address of the last block in the volume index. The address is in the form TTR, where TT is the track address and R is the block address.

Field 3: Half-word Count (1 byte long) -- This field contains the binary number five to indicate that five half-words follow this field in this entry.

Field 4: Catalog Upper Limit (3 bytes long) -- This field contains the relative track address of the last block in the catalog data set. The address is in the form TTR.

Field 5: Zero Field (1 byte long) -- This field contains binary zeros.

Field 6: First Available Block Address (3 bytes long) -- This field contains the relative track address of the unused block in the catalog that is closest to the beginning of the catalog data set.

Field 7: Zero Field (1 byte long) -- This field contains binary zeros.

Field 8: Unused Bytes in Last Block (2 bytes long) -- This field contains the binary count of the number of unused bytes in the last block of the volume index.

An index control entry is always the first entry in any index except in volume indexes. It indicates the amount of space available in the last block of the index, and the addresses of the first and last blocks assigned to the index. The entry is 18 bytes long and contains 6 fields.

Field 1: Name Field (8 bytes long) -- This field contains the binary number one to ensure that this entry will be the first entry in the first block of the index because it has the lowest binary name value.

Field 2: Last Block Address (3 bytes long) -- This field contains the relative track address of the last block assigned to the index. The address is in the form TTR.

Field 3: Half-word Count (1 byte long) -- This field contains the binary number three to indicate that three half-words follow this field in this entry.

Field 4: Index Lower Limit (3 bytes long) -- This field contains the relative track address of the block in which this entry appears. The address is in the form TTR.

Field 5: Number of Aliases (1 byte long) -- This field contains the binary count of the number of aliases to this index. If this index is not a high level index, this field must be zero.

Field 6: Unused Bytes in Last Block (2 bytes long) -- This field contains the binary count of the number of unused bytes remaining in the last block in the index.

An index link entry is the last entry in either the last block of an index or any block not physically contiguous to the next block of the same index. When the entry is in the last block of the index, it contains the hexadecimal number FF and an address field of zeros. When the entry is in a block that is not physically contiguous to the next block of the index, it contains the hexadecimal number FF and the address of the next block of the index. The entry is 12 bytes long and contains 3 fields.

Field 1: Name Field (8 bytes long) -- This field contains the hexadecimal number FF. That number ensures that this entry will appear as the last entry in any index block because it has the highest name value.

Field 2: Link Address (3 bytes long) -- This field contains the relative track address of the next block in the same index, if there is a next block in the index. Otherwise, the field contains binary zeros.

Field 3: Half-word Count (1 byte long) -- This field contains the binary number zero to indicate that this field is followed by no fields in this entry.

POINTER ENTRIES

An index pointer entry can appear in all indexes except generation indexes. It contains the name of an index in the same SYSCTLG data set and the relative address of the first block of that index. The entry is 12 bytes long and contains 3 fields.

Field 1: Name Field (8 bytes long) -- This field contains the name of the index being pointed to by field 2.

Field 2: Index Address (3 bytes long) -- This field contains the relative track address of the first block of the index named in field 1. The address is in the form TTR.

Field 3: Half-word Count (1 byte long) -- This field contains the binary number zero to indicate that no fields follow this field in this entry.

A data set pointer entry can appear in any index. It contains the simple name of a data set and from one to five 12-byte fields that each identify one volume on which the named data set resides. If the data set pointer entry appears in a generation index, the generation number and version number together form the simple name of the data set in the entry. The generation, but not the version number, of the data set in the entry is the ones complement of the true generation number. If the data set resides on more than five volumes, a volume control block must be used to point to the data set's volumes. The volume control block is identified by a volume control block pointer entry, not a data set pointer entry.

The data set pointer entry is variable-length. The length is determined by the formula (14+12m), where m is the number of volumes containing the data set. The variable m can be from 1 through 5. The entry can appear in any index, and it contains 5 fields.

Field 1: Name Field (8 bytes long) -- This field contains the simple name of the data set whose volumes are identified in field 5 of this entry. The simple name of a generation data set consists of the ones complement of the data set's true generation number followed by the true version number.

Field 2: Address Field (3 bytes long) -- This field contains the binary number zero.

Field 3: Half-word Count (1 byte long) -- This field contains the binary count of the number of half-words that follow this field in this entry. The number is found by the formula (6m+1), where m is the number of volumes upon which the data set resides. The variable m can be from 1 through 5.

Field 4: Volume Count (2 bytes long) -- This field contains the binary count of the number of volumes identified in field 5 of this entry.

Field 5: Volume Entries (12 to 60 bytes long, one to five entries) -- This field contains from one to five 12-byte entries. Each entry contains the device code, serial number and sequence number of one volume containing the data set named in field 1. The device code is 4 bytes long, the volume serial number is 6 bytes long, and the sequence number is 2 bytes long. The sequence number indicates the position of the data set on a tape volume. The sequence number is zero for direct-access volumes.

A volume control block pointer entry can appear in any index. It contains the simple name of the data set and the address of the first or only physical block of the volume control block for the data set. The volume control block can identify up to 20 volumes. When this entry appears in a generation index, the generation number and version number together form the simple name of the data set. The generation, but not the version number, is the ones complement of the actual EBCDIC coded generation number. The entry is 14 bytes long and contains 4 fields.

Field 1: Name Field (8 bytes long) -- This field contains the simple name of the data set identified by this entry. The data set resides on the volumes whose serial numbers are given in the volume control block pointed to by field 2.

Field 2: Address Field (3 bytes long) -- This field contains the relative track address of the volume control block identifying the volumes containing the data set named in field 1. The address is in the form TTR.

Field 3: Half-word Count (1 byte long) -- This field contains the binary number one to indicate that one half-word follows this field in this entry.

Field 4: Zero Field (2 bytes long) -- This field contains binary zeros.

A control volume pointer entry can appear only in volume indexes. It contains the name of a high level index name in the volume index on another control volume. It is used in installations whose catalogs extend over several volumes. A control volume pointer entry in one catalog data

set directs catalog searches from one control volume to another that contains the desired index. (A high level index is one that is identified by an entry in a volume index. The name of a high level index can have no qualifiers.) It also contains the serial number of the control volume containing that volume index. The entry is 18 bytes long and contains 4 fields.

Field 1: Name Field (8 bytes long) -- This field contains a high level name that appears in the volume index of the control volume identified in field 4.

Field 2: Address Field (3 bytes long) -- This field contains binary zeros.

Field 3: Half-word Count (1 byte long) -- This field contains the binary number three to indicate that three half-words follow this field in this entry.

Field 4: Control Volume Serial Number (6 bytes long) -- This field contains the volume serial number of the control volume whose volume index contains an entry identifying the high level name in field 1.

An alias entry can appear in volume indexes only. It contains an alias given to an index, the address of the index, and the true name of the index. An alias can be given only to indexes, and only if they are high level indexes. An alias entry is 20 bytes long and contains 4 fields.

Field 1: Name Field (8 bytes long) -- This field contains the alias of the high level index identified in field 2.

Field 2: Address Field (3 bytes long) -- This field contains the relative track address of the first block of the index named in field 4. The address is in the form TTR.

Field 3: Half-word Count (1 byte long) -- This field contains the binary number four to indicate that four half-words follow this field in this entry.

Field 4: True Name Field (8 bytes long) -- This field contains the name of the index to which was given the alias name that appears in field 1. The address of the index is in field 2.

A generation index pointer entry can appear in all indexes except generation indexes. It contains the name of a generation index, the address of the first block of the index, and control information about the generation data group identified in the index. The entry is 16 bytes long and contains 6 fields.

Field 1: Name Field (8 bytes long) -- This field contains the name of the generation index whose address is given in field 2.

Field 2: Address Field (3 bytes long) -- This field contains the relative track address of the generation index named in field 1. The address is in the form TTR.

Field 3: Half-word Count (1 byte long) -- This field contains the binary number two to indicate that two half-words follow this field in this entry.

Field 4: Flags (1 byte long) -- This field contains flags that govern the uncataloging of data sets from the index as specified by the DELETE or EMPTY options of the INDEX macro that were used when the generation index was created. The options and their hexadecimal codes are as follows:

EMPTY=01 DELETE=02 EMPTY and DELETE=03

Field 5: Maximum Generations Allowed (1 byte long) -- This field contains the binary count of the maximum number of generations allowed in the index at one time as specified in the INDEX macro when the index was created.

Field 6: Current Generation Count (2 bytes long) -- This field contains the binary count of the number of generations cataloged in the index.

THE VOLUME CONTROL BLOCK CONTENTS

The format of a volume control block is always the same. Each block of a volume control block contains up to twenty volume serial numbers for one data set. Each block is 256 bytes long and contains five fields. The fields are described in the following paragraphs.

Field 1: Number of Volumes (2 bytes long) -- This field contains the binary count of the number of volume serial numbers contained in this physical block and in all subsequent blocks chained to this block. If a data set is on 61 volumes, for example, it will have four volume control blocks. The first fields of those blocks will contain 61,41,21, and 1, respectively.

Field 2: Volume Serial Numbers (240 bytes long) -- This field contains up to 20 entries of 12 bytes each. Each entry identifies one volume of the data set by giving the volume's device code, serial number, and data set sequence number. The device code is 4 bytes long, the serial number is 6 bytes long, and the sequence number is 2 bytes long. The sequence number is zero for direct-access volumes.

Field 3: Zero Field (10 bytes long) -- This field contains binary zeros.

Field 4: Chain Address (3 bytes long) -- This field contains the relative track address of the next block of this volume control block, if additional blocks exist. The address is in the form TTR. If no next block exists, the field contains binary zeros. If this field is not zero, this block must contain twenty 12-byte fields identifying volumes of the data set.

Field 5: Zero Field (1 byte long) -- This field contains binary zeros.

A SAMPLE CATALOG

Figure 2 shows a sample catalog that consists of two catalog data sets: one on the system residence volume and the other on a control volume. The catalog contains volume identification for the following data sets:

Q
A.B.M(alias B.B.M)
D.B
E
A.J(alias B.J)
A.G(alias B.G)
D.C
D.A.B
A.B.K(alias B.B.K)
A.B.N(alias B.B.N)
D.A.C

It also identifies the following generation data sets:

F(0)
F(-3)
F(-1)
D.A.D(0)
F(-2)

System Residence Volume

Volume Index

Index Pointer Entry to the Index A	Control Volume Pointer Entry to Volume "X" Whose Volume Index Contains the Index Name D	Alias Entry to Index A that Indicates B as an Alias to A	Data Set Pointer Entry to Data Set Q	Volume Control Block Pointer Entry to Volume Control Block E for Data Set E	Generation Index Pointer Entry to Index F

Index A

Index Pointer Entry to Index B	Volume Control Block Pointer Entry for Data Set G	Data Set Pointer Entry to Data Set J

Index B

Data Set Pointer Entry to Data Set K	Data Set Pointer Entry to Data Set M	Data Set Pointer Entry to Data Set N

Volume Control Block for Data Set G

Volume Identification for Volumes Containing Data Set G

Volume Control Block for Data Set E

Volume Identification for Volumes Containing Data Set E

Generation Index F

Data Set Pointer Entry to Data Set That Is Latest Generation of Data Group	Data Set Pointer Entry to Data Set That Is Latest - 1 Generation	Data Set Pointer Entry to Data Set That Is Latest - 2 Generation	Volume Control Block Pointer Entry for Data Set That Is Latest - 3 Generation

Volume Control Block for Data Set F (-3)

Volume Identification for Volumes Containing the Latest - 3 Generation of Data Group F

Control Volume "X"

Volume Index

Index Pointer Entry to Index D

Index D

Index Pointer Entry to Index A	Data Set Pointer Entry to Data Set B	Volume Control Block Pointer Entry for Data Set C

Index A

Data Set Pointer Entry to Data Set B	Data Set Pointer Entry to Data Set C	Generation Index Pointer Entry to Generation Index D

Generation Index D

Data Set Pointer Entry to Latest Generation Data Set D

Volume Control Block C

Volume Identification for Volumes Containing Data Set C

Figure 2. A Sample Catalog

CATALOG MANAGEMENT ROUTINES

There are two catalog management routines: the catalog routine and the CVOL routine. The catalog routine provides the catalog, index, and locate functions. In providing the locate function, catalog management finds entries in the catalog by searching indexes specified in a qualified data set name. For the catalog function, catalog management inserts, deletes, or replaces data set pointer entries, volume-control block pointer entries, and volume control blocks. For the index function, catalog management enters into and deletes from the catalog the indexes, generation index pointer entries, index pointer entries, control volume pointer entries, and alias entries. The CVOL routine provides services to the catalog routine by opening catalog data sets for processing and by writing format blocks in new catalog data sets. The catalog routine consists of five modules: IGC0002F, IGG0CLC2, IGG0CLC3, IGG0CLC4, and IGG0CLC5. (See Figure 3.) The CVOL routine consists of two modules: IGC0002H and IGG0CLF2. (See Figure 4.) Each module is loaded separately into the transient area of main storage and is executed in the supervisor mode.

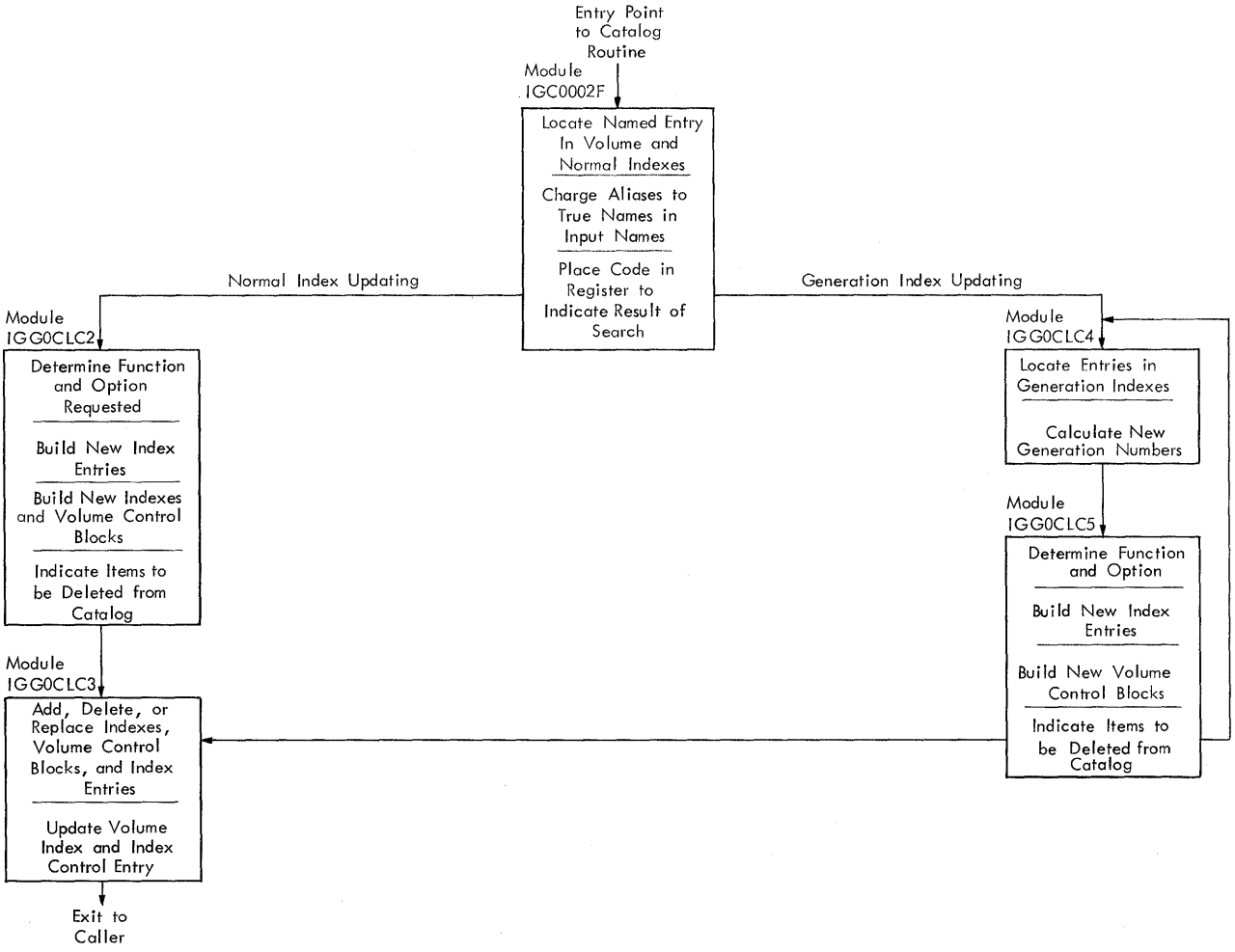


Figure 3. The Catalog Routine Modules

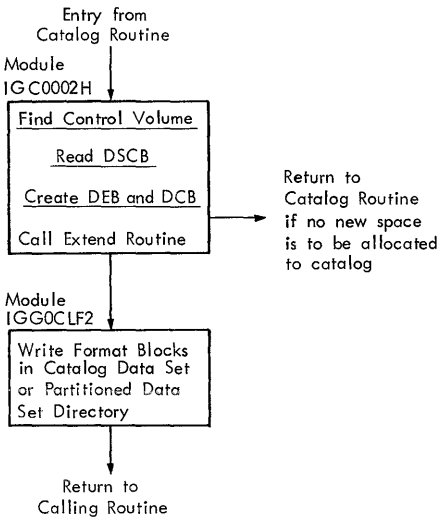


Figure 4. CVOL Routine Modules

THE LOCATE FUNCTION (MODULES IGC0002F AND IGG0CLC4)

In performing the locate function, the catalog routine finds entries in catalog data sets. The function is performed by modules IGC0002F and IGG0CLC4 of the catalog routine. Entry to the routine is made in module IGC0002F. The calling routine provides a qualified name (such as A.B.C), the function to be performed, and possibly identification of a control volume. Module IGC0002F searches for the named entry beginning at the volume index of the system residence volume, unless a control volume was specified in the input.

The search proceeds through each index level in the qualified name until all levels are found or until one of the named levels can not be found. If module IGC0002F encounters a generation index, control passes to module IGG0CLC4 to find the last level of the name.

When an index is being searched for a named entry, a channel program is constructed, and searches the keys for a name that is equal to or greater than the given name. Contiguous blocks of the index are searched by the channel program automatically. But the search program must terminate at a block that is not contiguous to the next block of the index. The termination is caused by the fact that the key name of a block not contiguous to the next is hexadecimal FF, a value greater than any name in an index. The search program can then be reinitiated at the next block of the index.

In Figure 1, for example, the locate function might be requested to find the item named F. A channel program would compare F to the key name of block A. Since E is less than F, the channel program would proceed to the next contiguous block. The key of that block would be compared to F. Since K is greater than F, the channel program would read in block B which contains, or should contain, the item named F.

When searching the index for the item named R, the channel program would compare the key names of the blocks A,B, and C to the name R. Since the key name in block C is greater than R, the channel program would read in block C. A comparision would then be made between the name R and the name in the next to the last entry in block C. Since that name, P, is not equal to or greater than R, catalog management would process the next block of the index at the address indicated in the last entry of block C. If the address in the last entry is zero, the search terminates because it has reached the end of the index.

When the locate search terminates, the module in control indicates the outcome of the search with an error code placed in a general register. The code indicates whether all levels were found, why the search failed, if it did, and what type of entry was the last found.

Module IGC0002F reads information from the catalog and passes it to the calling routine. The information passed depends on the kind of entry found. If the last entry found is a volume control block pointer entry, the entry and the control block are read from the catalog data set. If the entry is a data set pointer entry, the volume information from the entry is placed in the work area. If the last entry found is a normal or generation index pointer entry, the pointer entry and the first block of the index pointed to are placed in the work area.

Module IGC0002F always returns to the calling routine with the true name of the catalog entry that was located. For example, if S is an alias of TRUE in the catalog, and if the module finds S.T.U, the true name TRUE.T.U is returned by the locate function.

In addition to finding a named entry in the catalog, module IGC0002F reads any block in a catalog data set when the calling routine provides a relative track address instead of an entry name. A relative track address is supplied by a previous locate operation and is used to locate the second and subsequent blocks of an index. Only one block is read at one

time, and it is placed in the work area specified by the calling routine.

Module IGG0CLC4 must provide special operations to find an entry in a generation index. Generations of a group are indicated by the calling routine in one of two ways: by generation and version number (X.Y.Z.GnnnVmm) or by relative qualifier (X.Y.Z(-3)). When the data set name provided to the locate function contains the generation and version number, the module IGG0CLC4 must complement the generation number and search the proper generation index for the named entry. The volume identification provided by the entry is placed in the work area.

To find an entry specified by a relative qualifier, module IGG0CLC4 determines the required generation by the relative position of the generation's entry in the index. If the relative qualifier is zero, the latest generation is needed, and module IGG0CLC4 provides the information from the first entry in the index. If the relative qualifier is negative, the module finds the generation index entry that is the specified number of entries from the first in the index. If the relative qualifier were -2, for example, the module would find the generation index entry that was the third in the index.

If the relative qualifier were positive, indicating a new generation, module IGG0CLC4 would find the latest entry in the generation data group (first in the index) and would calculate the generation and version number of the new generation data set. The new generation and version numbers are found by the formulas:

(new generation) = (latest generation number) + (relative qualifier)
(version number) = 00

Whenever a relative qualifier is used, module IGG0CLC4 returns the corresponding absolute name of the generation. For example, if generation index A.B.C contained the latest generation numbered 25, and if module IGG0CLC4 were given the name A.B.C(0), the routine would return with the name A.B.C.G0025V00. If the module were given A.B.C(+3), the routine would return with the name A.B.C.G0028V00.

THE INDEX FUNCTION (MODULES IGC0002F, IGG0CLC2, AND IGG0CLC3)

In performing the index function, the catalog routine places into and removes from a catalog, indexes, aliases, and control volume pointer entries as well as the items those entries identify. The index function is performed by modules IGC0002F, IGG0CLC2, and IGG0CLC3. Entry to the routine is in module IGC0002F; the same entry point as that of the locate function. The calling routine specifies a name of an index, the option to be provided by the function, and possibly a control volume serial number. Other parameters must be supplied depending on the option requested. For example, if the routine is to create an alias entry, an alias name must be provided by the calling routine.

Module IGC0002F performs a locate operation as the first step in the index function. The locate operation consists of searching the catalog for the name provided in the input. At the completion of the search, a general register indicates the outcome of the search, and the work area contains the last entry found, as in any locate operation.

Module IGC0002F passes control to module IGG0CLC2. Module IGG0CLC2 interrogates the general register set by the locate operation to determine whether the requested index operation can be performed with the existing catalog. If the operation can not be performed, control returns to the caller with an indication of the error encountered. Otherwise, module IGG0CLC2 creates any new pointer entries and index blocks or sets an indication of what entry and index are to be deleted. Control passes to module IGG0CLC3.

Module IGG0CLC3 updates the index indicated by module IGG0CLC2. The updating of an index includes processing of the index to maintain ascending sequence of entry names and to eliminate "holes" created by deletion of entries. Updating also includes the assignment or release of index blocks when indexes are created or extended, or deleted or emptied.

THE CATALOG FUNCTION (MODULES IGC0002F, IGG0CLC2, IGG0CLC3, IGG0CLC4, AND IGG0CLC5)

In performing the catalog function, the catalog routine inserts into and deletes from a catalog index the data set pointer entries and volume control blocks and their pointer entries. Before a data set is cataloged, the indexes used by the catalog function must exist. The catalog function provides the CATALOG, UNCATALOG, and RECATALOG options and is performed by modules IGC0002F, IGG0CLC2, IGG0CLC3, IGG0CLC4, and IGG0CLC5. The sequence of module execution depends on whether the index to be updated is a normal (or volume) index or a generation index. Generation data sets are cata-

loged by modules IGC0002F, IGG0CLC4, IGG0CLC5, and IGG0CLC3. Other data sets are cataloged by modules IGG0002F, IGG0CLC2, and IGG0CLC3.

Entry to the catalog routine for the catalog function is in module IGC0002F. The calling routine must provide the name of the data set, the list of the data set's volume serial numbers, the option to be provided, device type, data set sequence number, and possibly a control volume serial number. Module IGC0002F locates the name of the data set by searching the catalog as in a normal locate operation. The module passes control to module IGG0CLC4 if a generation index is found. Otherwise, the outcome of the search is indicated in a general register and the entry and index found are placed in a work area, and control passes to module IGG0CLC2.

Module IGG0CLC2 interrogates the general register to determine if the requested function can be performed in the existing catalog. If not, control passes to the calling routine with an error indication. Otherwise, a new data set pointer entry or volume control block pointer entry is created by module IGG0CLC2 if cataloging was requested. If a volume control block is required, it is created by the module. If uncataloging was requested, module IGG0CLC2 sets an indication of what entry is to be deleted from the catalog. If recataloging was specified, module IGG0CLC2 creates the new entry and indicates what it is to replace in the catalog. Control then passes to module IGG0CLC3.

Module IGG0CLC3 updates the index as indicated by module IGG0CLC2. During the updating, the ascending sequence of entry names in the index must be maintained, and "holes" created by deletion of entries must be eliminated. Index processing also includes the assignment or release of index blocks when index blocks are emptied, when indexes are extended, or when volume control blocks are created or deleted.

When generation data sets are being cataloged, the catalog routine must be given the name of the data set with the generation and version numbers in the form GnnnnVmm. Modules IGG0002F and IGG0CLC4 find the generation index that is to receive the new entry and determine that the name of the new generation is not a duplicate of a name in the index. Module IGG0CLC5 creates any new index entries and volume control blocks, and module IGG0CLC3 places the new entries and blocks in the catalog or deletes entries and blocks as specified. Automatic recataloging of a data set occurs when duplicate generation numbers are found (version numbers are ignored).

When cataloging generation data sets, the catalog function may uncatalog a data set from the generation index. The uncataloging is specified by the generation index control entry that contains a number indicating the maximum number of entries allowed in the index at one time. The number is supplied by the user when the index is created. If a data set is uncataloged and the DELETE option was specified when the index was created, the catalog function calls the scratch routine of DADSM to scratch the data set. If the EMPTY option was specified and the index contains the maximum number of entries allowed by the user, cataloging of a new generation causes all existing entries to be removed from the index. Control passes from module IGG0CLC5 to module IGG0CLC4 if the EMPTY option was specified and the index overflows. Module IGG0CLC4 then creates a new generation number for the new pointer entry calculated on the basis of the empty generation index.

THE CVOL ROUTINE

The CVOL routine performs services for the catalog routine: it opens and extends catalog data sets, and writes format blocks in catalog data sets and in directories of partitioned data sets.

The open function is similar to the open function of data management. Before the catalog routine can process a catalog data set, the CVOL routine must find the DSCB of the data set and complete a DCB and DEB with extent descriptions from the DSCB. If a catalog data set has never been processed by catalog management, the CVOL routine writes format blocks in the data set at the same time that it opens it. When a catalog data set overflows its boundaries, the catalog routine requests the CVOL routine to allocate additional space to the data set and write format blocks in that space. The CVOL routine calls the extend routine of DADSM to allocate additional space and update the data set's DSCB.

THE CVOL ROUTINE MODULES (MODULES IGC0002H AND IGG0CLF2)

The CVOL routine consists of two separate modules that are loaded into the transient area of main storage at different times. As shown in Figure 4, module IGC0002H opens control volume catalog data

sets by creating a DCB and DEB from information in the data set DSCB. This module also calls the extend routine of DADSM to extend a catalog data set. Module IGG0CLF2 writes format blocks in catalog data sets with 256-byte data portions and 8-byte keys. The module also constructs a volume index in the first block of the data set and places a volume index control entry at the beginning of the index. When processing a partitioned data set directory, the module writes format blocks and places a special entry in the first block of the directory.

Address
  chain 6,12
  index 6,11
  last block 9
  link 9
  of first available block 7,9
  of generation index 11
  relative track 9-12,15
  volume index ending 9
Alias
  entries 14,16
  of an index 7,11

Block
  catalog data set last 9
  index 5,6,9-11
  key name of a 15
  key of 6
  volume control 5-7,10-11
    pointer entries 14,17
  volume index last 9

Catalog 5
  data set 5-7,9,15
    format block in 17,18
  function 14,17
    modules 16
    option 16
  macro-instructions 5
  routine 14-17
  searches 11
  upper limit 9
Chain address 6,12
Channel program 6,15
Code
  device 10,11
  error 15
Control
  entry 6,7,9
    generation index 17
    volume index 18
  volume 5-7,11,12,17
    pointer entry 11
    serial number 11,16
CVOL routine 14,15,17

DD statements 5
DELETE option 11,17
Direct-access volume 5
  sequence number of 10
Duplicate generation number 17

EMPTY option 11,17
Error code 15
Extend routine 17,18

Fields
  of control volume pointer entry 10
  of data set pointer entry 10
  of generation index pointer entry 11
  of index control entry 9
  of index link entry 9
  of index pointer entry 10
  of volume control block pointer entry 10
  of volume index control entry 9
Format block 14,17,18

Generation 10,16
  calculation of 16
  data groups 5,11
  data sets 5
    cataloging of 17
    simple name of 10
  index 5,7,10,15,16
    pointer entry 11
    relative track address of 11
  maximum number of 11
  numbers 5,10,16
    and recataloging 17

IGC0002F 14
  functions of module 15-17
IGC0002H 14
  functions of module 17,18
IGG0CLC2 14
  functions of module 16,17
IGG0CLC3 14
  functions of module 16,17
IGG0CLC4 14
  functions of module 15-17
IGG0CLC5 14
  functions of module 16,17
IGG0CLF2 14
  functions of module 18
Index 5,6,17,18
  address 6,11
  alias of an 11
  blocks 5,6,9-11
  catalog 16
  contents 7
  functions 14,16
  generation 5,10,11,16,17
    relative track address of 11
  high level 7,10
    name of 11
  link entry 9
  lower limit 9
  macro-instruction 11
  normal 5
  pointer entry 10
  processing 17
  true name of 11
  updating of an 16
  volume 5,6,9
    of system residence volume 15

Key of a block 6

Level
  high 7,9
  last 15
Link address 9

Locate
   function 5,14,15
   macro-instruction 5

Name
   alias 16
   entry 15
   fields of index entries 6
   of a generation index 11
   of an index 10,11
   of data set 5,10
   qualified 15
Normal index 5

Partitioned data set directory processing 18
Pointer
   entry 6,15,16
     control volume 10,14
     data set 10,17
     generation index 11,14
     index 10,14
     volume control block 10,14,15

Qualifier
   relative 16

Recatalog 16
Relative track address 9-12,15
Routine
   catalog 14-17
   CVOL 14,15,17
   extend 17,18
   scratch 17

Scratch program 15
Sequence number 10,11,17
SYSCTLG 5,10
System residence volume 5,12,15

Track address
   relative 9-12,15

VCB (see volume control block)
Version number 10,16,17
Volume
   control 5-7,11,12,17
   control block 2,5-7,10-12,15-17
   index 5-7,9-11,18
     block 7
     control entry 9
   serial number 6,10,11
   system residence 5,12,15

READER'S COMMENTS

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

Form Z28-6606-0

Is the material:	Yes	No
Easy to Read?	___	___
Well organized?	___	___
Complete?	___	___
Well illustrated?	___	___
Accurate?	___	___
Suitable for its intended audience?	___	___

How did you use this publication?
___ As an introduction to the subject ___ For additional knowledge
Other _____

fold

Please check the items that describe your position:

___ Customer personnel ___ Operator ___ Sales Representative
___ IBM personnel ___ Programmer ___ Systems Engineer
___ Manager ___ Customer Engineer ___ Trainee
___ Systems Analyst ___ Instructor Other_____

Please check specific criticism(s), give page number(s), and explain below:
___ Clarification on page(s)
___ Addition on page(s)
___ Deletion on page(s)
___ Error on page(s)

Explanation:

fold

Name _____

Address _____

FOLD ON TWO LINES, STAPLE AND MAIL
No Postage Necessary if Mailed in U.S.A.

fold fc

FIRST CLASS
PERMIT NO. 81

POUGHKEEPSIE, N.Y.

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

POSTAGE WILL BE PAID BY

IBM CORPORATION
P.O. BOX 390
POUGHKEEPSIE, N. Y. 12602

ATTN: PROGRAM LOGIC DOCUMENTATION
DEPT. D89

fold

IBM

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

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)