SY33-8562-5
File No.S370-30

Systems

DOS/VS LIOCS Volume 4 VSAM Logic

Program Number 5745-SC-VSM

Release 34

IBM

Sixth Edition (April 1977)

This edition, SY33-8562-5, applies to Version 5, Release 34, of the IBM Disk Operating System/ Virtual Storage (DOS/VS) and to all subsequent versions and releases until otherwise indicated in new editions or Technical Newsletters. This is a major revision of SY33-8562-4. Changes are continually made to the information herein. Before using this publication in connection with the operation of IBM systems, consult the latest edition of *IBM System/370 Bibliography,* GC20-0001, for the editions that are applicable and current.

Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Summary of Amendments

Changes are described on page iii.

This publication has been photocomposed through ATMS (an IBM Program Product) and TERMTEXT/Format (an IBM Installed User Program). For information regarding those programs, contact your IBM representative or the IBM branch office in your locality.

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 Programming Publications, Department G60, P.O. Box 6, Endicott, New York 13760. Comments become the property of IBM.

Summary of Amendments

Release 34

IBM 3330 Disk Storage, Model 11
IBM 3350 Direct Access Storage

VSAM now supports the 3330-11 and the 3350 in non-compatibility mode. All references to the 3330 now refer to the Model 11 also, unless otherwise stated.

Data Areas

The following data areas have been added to Section 5:

Command Control Block (CCB)
Channel Command Word (CCW)
CCW Skeleton DSECT (CWS)
CCW Skeletons
Close Work Area
EXCPAD Parameter List
I/O Arguments (IOARG)
I/O Driver Block (IODRB)
I/O Work Area (IOWKA)

IKQVDU Error Codes

These error codes are now documented in greater detail than previously.

Miscellaneous

Numerous miscellaneous changes have been made to increase clarity and accuracy.

Release 33

Local Shared Resources

VSAM now shares its buffers and channel programs amongst the tasks and data sets in a partition.

Control Block Manipulation Macros

The execution of these macros has been modified so as to provide better performance.

Control Block Allocation

The method used by OPEN to allocate virtual storage to the control blocks it builds for data sets being opened has been changed to provide better utilization of storage.

SHOWCAT Macro

This macro, which permits display of certain catalog information, is now supported by VSAM.

Release 32

No changes.

Preface

This logic manual is the fourth in a series of four volumes providing detailed information about the DOS/VS Logical IOCS program. The four volumes are:

Volume 1: General Information and Imperative Macros Logic, SY33-8559
Volume 2: SAM Logic, SY33-8560
Volume 3: DAM and ISAM Logic, SY33-8561
Volume 4: VSAM Logic, SY33-8562

This manual is mainly intended for persons involved in program maintenance and for system programmers who are altering the program design. Logic information is not necessary for the operation of the programs described.

This manual and the code it supports should be viewed as a maintenance set. This means that the module prologues and comments contain certain types of information and that this manual contains other kinds of information. Thus, the listings provide the description of the internal logic of modules, and the manual uses Method of Operations diagrams to show what the functions of VSAM are and how the modules work together to carry out those functions. The term *data set* is used in this manual instead of *file* to conform to the program listings.

Effective use of this publication requires an understanding of system operation, PL/S language, assembler language, and its associated macros.

Organization of This Publication

This publication is organized in the following manner:

- *Section 1. Introduction*, which describes the major components of VSAM.
- *Section 2. Method of Operation*, which describes the functions performed by VSAM.
- *Section 3. Program Organization*, which describes the information contained in VSAM program listings and the relationship of the program structures to the issued macro.
- *Section 4. Directory*, which contains lists of phases, components, modules, routines, catalog external entry points, and data areas.
- *Section 5. Data Areas*, which describes control blocks used by VSAM and describes VSAM data, index, and catalog records.
- *Section 6. Diagnostic Aids*, which contains diagnostic aids, such as error codes.
- *Glossary*, which defines terms relevant to VSAM.

- *Index*, which is a subject index to the publication.

Required Publications

The following publications should be read and understood before using this publication:

DOS/VS Data Management Guide, GC33-5372, which describes VSAM data management.

DOS/VS Supervisor and I/O Macros, GC33-5373, which tells how to code VSAM macros in application programs.

DOS/VS LIOCS Volume 1: General Information and Imperative Macros Logic, SY33-8559, which contains brief descriptions of how DOS/VS open, close and end-of-volume routines interact with VSAM.

Related Publications

Other publications that may be of interest in conjunction with this manual are:

DOS/VS Access Method Services User's Guide, GC33-5382, which explains how to use Access Method Services commands to carry out various utility operations.

DOS/VS Access Method Services Logic, SY33-8564, which documents the logic of Access Method Services.

DOS/VS System Management Guide, GC33-5371, which explains how DOS/VS is organized and how system facilities can be utilized.

DOS/VS System Control Statements, GC33-5376, which describes JCL as it relates to VSAM.

DOS/VS DASD Labels, GC33-5375, which describes VSAM label formats and label processing.

DOS/VS Messages, GC33-5379, which includes all messages originated by VSAM.

DOS/VS Serviceability Aids and Debugging Procedures, GC33-5380, which describes how to determine and correct errors in programs that use VSAM; it also discusses VSAM use of problem determination aids.

DOS/VS System Generation, GC33-5377, which describes how to include VSAM in a DOS/VS system.

Using This Publication

This publication is designed to be used with the VSAM program listings. The diagrams in *Method of Operation* describe the major functions performed by VSAM; these diagrams are intended to be your key to a module name (and

routine name, as appropriate) in the listing. See the *Method of Operation* chapter for a description of how to read these diagrams. For information on what is available in the program listings, see the chapter *Program Organization*.

The module directory in the *Directory* chapter lists the modules by symbolic name (all of which start with IKQ, IIP, IGG0, or $$B) and contains page references to the appropriate method of operation diagram or program structure that applies to each module. If you wish to see how modules are grouped according to component (such as open, record management, etc.) see the component directory. The routine directory, where relevant, further shows how the modules are subdivided into routines.

The *Directory* chapter also contains the names of the catalog external entry points (which start with IGGP). These external entry points are cross-referenced in the module directory by module name. As a further aid, charts showing program flow for each catalog module are contained in the *Program Organization* chapter. The charts are numbered sequentially by an alphameric code that corresponds to the last two characters of the symbolic module name, for example, module IGG0CLAG is flowcharted in Chart AG.

Contents

Illustrations

Figures

Method of Operation Diagrams

Catalog Program Flowcharts

Acronyms and Abbreviations

AC	allocation chain
ACB	access method control block
ACC	AMS Catalog communication area
ACE	argument control entry
ADDR	address
ADR	addressed accessing
AIX	alternate index
AMBL	access method block list
AMCBS	access method control block structure
AMDSB	access method data statistics block
AMDTF	access method define the file (ISAM only)
ANCHT	anchor table
ARDB	address range definition block
AU	allocation unit
BCB	buffer control block
BCR	base cluster record
BHD	buffer header
BKPHD	block pool header
BKWD (also BWD)	backward
BLK	block
BSPH	buffer subpool header
BUFF	buffer
BUFH	buffer header
CA	control area
CAT	catalog
CAXWA	catalog auxiliary work area
CB	control block
CCA	catalog communications area
CCB	command control block
CCR	catalog control record
CI or CNV	control interval
CIDF	control interval definition field
CINV	control interval
CIWA (also CIW)	control interval work area
CM	catalog management
CMS	catalog management services
CNV or CI	control interval
COMREG	communications region (DOS)
CP	channel program
CPA	channel program area
CPAH	channel program area header
CPL	catalog parameter list
CRA	catalog recovery area
CTGFL	catalog field parameter list
CTGFV	catalog field vector table
CTGPL	catalog parameter list
DADSM	direct-access device space management
DDname	data definition name
DIR	direct processing
DLBL	DASD label
DOS/VS	disk operating system/virtual storage
DS	data set
DSCB	data set control block (in DOS, DASD label)
DSN (also DSNAME)	data set name
DSORG	data set organization
DTF	define the file
ECB	event control block
EDB	extent definition block
EOD	end of data
EOF	end of file
EOV	end of volume
ESDS	entry-sequenced data set
EXCP	execute channel program
EXLST	exit list

FCDB	field control and data block
FKS	full key search
Fn	format n
FPL (also FL)	field parameter list
FS	free space
FVT	field vector table
FWD	forward
GEN	generic key search
GO	group occurrence
GOP	group occurrence pointer
ID	identifier
IIP	ISAM interface program
I/O	input/output
ISAM	indexed sequential access method
JIB	job information block
KEQ	search on key equal
KEY	keyed accessing
KGE	search on key greater or equal
KRDR	key range determination routine
KSDS	key-sequenced data set
KWTC	keyword type code
LOC	locate
LPMB	logical-to-physical mapping block
LRD	last record
LUB	logical unit block
MVE	move
n	number
NSP	note string position
NUB	no user buffer
NUP	no update
OAL	open ACB list
O/C/EOV	open/close/end of volume
OPNWA	open work area
PIB	program information block
PL/S	programming language/system
PLH	placeholder
PSW	program status word
PT or PTR	pointer
PUB	physical unit block
RAB	record area block
RBA	relative byte address
RDF	record definition field
REP	replication
Rn	register n
RPHD	resource pool header
RPL	request parameter list
RRDS	relative-record data set
RSCB	resource sharing control block
SCIB	search compressed index block
SEOF	software end of file
SEQ	sequential
SKP	skip sequential
SS	sequence set
SVC	supervisor call
THB	track hold block
TIC	transfer in channel
UBF	user buffer
UCAT	user catalog
UPD	update mode (or data modify)
USB	upgrade set block
USVR	user security verification routine
VOLID	volume identification
VRPPL	BLDVRP parameter list
VSAM	virtual storage access method
VSRT	VSAM shared resource table
VTOC	volume table of contents
WA	work area

Section 1. Introduction

Virtual Storage Access Method (VSAM) is an access method that operates under DOS/VS. VSAM is used with direct-access storage to provide fast storage and retrieval of data.

VSAM is divided into modules, which are logically grouped into the following components:

- Control block manipulation, which allows the user program to create, modify, display, and test the contents of some VSAM control blocks (the ACB, EXLST, and RPL, which are described under *Data Areas* in this publication), and to build or delete a VSAM resource pool.
- Open, which connects a user's program to a VSAM data set and builds the control blocks required to permit the user to read from and write to the data set.
- ISAM interface, which allows the user program to issue ISAM macro instructions to process records in a VSAM data set.
- Catalog management, which writes and updates catalog records. Catalog management processes the catalog to obtain information for Open, Close, end-of-volume, and Access Method Services.
- DADSM, which allows the system to maintain VTOC records for data sets. In VSAM, DADSM is used by the catalog to create and delete data spaces, both unique and nonunique.
- Record management, which reads and writes records in response to user-issued VSAM and ISAM macro instructions. This component also reads and writes records for the catalog management component.
- End of Volume, which mounts volumes and allocates space. End of Volume modifies the existing control blocks to reflect the newly mounted volumes and newly allocated space.
- Close, which disconnects a user's program from a data set and releases the data set's control blocks built by Open. Close also updates statistics in the VSAM catalog.
- Service aids, which enable program maintenance and Field Engineering personnel to obtain dumps, maintain DSCBs, and load phases.

For a list of the modules in these components, see the *Directory* in this publication.

Section 2. Method of Operation

Reading Method of Operation Diagrams

Method of operation diagrams depict the internal functions of a programming system, in this case, an access method. The internal functions are categorized by the macro instructions issued by the user, such as the GENCB, MODCB, OPEN, GET, PUT, CLOSE and ENDREQ macro instructions.

Diagram AB shows the basic organization of the method of operation diagrams according to the macro instructions mentioned above. References lead from the high-level charts showing subfunctions required to carry a request to its completion.

Note the relationship of function (exemplified by the macro instructions) to component. Starting with an OPEN issued by the user, a logical progression is made from Open modules to supporting Catalog modules. When a record management macro instruction such as PUT is issued by the user, not only the Record Management modules are involved (which include modules that perform buffer and I/O management and end-of-volume processing) but the Catalog modules which, in turn, call upon the DADSM modules for space management.

The diagram contain three blocks of information: input, processing, and output. The left-hand side of the diagram shows the data that serves as input to the processing steps on the center of the diagram, and the right-hand side shows the data that is output from the processing steps. Input is anything significant that program processing steps refer to or get. Processing is the steps that support the function or subfunction represented by the diagram. Output is any significant change effected by a processing step, for example, register contents, or control blocks created or modified. The processing steps are numbered and the numbers correspond to notes, if any, on the pages following the appropriate diagram(s). If notes are given, they include references to modules, routines, and/or labels shown on the extreme right-hand side of the diagram. These references are your link to the program listings. Figure 2 shows the symbols used in these diagrams and describes their meaning.

As an example of how to interpret a typical method of operation diagram, see page 2 of Diagram IA, which graphically depicts the CLOSE/TCLOSE functions. The left-hand side of the diagram shows the significant input required by the processing steps shown in the diagram. For example, register 0 points to a list of DTF pointers for an ISAM user and ACB pointers for a VSAM user. The data-set information in the ACB is input to steps 5 and 7 in the processing portion of the diagram. The processing portion of the diagram shows the processing steps required to fulfill the function described by the diagram. Note that the function described by one diagram may be performed by one or more VSAM modules; that is, the diagrams not only show program flow, but show the subfunctions that are required to carry out the function and that are subsequently shown in separate diagrams.

Note that some diagrams have more than one entry point. In Diagram IA, for example, there are three entry points:

(1) at step 1 for an automatic Close,
(2) at step 5 after a user-issued CLOSE macro
(3) at step 7 after a user-issued TCLOSE macro

The notes provide details about the processing shown in the diagram. For example, note 5 tells what action the DOS/VS Close Monitor takes (in step 5) when it examines the DTF-type field in the list of ACBs and DTFs passed by the user. The notes also name the modules and routines that perform the functions represented.

The diagrams are numbered in a sequence that follows the pattern CCn, where the first character, in general, represents a part of VSAM such as catalog management, the second character represents a category within catalog management, and the number represents the first, second, third, etc., page of that particular diagram. Thus, DE1 would be the first page (1) of the Locate function (E) for catalog management (D). See the list of diagrams for details.

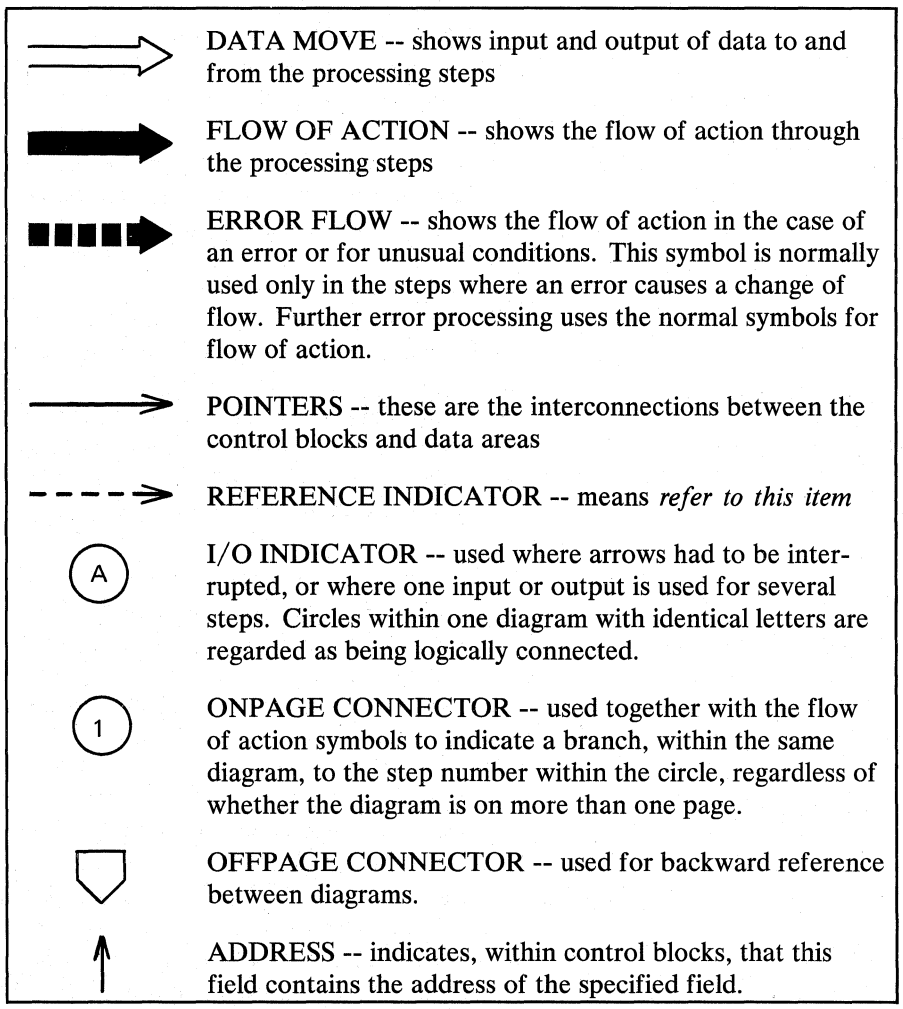


Figure 2.1 **Symbols used on method of operation diagrams**

Diagram AA1. Method of operation contents

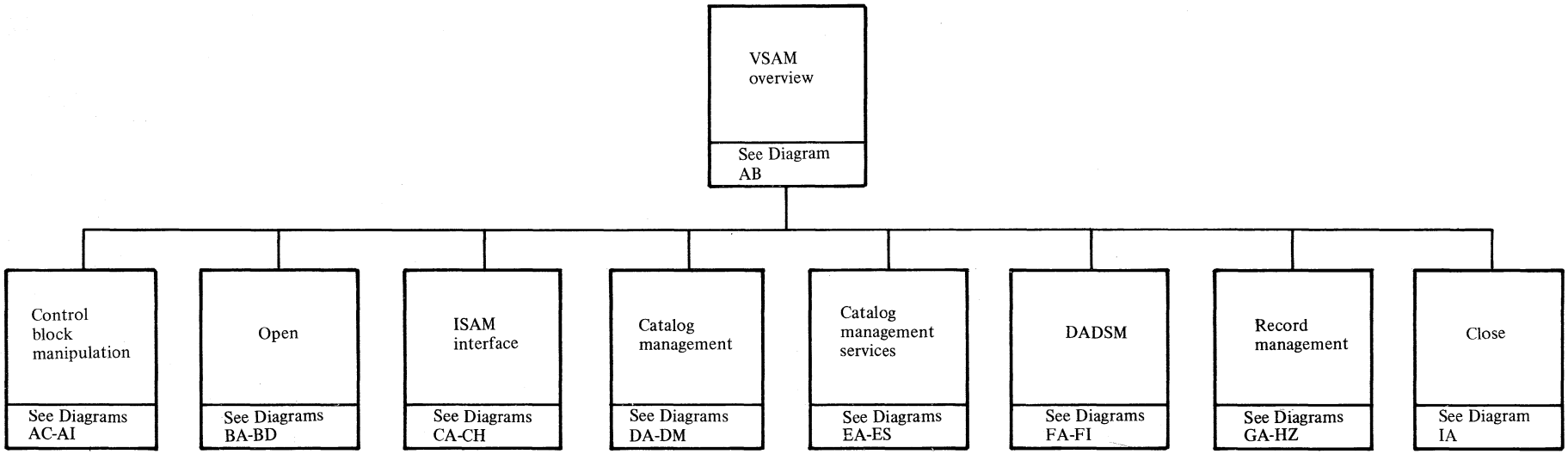


Diagram AB1. VSAM overview

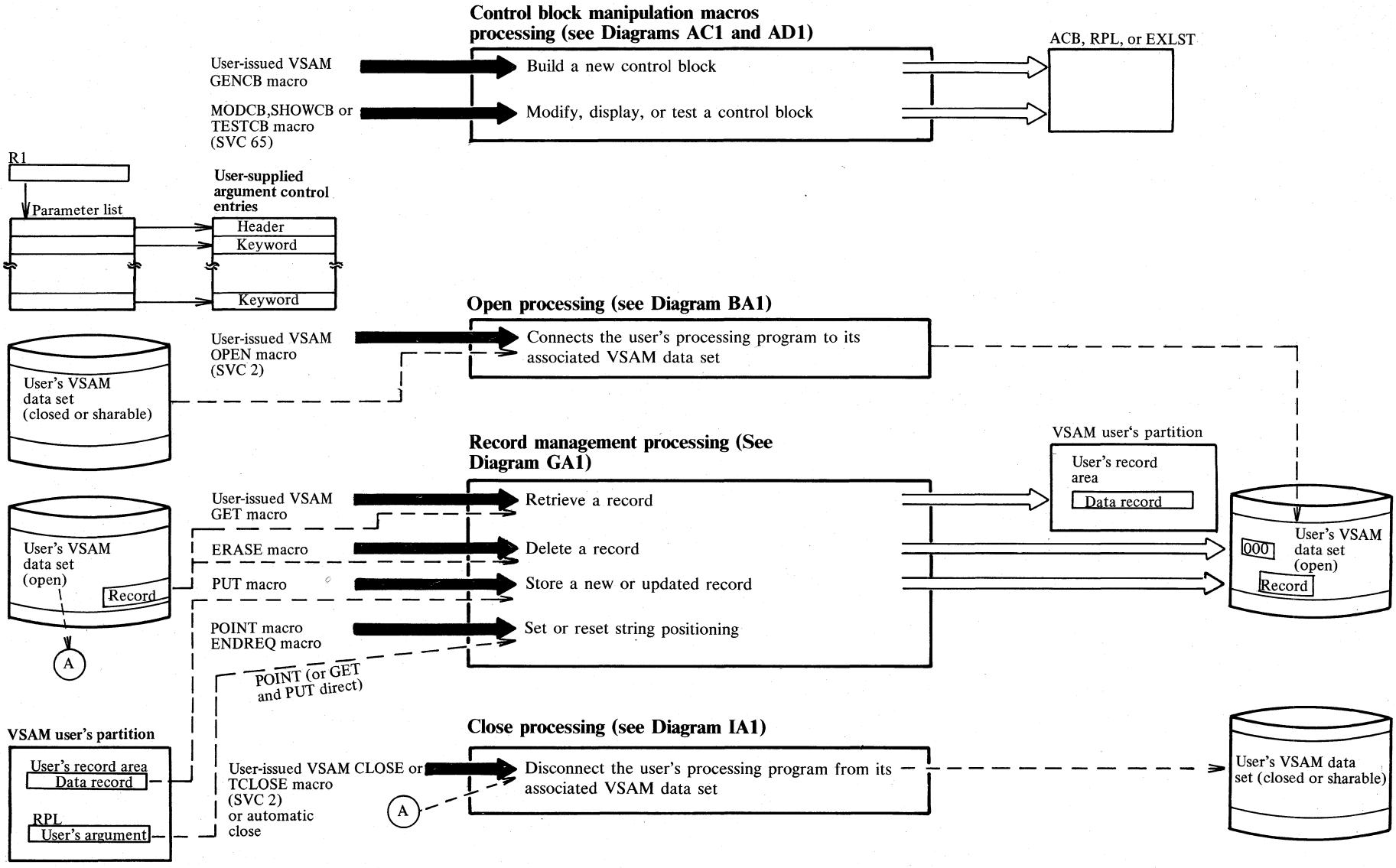


Diagram AC1. GENCB: Build a new control block

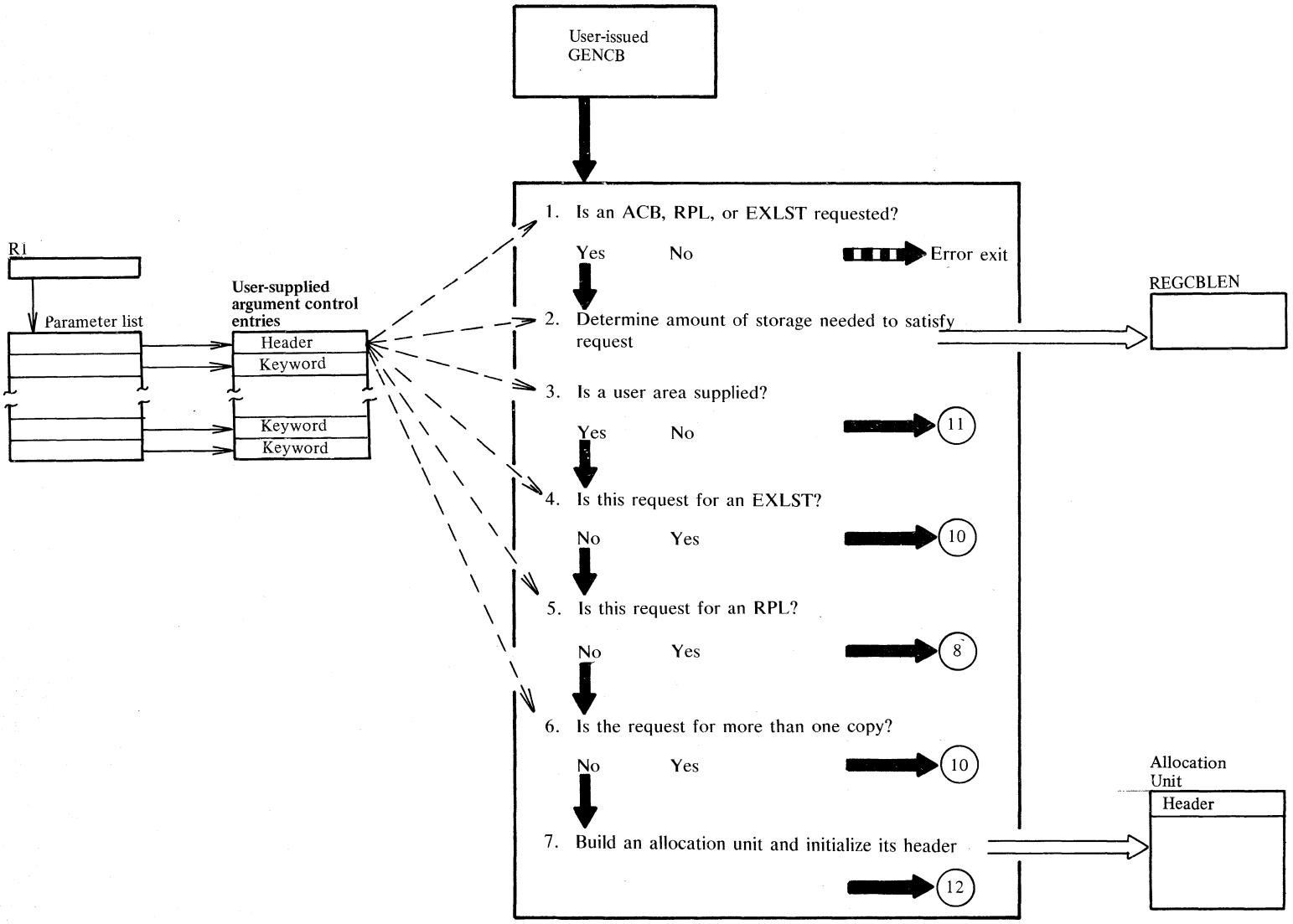


Diagram AC2. GENCB: Build a new control block

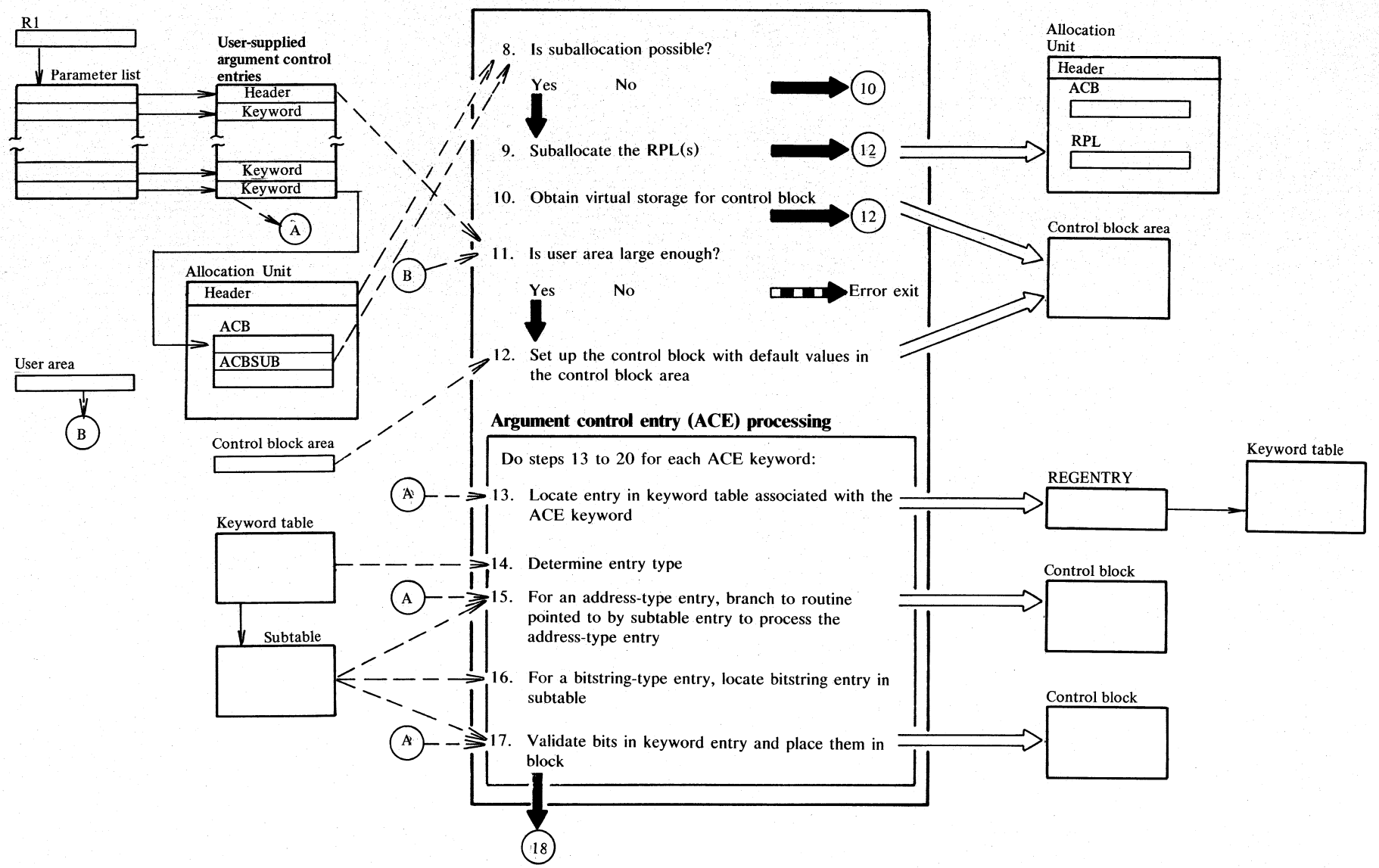
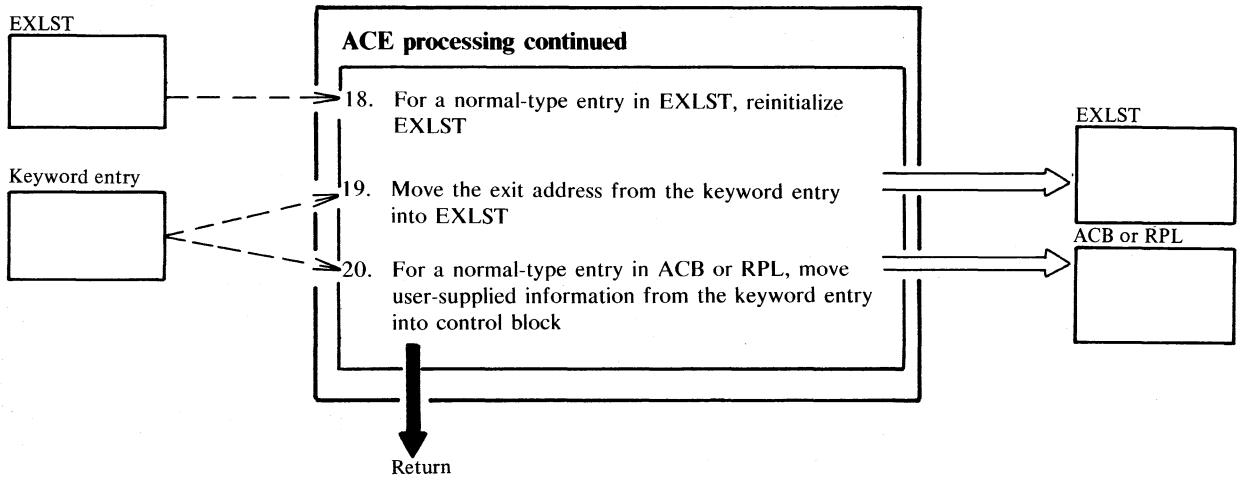


Diagram AC3. GENCB: Build a new control block



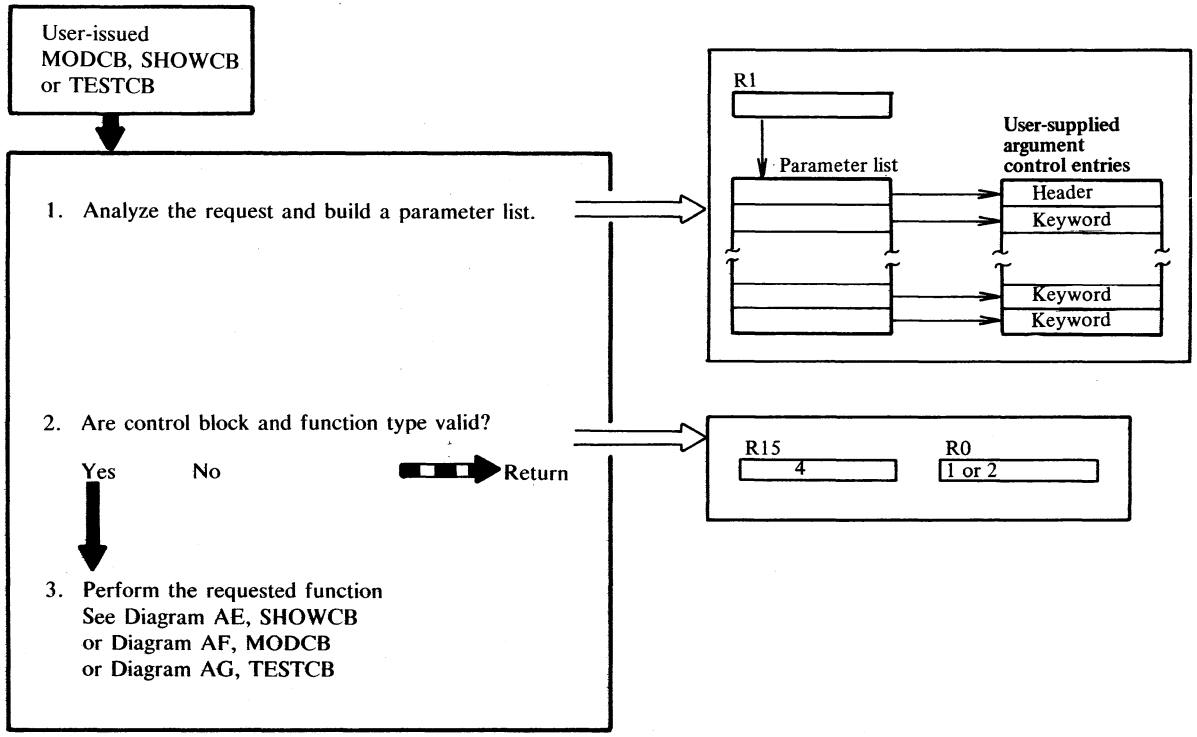
Module

IKQGEN

Notes for Diagram AC

	Description	Label
1	The GENCB macro is issued to create an ACB, RPL, or EXLST dynamically. If an ACB, RPL, or EXLST has not been requested, an error exit is taken.	IKQGEN00
2-3	The ACB and RPL are fixed-length control blocks but the EXLST length is variable. If a user area is supplied for an EXLST, this routine uses a minimum length of ten bytes to find out if the user area is large enough. If a user area is not supplied for an EXLST, this routine calculates the amount of space needed for the EXLST and any copies the user requested.	B1-B5
4-7	If only a single ACB is requested, and no user area is supplied, an allocation unit is built to contain an allocation unit header, an ACB, an AMBL, and as many RPLs and PLHs as indicated by the STRNO parameter.	BA1-BA5
8-9	If an RPL is requested for an ACB which is suballocated, and if there is sufficient space in the ACB's allocation unit, the RPL is suballocated.	C2-C2B
10	The routine issues a GETVIS macro to obtain the required virtual storage for any block for which a user area is not provided.	C2C-C4
12	The block is initialized to its default values. Information is subsequently added to the block as specified by the keyword entries.	C5-C8
13	For the last argument entry, the high-order bit in the parameter list pointer is 1.	D1
14	Three types of entries are identified in the keyword table: address, bitstring, and normal.	G1-G6
15	If the entry is an address type, the keyword table points to an offset in the subtable that contains a pointer to a routine that generates information to be placed in the appropriate control block field.	G5
16-17	If the entry is a bitstring type, the keyword table points to a series of bit entries in the subtable. The indicated bits are placed in the block.	M1-M4 N1-N4
18-19	The size of the EXLST is adjusted to the actual amount of space required to satisfy the user request. If the user-supplied area is not large enough, a return is made to the user. When the last keyword has been processed, any surplus space obtained by a GETVIS is reclaimed (surplus user-supplied space is not reclaimed). As each keyword is processed, the EXLST is checked to see if space is available at the proper offset for that keyword. If no slot is available, a GETVIS obtains additional space, provided the space was originally obtained by a GETVIS.	H1-H5 J1-J3
14	If the entry is a normal type, the value in the argument control entry is moved into the block.	K1-K3

Diagram AD1. MODCB, SHOWCB, TESTCB: Modify, display, or test a control block



Notes for Diagram AD

Description

The MODCB, SHOWCB, and TESTCB macros are issued to modify, display, and test, respectively, the ACB, RPL, or EXLST.

The macros are analyzed by the user macro IKQCB2. If this user macro actually builds the parameter list, it can ensure that the parameters are valid. In this case, control is then passed to the module IKQTMSF (F = fast = without diagnostics).

If the user provides a ready-built parameter list, the macro IKQCB2 cannot check the validity of the parameters and passes control to the module IKQTMSD (D = diagnostic).

Each module contains routines for processing all possible combinations of request and control block (such as SHOWCB for ACB, TESTCB for an RPL, etc.).

Diagram AE1. SHOWCB: Display a control block

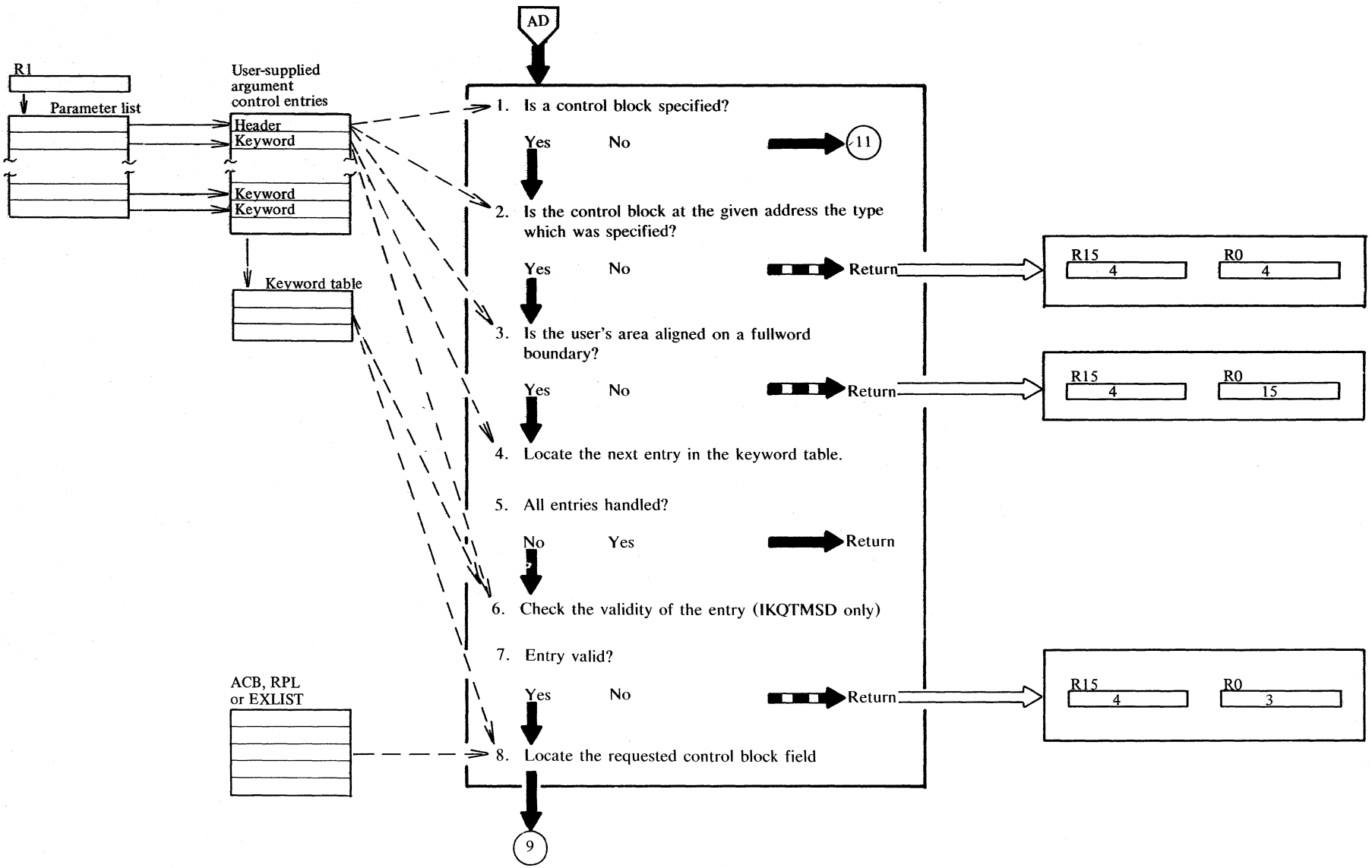
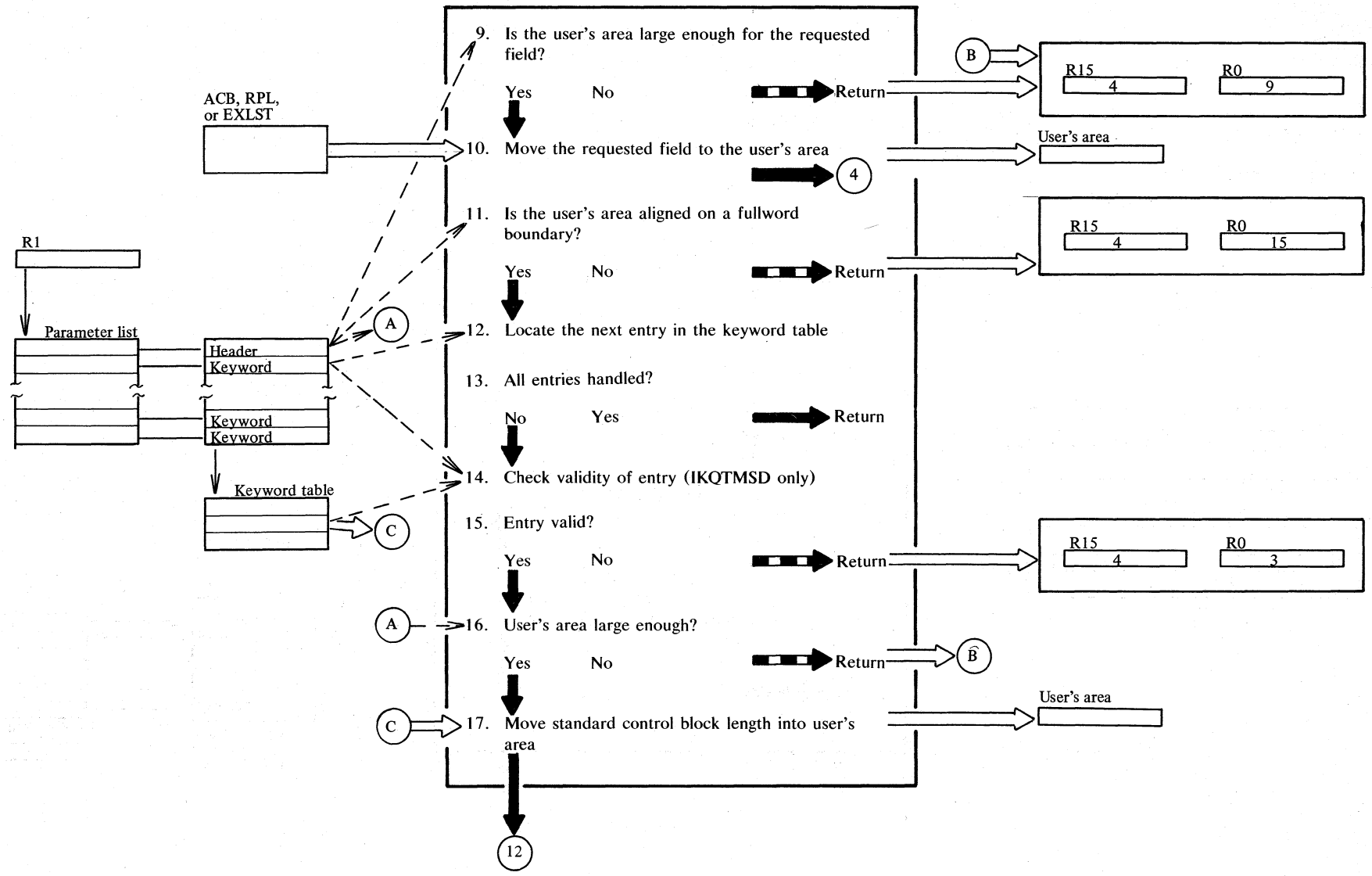


Diagram AE2. SHOWCB: Display a control block



Notes for Diagram AE

The processing shown here can take place in module IKQTMSF or IKQTMSD (see notes for Diagram AD).

The subroutines, which have identical names in both modules, are:

Block	Label
ACB	SHOWACB
RPL	SHOWRPL
EXLST	SHOWEXL
None	SHOWNOB

Diagram AF1. MODCB: Modify a control block

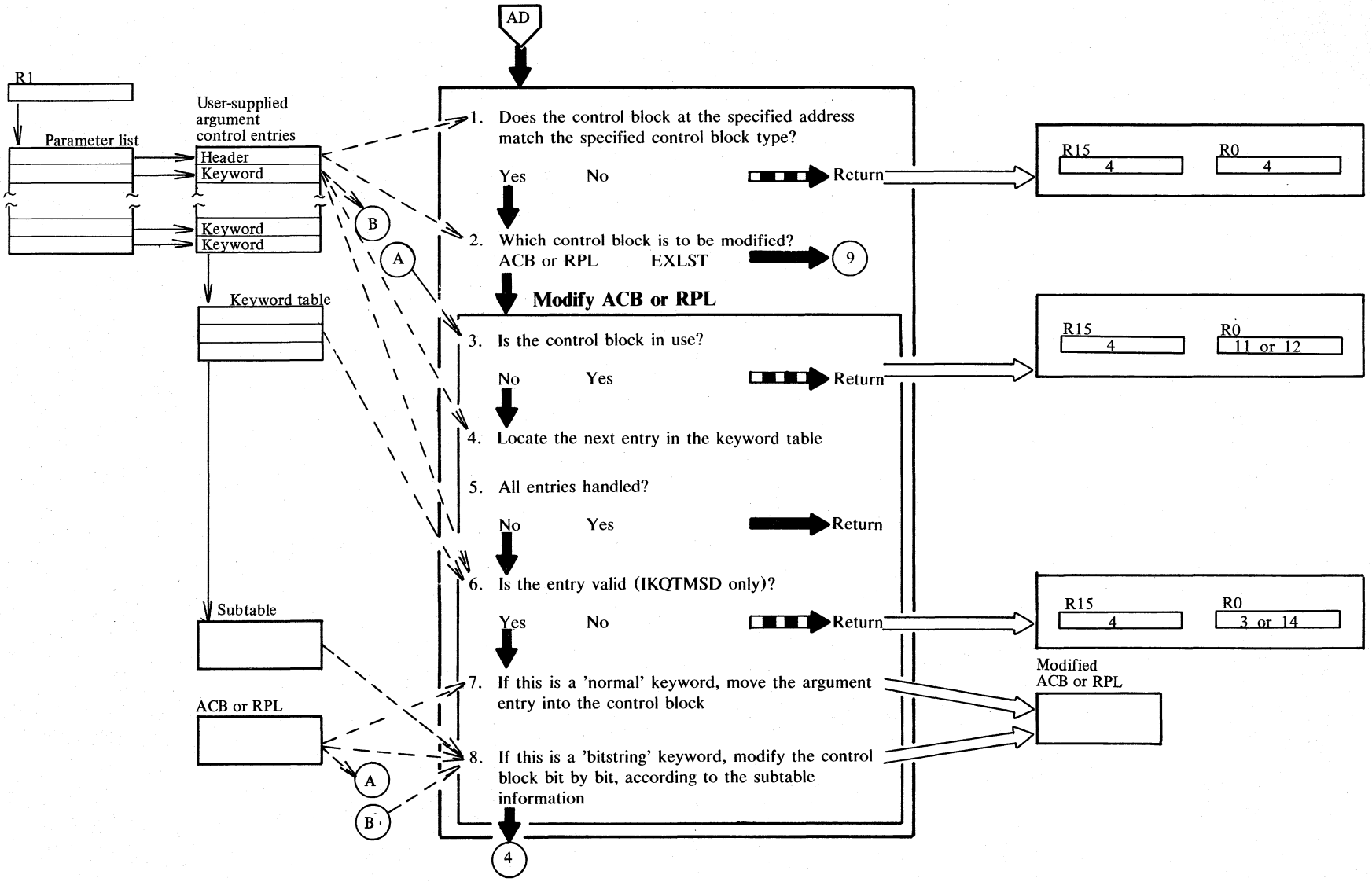
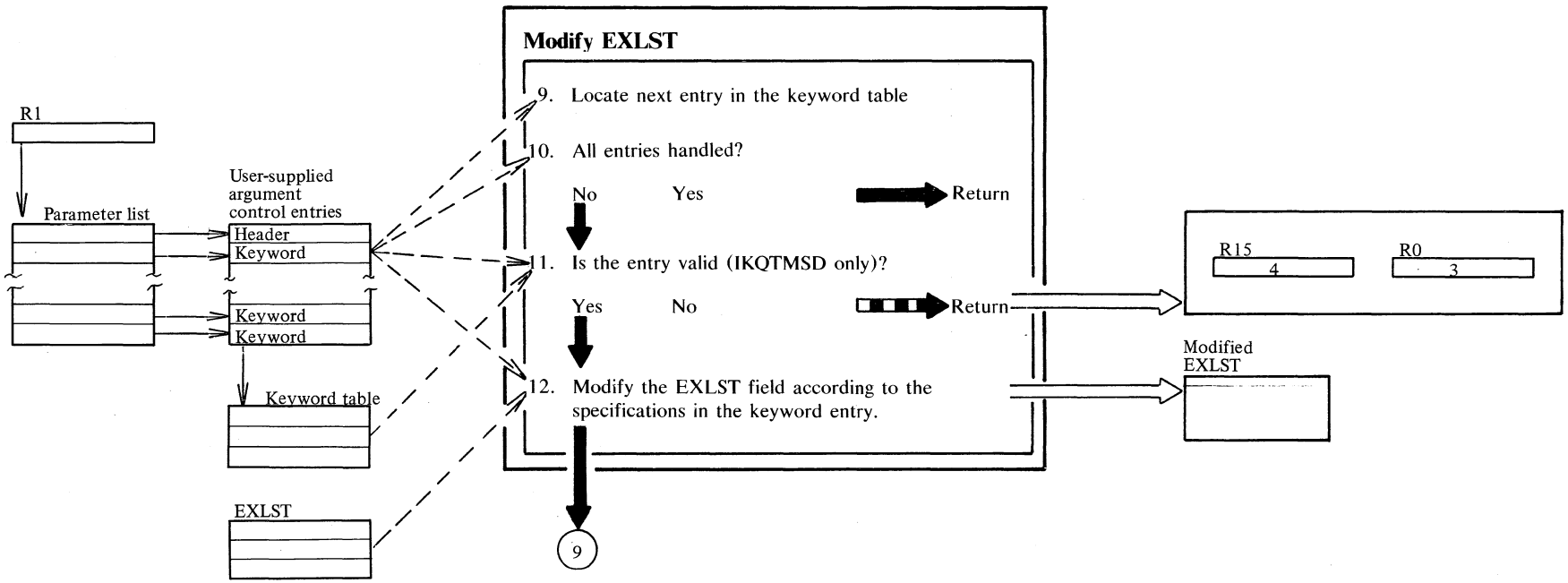


Diagram AF2. MODCB: Modify a control block



Notes for Diagram AF

The processing shown here can take place in module IKQTMSF or IKQTMSD (see notes for Diagram AD).

The subroutines used for MODCB, which have identical names in both modules are:

Block	Label
ACB	MODACB
RPL	MODRPL
EXLST	MODEXL

Diagram AG1. TESTCB: Test a control block

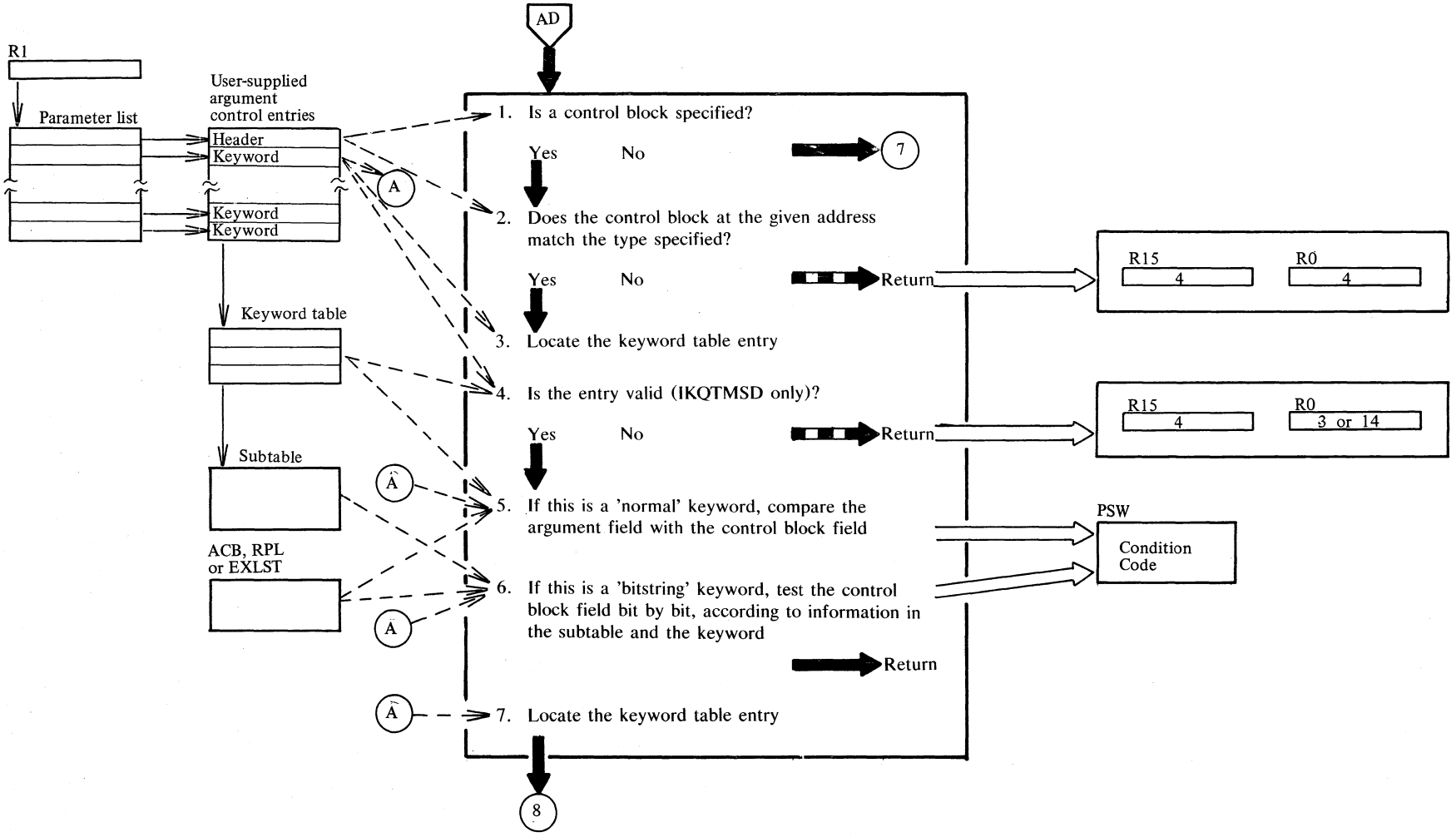
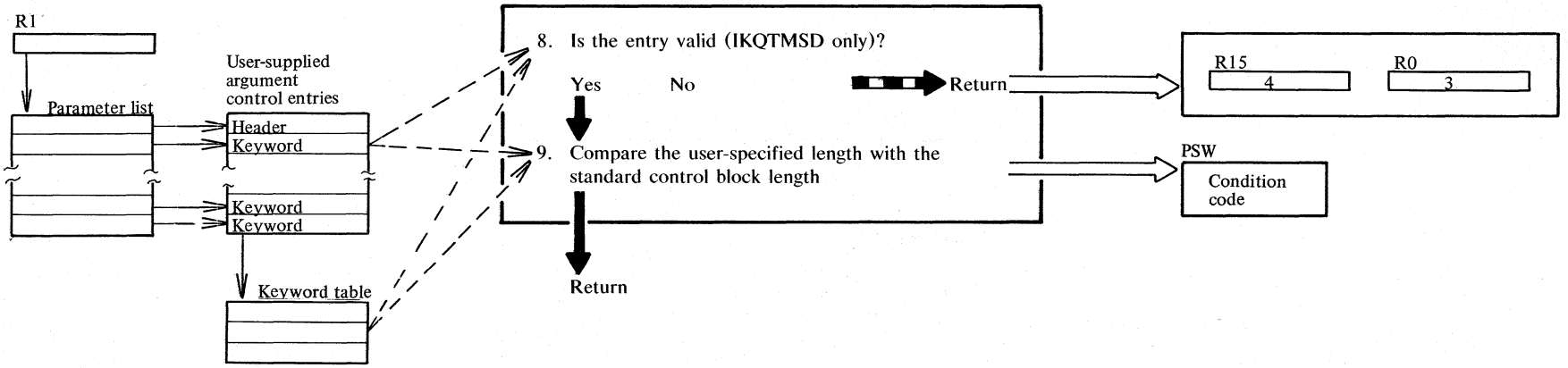


Diagram AG2. TESTCB: Test a control block



Notes for Diagram AG

The processing shown here can take place in module IKQTMSF or IKQTMSD (see notes for Diagram AD).

The subroutines used for TESTCB, which have identical names in both modules, are:

Block	Label
ACB	TESTACB
RPL	TESTRPL
EXLST	TESTEXL
None	TESTNOB

Diagram AH1. BLDVRP: Build VSAM Shared Resource Pool

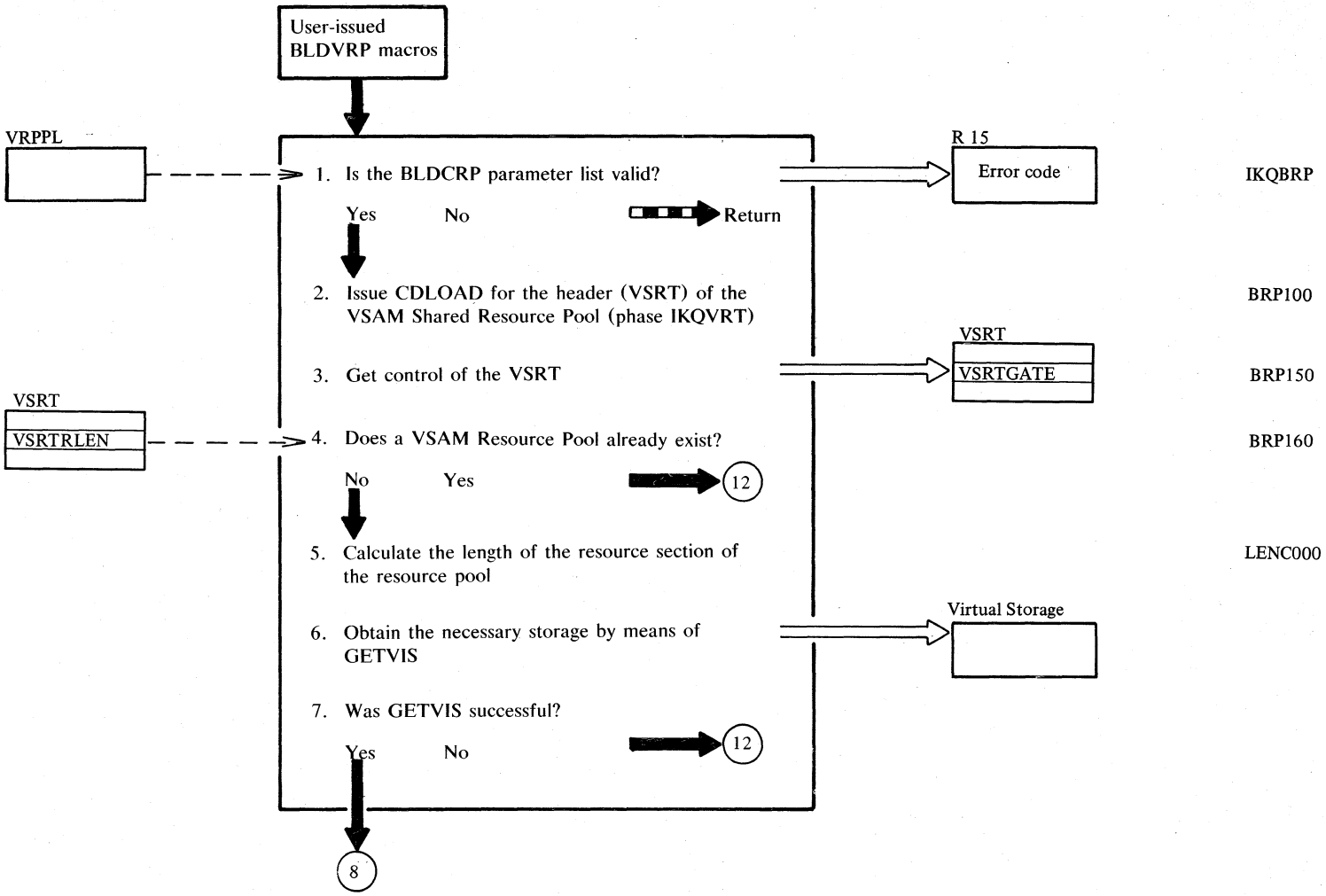
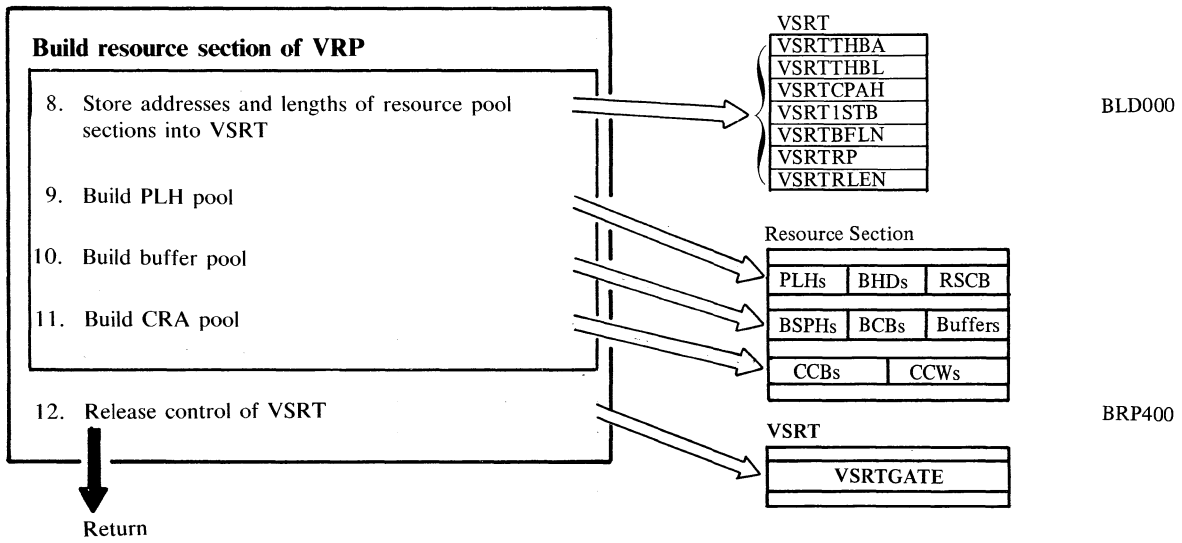


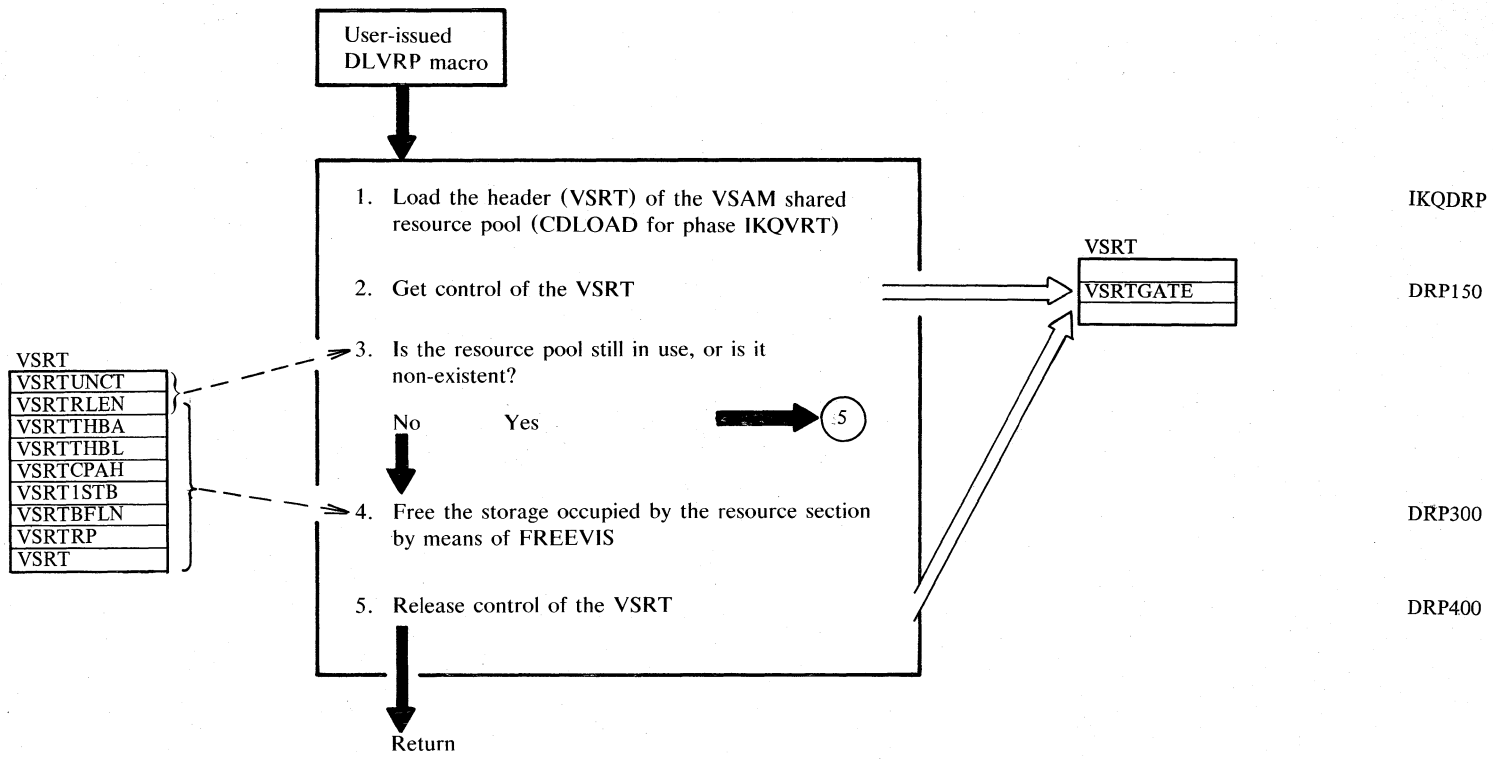
Diagram AH2. BLDVRP: Build VSAM Shared Resource Pool



Note:

2. The phase IKQVRT is an assembled CSECT that contains only the VSRT.

Diagram AI1. DLVRP: Delete VSAM Shared Resource Pool



Note:

1. For an existing VSRT, CDLOAD simply provides addressability to the VSRT.

Diagram BA1. OPEN: Connect a user's program to a VSAM data set

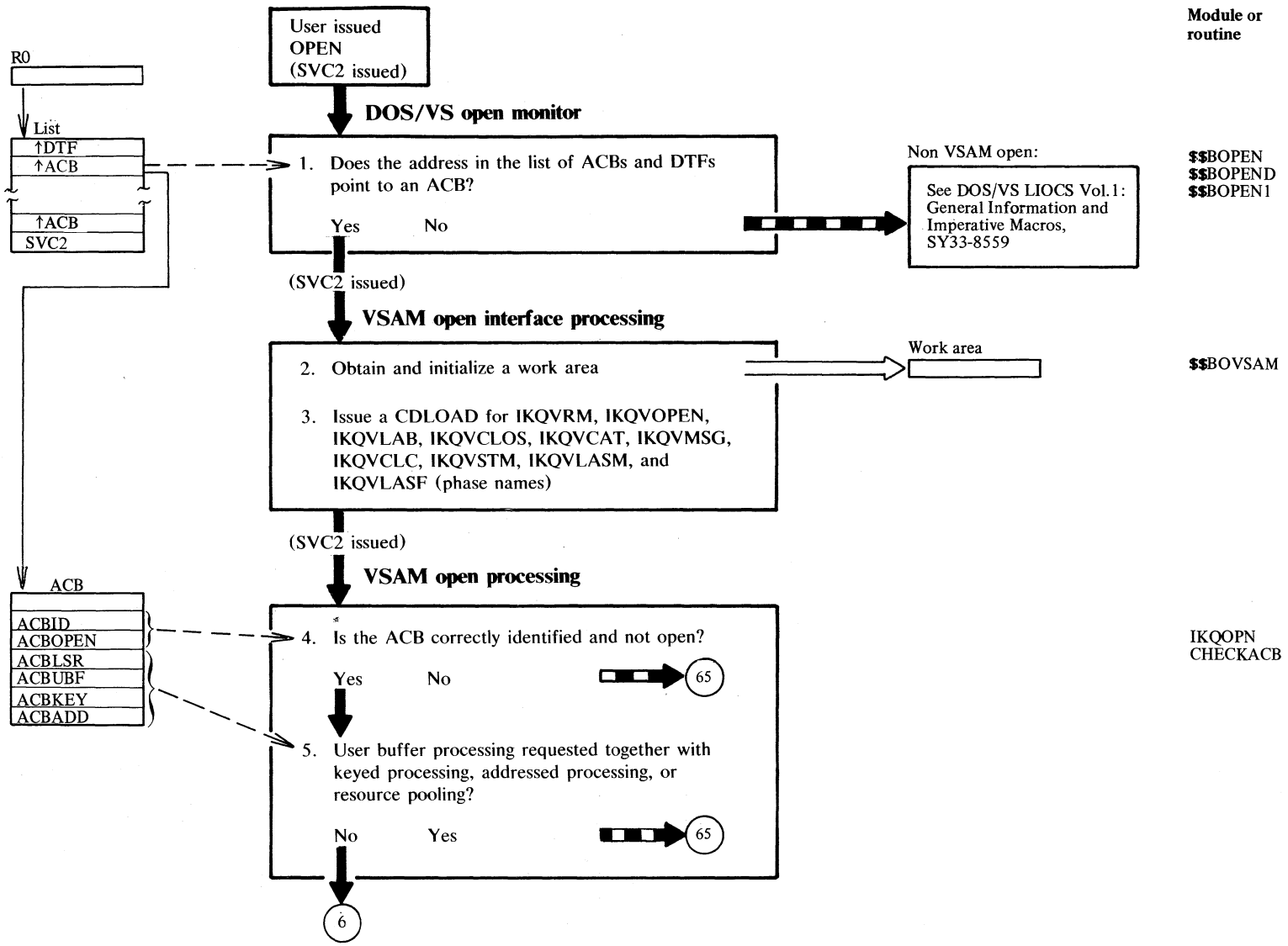


Diagram BA2. OPEN: Connect a user's program to a VSAM data set

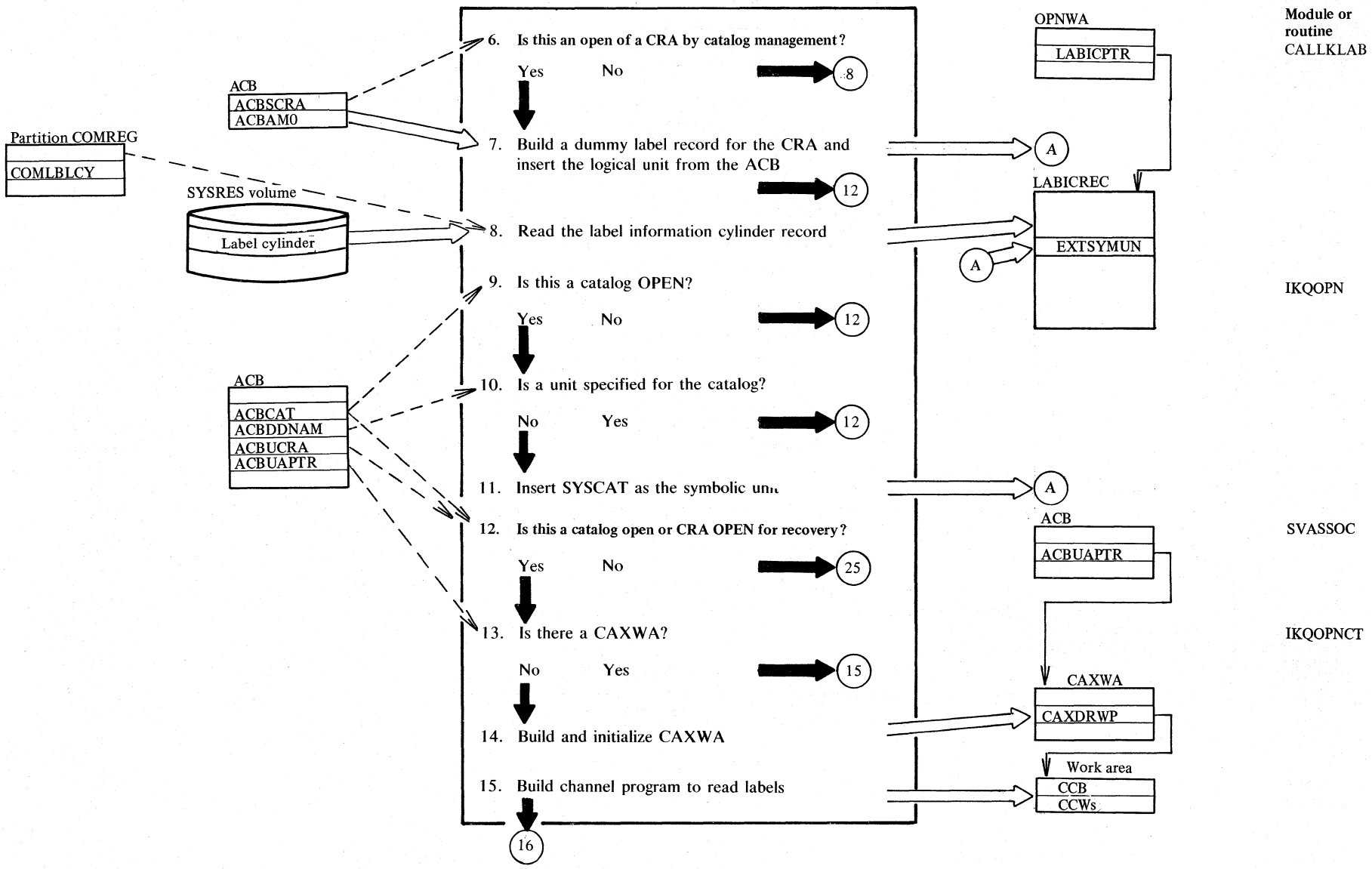


Diagram BA3. OPEN: Connect a user's program to a VSAM data set

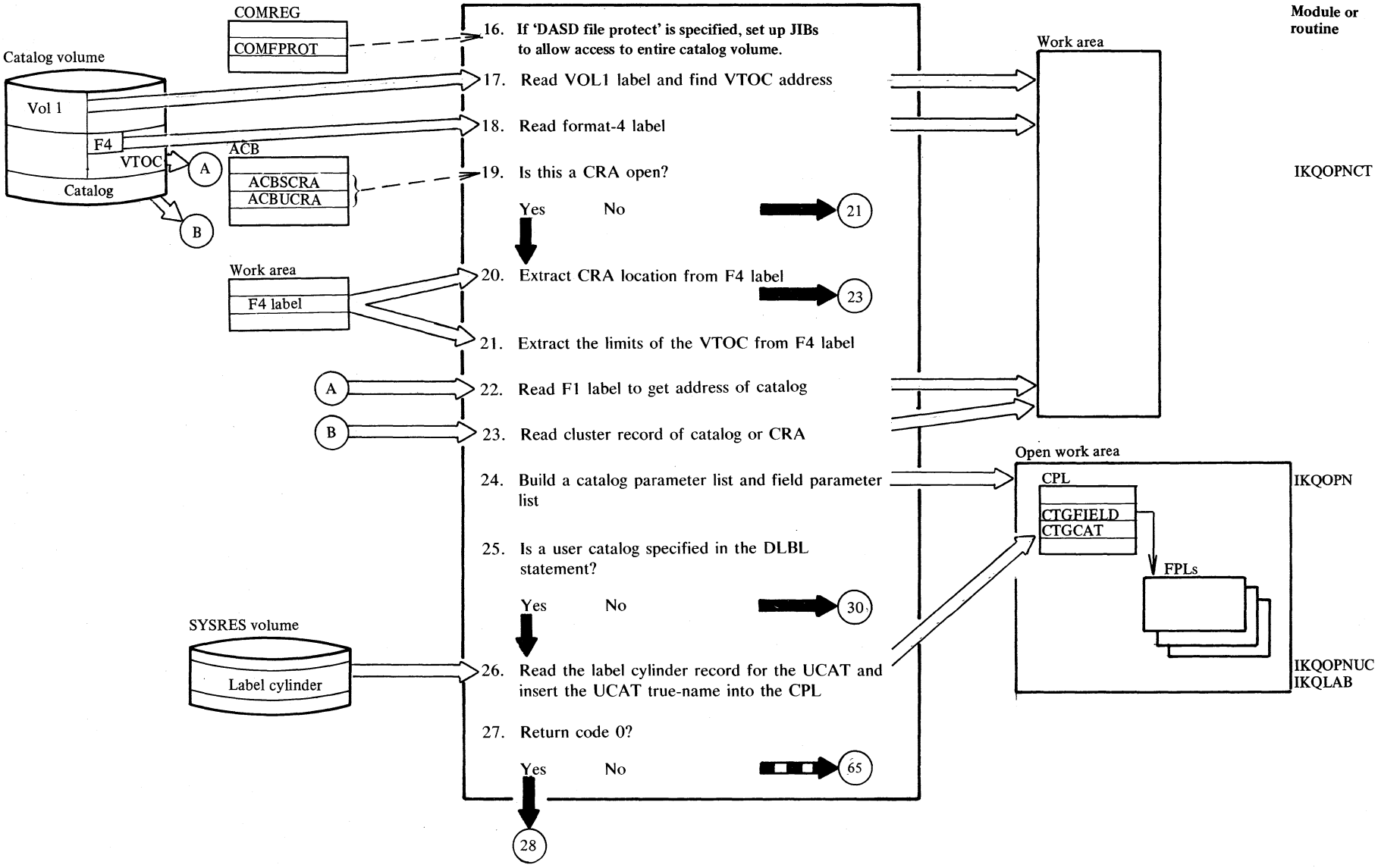


Diagram BA4. OPEN: Connect a user's program to a VSAM data set

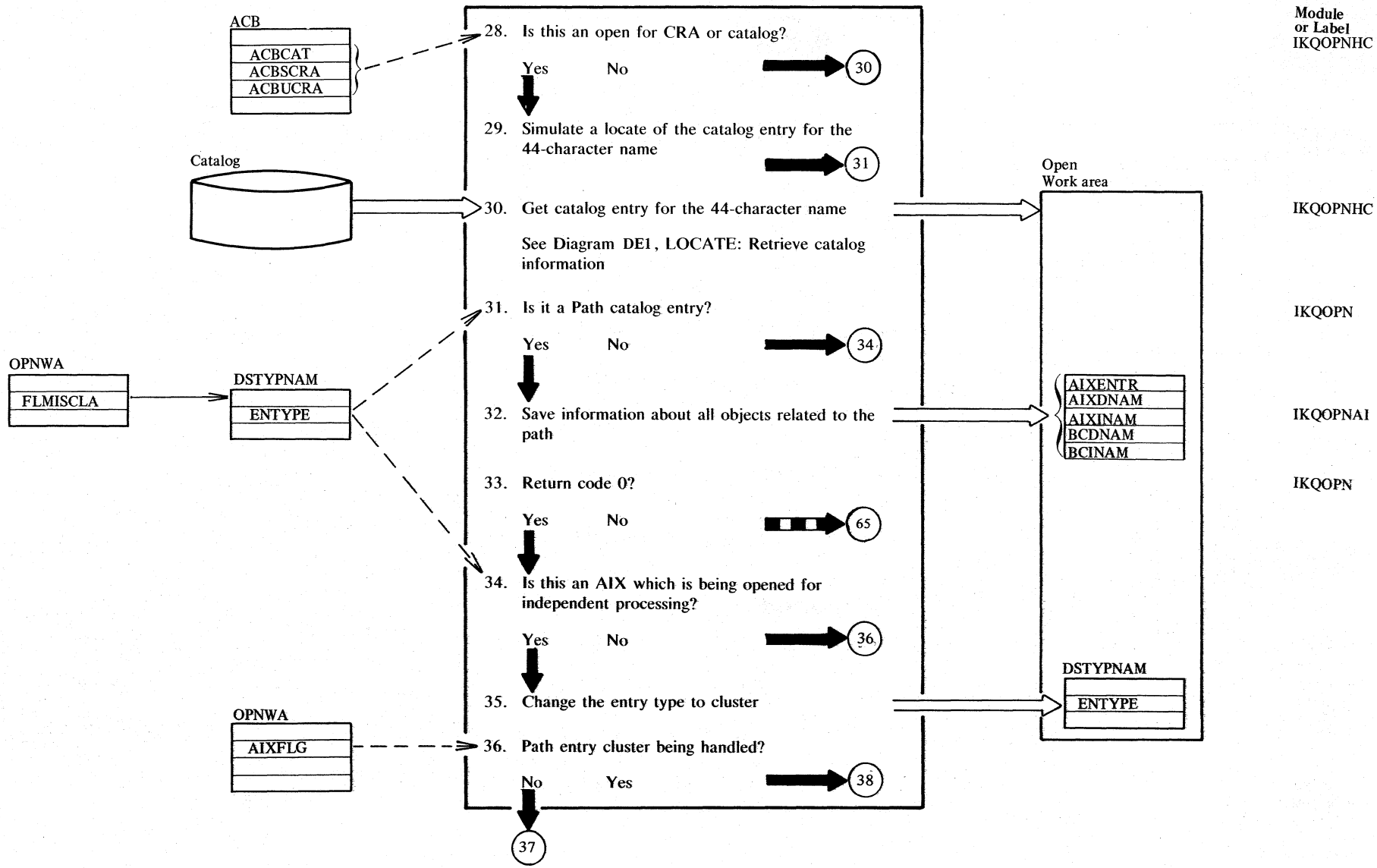


Diagram BA5. OPEN: Connect a user's program to a VSAM data set

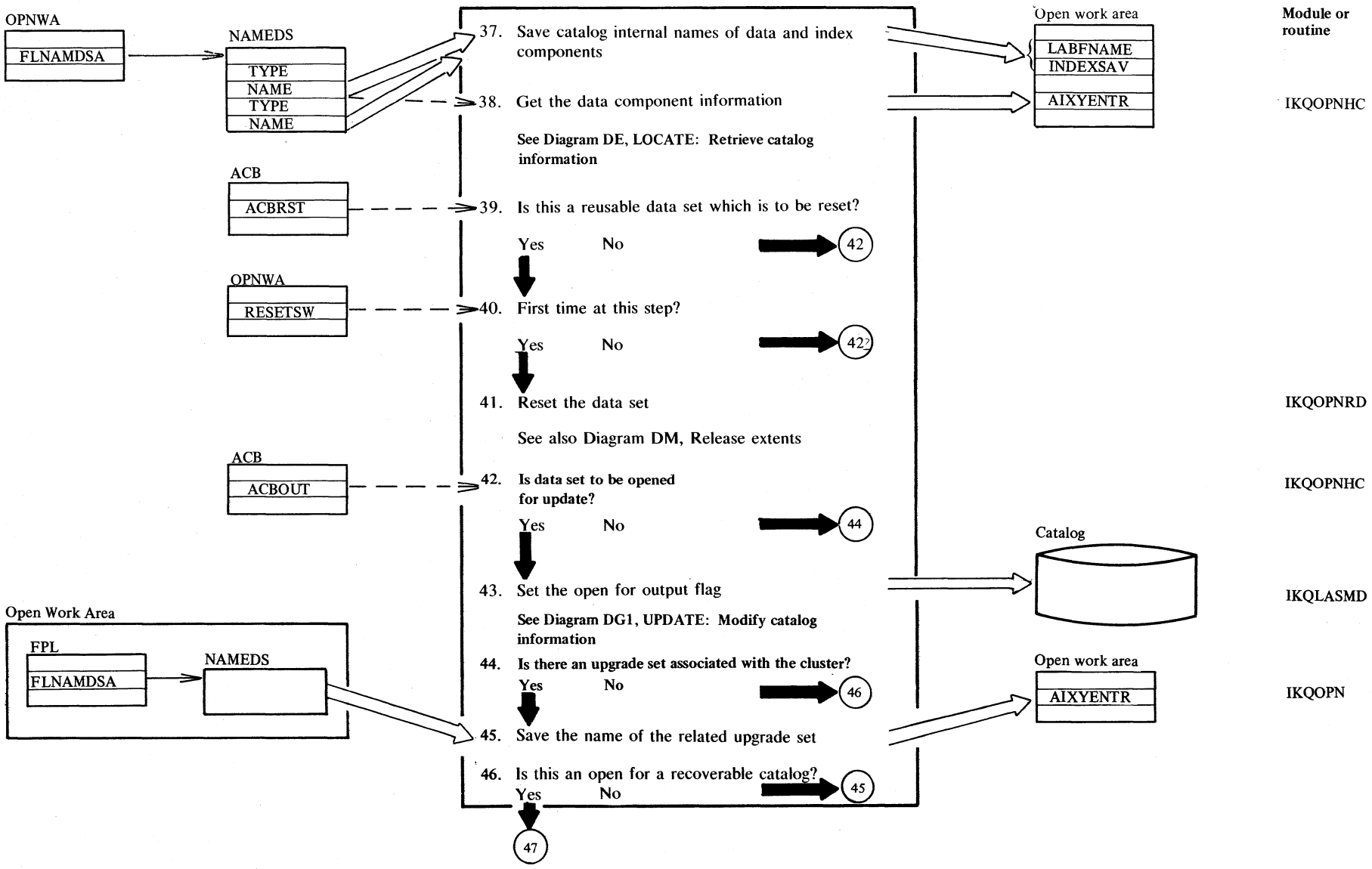


Diagram BA6. OPEN: Connect a user's program to a VSAM data set

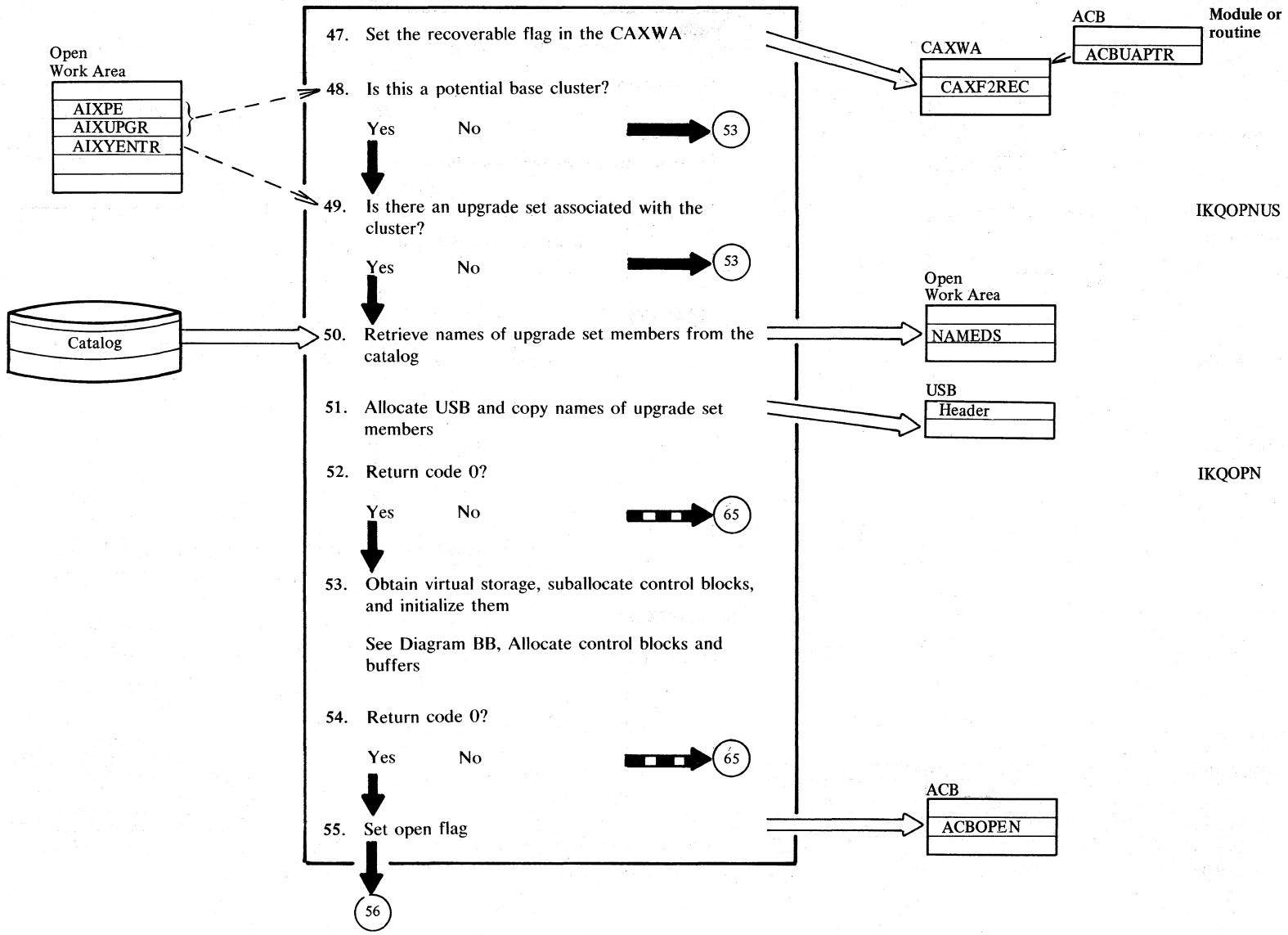


Diagram BA7. OPEN: Connect a user's program to a VSAM data set

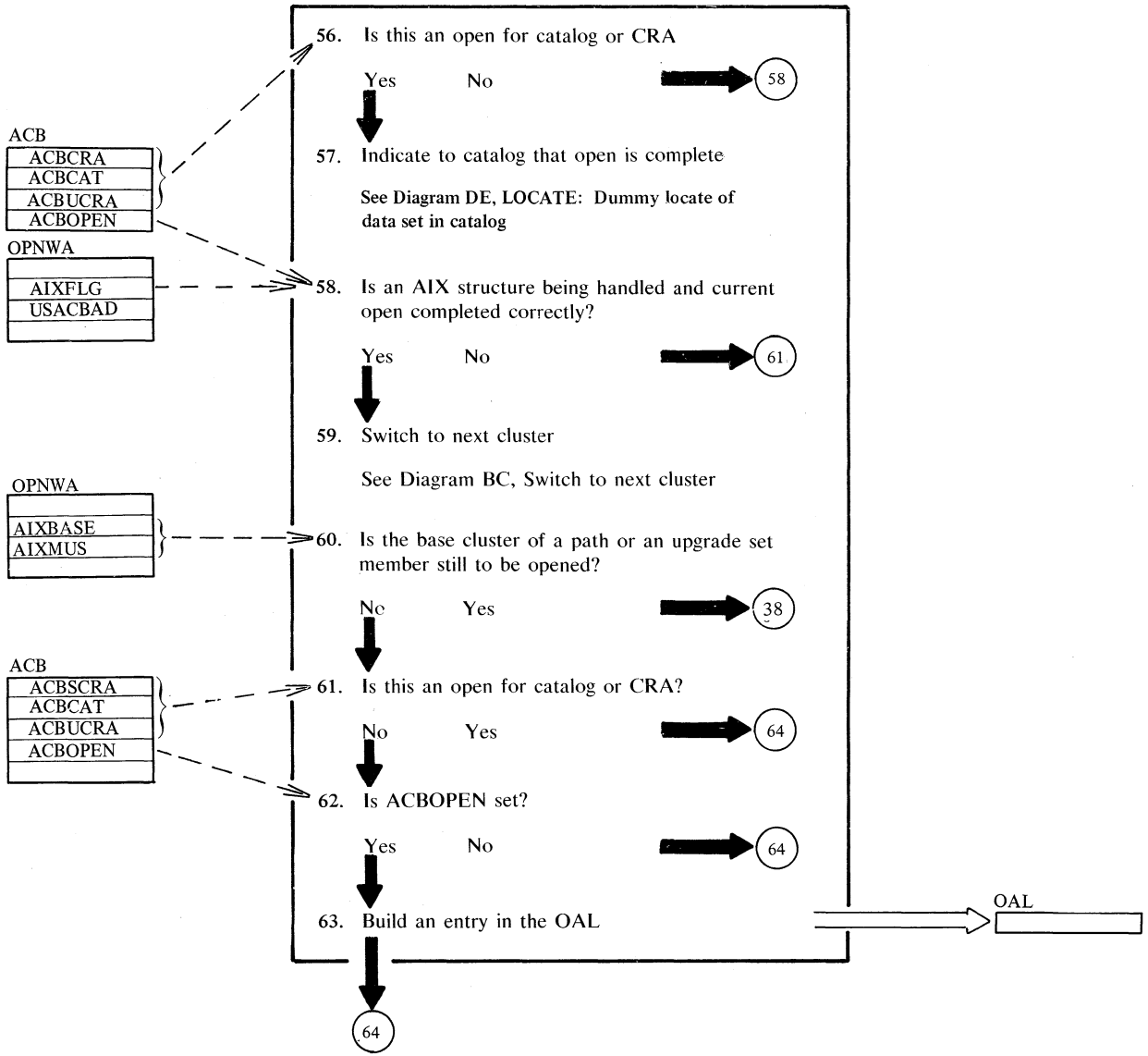
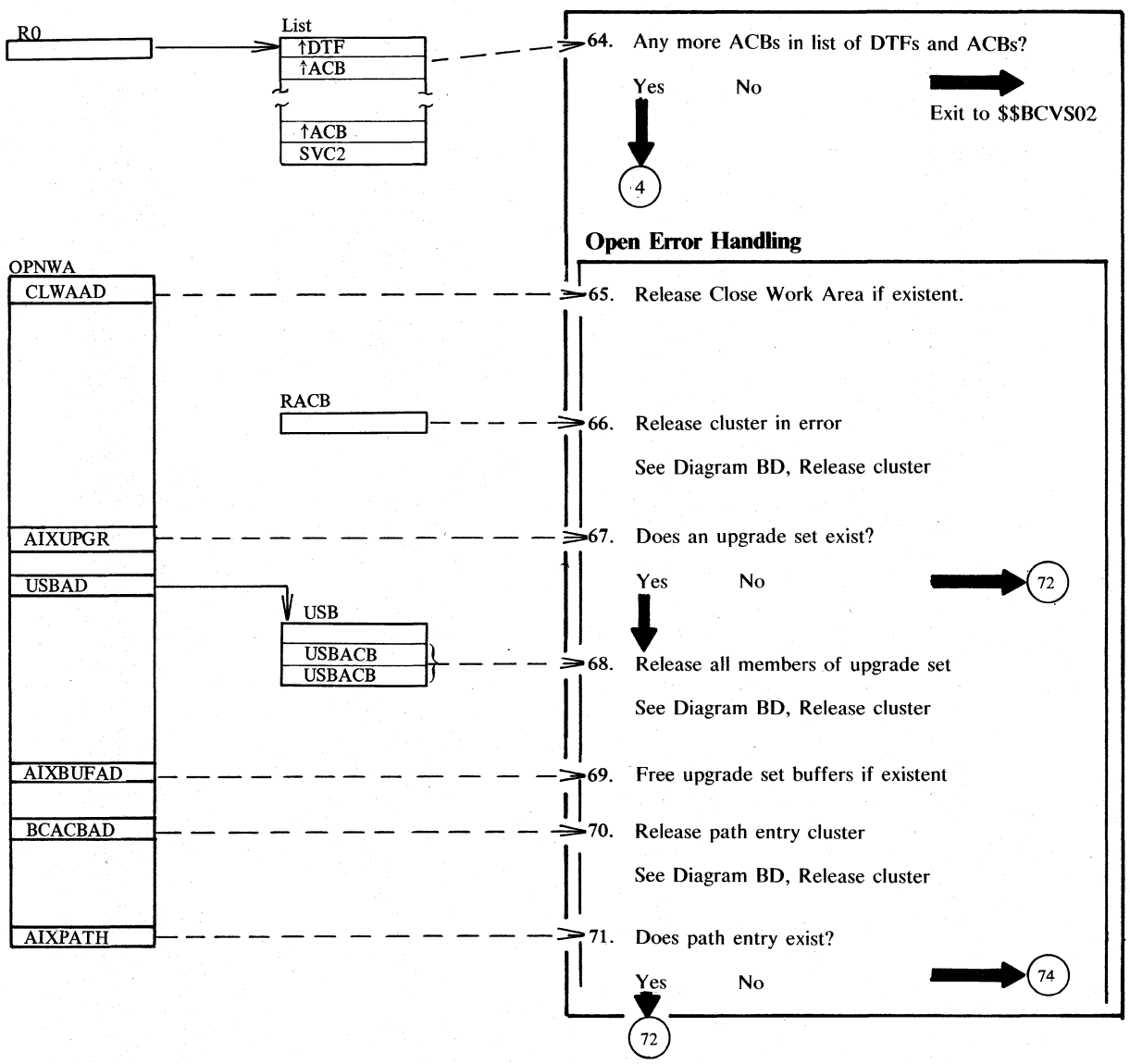


Diagram BA8. OPEN: Connect a user's program to a VSAM data set

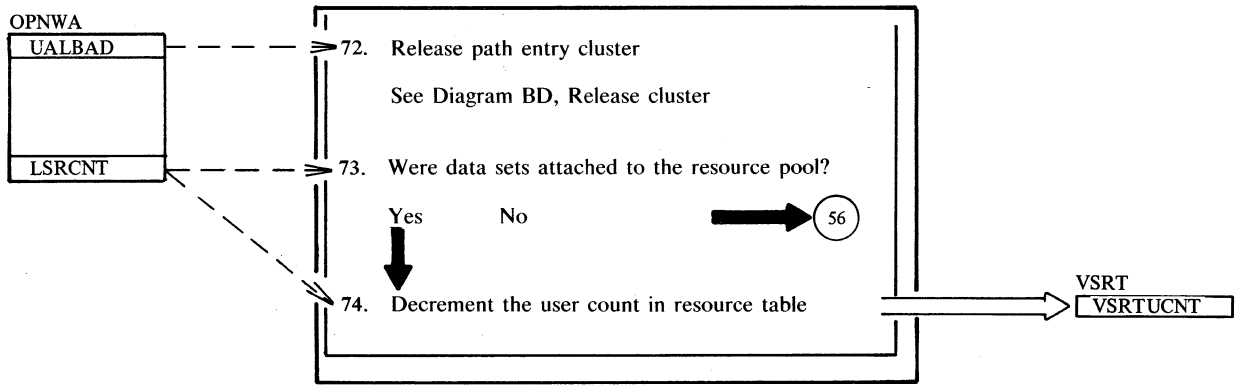


Module or routine

OPWAPUP

IKQOPNDO

Diagram BA9. OPEN: Connect a user's program to a VSAM data set



Module or routine

Notes for Diagram BA

17. **If the user catalog is not mounted before the first job is run that opens that catalog, the job is terminated.**

24. The contents of the following combination names are fetched from the catalog:
NAMEDS
DSTYPNAM

30. The contents of the following fields and combination names are fetched from the catalog:
NAMEDS
DSTYPNAM
RGATTR
DSCATACB

36. Path entry cluster is an AIX which is part of a VSAM path

38. The contents of the following fields and combination names are fetched from the catalog:

ENTVOL
DSATTR
OPENIND
BUFSIZE
HURBADS
NAMEDS
EXCEPEXIT
AMDSB
CATFILT
DSCATACB

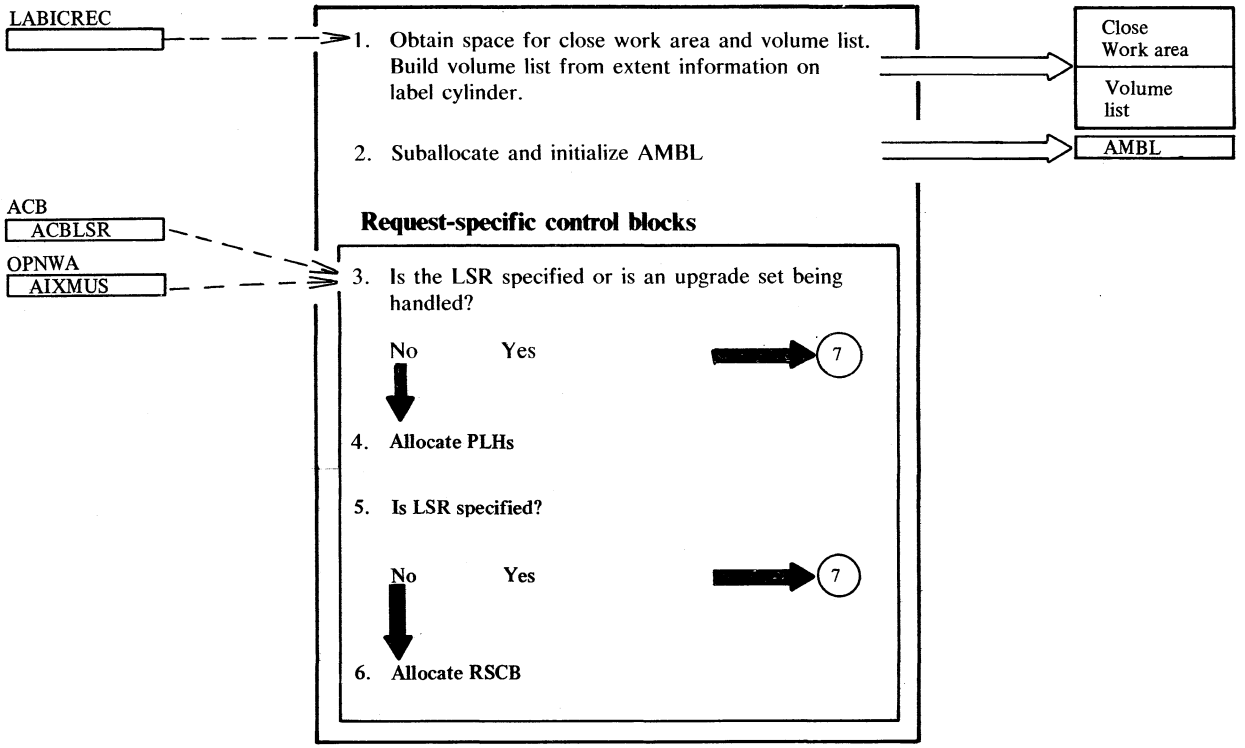
48. Base cluster is a cluster which is not PE or UPGR

50. The contents of combination name NAMEDS are fetched from the catalog.

57. This releases resources obtained by the catalog.

59. This can be a base cluster of a path or upgrade set member(s) for a base cluster

Diagram BB1. Allocate control blocks and buffers



IKQOPN

Diagram BB2. Allocate control blocks and buffers

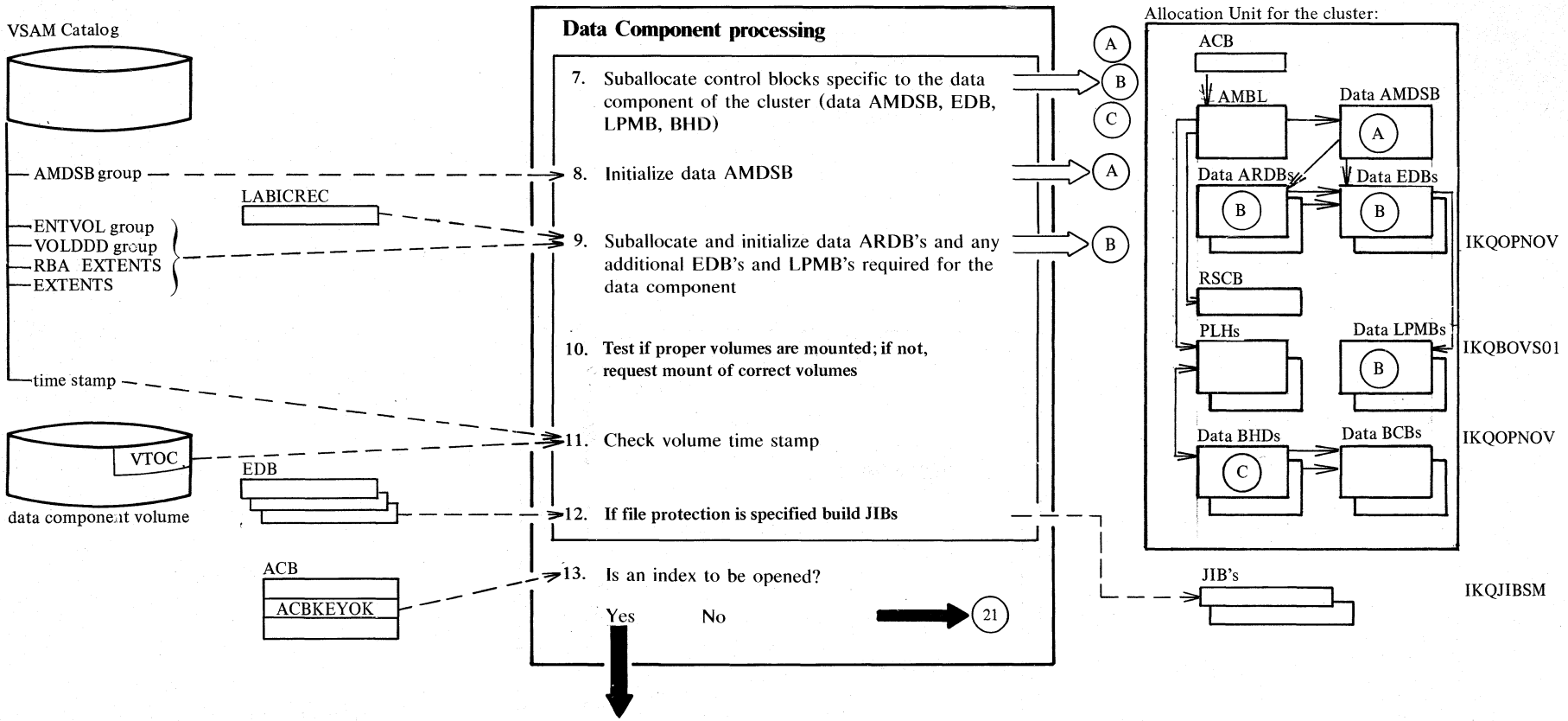


Diagram BB3. Allocate control blocks and buffers

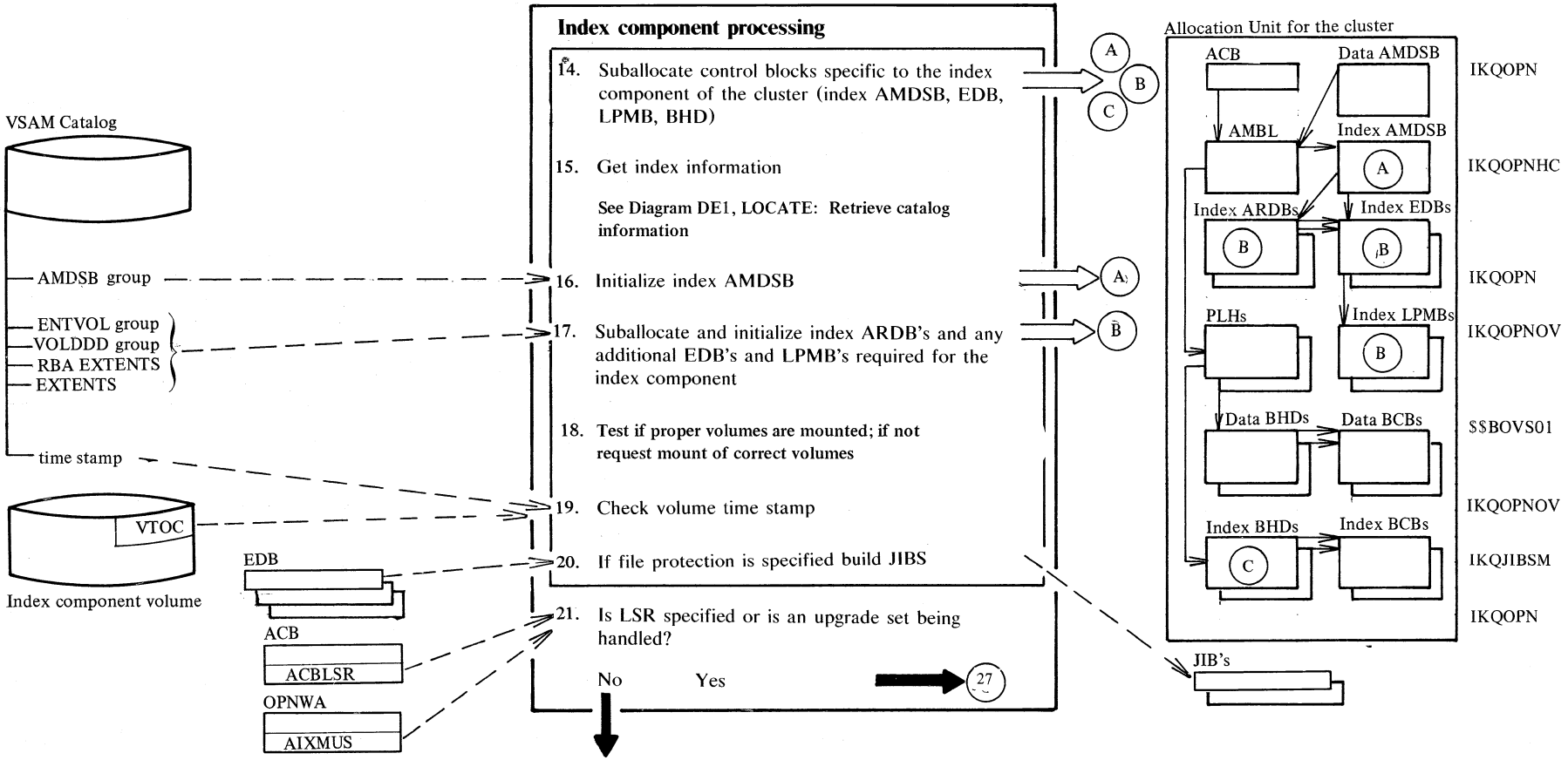


Diagram BB4. Allocate control blocks and buffers

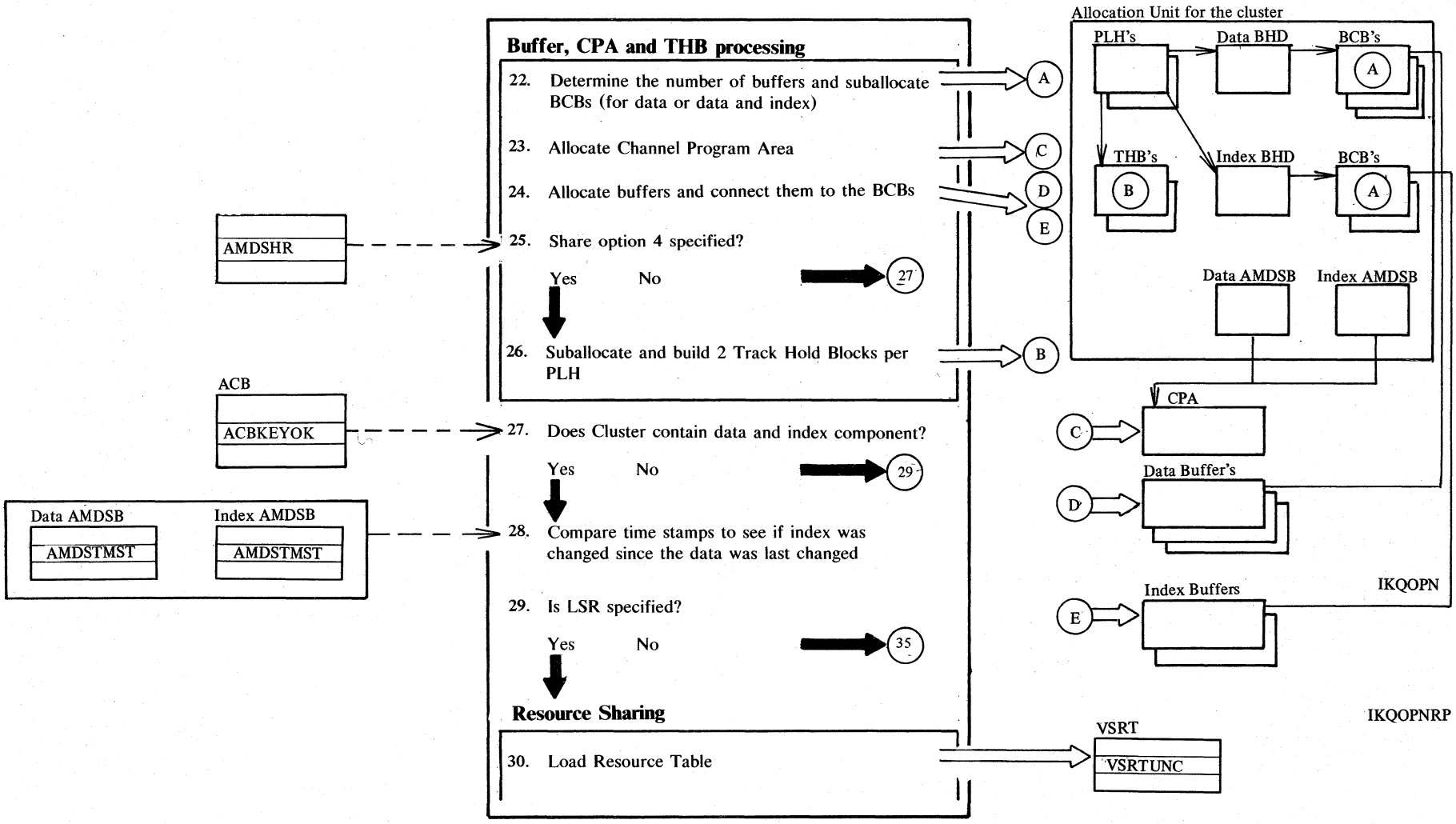
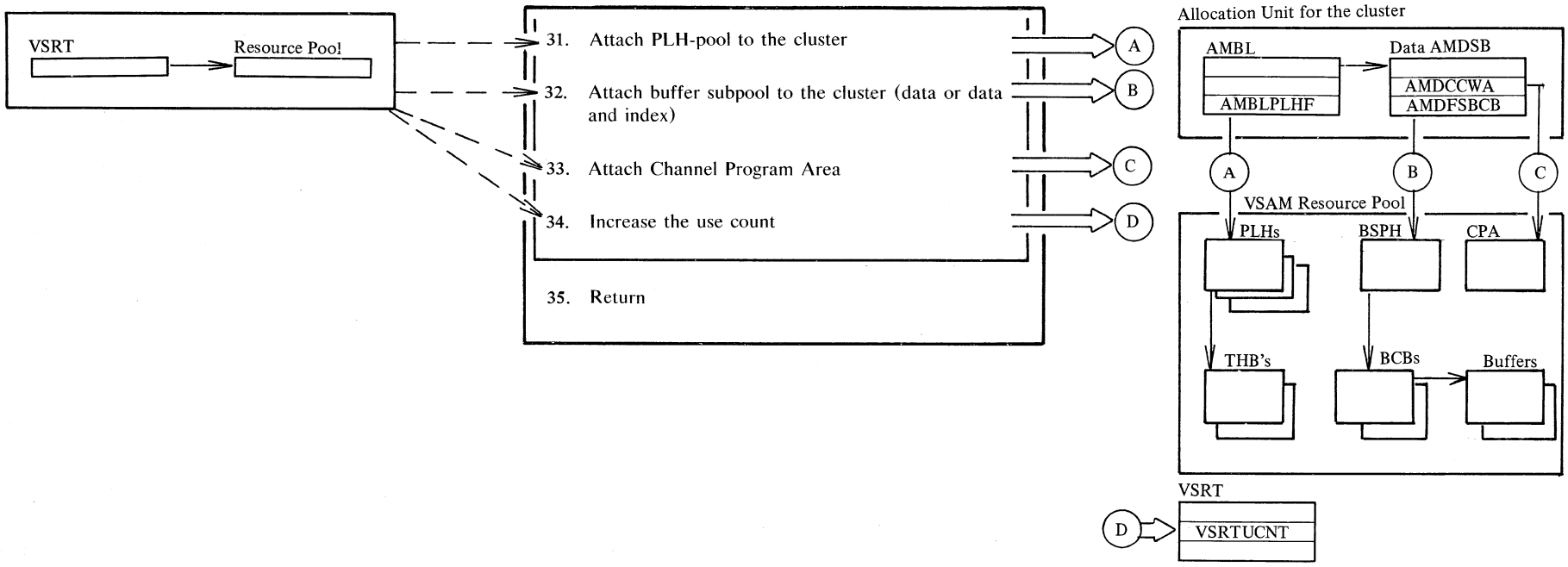


Diagram BB5. Allocate control blocks and buffers



Notes for Diagram BB

4. PLHs for data sets with LSR already exist in the resource pool. Members of an upgrade set share one PLH which was allocated by IKQOPNNC. The RSCB is not allocated for LSR because it is a member of the resource pool.

25. If the data set was defined SHARE (4) but the DOS/VS supervisor was not generated with track hold support, then the AMDSB is altered in storage to reflect SHARE (2). The AMDSB group occurence in the catalog is not modified.

Diagram BC1. Switch to next cluster

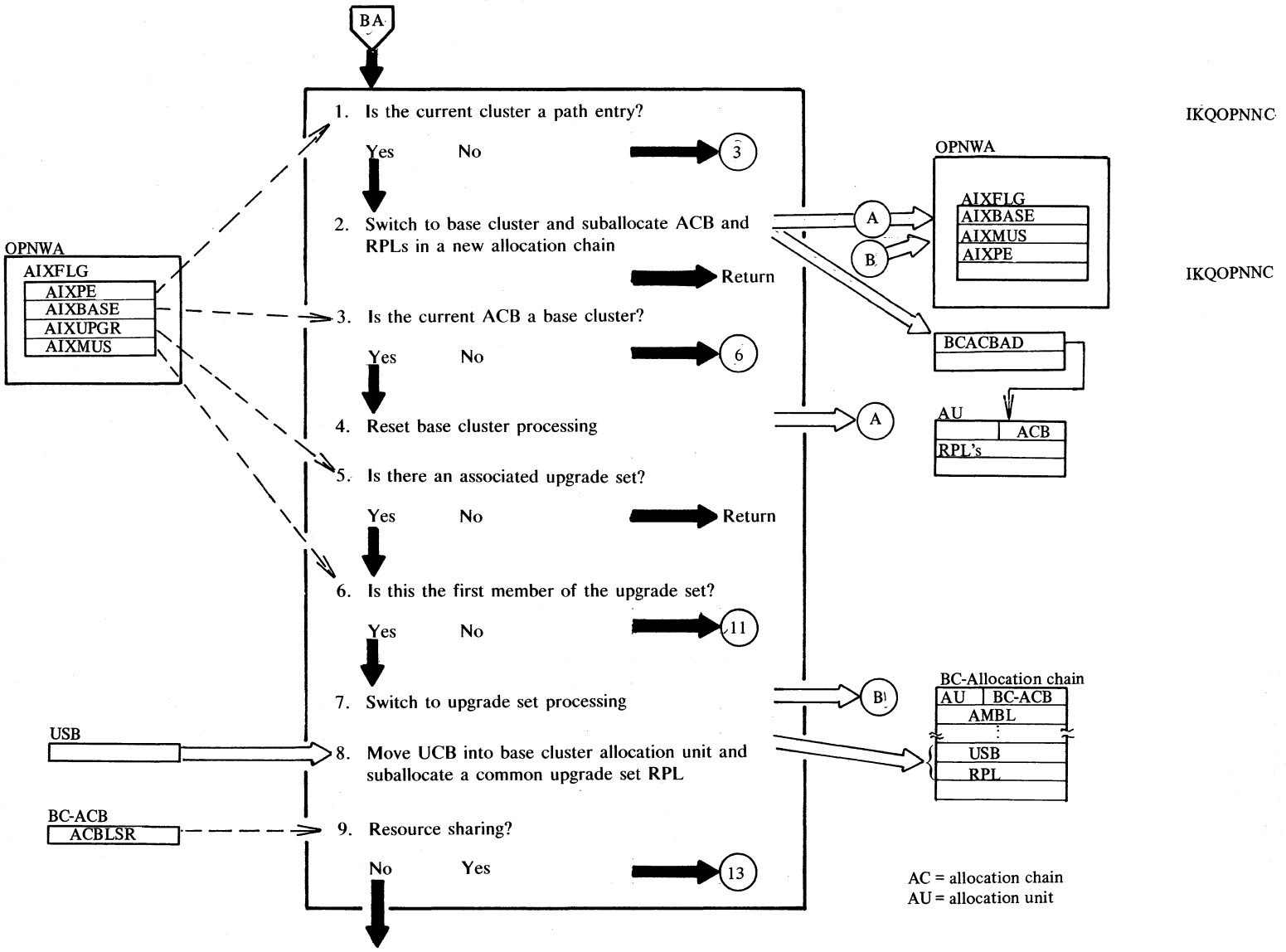


Diagram BC2. Switch to next cluster

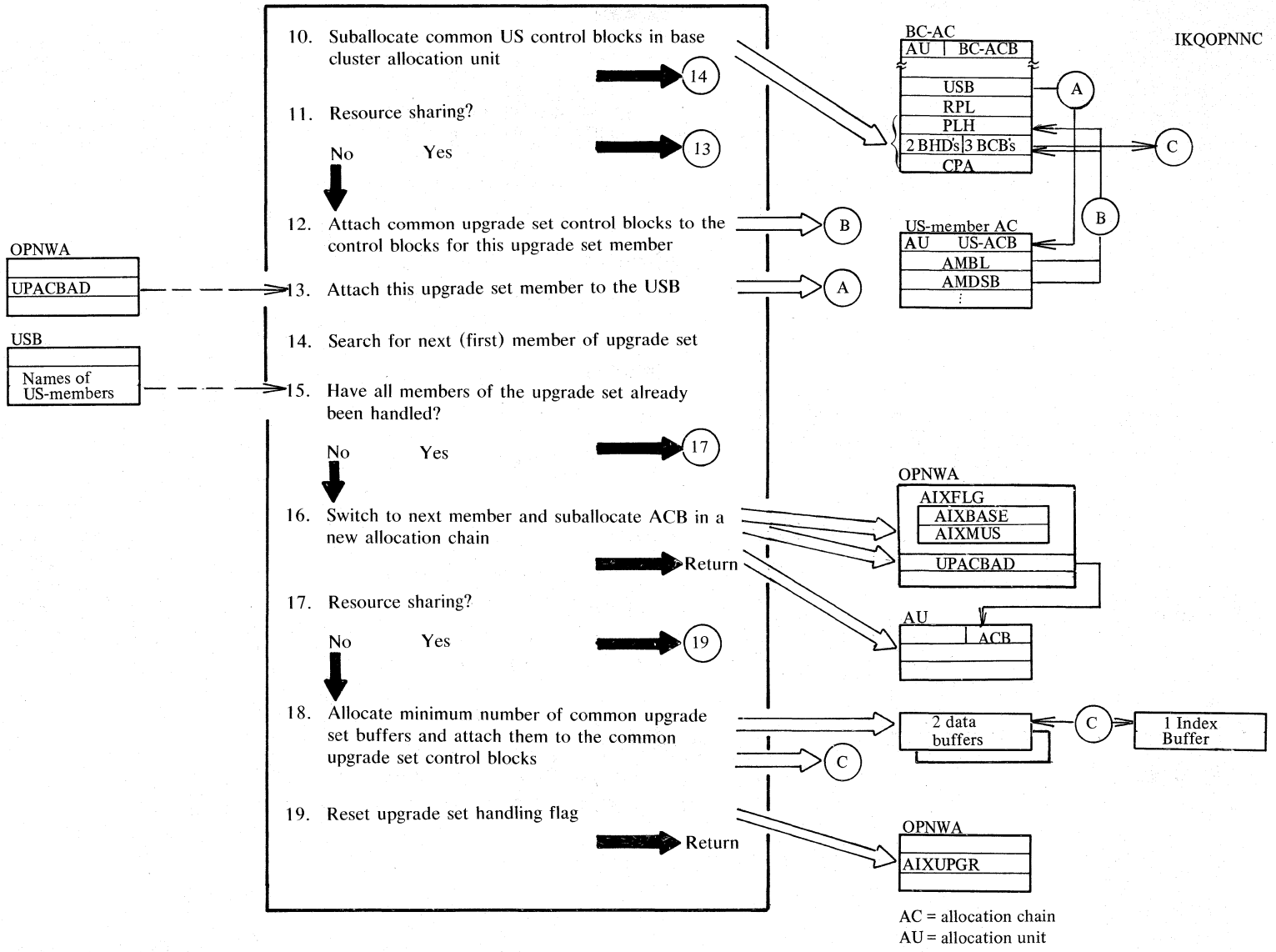


Diagram BD1. Release a cluster

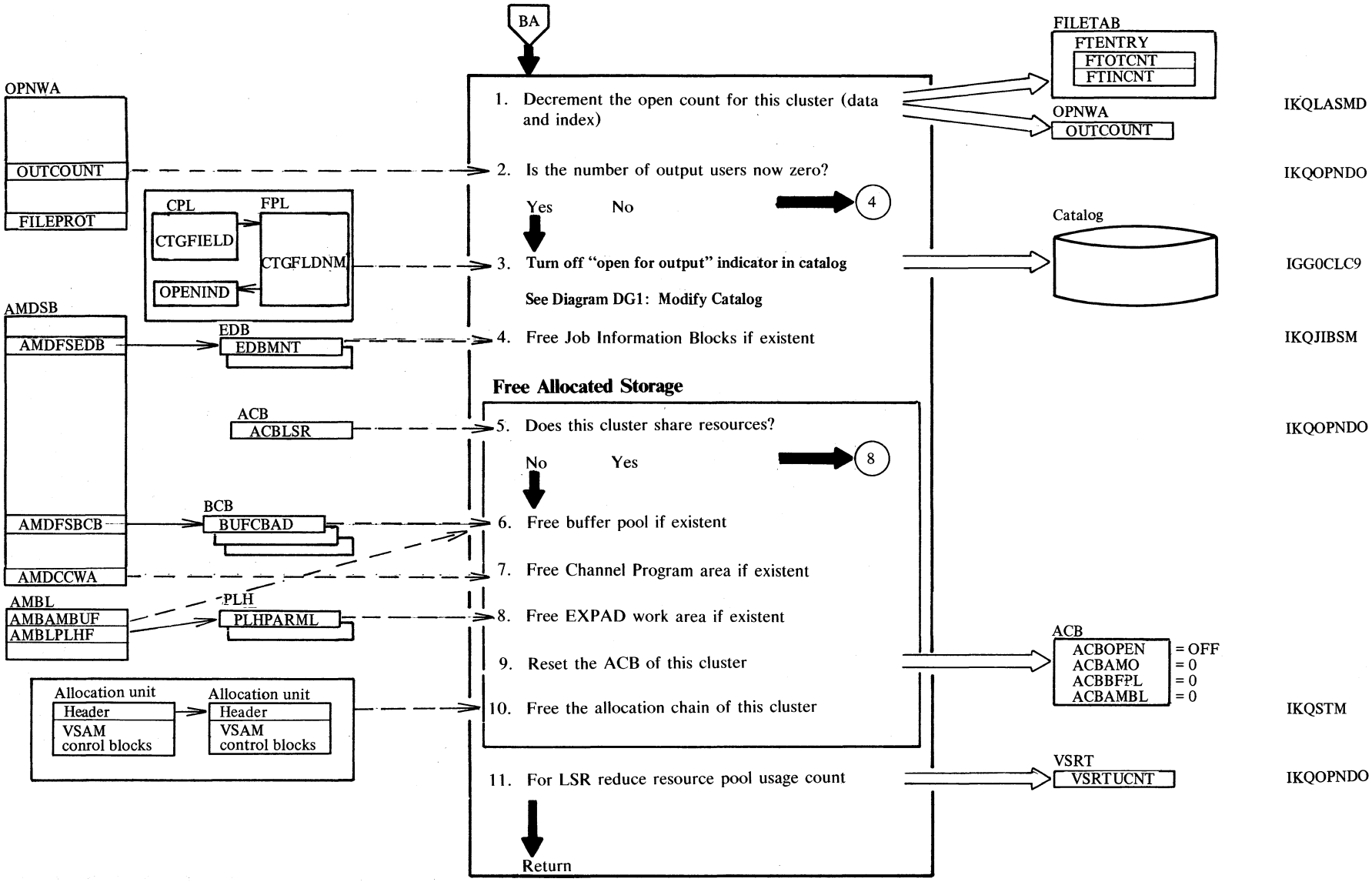


Diagram CA1. ISAM interface contents

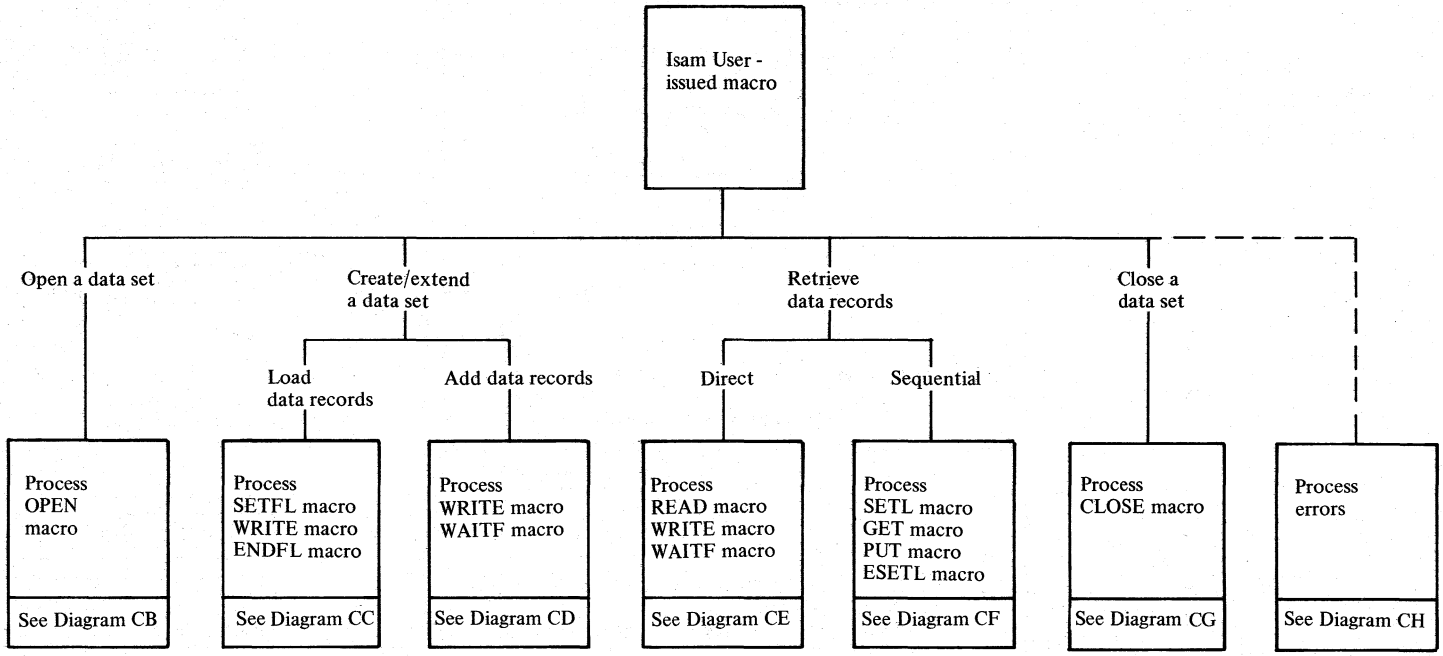


Diagram CB1. OPEN: Connect a user's ISAM program to a VSAM data set

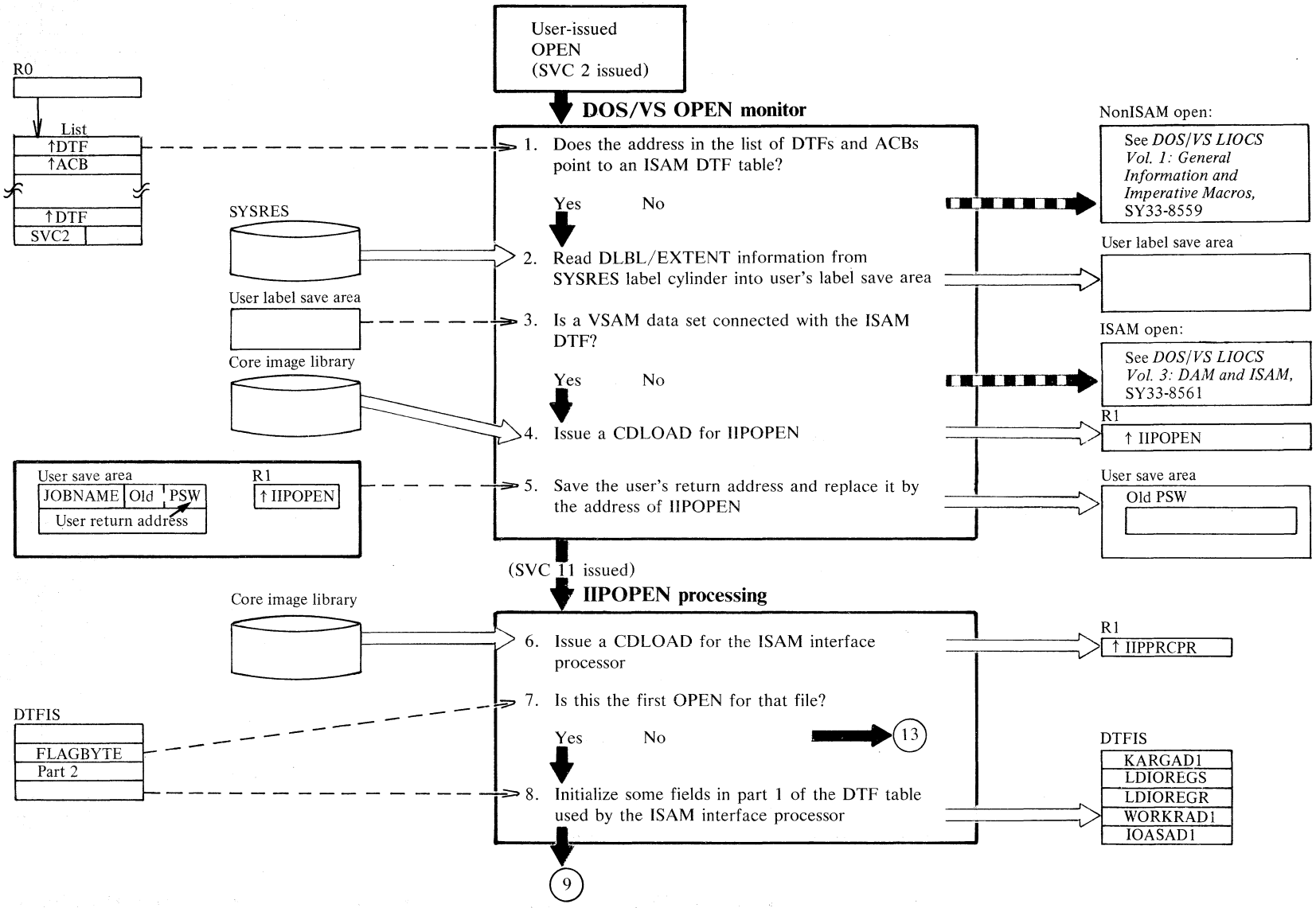
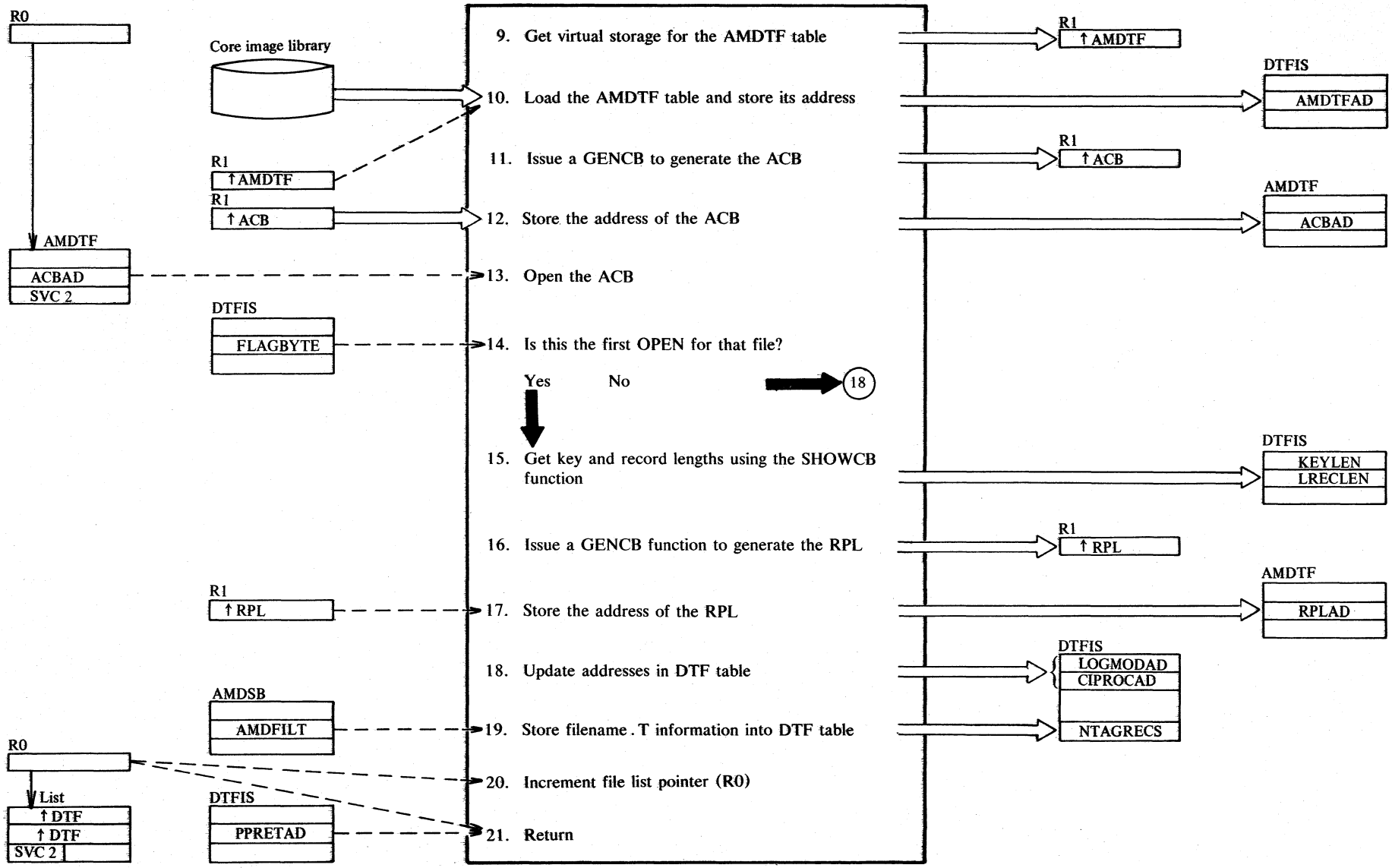


Diagram CB2. OPEN: Connect a user's ISAM program to a VSAM data set



Notes for Diagram CB (Part 1 of 2)

	Description	Module
1	The DOS/VS OPEN Monitor examines the DTF-type field (offset 20 of the address passed in the list) of the DTFs or ACBs. If the byte indicates an ISAM file (X'24', X'25', X'26', or X'27'), an SVC 2 is issued and $$BOPEN2 is fetched into the B-transient area, (If the file is not an ISAM file, the regular OPEN continues.) The list pointed to by register 0 may consist of pointers to DTF tables and/or ACBs. Register 0 is provided by the user program.	$$BOPEN $$BOPEND $$BOPEN1
3	LABEL TYPE'A' means that an ISAM file is connected with a VSAM data set.	$$BOPEN2
4	IIPOPEN is loaded into the SVA, and its address (in the SVA) is returned in R1	
6	IIPPRCPR is loaded into the SVA, and its address (in the SVA) is returned in R1.	
7	Bit X'80' in FLAGBYTE is set by IIPOPEN after the first successful OPEN for the VSAM data set.	IIPOPN00
8	The information needed to initialize the fields in part 1 of the DTF table is derived from part 2 of that table. The only purpose of this transformation is to provide better access to these items for the ISAM interface processor.	
10	The AMDTF table contains the parameter lists for the GENCB ACB, GENCB RPL, SHOWCB, and MODCB RPL macros, and the ERREXT parameter list used by the ISAM program in case of errors. The ADMTF table is loaded via the LOAD macro.	

	Description	Module
11	The parameter list in the AMDTF table for the GENCB macro is completed as follows: . Copy the filename from the DTF table . Specify the MACRF element according to the IOROUT parameter in DTFIS: IOROUT KEY DIR SEQ IN OUT LOAD X X X ADD X X X ADDRTR X X X X X RETRVE X X X X X	
13	The address of the ACB (pointed to by R0) is followed by a non-zero byte to make sure that control returns to IIPOPEN after opening the ACB.	
15	Since key length and record length are not generated in the DTFIS macro expansion for retrieve files, this information is extracted from the ACB via a SHOWCB, The field LRECLEN eventually contains the VSAM record length for blocked records and the VSAM record length diminished by the key length for unblocked records.	IIPOPN00
16	The parameter list for the GENCB macro in the AMDTF table is completed by storing the key length, record length, and ACB address in it.	
18	The logic module address in the DTF table is replaced by the address of the LOAD branch vector in IIPPRCPR or the address of the general branch vector, depending on whether the ISAM file is a LOAD file or not (LOGMODAD). The beginning address of IIPPRCPR is also stored (CIPROCAD).	

Notes for Diagram CB (Part 2 of 2)

Description

18 (cont) The LOGMODAD field is used to pass control from the user to data management. The address of the ISAM module is replaced by the branch vector address in IIPPRCPR and control automatically goes to the ISAM interface processor instead of the ISAM logic module. CIPROCAD is referenced when the user issues a SETFL, ENDFL, or SETL macro to pass control from the $$B-phases to the ISAM interface processor (CIPROCAD is a pointer to the address list at the very beginning of IIPPRCPR).

21 If there is no further element in the list, control is returned to the instruction in the user program that follows the SVC 2. Otherwise, control is returned to the SVC 2 instruction.

Diagram CC1. Load data records

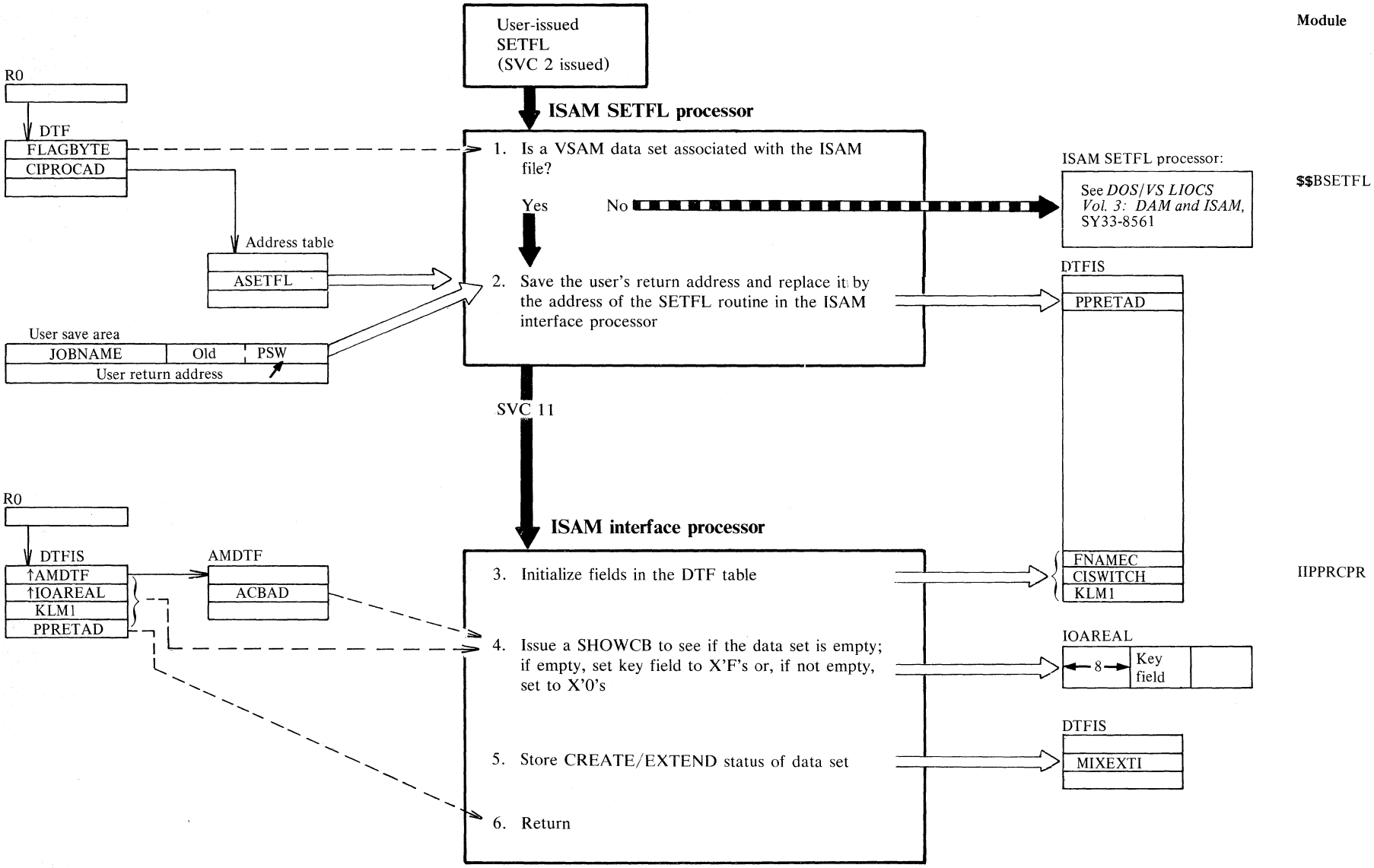


Diagram CC2. Load data records

Module

IIPPRCPR

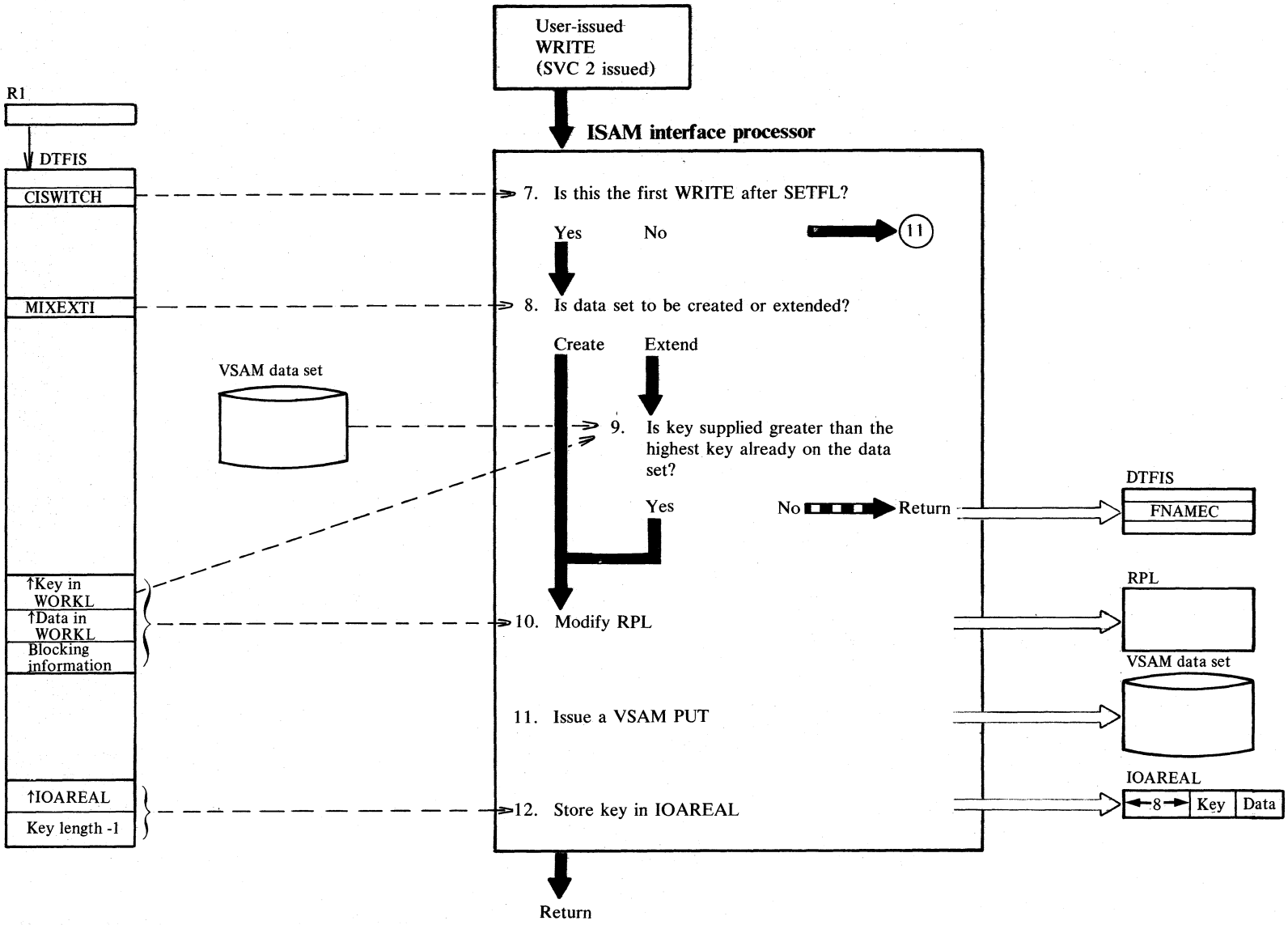
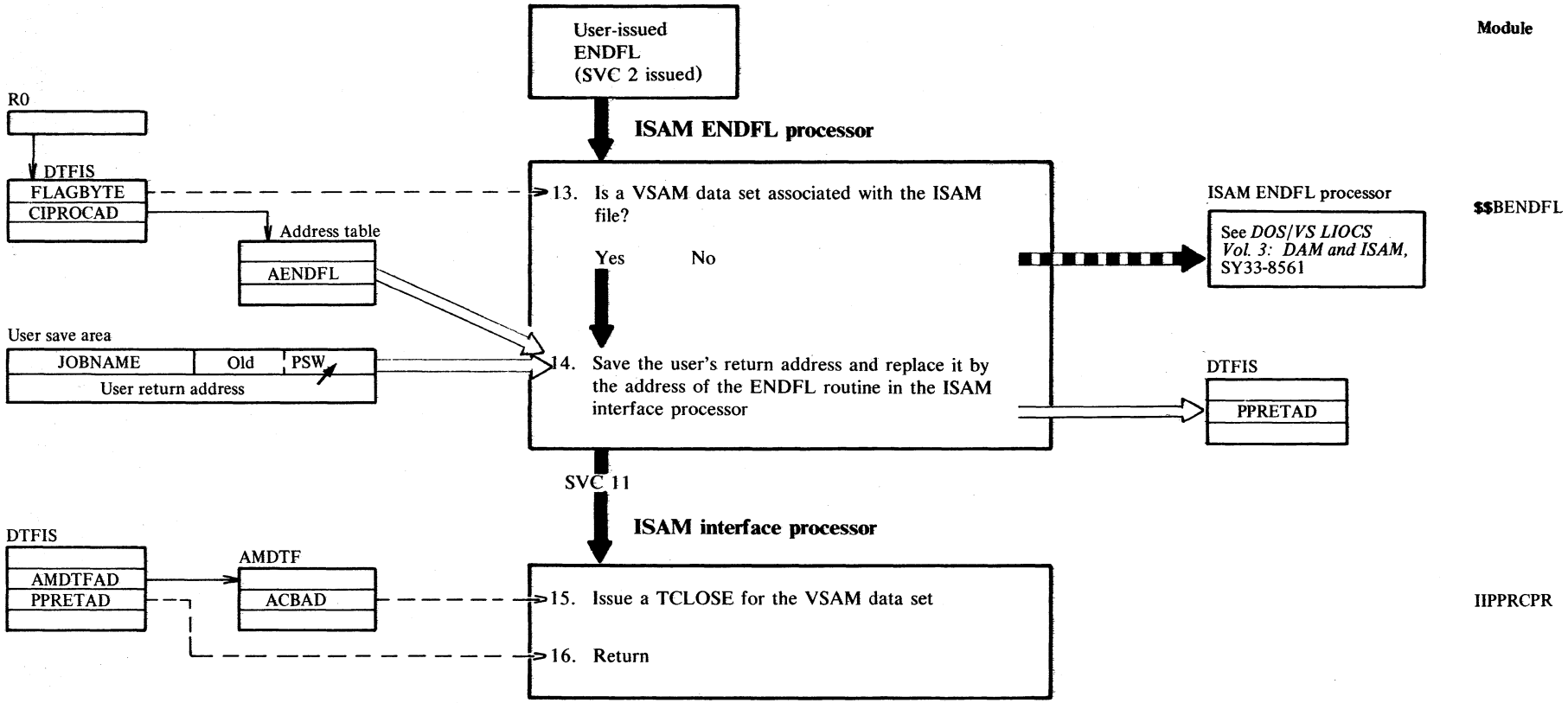


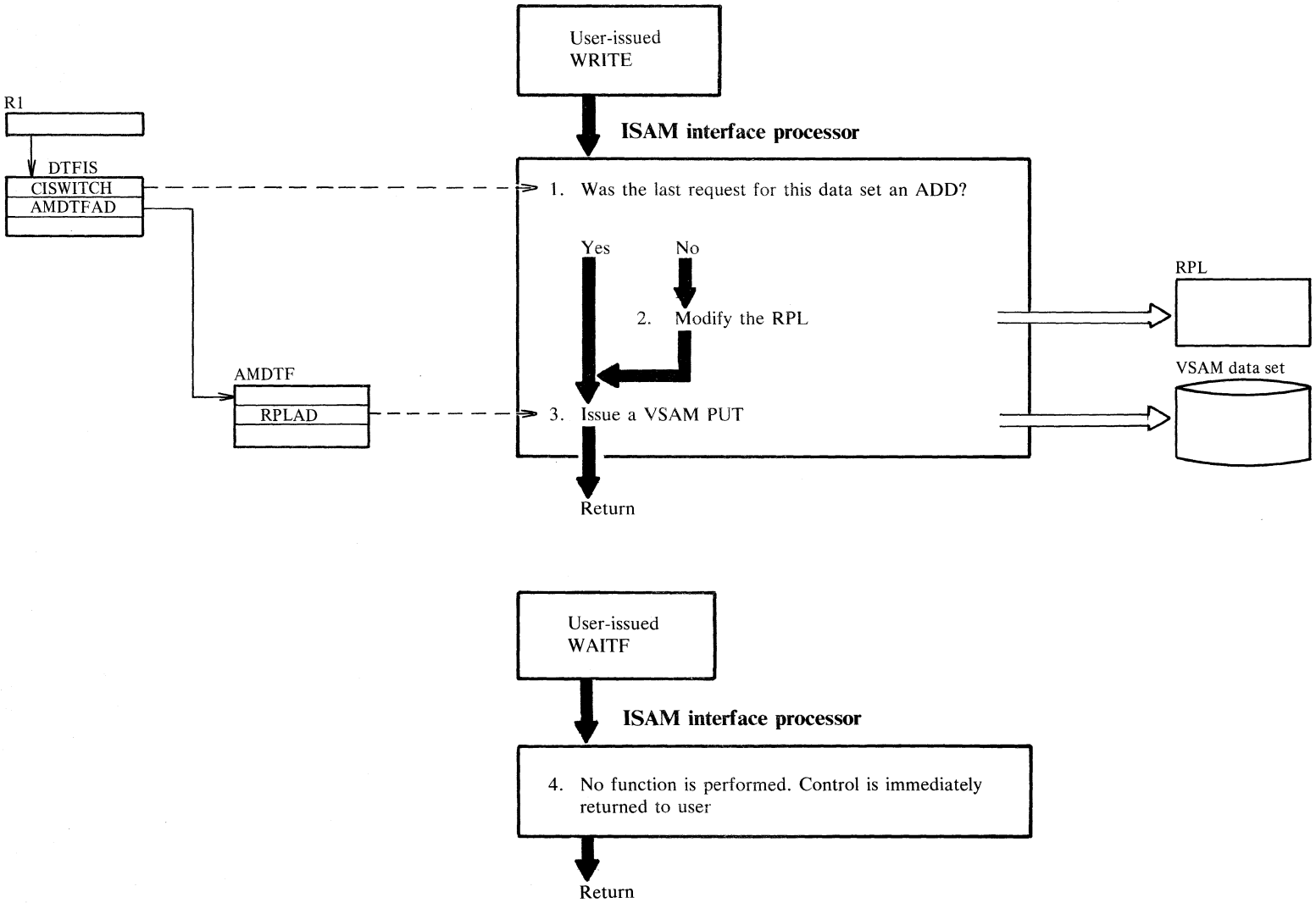
Diagram CC3. Load data records



Notes for Diagram CC

	Description	Module	Label
1	Bit 0 in the flag byte (offset 16 in the DTF table) is set by IIPOPEN after a successful OPEN of the data set.	$$BSETFL	
3	Fields in the DTF are set as follows: • FNAMEC to X'00' • CISWITCH to 'LOAD' and 'FIRST WRITE' • KLM1 is initialized with 'key length -1'. The key length is derived from the field KEYLEN which was initialized by IIPOPEN.	IIPPRCPR	SETFL
4	The key information in this form in IOAREAL is used after SETFL by various problem programs, for example, PL/I.		
7	Control is passed from the user to the ISAM interface processor via the LOGMODAD field in the DTF table, which, after OPEN, contains the address of the LOAD branch vector in the ISAM interface processor.		
7-8	The two switches referenced in these steps are set by the ISAM interface processor routine SETFL.		WRITENKL
9	To perform the sequence check, a dummy GET is issued that is prepared by a MODCB with the OPTCD code: KEY, DIR, SYN, NUP, KGE, FKS, LOC. If the return code following the GET operation is zero, this indicates that a sequence error has occurred, that is, that a record with a key equal to or greater than that in WORKL is already on the data set. If the return code following the GET operation is not zero, control is passed to the general error routine ERGN in IIPPRCPR. This routine analyzes the error. If the error is 'NO RECORD FOUND' and the 'FIRST WRITE' bit in CISWITCH is on, the error is ignored and normal processing continues. Note: The sequence check for subsequent PUTs is done by VSAM.		
10	A MODCB is issued with the following OPTCD code: KEY, SEQ, SYN, NUP, MVE. The AREA element in the MODCB parameter list is initialized. It specifies the address of WORKL. Since the address of WORKL is not supplied in the DTF macro expansion for LOAD files, it is derived from the address of the key in WORKL if RECFORM=FIXUNB and the address of the data in WORKL if RECFORM=FIXBLK.		
12	The key is stored in IOAREAL mainly for the benefit of problem programs (for example, PL/I) which may check this field.		
13	Bit 0 in the flag byte (offset 16 in the DTF table) is set by IIPOPEN after a successful OPEN of the VSAM data set.	$$BENDFL	
15-16		IIPPRCPR	ENDFL

Diagram CD1. Add data records



Notes for Diagram CD

	Description	Module	Label
1	Control is passed from the user to the ISAM interface processor via the LOGMODAD field in the DTF table, which, after OPEN, contains the address of the general branch vector in the ISAM interface processor.	IIPPRCPR	WRITENKA
2	A MODCB is issued with the following OPTCD code: KEY, DIR, SYN, NUP, MVE.		
4	See note 1.		WAITF

Diagram CE1. Direct processing of data records

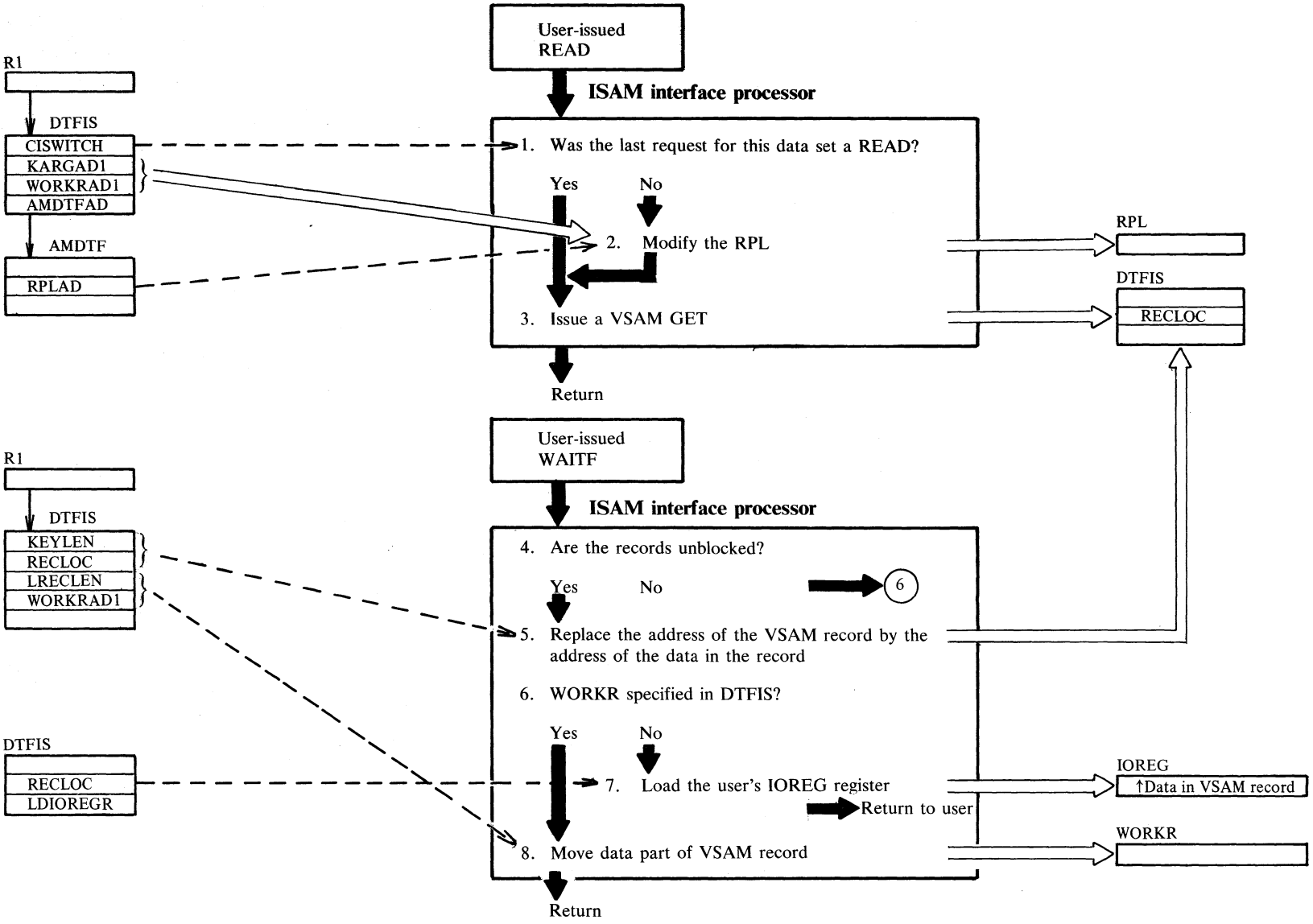
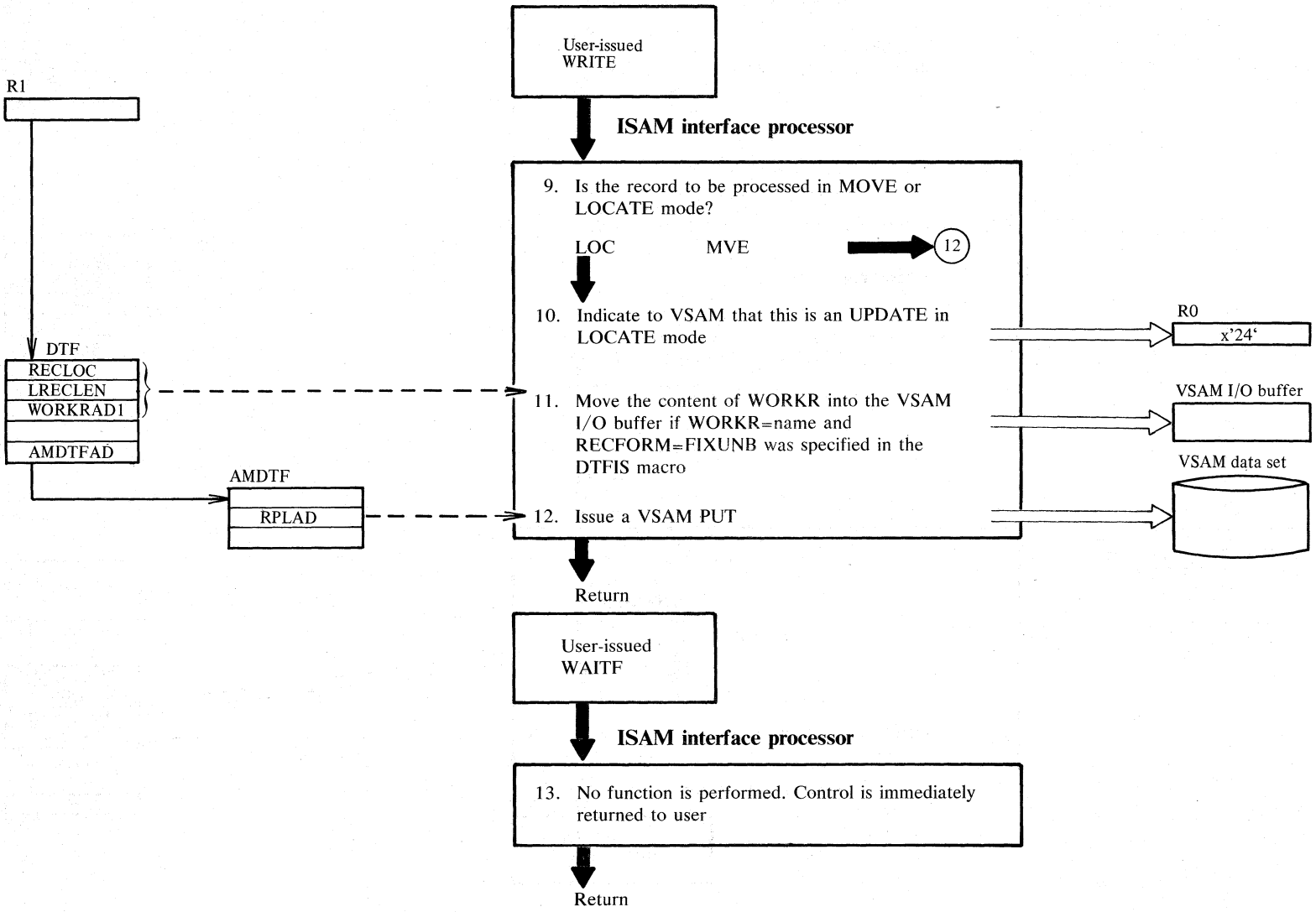


Diagram CE2. Direct processing of data records

Module

IIPPRCPR



Notes for Diagram CE

	Description	Module	Label
1	Control is passed from the user to the ISAM interface processor via the LOGMODAD field in the DTF table, which, after OPEN, contains the address of the general branch vector in the ISAM interface processor.	IIPPRCPR	READLEU
2	The records are processed by VSAM in MOVE mode when WORKR=name and RECFORM=FIXBLK are specified in the DTFIS. The OPTCD parameters for the RPL modifications are KEY, DIR, SYN, UPD, KEQ, FKS, and MVE. The records are processed by VSAM in LOCATE mode in all other cases, and the OPTCD parameters are KEY, DIR, SYN, UPD, KEQ, FKS, and LOC. The WORKRAD1 field in the DTF table (which contains a pointer to WORKR) used for MOVE mode is initialized by IIPOPEN, as well as KARGAD1 (which contains a pointer to the key).		
3	If LOCATE mode is specified, VSAM returns the address of the VSAM record in the DTF AFTER the GET operation.		
4	See note 1.		WAITF
9	See notes 1 and 2.		WRITEKEY
10	Since VSAM does not allow update of records in LOCATE mode on the one hand and the ISAM interface processor, on the other hand, needs update in LOCATE mode, a special PUT is issued which, by means of the value passed in register 0, indicates to VSAM that the request comes from the ISAM interface program.		
13	See note 1.		WAITF

Diagram CF1. Sequential processing of data records

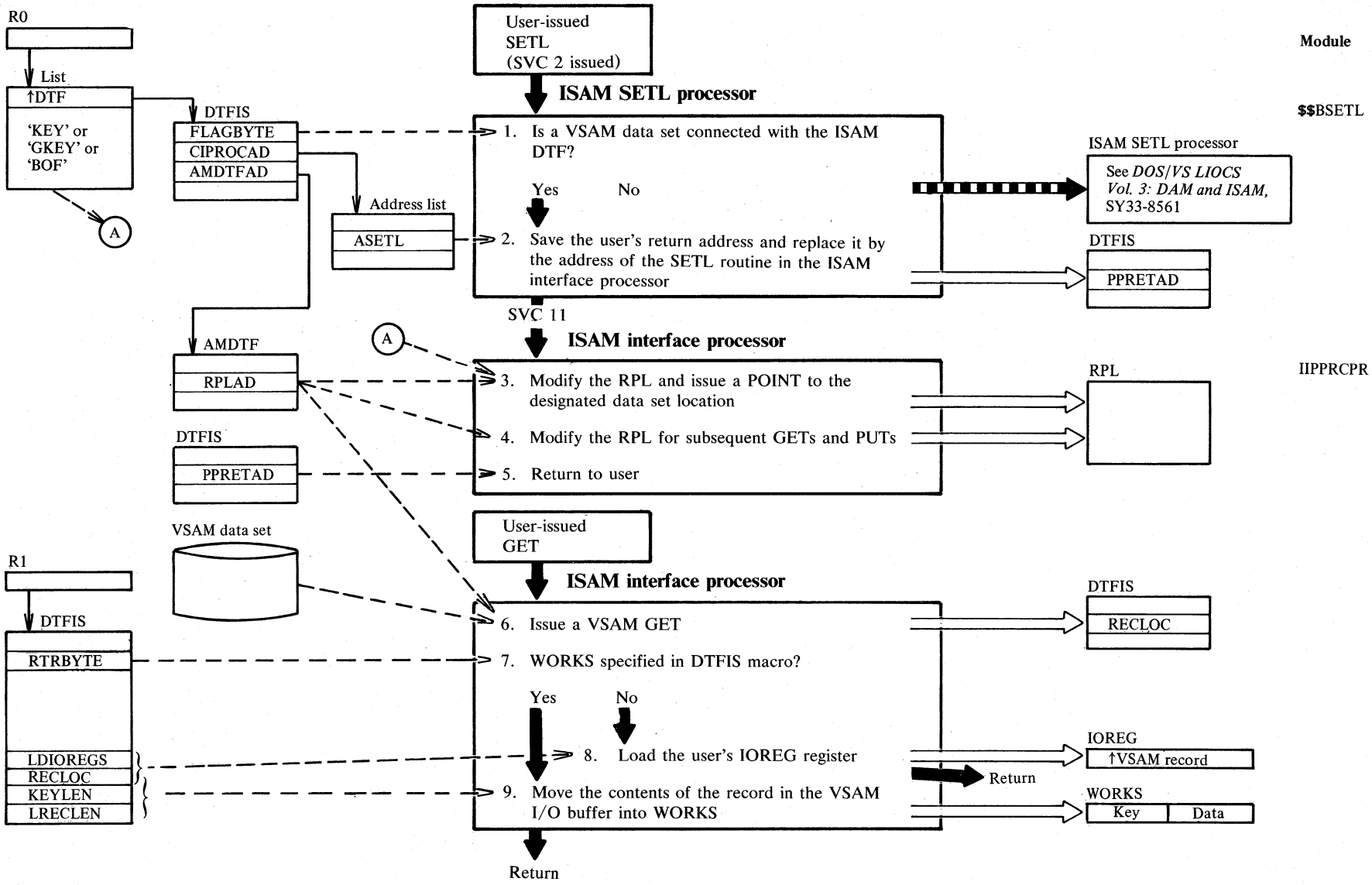
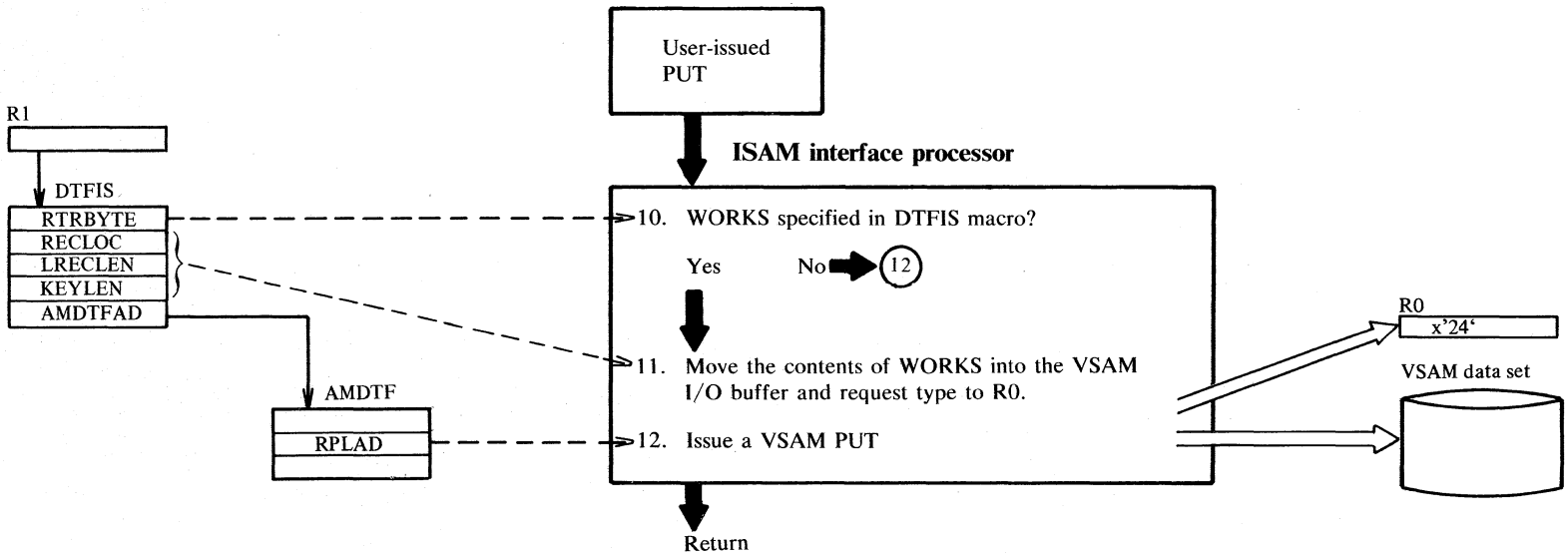
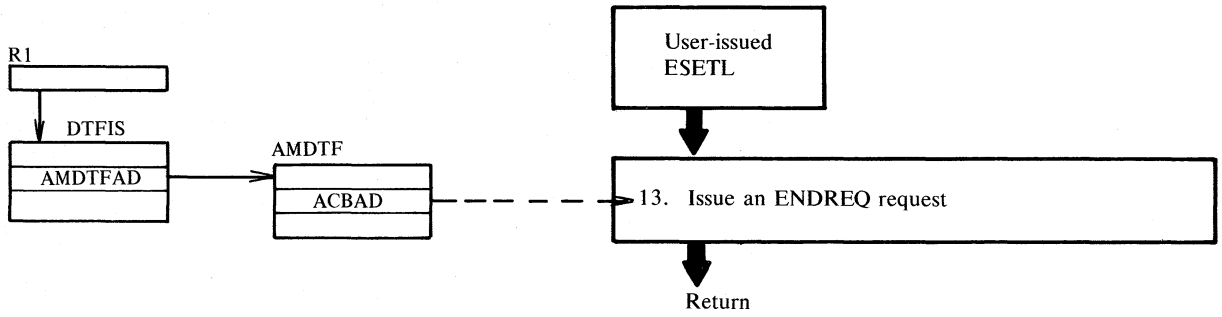


Diagram CF2. Sequential processing of data records

Module

IIPPRCPR

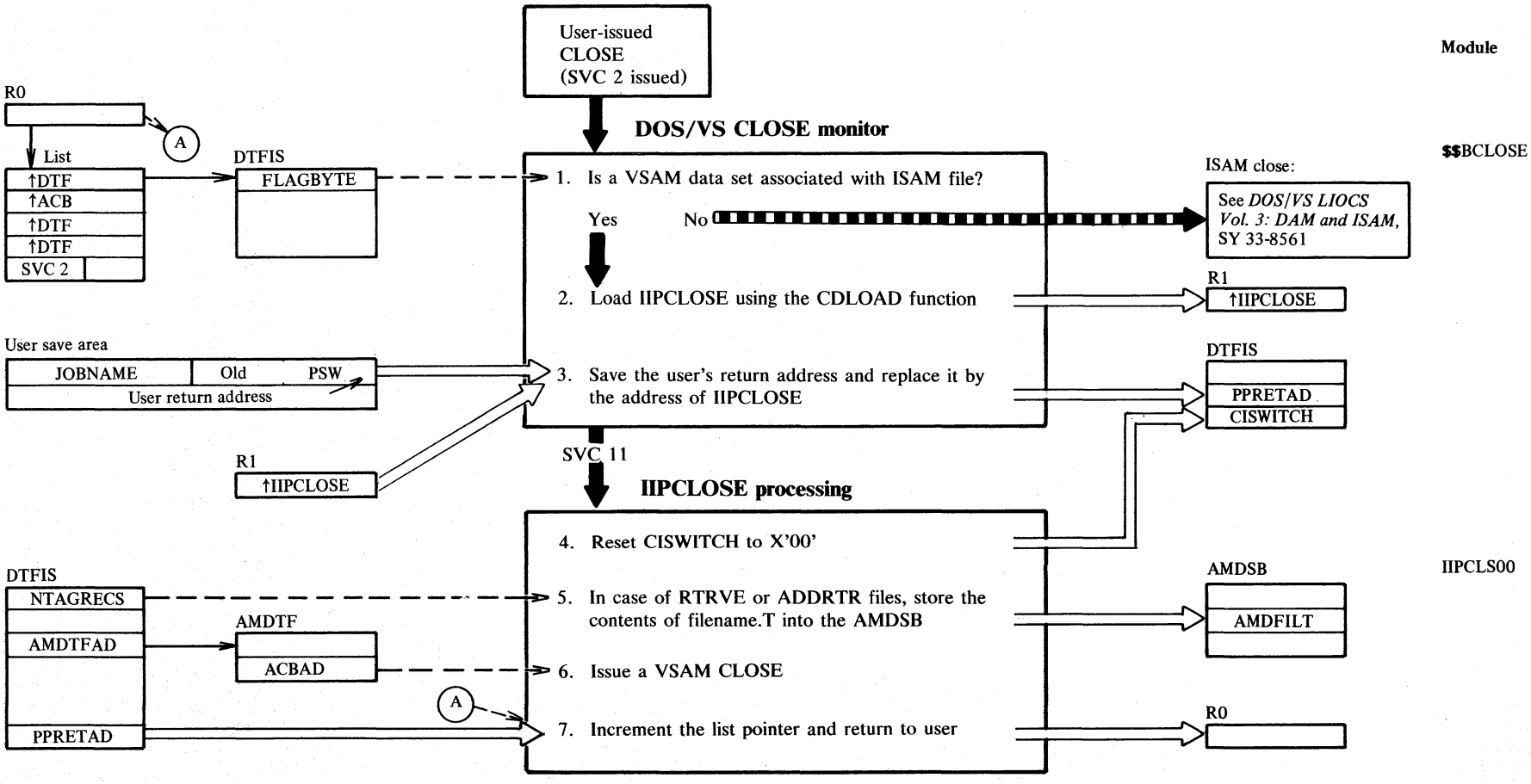




Notes for Diagram CF

	Description	Module	Label
1	Bit 0 in the flag byte (offset 16 in the DTF table) is set by IIPOPEN after a successful OPEN of the data set.	$$BSETL	
3-5	A MODCB is issued with the following OPTCD parameters to prepare for subsequent GETs and PUTs: KEY, SEQ, SYN, UPD, and LOC. Thus, all records are to be processed in LOCATE mode.	IIPPRCPR	SETL
6	Control is passed from the user to the ISAM interface processor via the LOGMODAD field in the DTF table, which, after OPEN, contains the address of the general branch vector in the ISAM interface processor.		GET
10	See note 6.		PUT
10-12	Since VSAM does not allow update of records in LOCATE mode on one hand and the ISAM interface processor, on the other hand, needs update in LOCATE mode, a special PUT is issued which, by means of the value passed in R0, indicates to VSAM that the request comes from the ISAM interface program.		
13	See note 6.		ESETL

Diagram CG1. CLOSE: Disconnect a user's ISAM program from a VSAM data set



Notes for Diagram CG

	Description	Module
1	Bit 0 in the flag byte (offset 16 in the DTF table) is set by IIPOPEN after a successful OPEN of the data set.	IIPCLS00
7	When no elements are left in the list, control is returned to the instruction immediately following the SVC 2 in the user program. Otherwise, control is returned to the SVC 2.	$$BCLOSE

Diagram CH1. ISAM interface error processing

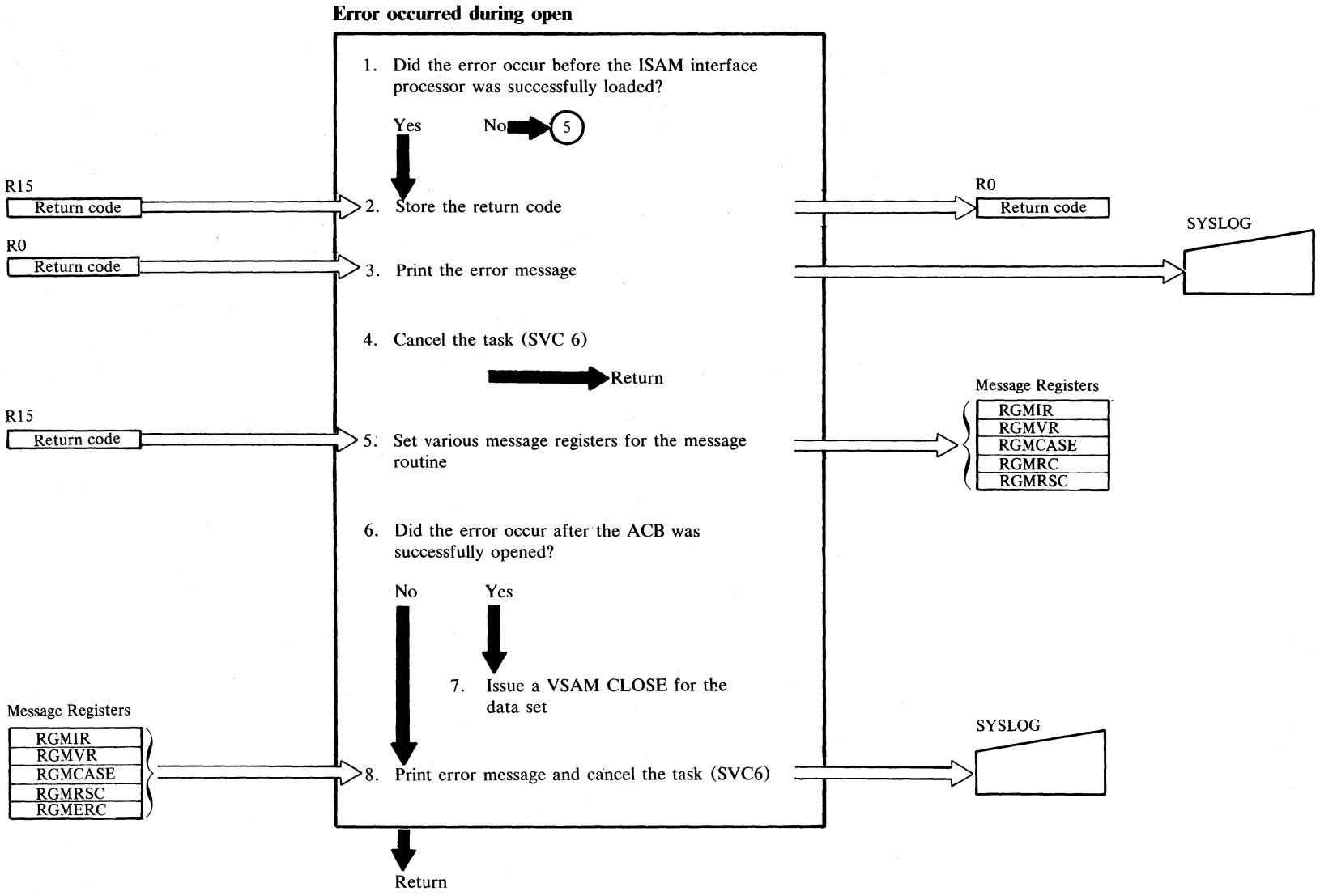
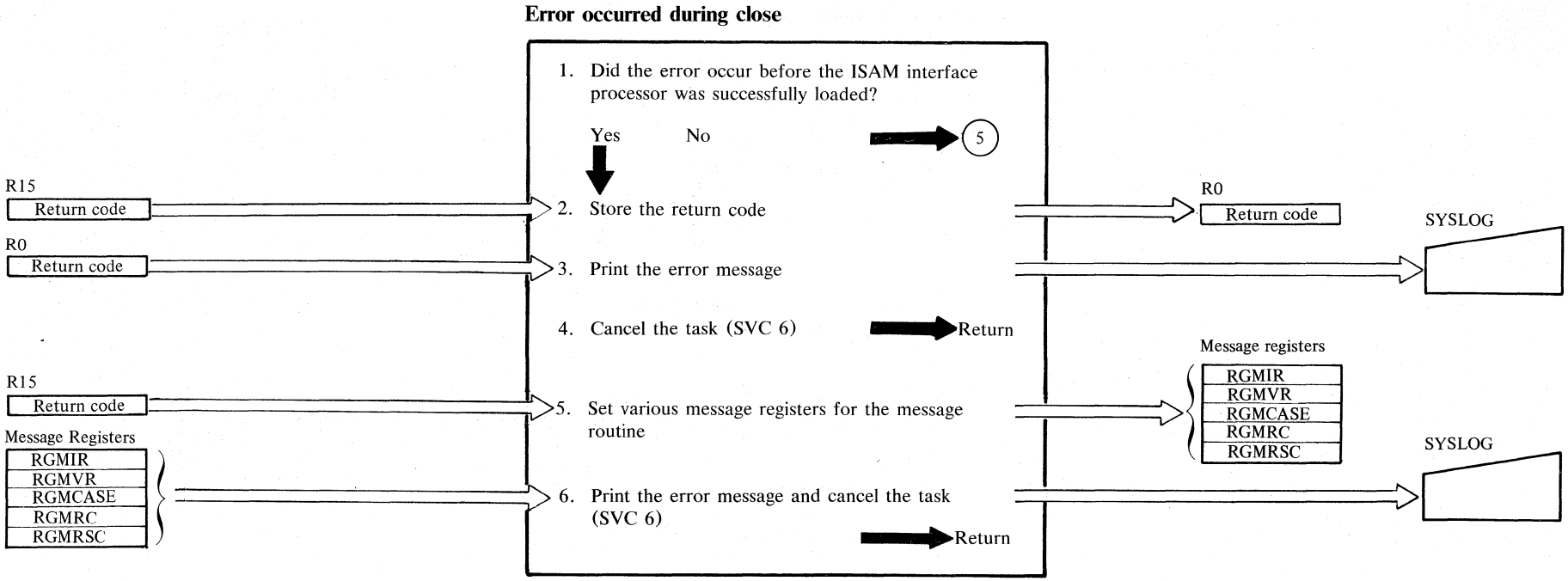


Diagram CH2. ISAM interface error processing



Notes for Diagrams CH1, CH2

	Diagram CH1	Module
1	If the ISAM interface processor is not loaded, a separate message routine must be used.	$$BOPEN2 IIPOPN00
2	Return code is moved to R0, in preparation for the message routine.	
3	Return code loaded into R0 during step 2 determines message.	IIPBMR00
5	Registers are set up for IIPPRCMR	IIPOPN00
7	Data set must be closed	IIPPRCPR
8	Uses registers set up in step 5.	IIPPRCMR

	Diagram CH2	Module
1	If the ISAM interface processor is not loaded, a separate message routine must be used.	$$BCLOSE
2	Return code is moved to R0, in preparation for the message routine.	
3	Return code loaded into R0 during step 2 determines message.	IIPBMR00
5	Registers are set up for IIPPCRMR	IIPCLS00
6	Uses registers set up in step 5.	IIPPRCMR

Diagram CH3. ISAM interface error processing

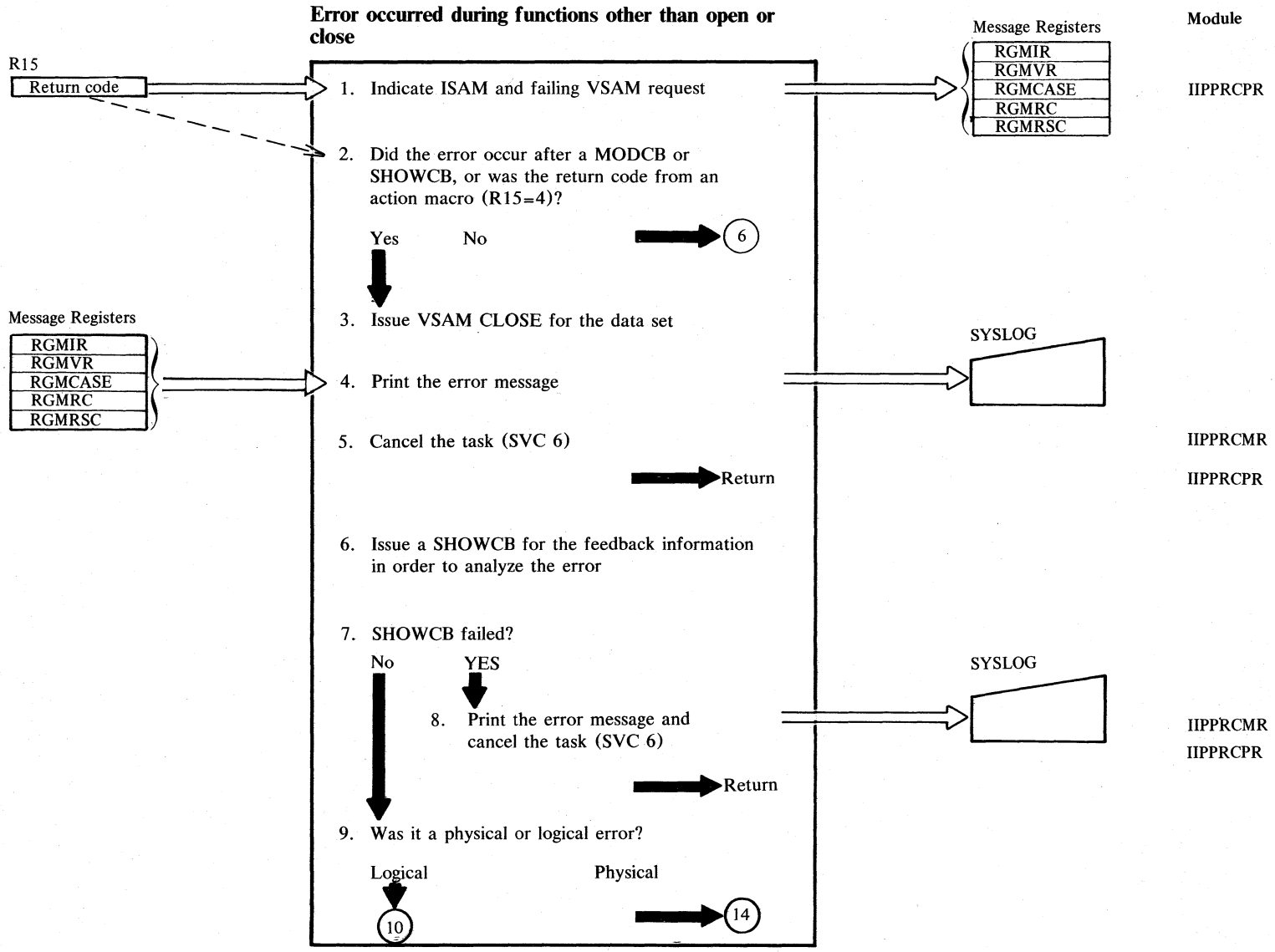


Diagram CH4. ISAM interface error processing

Error occurred during functions other than open or close (continued)

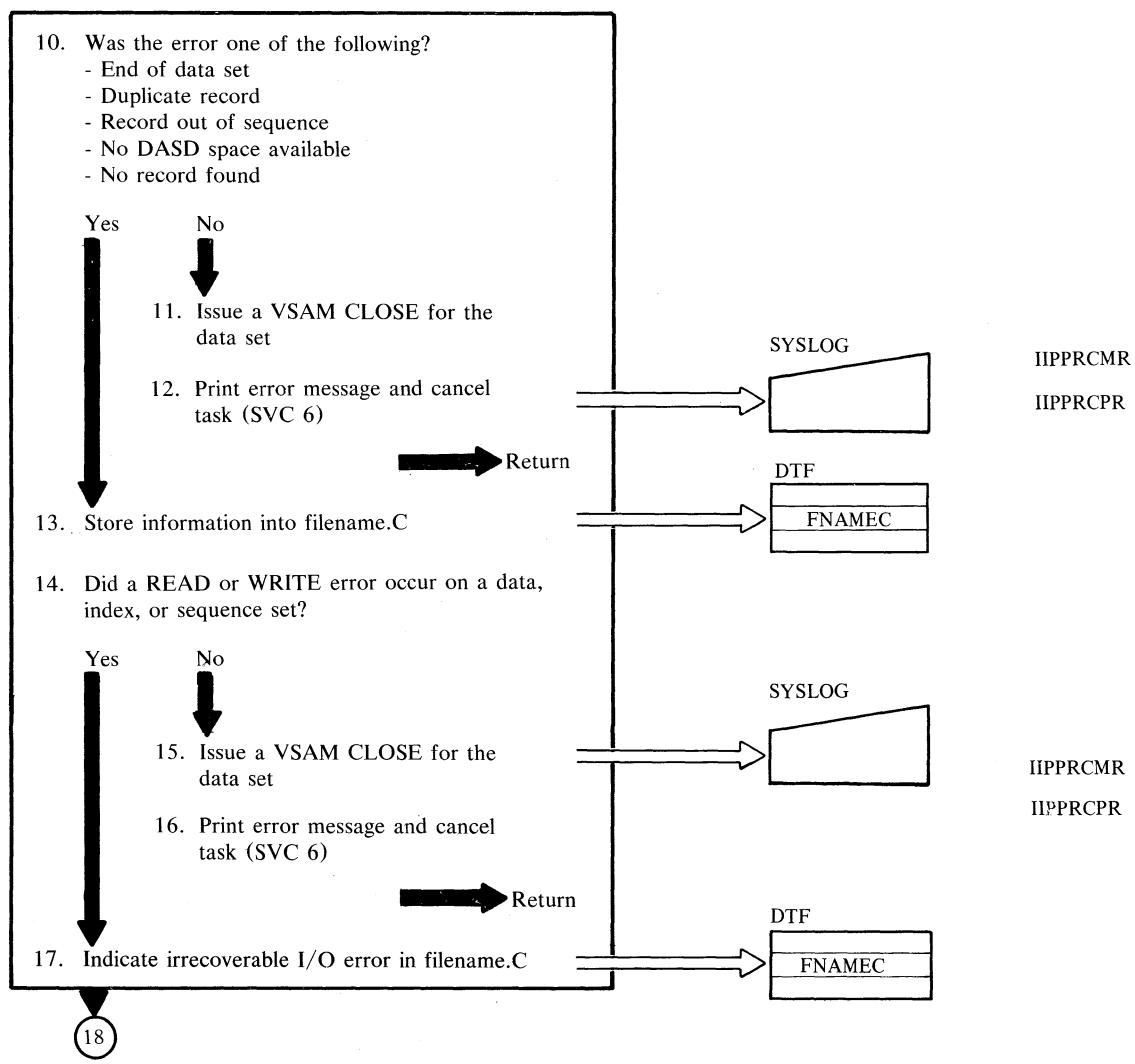


Diagram CH5. ISAM interface error processing

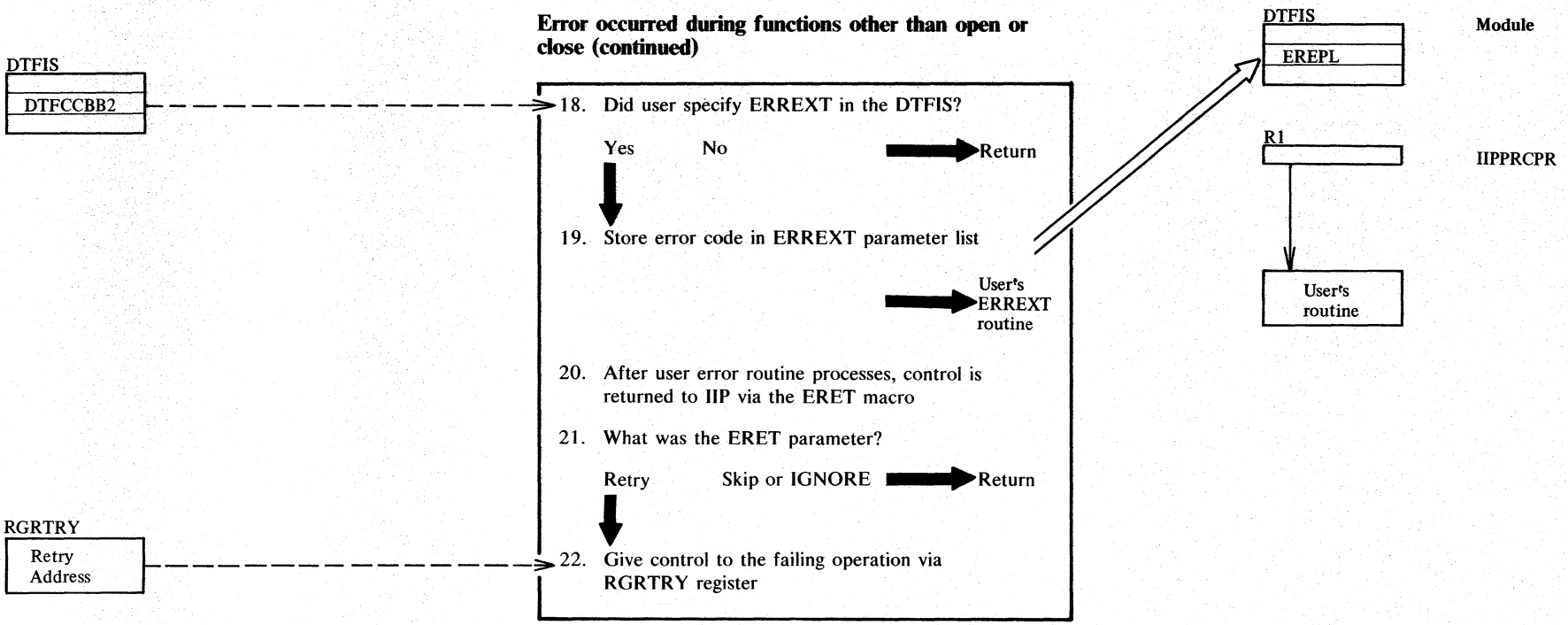


Diagram DA1. Catalog management contents

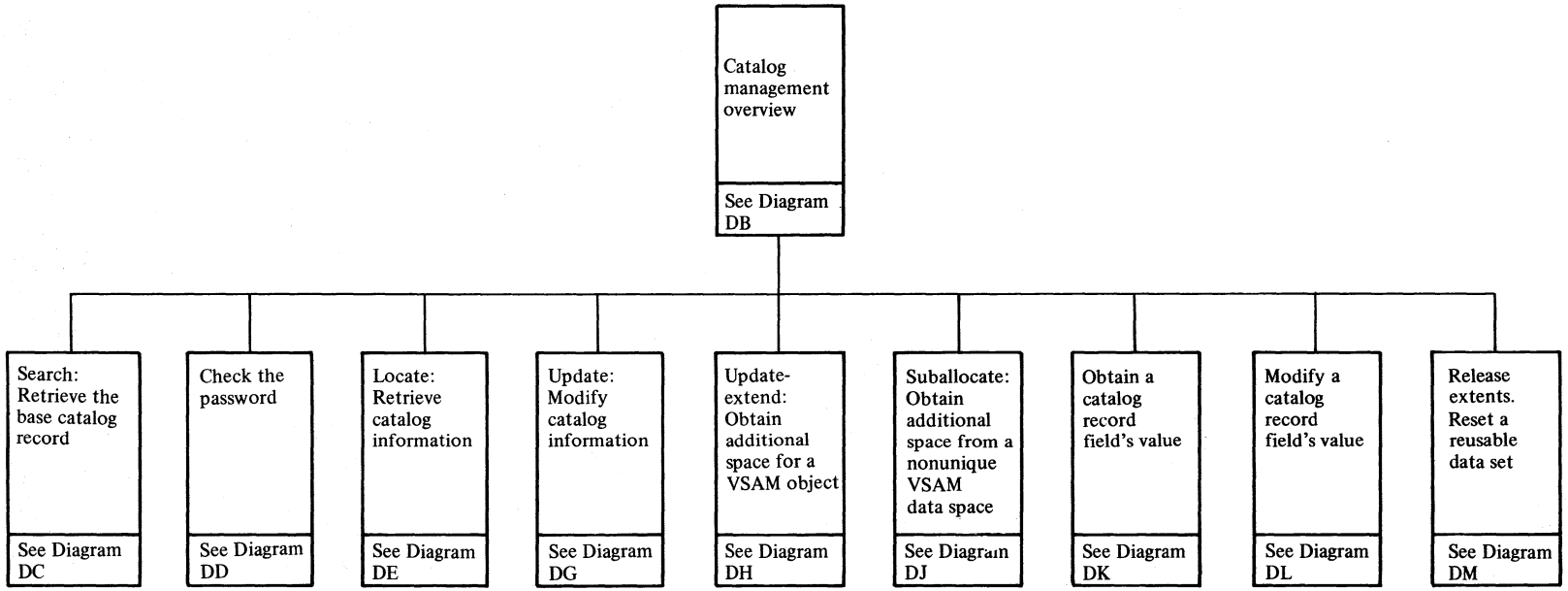


Diagram DB1. Catalog management overview

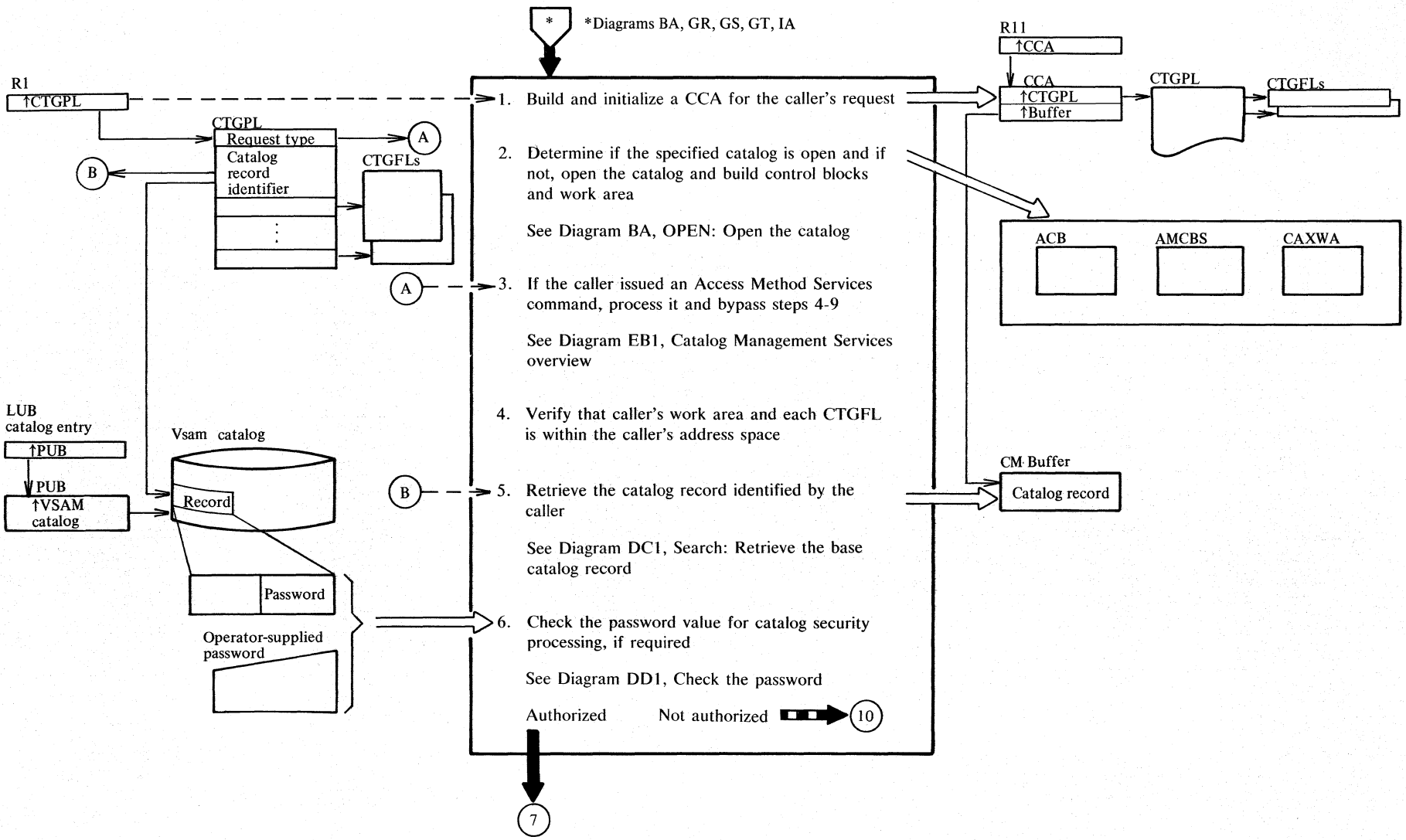
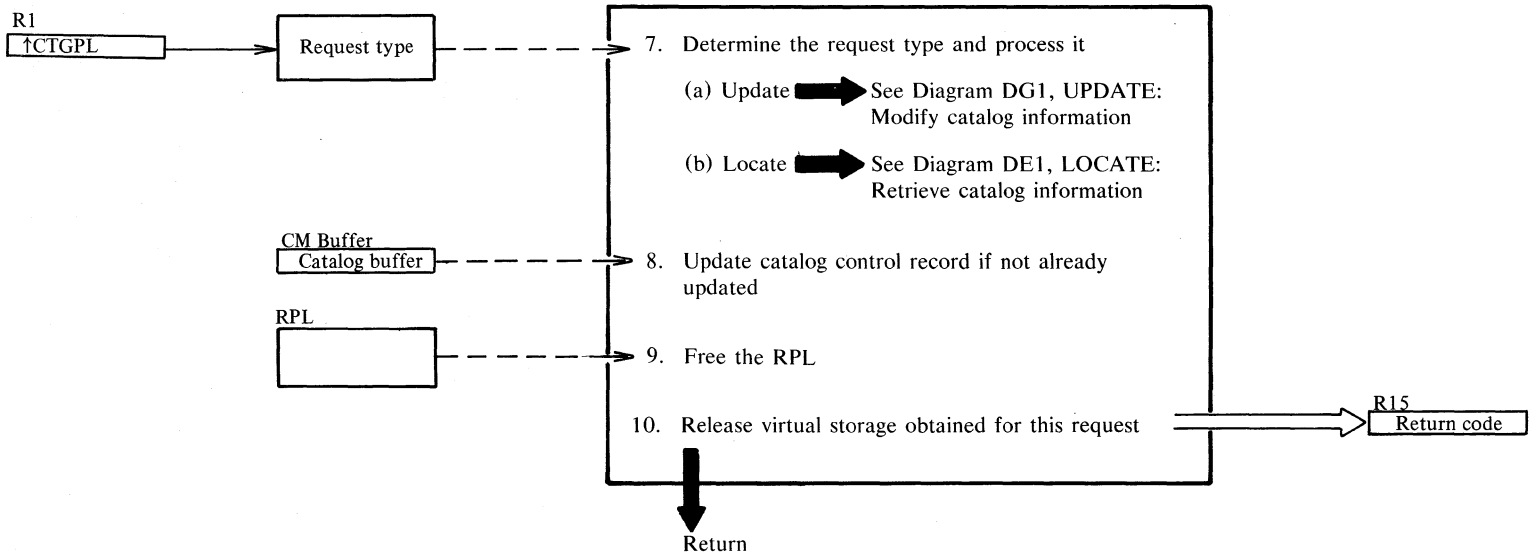


Diagram DB2. Catalog management overview



Notes for Diagram DB (part 1 of 2)

	Description	Module	Procedure
	VSAM catalog management (CM) is called with a CATLG macro instruction by VSAM Open, Close, and end-of-volume routines. In addition, a user's program can process VSAM catalog records by issuing an Access Method Services request. Access Method Services also issues the CATLG macro, which translates the request into an SVC 65 and a catalog parameter list (CTGPL).		
	The CATLG macro instruction checks to see if the called VSAM CM module is in storage. If it isn't, the module is loaded from the core image library. Register 1 contains the address of the caller's CTGPL. The CTGPL identifies which catalog record to process and what process to be perform.		
1	The catalog control area (CCA) contains data about catalog records retrieved to process the request. The CCA also contains a register save area that shows the flow of control between CM routines used to process the request.	IGG0CLC9	BLDCCA
	Each time a CM routine is called by another CM routine, the contents of registers 12, 13, and 14 are put in the CCA's register save area. See Data Areas for details about the CCA and CTGPL and Diagnostic Aids for information about the CCA register save area.		
	The catalog driver is then called to determine what request was issued and which routine processes the request.	IGG0CLAB	IGGPACDV
2	The master catalog's AMCBS is built, if it has not already been built. A check is then made to see if the master catalog is open. If it is not open, IGG0CLAD calls $$BOPEN to open the master catalog and then builds the master catalog's ACB and CAXWA. Control is returned to IGG0CLAC and then to IGG0CLAB.	IGG0CLAC IGG0CLAD $$BOPEN IGG0CLAD	IGGPMCO IGGPMCO2 IGGPMCO2
3	An Access Method Services command is translated into a catalog management services (CMS) request to define (create), alter, delete, or list catalog records. IGG0CLAT then returns to IGG0CLAB.	IGG0CLAT	IGGPCDVR
4	The caller's work area and each CTGPL is checked to ensure that it is within the caller's address space.	IGG0CLAY	IGGPSCNC
	The field-name value in the field parameter list (CTGFL) is used to obtain dictionary data that defines the field's characteristics and location within the record. Control is returned to IGG0CLAB.	IGG0CLAY	IGGPSCNC
5	The catalog record is identified by the caller's DSNAME value, volume serial number, or control interval number. If the CTGPL's catalog identifier addresses the record's control interval number, the catalog record can be retrieved without a search of the catalog's index. Control returns to the calling function.	IGG0CLAH	IGGPSCAT
6	The caller's request type determines the password value that, when supplied by the operator or input stream, allows the VSAM CM routines to complete the caller's request.	IGG0CLBM	IGGPCKAU
	The catalog cluster record is retrieved for catalog security processing. The password group occurrence data must also be retrieved, if it exists. Control is then returned to the catalog calling function.	IGG0CLAG IGG0CLAZ	IGGPGET IGGPEXT

Notes for Diagram DB (part 2 of 2)

	Description	Module	Procedure
7 (a)	An Update request modifies information in a catalog record. An Update request can also obtain direct-access space for the data set or index identified by the DSNAME value. Control is then returned to the catalog calling functions.	IGG0CLAV IGG0CLBB	IGGPUPD IGGPUPDE
(b)	A Locate request retrieves information from the catalog record. IGG0CLAB is called to determine those occurrences from which field data should be retrieved. Control is then returned to the catalog calling functions.	IGG0CLAZ IGG0CLBA	IGGPLOC IGGPTSTS IGGPGVAL IGGPGREC
8	When the VSAM catalog driver (IGG0CLAB) returns to the catalog first load module, IGGPRCU finds out if the CCR has been updated in virtual storage. If so, IGGPCCCR is called to update the CCR and returns to caller.	IGG0CLC9 IGG0CLAG	IGGPRCU IGGPCCCR
9	IGGPRPLF frees the RPL for other requests and returns to the catalog first load module.	IGG0CLAB	IGGPRPLF
	All virtual storage obtained for work areas and control blocks is freed and returned to the DOS/VS system. A return code is set in register 15 and control returns to the caller.	IGG0CLC9	IGGPRCU

Diagram DC1. Search: Retrieve the base catalog record

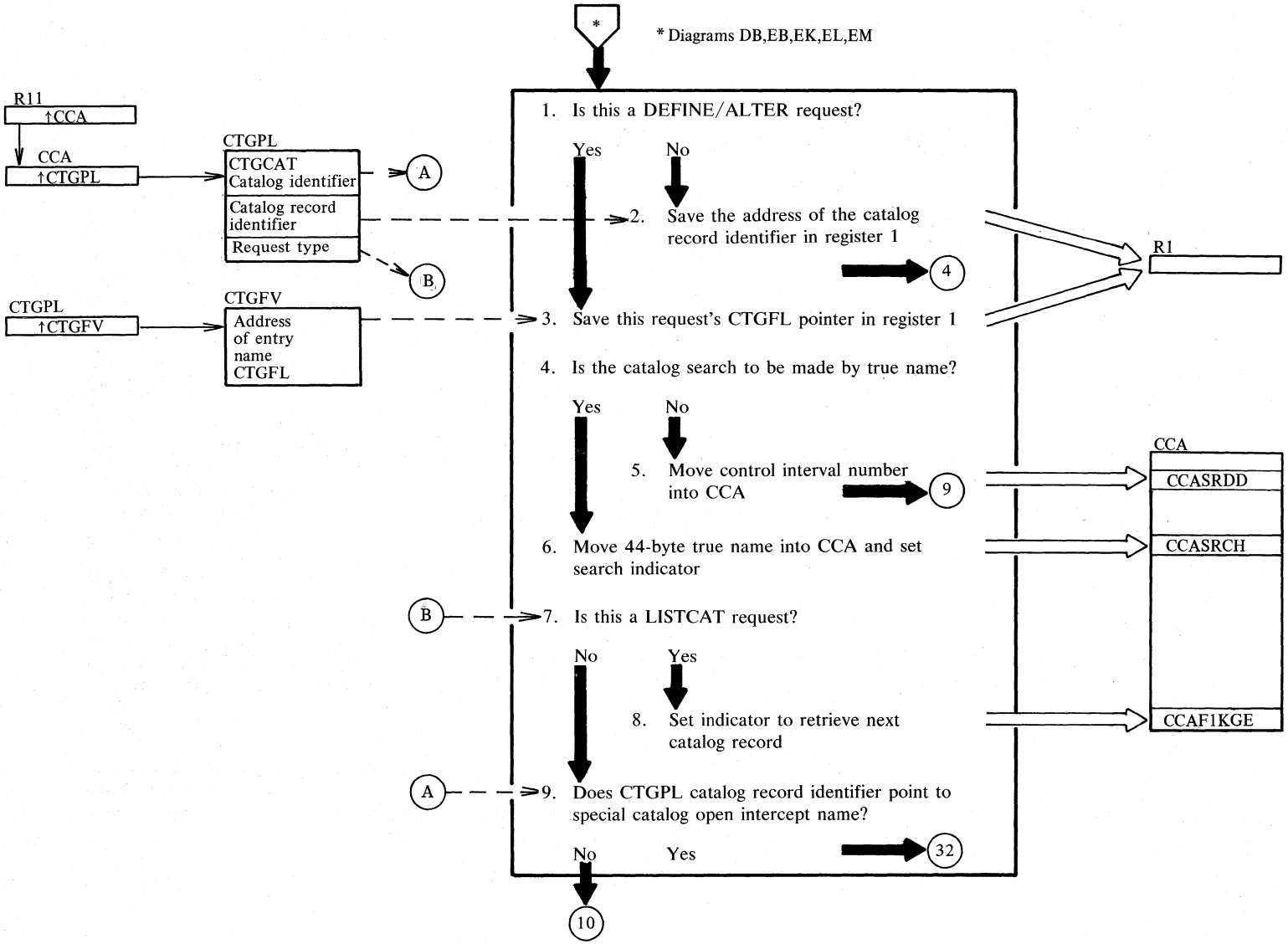


Diagram DC2. Search: Retrieve the base catalog record

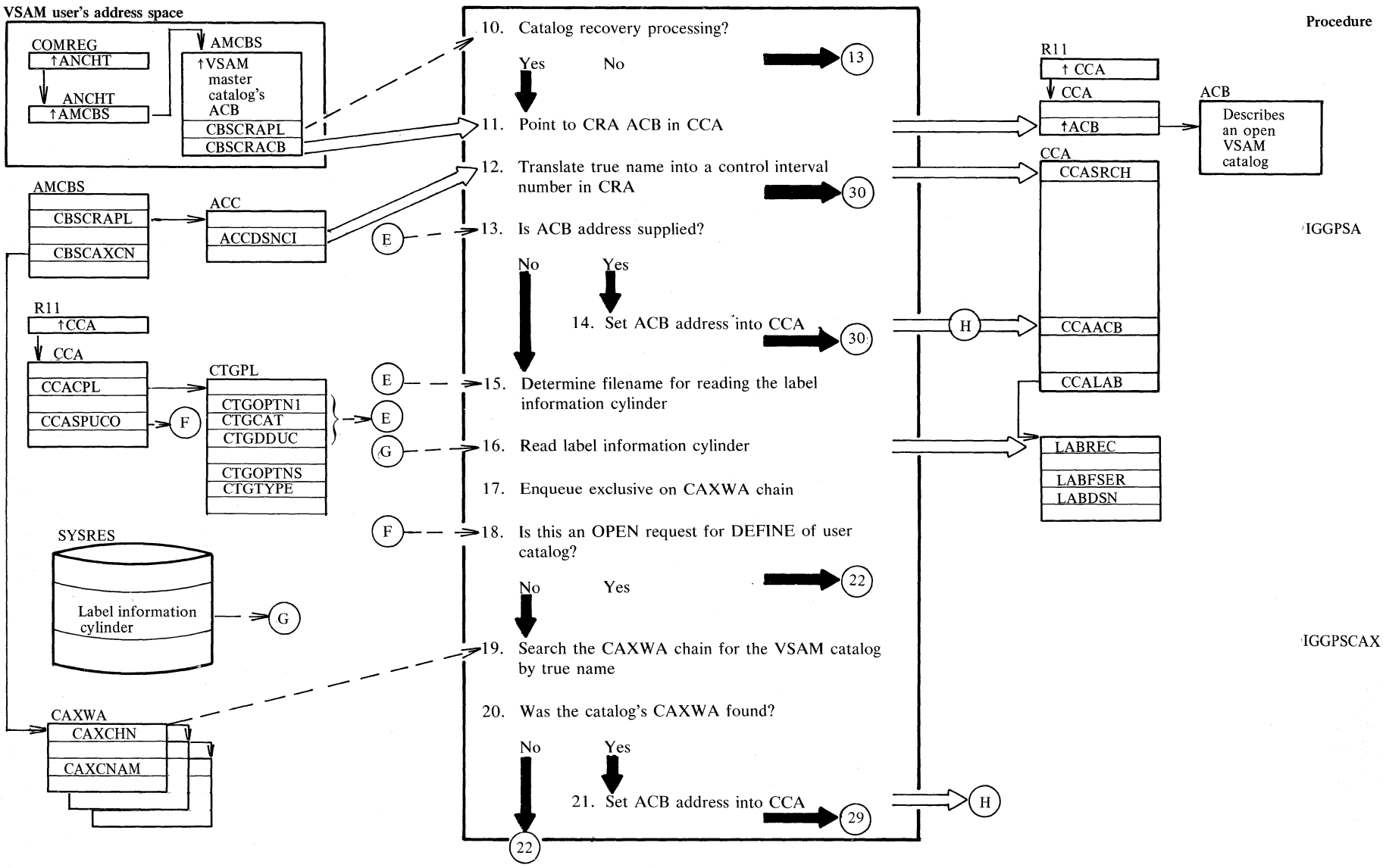


Diagram DC3. Search: Retrieve the base catalog record

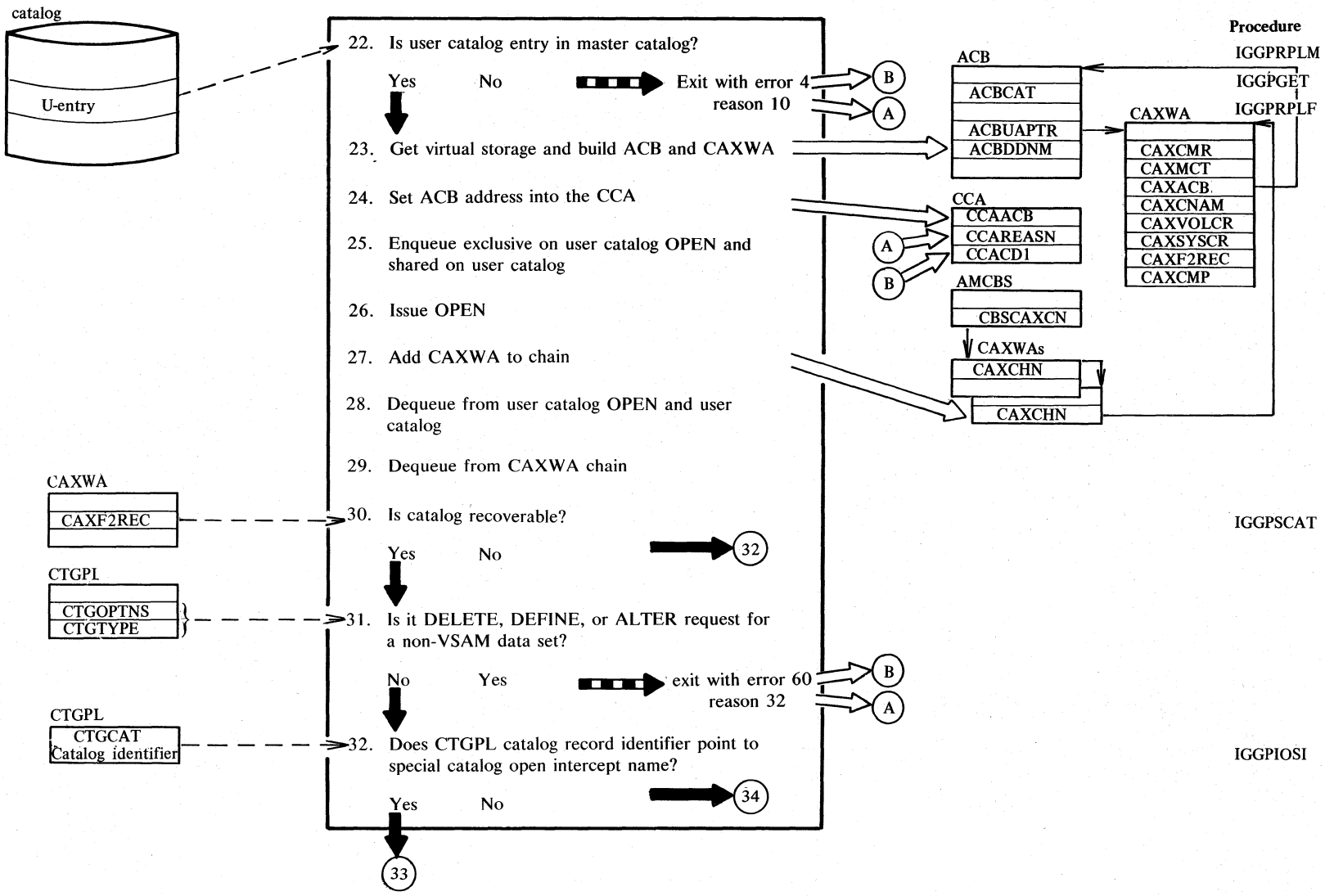
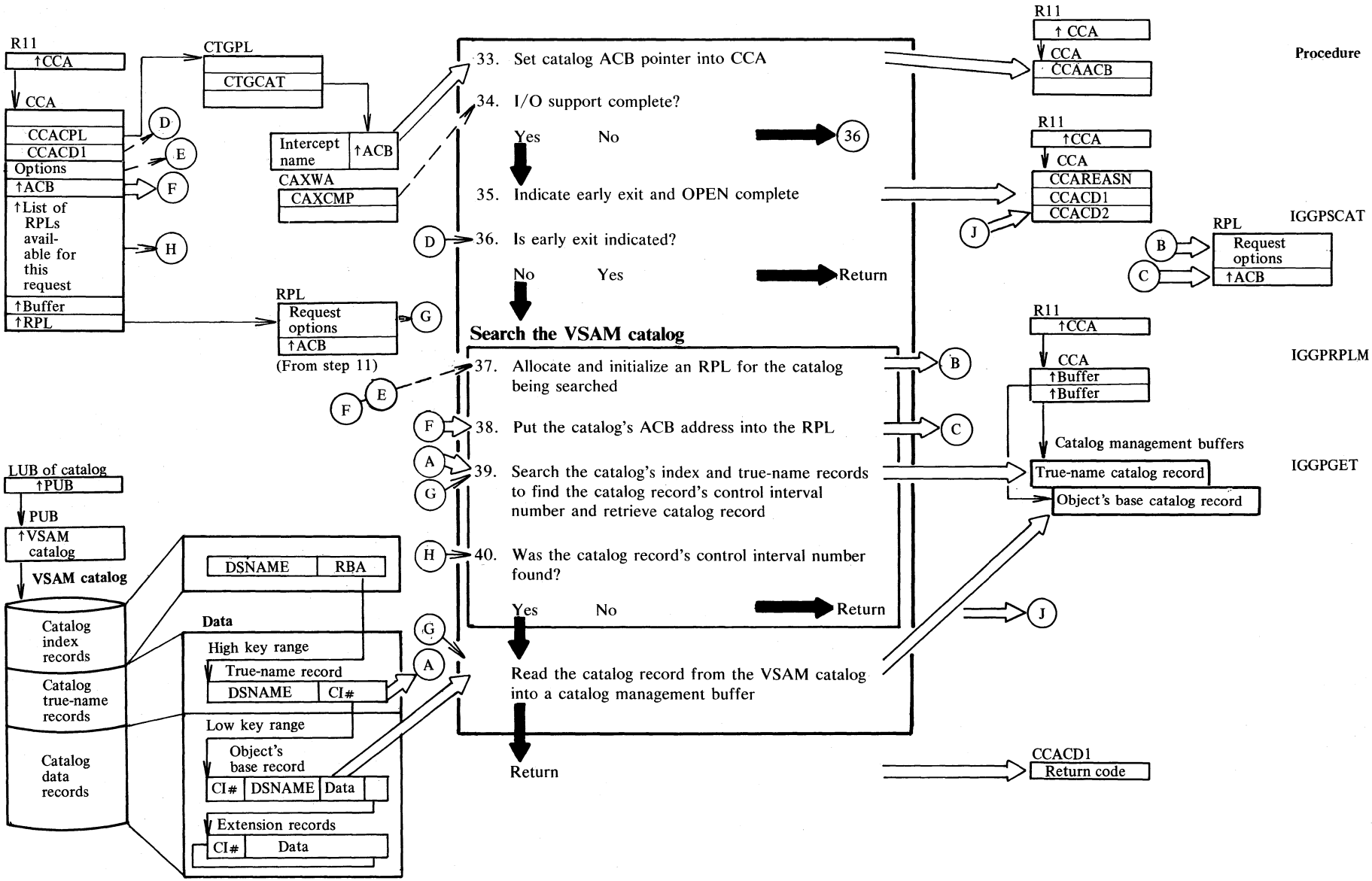


Diagram DC4. Search: Retrieve the base catalog record



Notes for Diagram DC (Part 1 of 2)

	Description	Module	Procedure
	The CTGPL's catalog identifier field, set by the caller, can contain the address of a catalog's ACB, the address of a catalog's DSNAME, or 0.		
	See 'Data Areas' for details about the CCA, ACB, and CTGPL.		
	The CMS DEFINE routine calls the search catalog routine to confirm that when a caller wants to create a VSAM cluster or catalog, the new cluster or catalog DSNAME is not duplicated in the catalog. The caller (CMS DEFINE) expects the 'no record found' return code.		
4	If the catalog record identifier is not a true name, it must be a control interval number. In this case, the catalog identifier (CTGCAT) must contain the address of the catalog's ACB.	IGG0CLAH	IGGPSCAT
8	The caller did not know the address of the catalog's ACB and IGG0CLAH must locate it.	IGG0CLAH	IGGPSCAT
10-12	If the AMCBS indicates the presence of IDCDF60 (an AMS parameter list) catalog management must use the CRA to retrieve data set information. This is the type of processing used by the AMS EXPORTRA and IMPORTRA functions.	IGG0CLAH	IGGPSCAT
13-16	If the CTGPL's catalog identifier field CTGCAT and/or the CTGPL's pointer to the user catalog's filename CTGDDUC contain 0, then the job catalog is searched if it exists, and if it does not, the master catalog is searched. Otherwise the specified user catalog is used. The AMCBS contains the address of the CAXWA chain.		
18	For DEFINE of user catalog the CAXWA chain cannot contain the requested CAXWA. Therefore no scanning is necessary.		
19-20	The CAXWA chain contains only CAXWAs referring to the ACBs of open catalogs or CRAs.		
22	If the CAXWA was not found, then the corresponding user-catalog must be opened, if an entry exists in the master catalog.	IGG0CLAB IGG0CLAG IGG0CLAB	IGGPRPLM IGGPGET IGGPRPLF
26	OPEN recursively calls the search catalog module. Procedure IGGPIOSI takes account of OPEN-build case.		
37	If CTGCAT contains a pointer to a 44-character name, it must be the data set name of the catalog as contained in the CAXWA.	IGG0CLAH	IGGPSCAT
	If the CTGPL's catalog record identifier field addresses the record's control interval number, the catalog record can be retrieved without a search of the catalog's index.	IGG0CLAG	IGGPGET
	The search catalog routine assigns an RPL to the caller. Catalog management (CM) routines issue GET and PUT macro instructions to retrieve and write catalog records. Each record management request (GET, PUT, etc.) needed to satisfy the caller's CM request refers to the RPL. The RPL is initialized for a caller and used as often as necessary to process the caller's CM request. When the caller's CM request is completed, the RPL is assigned to another caller.	IGG0CLAB	IGGPRPLM

Notes for Diagram DC (Part 2 of 2)

	Description	Module	Procedure
38	At this time, the CCA (CCAACB) contains the ACB address of the catalog or CRA.	IGG0CLAH	IGGPSCAT
39	The goal of the search is to find the true-name record identified by the DSNAME or the volume serial number. The true-name record contains the cluster's DSNAME or volume serial number and the control interval number of the cluster or volume catalog record. The search catalog routine sets the 'no record found' error code in the field CCACD2 of the CCA and returns to the caller. If the VSAM catalog has been unsuccessfully searched, the search catalog routine returns to the caller with the same error code set. See 'Diagnostic Aids' for CM error codes.	IGG0CLAG	IGGPGET
41	The catalog record is located by its control interval number and read into a CM buffer. The buffer's address is put into the CCA.	IGG0CLAG	IGGPGET

Diagram DD1. Check the password

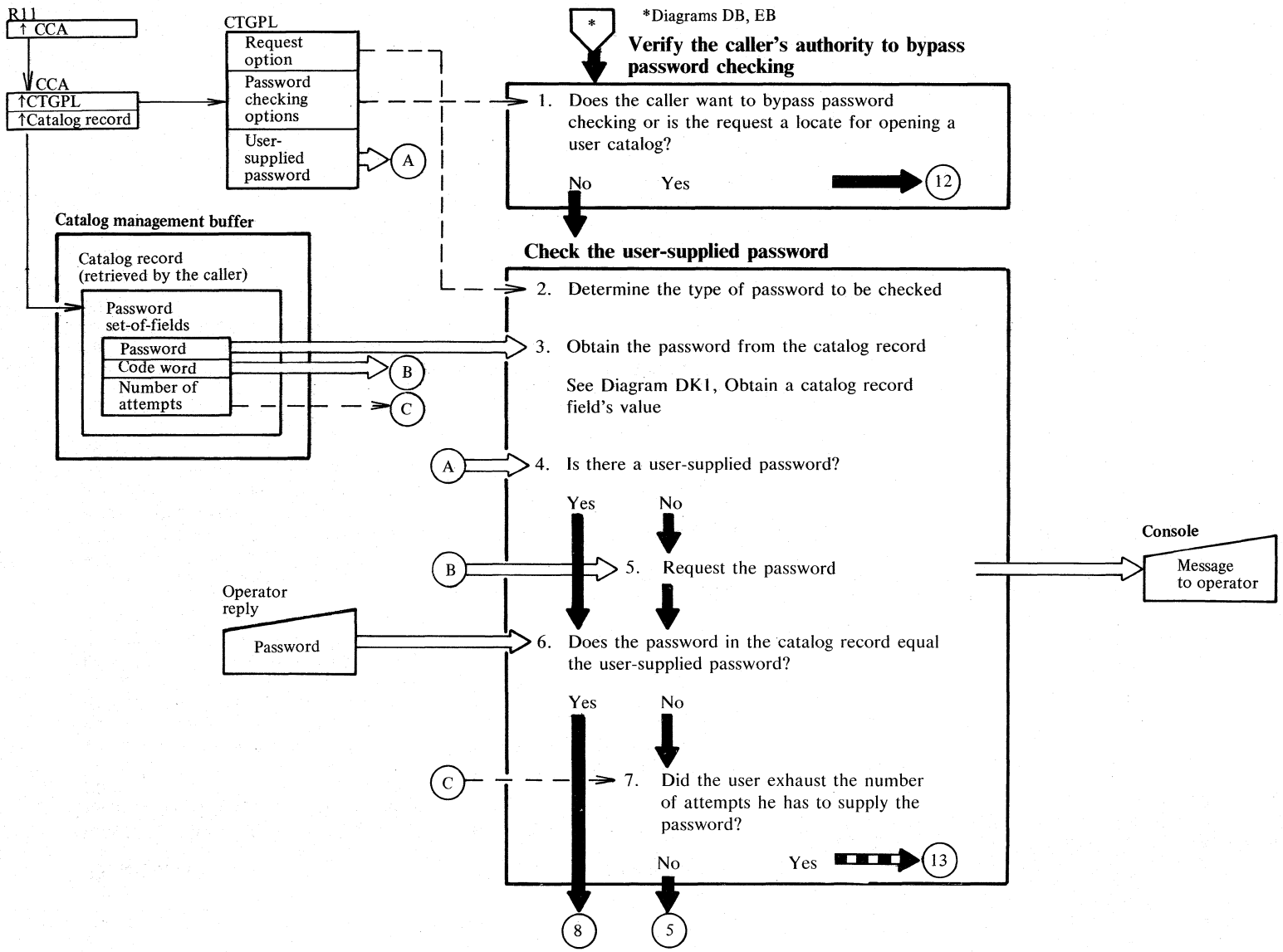
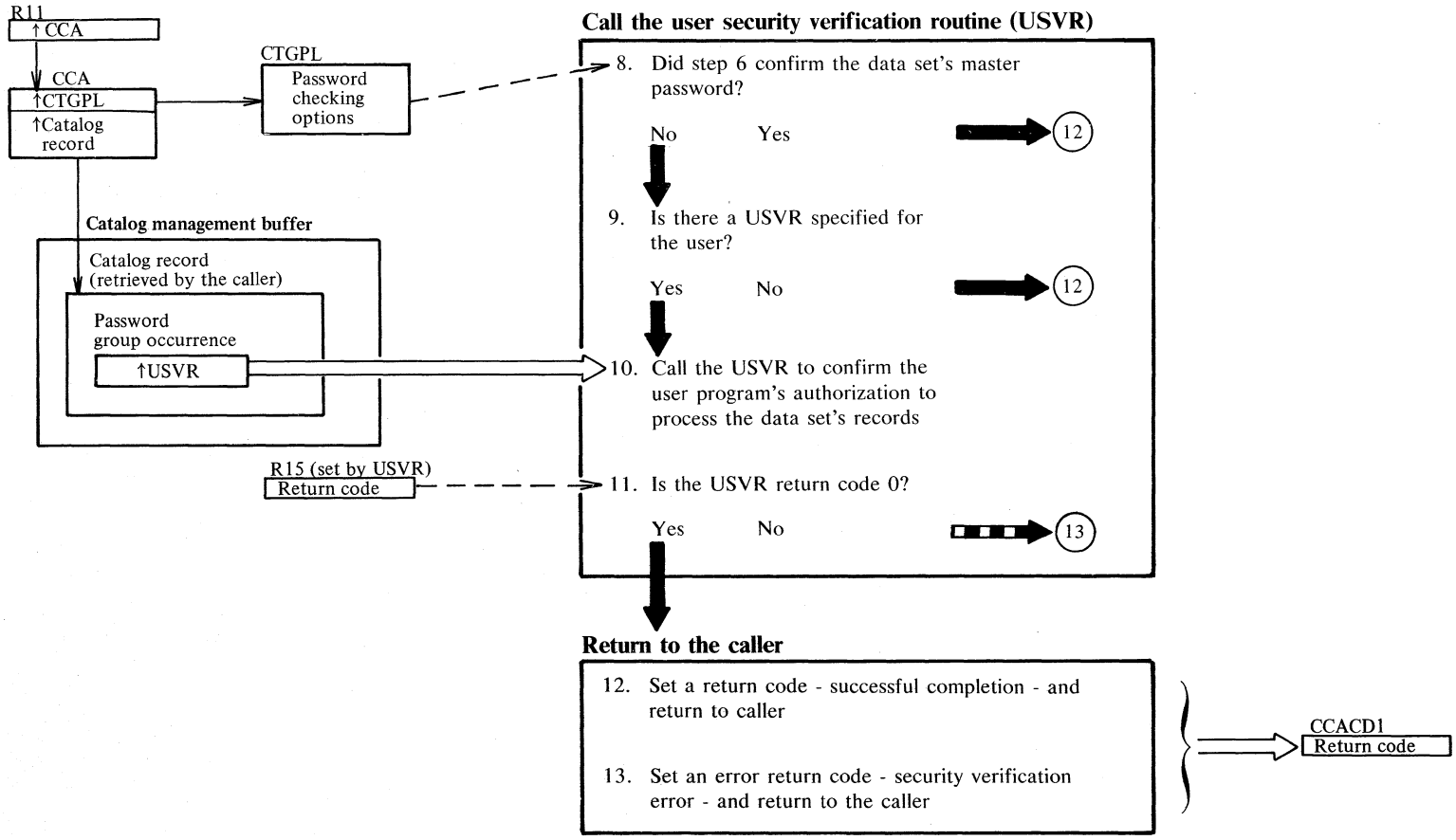


Diagram DD2. Check the password



Notes for Diagram DD

	Description	Module	Procedure
	When the VSAM Open routine (IKQOPN) calls VSAM catalog management to retrieve a cluster catalog record, the password checking routine (IGG0CLBM) confirms the user's authorization to gain access to the cluster.		
	When an Access Method Services routine calls a catalog management services routine (see Diagram EB1, step 1), the password checking routine confirms the user's password to gain access to the VSAM catalog or a specific catalog record.		
	The catalog record containing the password(s) is available in the buffer addressed in the caller's CCA.		
	The type of processing that the user is allowed to do with the data set is determined by the password:		
	Master password: The user is allowed to modify passwords and catalog records that describe his data set, and to process his data set's control intervals and records.		
	Control-interval password: The user is allowed to process the data set's control intervals as well as its records.		
	Update password: The user is allowed to process his data set's records.		
	Read-only password: The user is allowed to read, but not to write (add or update), records in his data set.		
1	If the user's password has been verified during a previous CM request, the caller (VSAM Open or CMS) can set the CTGPL's bypass-password-checking flag on.	IGG0CLBM	IGGPCKAU
2	The caller can indicate the minimum level of password to be verified with the CTGPL, but the password checking routine determines the type of password required for the request.	IGG0CLBM	IGGPSPSC
3	The password is in the password group occurrence in the catalog record.	IGG0CLAZ	IGGPEXT
5	The console operator can reply to the VSAM request for a password message with a password. If the operator replies with CANCEL or EOB, error code 56 is returned and module IGG0CLC9 cancels the job.	IGG0CLBM IKQDCN	IGGPPWGT
6-7		IGG0CLBM	IGGPPWVR
8	If the user supplied the correct master password, the user security verification routine (USVR), if it exists, is bypassed. If a USVR exists, the USVR exit is taken even though the user provided another type of password correctly.	IGG0CLBM USVR	IGGPINMD
9	If a user security verification routine exists for the user, its name is in the catalog record's password group occurrence. See Data Areas for details about the cluster catalog record and password group occurrence.	IGG0CLBM	IGGPINMD
10	The user security verification routine is an installation-supplied routine that confirms a user's authorization to gain access to the data set. The USVR confirms that the user satisfies the installation's security verification criteria.	IGG0CLBM	IGGPINMD

Diagram DE1. Locate: Retrieve catalog information

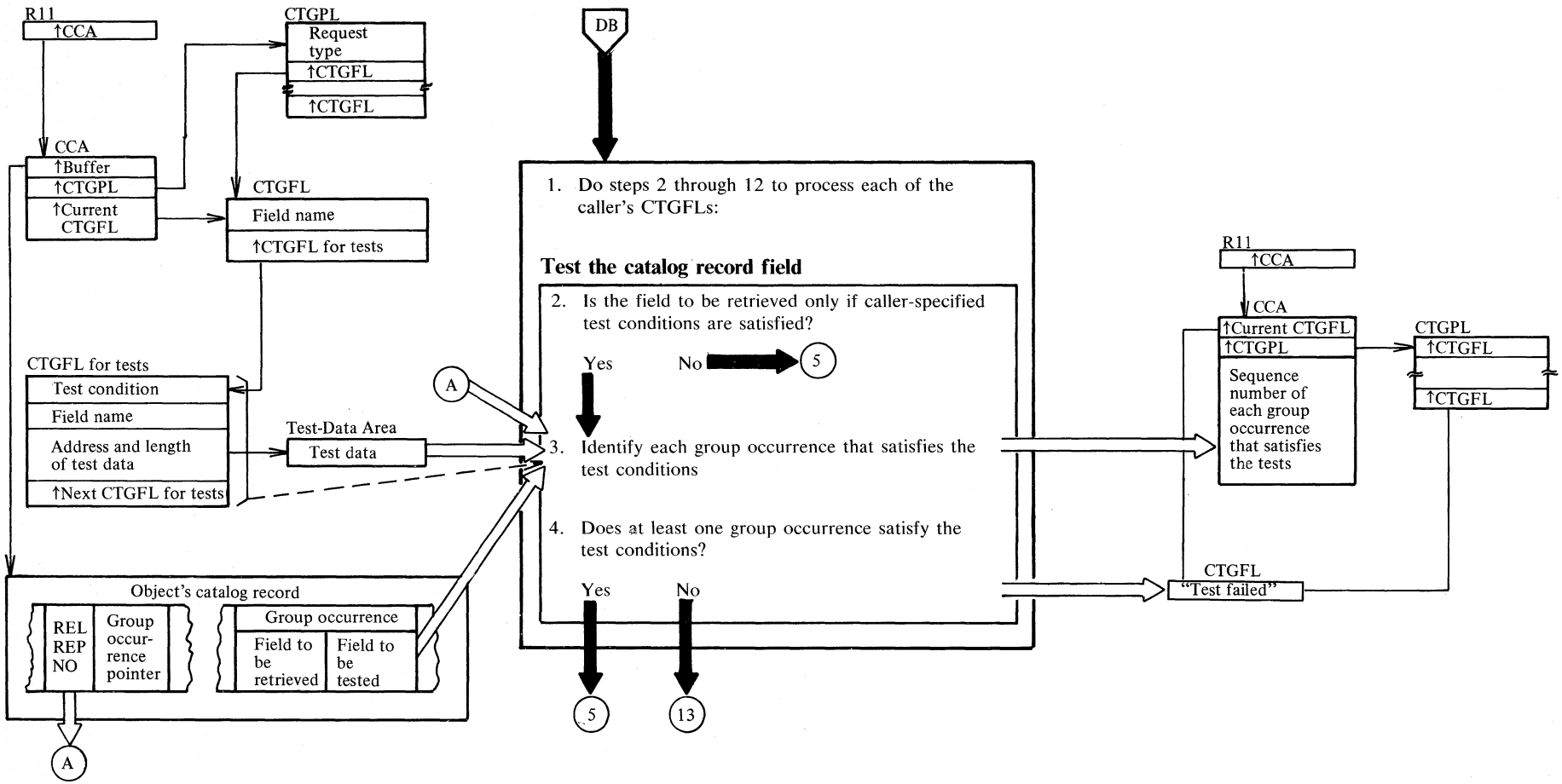
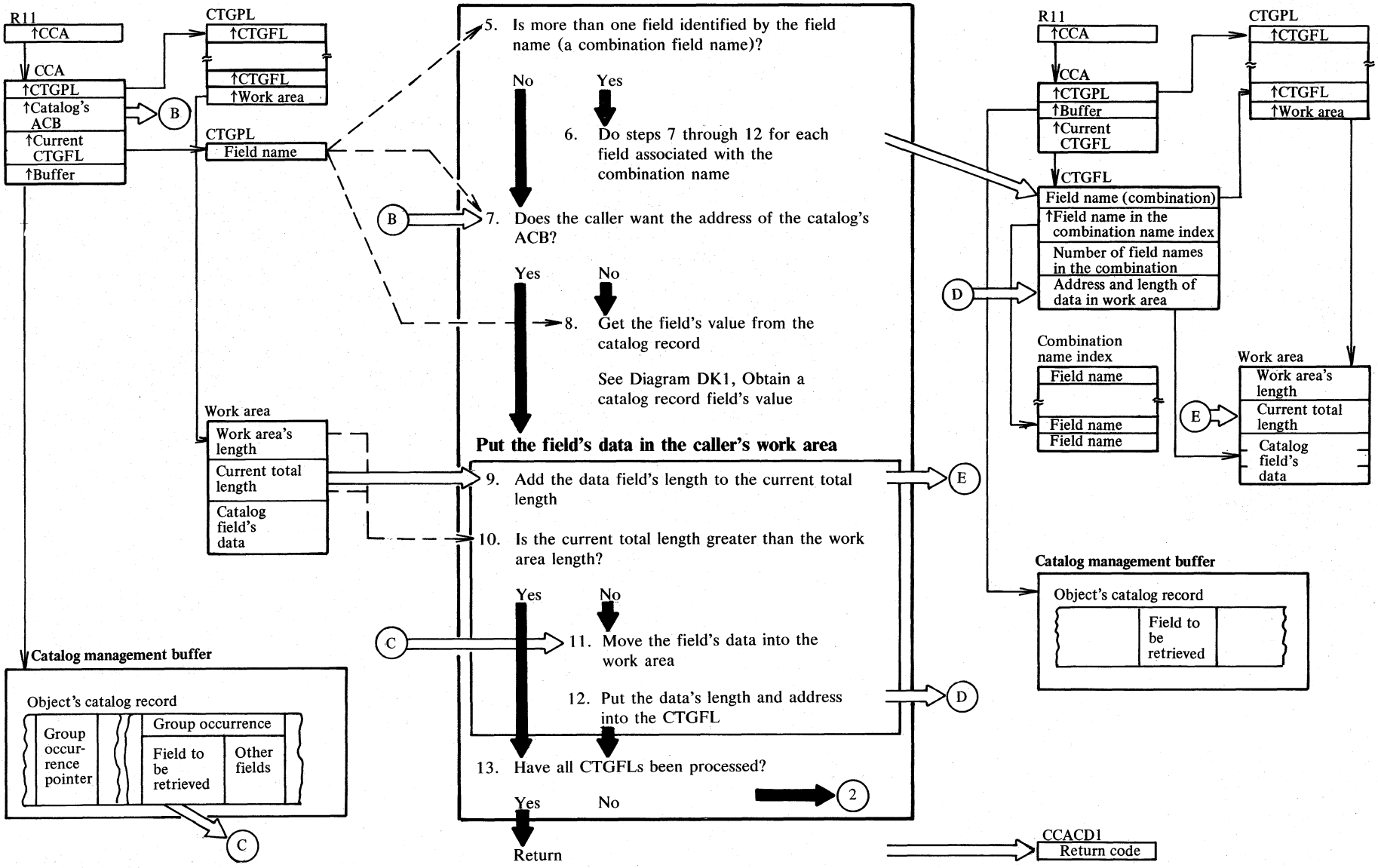


Diagram DE2. Locate: Retrieve catalog information



Notes for Diagram DE (part 1 of 2)

	Description	Module	Procedure
	The VSAM Open routine (IKQOPN) issues the CATLG instruction to obtain data set and volume information about the user's data set and index.		
	The VSAM end-of-volume routine (IKQEDX) issues the CATLG macro instruction to obtain volume information about the extents added to the user's data set.		
	When the caller issues a CATLG macro instruction, register 1 points to the caller's CTGPL. The CTGPL's request options are decoded and the base catalog record is retrieved for the request.		
1	The Locate routine processes each CTGFL associated with the caller's CTGPL and returns as much caller-requested data (in the caller's work area) as the caller's test conditions and work area size permit.	IGG0CLAZ	IGGPLOC
2	The caller's CTGFL list (CTGFIELD in the CTGPL) contains the address of each CTGFL required to satisfy the caller's need for catalog information. Each CTGFL describes one of the catalog record fields to be retrieved. Each CTGFL is completely processed before the next one is started. A caller might make conditional requests for retrieval of catalog record fields. For example, a chain of CTGFLs might be supplied with the request and processed together. The first CTGFL identifies a field to be retrieved and points to subsequent CTGFLs that contain the names of the catalog fields to be tested, the test conditions (equal, low, high, etc.) and the address and length of the caller's test data area. The catalog record fields identified by the second and subsequent CTGFLs are compared to (or tested against) the caller's data. If the comparison satisfies the test conditions, the catalog record field specified by the first CTGFL is retrieved.		
3	If the caller wants to retrieve a catalog record's header field, the field's data is retrieved if all tests are satisfied. If the caller wants to retrieve a field from one of the group occurrences that follow the header field, the field's data is retrieved from each group occurrence that satisfies all tests. The sequence number of each group occurrence that satisfies the tests is put in the CCA. When all group occurrences have been tested, the sequence numbers in the CCA are used to identify each group occurrence that contains caller-requested data.	IGG0CLBA	IGGPTSTS IGGPGREC
4	If none of the group occurrences satisfy the test conditions specified by test CTGFLs, the next CTGFL in the catalog parameter list (CTGPL) is processed.	IGG0CLAZ	IGGPSCNF
5	A combination name refers to a set of related catalog field names, and is used by the caller instead of a separate CTGFL for each field name.	IGG0CLAZ	IGGPLOC2
6	The combination name index has an entry for each field name in the combination. The Locate routine processes each field name entry in the combination name index sequentially, starting at the address of the first field name entry for the combination, and ending when the number of entries processed equals the number of field	IGG0CLAZ	IGGPLOC2

Notes for Diagram DE (part 2 of 2)

	Description	Module	Procedure
	names associated with the combination name. The test sequence, if any, associated with a combination-name CTGFL is done only once, not once for each field name in the combination.		
7	The address of the catalog's ACB is in the CCA. All other catalog record fields that the caller can request are in the catalog record. Each catalog record field is identified by its field name.	IGG0CLAZ	IGGPLOC2
8	Diagram DK1 shows how the requested catalog record field (specified by its field name in the CTGFL) is located for the Locate routine.	IGG0CLBA	IGGPGVAL IGGPGREC
9	The first two fields in the caller's work area specify the number of bytes the caller allocated to the work area and the number of bytes that contain catalog record field data (the current total length field).	IGG0CLAZ	IGGPLOC2 IGGPSHIN
10	If the current total length exceeds the work area length, the current total length field is updated with the length of the catalog record data, but the data itself is not moved in the caller's work area.	IGG0CLAZ	IGGPSHIN
11	The Locate routine puts the beginning address and the length of the catalog field into the CTGFL's field-data entry.	IGG0CLAZ	IGGPSHIN
12	The CTGFL's field-data entry contains the beginning address and length of the data in the caller's work area. When control is returned to the caller, the caller can use the field-data entry to locate a specific field's data in the work area.	IGG0CLAZ	IGGPSHIN

Diagram DG1. Update: Modify catalog information

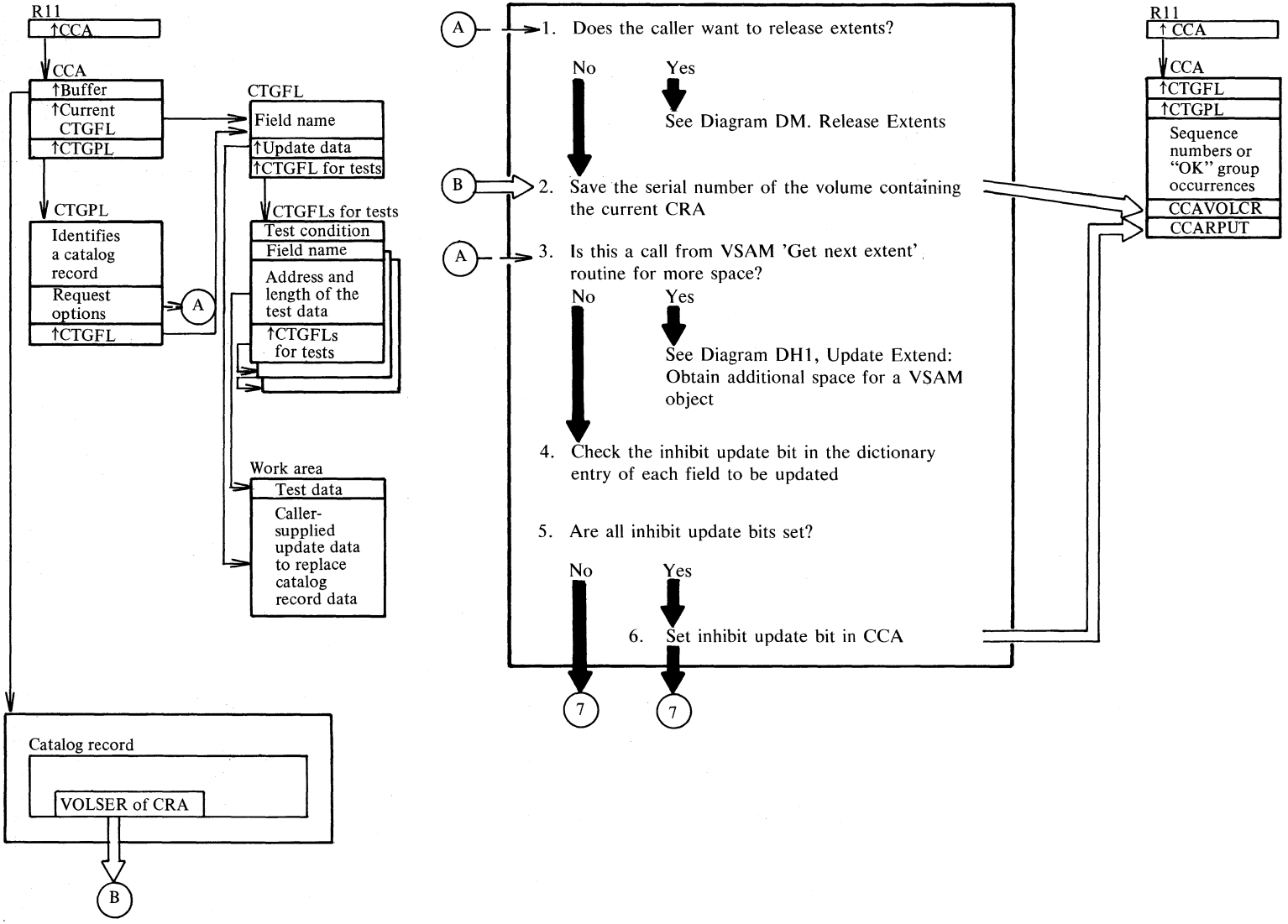
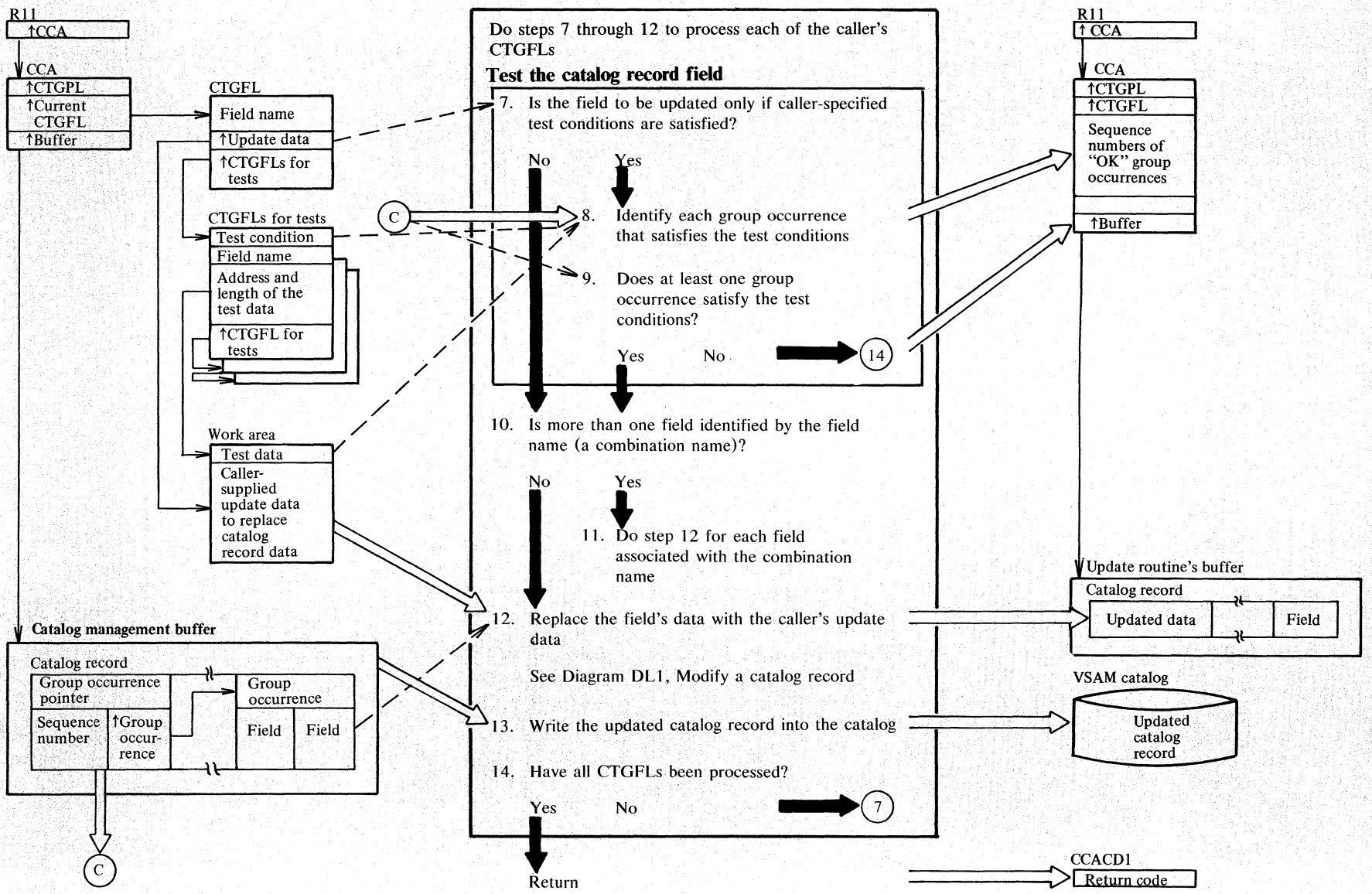


Diagram DG2. Update: Modify catalog information



Notes for Diagram DG (part 1 of 2)

	Description	Module	Procedure
	The VSAM Close routine (IKQCLO) issues the CATLG macro instruction to modify the data set and index statistics maintained in the catalog record's copy of the AMDSB.		
	The VSAM 'Get new extent'routine (IKQNEX) issues the CATLG macro instruction to obtain more space for a data set.		
	When the caller issues the CATLG macro instruction, register 1 points to the caller's CTGPL. The CTGPL request options are decoded and the base catalog record is retrieved for the request.		
1	If this is a call to release secondary extents (for a reusable data set), routine IGGPRELE in module IGG0CLCB is called.	IGG0CLAV	IGGPUPD
2	This information is saved in order to be able to supply it for extension records.	IGG0CLAV	IGGPSFPL
3	If more space is required for the data set, the UPDATE-Extend routine (IGG0CLBB) processes the caller's Update request and returns directly to the caller (the VSAM 'Get new extent' routine). See Diagram DH for information about UPDATE-Extend processing.	IGG0CLAV	IGGPUPD
4	Scan through the dictionary information of all fields to be updated.	IGG0CLAV	IGGPSFPL
5	If there are one or more fields in the list which require CRA update, the inhibit CRA update bit in the CCA (which was initially set 'on') is cleared.	IGG0CLAV	IGGPSFPL
	Steps 7 through 12 are performed to update each of the catalog record fields identified by the caller's CTGFL.		
7	The caller's CTGFL list (CTGFIELD in the CTGPL) contains the address of each CTGFL needed to satisfy the caller's updating requirements. Each CTGFL describes one of the catalog record fields to be updated. Each CTGFL is completely processed before the next one is started.	IGG0CLAV	IGGPSFPL
	The caller may want to update a field only if another field's value, when compared to the caller's test value, satisfies the caller's test conditions. If so, the caller builds a CTGFL that contains the name of the catalog field to be tested, the test conditions (equal, high, low, etc.), and the address and length of the caller's test value. If a CTGFL contains the address of another CTGFL, the second CTGFL describes a catalog record field that is to be compared to the caller's data. If the comparison satisfies the test conditions, the catalog record field specified by the first CTGFL is updated with the caller's data.	IGG0CLBA	IGGPTSTS
8	If the caller wants to update a catalog record's header field, the field's data is updated with the caller's data if all tests are satisfied.	IGG0CLBA	IGGPGVAL
	If the caller wants to update a field from one of the group occurrences that follow the header field, the field's data is updated with the caller's data for each group occurrence field that satisfies all tests. The group occurrence that contains the field to be updated can also be identified by its sequence number.		

Notes for Diagram DG (part 2 of 2)

	Description	Module	Procedure
	The sequence number of each group occurrence that satisfies the tests is put in the CCA. When all group occurrences have been tested, the sequence numbers are used to identify each group occurrence that contains caller-requested data.		
10	A combination name refers to a set of related catalog field names, and is used by the caller instead of a separate CTGFL for each field name.	IGG0CLAX	IGGPALT2
11	The CCA's combination name index has an entry for each field name in the combination. The Update routine processes each field name entry in the combination name index sequentially, starting at the address of the first field name entry for the combination, and ending when the number of entries processed equals the number of field names associated with the combination name. The combination name's CTGFL contains the beginning address and the total length of the group of update data fields in the caller's work area. The test sequence, if any, associated with a combination-name CTGFL is done only once, not once for each field name in the combination.	IGG0CLAX	IGGPALT2
13	When the catalog record is updated (in a buffer in the Update routine's virtual storage), the Update routine sets the 'must write' flag on to indicate that the I/O manager must write the buffer from virtual storage into the catalog before the buffer can be made available to contain another catalog record. When the caller's Update routine needs the buffer to process another catalog record associated with the request, the Update routine writes the catalog record from the buffer into the VSAM catalog (on a direct-access storage device). Note: If the catalog is recoverable, the call to IGG0CLAG will result in a CRA update unless bit CCARPUT (the update inhibit bit) is set to '1'. This bit is set in the calling procedures, which decide whether CRA updating is necessary for a given operation.	IGG0CLAW IGG0CLAG	IGGPPREC IGGPPUPC IGGPPAD

Diagram DH1. Update-extend: Obtain additional space for a VSAM object

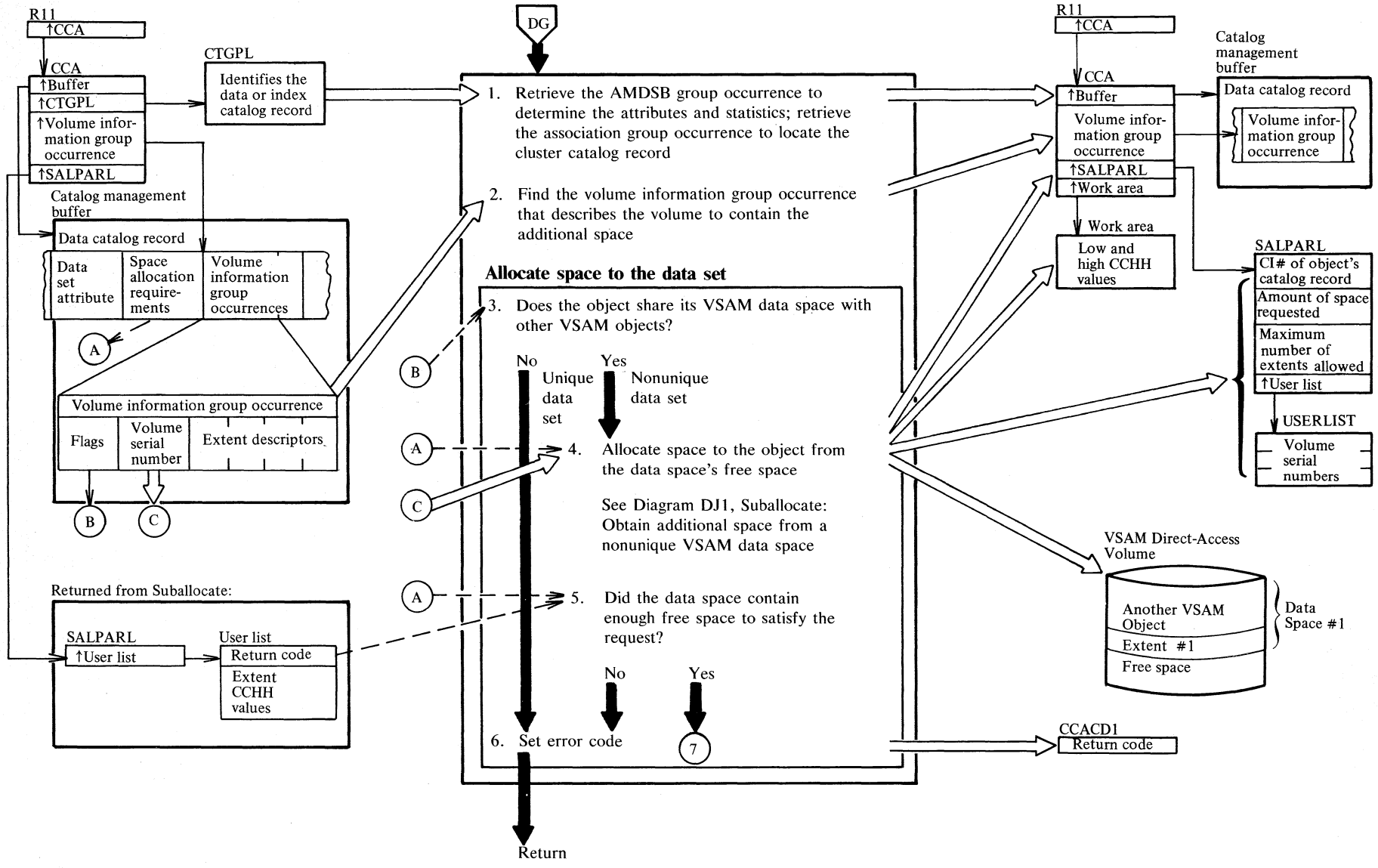


Diagram DH2. Update-extend: Obtain additional space for a VSAM object

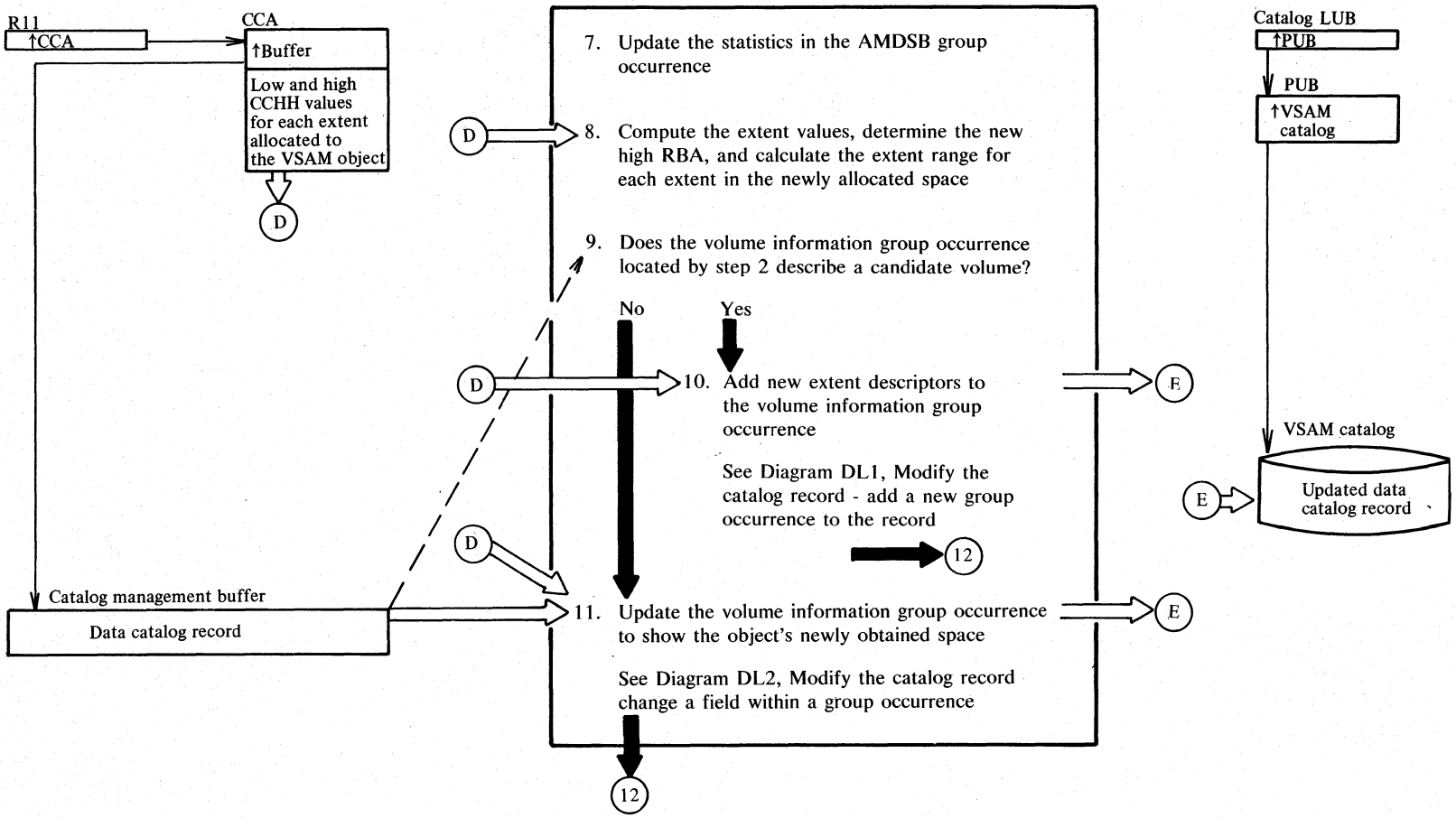
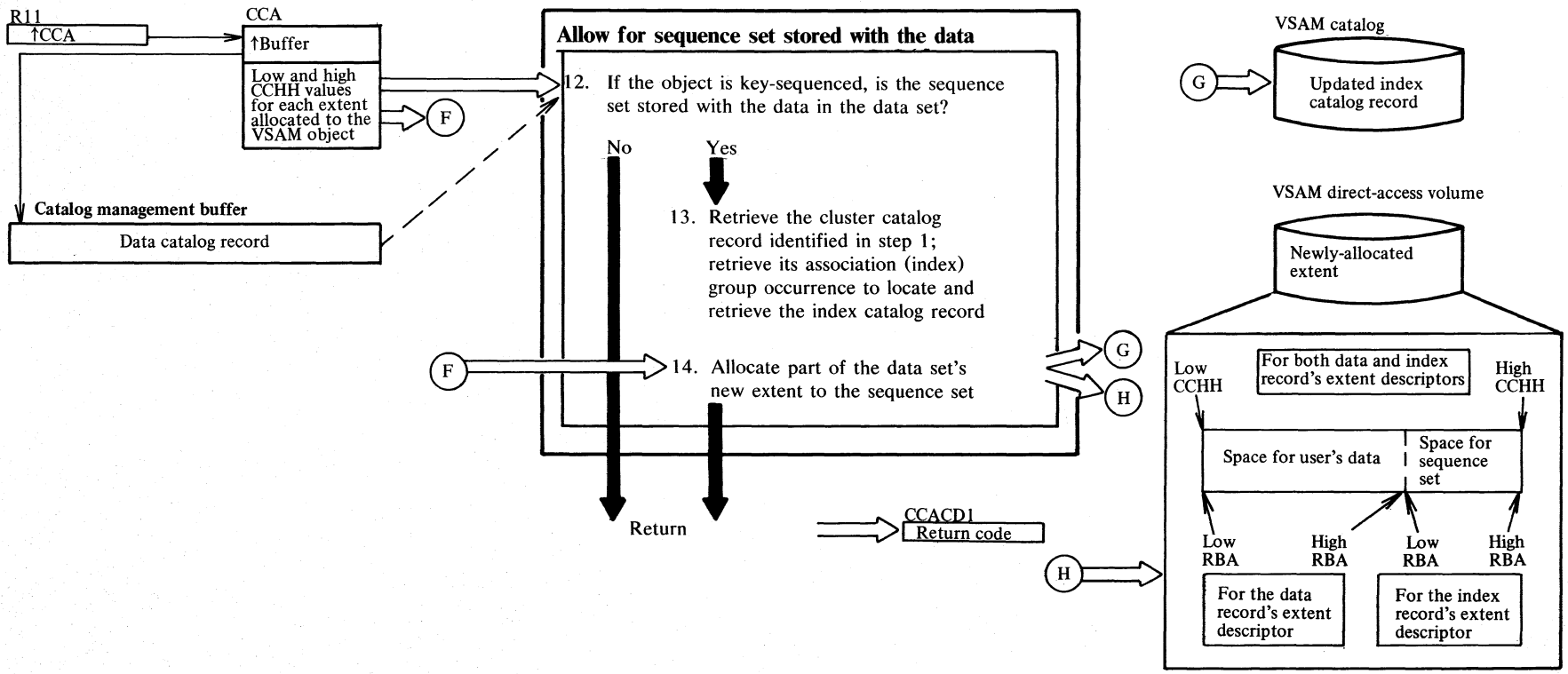


Diagram DH3. Update-extend: Obtain additional space for a VSAM object



Notes for Diagram DH

	Description	Module	Procedure
	The UPDATE-Extend routine is called whenever a VSAM object (data set, index, or catalog) needs more space to store its records.		
	The VSAM 'Get new extent' routine calls the CM Update routine and an amount of space, based on the object's direct-access space allocation requirements, is allocated from a shared VSAM data space that has enough free space to satisfy the allocation requirements.	IGG0CLBB IGG0CLBC IGG0CLAZ IGG0CLBB IGG0CLBC	IGGPUPDE IGGPINIT IGGPEXT IGGPUPDE IGGPSVOL
2	The volume information group occurrence is identified by either an RBA or a key value.		
3	A shared (nonunique) VSAM data space contains all or parts of two or more VSAM objects. A unique VSAM data space contains all or part of only one VSAM object, and is not allowed to contain records of another object.	IGG0CLBB IGG0CLAG	IGGPUPDE IGGPUALL IGGPGET
4	If the object shares its data space with other VSAM data sets or indexes, there might be enough free space in one of the data spaces on the volume to satisfy the object's direct-access space allocation requirements.	IGG0CLBB IGG0CLAR	IGGPUPDE IGGPCSAL IGGPSALL
7	The object's catalog record contains a volume information group occurrence to describe the object's space on each volume that contains a part of the object. If the object's newly obtained extent is on a new volume, the UPDATE-Extend routine builds a volume information group occurrence to describe the new volume and extent. Otherwise, an existing volume information group occurrence is updated with the high-allocated RBA and extent information in the form SS CCHH CCHH DDDD DDDD where: SS identifies the VSAM data space CCHH are the low and high cylinder and track addresses DDDD are the low and high RBAs	IGG0CLBB IGG0CLAV	IGGPMVOL IGGPMOD
14	The low and high cylinder and track addresses in the index catalog record's volume information group occurrence are those of the extent obtained for the data set. The low and high RBA values are for the sequence set.	IGG0CLBB IGG0CLAG IGG0CLBB IGG0CLBC IGG0CLBB IGG0CLBC	IGGPSSWD IGGPGET IGGPSSWD IGGPINIT IGGPSSWD IGGPSVOL

Diagram DJ1. Suballocate: Obtain additional space from a nonunique VSAM data space

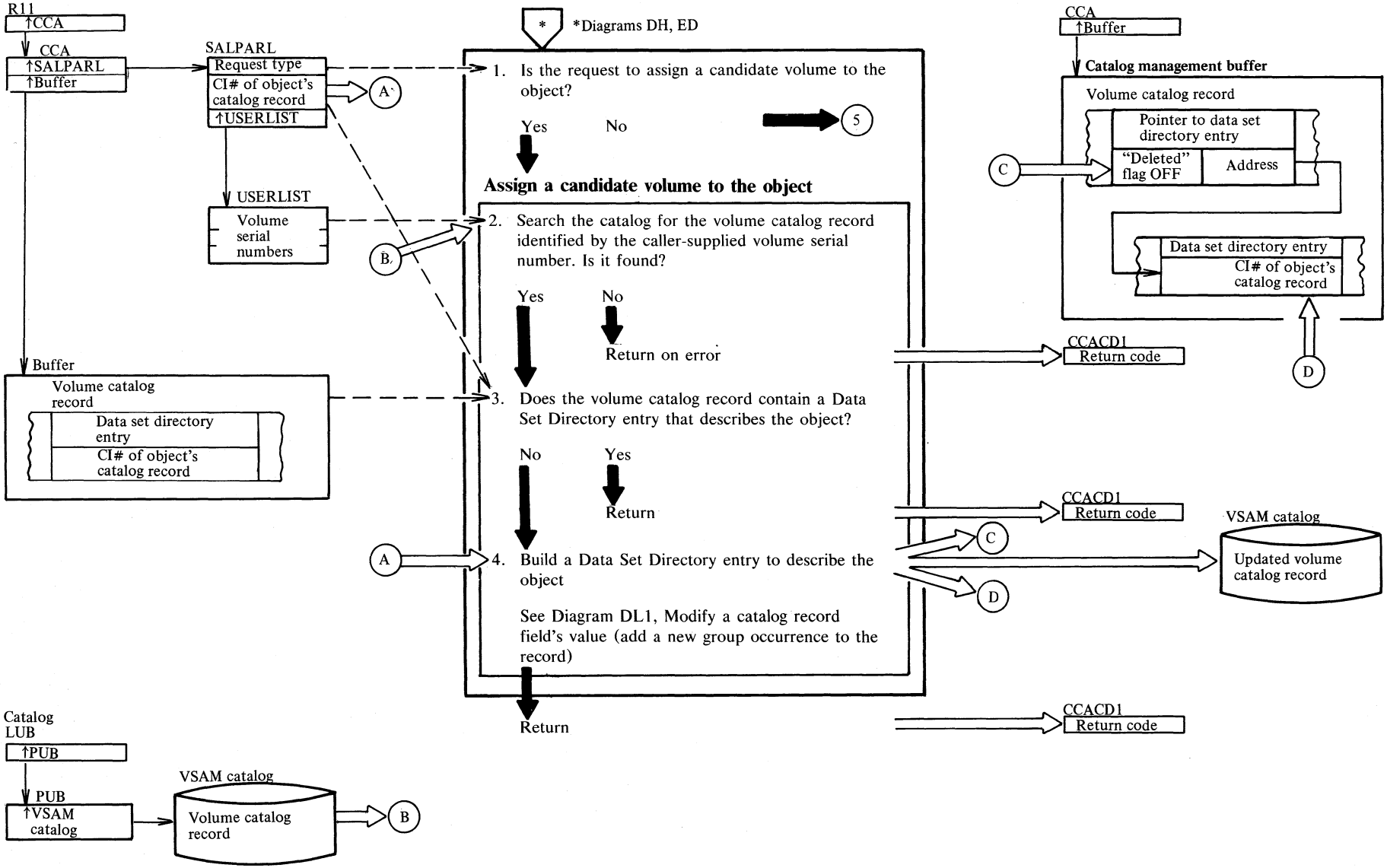
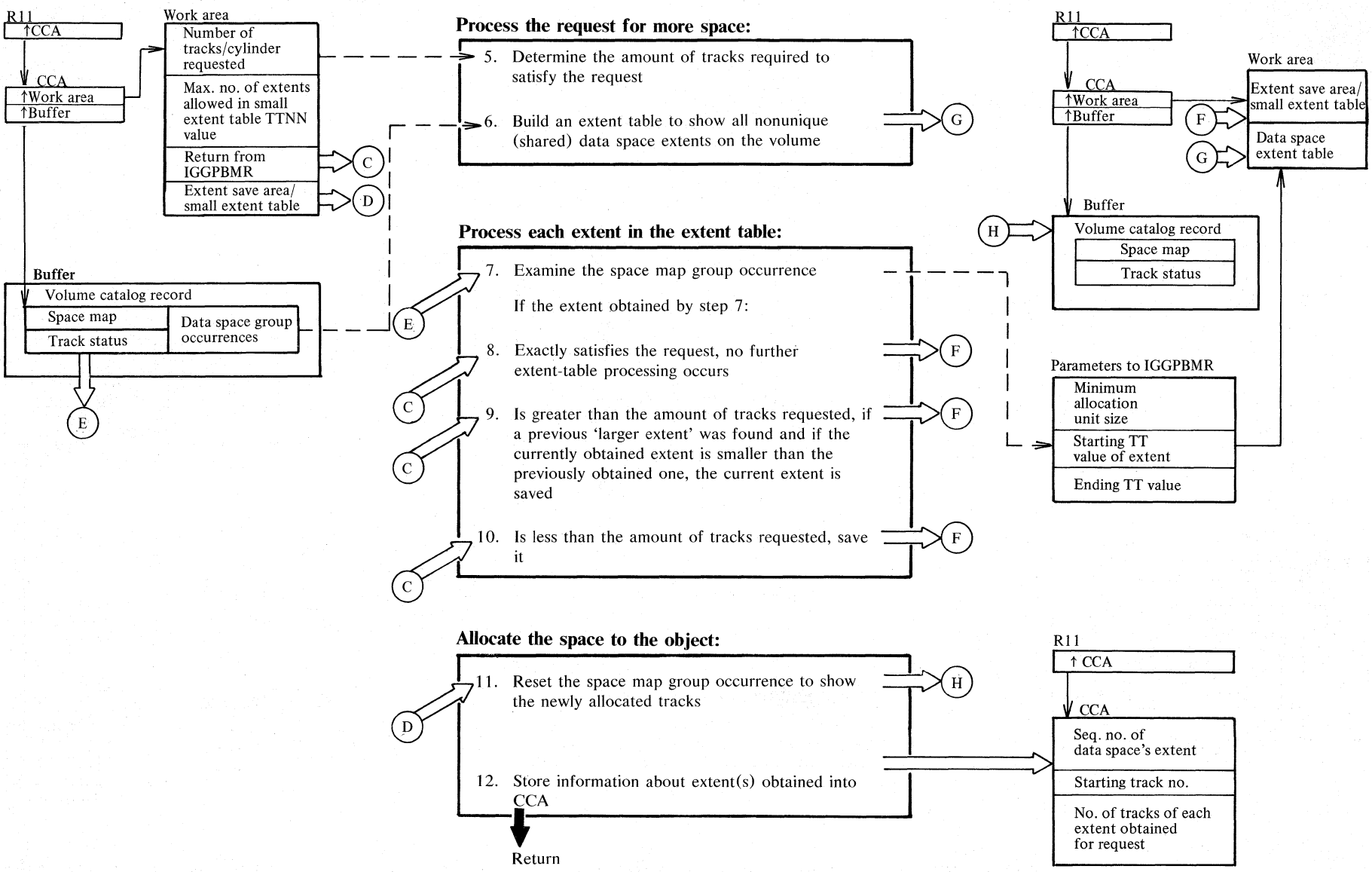


Diagram DJ2. Suballocate: Obtain additional space from a nonunique VSAM data space



Notes for Diagram DJ (part 1 of 2)

	Description	Module	Procedure
	The suballocate routine, IGG0CLAR, is called to assign a candidate volume to a VSAM object (data set, index, or catalog) and to assign available spaces on the caller-specified volume. The caller, either the UPDATE-Extend routine (Diagram DH) or the DEFINE Cluster routine (Diagram ED) builds a list of volume serial numbers to identify each volume to be assigned to the object as a candidate volume. If the caller requests space allocated to the object, the list contains one volume serial number.		
1	A candidate volume is available to contain a VSAM object's catalog records, but no space is allocated to the data space from the volume as yet.	IGG0CLAR	IGGPSALL
2	The volume must be owned by the same catalog that contains the object.	IGG0CLAR IGG0CLAG	IGGPSALL IGGPGET
3	The volume catalog record already contains a data set directory entry, the volume is either already assigned to the VSAM object as a candidate volume or has some of its space allocated to the VSAM object.	IGG0CLAR IGG0CLAZ	IGGPSALL IGGPEXT
4	The new data set directory is added to the volume catalog record.	IGG0CLAR IGG0CLAG IGG0CLAR IGG0CLAV	IGGPSALL IGGPISCI IGGPSALL IGGPMOD
5	If the amount of space requested is a number of cylinders, convert it to a number of tracks.	IGG0CLAR IGG0CLAU	IGGPSALL IGGPSALS
6	The extent table is built by retrieving each extent descriptor (from each data space occurrence) that might contain enough free space to satisfy the request's minimum allocation requirement (the number of tracks in one control area).	IGG0CLAZ	IGGPEXT
	All extents of shared data spaces are described in the table until there are no more extents to describe or the table is full. If the table is full, step 6 is repeated when steps 7-10 are completed, until the extents of all shared data spaces have been examined.		
7	Each extent descriptor in the extent table is processed beginning with the lowest extent starting track number (TT) in the table, in ascending sequence, until all extent descriptors have been processed. IGGPBMR examines each extent to find an amount of contiguous unallocated tracks at least as large as the request's minimum allocation unit. IGGPBMR examines the space map group occurrence, starting at bit position TT (track indicator) and ending at bit position TT plus the number of tracks in the extent (NN) minus 1. If IGGPBMR finds a large enough amount of unallocated tracks, it returns to IGGPEDS with the beginning track number (TT) and the number of tracks (NN). If the data space's extent contains another amount of unallocated tracks at least as large as the request's minimum allocation unit, IGGPEDS calls IGGPBMR to examine the rest of the data space's extent.	IGG0CLAU IGG0CLBR	IGGPSALS IGGPEDS IGGPBMR
8	If the extent returned by IGGPBMR is the exact number of tracks required to satisfy the caller's request, no further extent table processing is done. Larger or smaller extents obtained from previous extent table entries are ignored.	IGG0CLAU	IGGPEDS

Notes for Diagram DJ (part 2 of 2)

	Description	Module	Procedure
9	If the extent returned by IGGPBMR is larger than the amount of tracks required to satisfy the request, the extent is saved if either: • No other larger-than-requested-amount extent has been returned yet, or • The current extent is smaller than a previously obtained larger-then-requested-amount extent. In either case, only one larger-than-requested-amount extent value is saved. The small extent table (built in step 10) is ignored and no longer used.	IGG0CLAU	IGGPEDS
10	If the extent returned by IGGPBMR is smaller than the amount of tracks required to satisfy the request, its TTNN value is adjusted so that TT is on a cylinder boundary. If NN is now at least as large as the request's minimum allocation unit (number of tracks for one control area), the extent is saved in the small extent table if: . The table has fewer than five entries (or a caller-specified maximum less than five) in it, or . The table is full and the current extent's NN value is greater than the table's smallest extent. The current extent replaces the table's smallest extent. In either case, the extent is not put in the small extent table if it is too small (adjusted NN is less than the minimum allocation unit) or if a larger-than-requested-amount extent already exists (see step 9).	IGG0CLAU	IGGPEDS

	Description	Module	
	If, after all data spaces have been examined, the total of the NN values in the small extent table is less than the amount required to satisfy the request, no space is allocated to the object.		
11	If the selected extent is larger than or equal to the amount of space requested, IGGPBMR adjusts the space map group occurrence starting at bit position TT (track indicator) by turning off NN bits (NN is the exact number of tracks required to satisfy the request). If the space is allocated to an object from a number of extents, the small extent table is sorted so that the largest NN value is first, the smallest last. IGGPBMR then adjusts the space map group occurrence for each TTNN value in the small extent table, until the amount of allocated tracks equals the amount of tracks requested.	IGG0CLAU IGG0CLBR	IGGPSALS IGGPBMR
12	IGGPSALS returns the sequence number of the data space's extent, starting track number, and number of tracks of each extent obtained for the request. The caller uses this information to build extent descriptor entries in the VSAM object's volume information group occurrence.	IGG0CLAU	IGGPSALS

Diagram DK1. Obtain a catalog record field's value

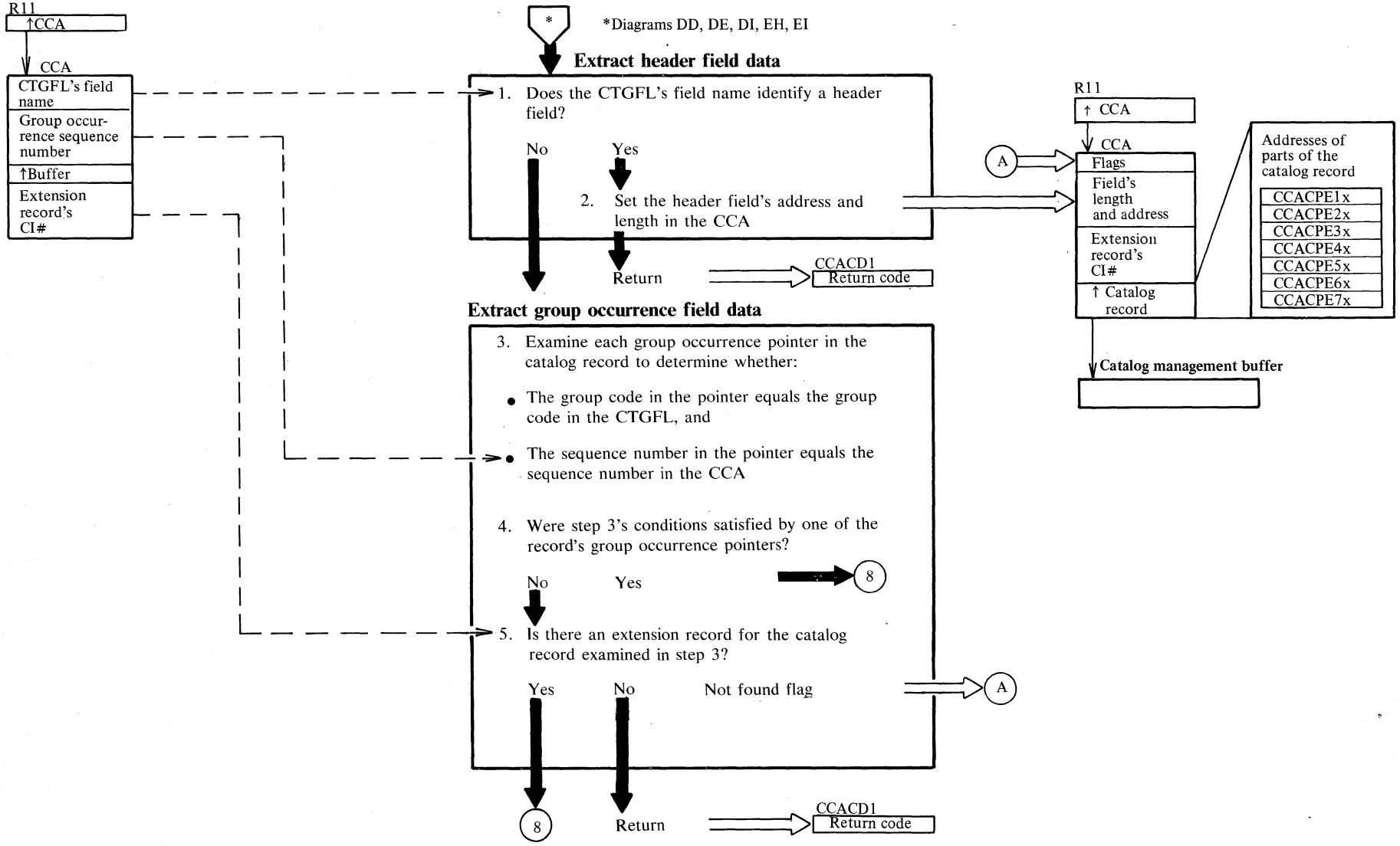


Diagram DK2. Obtain a catalog record field's value

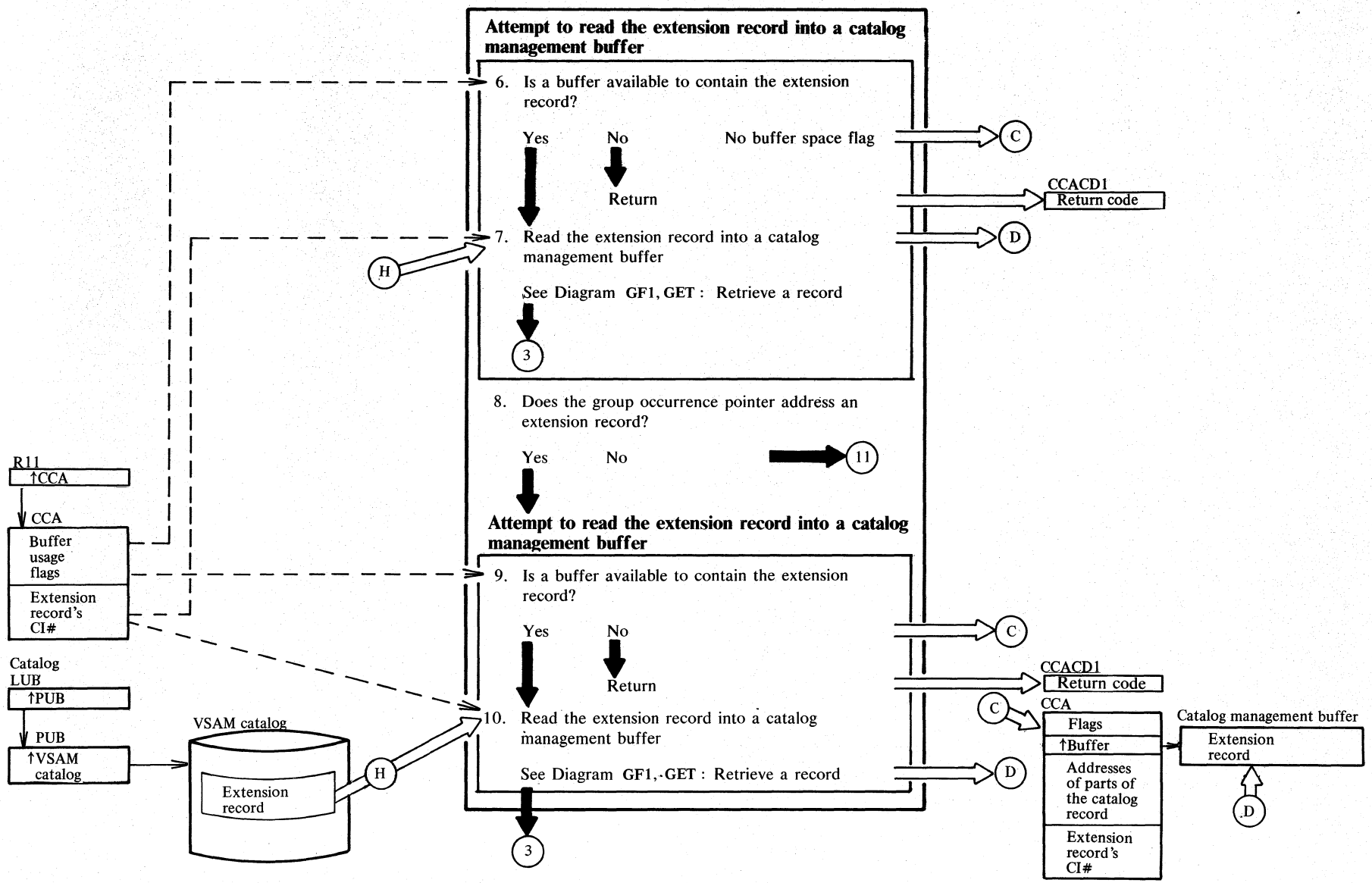
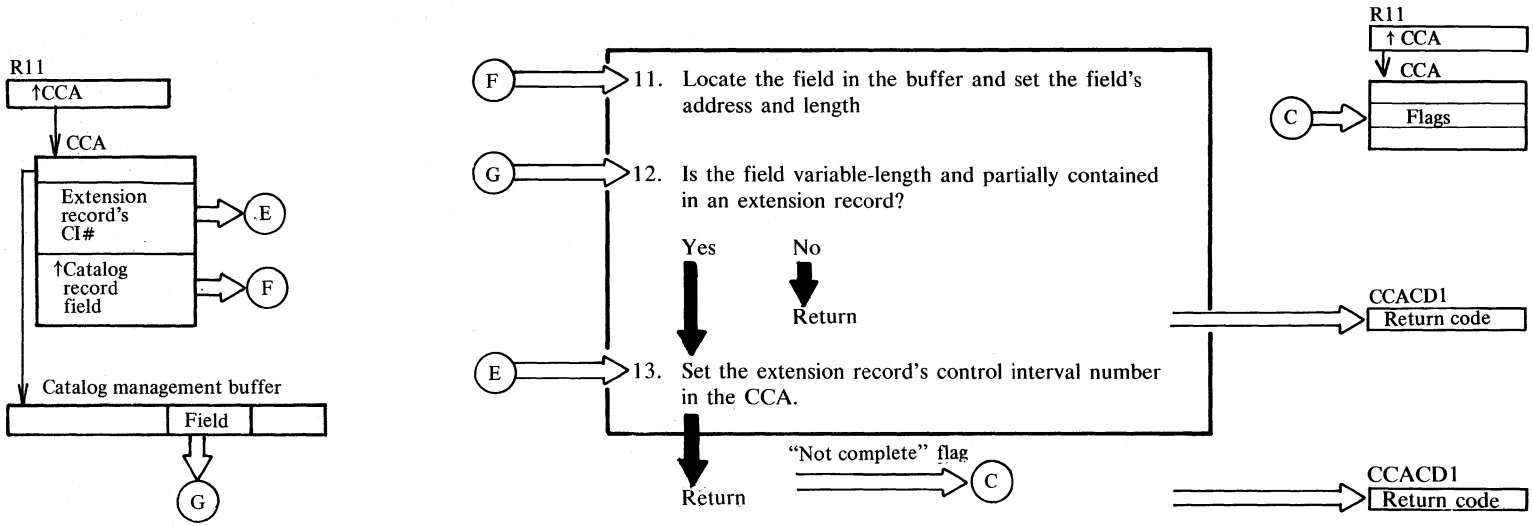


Diagram DK3. Obtain a catalog record field's value



Notes for Diagram DK (part 1 of 2)

	Description	Module	Procedure
	The obtain-field-value routine is called by other catalog management (CM) routines to obtain the location and length of a field in a catalog record. The record is in a CM buffer in virtual storage. The following results could occur: • The field is entirely contained in the record in the buffer, and the field's address and length are set in the caller's CCA. • The field is partially contained in the record in the buffer, and the field's address and partial length are set in the caller's CCA. The CCA also has the not complete flag on and contains the control-interval number of the catalog record's extension, which contains more of the field. • The field is not retrieved because it doesn't exist in the caller-specified group occurrence, or because there are no more group occurrences in the record, or because no buffer space is available to contain extension record.		
1	The derived volume entry fields are retrieved.	IGG0CLBS	IGGPXVAL
	The field-name dictionary is a read-only catalog management table. The catalog field name dictionary contains an entry for each type of catalog record field, based on its field name. The caller puts the dictionary entry identified by the CTGFL's field name into the CCA before calling the obtain-field-value routine.	IGG0CLBA	IGGPGVAL
	If the field name identifies a header field, the field's type code (in its dictionary entry) is 0. A nonzero group code identifies the group occurrence that contains the field identified by the CTGFL's field name. If the field name identifies a header field and the field is fixed-length, it is at a fixed displacement from the beginning of the catalog record. The field's address is obtained by adding the displacement in the CTGFL's dictionary entry to the beginnning address of the record (the contents of CCACPE1x). The field's length is part of the CTGFL's dictionary entry.		
3	The group occurrence pointer (GOP) is used to locate a group occurrence. The GOPs are grouped together by a group code. Within each group of GOPs, the pointers are ordered by sequence number.	IGG0CLBA	IGGPLVAL
4	If the caller-specified GOP, identified by its sequence number, is found, its displacement field and flag field specify the location of its group occurrence as: • the number of bytes from the beginning of the record's group occurrences (the contents of CCACPE3x plus the GOP's displacement field value), or • the control-interval number of the extension record that contains the group occurrence. The extension record contains a GOP that specifies the group occurrence's location as a number of bytes from the beginning of the record's group occurrences.	IGG0CLBA	IGGPLVAL

Notes for Diagram DK (part 2 of 2)

	Description	Module	Procedure
5	The actual group occurrences containing the field to be retrieved may be contained in an extension record.	IGG0CLBA	IGGPLVAL
6	Each catalog record in the CM buffer is identified by a record area block (RAB) within the CCA. The RAB contains flags that indicate whether or not the buffer can be used to contain another record. If the RAB's 'must write' flag is on, the buffer cannot be used for another record until its contents have been written into the catalog.	IGG0CLBA IGG0CLAW IGG0CLBA IGG0CLAG	IGGPGREC IGGPPREC IGGPGREC IGGPGET
	Each CCA contains six record area blocks (RABs). Each CM request can use a maximum of five buffers. If all buffers are filled and cannot be released, the obtain-field-value routine sets the CCA's no buffer space flag on.	IGG0CLBA	IGGPLVAL
7	The CCA's not found flag is set off before returning to step 3 to examine the extension record's GOPs.	IGG0CLBA IGG0CLAG	IGGPGREC IGGPGET
8	If the GOP contains the control-interval number of an extension record, the group occurrence is contained on that extension record.	IGG0CLBA	IGGPGVAL
9	See step 6.		
10	See step 7.		
11	The field's length is obtained from the CTGFL's dictionary entry (for a fixed-length field) or the first two bytes of the field (which are the length bytes of a variable length field). The field's address is the sum of the address of the group occurrence and the number of bytes from the beginning of the group occurrence.	IGG0CLBA	IGGPLVAL
12	A variable-length field might be partially contained in an extension record. If so, the field's length is greater than the number of bytes remaining in the record.	IGG0CLBA	IGGPGVAL
13	The caller's information requirements might be satisfied with the part of the field that is currently available. If not, the caller (a CM routine) returns to IGG0CLBA to obtain the value of the next part of the field from the extension record.	IGG0CLBA	IGGPLVAL
	The caller can move that part of the field currently in the buffer into a work area. If the rest of the field is required, the caller can return to IGG0CLBA to retrieve the extension record.		

Diagram DL1. Modify a catalog record field's value

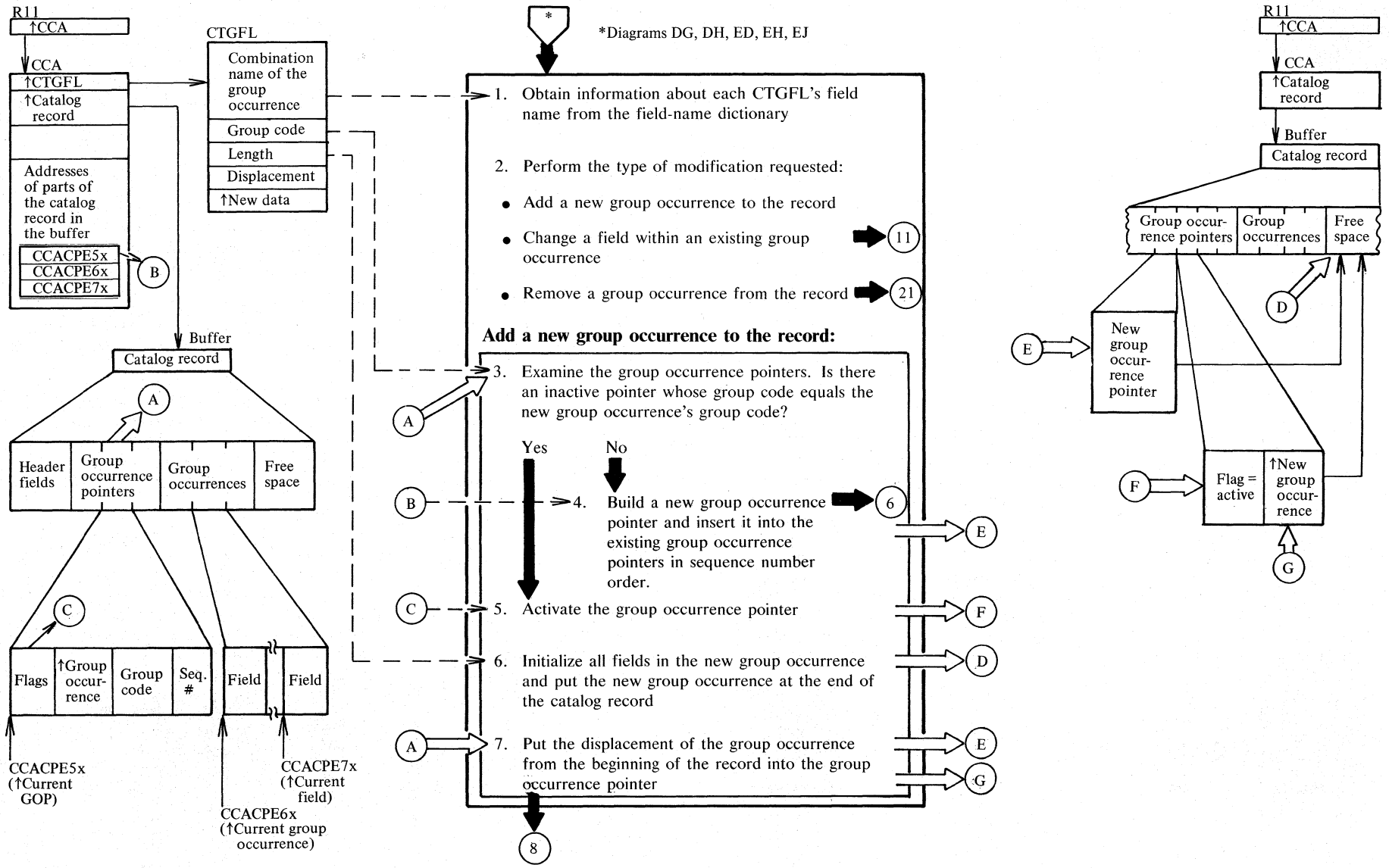


Diagram DL2. Modify a catalog record field's value

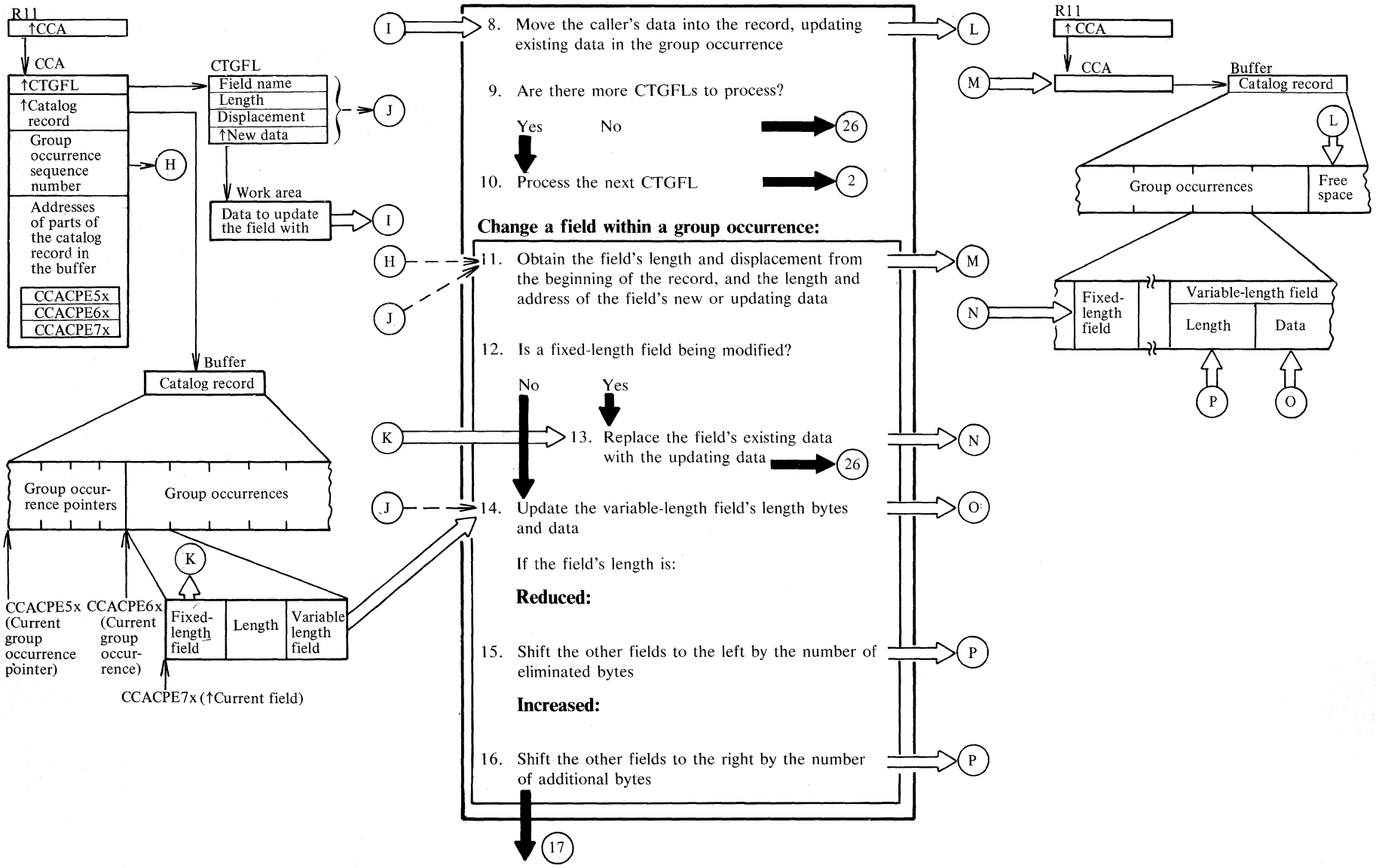


Diagram DL3. Modify a catalog record field's value

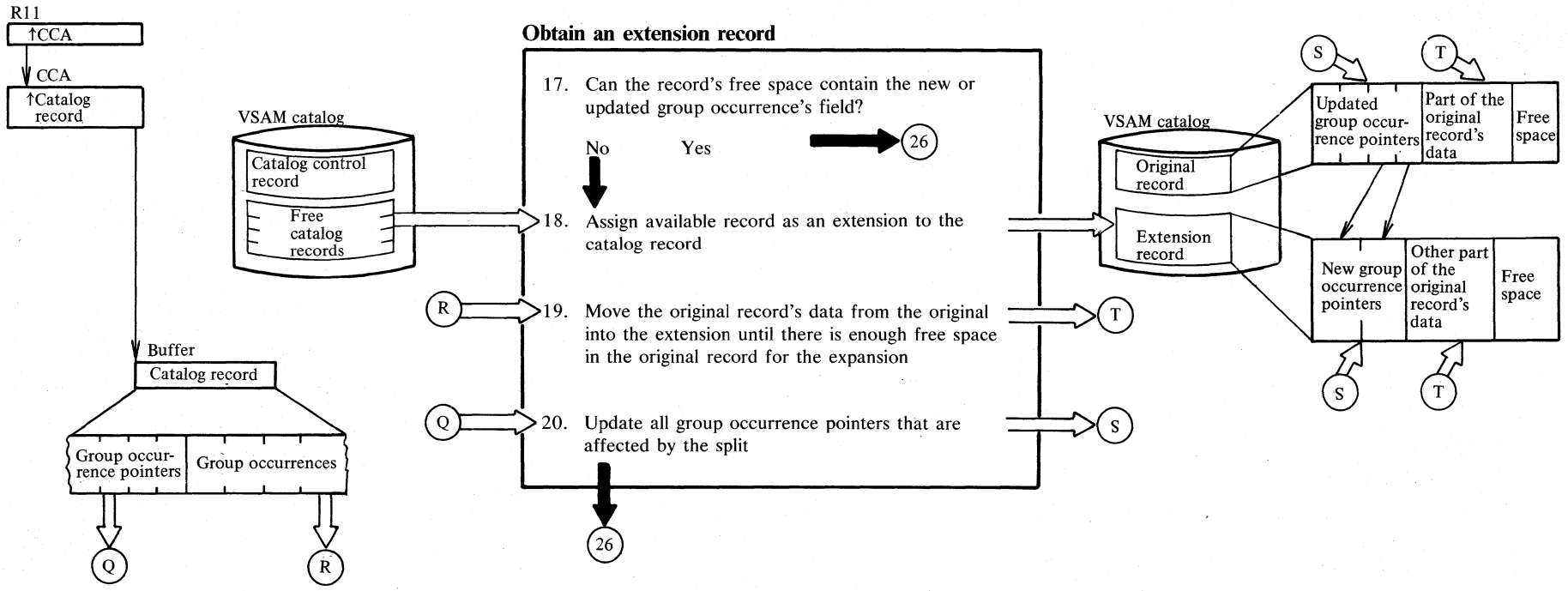
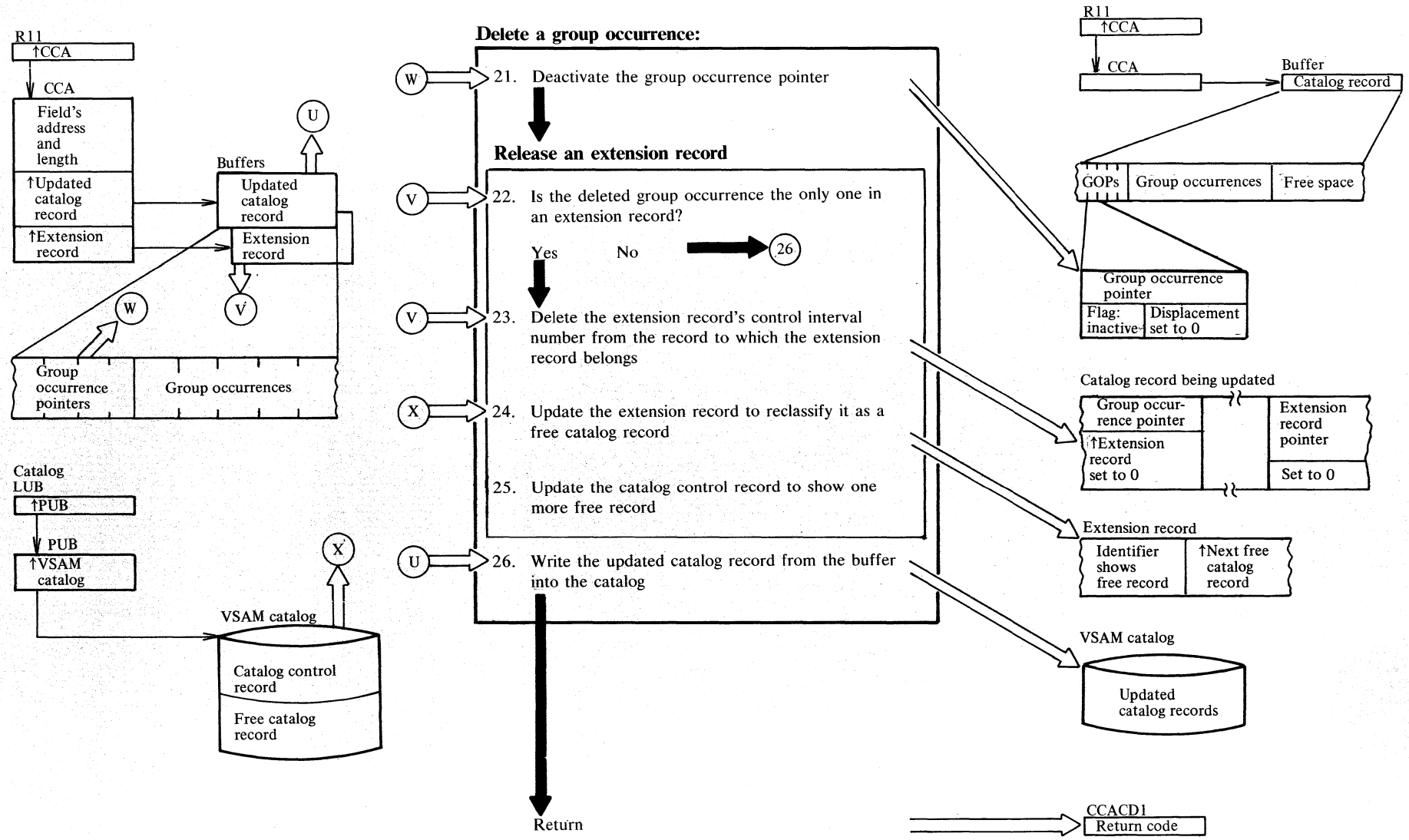


Diagram DL4. Modify a catalog record field's value



Notes for Diagram DL (part 1 of 3)

	Description	Module	Procedure
1	The modify routine initializes each CTGFL with the dictionary entry associated with the CTGFL's field-name value.	IGG0CLAV IGG0CLAY	IGGPMOD IGGPSCNC
2	The field parameter list (CTGFL) contains the field's name, type code, length, and displacement from the beginning of the record or group occurrence. If the field exists, it is either a header field (group code 0) or a field within a group occurrence. If the caller supplied modifying data and test conditions, the field is being altered. If the caller supplied modifying data and no test conditions, a group occurrence is to be added to the records. If the caller identified a group occurence combination field name but didn't supply modifying data, the group occurrence is being deleted.	IGG0CLAV IGG0CLBA IGG0CLAV IGG0CLBT IGG0CLAV IGG0CLBT IGG0CLAV IGG0CLBT IGG0CLAV IGG0CLBA	IGGPMOD IGGPSFPL IGGPTSTS IGGPSFPL IGGPXLT2 IGGPSFPL IGGPXDGO IGGPSFPL IGGPXEL2 IGGPSFPL IGGPGREC
3	A new group occurrence requires a group occurrence pointer (GOP).	IGG0CLAV IGG0CLAW	IGGPSFPL IGGPADGO
4	If a new GOP is built, it is put into the catalog record at the end of its group of GOPs. The GOPs are grouped by group code and, within the group-code group, are ordered by sequence number. If the new GOP causes the catalog record to overflow, an extension record is obtained from the catalog's free control intervals. All group occurrences in the original record are put into the extension record. The GOPs displacement value (in the original record) is replaced with the control interval number of the extension record. In addition, a GOP is built and put into the extension record for each group occurrence in the extension record. The GOP in the extension record contains the displacement from the beginning of the record to its group occurrence.	IGG0CLBA IGG0CLAW IGG0CLAG IGG0CLAW IGG0CLBA	IGGPGREC IGGPADGO IGGPAXCI IGGPADGO IGGPGREL IGGPGREC
5	If the new GOP causes the catalog record to overflow and the catalog record contains only GOPs, an extension record is obtained to contain the new GOP. The original record's extension field contains the control interval number of the extension record. This routine activates the GOP by setting its inactive flag off.	IGG0CLAW IGG0CLAG IGG0CLAW IGG0CLBA IGG0CLAW IGG0CLAX IGG0CLAW	IGGPAGOP IGGPAXCI IGGPAGOP IGGPGREC IGGPAGOP IGGPMGO IGGPAGOP IGGPIGOP
6	The new group occurrence contains fixed-length fields and, possibly, variable-length fields. If the new group occurrence causes the record to overflow, an extension record is obtained to contain the new group occurrence.	IGG0CLAW	IGGPADGO
8	Replace the initial field values (from step 6) with the caller-supplied values addressed by the CCA.	IGG0CLAW	IGGPADGO IGGPMVGO
9	If there are no more CTGFLs to process, IGGPSFPL calls IGGPPREC to write each updated catalog record into the catalog.	IGG0CLAV IGG0CLAW	IGGPSFPL IGGPPREC
11	The CCACPE7x field contains the field's address. The CTGFL contains the address and length of the data with which to update the field.	IGG0CLAX IGG0CLBA	IGGPALT2 IGGPGVAL
12	The CTGFL flags field (from the catalog field name directory) specifies field type.	IGG0CLAX	IGGPALT2
13	The CTGFL contains the length and address of the updating data. The data is in the caller's work area.	IGG0CLAX	IGGPALT2

Notes for Diagram DL (part 2 of 3)

	Description	Module	Procedure
14	The CTGFL flags field (from the catalog field name directory) specifies field type. If the length of the data to update the field within the CTGFL isn't equal to the field's length in the CCA, the variable-length field's length is either increased or decreased, causing a corresponding reduction or increase in the catalog record's amount of free space. Note: If the catalog is recoverable, the call to IGG0CLAG will result in a CRA update unless bit CCARPUT (the update inhibit bit) is set to '1'. This bit is set in the calling procedures, which decide whether CRA updating is necessary for a given operation.	IGG0CLAX	IGGPALT2
			IGGPMVAR
			IGGPMBGO
			IGGPMVAR
		IGG0CLBA	IGGPGVAL
		IGG0CLAX	IGGPMVAR
		IGG0CLAG	IGGPAXCI
		IGG0CLAX	IGGPMVAR
		IGG0CLAW	IGGPGNEX
		IGG0CLAX	IGGPMVAR
		IGG0CLBA	IGGPGREC
		IGG0CLAX	IGGPMVAR
			IGGPMGO
			IGGPMVAR
		IGG0CLBW	IGGPDEIN
			IGGPRISE
		IGG0CLAG	IGGPAXCI
		IGG0CLBW	IGGPRISE
		IGG0CLBA	IGGPGVAL
		IGG0CLBW	IGGPRISE
		IGG0CLAX	IGGPSHNK
		IGG0CLBW	IGGPRISE
		IGG0CLAW	IGGPPREC
		IGG0CLBW	IGGPDEIN
			IGGPSINK
		IGG0CLAX	IGGPSHNK
		IGG0CLBW	IGGPSINK
		IGG0CLAX	IGGPEXPD
		IGG0CLBW	IGGPDEIN
			IGGPSNK2
		IGG0CLAG	IGGPAXCI
		IGG0CLBW	IGGPSNK2
			IGGPDOWN
		IGG0CLAG	IGGPAXCI
		IGG0CLBW	IGGPDOWN
		IGG0CLBA	IGGPGVAL
		IGG0CLBW	IGGPDOWN
		IGG0CLAW	IGGPGNEX
		IGG0CLBW	IGGPDOWN
		IGG0CLAX	IGGPSHNK
		IGG0CLBW	IGGPDOWN

	Description	Module	Procedure
		IGG0CLAX	IGGPEXPD
		IGG0CLBW	IGGPDOWN
		IGG0CLAW	IGGPPREC
		IGG0CLBW	IGGPSNK2
		IGG0CLAW	IGGPPREC
		IGG0CLBW	IGGPSNK2
		IGG0CLBA	IGGPGREC
		IGG0CLBW	IGGPSNK2
		IGG0CLAG	IGGPPDE
		IGG0CLBW	IGGPDEIN
		IGG0CLBA	IGGPGVAL
	If the modification occurs in a base catalog record, the affected group occurrence is moved from the base catalog record into an extension record. If the available space pointer doesn't have an extension control interval number, an extension control interval number is assigned to the caller.	IGG0CLAX	IGGPMBGO
		IGG0CLAG	IGGPAXCI
		IGG0CLAX	IGGPMBGO
			IGGPMGO
	The length of the group occurrence to be moved is computed. The group occurrence and the GOPs are moved and updated.	IGG0CLAX	IGGPMGO
		IGG0CLAV	IGGPSGOP
		IGG0CLAX	IGGPMGO
		IGG0CLAW	IGGPIGOP
	The variable-length field's length bytes are replaced with the length of the data with which to update the field (in the CTGFL).	IGG0CLAX	IGGPMGO
			IGGPCGO
			IGGPMGO
			IGGPDGO
			IGGPMGO
			IGGPDGOP
15	The eliminated bytes at the end of the record are added to the record's free space.	IGG0CLAX	IGGPMVAR
			IGGPSHNK
16	The additional bytes are obtained by reducing the record's free space.	IGG0CLAX	IGGPMVAR
			IGGPEXPD

Notes for Diagram DL (part 3 of 3)

	Description	Module	Procedure
	If the increased length causes the catalog record to overflow, an extension record is obtained. The original record's data is split so that part remains in the original record and part is moved into the extension record. Each associated GOP is updated to show the new position of its group occurrence.		
18	The catalog control record (CCR) contains pointers to two chains: one of unassigned records (CIs) and one of free (F type) records. The unassigned records are used first, down to a minimum 'reserve' of 4 (needed for catalog extension), after which the requests are filled from the free record chain.	IGG0CLAX	IGGPALT2
19	The group occurrence data is split so that part remains in the original record and part is moved into the extension record.	IGG0CLAX	IGGPALT2
20	Each GOP is updated to show the new position of its group occurrence.	IGG0CLAX	IGGPALT2
21	The GOP's inactive flag is set on, thereby deactivating it. The GOP's type code and sequence number fields are unchanged. The displacement field is set to 0.	IGG0CLAV	IGGPDEL2
22	If an extension record contains only deleted data, the record is reclaimed by CM as a free control interval.	IGG0CLAV	IGGPDEL2
23-25	The CCR contains the count of free control intervals available to the catalog and the control interval number of the first free control interval. All free control intervals are chained together. When a control interval is added to the free control interval chain, CM puts the CCR's free	IGG0CLAV IGG0CLAX IGG0CLAV IGG0CLBA IGG0CLAV IGG0CLAW IGG0CLAV	IGGPDEL2 IGGPDGO IGGPDEL2 IGGPGREC IGGPDEL2 IGGPPREC IGGPDEL2

Description	Module	Procedure
control interval number into the new free control interval and puts the new free control interval's control-interval number into the CCR. The CCR's free control-interval count is increased by 1. If a variable-length field within a group occurrence spans several physical records, multiple extension records must be deleted. Note: If the catalog is recoverable, the call to IGG0CLAG will result in a CRA update unless bit CCARPUT (the update inhibit bit) is set to '1'. This bit is set in the calling procedures, which decide whether CRA updating is necessary for a given operation.	IGG0CLAG IGG0CLAV IGG0CLAX	IGGPPDE IGGPDEL2 IGGPDGOP

Diagram DM1. Release extents

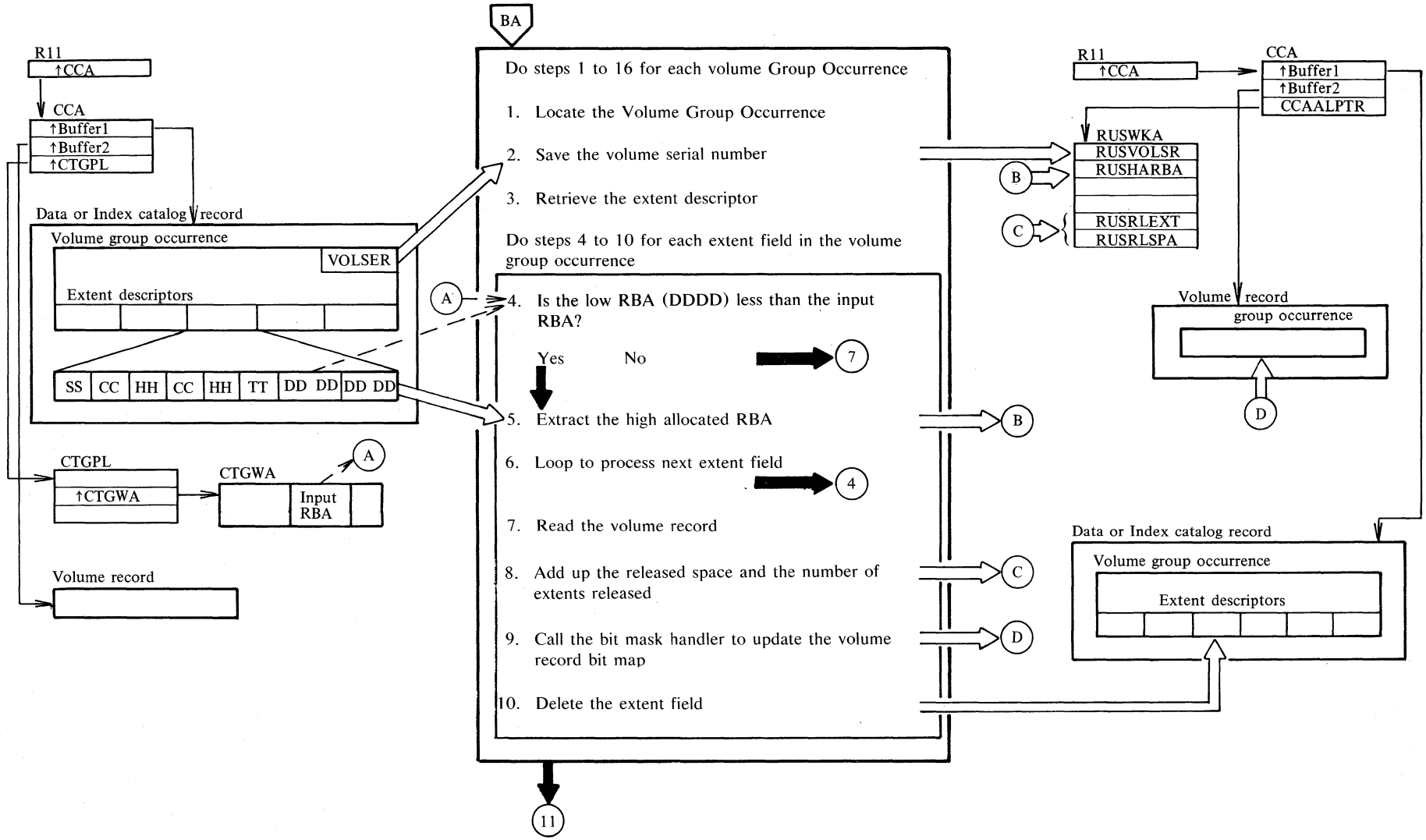


Diagram DM2. Release extents

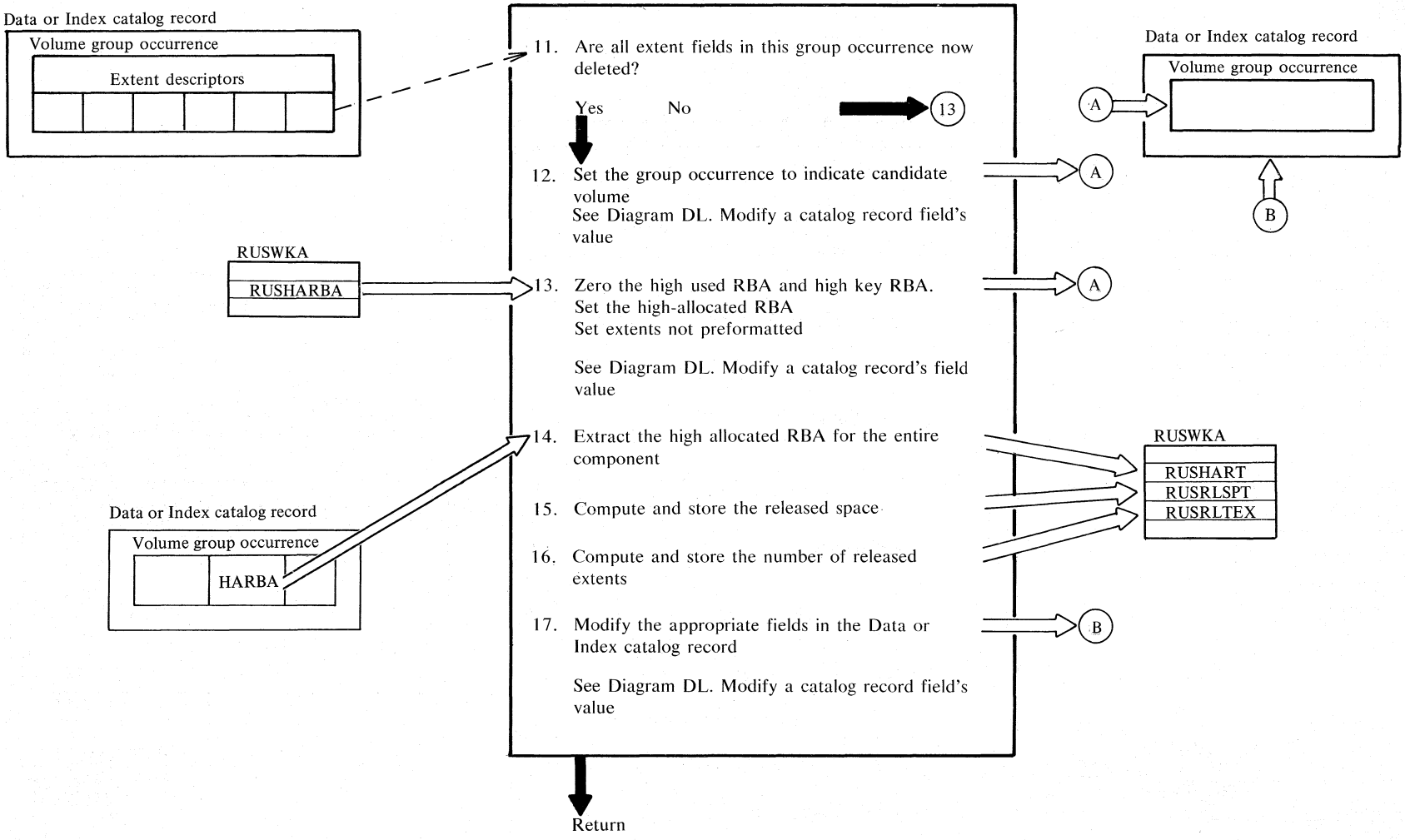


Diagram EA1. Catalog management services contents

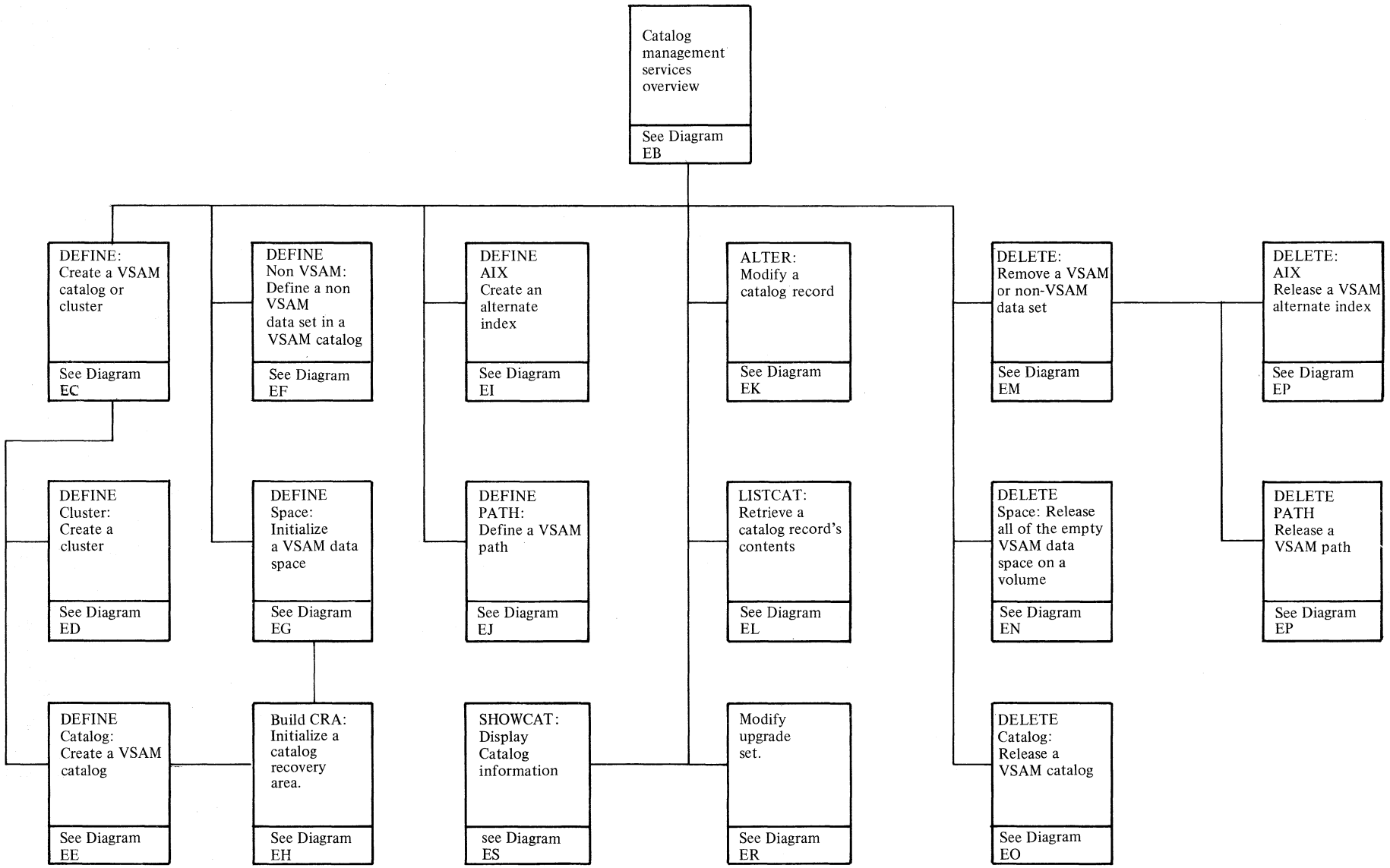


Diagram EB1. Catalog management services overview

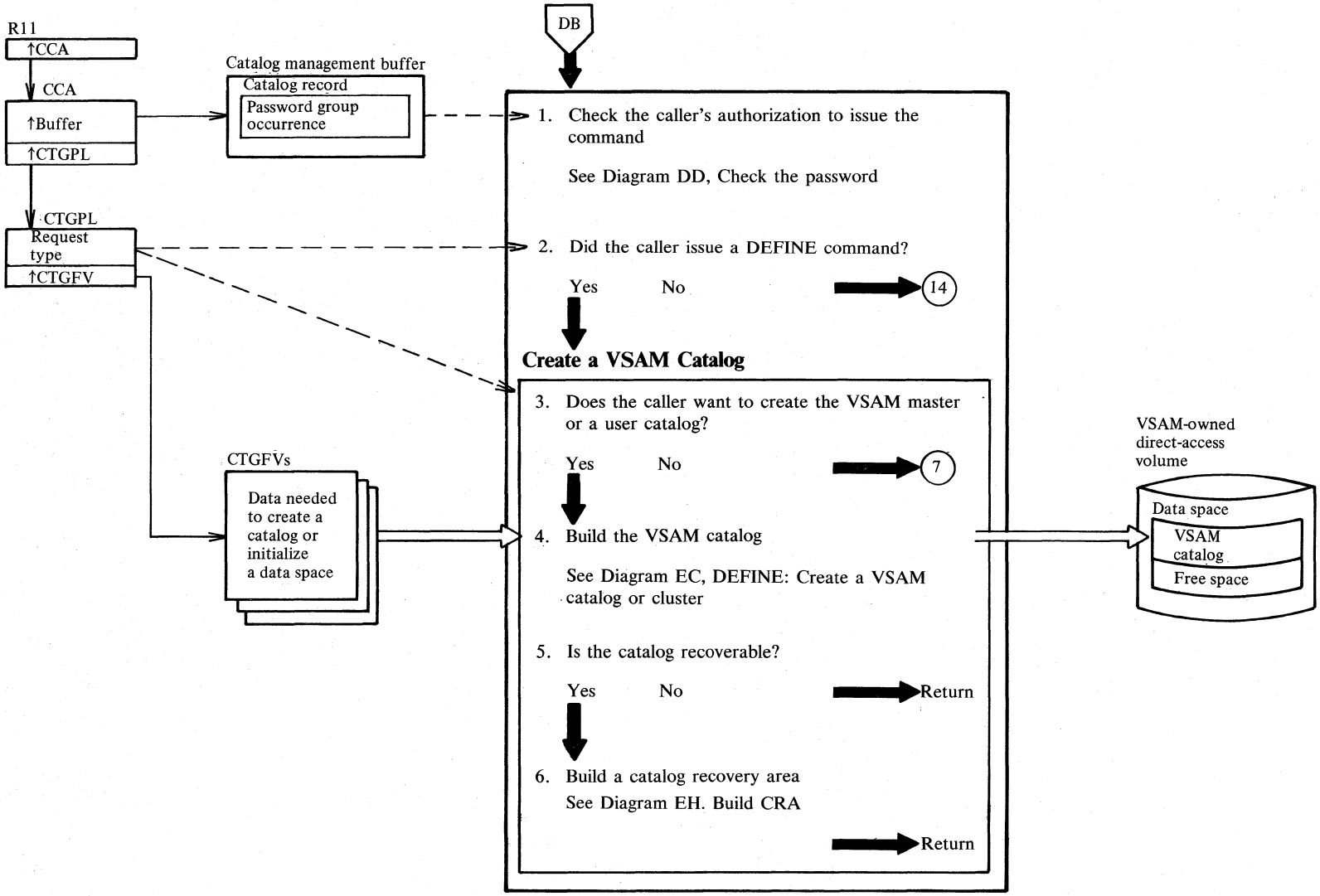


Diagram EB2. Catalog management services overview

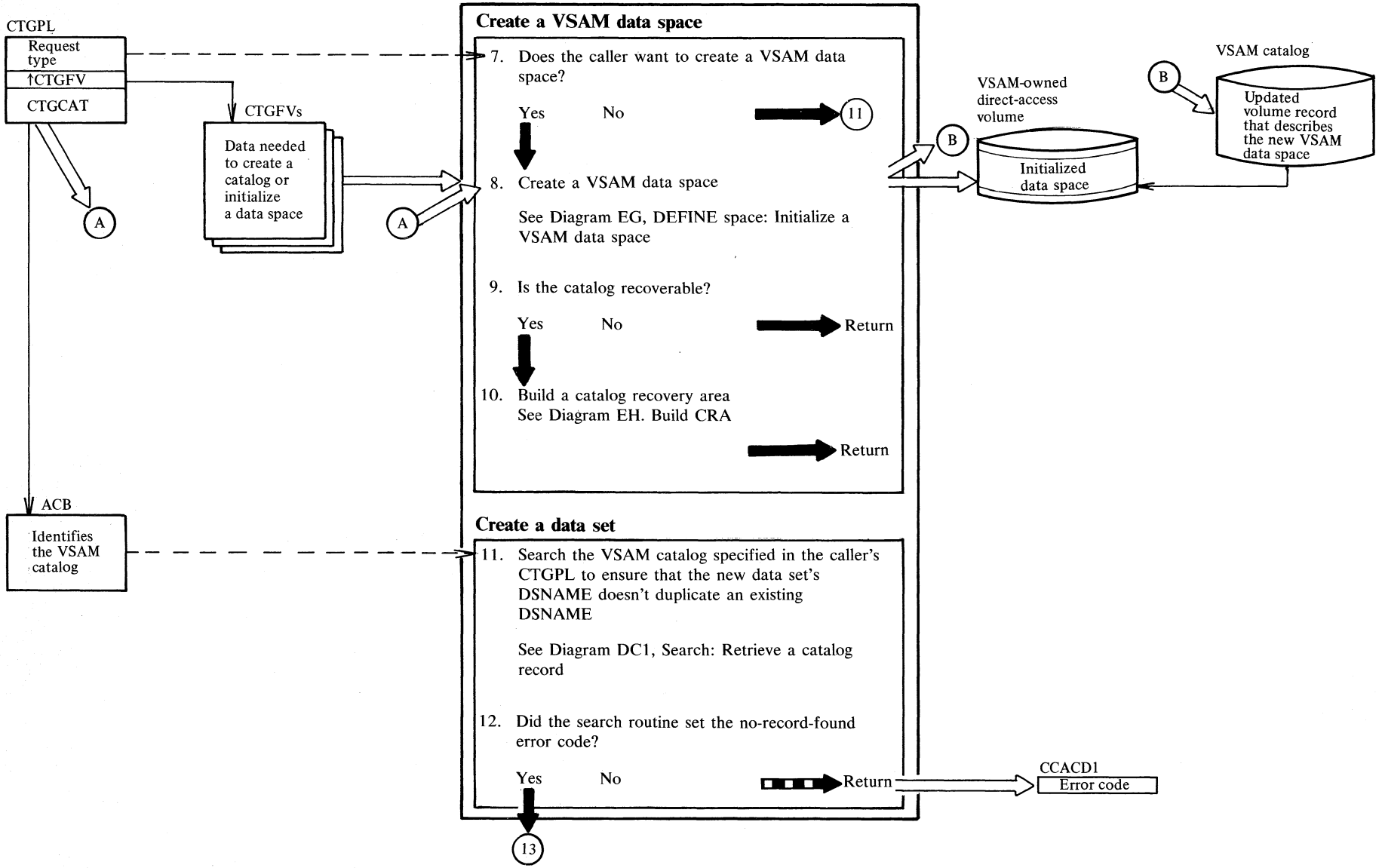


Diagram EB3. Catalog management services overview

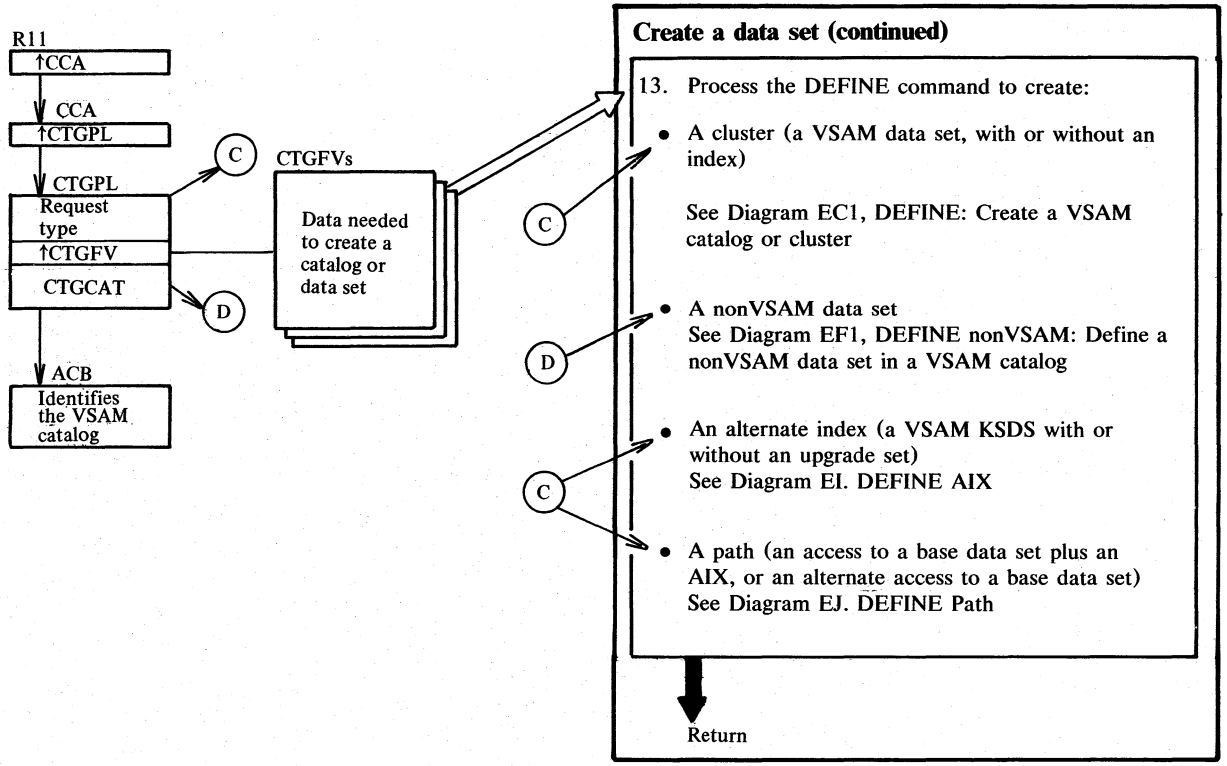


Diagram EB4. Catalog management services overview

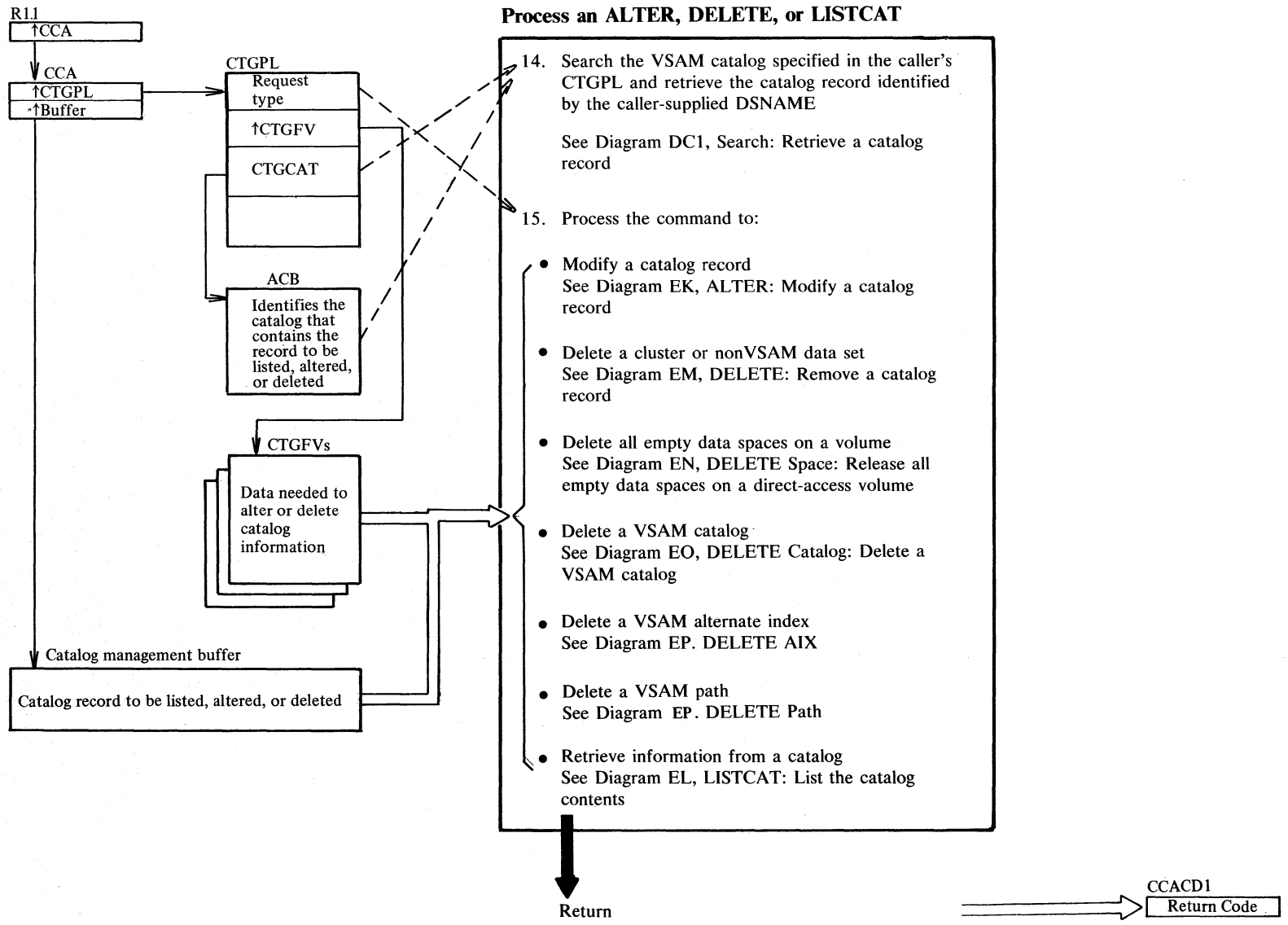
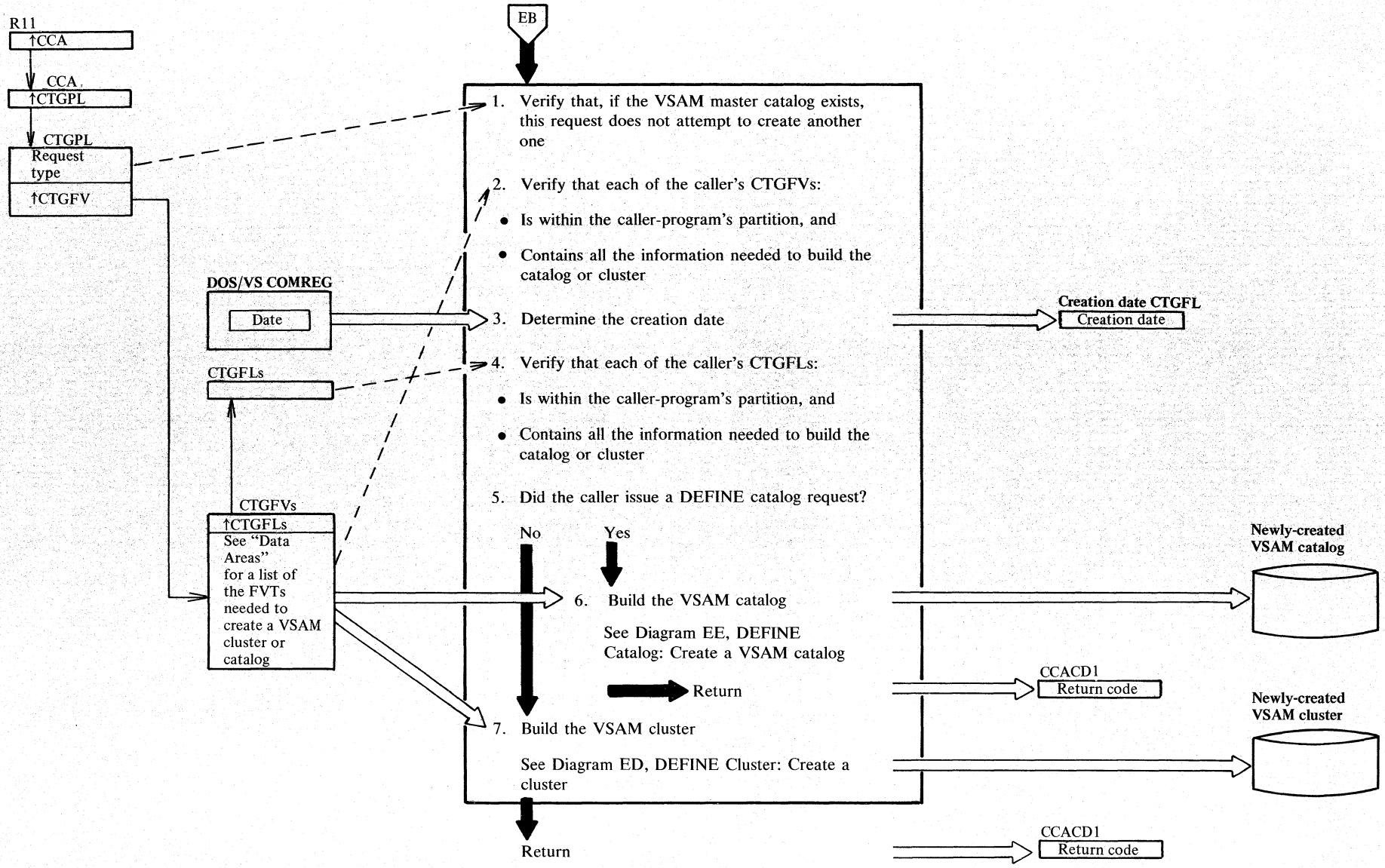


Diagram EC1. DEFINE: Create a VSAM catalog or cluster



Notes for Diagram EC

	Description	Module	Procedure
	When the user issues the Access Method Services DEFINE command to create a catalog or a cluster, the CMS DEFINE initial processing routine insures that the caller has provided all the information necessary to create a catalog or a cluster. The field CCACD1 is checked for error codes. If no errors are found, CCACD1 will be 0. See the module prologue for IGG0CLAL for specific error conditions.		
1	Only one master catalog is permitted. An attempt to define a second master catalog will result in an error.	IGG0CLAL	IGGPDEF
2	CTGFVs and CTGFLs are checked by internal procedures in IGG0CLAL:		
	Cluster CTGFV	IGG0CLAL	IGGPDCWC
	Caller's work area		IGGPDCWC
	Data CTGFV structure		IGGPDFSC
	Index CTGFV structure		IGGPDFSC
	Data DSNAME CTGPL		IGGPDDNP
	Index DSNAME CTGPL		IGGPDDNP
	Data space CTGFV		IGGPDBVC
	Catalog space CTGPL	IGG0CLAL	IGGPDCSF
	Key range CTGPL		IGGPDRPG
	File CTGFV		IGGPDEF
3	Get the current date from COMREG and insert it in the creation date in the CTGFL.	IGG0CLAL IGG0CLAG	IGGPDEDE IGGPGET
4	Calculate the amount of work area used or needed. The password and owner ID are checked in the CTGFL. The CTGFLs are then checked and completed by IGG0CLAN:	IGG0CLAL	IGGPDDEP IGGPDSTY
			IGGPDEF
		IGG0CLAN	IGGPDSCB
			IGGPDSPF
	Space parameter CTGFLs		IGGPDBSF
	Buffer-size CTGFLs	IGG0CLBX	IGGPDALR
	Average-record size CTGFLs		

	Description	Module	Procedure
6		IGG0CLAN	IGGPDSCB
		IGG0CLAP	IGGPDCDA
7		IGG0CLAN	IGGPDCSB
			IGGPDRDA
			IGGPDSCB
		IGG0CLAS	IGGPDEFC
		IGG0CLAN	IGGPDSCB
		IGG0CLAJ	IGGPDBDI

Diagram ED1. DEFINE Cluster: Create a VSAM cluster

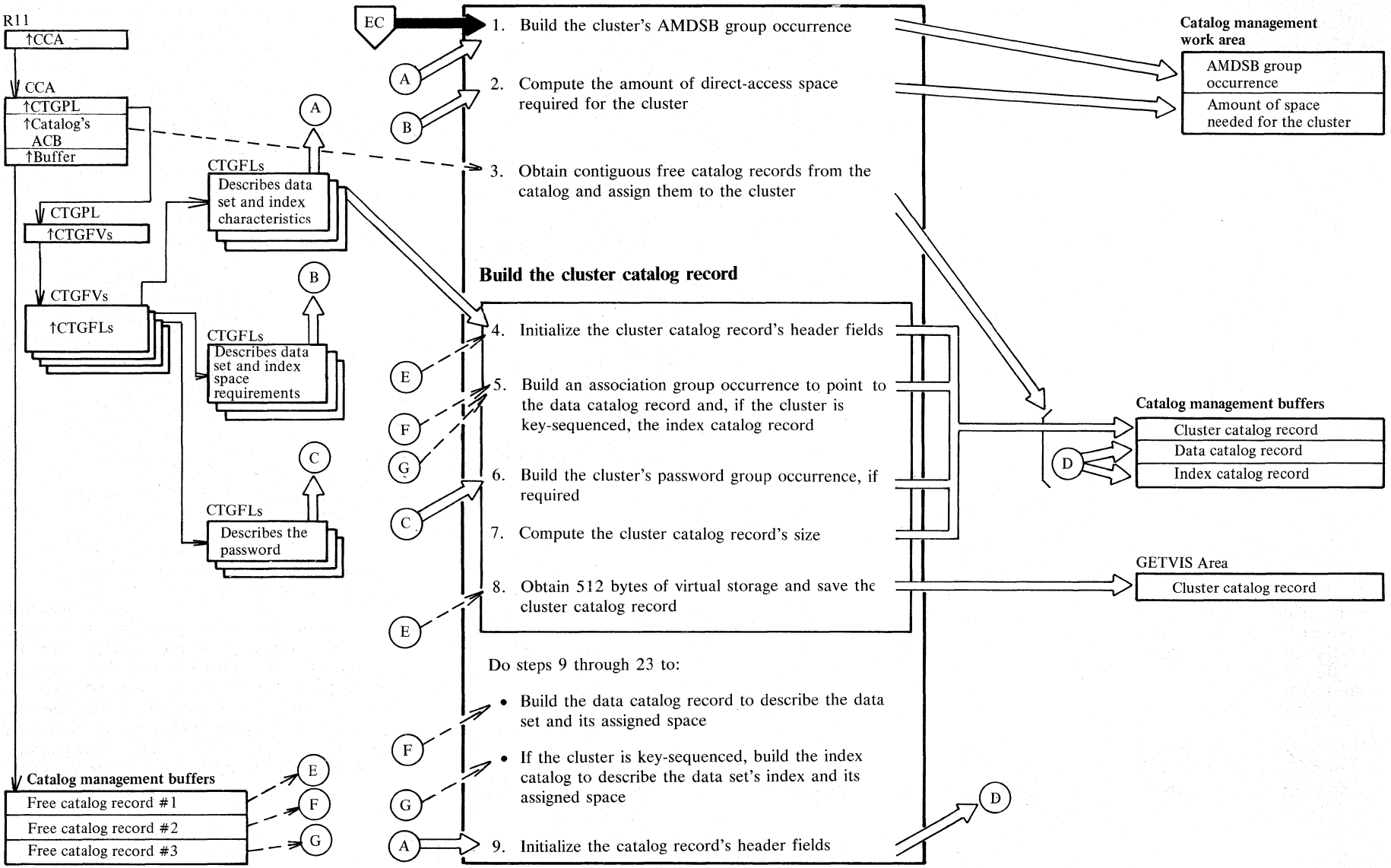


Diagram ED2. DEFINE Cluster: Create a VSAM cluster

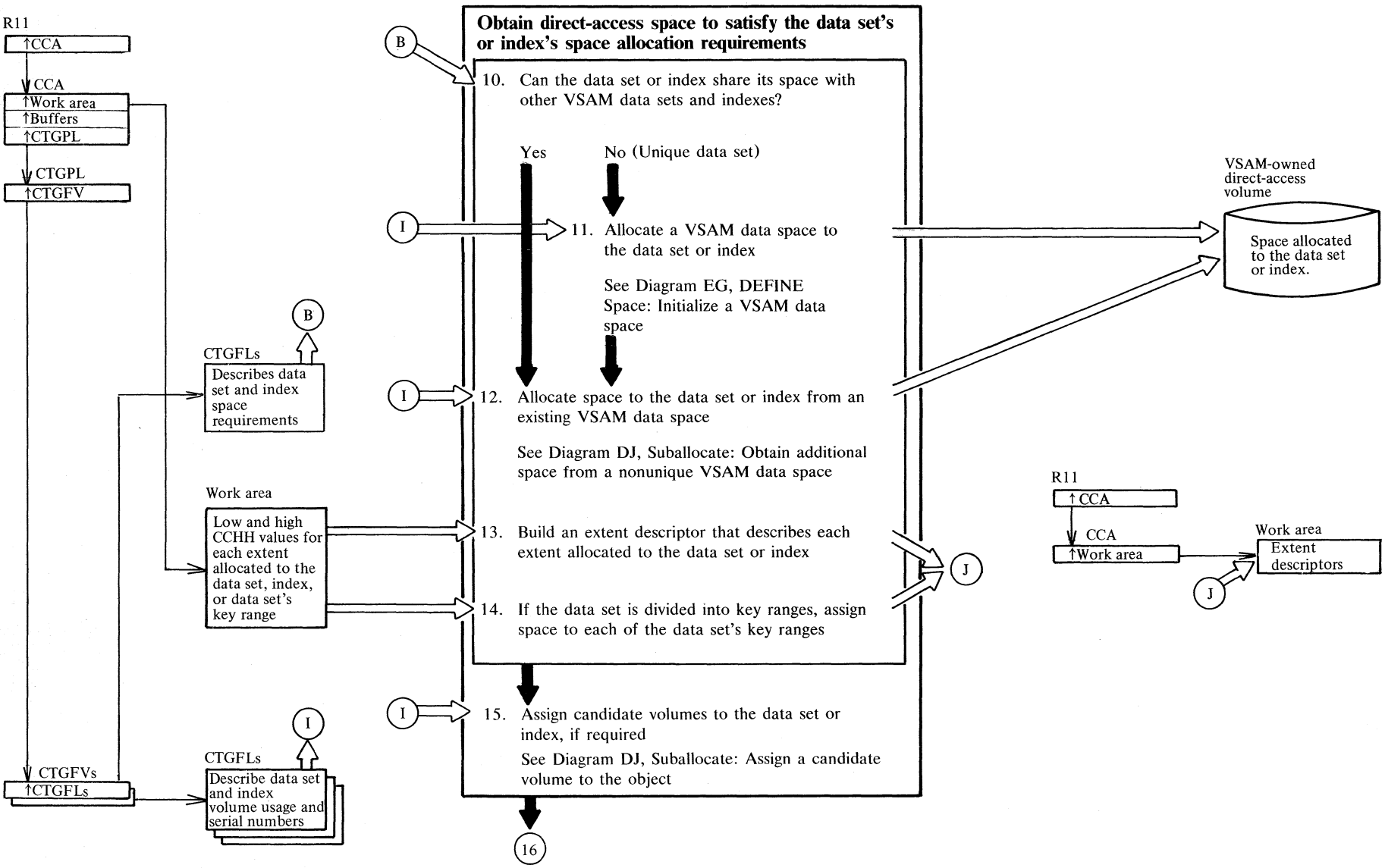
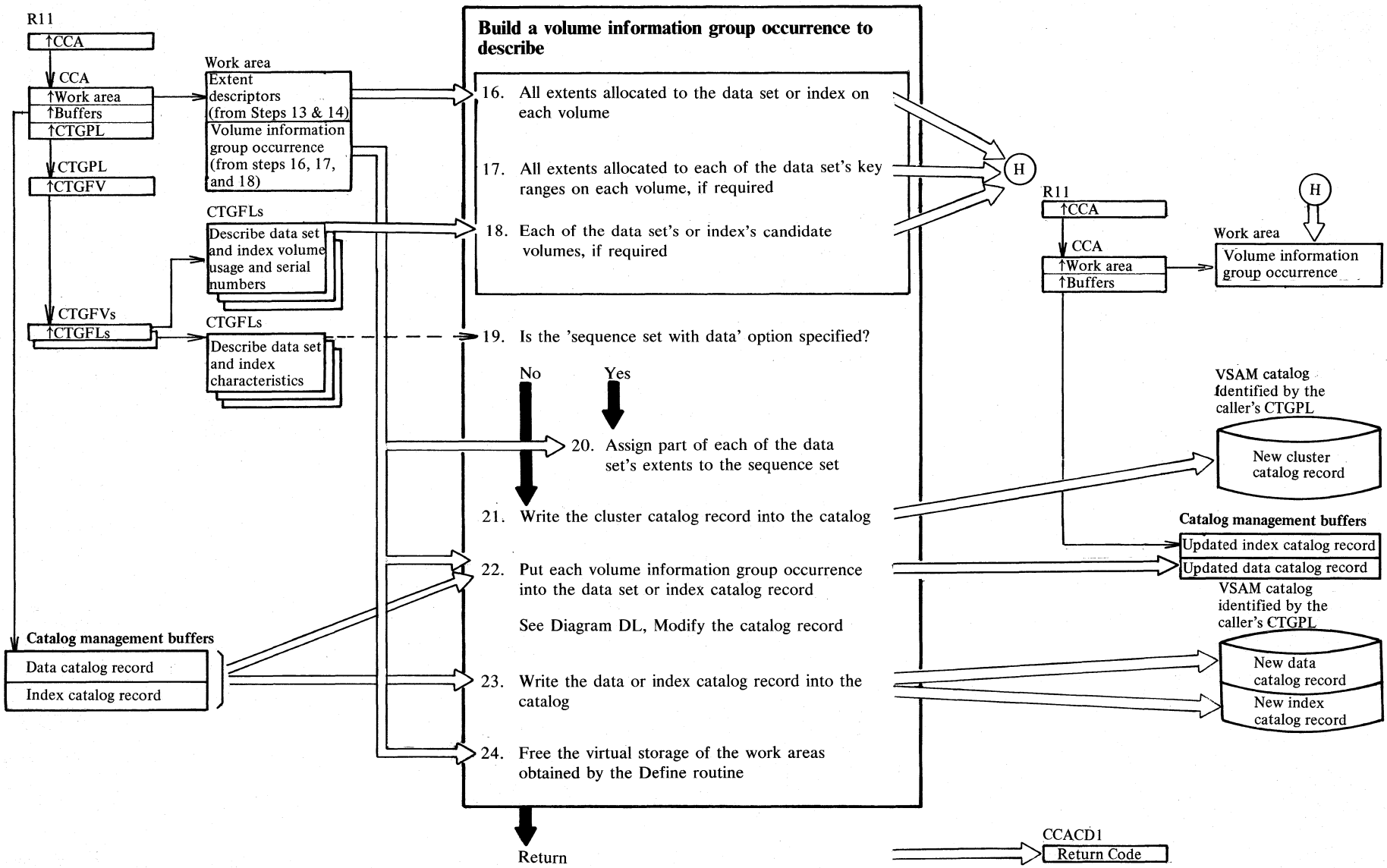


Diagram ED3. DEFINE Cluster: Create a VSAM cluster



Notes for Diagram ED (part 1 of 2)

	Description	Module	Procedure
1	The AMDSB contains the cluster's statistics and fixed characteristics. Each time the cluster is opened, the AMDSB is retrieved from the data catalog record. When the cluster is closed, the AMDSB is updated and rewritten into the data catalog record.	IGG0CLAN	IGGPDSCB IGGPDRDA
2	The field vector table (CTGFV) contains addresses of buffer-size and record-length field parameter lists (CTGFLs). This data is used to determine the data set's control-interval and control-area size. If the data set is key-sequenced, other buffer-size and record-length CTGFLs determine the index's control-interval and control-area size. If the key-sequenced data set is divided into key ranges, the size of each key range is determined. If no errors are detected, then control is returned to IGG0CLAN.	IGG0CLBX IKQVDTPE IGG0CLBX IGG0CLAG IGG0CLAN	IGGPDSPC IGGPDDCE IGGPDDCE IGGPGET IGGPDRDA
	IGG0CLBY is called to perform space parameter calculations.	IGG0CLBY	IGGPDRSP
3	A user's data set is described by a cluster catalog record, a data catalog record, and, if the data set is key-sequenced, an index catalog record.	IGG0CLAN	IGGPDCCE
	IGG0CLAG is called to get control-interval numbers assigned.	IGG0CLAG	IGGPAOCI
8	For recoverable catalogs, the CRA volume has to be known. The cluster record is therefore saved until this volume is known.	IGG0CLAN	IGGPDCCE
10		IGG0CLAN IGG0CLAJ	IGGPDSCB IGGPDBDI IGGPDSPO

	Description	Module	Procedure
11	A data space entry is added to the volume catalog record, and the data set's name is added to the volume catalog record's data set directory.	IGG0CLAJ IGG0CLAQ	IGGPDSPO IGGPDEFS
12	The data set's name is added to the volume catalog record's data set directory, and the volume catalog record's data space entry is updated to show the cylinders and tracks allocated to the new data set or index.	IGG0CLAJ IGG0CLAR	IGGPDSPO IGGPSALL
13		IGG0CLAJ	IGGPDSEX
14	Each key range is assigned physical space and space allocation continues for each range.	IGG0CLAJ	IGGPDSPO
15	A candidate volume is available to contain part of the cluster if more space is needed later. None of the candidate volume's space is allocated to the data set or index when the cluster is created.	IGG0CLAJ IGG0CLAR IGG0CLAJ IGG0CLAK	IGGPDCNV IGGPSALL IGGPDBDI IGGPDCMB
16	Each volume that contains a part of the data set or index is described by a volume information group occurrence within the data or index catalog record.	IGG0CLAK	IGGPDBVO
17	If the data set is divided into key ranges, each key range's space on a volume is described in a separate volume information group occurrence. If the key range's space is on more than one volume, each volume that contains part of the key range's space is described in a separate volume information group occurrence.	IGG0CLAK	IGGPDRNG
18	Construct the volume information group occurrence for all candidate volumes of the data set.	IGG0CLAK	IGGPDBCV

Notes for Diagram ED (part 2 of 2)

	Description	Module	Procedure
20	If the 'sequence set with data' (IMBED) option is specified, part of the data set's space is allocated to the index for sequence-set records. The low and high CCHH values in the index record's volume information group occurrence are those of the extent obtained for the data set. The low and high RBA values are for the sequence set and are relative to the index addresses.	IGG0CLAK	IGGPDSSP
		IGG0CLAK	IGGPDBCV
21	The module calls the clear buffer routine to write the cluster record into the catalog.	IGG0CLCB	IGGPCLBF
		IGG0CLAG	IGGPPAD
	Note: If the catalog is recoverable, the call to IGG0CLAG will result in a CRA update unless bit CCARPUT (the update inhibit bit) is set to '1'. This bit is set in the calling procedures, which decide whether CRA updating is necessary for a given operation.	IGG0CLCY	IGGPDMOP
		IGG0CLAV	IGGPMOD
22	The modify routine (IGG0CLAV) inserts each volume information group occurrence into the record.	IGG0CLCY	IGGPDMOP
23	A CM routine writes the completed data or index catalog record into the VSAM catalog and frees the CM buffer that contains the record.	IGG0CLCY	IGGPDMOP
		IGG0CLAV	IGGPMOD
24	Free all unneeded storage resources.	IGG0CLA8	IGGPDFRS

Diagram EE1. DEFINE Catalog: Create a VSAM catalog

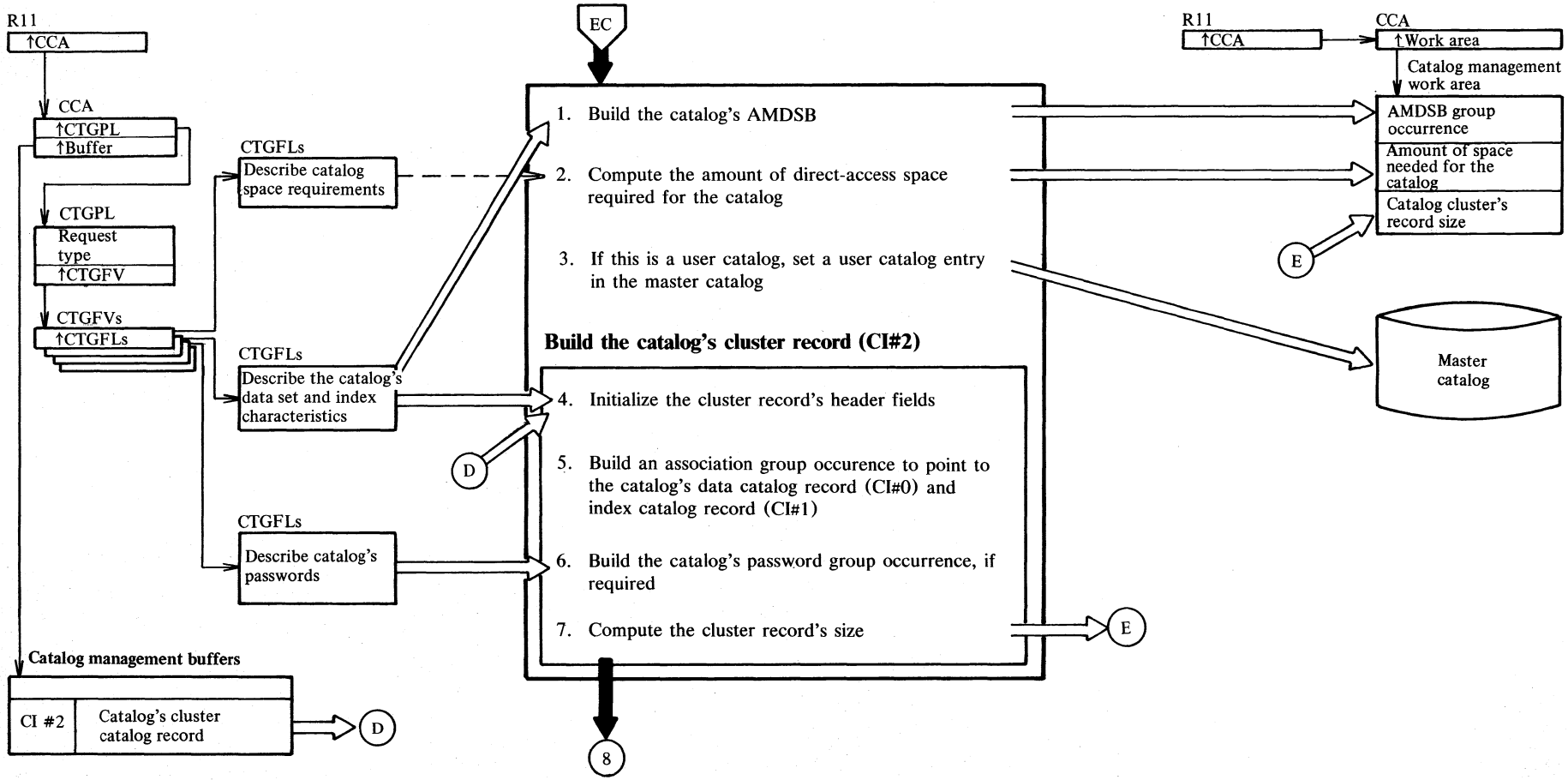


Diagram EE2. DEFINE Catalog: Create a VSAM catalog

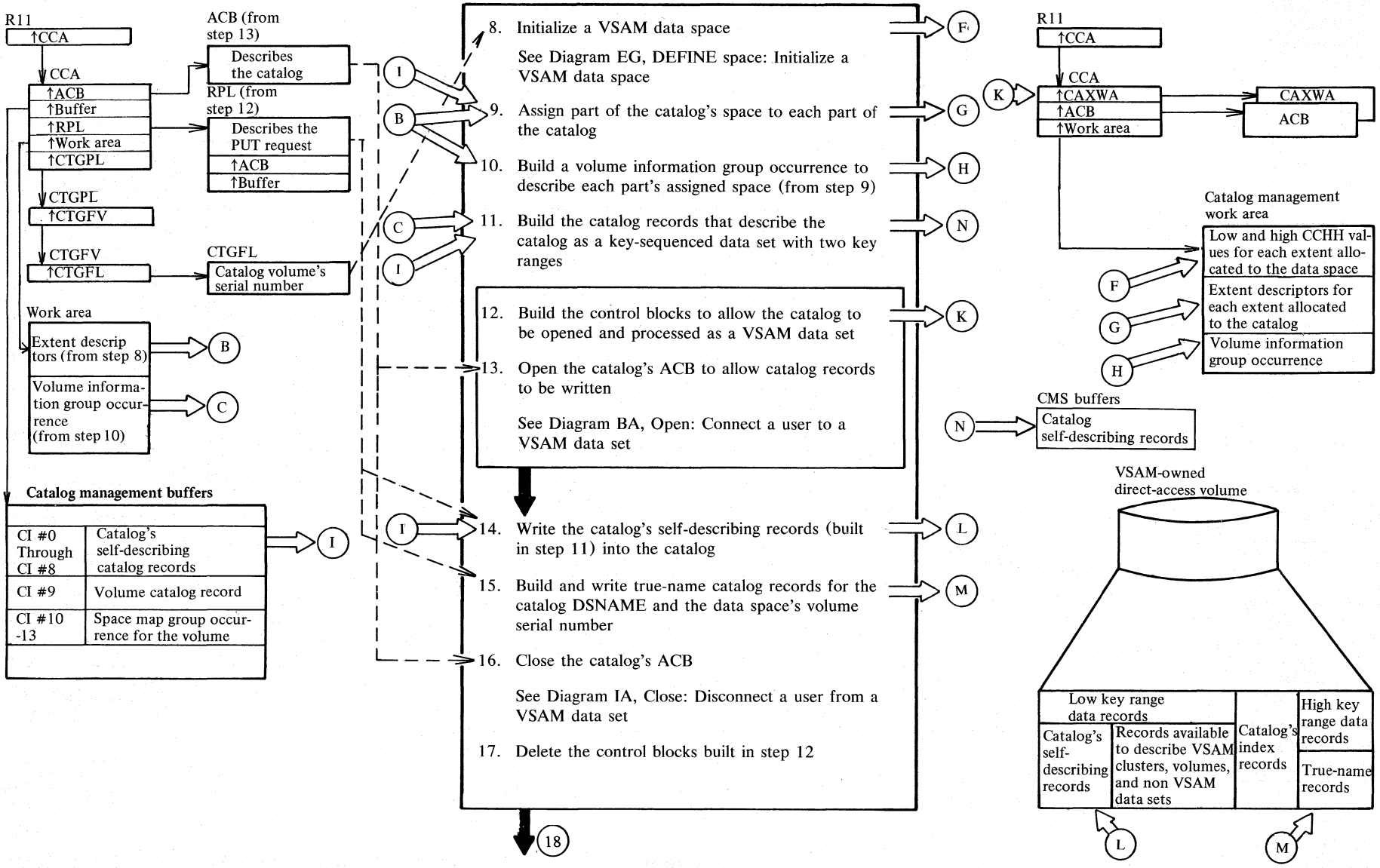
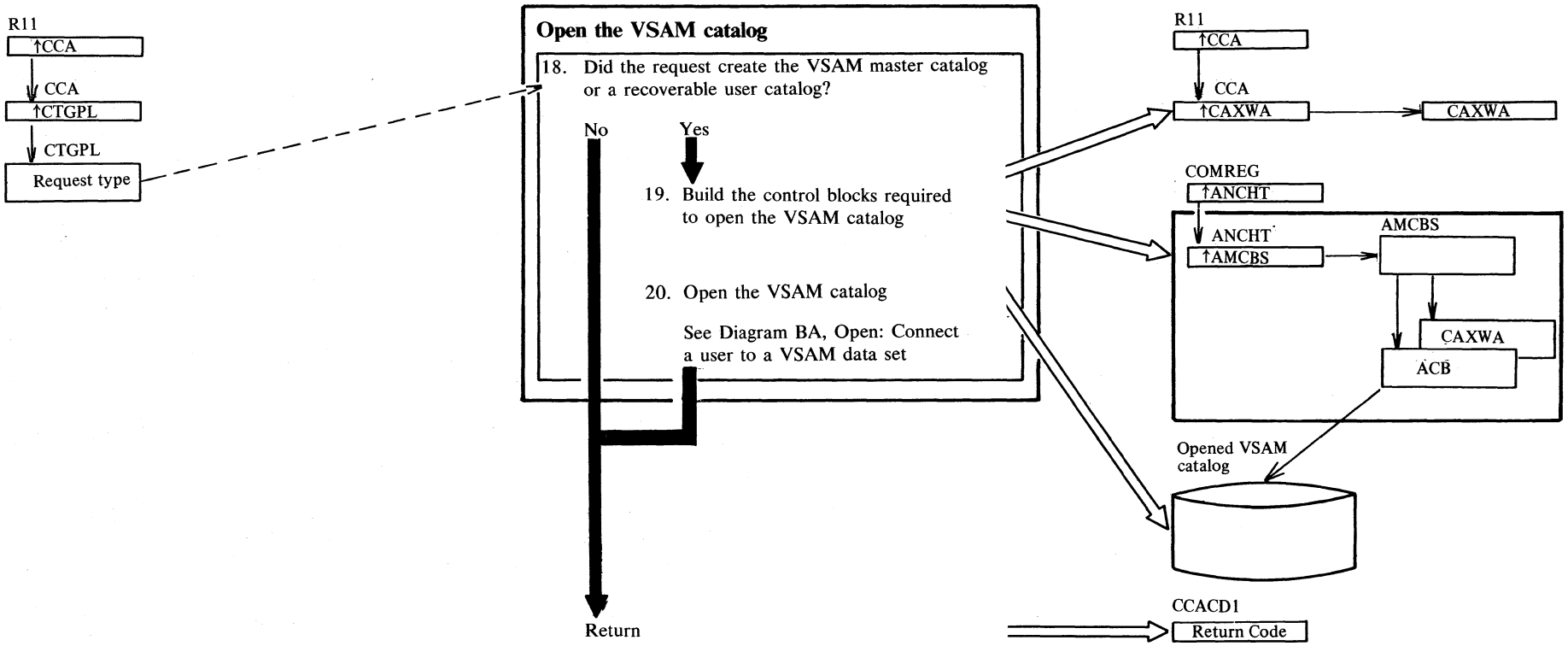


Diagram EE3. DEFINE Catalog: Create a VSAM catalog

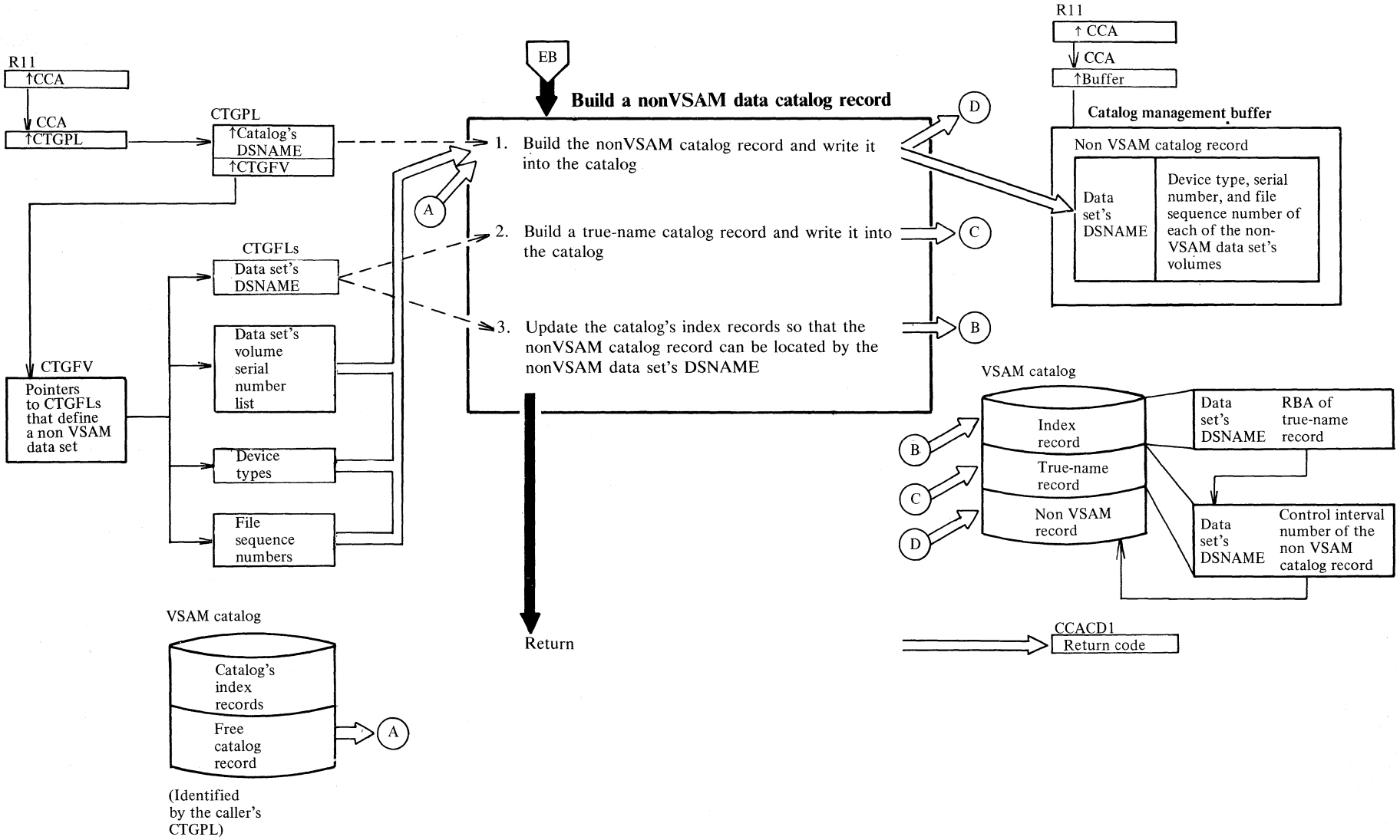


Notes for Diagram EE

	Description	Module	Procedure
1	The AMDSB contains the catalog's statistics and fixed characteristics. Each time the catalog is opened, the AMDSB is retrieved from the catalog's data catalog record (CI #0). When the catalog is closed, the AMDSB is updated and rewritten into the data catalog record.	IGG0CLAP IKQVDTPE	IGGPDCDA IGGPDCVS
2	The field vector table (CTGFV) contains addresses of buffer-size and record-length CTGFLs. This data is used to determine the catalog's control-interval and control-area size, and the amount of space required for the catalog.	IGG0CLAP	IGGPDCSP
3	User catalogs are internally identical to master catalogs. The only difference is that the master catalog contains an entry pointing to the user catalog.		
5	The cluster catalog contains an associated data set group occurrence to locate the catalog's data catalog record (CI #0) and an associated index group occurence to locate the catalog's index catalog record (CI #1).	IGG0CLAN	IGGPDCCE
8	The VSAM catalog is always built in a data space that can contain other VSAM data sets and indexes. A new data space is allocated to the VSAM routine, and the data space is assigned to the new data set.	IGG0CLAS IGG0CLAQ	IGGPDCSP IGGPDEFS
	A data set directory entry containing the data set's DSNAME is added to the volume catalog record.	IGG0CLAS IGG0CLAR	IGGPDCSP IGGPSALL
9	The catalog might contain records that describe the user's VSAM data sets, the user's nonVSAM data sets, and direct-access volumes.	IGG0CLAS	IGGPDCLD
10		IGG0CLAS	IGGPDCVO

	Description	Module	Procedure
11	See 'Data Areas' for details about the first ten records in the catalog. These records define the catalog as a key-sequenced VSAM data set, describe the space allocated to the catalog's data records, index records, and true-name records, describe the free space within the catalog, and describe the allocated tracks on the catalog volume.	IGG0CLAS IGG0CLAV	IGGPDCBE IGGPDCEB IGGPDCME IGGPMOD
12		IGG0CLAE	IGGPDCOC IGGPDCCB
13		$$BOPEN	
14		IGG0CLAE IGG0CLCG	IGGPDCOC IGGPDCPR IGGPXIO
15		IGG0CLAE	IGGPDCOC
16		$$BCLOSE	
17		IGG0CLAE	IGGPDCOC
19		IGG0CLAD	IGGPMCO2
20			IGGPDCOC
	The define routine (IGG0CLAS) sets a return code in CCACD1 and returns to the caller whenever an error is detected.	IGG0CLAT	IGGPCDVR

Diagram EF1. DEFINE Non-VSAM: Define a non-VSAM data set in a VSAM catalog



Notes for Diagram EF

	Description	Module	Procedure
1	The define nonVSAM routine transfers the nonVSAM catalog record from a CM buffer in virtual storage to the VSAM catalog specified by the caller's DEFINE command.	IGG0CLBH	IGGPDEFA IGGPDAIN
	Control interval numbers are assigned for the nonVSAM data set.	IGG0CLBH IGG0CLAG	IGGPDAIN IGGPAOCI
	The 8-character device name is converted into a 4-character device code, using the device name table.	IGG0CLBH IKQDNT	IGGPDAIN IGGPDANL
	IGG0CLAV is called to add volume occurrences.	IGG0CLBH IGG0CLAV	IGGPDAVO IGGPMOD
	If an error occurs during the define process, IGG0CLAN is called to back out any CI numbers assigned and any entries put into the VSAM catalog.	IGG0CLBH IGG0CLAN	IGGPDAVO IGGPDEFA

Diagram EG1. DEFINE Space: Initialize a VSAM data space

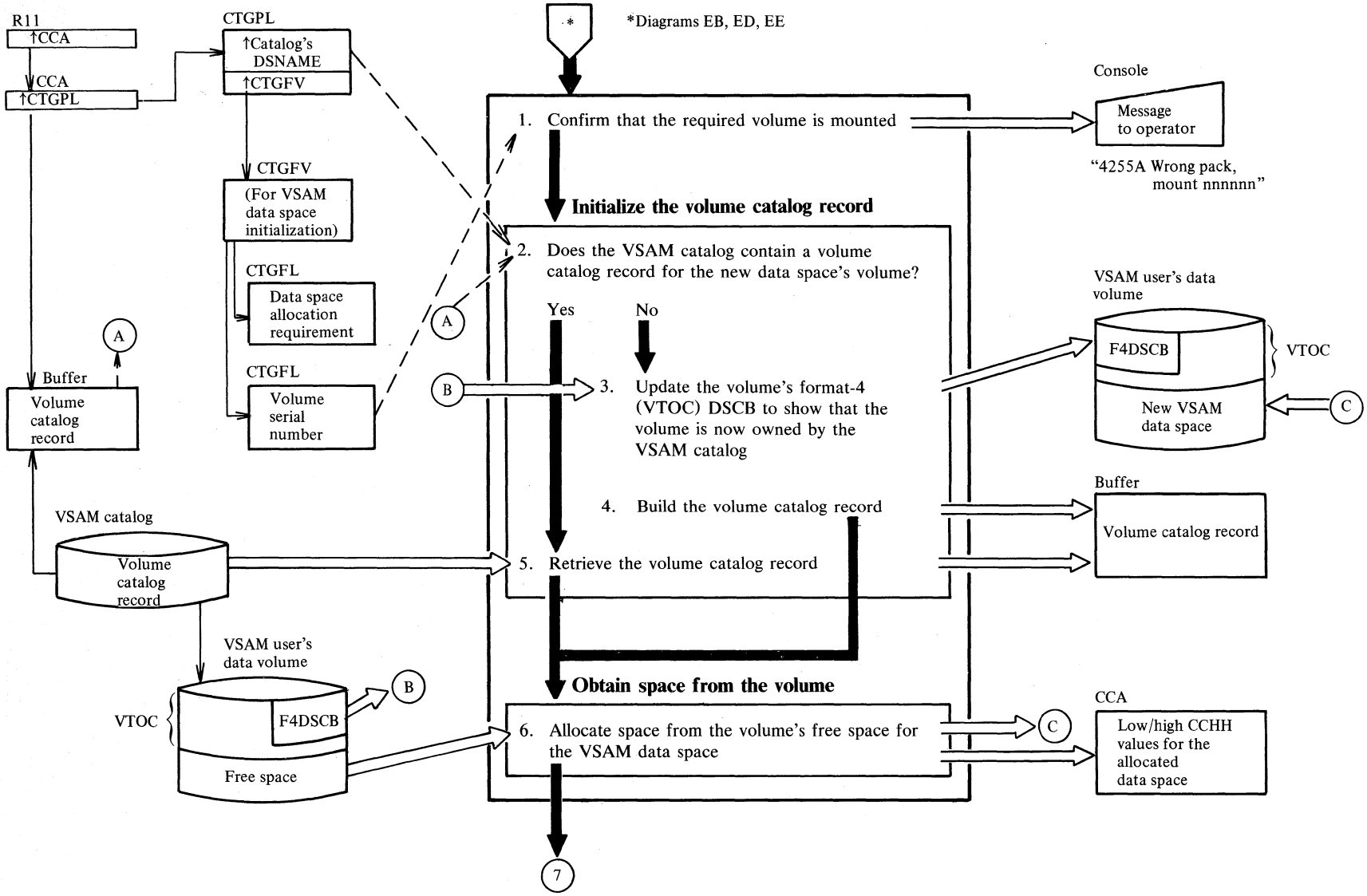
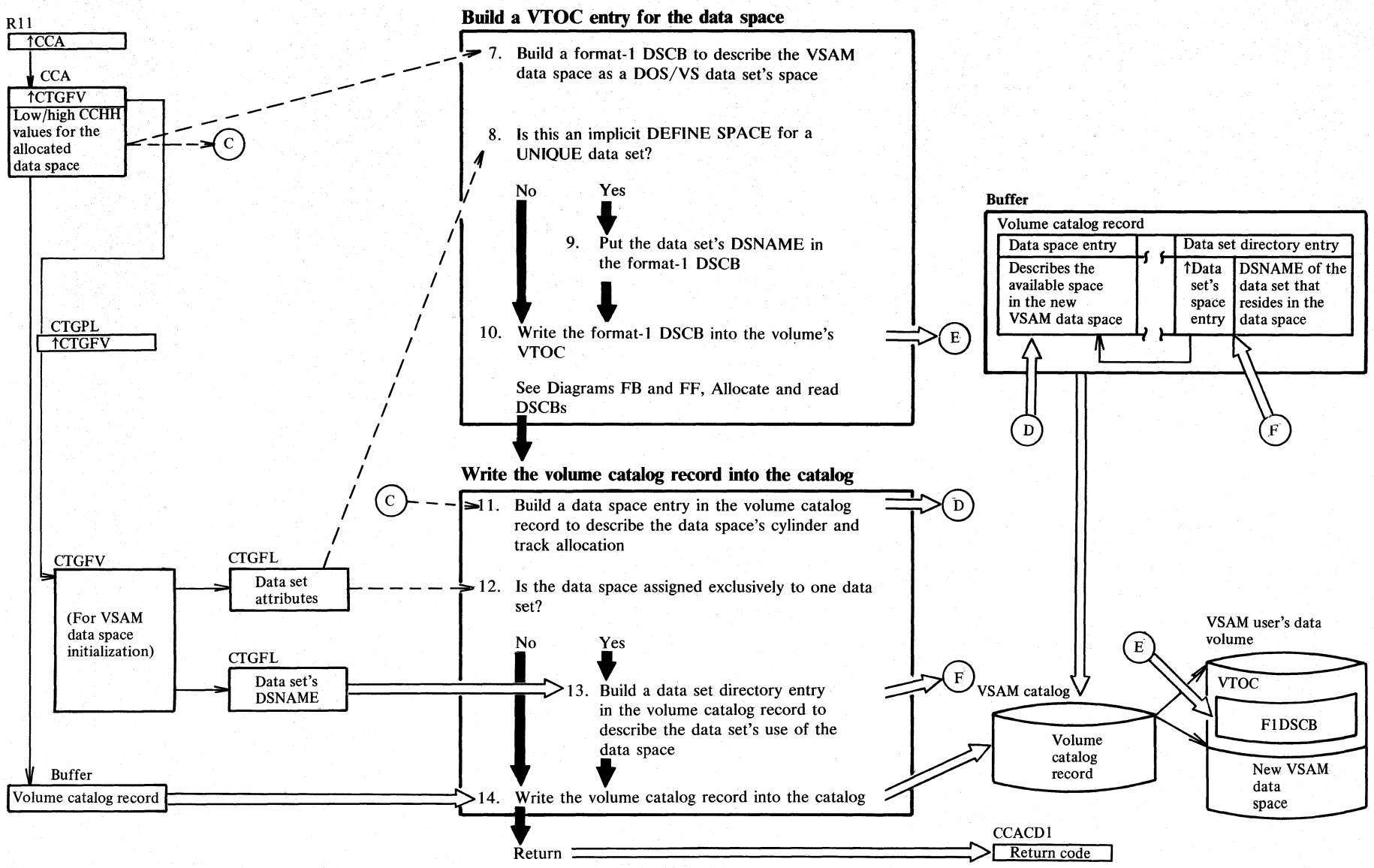


Diagram EG2. DEFINE Space: Initialize a VSAM data space



Notes for Diagram EG

	Description	Module	Procedure
1		IGG0CLAQ IGG0CLA6 $$BOVS01 IKQVTC00	IGGPDEFS IGGPVMTV
2	If a volume catalog record exists for the volume and if the volume already contains a VSAM data space, a data space group occurrence is added to the volume catalog record to describe the new VSAM data space.		
3	If the volume is a candidate volume (one that is eligible to contain a VSAM data space, but doesn't as yet), the volume's format-4 (VTOC) DSCB is updated to show that the VSAM catalog is now the volume owner.	IGG0CLAQ IGG0CLA6 IGG0CLBU	IGGPDEFS IGGPF4PR IGGPF4RD IGGPF4DQ IGGPF4WR
4	See 'Data Areas' for details about the volume catalog record.	IGG0CLAQ	IGGPIVER
5	The volume catalog record is identified by the volume's serial number.	IGG0CLAQ IKQVDTPE	IGGPDEFS
9	A VSAM data space assigned exclusively to one data set is described by a format-1 DSCB that contains the data set's DSNAME. If a data space can be assigned to more than one data set, its format-1 DSCB contains a DSNAME generated by the define space routine.	IGG0CLAQ	IGGPDEFS

	Description	Module	Procedure
10	Prior to calling IKQALL00, IGG0CLAQ sets the address of the DADSM exit routine IKQDXT (located in IGG0CLAQ) into the DADSM parameter list. This routine is called by DADSM during space allocation to place the Access Method Services-specified secondary space quantity and an expiration date into those fields of the format-1 DSCB before it is written into the VTOC. The secondary space quantity is for OS/VS compatibility only. This information is not used by DOS/VS. Note: If the catalog is recoverable, the call to IGG0CLAG will result in a CRA update unless bit CCARPUT (the update inhibit bit) is set to '1'. This bit is set in the calling procedures, which decide whether CRA updating is necessary for a given operation.	IGG0CLAG IGG0CLAQ IKQALL00 IGG0CLAQ IGG0CLAQ IKQVTC00 IGG0CLAQ IKQRDS00 IGG0CLAQ $$BOVS01	IGGPPAD IGGPDEFS IGGPDEFS IGGPCOBT IGGPCOBT IGGPCOBT
11	See 'Data Areas' for details about the data space entry.	IGG0CLA6 IGG0CLAV	IGGPDFS2 IGGPCSHG IGGPCSDG IGGPDSMD IGGPMOD
13	The volume catalog record contains the identifier of each data set that resides in part or full on the volume.	IGG0CLA6	IGGPCDSD

Diagram EH1. DEFINE CRA: Create a catalog recovery area

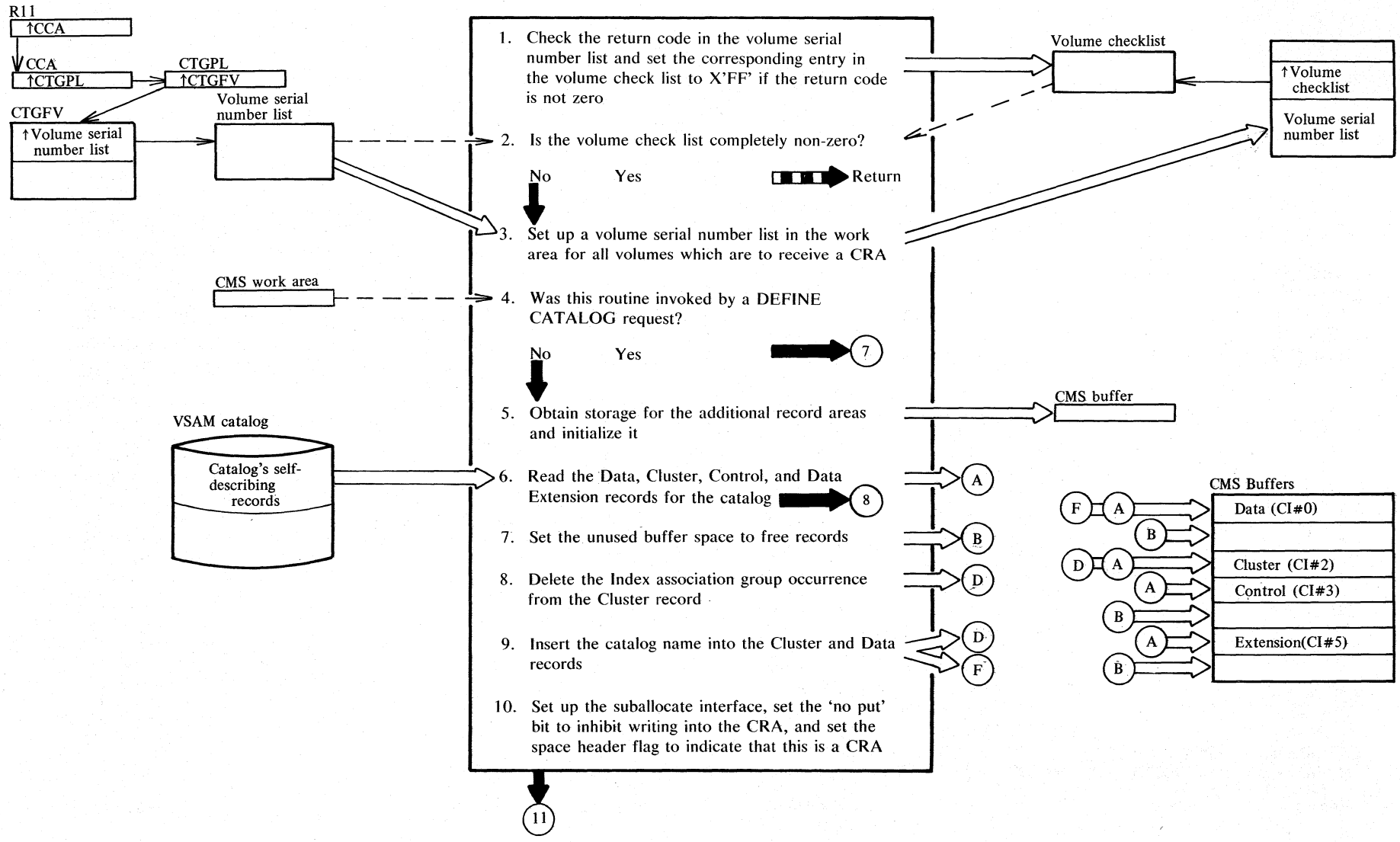


Diagram EH2. DEFINE CRA: Create a catalog recovery area

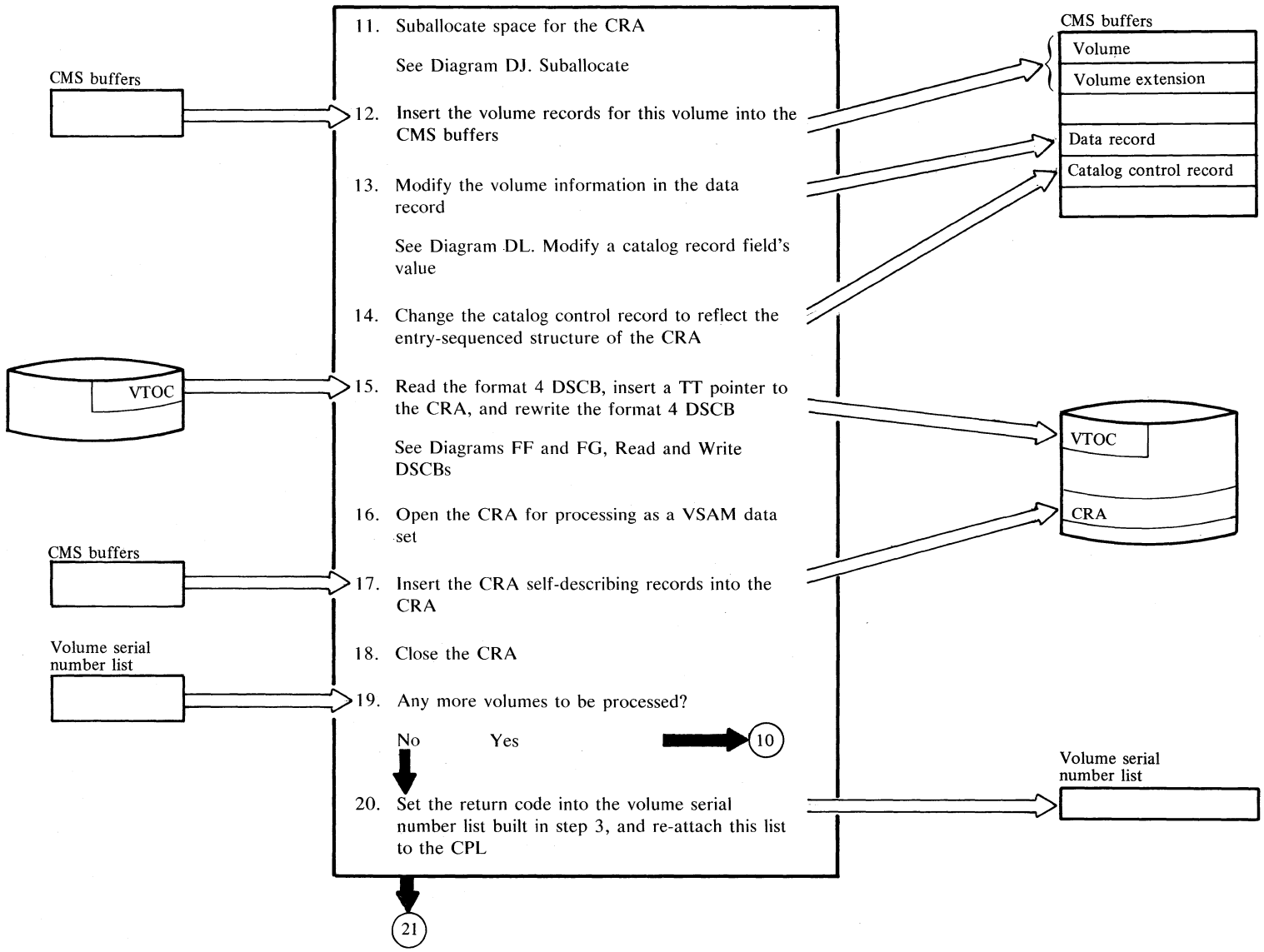
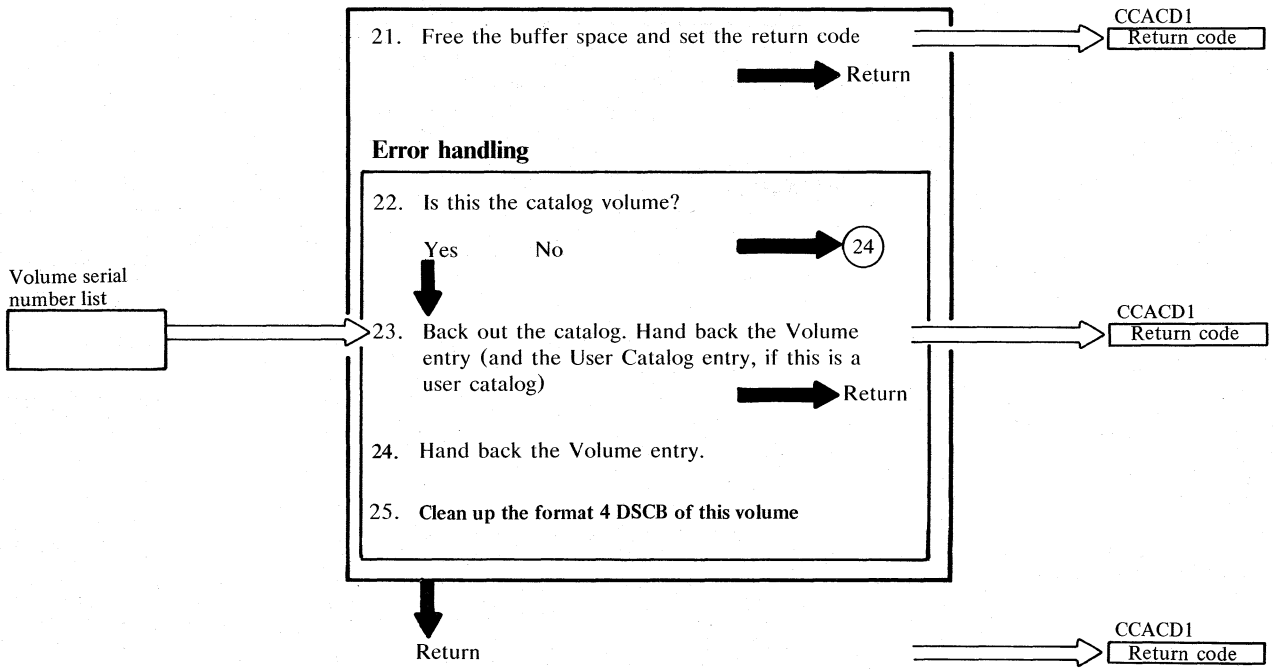


Diagram EH3. DEFINE CRA: Create a catalog recovery area



Note: Control is passed to the error handling routine (Step 22) whenever an error is detected.

Notes for Diagram EH

STEP	MODULE	PROCEDURE
1-3	IGG0CLCR	IGGPCADR
4-5	IGG0CLCR	IGGPBCRA
6	IGG0CLCR	IGGPBCRA
	IGG0CLAG	IGGPGET
7	IGG0CLCR	IGGPBCRA
8	IGG0CLCR	IGGPCRDI
	IGG0CLAV	IGGPMOD
9-10	IGG0CLCS	IGGPCRVL
11	IGG0CLAR	IGGPSALL
12	IGG0CLCS	IGGPCRVL
	IGG0CLAG	IGGPGET
	IGG0CLCS	IGGPCRVL
13	IGG0CLAS	IGGPDCRC
	IGG0CLCS	IGGPCRVL
	IGG0CLAV	IGGPMOD
14	IGG0CLCS	IGGPCRVL
15	IGG0CLA6	IGGPVMTV
	IGG0CLBU	IGGPF4RD
	IGG0CLCS	IGGPCRVL
	IGG0CLBU	IGGPF4WD
16	IGG0CLCS	IGGPPRDS
	IGG0CLCO	IGGPCRAO
17	IGG0CLCG	IGGPPXIO
	IGG0CLCS	IGGPPRDS
18	$$BCLOSE	--
19	IGG0CLCS	IGGPCRVL
20	IGG0CLCR	IGGPBCRA
21	IGG0CLCR	IGGPCADR
22	IGG0CLCR	IGGPBCRA
23	IGG0CLCR	IGGPBCRA
	IGG0CLAE	IGGPDCBO
	$$BCLOSE	--
	IGG0CLCO	IGGPSCAX
	$$BCLOSE	--
24	IGG0CLCR	IGGPBCRA
	IGG0CLCR	IGGPCRBO
	IGG0CLAG	IGGPGET
	IGG0CLAG	IGGPPDE
25	IGG0CLBU	IGGPF4RD
	IGG0CLBU	IGGPF4WR
	IGG0CLAF	IGGPSKSP

Diagram EI1. DEFINE AIX: Create an alternate index

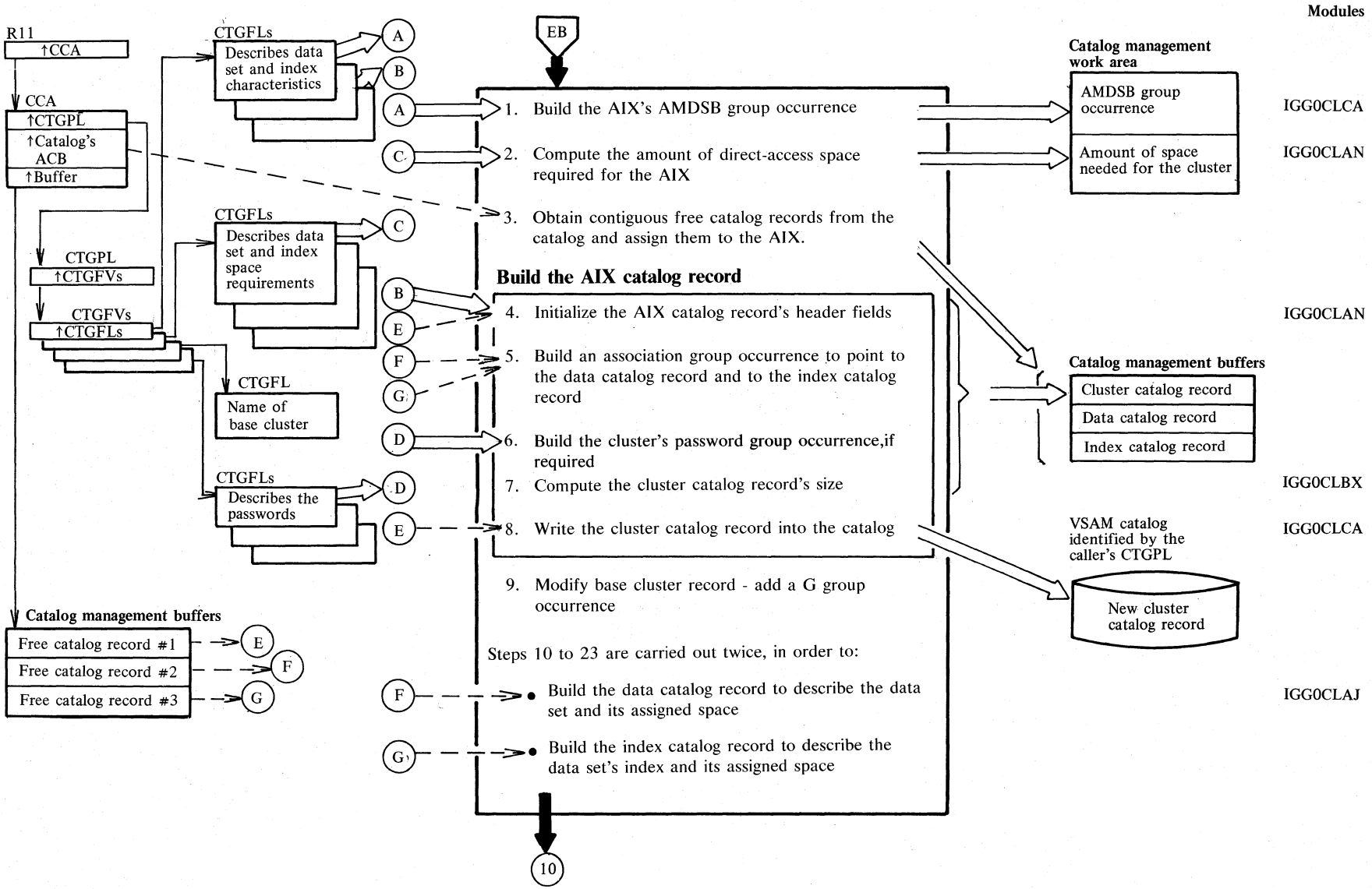
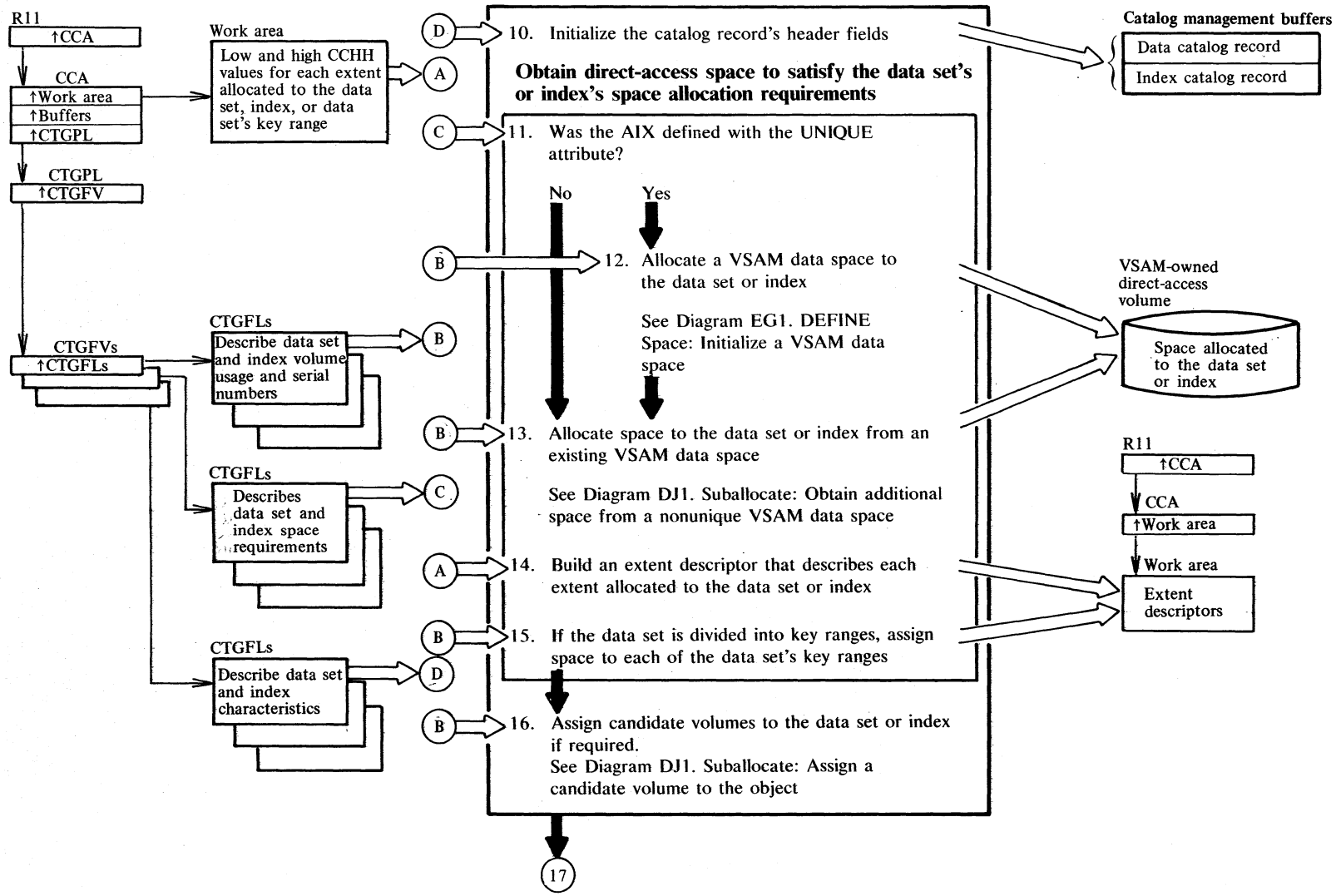


Diagram EI2. DEFINE AIX: Create an alternate index



IGG0CLAJ

Diagram EI3. DEFINE AIX: Create an alternate index

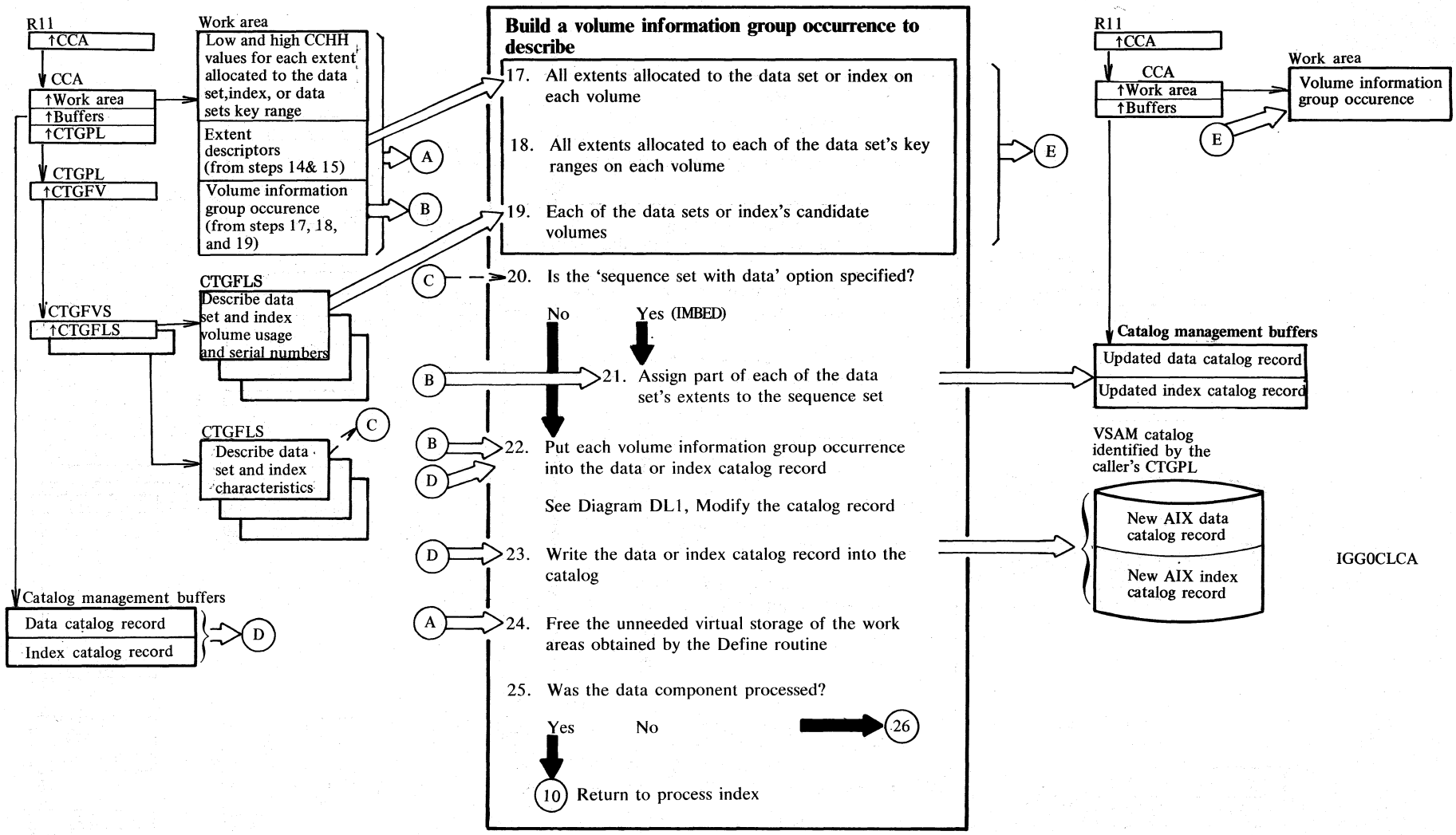


Diagram EI4. DEFINE AIX: Create an alternate index

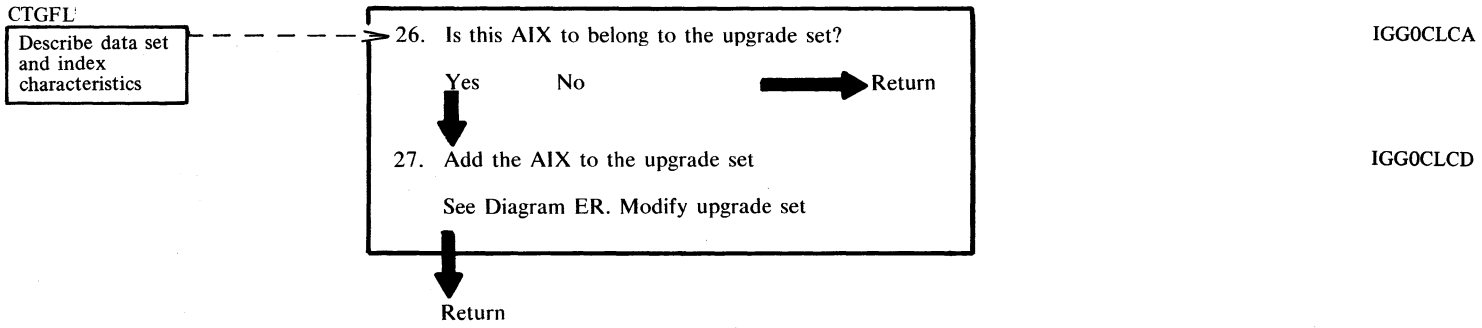


Diagram EJ1. DEFINE Path: Create a VSAM path

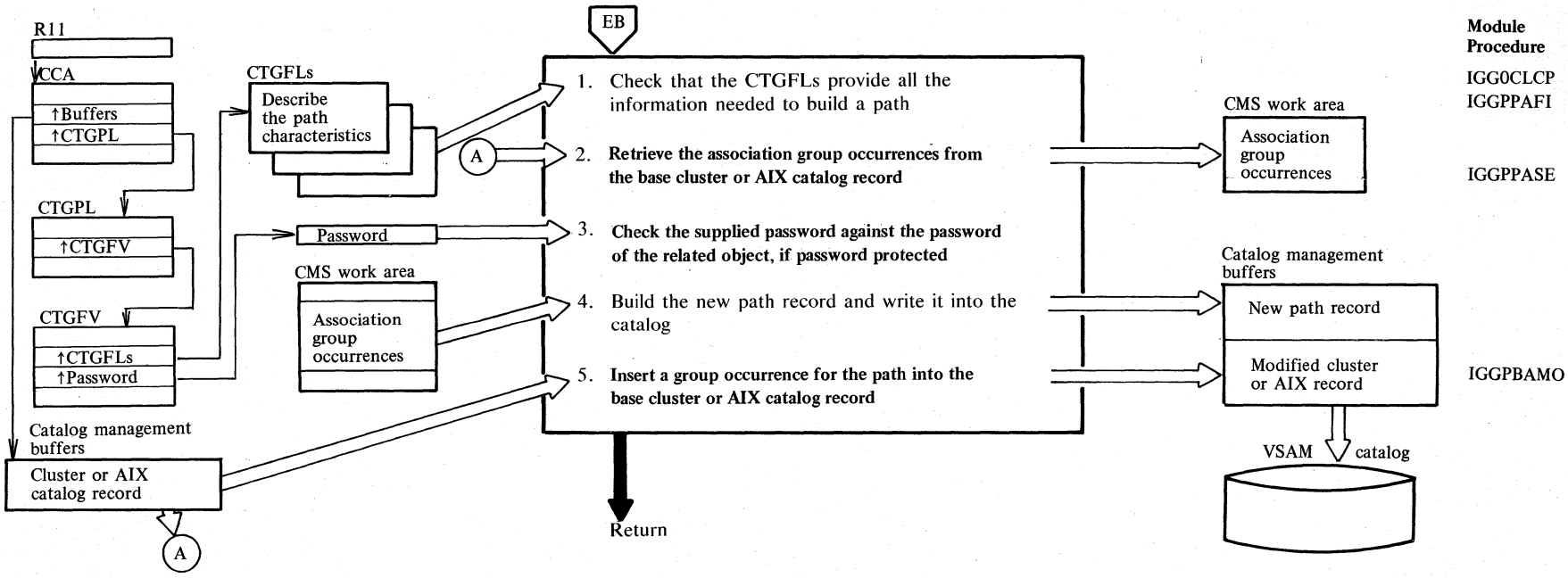


Diagram EK1. ALTER: Modify a catalog record

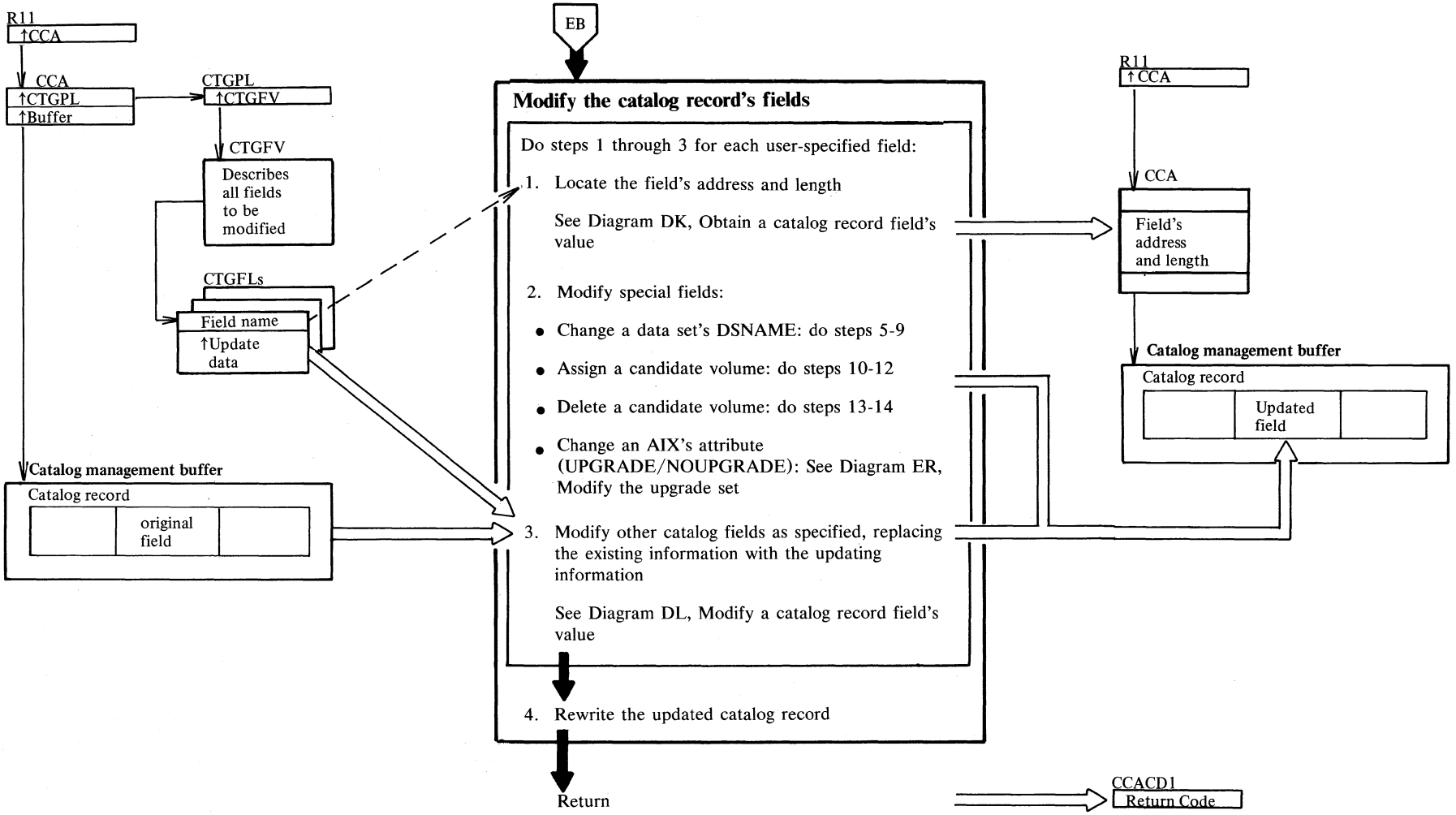


Diagram EK2. ALTER: Modify a catalog record

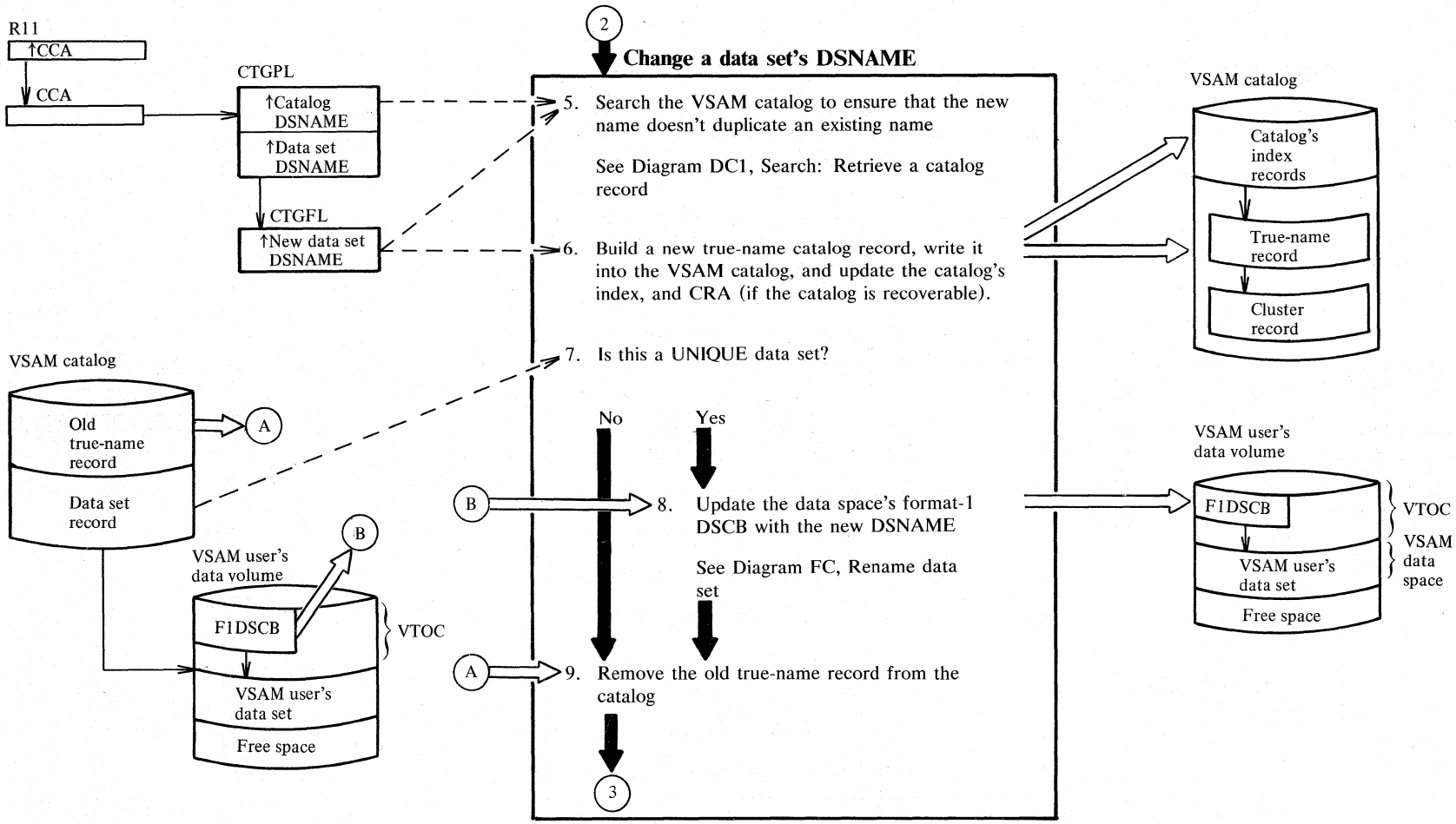
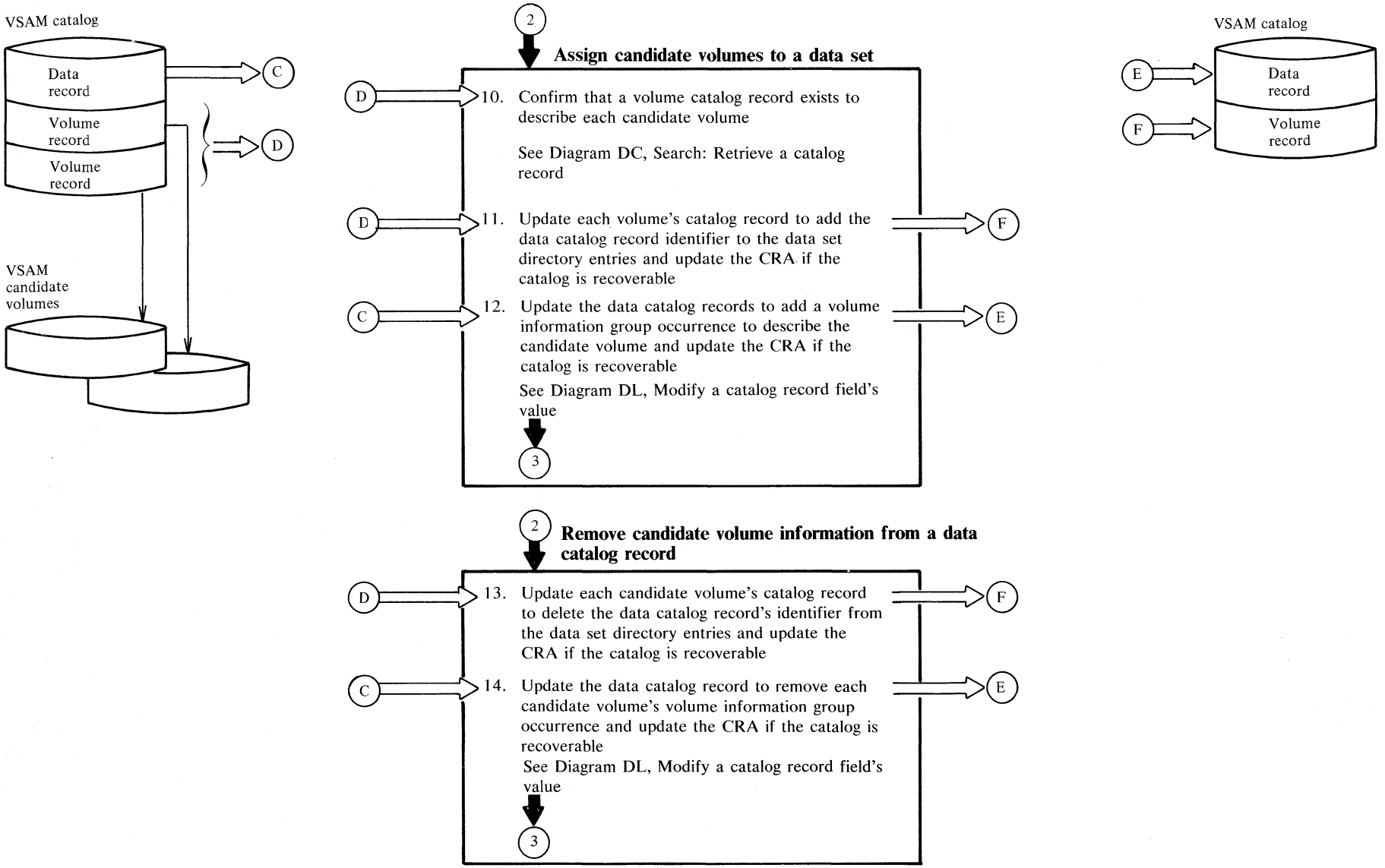


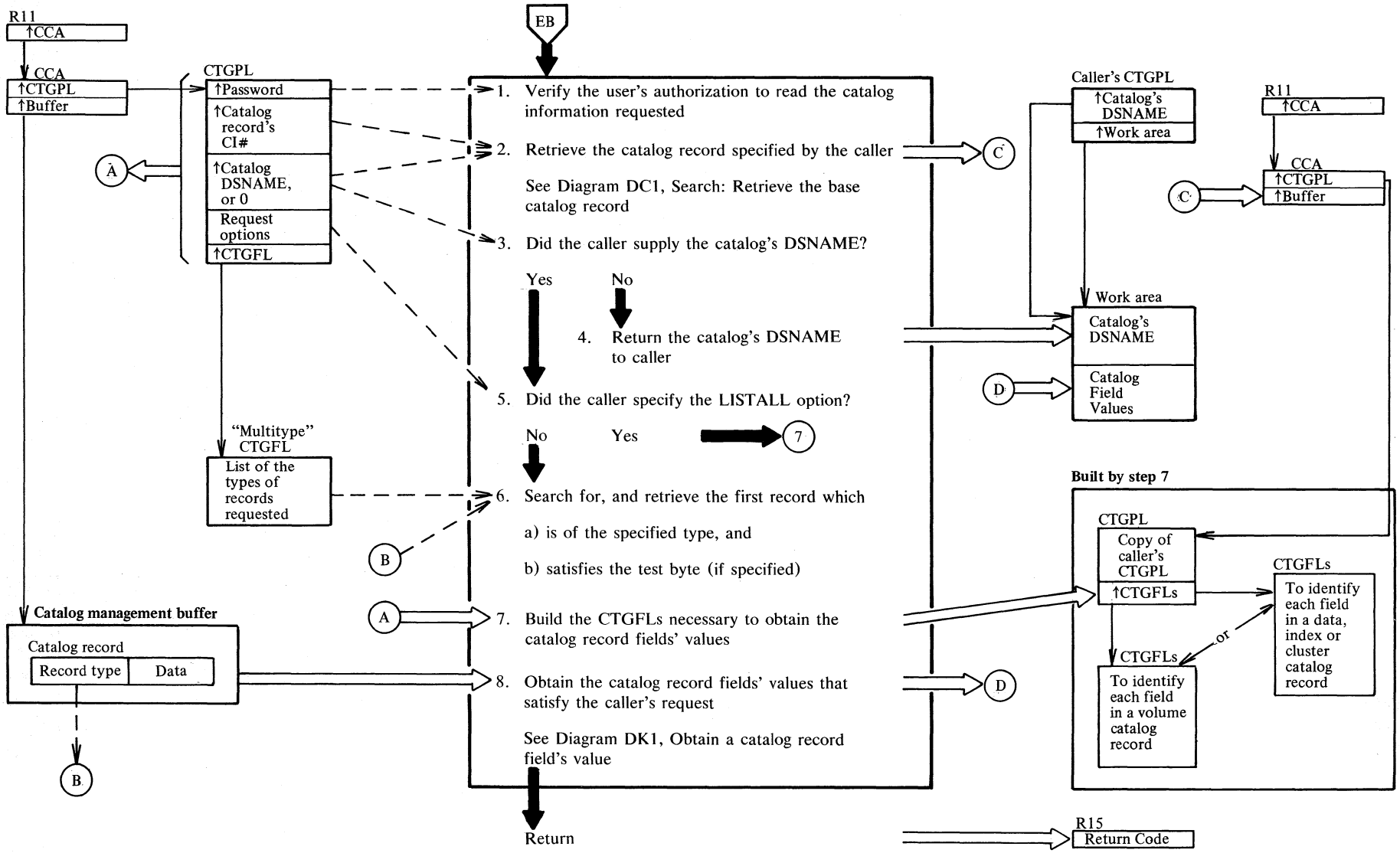
Diagram EK3. ALTER: Modify a catalog record



Notes for Diagram EK

	Description	Module	Procedure
	The ALTER command enables the user to modify some of the information he established when he created a VSAM data set.		
2	When the data set name or allocated candidate volumes are changed, other catalog records besides the data set catalog record must be updated.	IGG0CLBD IGG0CLAG IGG0CLBD IGG0CLAG IGG0CLBD IGG0CLBE IGG0CLBD IGG0CLAV	IGGPALT IGGPGET IGGPALT IGGPPUPC IGGPALT IGGPALVL IGGPALT IGGPALMD IGGPMOD
5	The catalog specified by the ALTER command's CATALOG parameter is examined. The new name to be added must not exist in the catalog.	IGG0CLBD IGG0CLAG IGG0CLBD IGG0CLAG IGG0CLBD IGG0CLAG	IGGPALNM IGGPGET IGGPALNM IGGPPAD IGGPALNM IGGPPDE
6	A new entry is placed in the high key range portion of the catalog.	IGG0CLBD	IGGPALNM
7	A determination must be made as to whether or not the data set is unique.		
8	All volumes that contain a format-1 DSCB must have their names modified in the VTOC label.	IKQREN00	
9	**The name and control-interval number fields in the data set's true-name record are set to 0 and the record identifier field of the catalog record pointed to by the true-name entry is set to C"F".**	IGG0CLBD IGG0CLAZ	IGGPALF1 IGGPALGV IGGPEXT
10	IGGPALVA calls IGGPSALL (IGG0CLAR) to assign the candidate volume to the data set. If a volume catalog record does not exist for the candidate volume, the suballocate routine (IGG0CLAR) returns an error code. See Diagram DJ for details of the suballocate routine.	IGG0CLBD IGG0CLBE IGG0CLAZ IGG0CLBE IGG0CLAR	IGGPALT IGGPALVL IGGPEXT IGGPALVL IGGPALVA IGGPSALL
11	The volume catalog record contains a data set directory that describes each VSAM data set's use of the volume's VSAM space.	IGG0CLBE IGG0CLAZ	IGGPALVA IGGPEXT
12		IGG0CLBE IGG0CLAR IGG0CLBE IGG0CLAV IGG0CLBE IGG0CLAG	IGGPALVA IGGPALSA IGGPSALL IGGPALSA IGGPMOD IGGPALSA IGGPALEC IGGPGET
13		IGG0CLBE IGG0CLBN	IGGPALVL IGGPALVR
14	Note: If the catalog is recoverable, the call to IGG0CLAG will result in a CRA update unless bit CCARPUT (the update inhibit bit) is set to '1'. This bit is set in the calling procedures, which decide whether CRA updating is necessary for a given operation.	IGG0CLBN IGG0CLAV IGG0CLBN IGG0CLAZ IGG0CLBN IGG0CLAG IGG0CLBN IGG0CLAV IGG0CLBN IGG0CLAG IGG0CLBN IGG0CLAV IGG0CLBN IGG0CLBE	IGGPALVR IGGPMOD IGGPALVR IGGPEXT IGGPALVR IGGPALVE IGGPGET IGGPALVE IGGPMOD IGGPALVE IGGPPUPC IGGPALVE IGGPALVR IGGPMOD IGGPALVR IGGPALEC

Diagram EL1. LISTCAT: Retrieve a catalog record's contents



Notes for Diagram EL

	Description	Module	Procedure
	The LISTCAT command allows the user to list all or a part of a VSAM catalog's contents. This routine, however, can return only information from one record (including extension records, if any are present) each time it is called. It is AMS's responsibility to specify the starting point for the search operation, so that records which have already been processed are not retrieved again. This applies regardless of whether LISTALL has been specified or only certain types of records are to be handled.		
1		IGG0CLBQ IGG0CLBM	IGGPLSTC IGGPCKAU
2	The caller can request the information contained in a specific catalog record by providing the record's DSNAME (for a cluster, nonVSAM data set, or catalog) or volume serial number (for a volume). **See DOS/VS: Access Method Services User's Guide, GC33-5382, for details about the ENTRIES parameter.** The true-name catalog records contain the DSNAME of each cluster and nonVSAM data set described in the catalog. Each true-name record also contains the control-interval number of its associated catalog record.	IGG0CLBQ	IGGPLSTC
6		IGG0CLBQ IGG0CLAG	IGGPLSTC IGGPGET
8		IGG0CLBQ IGG0CLAZ	IGGPLSTC IGGPEXT

Diagram EM1. DELETE: Remove a VSAM or non-VSAM data set

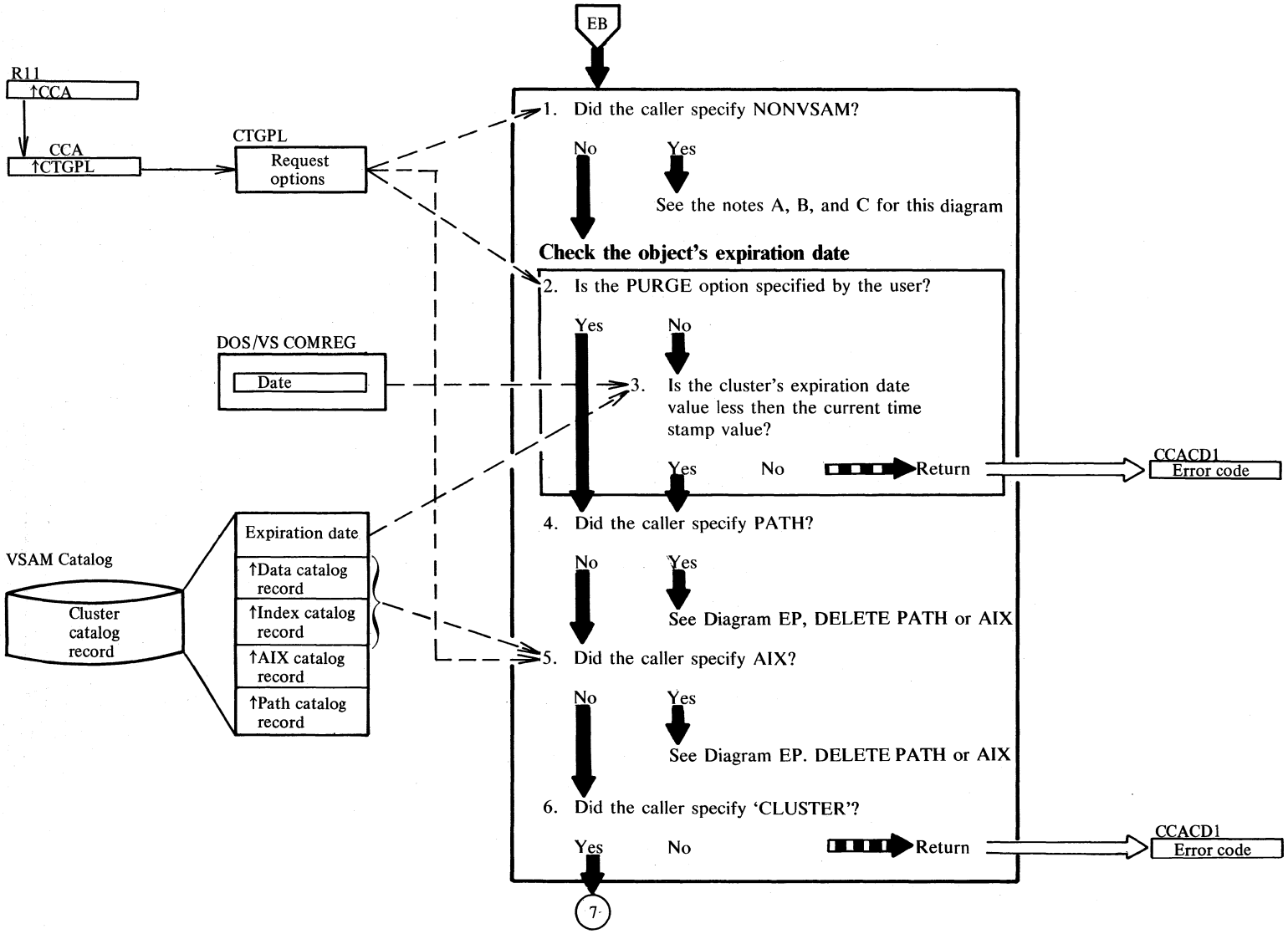


Diagram EM2. DELETE: Remove a VSAM or non-VSAM data set

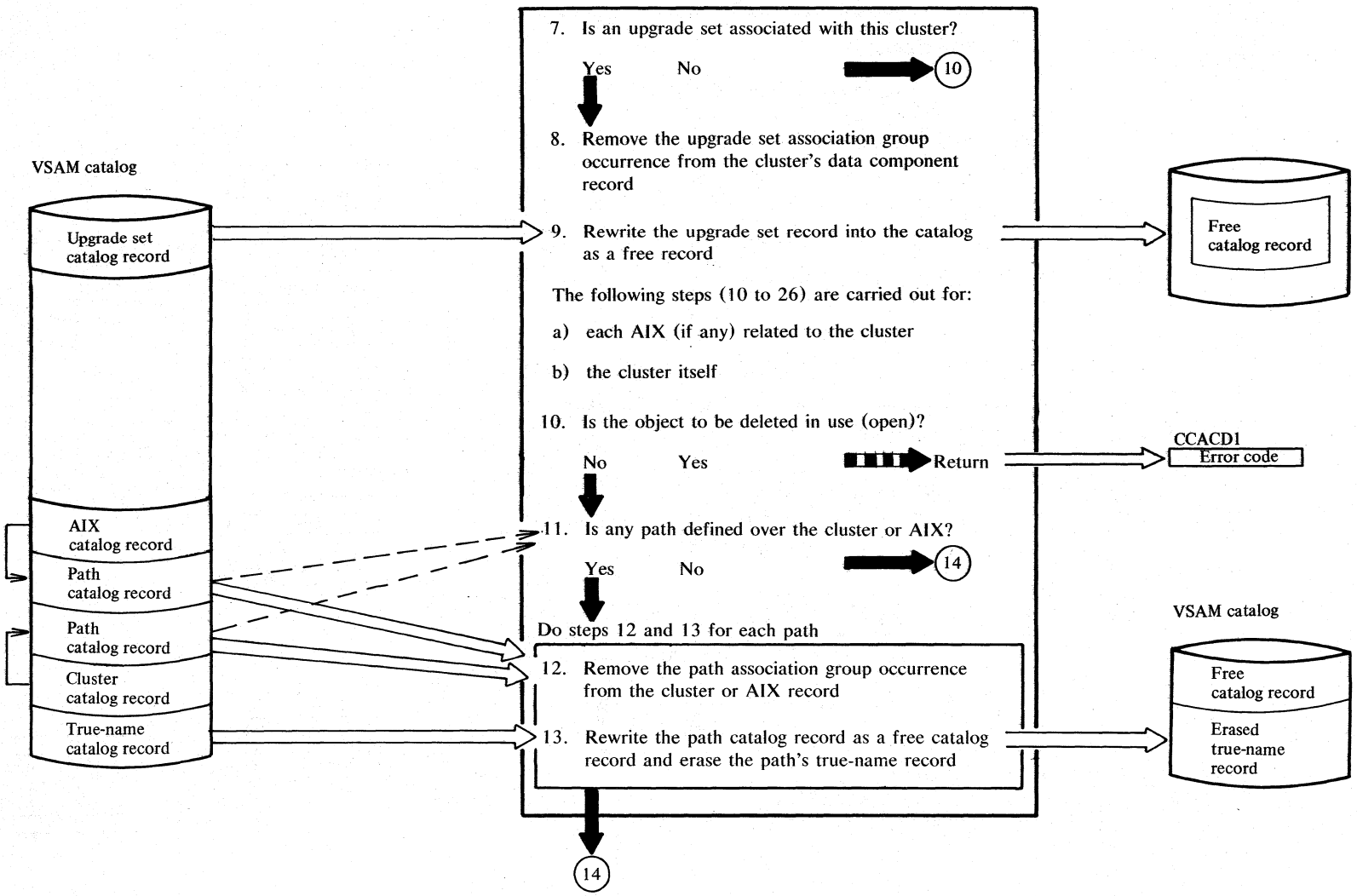


Diagram EM3. DELETE: Remove a VSAM or non-VSAM data set

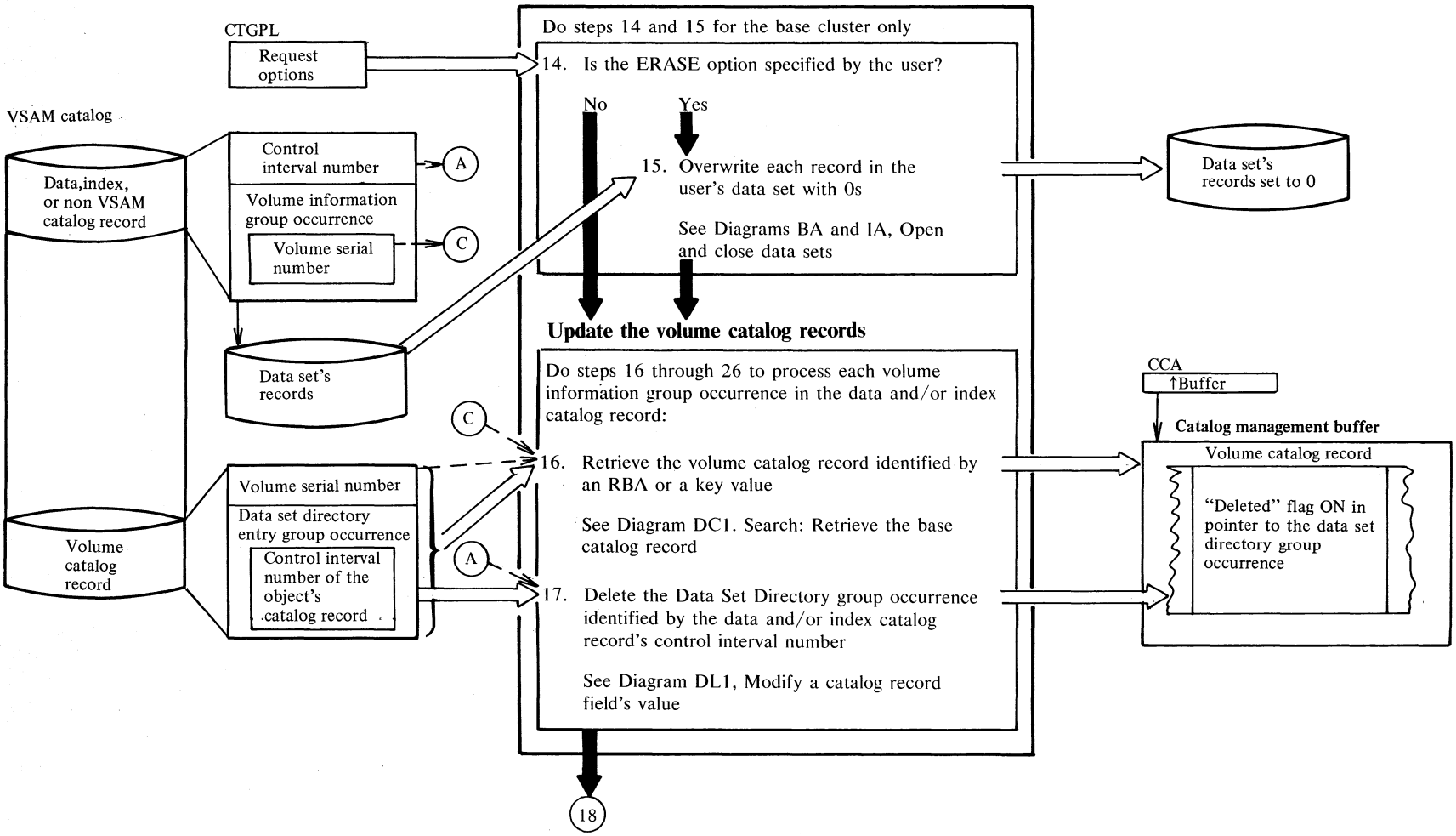


Diagram EM4. DELETE: Remove a VSAM or non-VSAM data set

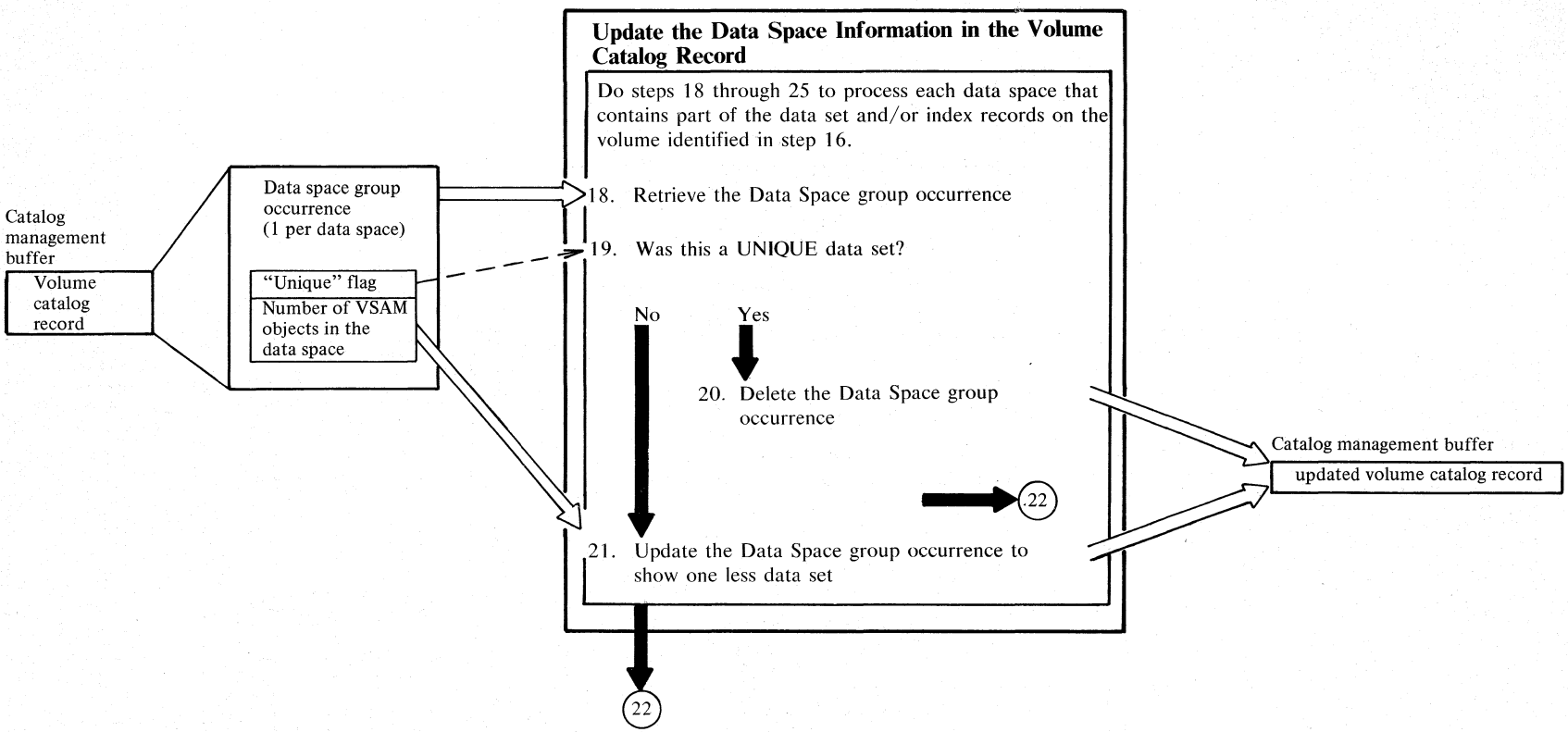


Diagram EM5. DELETE: Remove a VSAM or non-VSAM data set

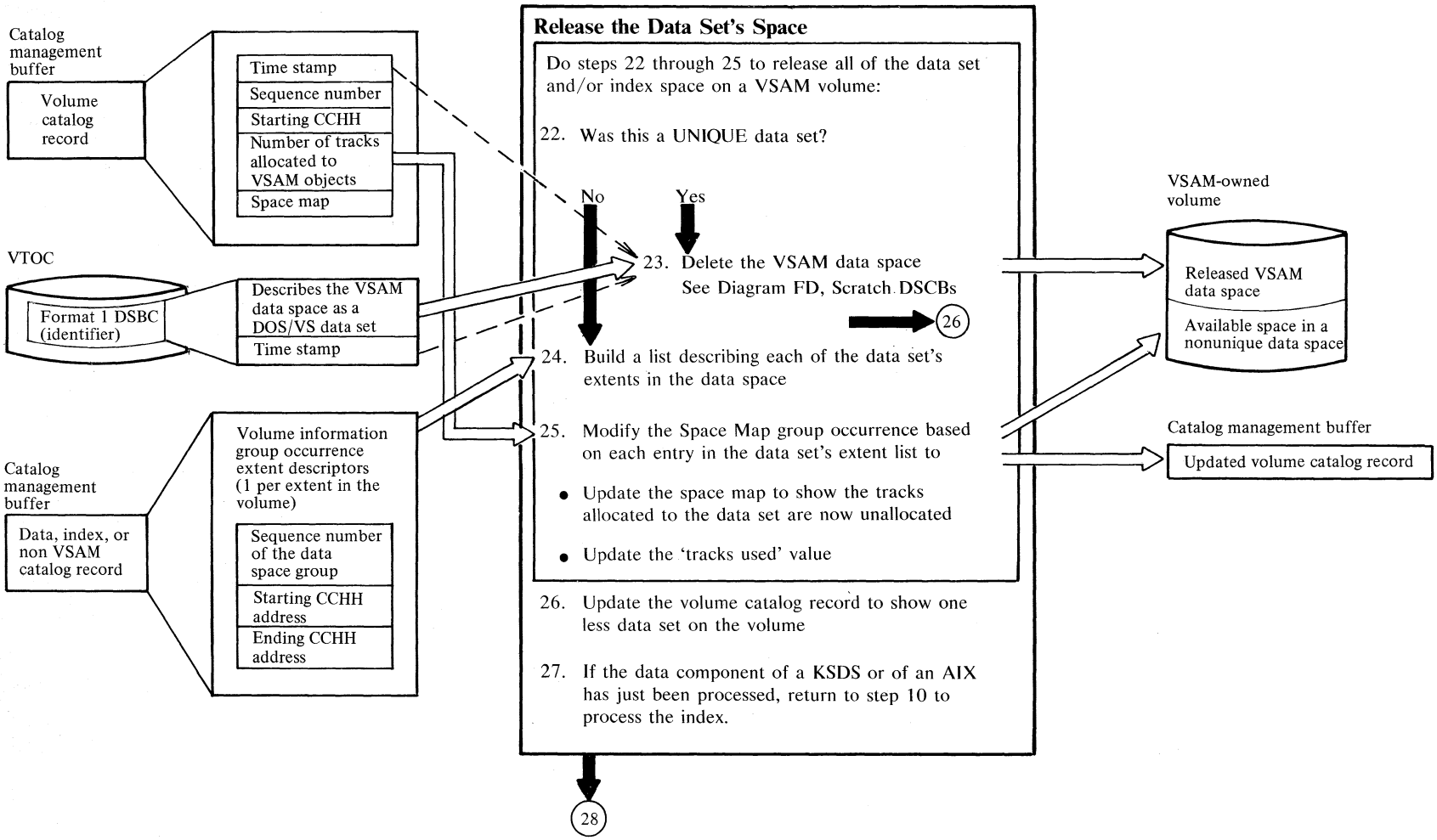
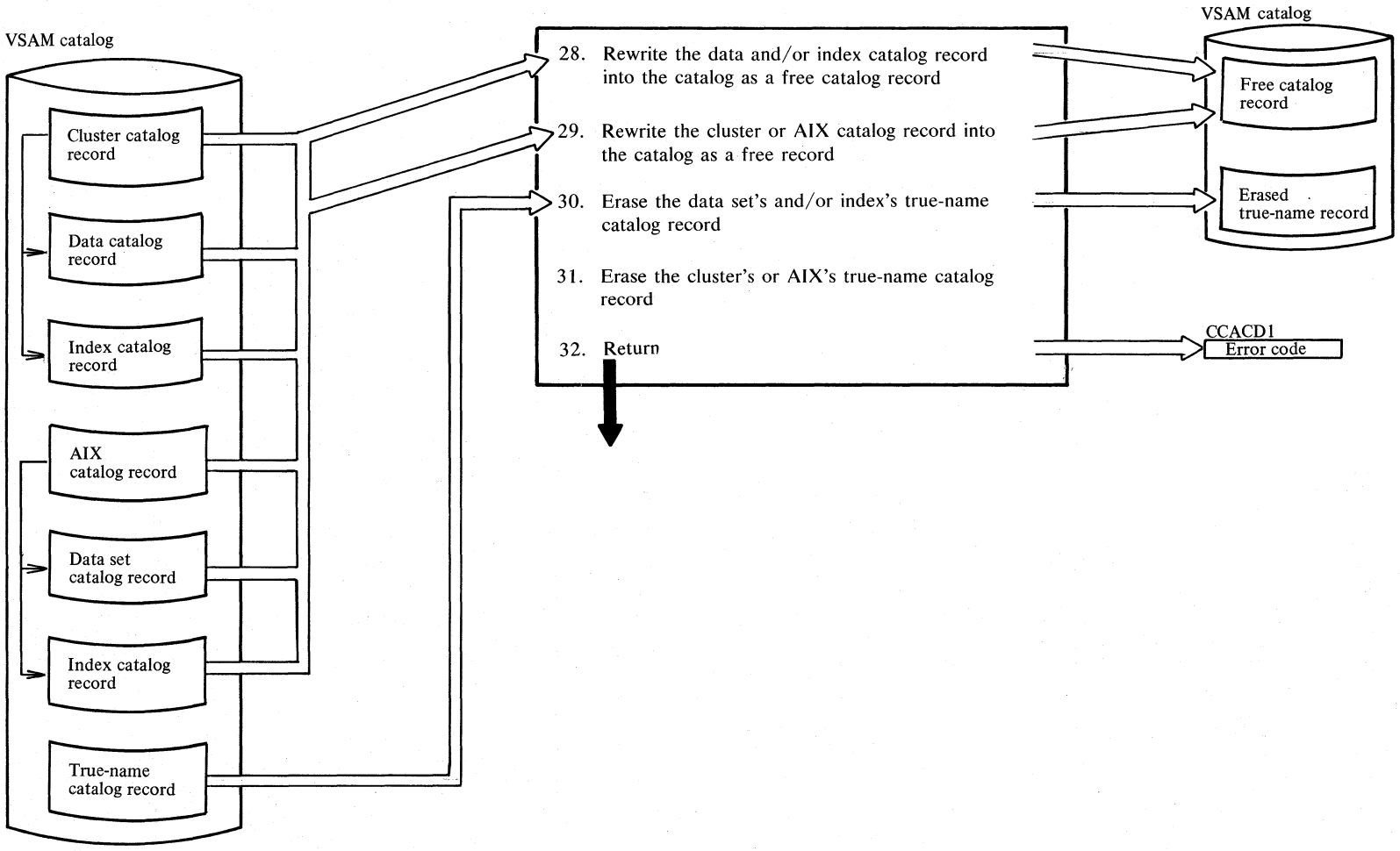


Diagram EM6. DELETE: Remove a VSAM or non-VSAM data set



Notes for Diagram EM (Part 1 of 4)

	Description	Module	Procedure
	The DELETE command enables the user to remove from the catalog all information about a specified cluster or nonVSAM data set.		
1	The catalog record identifier is examined to determine the record type and verify that the contents of the parameter list field CTGTYPE are correct. For processing of a nonVSAM delete (CTGTYPE = A), see the additional notes A,B, and C following these Notes.	IGG0CLBG	IGGPDEL
2	If the user specified PURGE, the data set's expiration date is ignored. See DOS/VS Utilities: Access Method Services, GC33-5382, for details about the RETAIN and PURGE parameters.	IGG0CLBG	IGGPDEL
3	If the user who created the data set specified an expiration date, the data set cannot be deleted until after that date (unless the PURGE parameter is specified: see step 2).	IGG0CLBG	IGGPDEL
4	If the user wants to delete a path, control is transferred to the delete path driver.	IGG0CLBG IGG0CLCX IGG0CLBG	IGGPDEL IGGPDELX IGGPDEL
5	If the user wants to delete an alternate index, control is transferred to the delete AIX driver.	IGG0CLBG IGG0CLCX IGG0CLBG	IGGPDEL IGGPDELX IGGPDEL
6	If the user wants to delete a cluster, control is transferred to the delete cluster driver.	IGG0CLBG	IGGPDEL IGGPDLCL IGGPDEL
7	The upgrade set is retrieved via the cluster's data record containing the upgrade set (Y) association. In order to avoid having to update the Y record each time an AIX related to this cluster is deleted, the Y record is deleted at this point.	IGG0CLBG IGG0CLCX IGG0CLBG	IGGPDLCL IGGPDELY IGGPDLCL
8		IGG0CLCX IGG0CLAV IGG0CLCX	IGGPDELY IGGPMOD IGGPDELY
9	Note: If the catalog is recoverable, the call to IGG0CLAG will result in a CRA update unless bit CCARPUT (the update inhibit bit) is set to '1'. This bit is set in the calling procedures, which decide whether CRA updating is necessary for a given operation.	IGG0CLCX IGG0CLAG IGG0CLCX	IGGPDELY IGGPPDE IGGPDELY

Notes for Diagram EM (Part 2 of 4)

	Description	for the cluster itself		for each AIX related to the cluster:	
		Module	**Procedure**	**Module**	**Procedure**
10	A delete request is rejected if the data set is open. The information indicating that a data set is open is kept in the Look-Aside-Table.	IGG0CLBG	IGGPDLCL	IGG0CLCX	IGGPDAIX
		IGG0CLCX	IGGPDELO	IGG0CLCX	IGGPDELO
		IKQLASMD		IKQLASMD	
		IGG0CLCX	IGGPDELO	IGG0CLCX	IGGPDELO
		IGG0CLBG	IGGPDLCL	IGG0CLCX	IGGPDAIX
12		IGG0CLBG	IGGPDLCL	IGG0CLCX	IGGPDAIX
13	Note: If the catalog is recoverable, the call to IGG0CLAG will result in a CRA update unless bit CCARPUT (the update inhibit bit) is set to '1'. This bit is set in the calling procedures, which decide whether CRA updating is necessary for a given operation.	IGG0CLAV	IGGPMOD	IGG0CLAV	IGGPMOD
		IGG0CLBG	IGGPDLCL	IGG0CLCX	IGGPDAIX
		IGG0CLBG	IGGPDLCL	IGG0CLCX	IGGPDAIX
		IGG0CLAG	IGGPPDE	IGG0CLAG	IGGPPDE
		IGG0CLBG	IGGPDLCL	IGG0CLCX	IGGPDAIX

Notes for Diagram EM (Part 3 of 4)

	Description	Module	Procedure
14	Each of the cluster data set's records is sequentially retrieved and overwritten with 0s.	IGG0CLBG	IGGPDLDS
			IGGPERAS
		$$BOPEN	
		IGG0CLBG	IGGPERAS
		$$BCLOSE	
		IGG0CLBG	IGGPERAS
			IGGPDLDS
16		IGG0CLBG	IGGPDLDS
		IGG0CLAG	IGGPGET
		IGG0CLBG	IGGPDLDS
17	The volume catalog record also contains a data set directory group occurrence to describe each VSAM data set that is contained, partially or completely, on the volume. If the volume is a candidate volume for a data set or index, the data set or index is also described by a data set directory group occurrence.	IGG0CLBG	IGGPDLDS
		IGG0CLA7	IGGPVMSC
			IGGPDEMV
		IGG0CLAZ	IGGPEXT
		IGG0CLA7	IGGPDEMV
			IGGPVMSC
			IGGPDEDD
		IGG0CLAZ	IGGPEXT
		IGG0CLA7	IGGPDEDD
		IGG0CLAV	IGGPMOD
		IGG0CLA7	IGGPDEDD
			IGGPVMSC
18-21	The volume catalog record contains a data space group occurrence to describe each VSAM data space on the volume.	IGG0CLA7	IGGPVMSC
			IGGPDESH
		IGG0CLAZ	IGGPEXT
		IGG0CLA7	IGGPDESH
		IGG0CLAV	IGGPMOD
		IGG0CLA7	IGGPDESH
			IGGPVMSC

	Description	Module	Procedure
23	If the data or index space is unique, its data space is also deleted. Before the data space is deleted, the volume containing it is mounted. The volume containing the data space is specified by the FILE parameter. The extents in the data space's format-1 DSCB and format-3 DSCB, if present, are scratched from the VTOC.	IGG0CLA7	IGGPVMSC
			IGGPDVMV
		$$BOVS01	
		IGG0CLA7	IGGPDVMV
			IGGPVMSC
			IGGPDUSC
		IKQSCR00	
		IGG0CLA7	IGGPDUSC
			IGGPVMSC
24	Each entry in the list identifies one of the data set or index extents in one of the data spaces on the volume.	IGG0CLA7	IGGPVMSC
			IGGPDEVG
		IGG0CLAZ	IGGPEXT
		IGG0CLA7	IGGPDEVG
			IGGPVMSC
25	Each of the data space's extents is described in the data space group occurrence.	IGG0CLA7	IGGPVMSC
		IGG0CLBF	IGGPSSCR
			IGGPESDG
		IGG0CLAZ	IGGPEXT
		IGG0CLBF	IGGPSSCR
			IGGPSCAN
			IGGPMRLC
		IGG0CLAV	IGGPMOD
		IGG0CLBF	IGGPSSCR
		IGG0CLA7	IGGPVMSC
27		IGG0CLBG	IGGPDLDS

Notes for Diagram EM (Part 4 of 4)

	Description	Module	Procedure
28-31	The delete routines erase the data set's truename record and delete all references to the data set's DSNAME in the catalog's index.	IGG0CLBG IGG0CLAN IGG0CLAG IGG0CLAN IGG0CLBG	IGGPDLDS IGGPDUND IGGPDEUN IGGPGET IGGPDDE IGGPDEUN IGGPDUND IGGPDLDS

Notes for non-VSAM DELETE:

	Description	(for nonVSAM) Module	Procedure
A	For CTGTYPE = A, control is transferred to the delete Alien driver.	IGG0CLBG	IGGPDEL IGGPDELA IGGPDEL
B	The VTOC of the volume in which the nonVSAM resides is updated only if 'scratch' is specified. (The volume information is extracted from the 'A' record. A check is done if the volume is mounted and the DSCB is scratched from the VTOC).	IGG0CLBG IGG0CLA7 IGG0CLAZ IGG0CLA7 IGG0CLBG IGG0CLA7 $$BOVS01 IGG0CLA7 IGG0CLBG IGG0CLA7 IKQSCR00 IGG0CLA7	IGGPDELA IGGPDEMV IGGPEXT IGGDEMV IGGPDELA IGGPDVMV IGGPDVMV IGGPDELA IGGPDUSC IGGPDUSC
C	The 'A' record is deleted from the catalog Note: If the catalog is recoverable, the call to IGG0CLAG will result in a CRA update unless bit CCARPUT (the update inhibit bit) is set to '1'. This bit is set in the calling procedures, which decide whether CRA updating is necessary for a given operation.	IGG0CLBG IGG0CLAN IGG0CLAG IGG0CLAN IGG0CLBG	IGGPDELA IGGPDUND IGGPDDE IGGPDUND IGGPDELA

Diagram EN1. DELETE Space: Release all of the empty VSAM data space on a volume

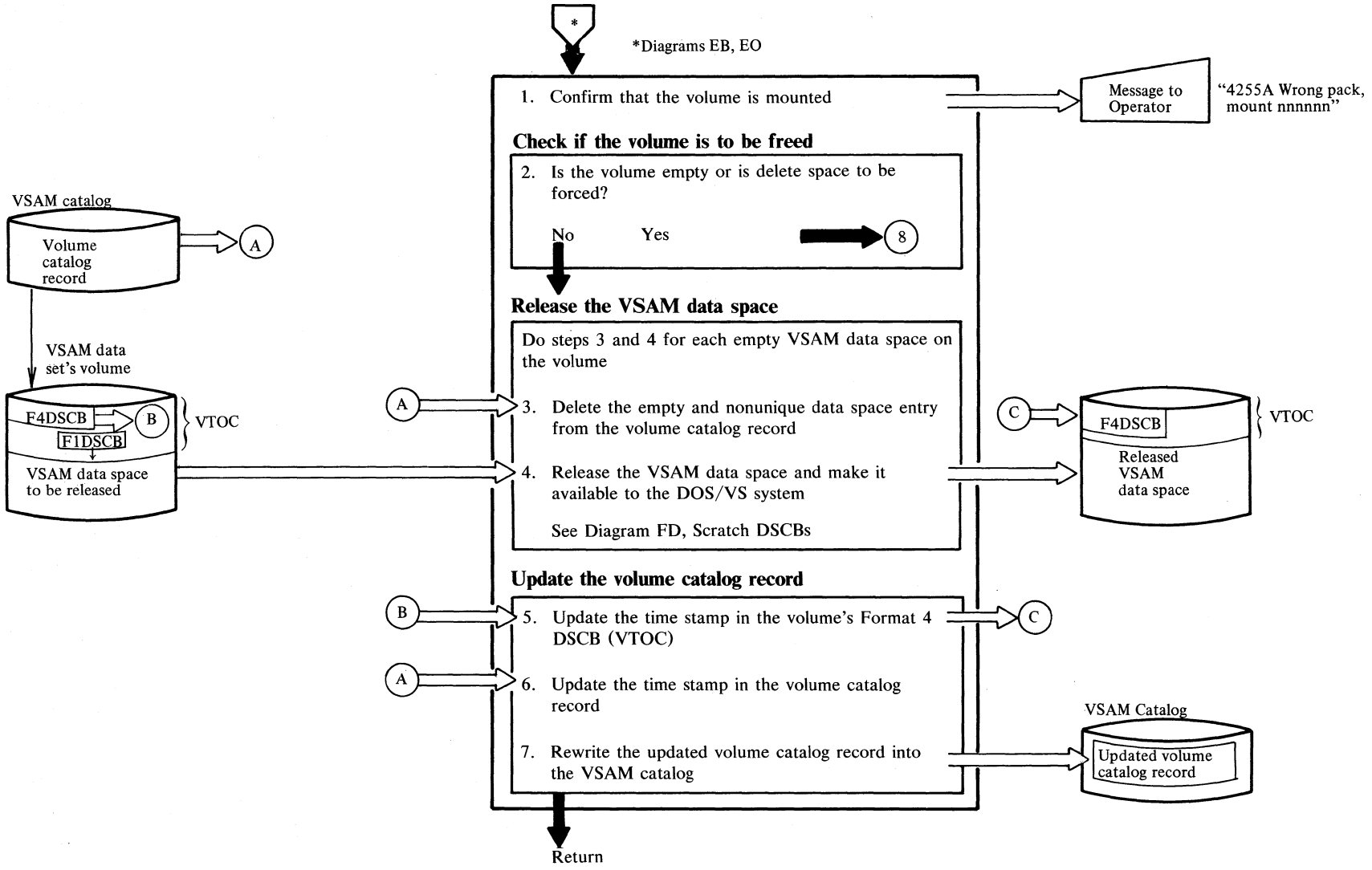


Diagram EN2. DELETE Space: Release all of the empty VSAM data space on a volume

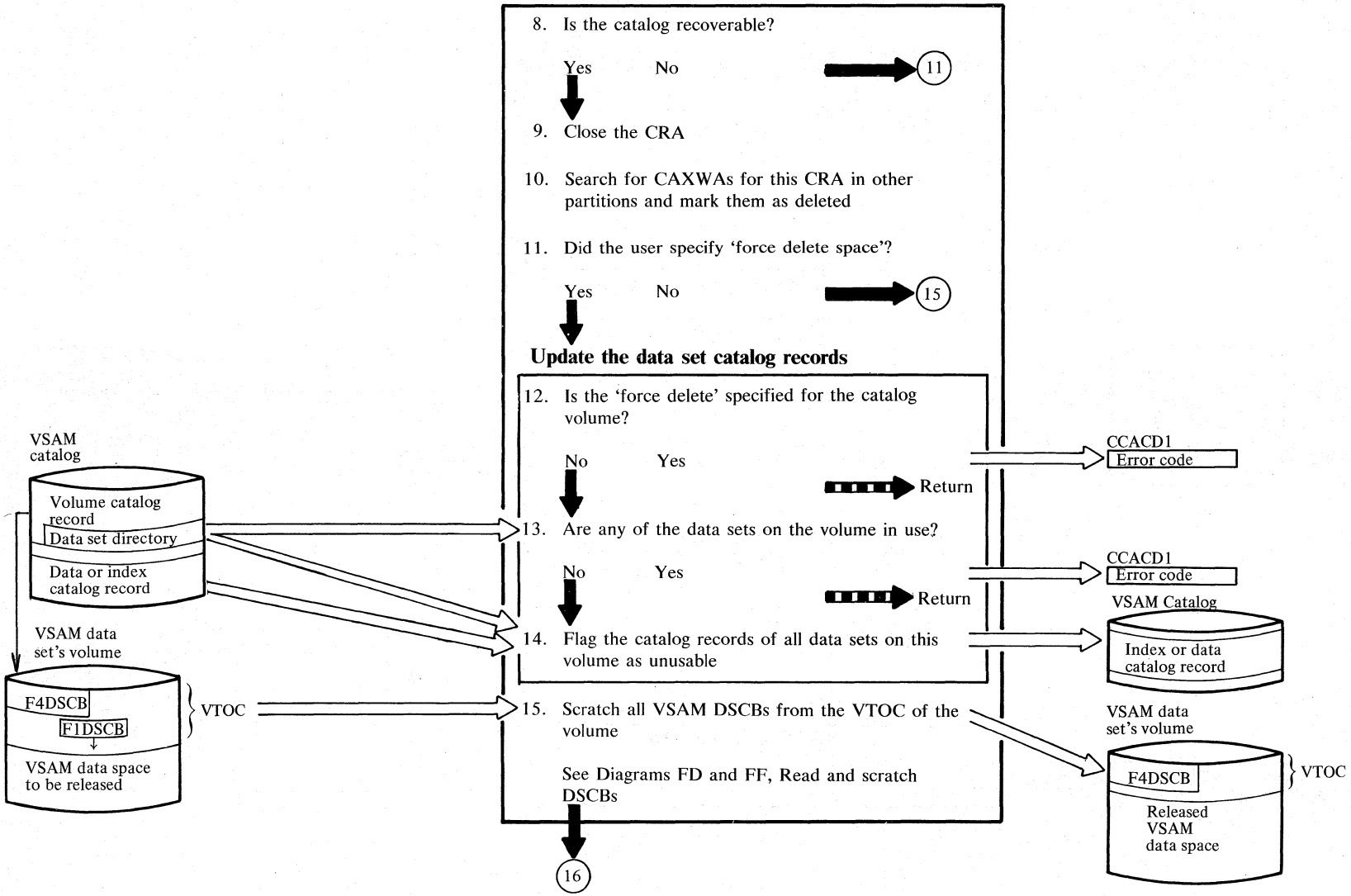
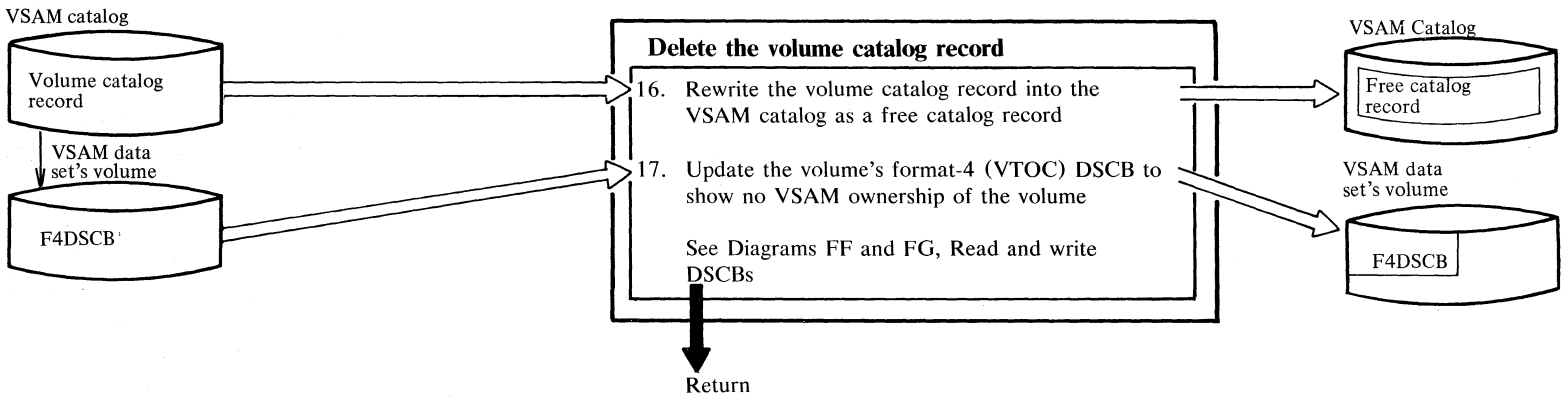


Diagram EN3. DELETE Space: Release all of the empty VSAM data space on a volume



Notes for Diagram EN (Part 1 of 2)

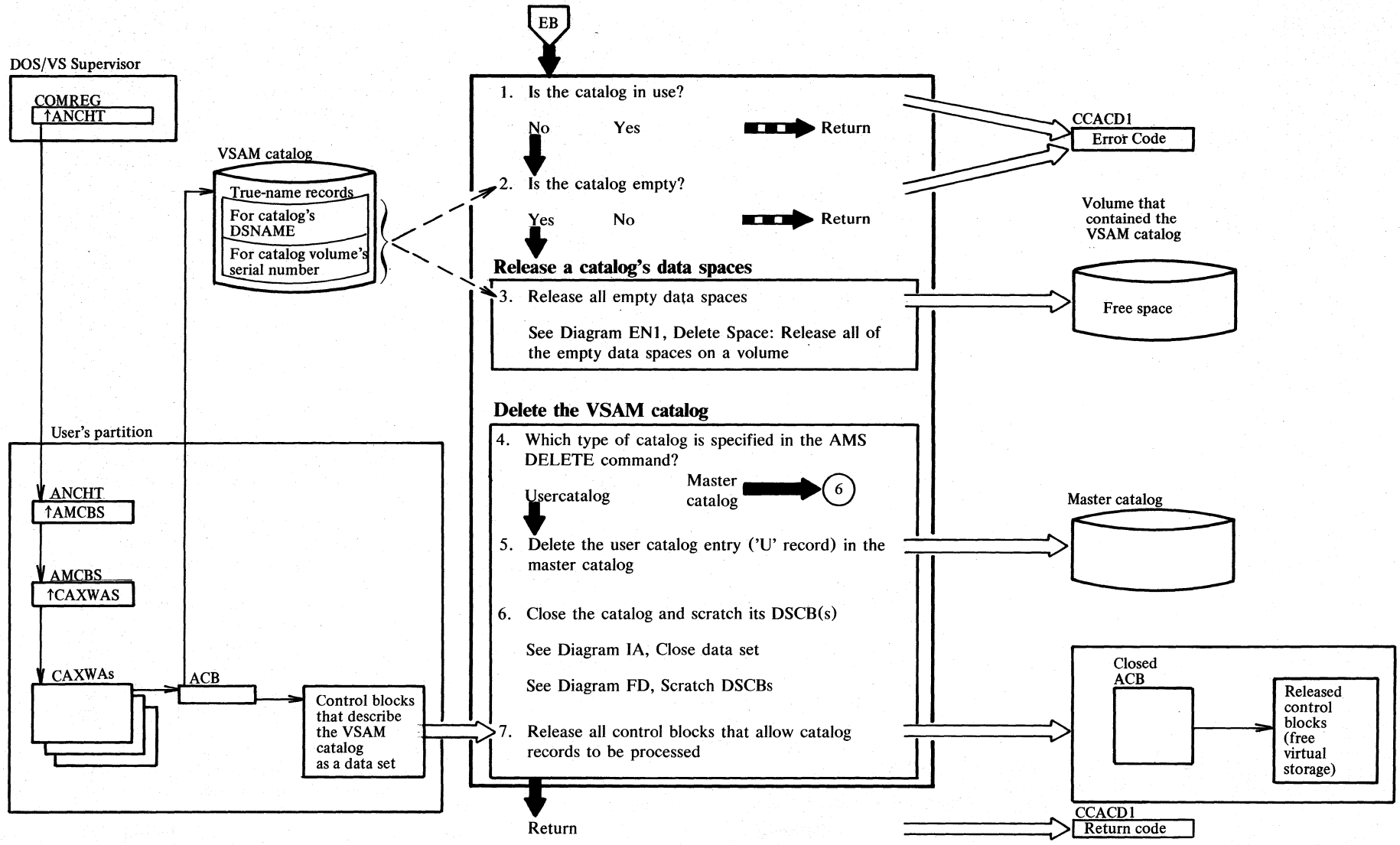
	Description	Module	Procedure
The DELETE space command enables the user to release all VSAM data spaces on a specified volume.			
1	If the volume isn't already mounted and available for use, the DELETE space routine issues the appropriate mount message to the operator.	IGG0CLBL $$BOVS01 IGG0CLBL IKQVDTPE	IGGPDELS IGGPDLVM IGGPDLVM
2	A volume is empty when its volume catalog record contains no data set directory group occurrence (which normally describes data sets on the volume). 'Force' is an option the user may specify to delete a volume although there are still data sets on this volume.	IGG0CLBL IGG0CLAZ IGG0CLBL	IGGPDELS IGGPDLVC IGGPEXT IGGPDLVC IGGPDELS
3	The volume catalog record contains a data space group occurrence to describe each VSAM data space on the volume.	IGG0CLBL IGG0CLAZ IGG0CLBL IGG0CLAV IGG0CLBL IGG0CLAV	IGGPDLCD IGGPEXT IGGPDLSH IGGPMOD IGGPDLSD IGGPMOD
4	The DADSM routine IKQSCR00 releases VSAM data space and makes its space available to other DOS/VS system users	IGG0CLBL IKQSCR00	IGGPDLSC
5	The time stamp in the VTOC is updated to indicate when the last change to the VTOC was made by VSAM	IGG0CLBL IGG0CLBU IGG0CLBL IGG0CLBU IGG0CLBL	IGGPDELS IGGPDLTS IGGPF4RD IGGPDLTS IGGPF4WR IGGPDLTS
6		IGG0CLBL	IGGPDELS

	Description	Module	Partition
7	Note: If the catalog is recoverable, the call to IGG0CLAG will result in a CRA update unless bit CCARPUT (the update inhibit bit) is set to '1'. This bit is set in the calling procedures, which decide whether CRA updating is necessary for a given operation.	IGG0CLBL IGG0CLAG IGG0CLBL	IGGPDELS IGGPPUPC IGGPDELS
8	If a volume owned by a recoverable catalog is to be freed, its CRA will be scratched. Care must thus be taken that no partitions will use this CRA again.	IGG0CLBL	IGGPDELS
9	By closing the CRA, all control blocks in the requesting partition will be freed.	IGG0CLBL IGG0CLCO IGG0CLBL $$BCLOSE IGG0CLBL	IGGPDELS IGGPDLCR IGGPSCAX IGGPDLCR IGGPDLCR
10	The control blocks for the CRA are erased at the end of the current job (which issued the DELETE SPACE command). If, however, a job in another partition issues a DEFINE SPACE for the same volume before the end of the current job, the DEFINE SPACE routine must know that is has to build a new CRA. For this reason, the CRA's control blocks in all other partitions are marked invalid.	IGG0CLBL $$BCLCRA IGG0CLBL IGG0CLBL	IGGPDLCR IGGPDLCR IGGPDELS
13	As long as any of the data sets on the volume is in use, the forced delete space is rejected.	IGG0CLBL IGG0CLCL IGG0CLAZ IGG0CLCL IGG0CLCX IGG0CLCL	IGGPDELS IGGPDLSF IGGPDXDS IGGPEXT IGGPDLSF IGGPDELO IGGPDLSF

Notes for Diagram EN (Part 2 of 2)

	Description	Module	Procedure
14.	The catalog records of all data sets on the volume are flagged as unusable. Since their space will be partly or completely gone, they are thus marked to inhibit any open requests for output, and to warn CMS delete cluster. (CRA duplicates may be gone, volume occurrences may point to non-existing volumes.) Note: If the catalog is recoverable, the call to IGG0CLAG will result in a CRA update unless bit CCARPUT (the update inhibit bit) is set to '1'. This bit is set in the calling procedures, which decide whether CRA updating is necessary for a given operation.	empty volume: force delete: IGG0CLCL IGG0CLAZ IGG0CLCL IGG0CLAG IGG0CLCL IGG0CLAG IGG0CLCL IGG0CLAV IGG0CLCL	empty volume: force delete: IGGPDLSF IGGPDXDS IGGPEXT IGGPDLSF IGGPGET IGGPDLSF IGGPPUPC IGGPDLSF IGGPDMDS IGGPMOD IGGPDLSF
15	The VTOC is scanned for VSAM DSCBs and all VSAM DSCBs are scratched thus returning all space occupied by VSAM to the VTOC.	IGG0CLBL IGG0CLCL IGG0CLCL IKQVTC00 IGG0CLCL IKQRDS00 IGG0CLCL IKQSCR00 IGG0CLCL IKQVTC00 IGG0CLCL IGG0CLCL IGG0CLBL	IGGPDELS IGGPDLS IGGPDVSC IGGPDVSC IGGPDVSC IGGPDVSC IGGPDVSC IGGPDSF IGGPDELS
16	Reset the volume record and all its extension records so that it is available for future assignment.	IGG0CLBL IGG0CLAG IGG0CLBL IGG0CLAG IGG0CLBL	IGGPDELS IGGPDLET IGGPGET IGGPDLET IGGPPDEC IGGPDLET
17	The format-4 DSCB is the first entry in a direct-access volume's VTOC. It contains the volume owner's identification and information on how the volume is used. Note: If the catalog is recoverable, the call to IGG0CLAG will result in a CRA update unless bit CCARPUT (the update inhibit bit) is set to '1'. This bit is set in the calling procedures, which decide whether CRA updating is necessary for a given operation.	IGG0CLBL IGG0CLBU IGG0CLBL IGG0CLBU IGG0CLBL	IGGPDLET IGGPF4RD IGGPDLET IGGPF4WR IGGPDLET IGGPDELS

Diagram EO1. DELETE Catalog: Release a VSAM catalog



Notes for Diagram EO

	Description	Module	Procedure
	The DELETE catalog command enables the user to release a catalog's space and make it available to other DOS/VS system users. The catalog must be empty or the request is rejected.		
1	IKQLASMD is called to determine if the catalog is in use by some other task.	IKQLASMD	
2	If the catalog contains more than two true-name records, it is not empty and cannot be deleted.	IGG0CLAF	IGGPDELC
3	The volume catalog record contains an entry for each VSAM data space allocated on the volume. Each entry contains the data necessary to free the data space.	IGG0CLAF	IGGPSDSP
4	One of these types must be specified for a DELETE CATALOG operation.		
5	Each user catalog has an entry in the master catalog.		

	Description	Module	Procedure
6	The communications region (COMREG) points to the ANCHT which points to the AMCBS. The AMCBS points to the control blocks that describe the VSAM catalog to the DOS/VS system. Note: If the catalog is recoverable, the call to IGG0CLAG will result in a CPA update unless bit CCARPUT (the update inhibit bit) is set to '1'. This bit is set in the calling procedures, which decide whether CRA updating is necessary for a given operation.	IGG0CLAF IGG0CLAL IGG0CLAF IGG0CLAG IGG0CLAF IGG0CLAG IGG0CLAF IKQVDTPE IGG0CLAF IGG0CLAZ IGG0CLAF $$BCLOSE IGG0CLAF IKQSCR00 IGG0CLAF IGG0CLBU IGG0CLAF $$BOVS01 IGG0CLAF IGG0CLBU	IGGPDELC IGGPDBVC IGGPDELC IGGPPDE IGGPDELC IGGPGET IGGPDELC IGGPDELC IGGPEXT IGGPDELC IGGPDELC IGGPSDSP IGGPDELC IGGPF4RD IGGPDELC IGGPDELC IGGPF4WR

Diagram EP1. DELETE AIX or Path: Release an alternate index or a VSAM path

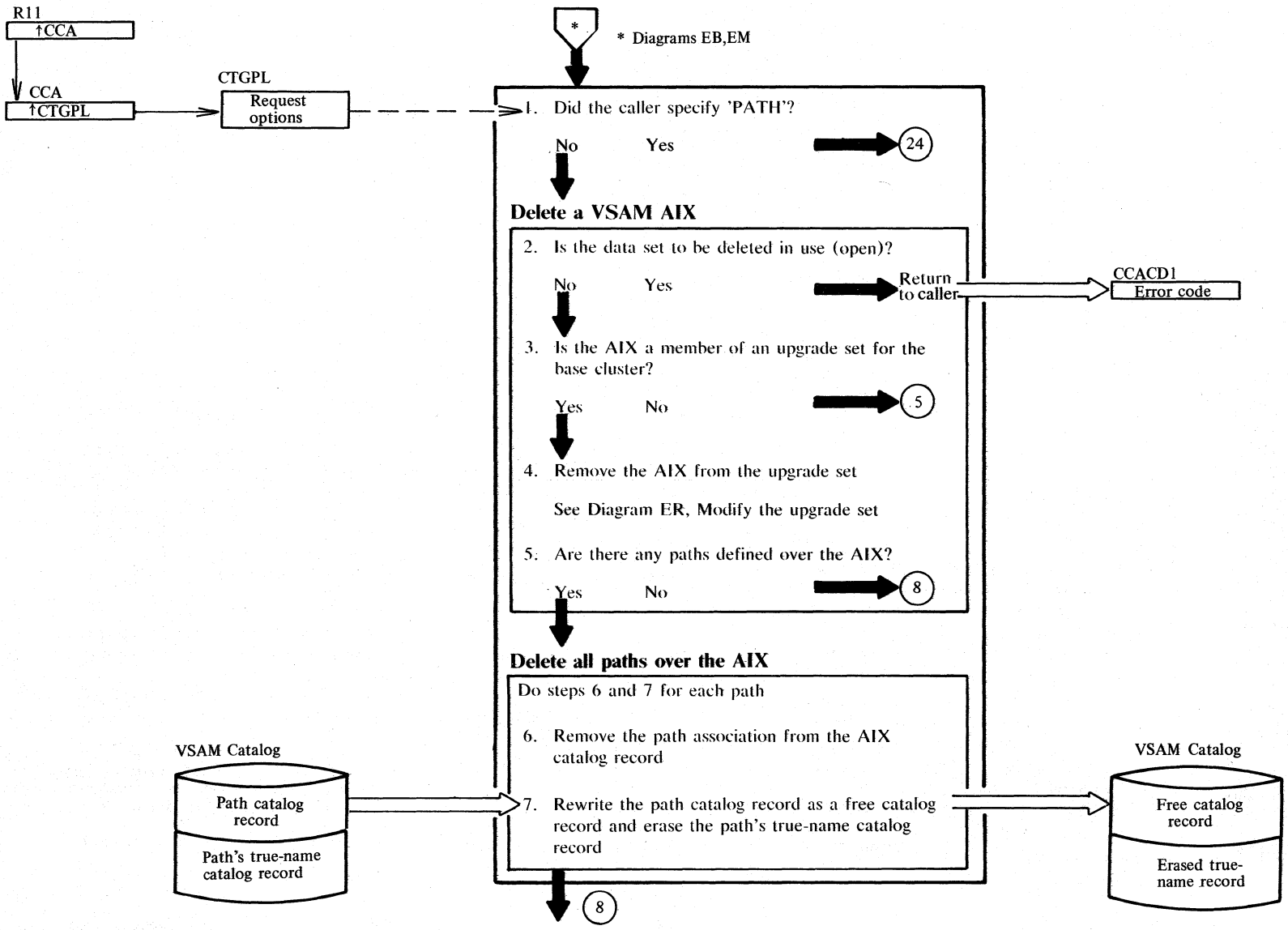


Diagram EP2. DELETE AIX or Path: Release an altenate index or a VSAM path

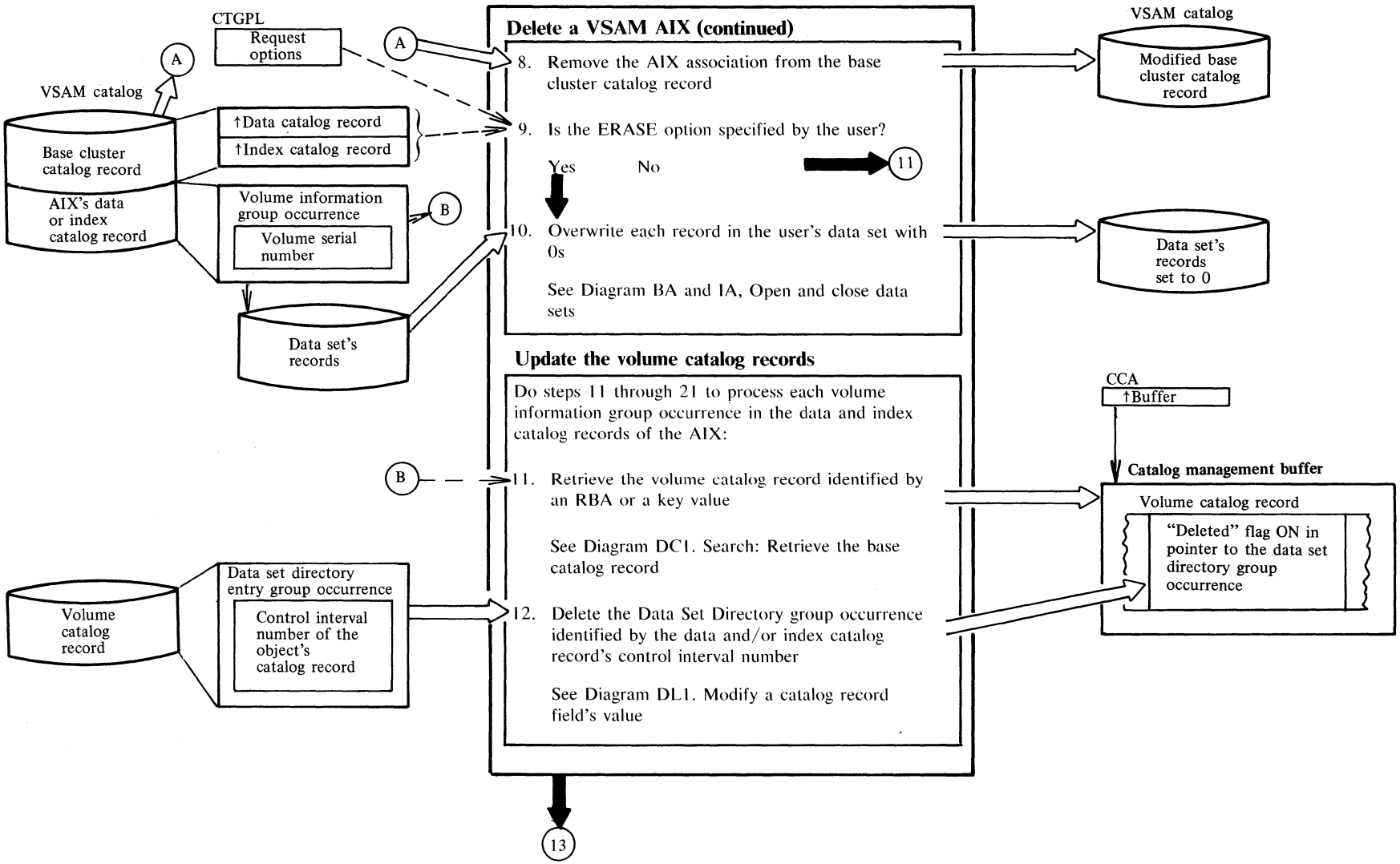


Diagram EP3. DELETE AIX or Path: Release an alternate index or a VSAM path

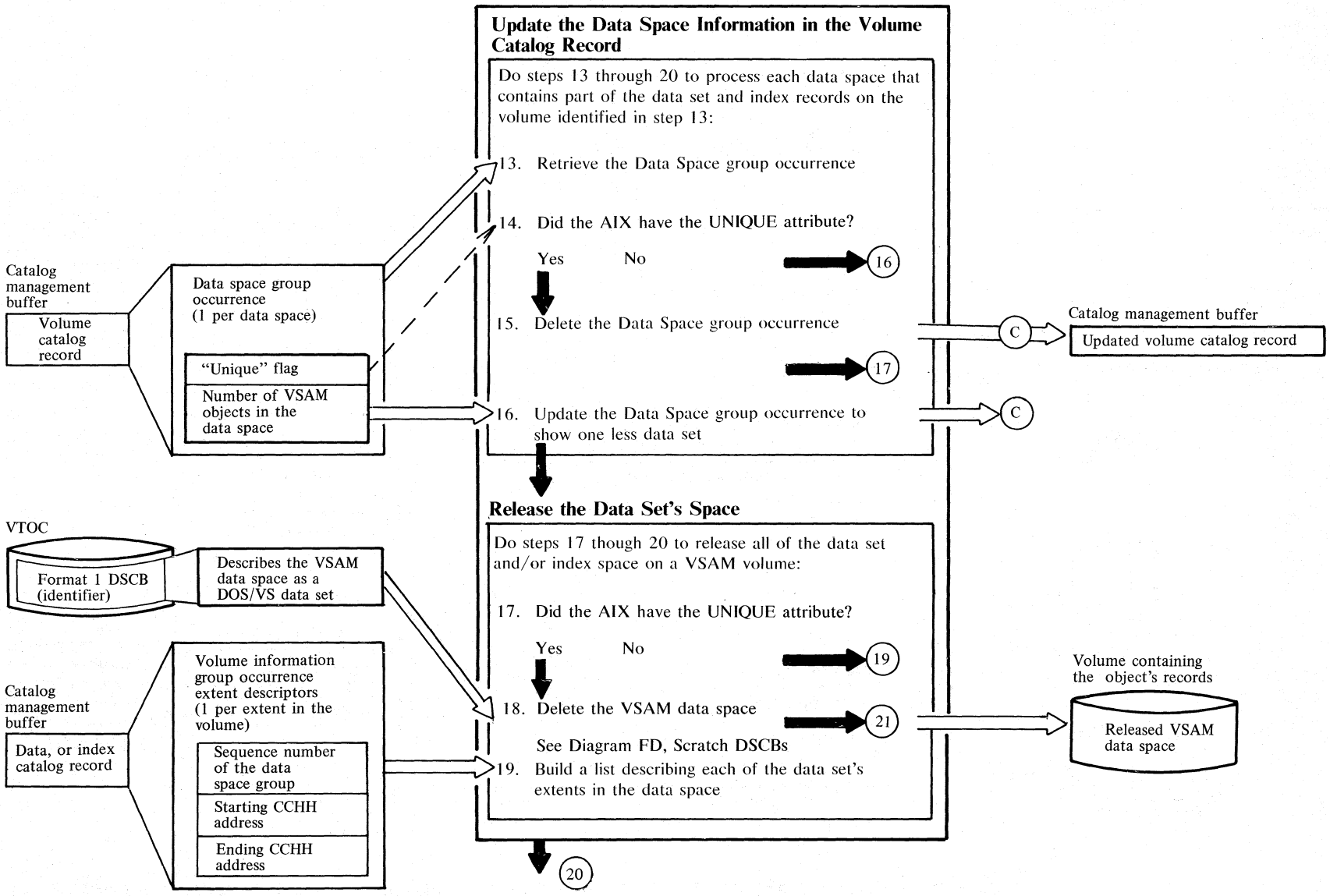
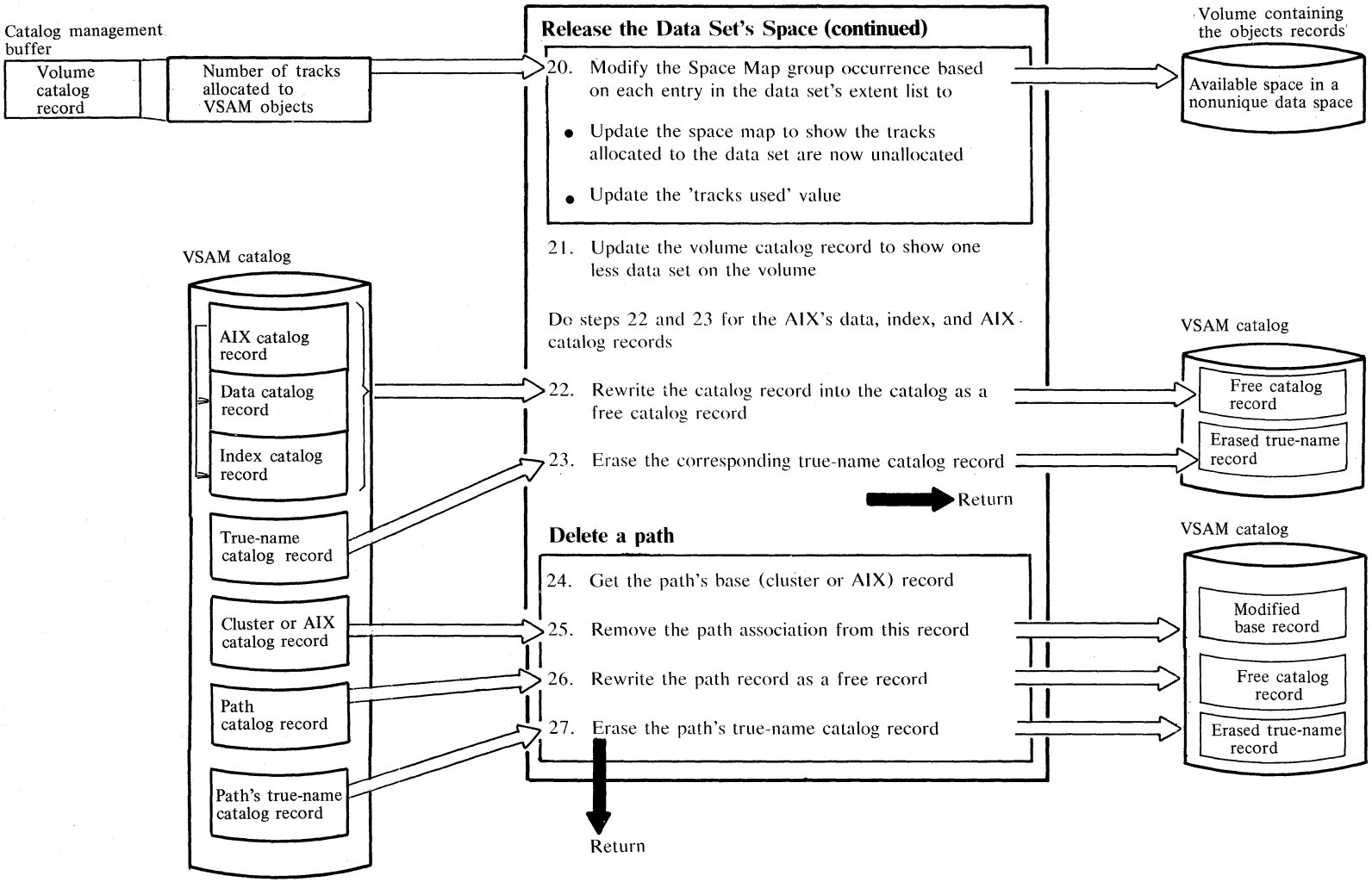


Diagram EP4. DELETE AIX or Path: Release an alternate index or a VSAM path



Notes for Diagram EP (Part 1 of 3)

	Description	Module	Procedure
	The DELETE command enables the user to remove from the catalog all information about a specified AIX or path.		
1	If the user wants to delete a path, control is transferred to the delete path driver	IGG0CLBG IGG0CLCX IGG0CLBG	IGGPDEL IGGPDELP IGGPDEL
2	A delete request is rejected if the data set is open. The information whether a data set is open is kept in the look-aside table.	IGG0CLCX IGG0CLCX IKQLASMD IGG0CLCX IGG0CLCX	IGGPDELX IGGPDELO IGGPDELO IGGPDELX
3-4	The upgrade set is retrieved via the AIX's base cluster data component containing the upgrade set association. In case this AIX is the last member in the upgrade set, the upgrade set and its association are deleted. Note: If the catalog is recoverable, the call to IGG0CLAG will result in a CRA update unless bit CCARPUT (the update inhibit bit) is set to '1'. This bit is set in the calling procedures, which decide whether CRA updating is necessary for a given operation.	IGG0CLCX IGG0CLAG IGG0CLCX IGG0CLAZ IGG0CLCX IGG0CLCD IGG0CLAG IGG0CLCD IGG0CLAZ IGG0CLCD IGG0CLAG IGG0CLCD IGG0CLAV IGG0CLCD IGG0CLAG IGG0CLCD IGG0CLAV	IGGPDELX IGGPGET IGGPDELX IGGPEXT IGGPDELX IGGPUPG IGGPGET IGGPUPG IGGPEXT IGGPUPG IGGPGET IGGPUPG IGGPMOD IGGPUPG IGGPPDEC IGGPUPG IGGPMOD

	Description	Module	Procedure
5		IGG0CLCX IGG0CLAZ IGG0CLCX	IGGPDELX IGGPEXT IGGPDELX
6	The path association is removed from its AIX before the path is actually deleted. Note: If the catalog is recoverable, the call to IGG0CLAG will result in a CRA update unless bit CCARPUT (the update inhibit bit) is set to '1'. This bit is set in the calling procedures, which decide whether CRA updating is necessary for a given operation.	IGG0CLCX IGG0CLAV IGG0CLCX	IGGPDPTH IGGPMOD IGGPDPTH
7		IGG0CLCX IGG0CLAG IGG0CLCX	IGGPDPTH IGGPPDE IGGPDPTH
8	The AIX association is removed from its base cluster before the AIX is actually deleted.	IGG0CLCX IGG0CLAV IGG0CLCX	IGGPDELX IGGPMOD IGGPDELX
10	Each of the cluster data set's records is sequentially retrieved and overwritten with 0s.	IGG0CLBG $$BOPEN IGG0CLBG $$BCLOSE IGG0CLBG	IGGPDLDS IGGPERAS IGGPERAS IGGPERAS IGGPDLDS
11		IGG0CLBG IGG0CLAG IGG0CLBG	IGGPDLDS IGGPGET IGGPDLDS

Notes for Diagram EP (Part 2 of 3)

	Description	Module	Procedure
12	The volume catalog record also contains a data set directory group occurrence to describe each VSAM data set that is contained, partially or completely, on the volume. If the volume is a candidate volume for a data set or index, the data set or index is also described by a data set directory group occurrence.	IGG0CLBG IGG0CLA7 IGG0CLAZ IGG0CLA7 IGG0CLAZ IGG0CLA7 IGG0CLAV IGG0CLA7	IGGPDLDS IGGPVMSC IGGPDEMV IGGPEXT IGGPDEMV IGGPVMSC IGGPDEDD IGGPEXT IGGPDEDD IGGPMOD IGGPDEDD IGGPVMSC
13-16	The volume catalog record contains a data space group occurrence to describe each VSAM data space on the volume.	IGG0CLA7 IGG0CLAZ IGG0CLA7 IGG0CLAV IGG0CLA7	IGGPVMSC IGGPDESH IGGPEXT IGGPDESH IGGPMOD IGGPDESH IGGPVMSC
18	If the data set or index space is unique, its data space is also deleted. Before the data space is deleted, the volume containing it is mounted. The volume containing the data space is specified by the FILE parameter. The extents in the data space's format-1 (identifier) DSCB and format-3 (extension) DSCB are scratched from the VTOC.	IGG0CLA7 $$BOVS01 IGG0CLA7 IKQSCR00 IGG0CLA7	IGGPVMSC IGGPDVMV IGGPDVMV IGGPVMSC IGGPDUSC IGGPDUSC IGGPVMSC

	Description	Module	Procedure
19	Each entry in the list identifies one of the data set or index extents in one of the data spaces on the volume.	IGG0CLA7 IGG0CLAZ IGG0CLA7	IGGPVMSC IGGPDEVG IGGPEXT IGGPDEVG IGGPVMSC
20	Each of the data space's extents is described in the data space group occurrence.	IGG0CLA7 IGG0CLBF IGG0CLAZ IGG0CLBF IGG0CLAV IGG0CLBF IGG0CLA7	IGGPVMSC IGGPSSCR IGGPESDG IGGPEXT IGGPSSCR IGGPSCAN IGGPMRLC IGGPMOD IGGPSSCR IGGPVMSC
21		IGG0CLBG	IGGPDLDS
22-23	The delete routine erase the data set's true-name record and delete all references to the data set's DSNAME in the catalog's index. **Note: If the catalog is recoverable, the call to IGG0CLAG will result in a CRA update unless bit CCARPUT (the update inhibit bit) is set to '1'. This bit is set in the calling procedures, which decide whether CRA updating is necessary for a given operation.**	IGG0CLBG IGG0CLAN IGG0CLAG IGG0CLAN IGG0CLBG	IGGPDLDS IGGPDUND IGGPDEUN IGGPGET IGGPDDE IGGPDEUN IGGPDLDS

Notes for Diagram EP (Part 3 of 3)

	Description	Module	Procedure
24	The base cluster or AIX association is extracted from the path and the base record is read.	IGG0CLCX IGG0CLAZ IGG0CLCX IGG0CLAG IGG0CLCX	IGGPDELP IGGPEXT IGGPDELP IGGPGET IGGPDELP
25		IGG0CLCX IGG0CLAV IGG0CLCX	IGGPDELP IGGPMOD IGGPDELP
26-27	Note: If the catalog is recoverable, the call to IGG0CLAG will result in a CRA update unless bit CCARPUT (the update inhibit bit) is set to '1'. This bit is set in the calling procedures, which decide whether CRA updating is necessary for a given operation.	IGG0CLCX IGG0CLAG IGG0CLCX	IGGPDELP IGGPPDE IGGPDELP

Diagram ER1. Modify the upgrade set

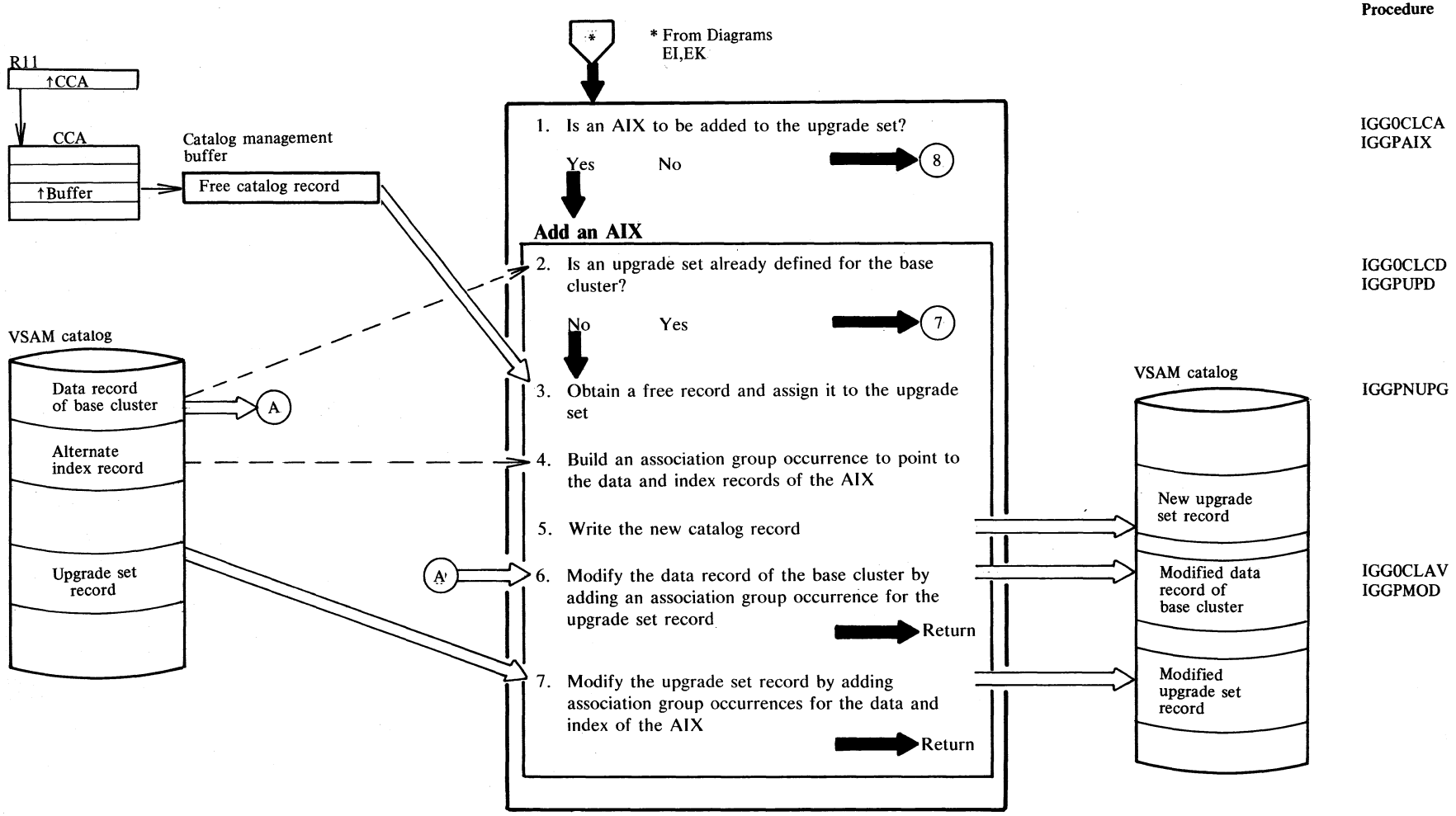


Diagram ER2. Modify the upgrade set

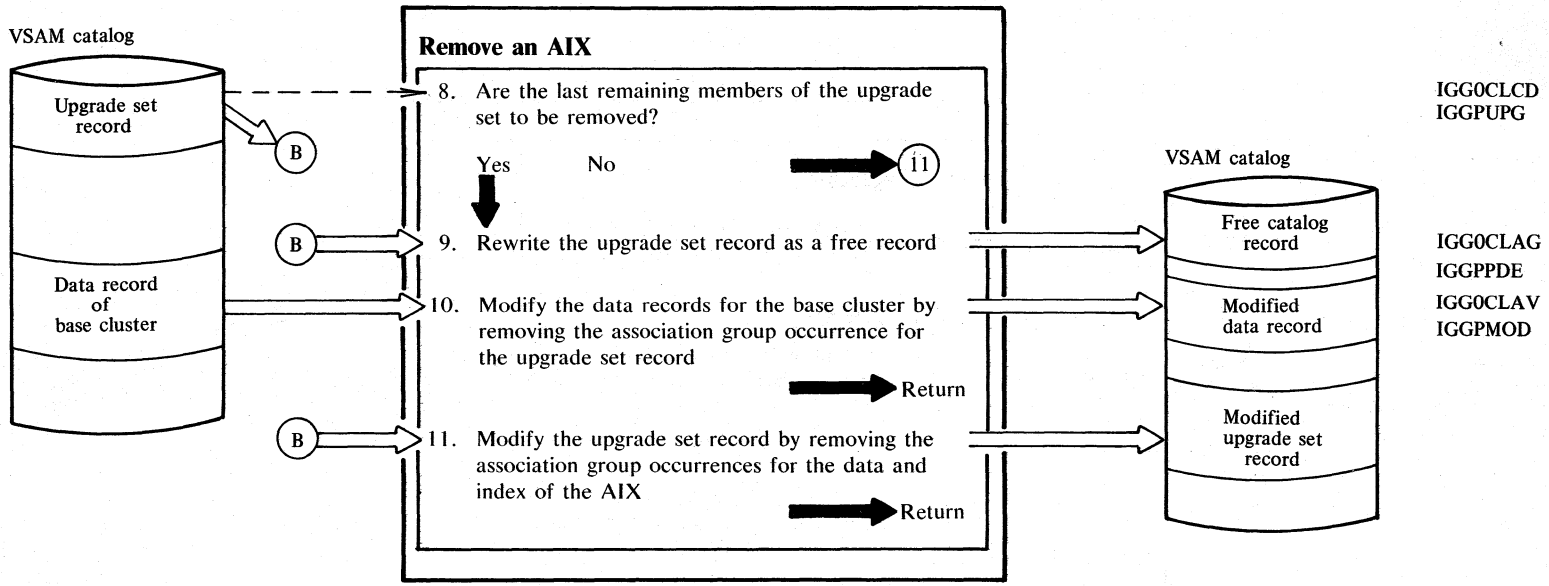


Diagram ES1. SHOWCAT: Display catalog information

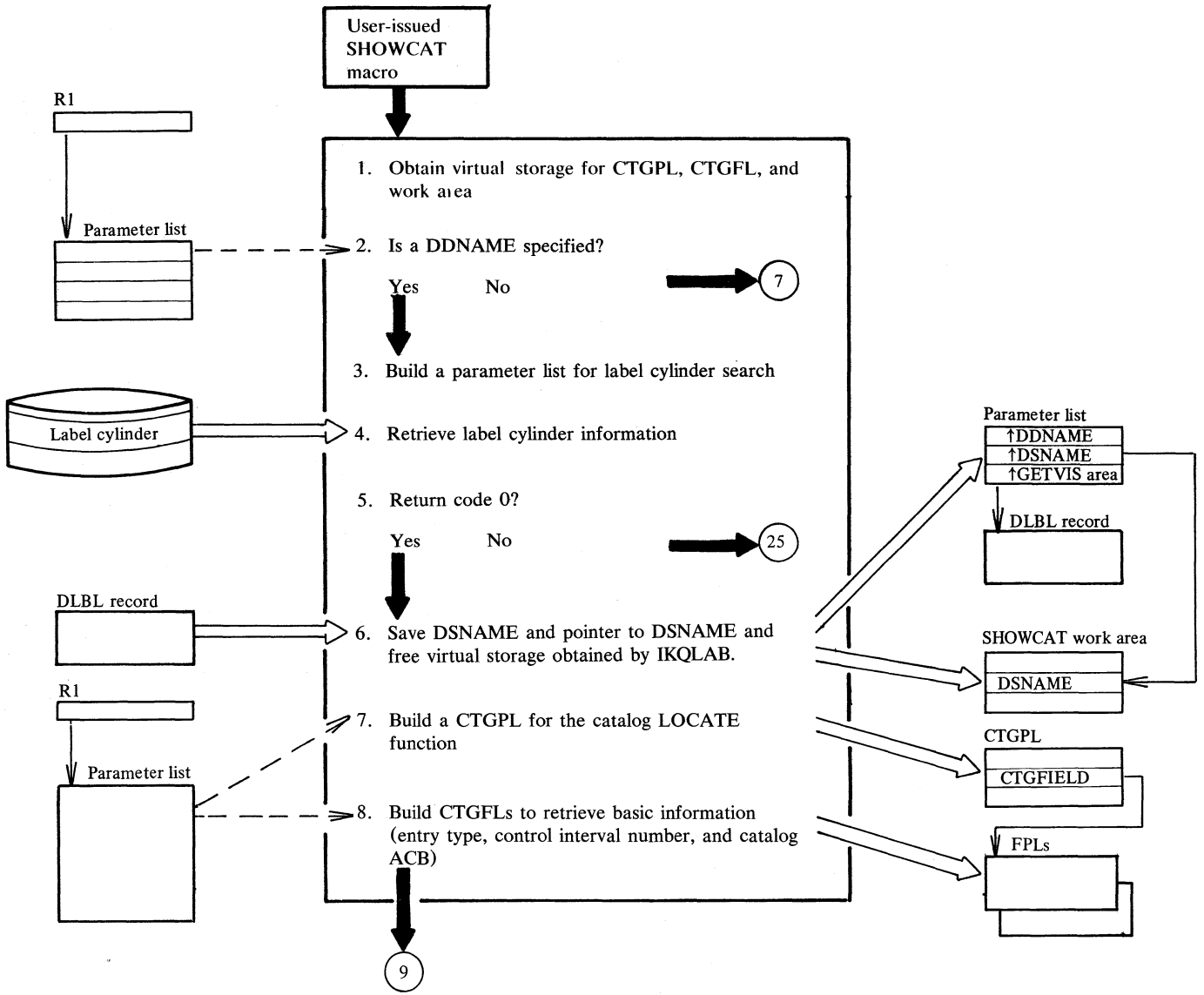


Diagram ES2. SHOWCAT: Display catalog information

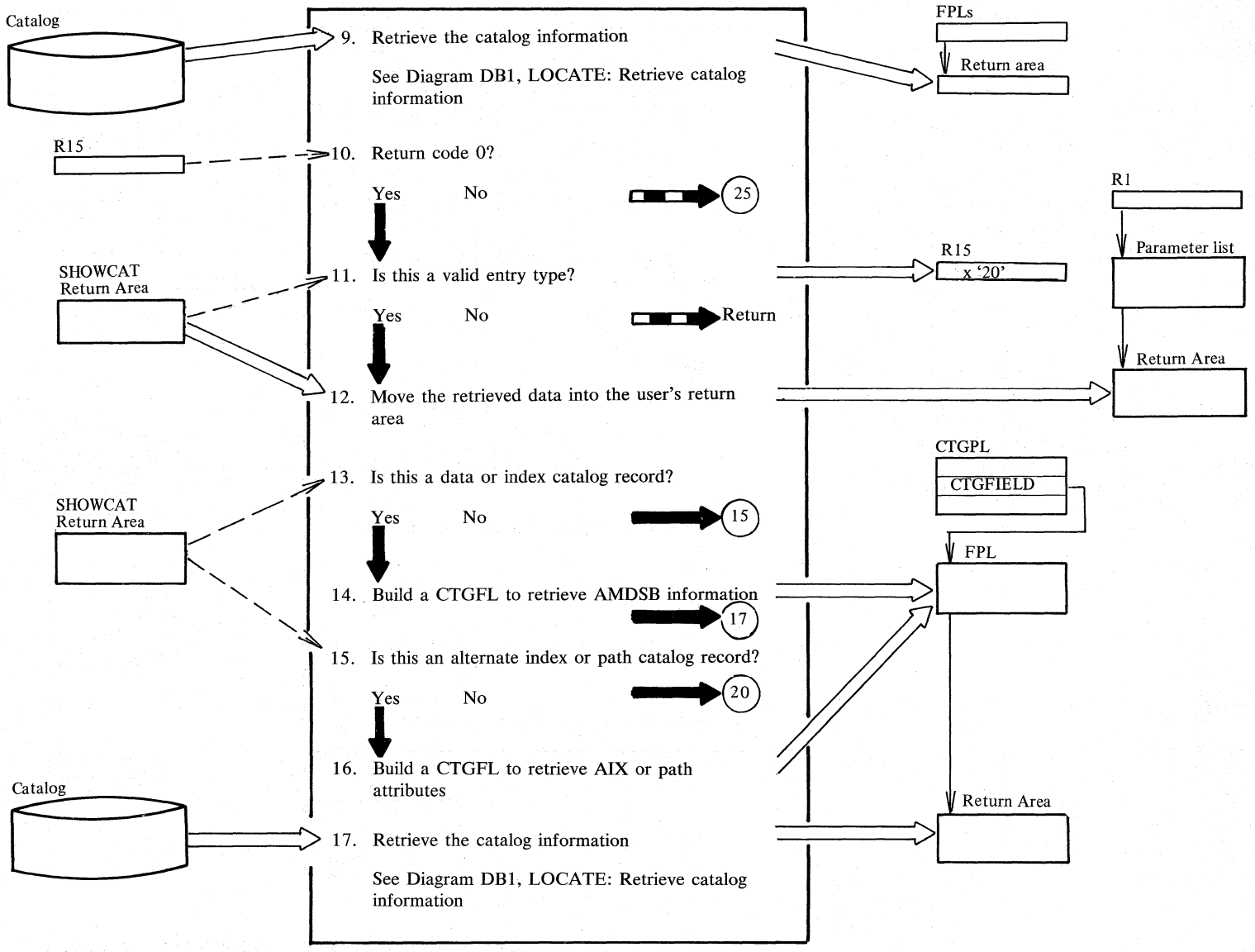


Diagram ES3. SHOWCAT: Display catalog information

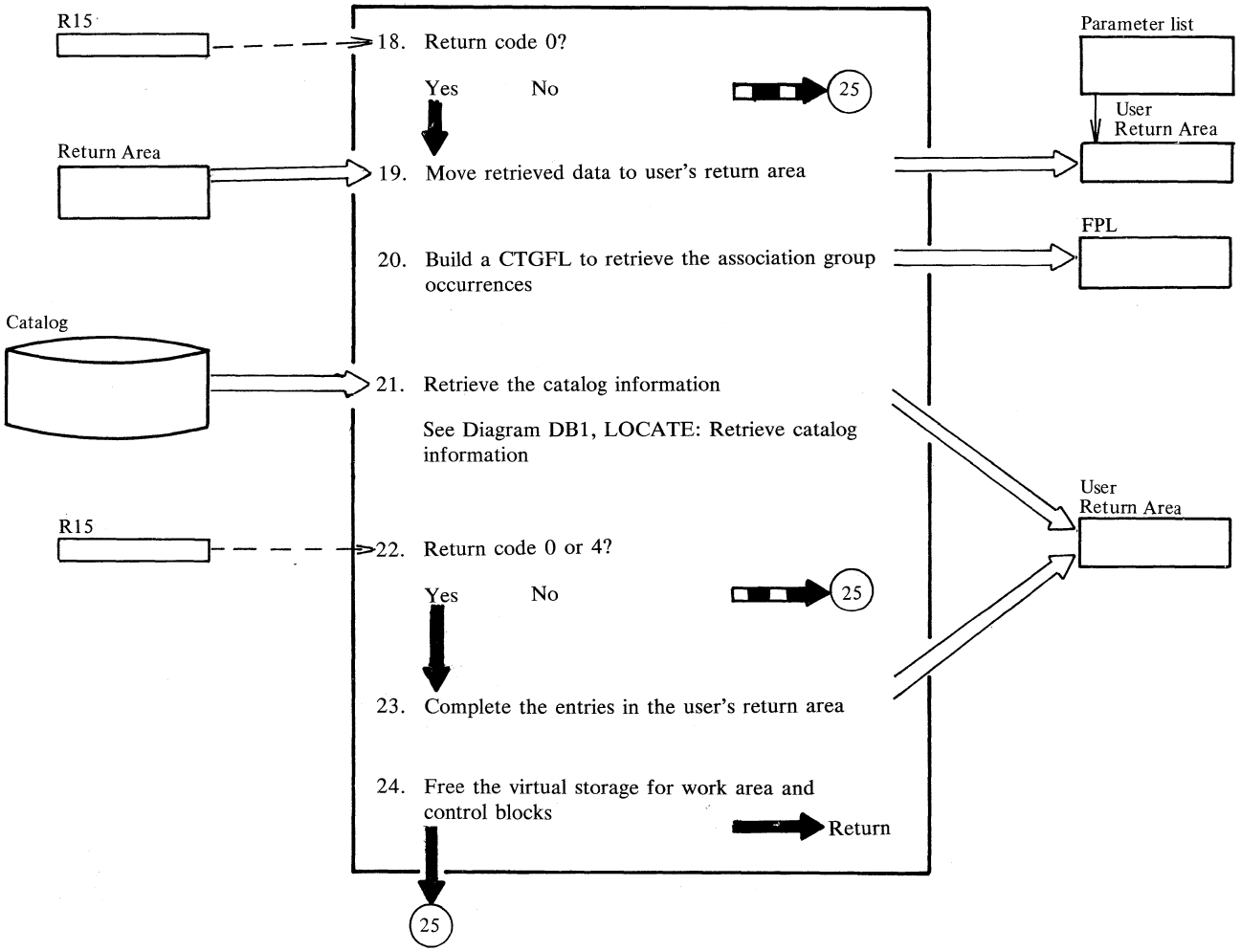
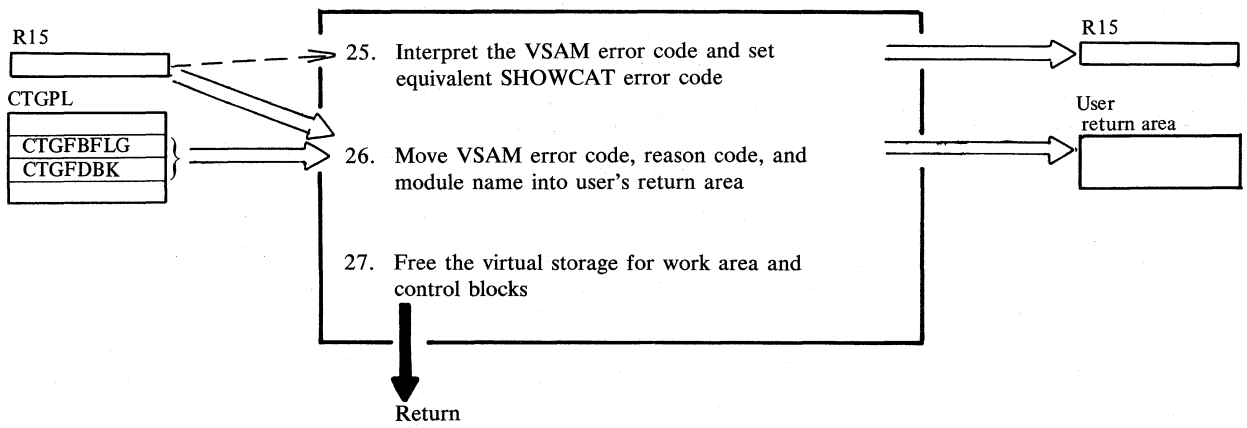


Diagram ES4. SHOWCAT: Display catalog information



Notes for Diagram ES

Description

1 256 bytes of virtual storage are needed.

8 If the CI number was specified, the contents of catalog fields ENTYPE and CATACB are retrieved. If the true name was specified, the field DSTYPNAM must be retrieved instead of ENTYPE, in order to find the CI number.

11 Only the following catalog record types may be retrieved with the SHOWCAT macro:

Cluster
Data
Index
Alternate index
Path
Upgrade set

22 Return code 4 indicates that the return area was too small to accept all group occurrences; the information retrieved (as much as would fit in the return area) is passed to the user.

23 Information saved before the LOCATE is restored and the length counts are updated.

Diagram FA1. DADSM contents

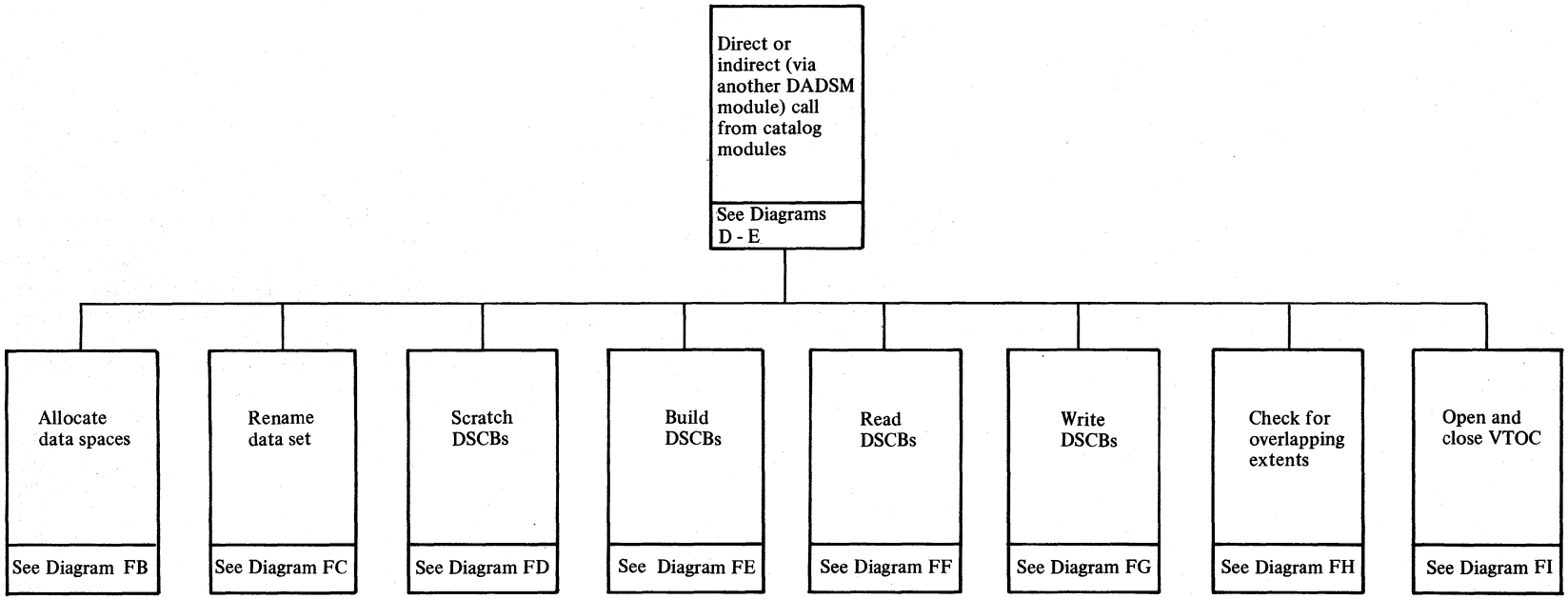


Diagram FB1. Allocate data spaces

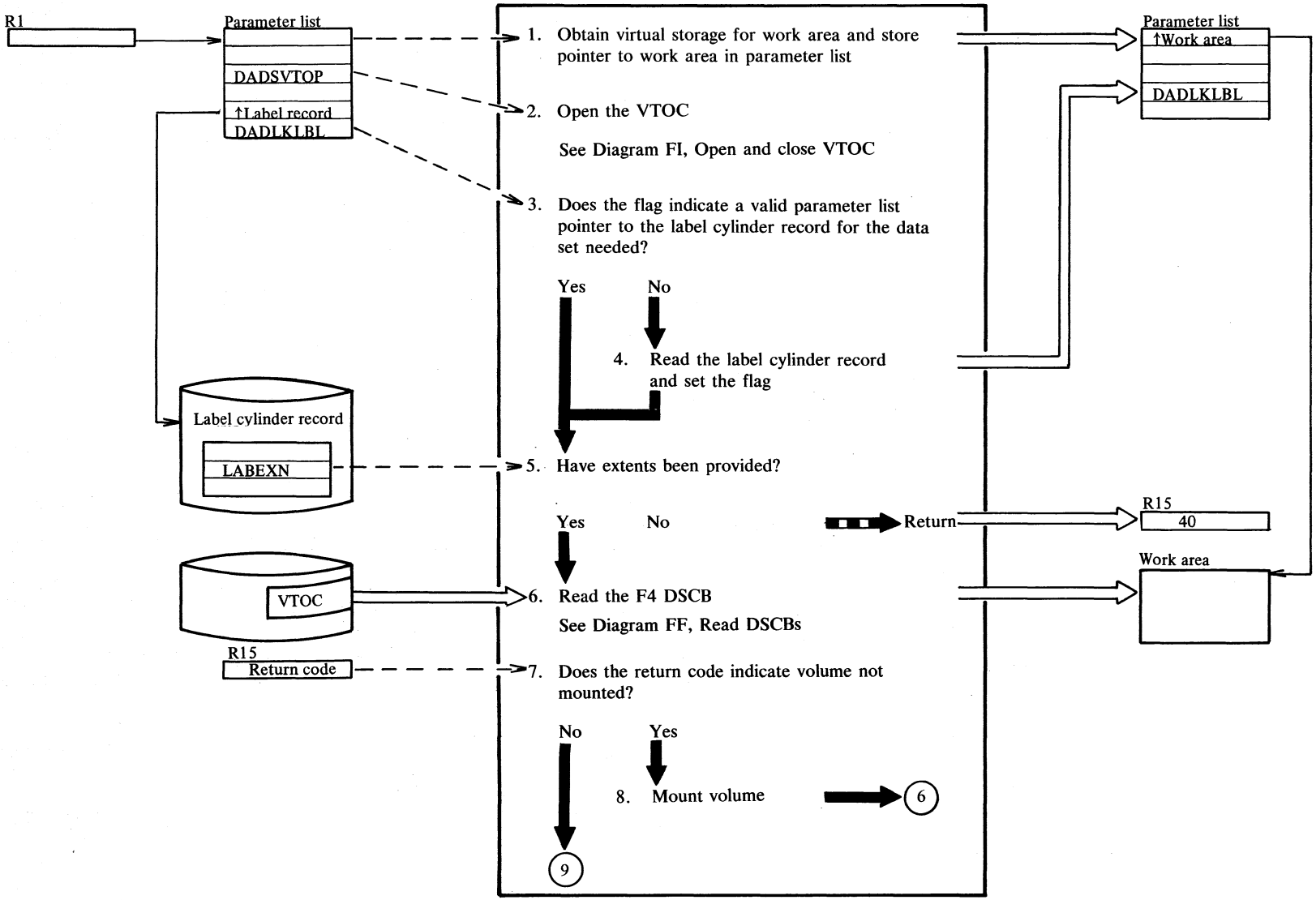
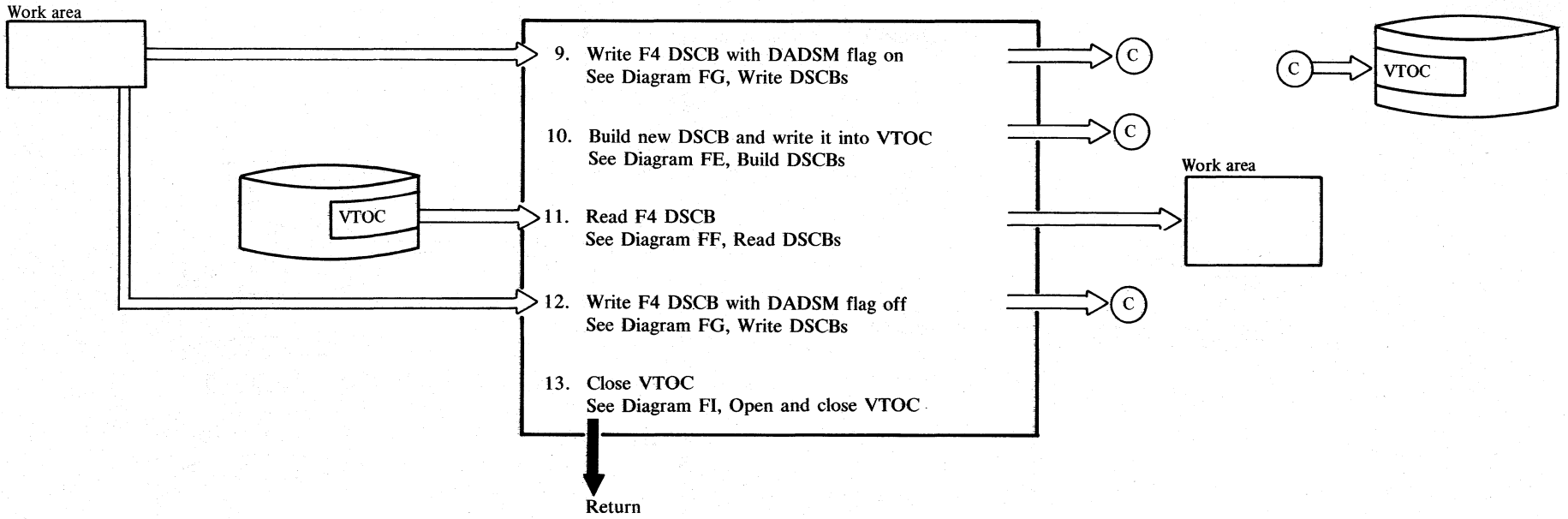


Diagram FB2. Allocate data spaces



Notes for Diagram FB

	Description	Module
1		IKQALL00
2	A JIB must be built if the file protect feature is present.	IKQVTC00
4		IKQLAB
5	The F4 DSCB is needed to obtain device information.	IKQRDS00
7		$$BOVS01
9	The F4 DSCB must be flagged while the DADSM routines are using the VTOC.	IKQWDS00
10	After the new DSCB is built, the DADSEXIT field in the DADSM parameter list is checked. If this field contains an address of a DADSM exit routine, that exit is taken. See Notes for Diagram EG for an explanation of IKQDXT processing.	IKQPOP00
11,12	The DADSM flag bit is turned off in the F4 DSCB.	IKQRDS00 IKQWDS00
13		IKQVTC00

Note:
The DADSM flag referenced is this diagram is bit 5 in byte 59 (decimal) of the F4 DSCB.

Diagram FC1. Rename unique data set

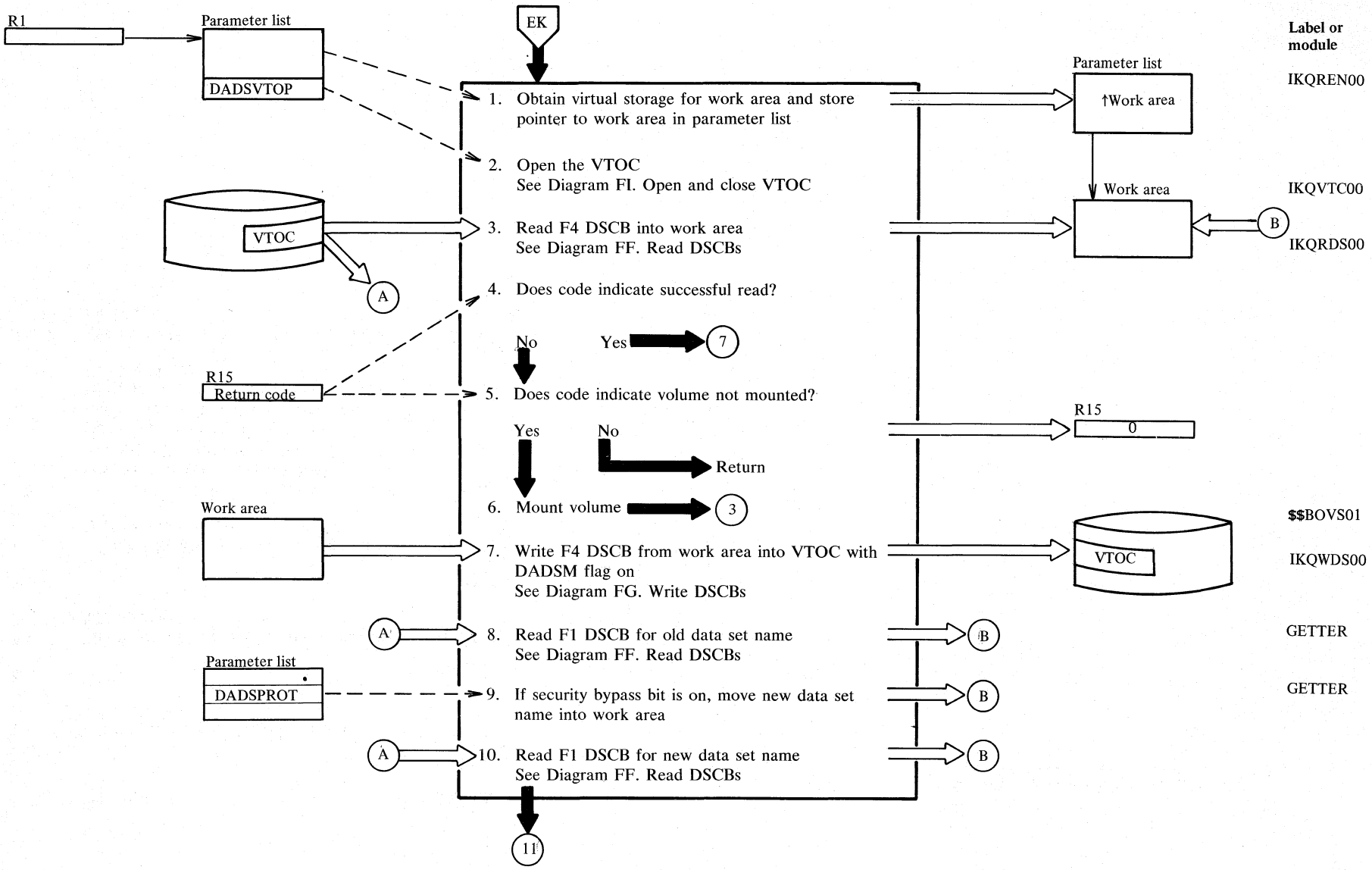
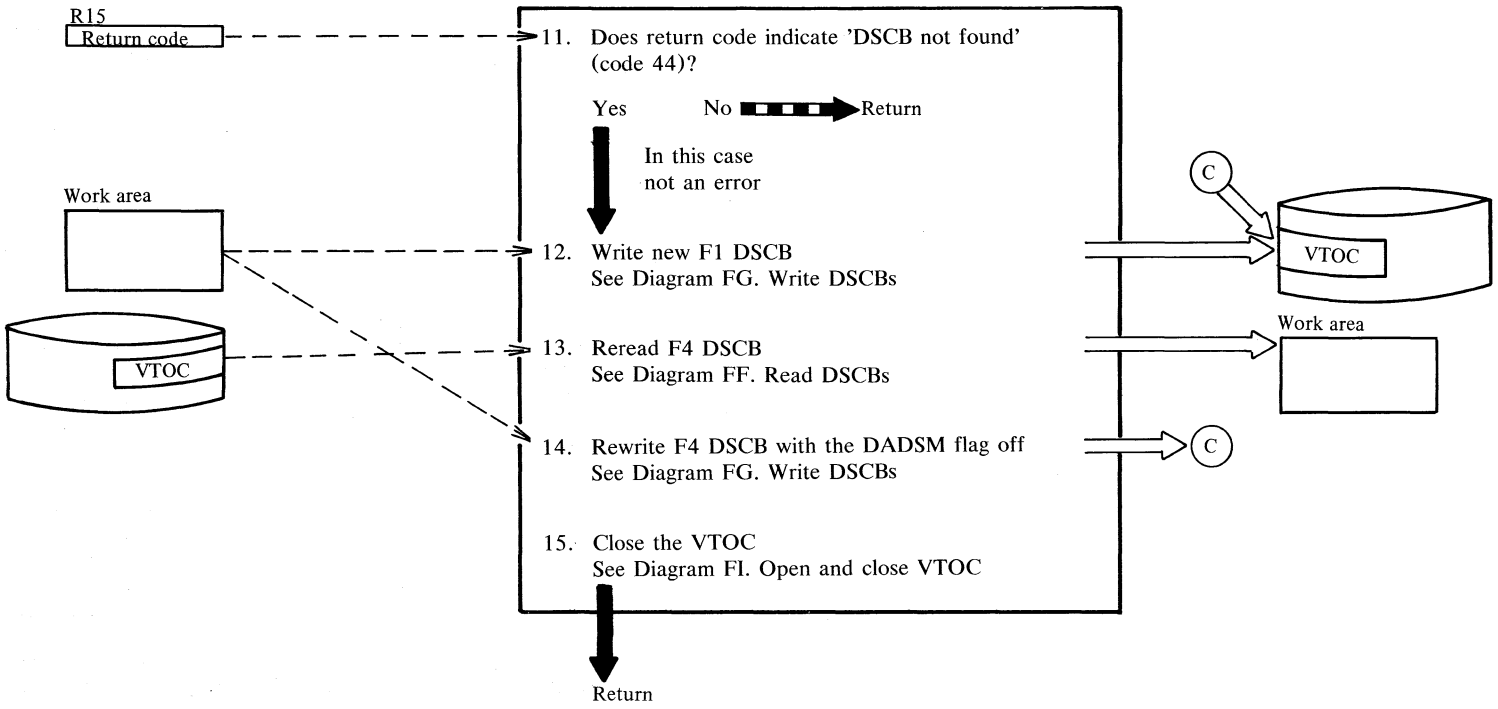


Diagram FC2. Rename unique data set



Notes for Diagram FC

	Description	Module
1		IKQREN00
2	A JIB must be built if the file protect feature is present.	IKQVTC00
3	The F4 DSCB is needed for device information.	IKQRDS00
6		$$BOVS01
7	The F4 DSCB must be flagged while DADSM routines are using the VTOC.	IKQWDS00
8	Fetch the F1 DSCB corresponding to the old data set name.	IKQRDS00
10	Try to find F1 DSCB for new data set name (check for duplicate name already in VTOC).	IKQRDS00
11	'DSCB not found' indicates no duplicate name found.	
12		IKQWDS00
13,14	The DADSM flag must now be turned off.	IKQRDS00 IKQWDS00
15	Any JIB(s) built in step 2 must be freed.	IKQVTC00

Note:
The DADSM flag referenced in this diagram is bit 5 in byte 59 (decimal) of the F4 DSCB.

Diagram FD1. Scratch DSCBs

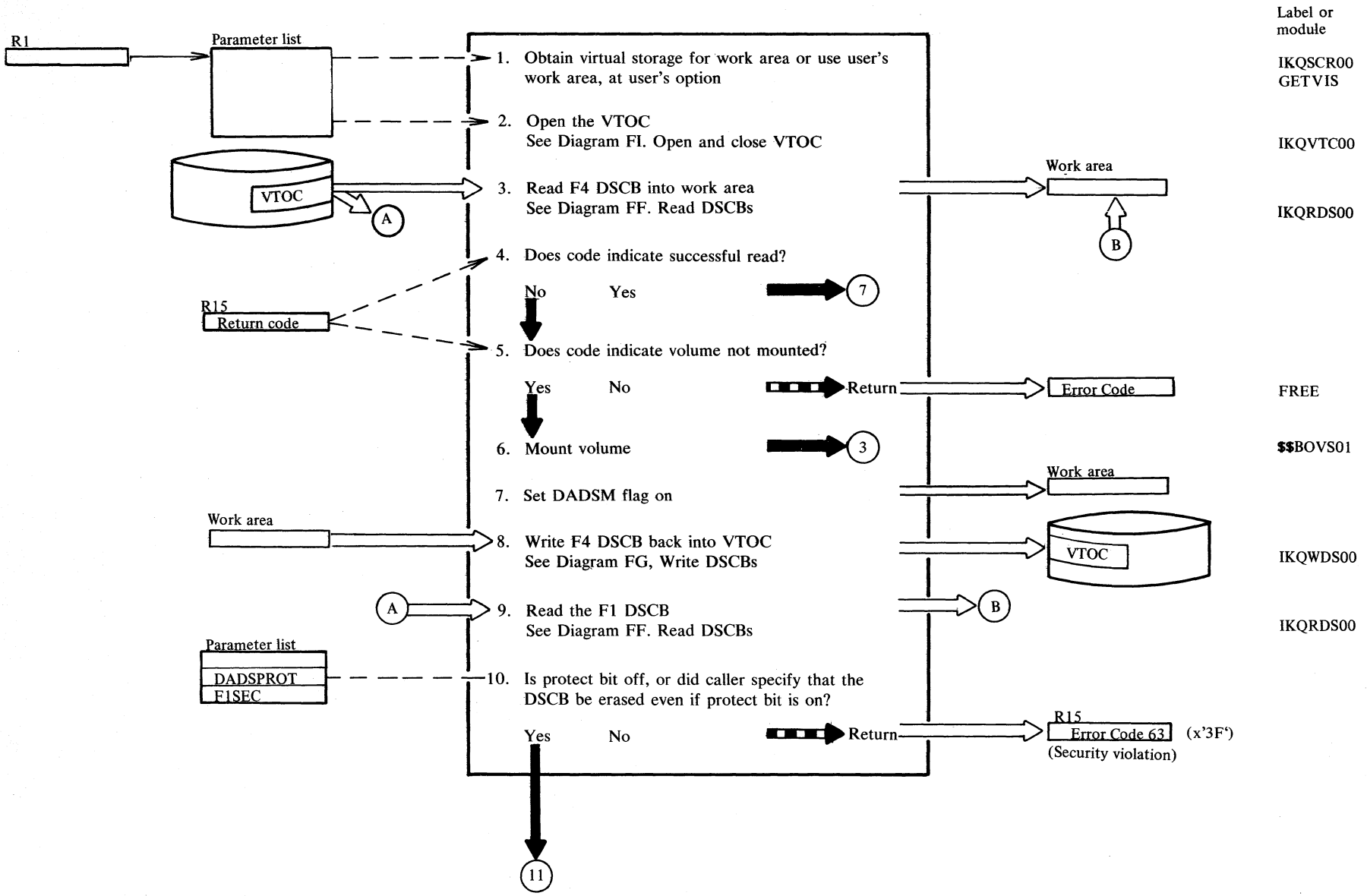
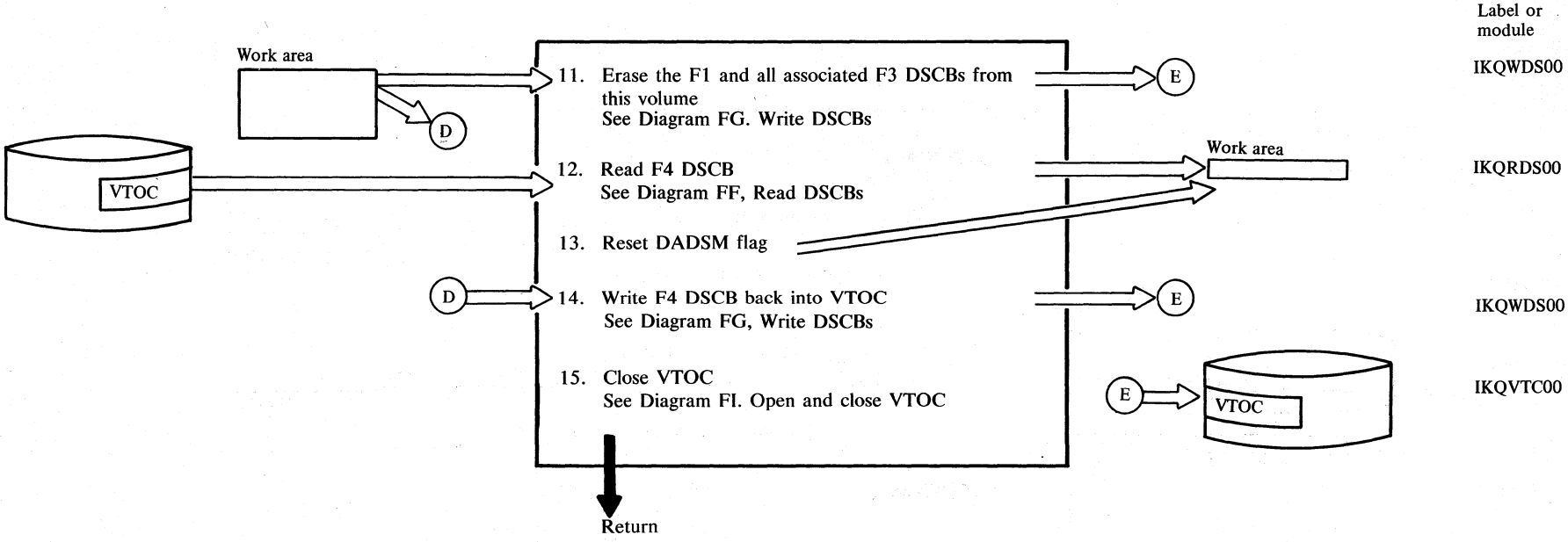


Diagram FD2. Scratch DSCBs



Note:
The DADSM flag referenced in this diagram is bit 5 of byte 59 (decimal) of the F4 DSCB.

Diagram FE1. Build DSCBs

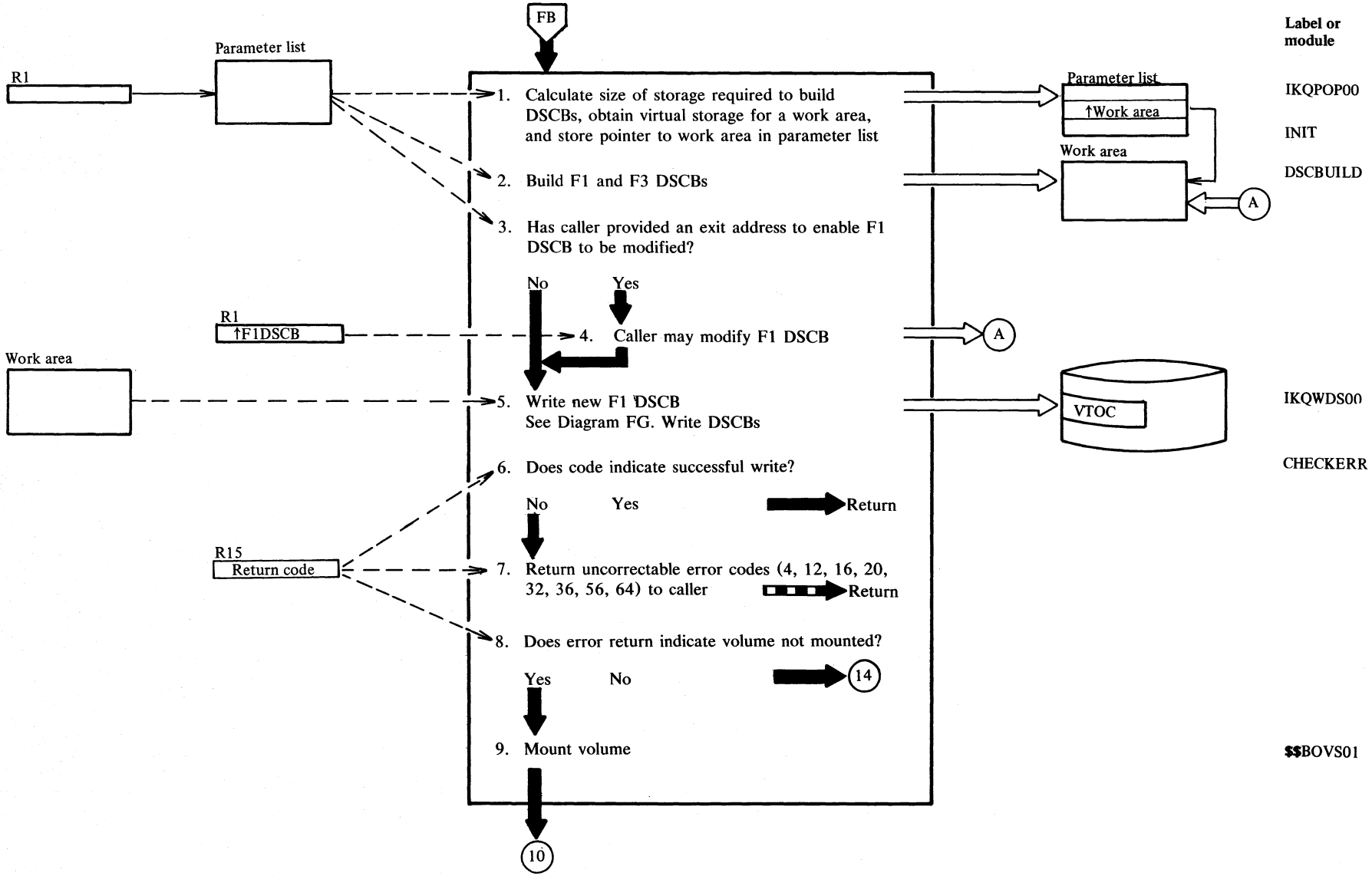


Diagram FE2. Build DSCBs

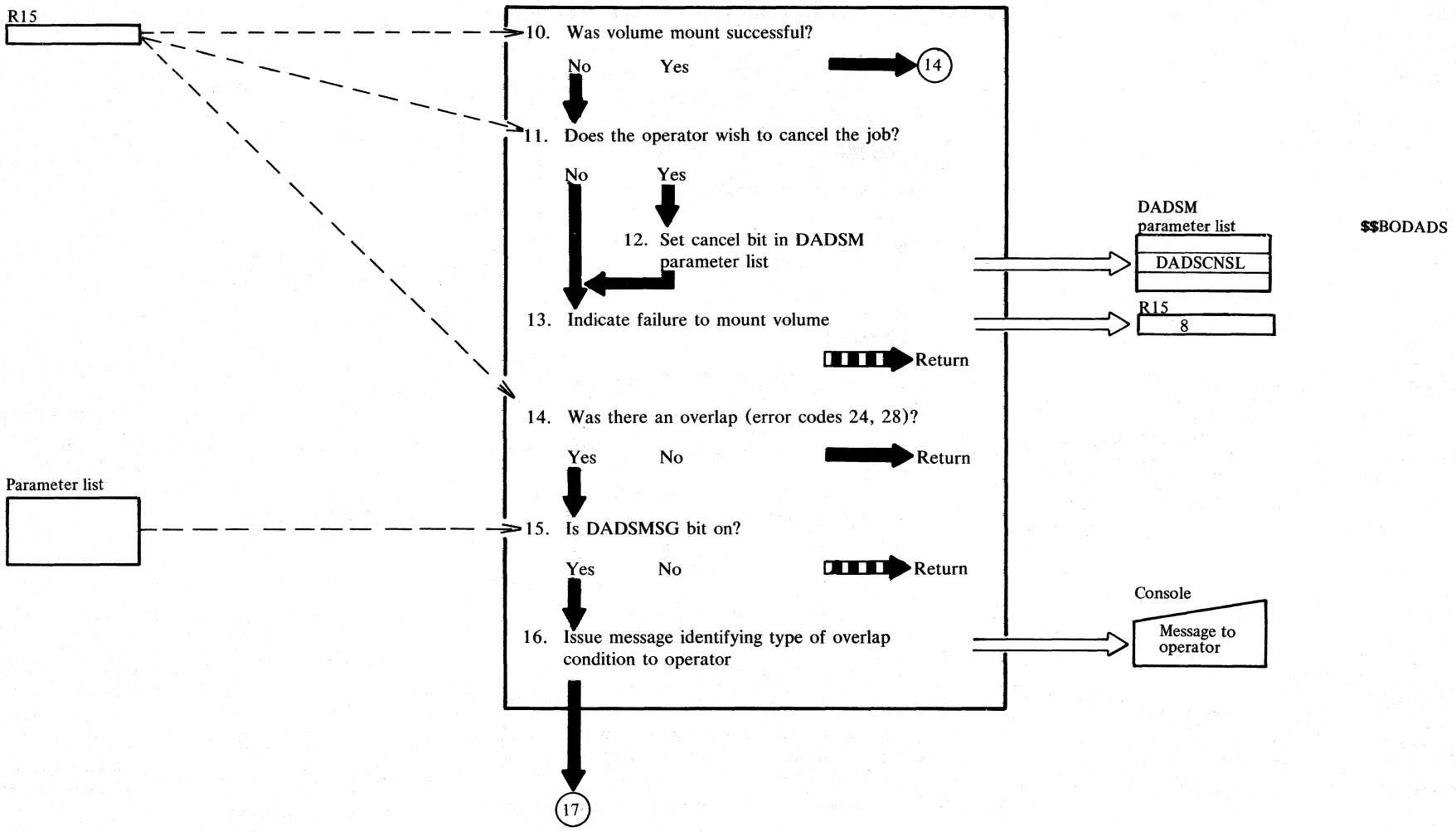


Diagram FE3. Build DSCBs

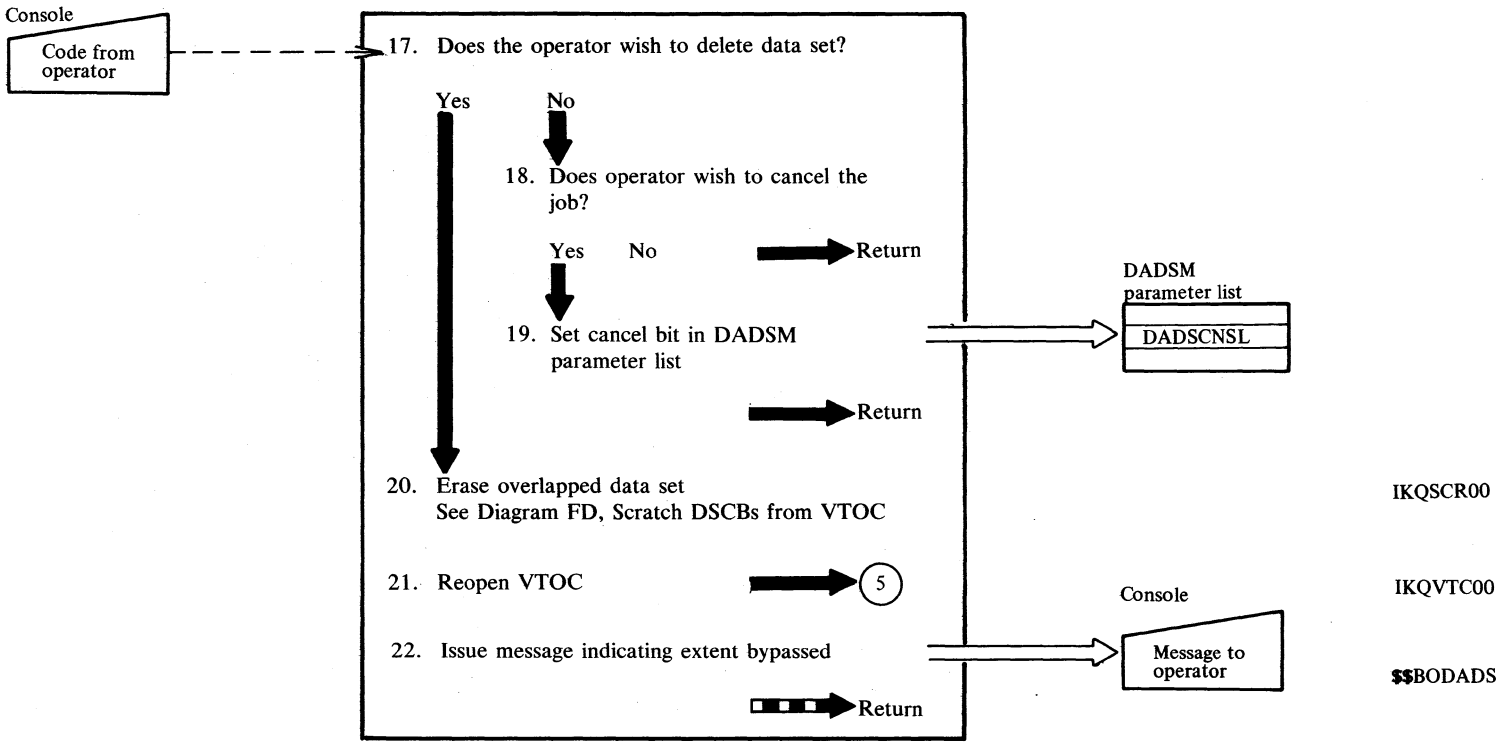


Diagram FF1. Read DSCBs

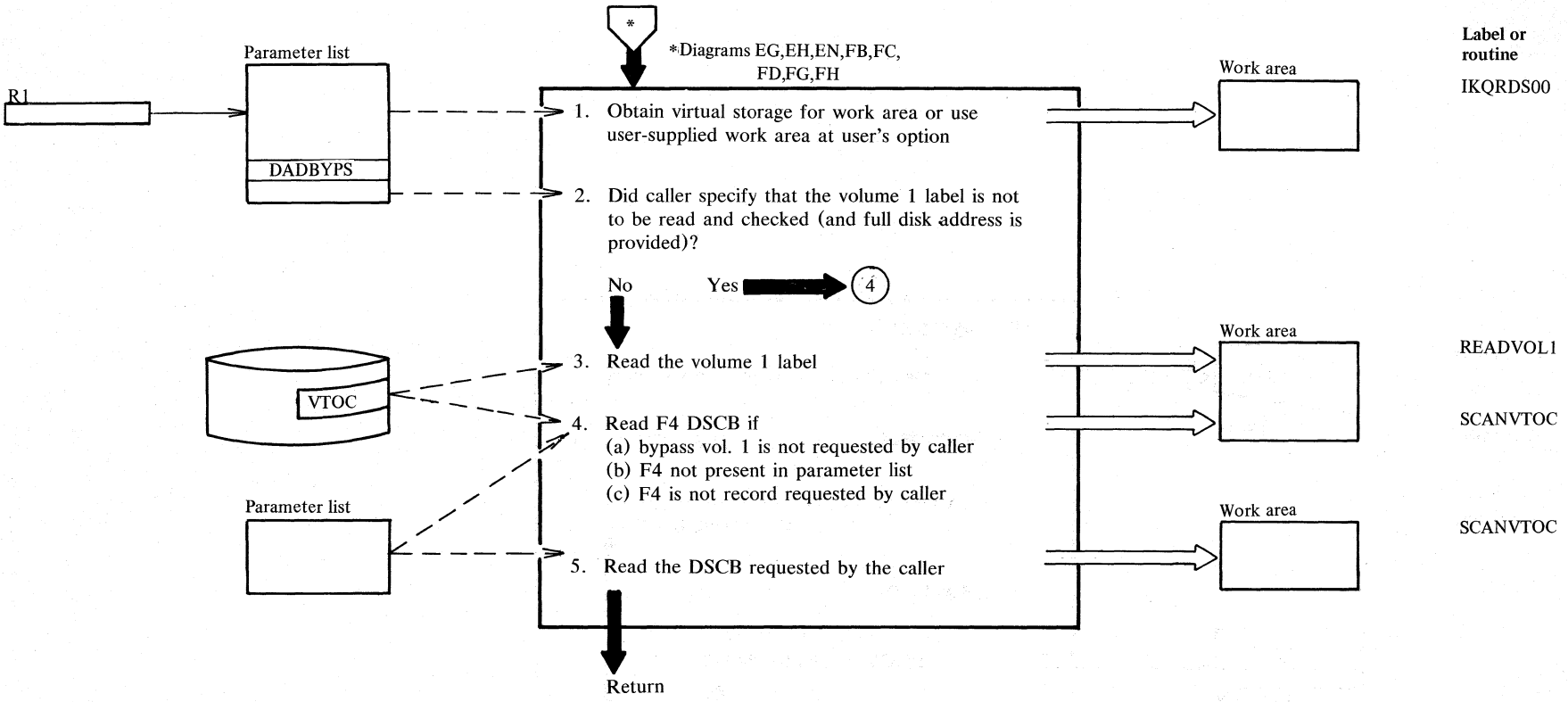


Diagram FG1. Write DSCBs

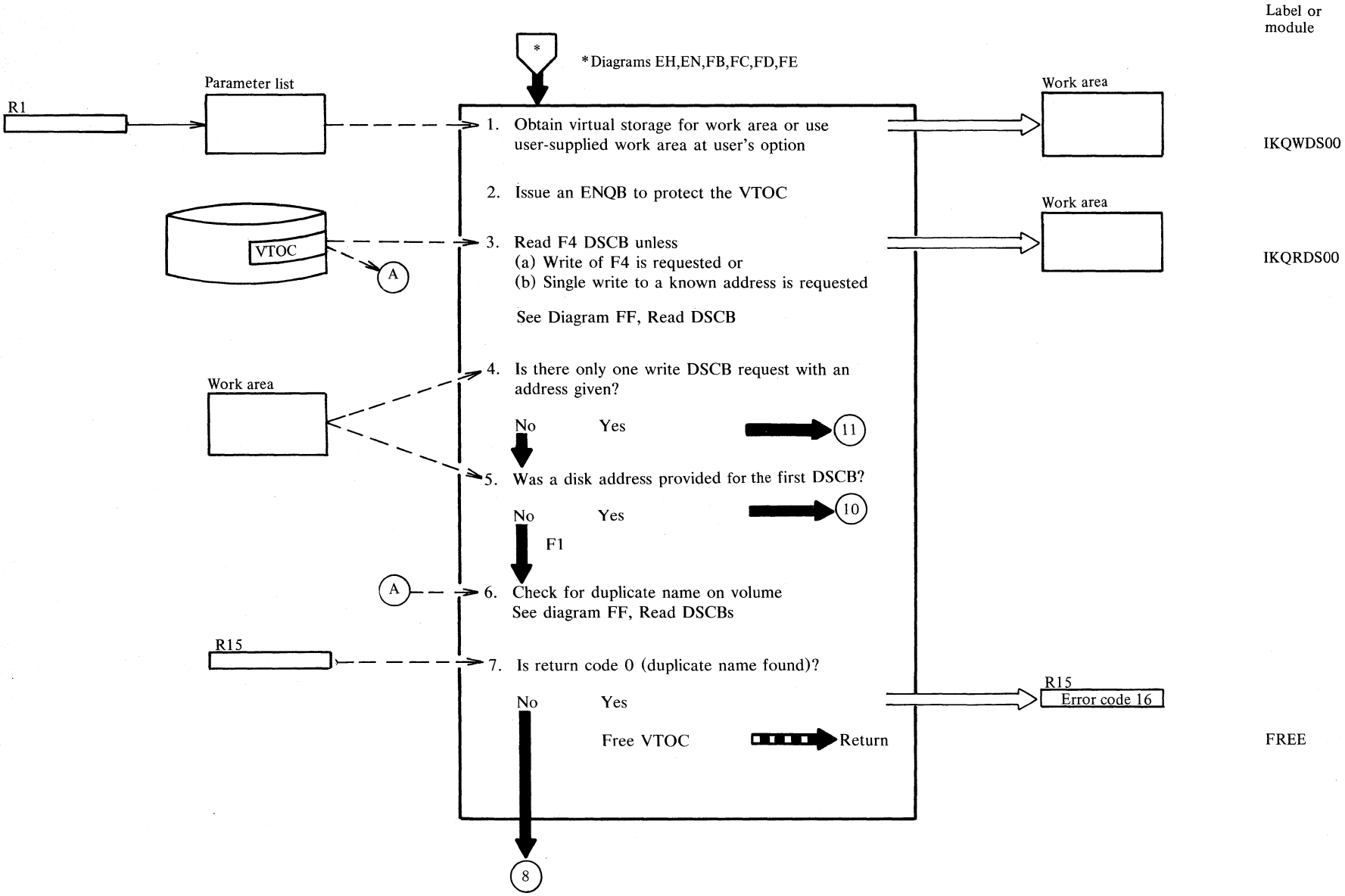
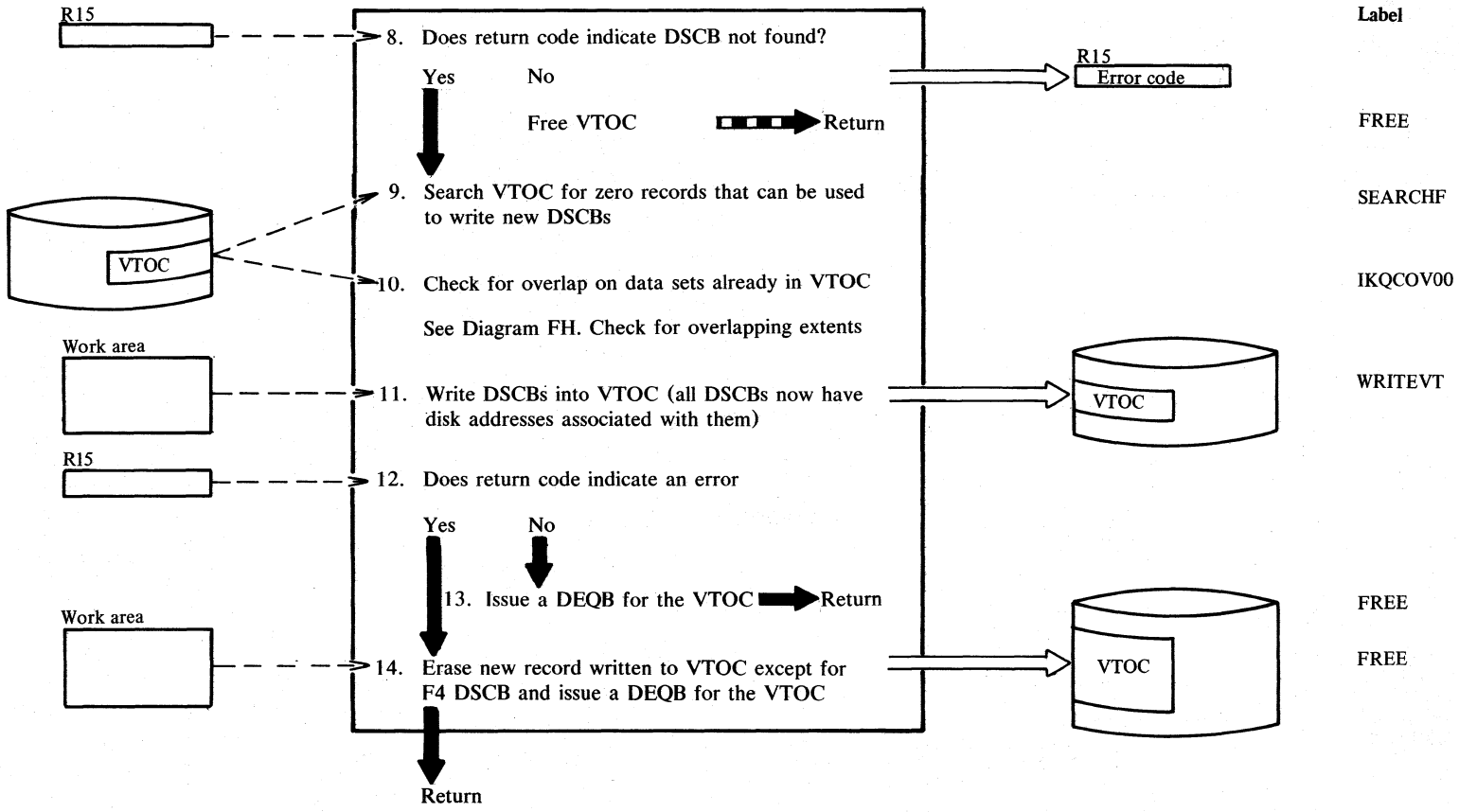


Diagram FG2. Write DSCBs



Notes for Diagram FG

Description

2 ENQB blocks the B-transient area, resulting in serialization, which prevents the VTOC from being processed by other VSAM tasks or other access methods.

3 If the request is not to write the Format-4 DSCB and not to write at a given address, the Format-4 DSCB must be read to get the starting and ending addresses of the VTOC.

5 If a disk address is not provided, then the DSCB is a Format-1 label. It can be written in any available slot in the VTOC.

6-8 This DSCB can be added to the VTOC only if a DSCB with the same name is not found.

10 The file described by this DSCB must not have extents that overlap files already defined by DSCBs in the VTOC.

14 If an error occurs, an attempt is made to restore the VTOC to its previous status by erasing any new records written for this request.

Diagram FH1. Check for overlapping extents

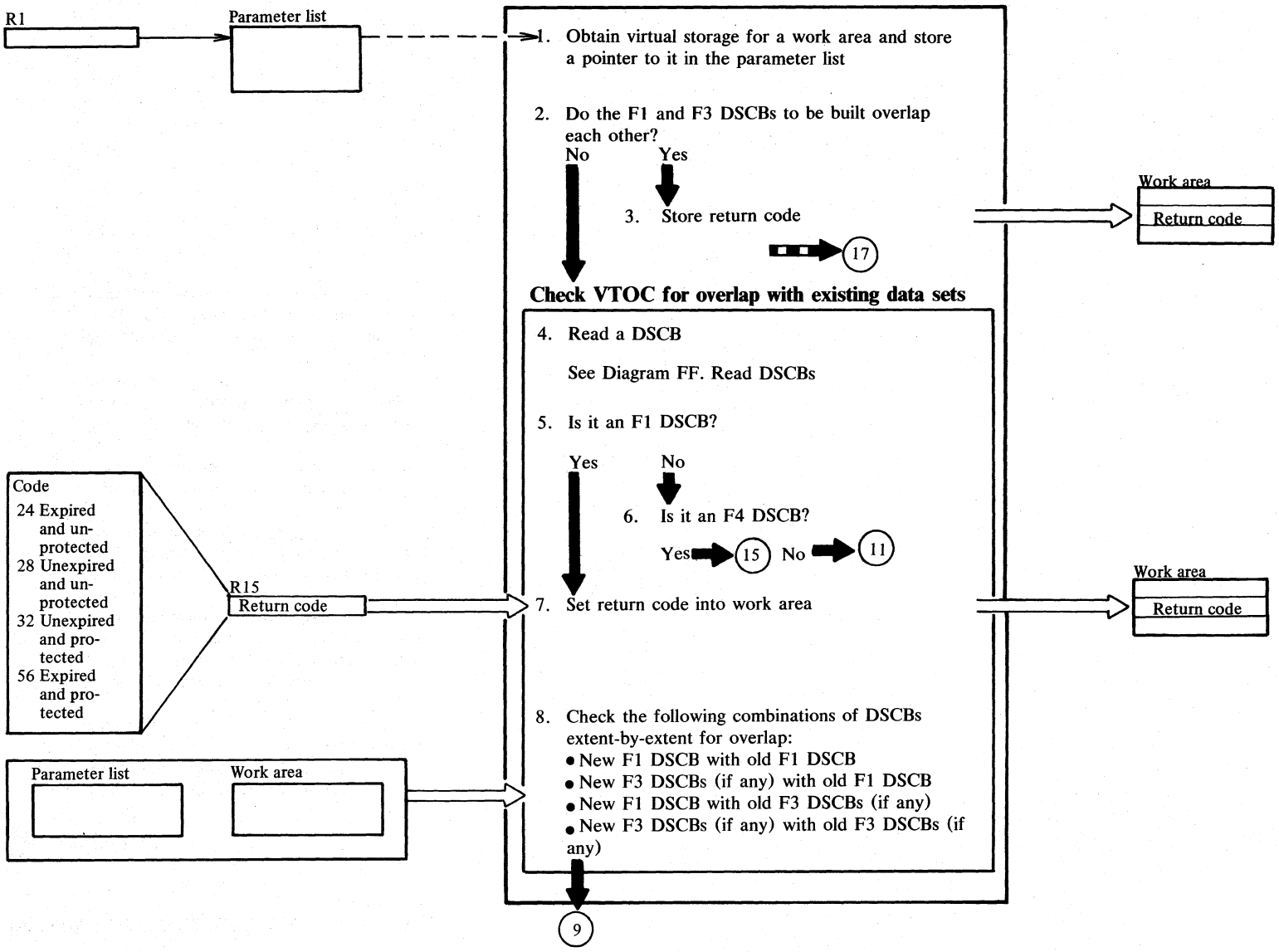
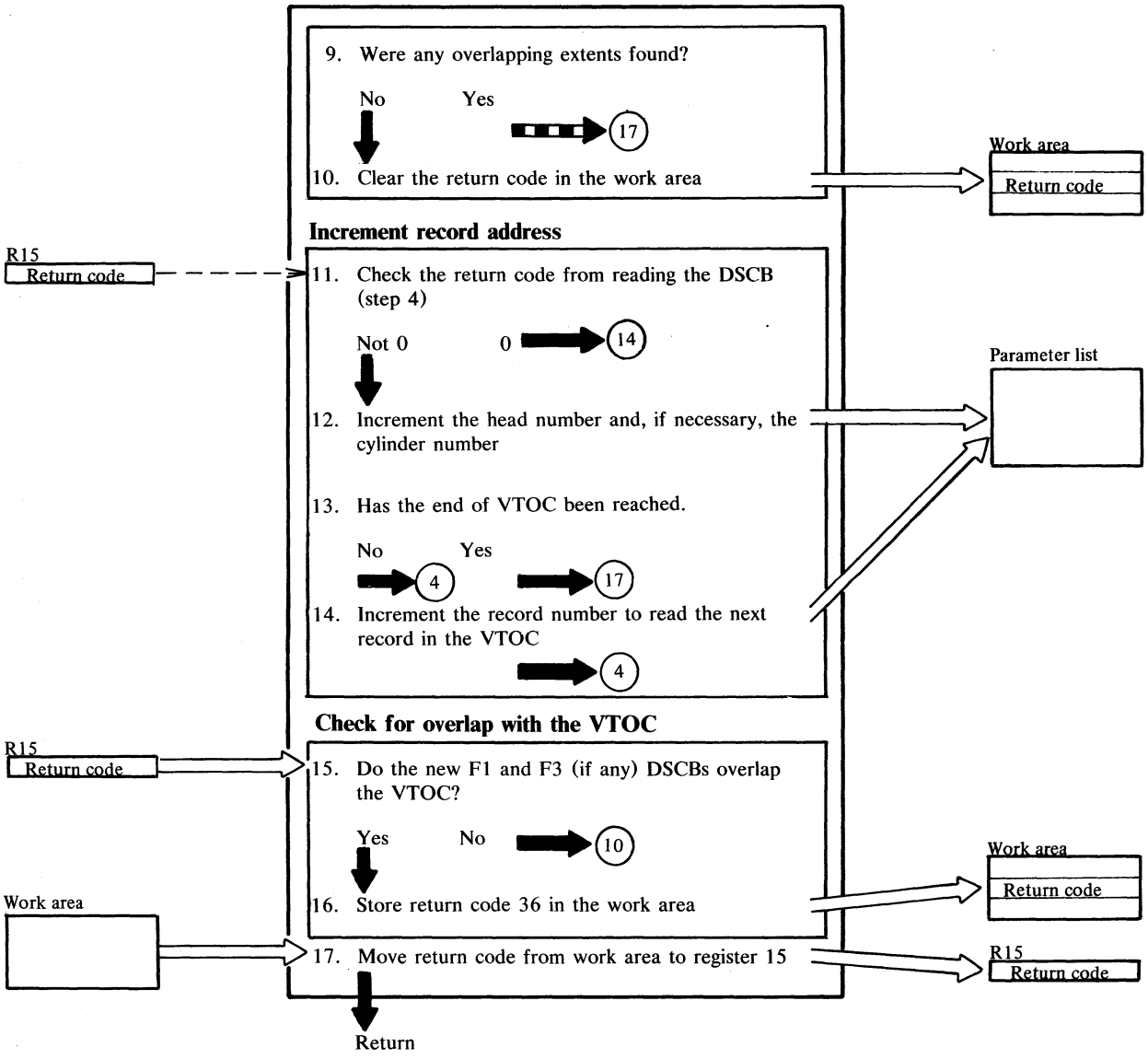


Diagram FH2. Check for overlapping extents



Notes for Diagram FH

	Description
1	
2	The DSCB(s) to be built are checked for mutual overlaps.
4,5,6	A DSCB is read each time the module loops. Only F1 and F4 DSCBs are needed. F3 DSCBs are checked in combination with their F1 DSCBs. Other DSCBs have no extent information.
7	The return code is stored in the work area in case it is needed later.
8	This step naturally involves reading many DSCBs from disk. These read operations, which are carried out by module IKQRDS00, are not shown here.
9	The first overlap actually found in step 8 causes further checking to be abandoned.
10	If there were no overlaps, the return code set up in step 7 is cleared.
11	A non-zero return code means that the record number requested does not exist. The fact that this check is carried out rather late means that the record read on the 'previous' loop is, in fact, checked twice, as it is still in the work area.
12	The head number, and the cylinder number if necessary, are incremented by one, and the record number is reset to one.
13	The incremented track and cylinder numbers are checked against information saved from the F4 DSCB, to see if the end of VTOC has been reached.

Module		Description
IKQCOV00	15	When the F4 DSCB is found, the new F1 and F3 DSCBs (if any) are checked for overlap with extents belonging to the VTOC.
IKQRDS00	17	The return code previously stored in the work area is moved to register 15 before returning to the caller.
IKQCOV00		

Diagram FI1. Open and close VTOC

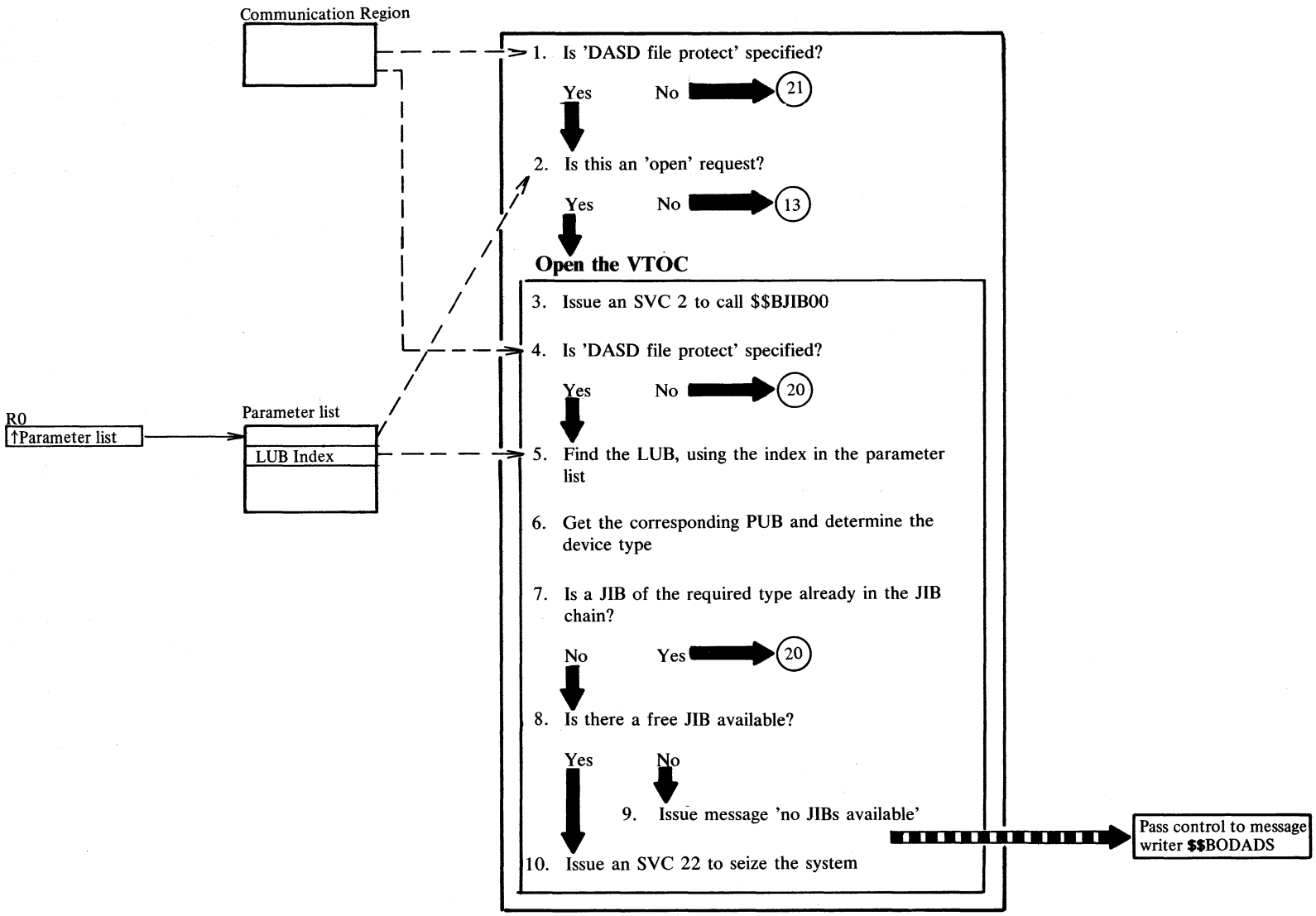
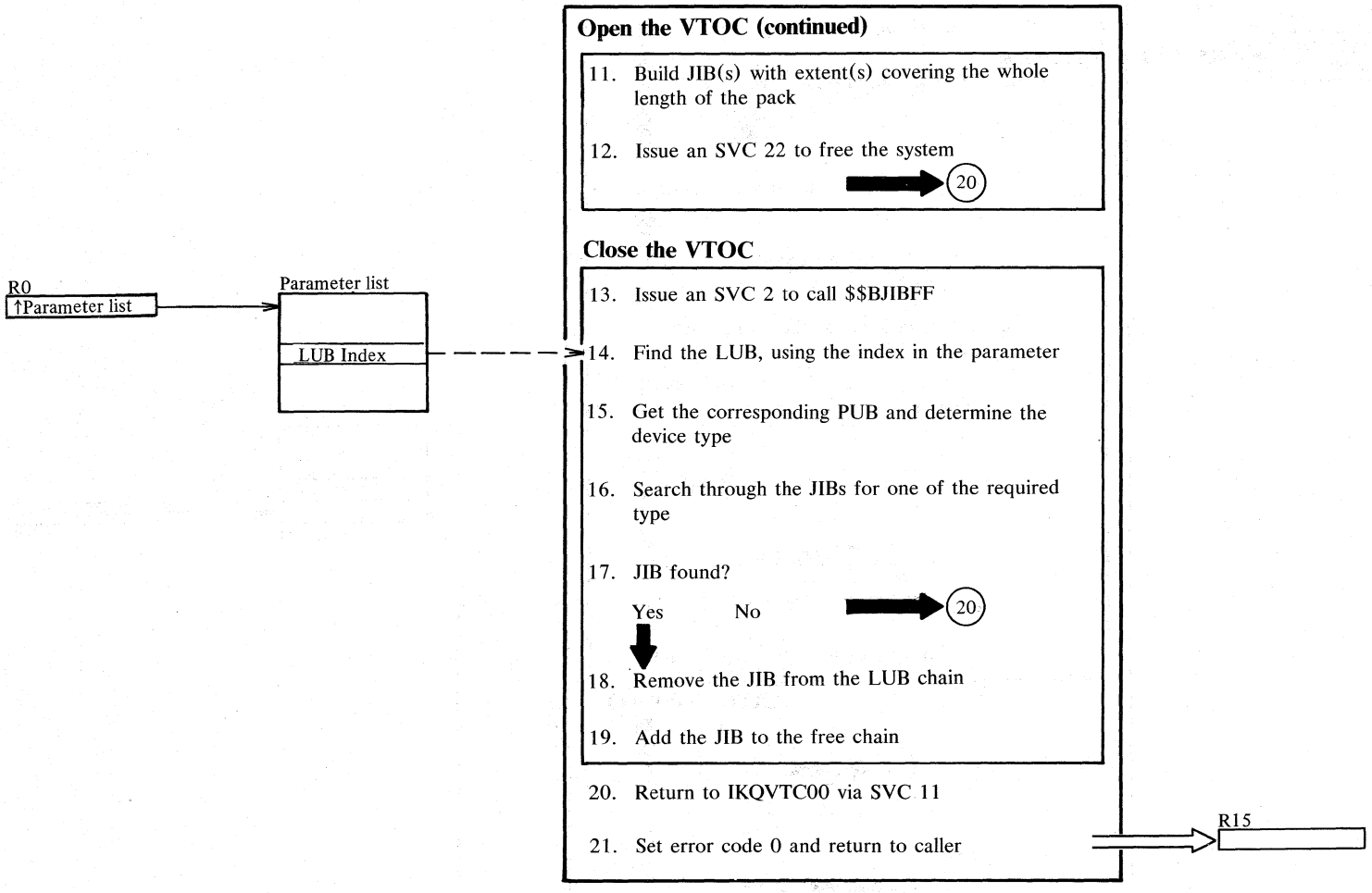
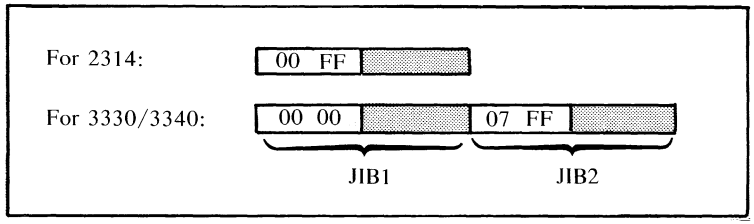


Diagram FI2. Open and close VTOC



Notes for Diagram FI

	Description	Module
1	DASD file protect is a SYSGEN option and is flagged in the communications region.	IKQVTC00
2	It is an open request if the bit DADSVTOP in the parameter list is on. Note: At this point, the parameter list pointer is in register 1. It is transferred to register 0, by module IKQVTC00, shortly before the SVC 2 is issued.	
4	See note for step 1.	$$BJIB00
5	The parameter list is a 'pseudo CCB' and contains the LUB index at the same offset as a CCB.	
7	The required type of JIB is identified by its extent information. See the diagram below.	
9	Control is passed to the message writer $$BODADS which writes the message and returns control to the caller.	
11	See diagram below for details of the extent information in the JIB(s).	
14	See note for step 5.	$$BJIBFF
16	See note for step 7.	
21	Module IKQVTC00 returns no error codes.	IKQVTC00



Extent information in the required type of JIB

Diagram GA1. Record management contents

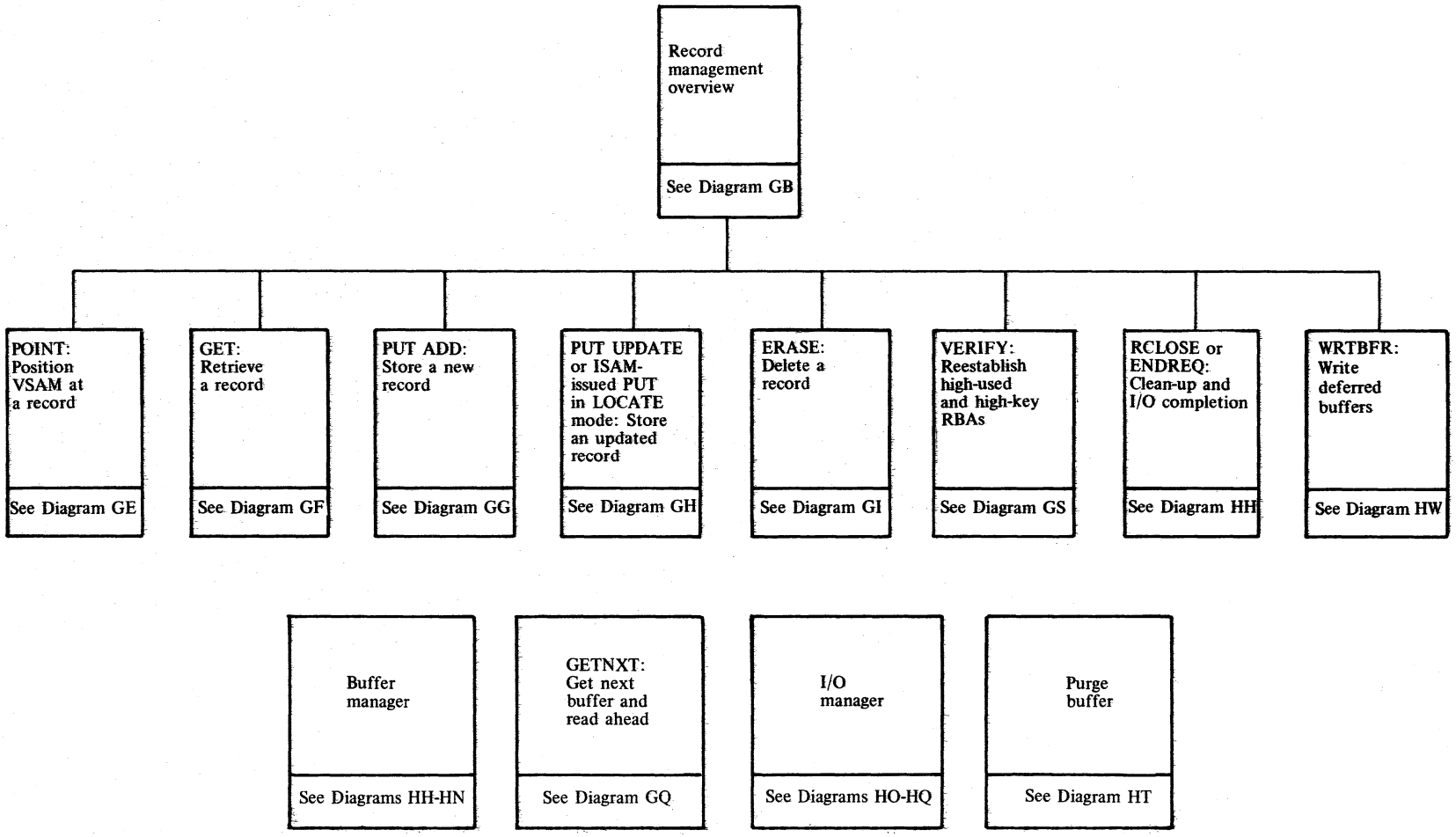


Diagram GB1. Record management overview

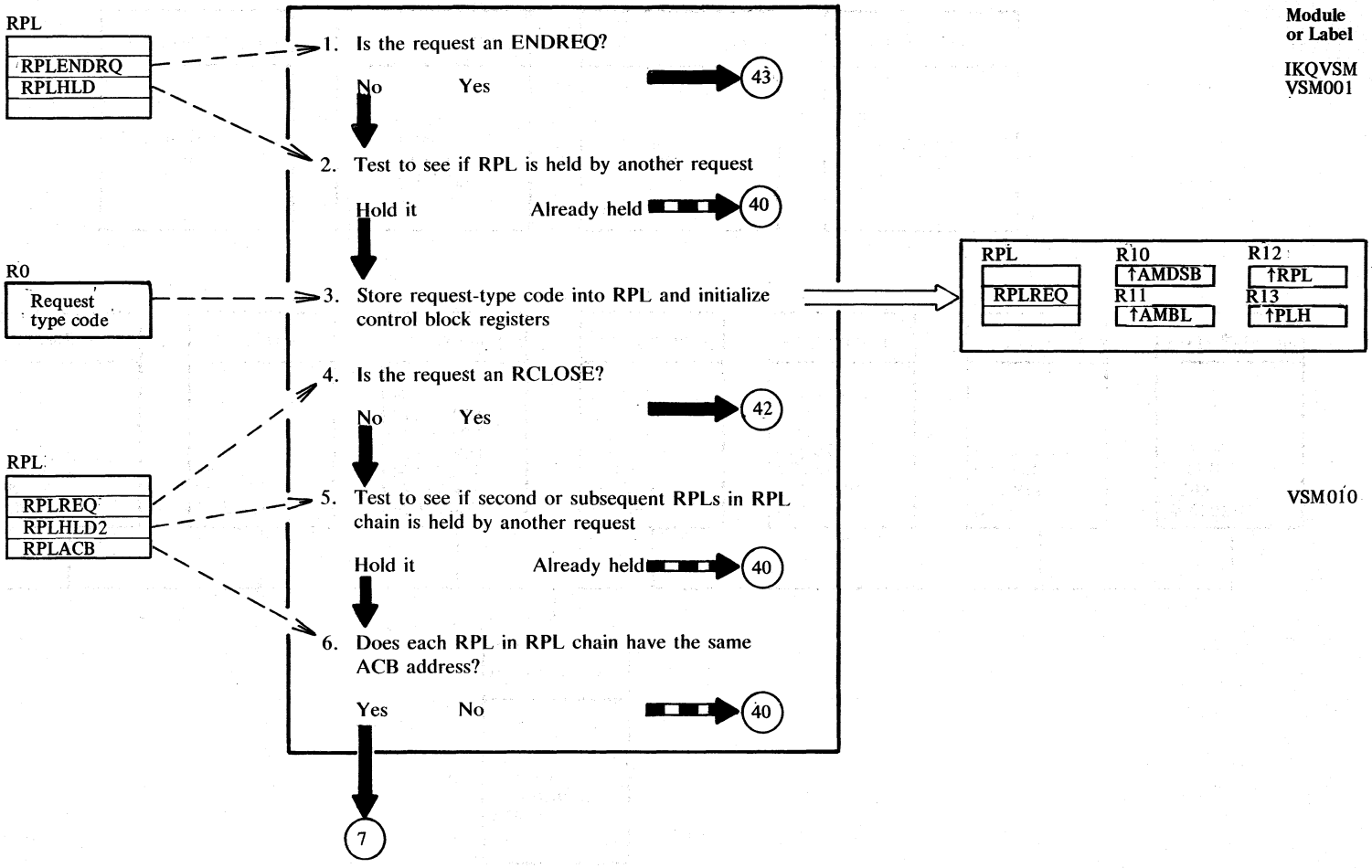


Diagram GB2. Record management overview

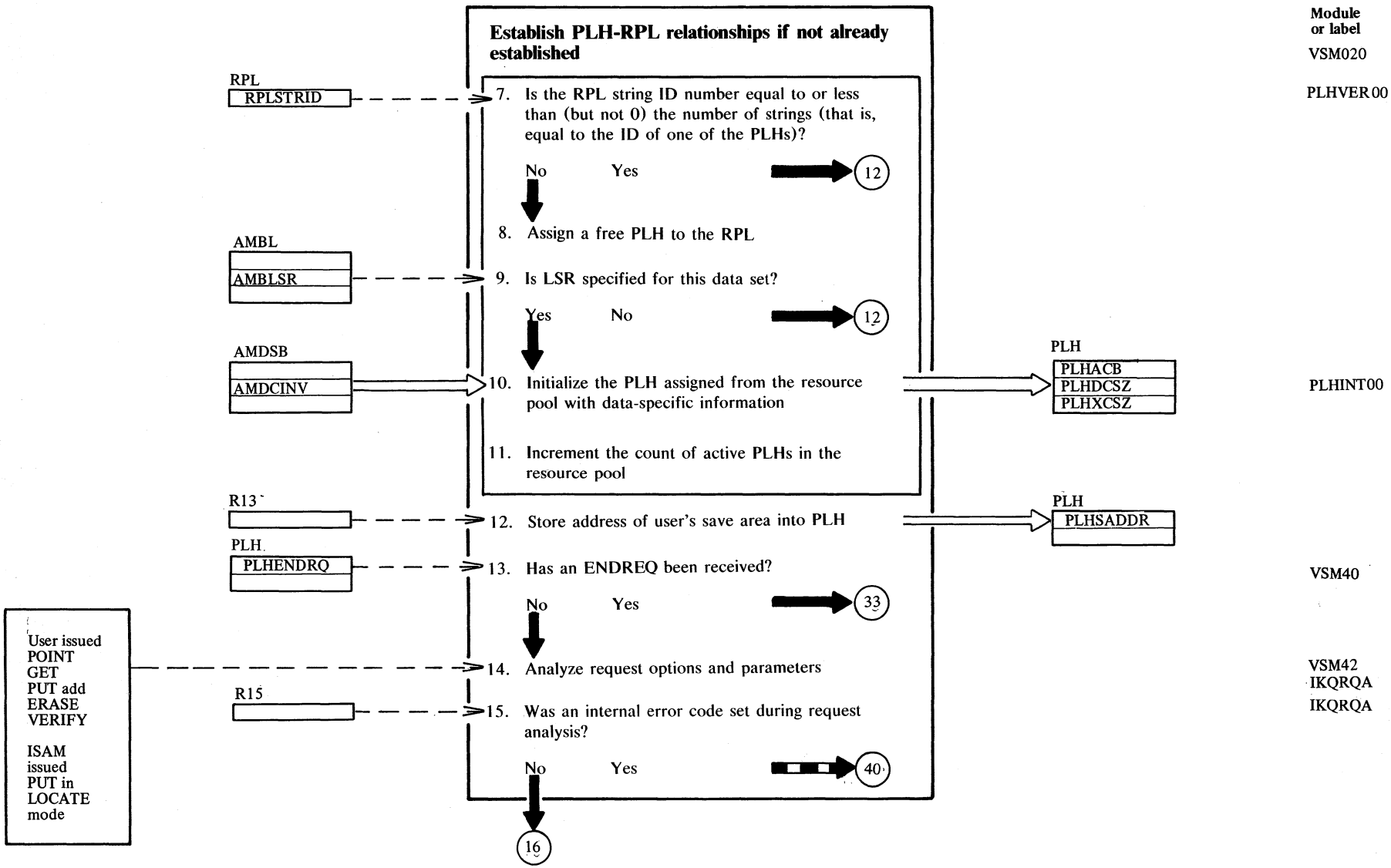


Diagram GB3. Record management overview

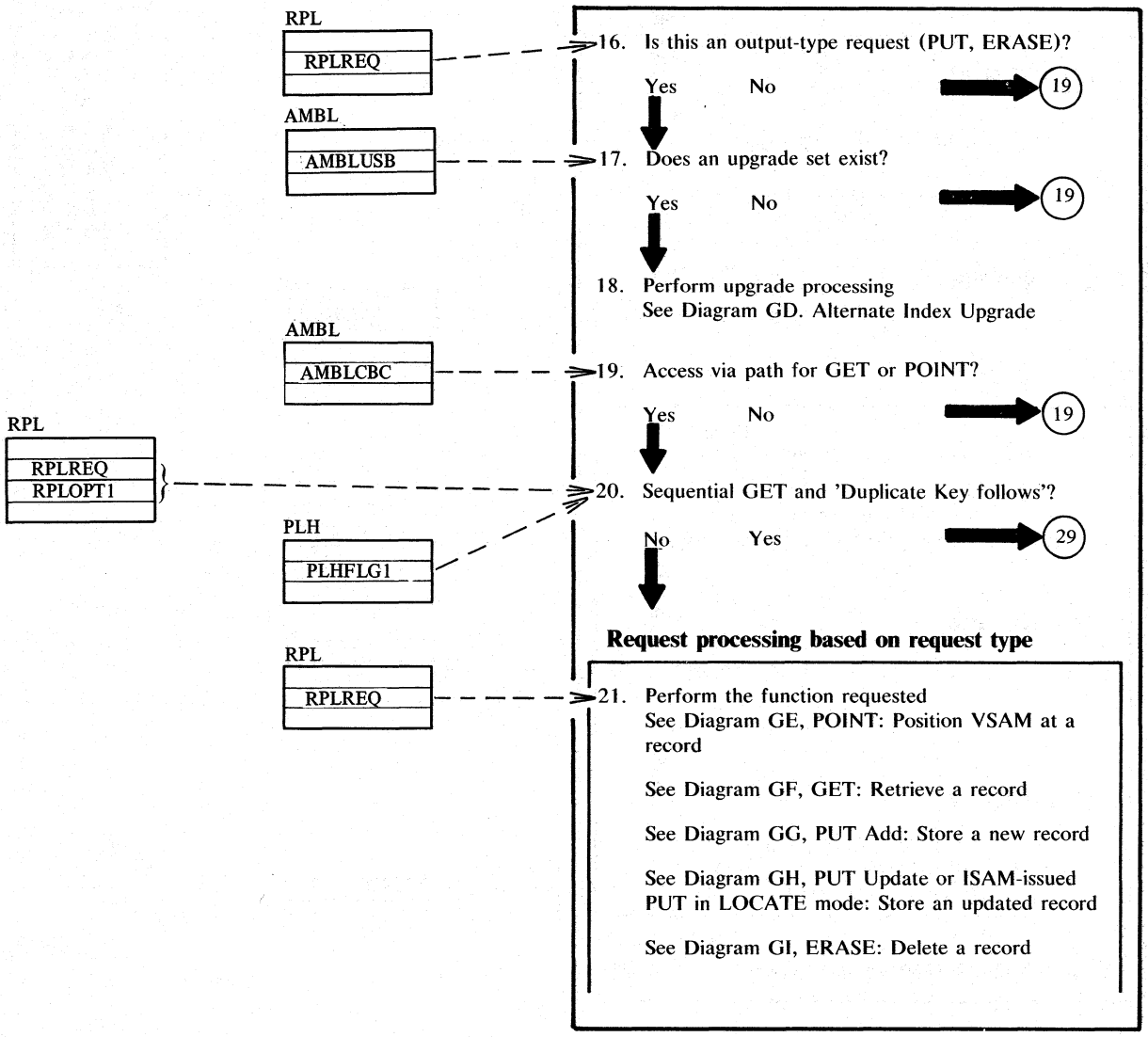


Diagram GB4. Record management overview

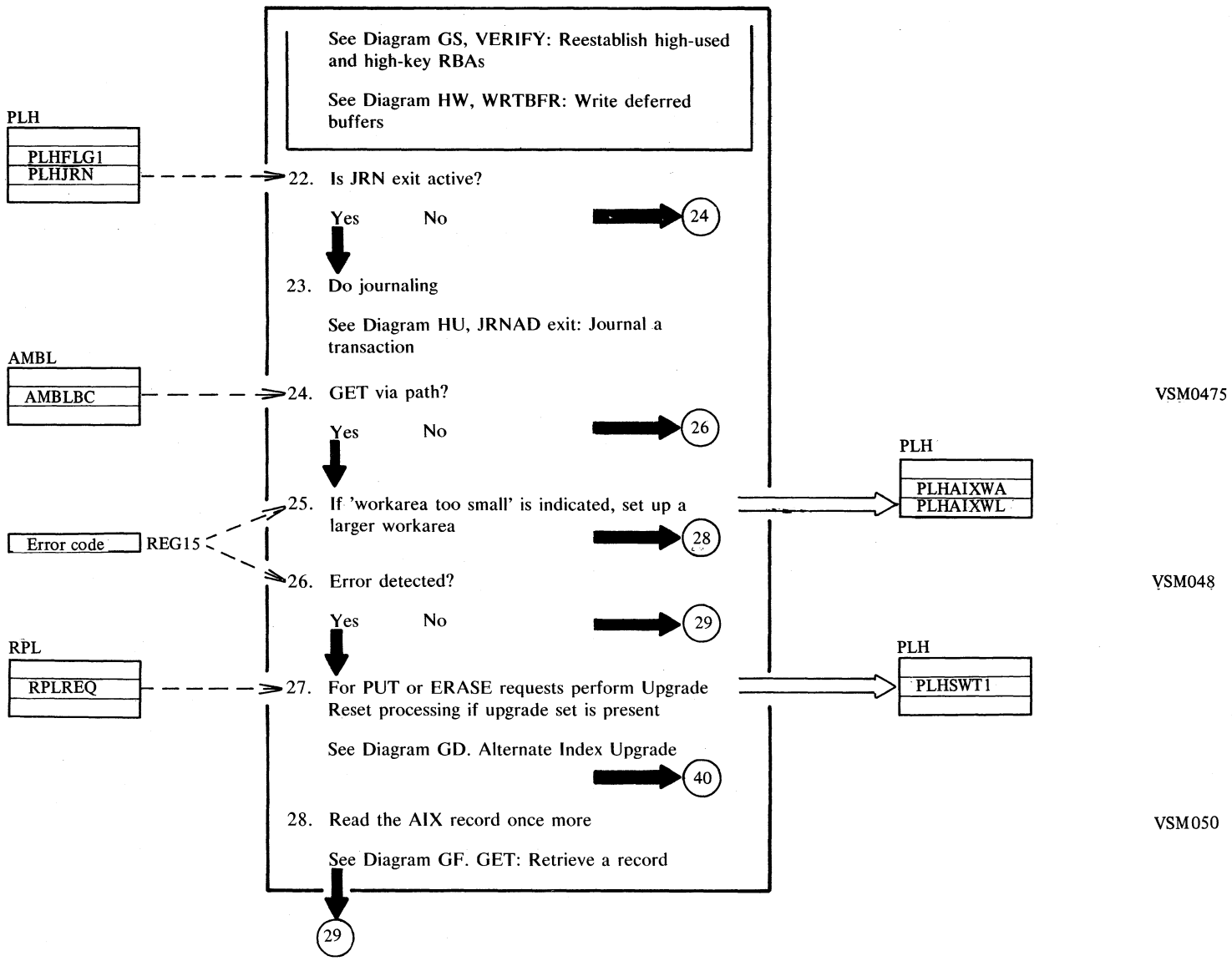


Diagram GB5. Record management overview

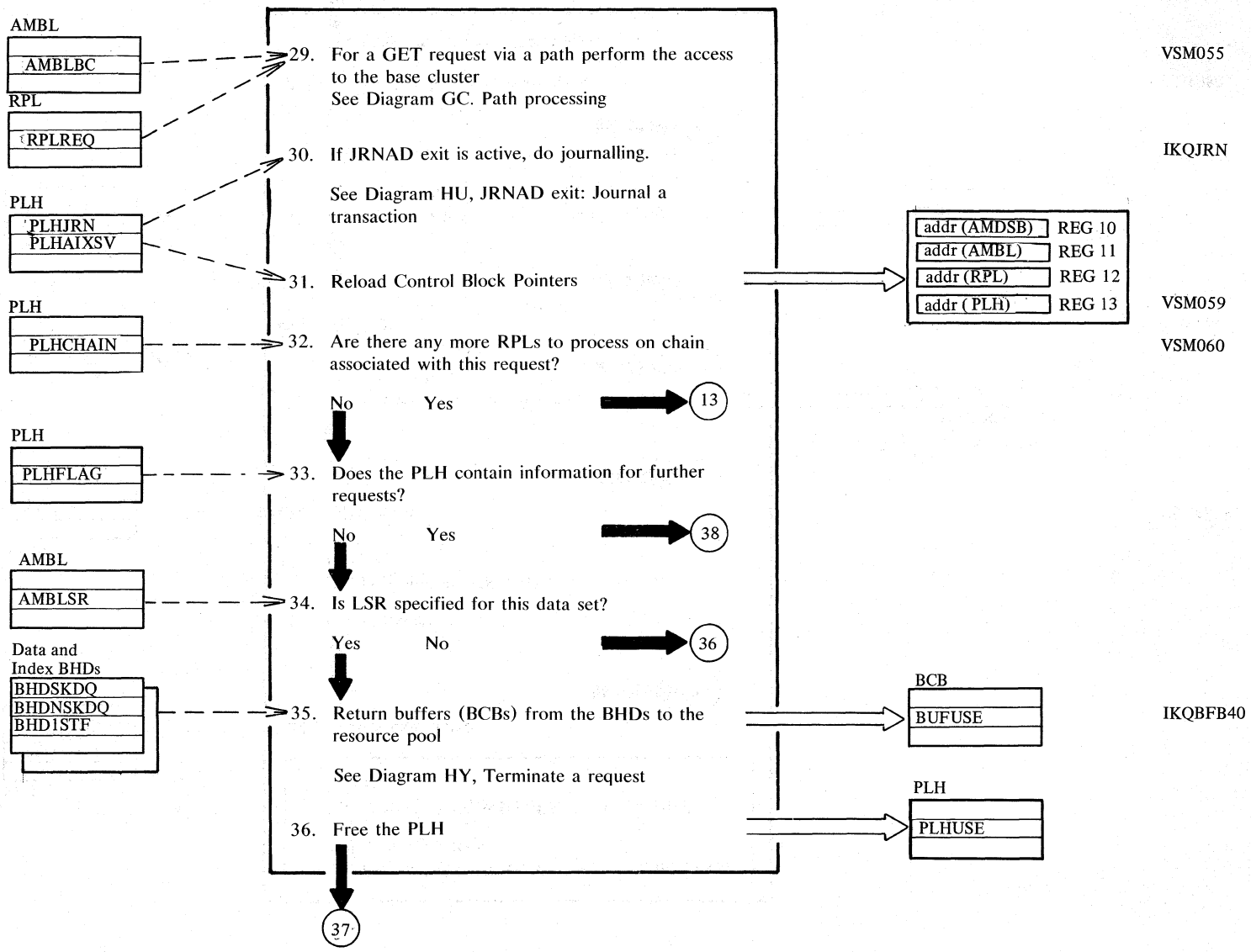


Diagram GB6. Record management overview

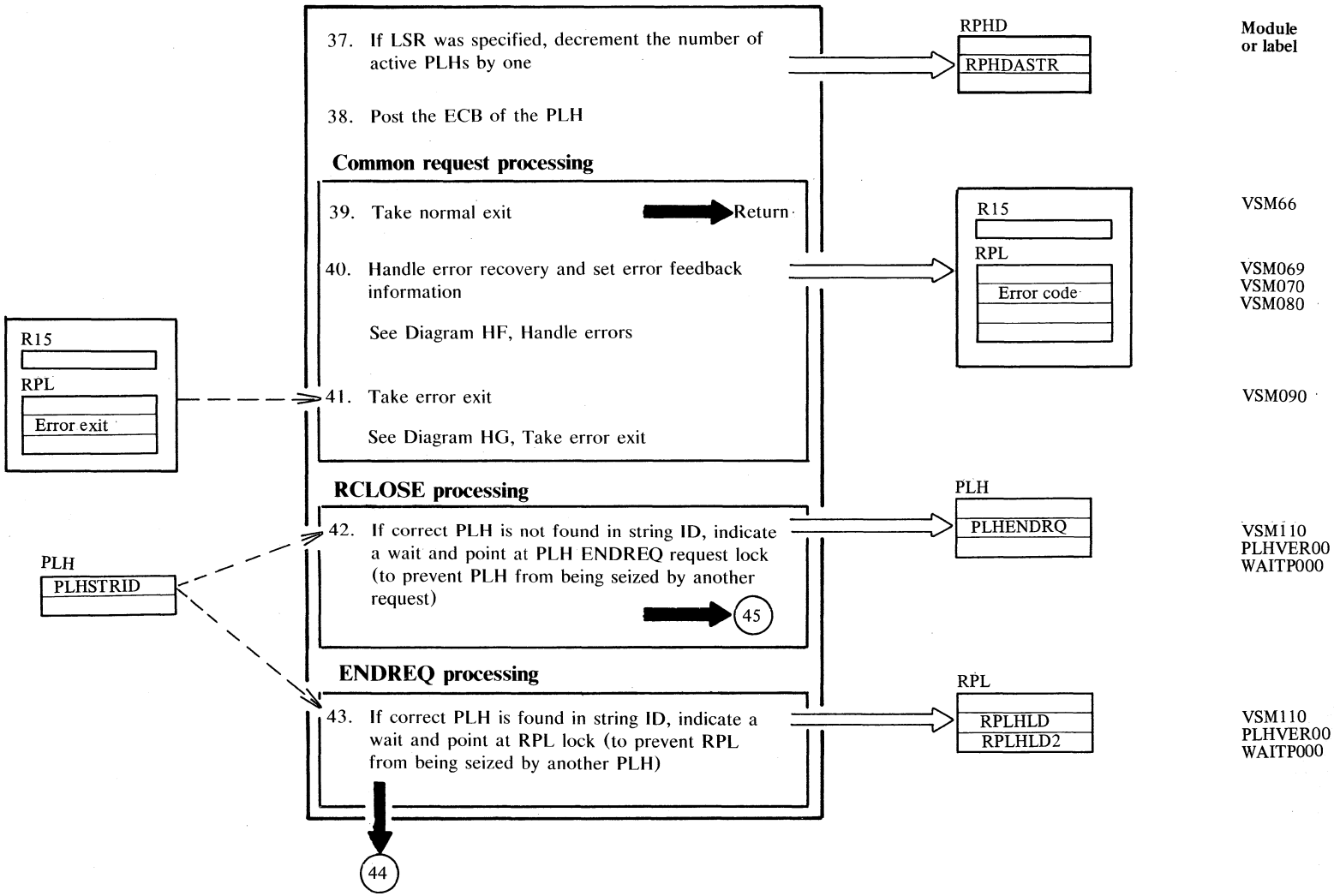
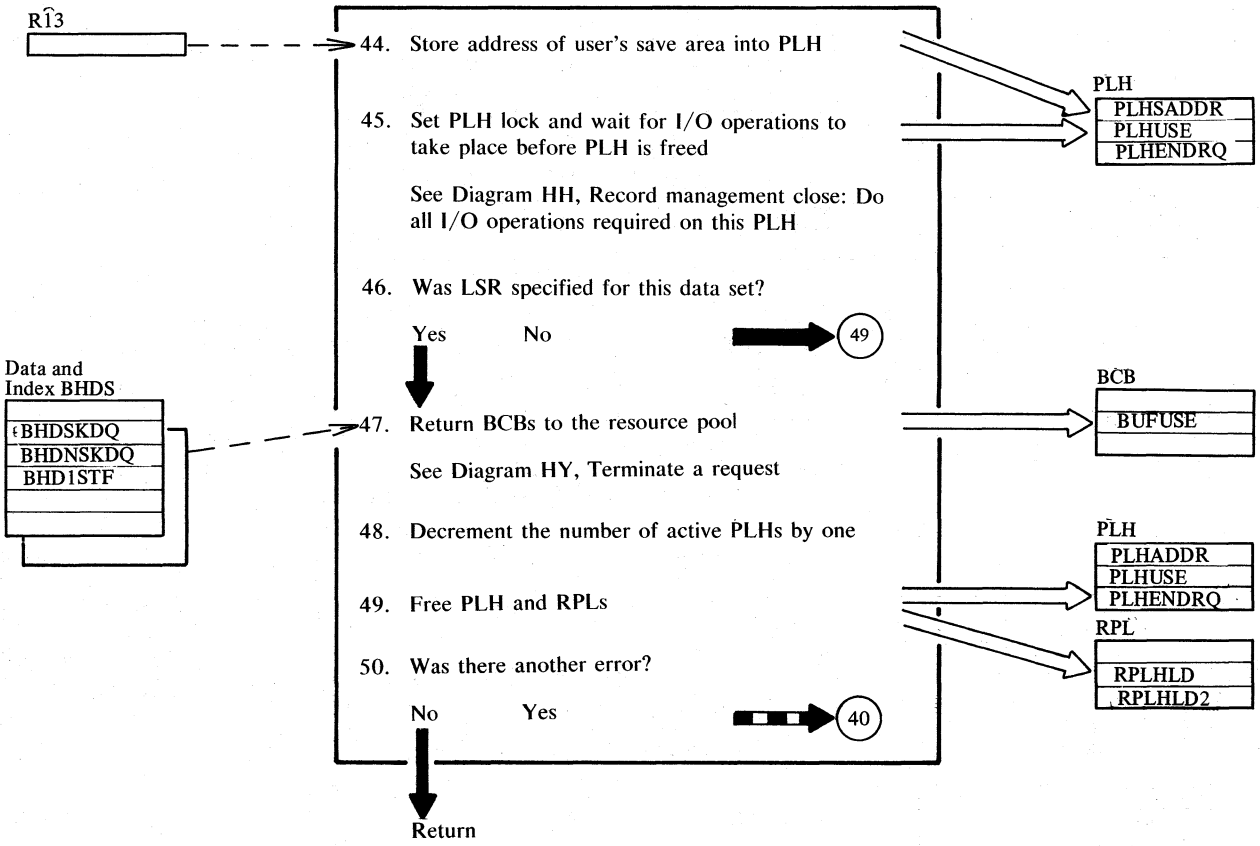


Diagram GB7. Record management overview



Module or label

VSM160

Diagram GC1. Path processing

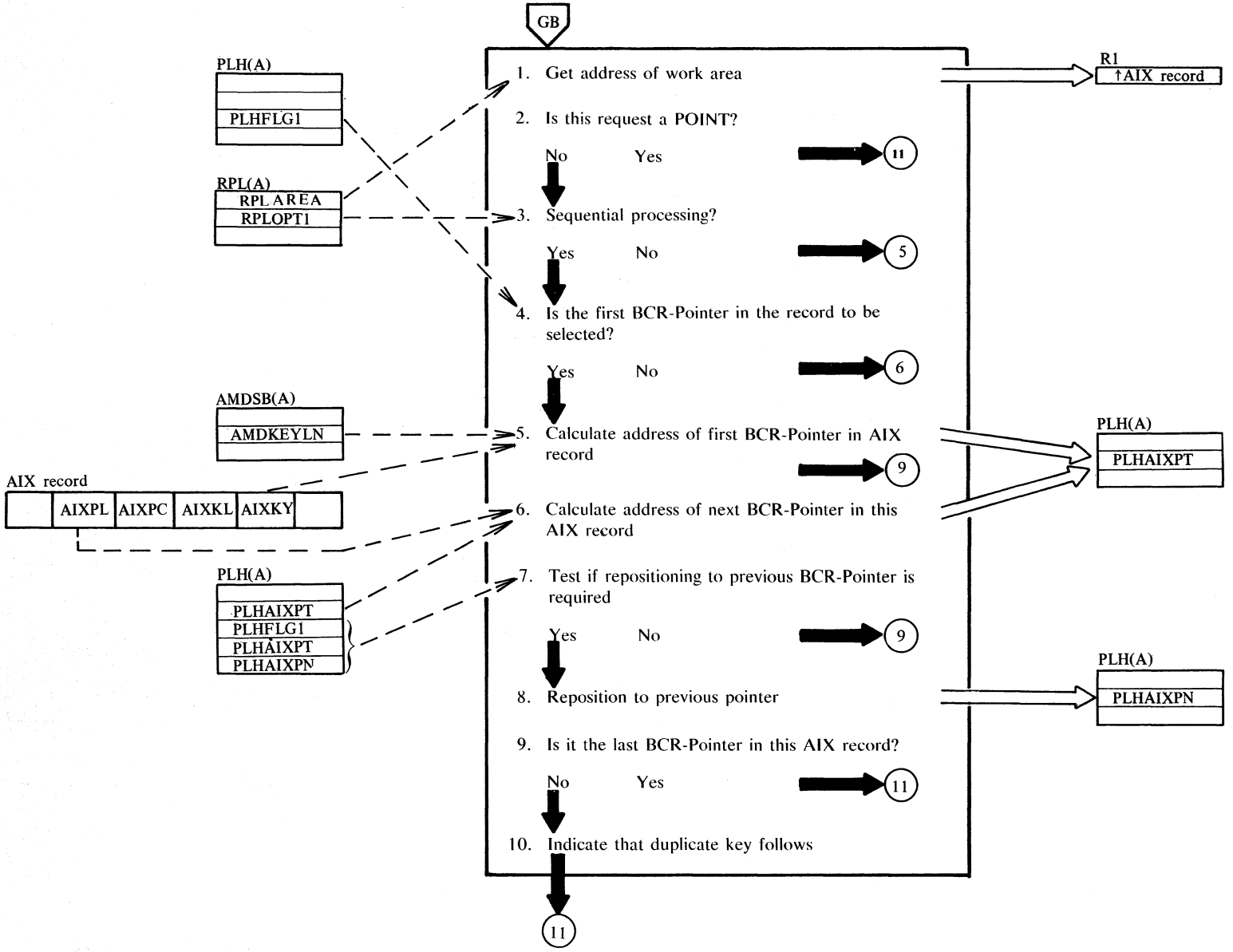


Diagram GC2. Path processing

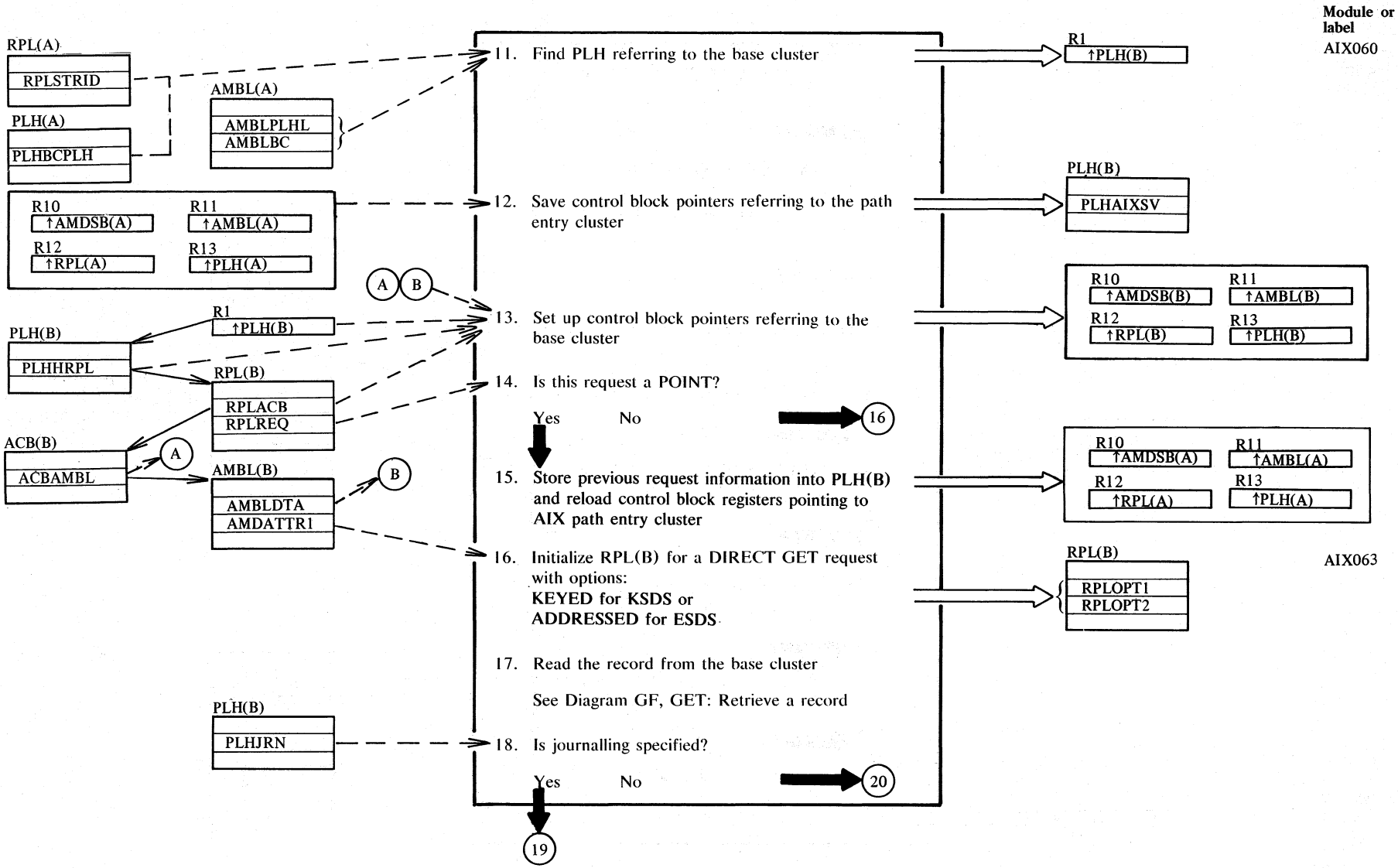
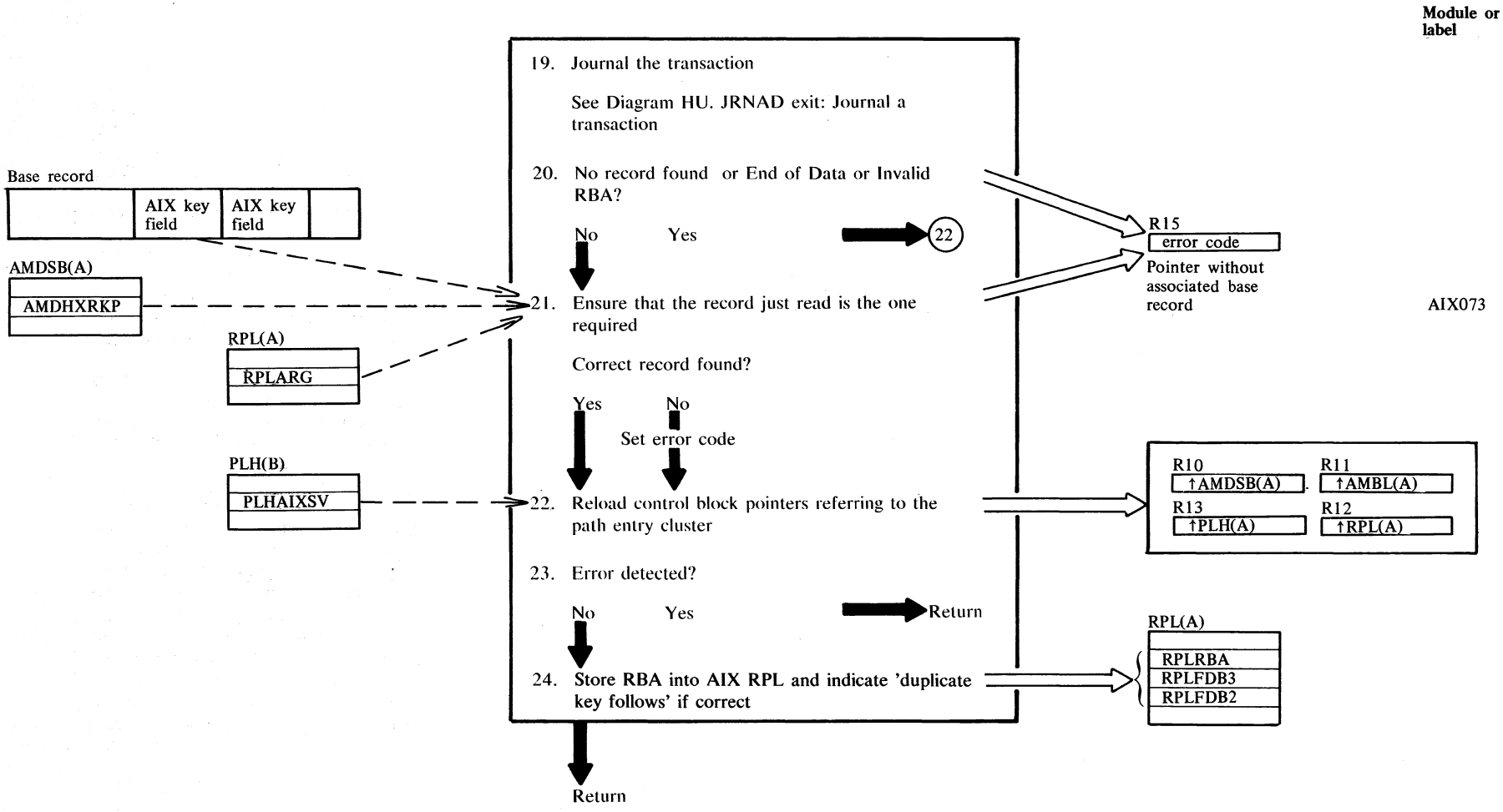


Diagram GC3. Path processing



Notes for Diagram GC

Description

3. If sequential processing is not specified, only the first base record associated with a given AIX key is retrieved. The indicator 'Duplicate key follows' is set if applicable.

7. Repositioning to the previous BCR pointer is required only if the previous sequential request ended with a 'no record found' condition in base cluster. To prevent the user from simply skipping this BCR pointer (which represents a lack of compatibility between AIX and base cluster), the PLH, which had been incremented in the previous request to point to the next BCR pointer, is returned to the faulty BCR pointer. A series of sequential GETs will thus return a series of 'no record found' conditions, all for the same base record.

11. A direct pointer (PLHBCPLH) is used for local shared resources. Otherwise the base cluster PLH is indexed by the AMBL (base cluster).

14. A POINT request is completed when the PLH is positioned to the correct BCR pointer. A GET request continues and actually retrieves the base cluster record indicated by the current BCR pointer.

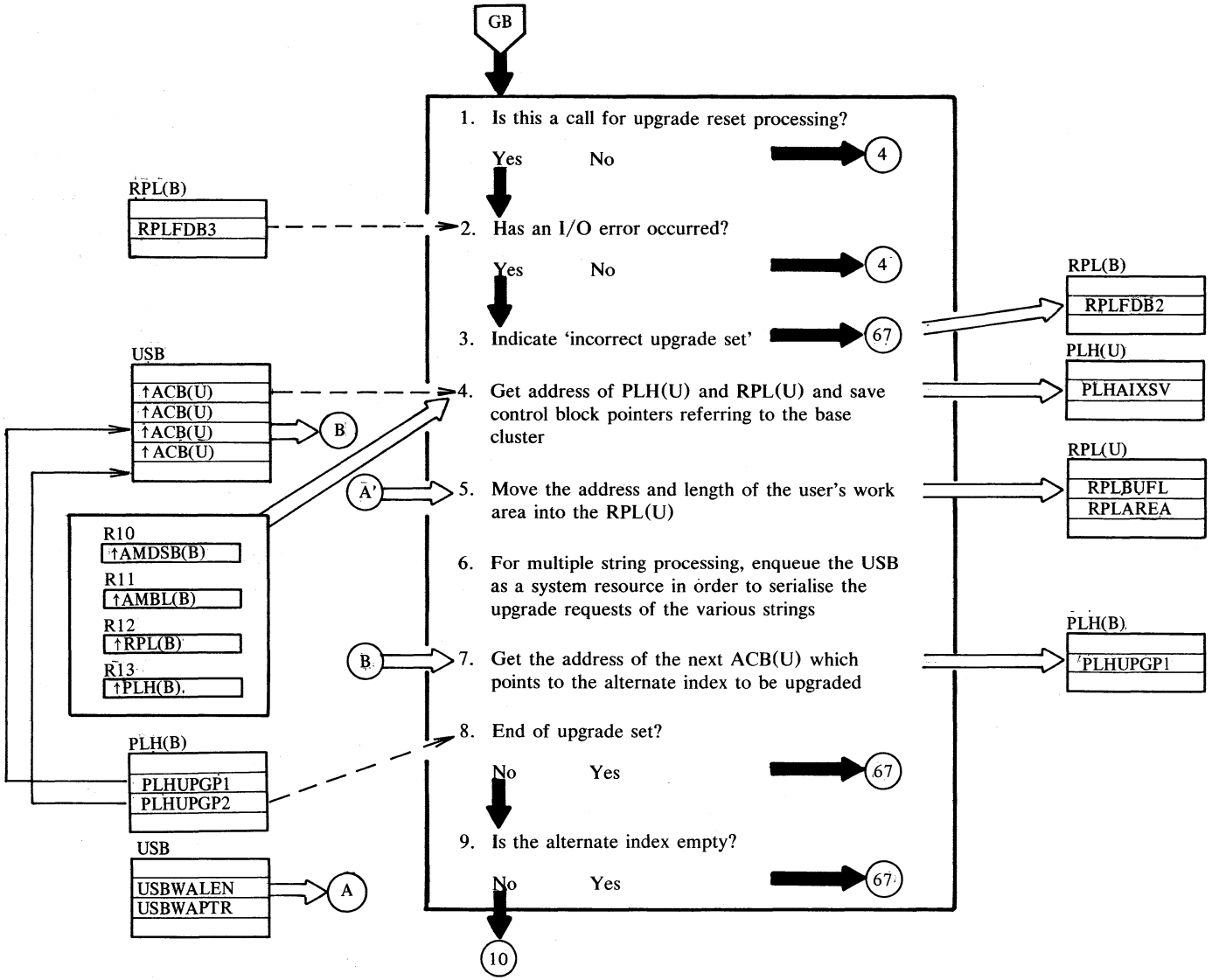
General note for sequential processing

During sequential processing, the base records are returned in the order of the BCR pointers in the AIX record, regardless of the 'direction' of sequential processing (forward or backward).

Control block notation:

RPL(A), PLH(A), etc: Control block referring to the path entry (AIX)
RPL(B), PLH(B), etc: control blocks referring to the base cluster.

Diagram GD1. Alternate index upgrade

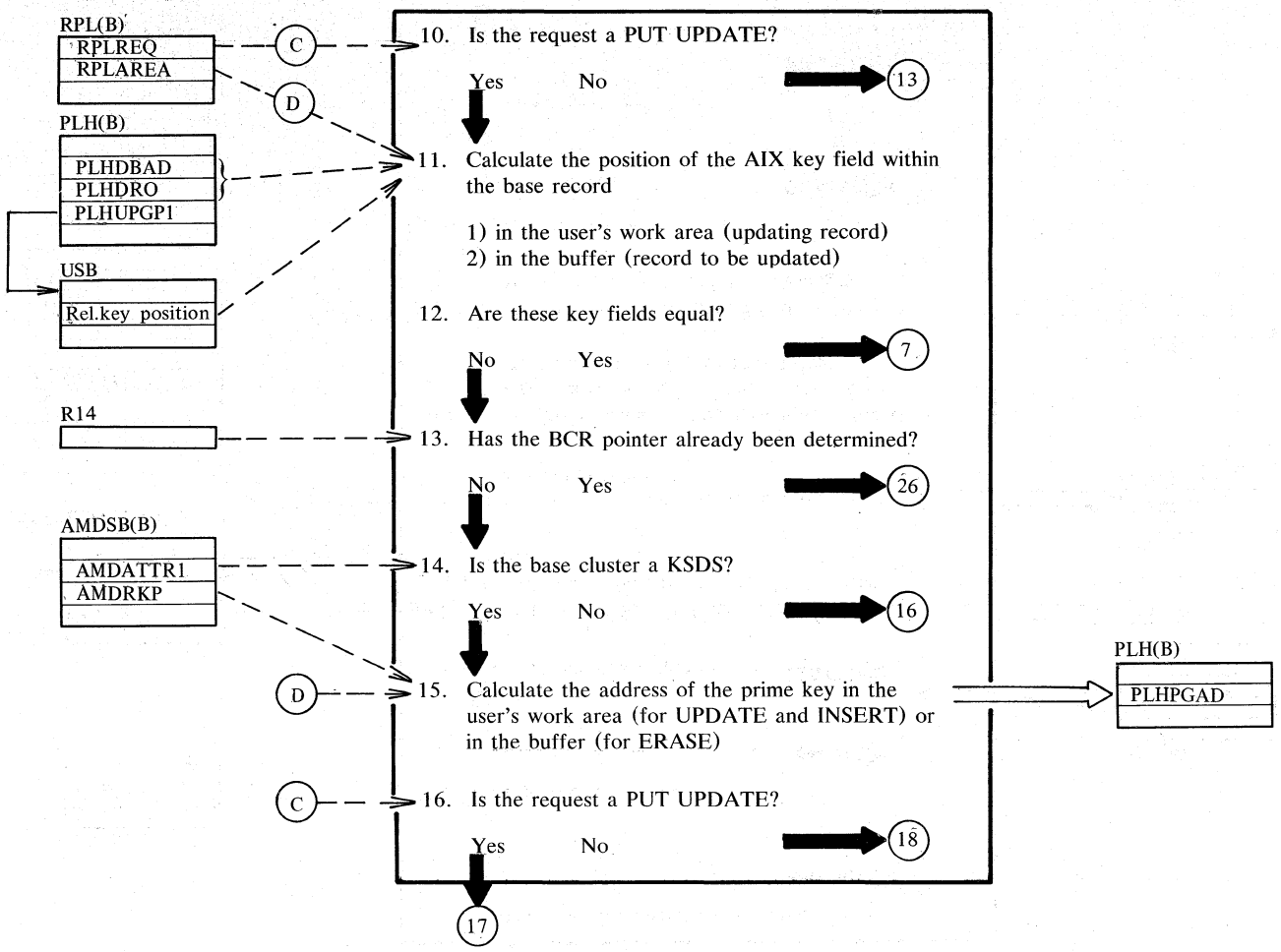


Module or label
IKQUPG
UPG0000

UPG0010

UPG1000

Diagram GD2. Alternate index upgrade



Module or label

UPG1020

UPG2030

Diagram GD3. Alternate index upgrade

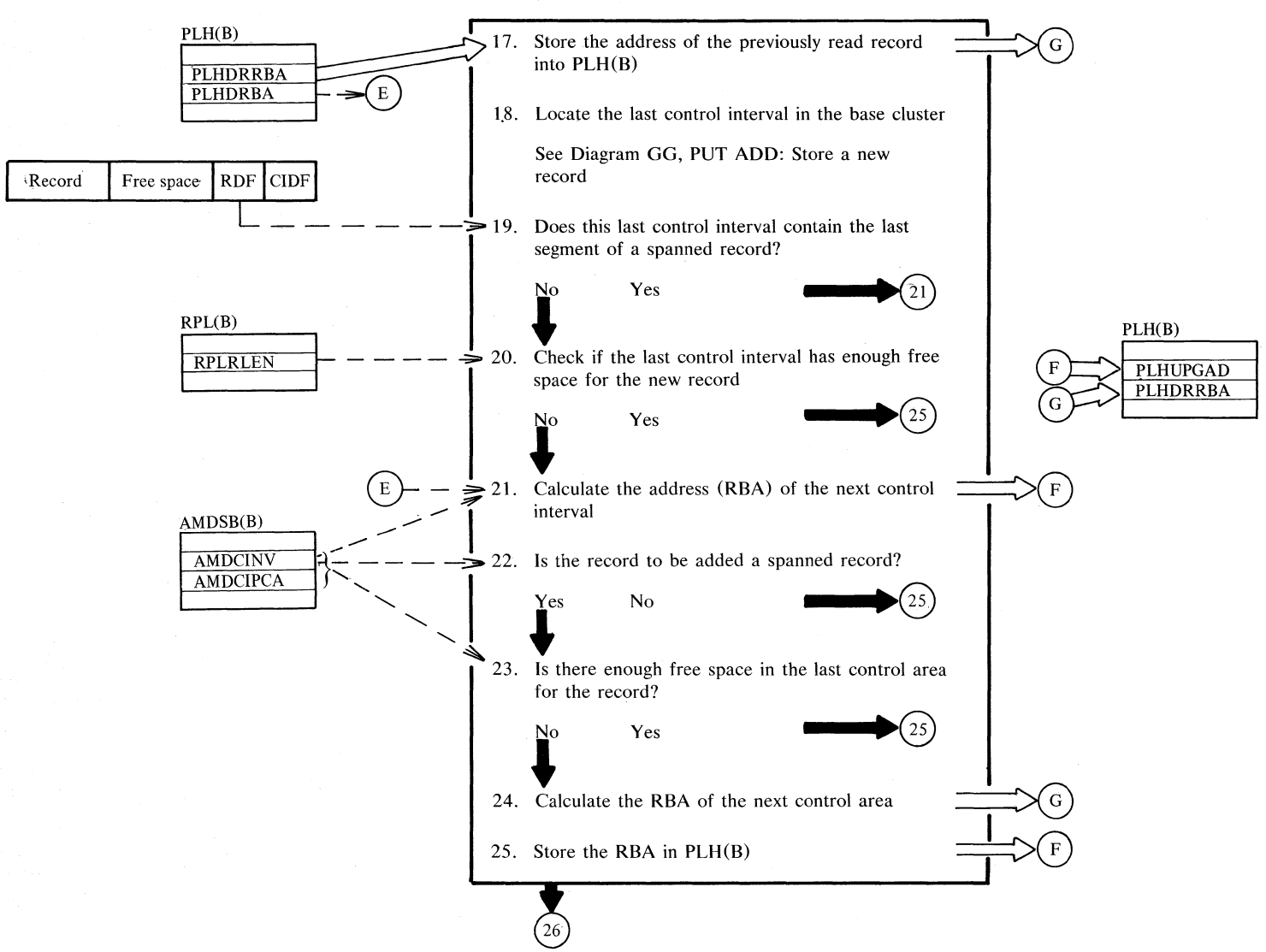


Diagram GD4. Alternate index upgrade

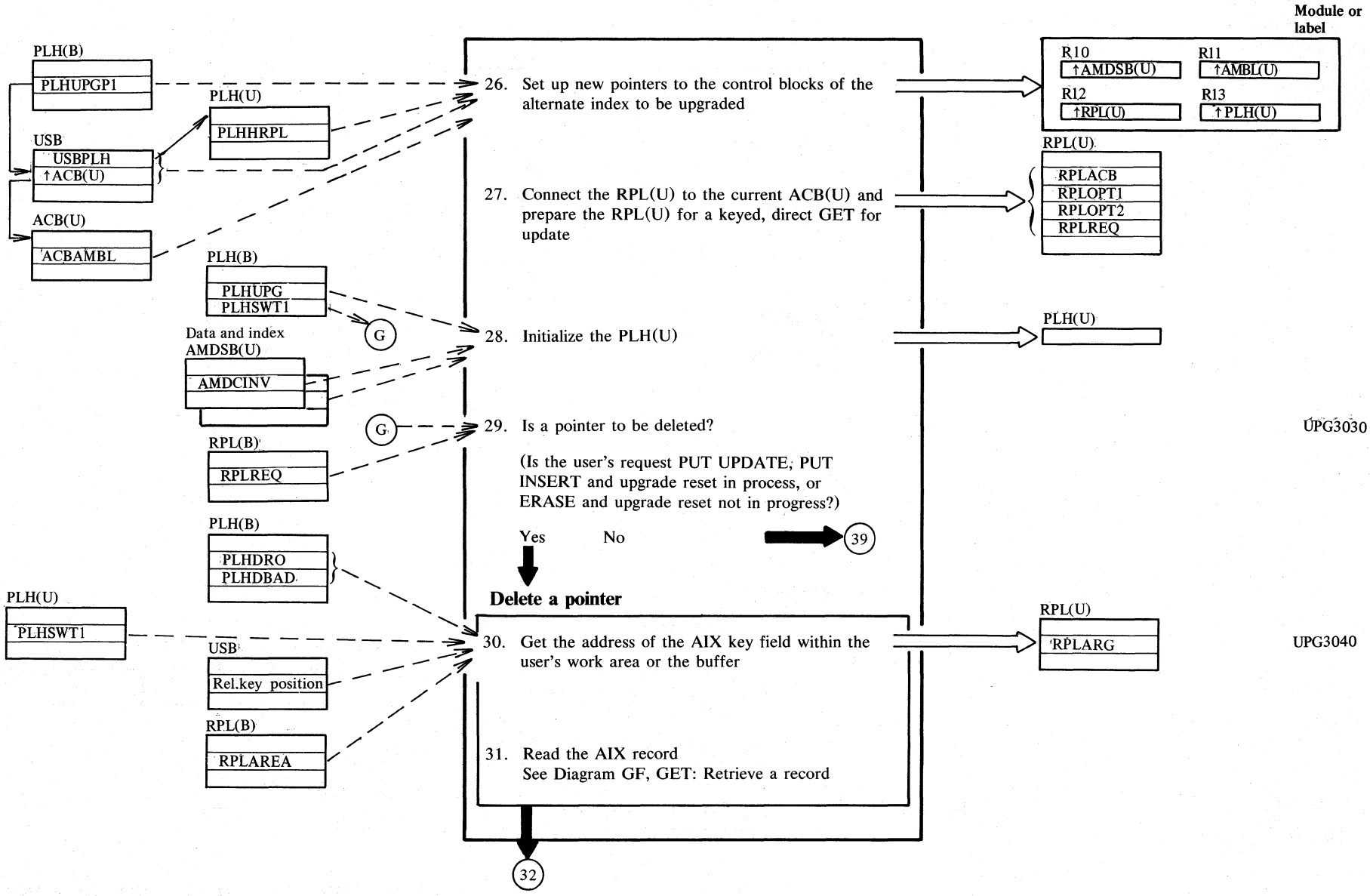


Diagram GD5. Alternate index upgrade

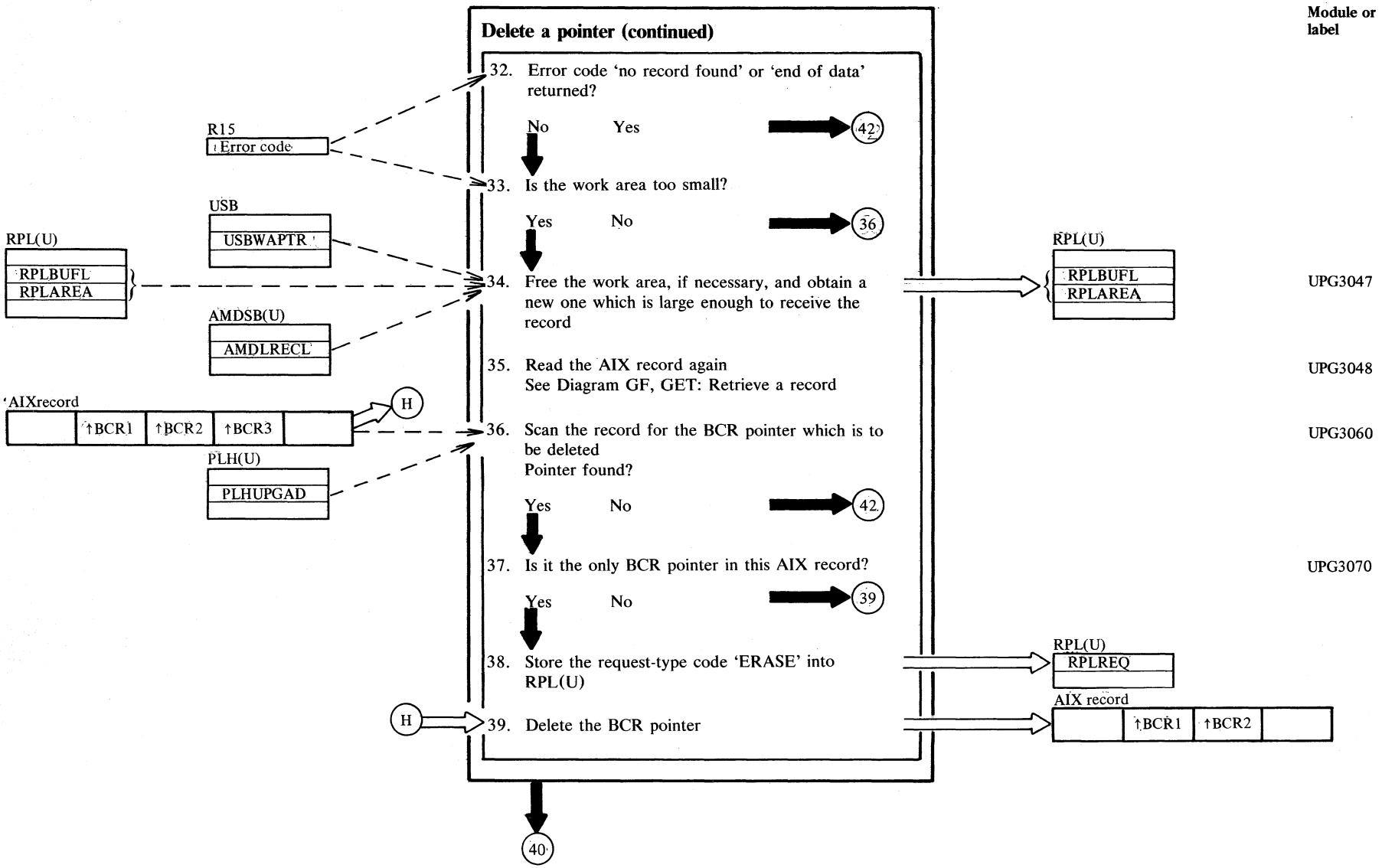


Diagram GD6. Alternate index upgrade

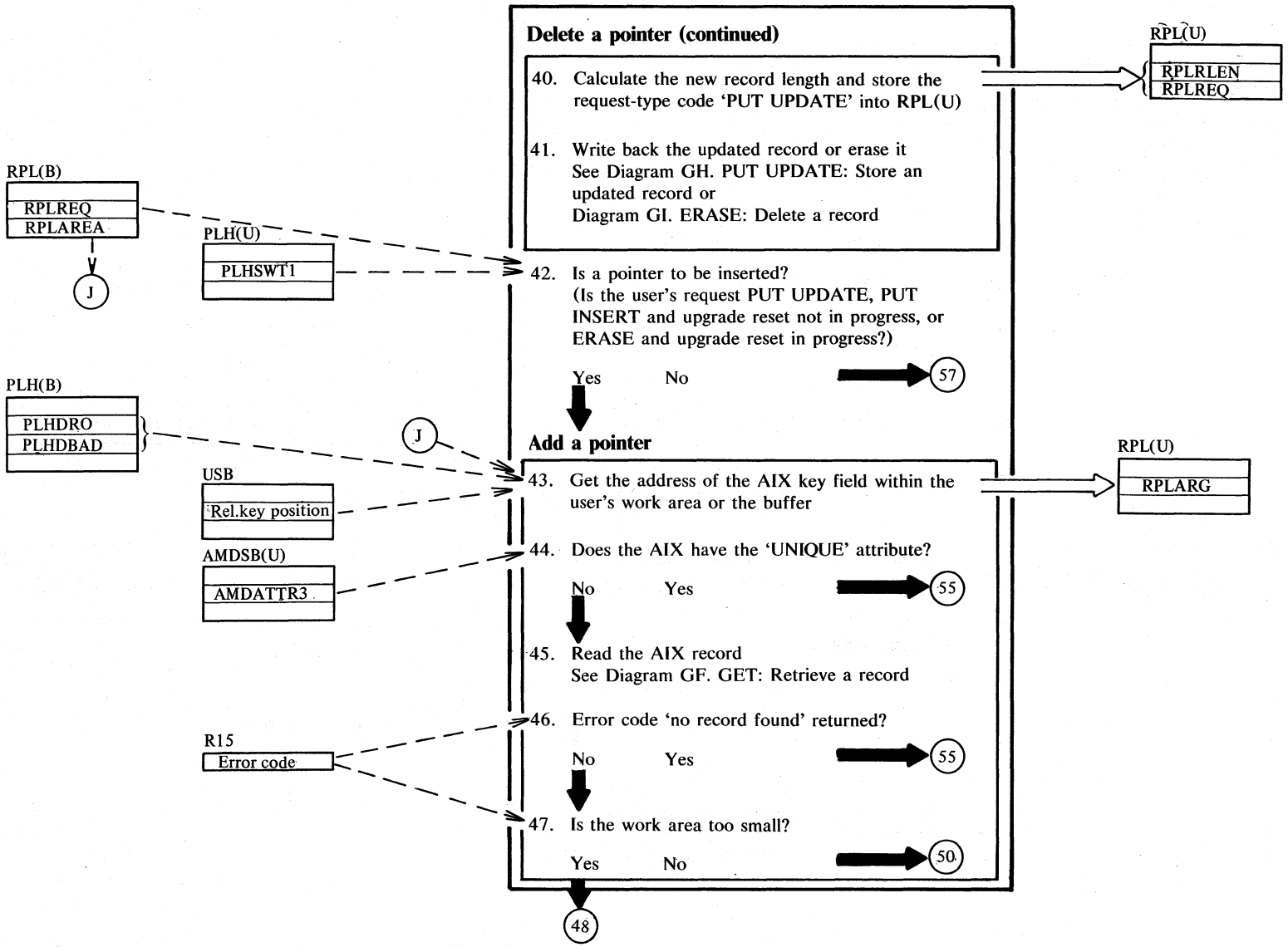


Diagram GD7. Alternate index upgrade

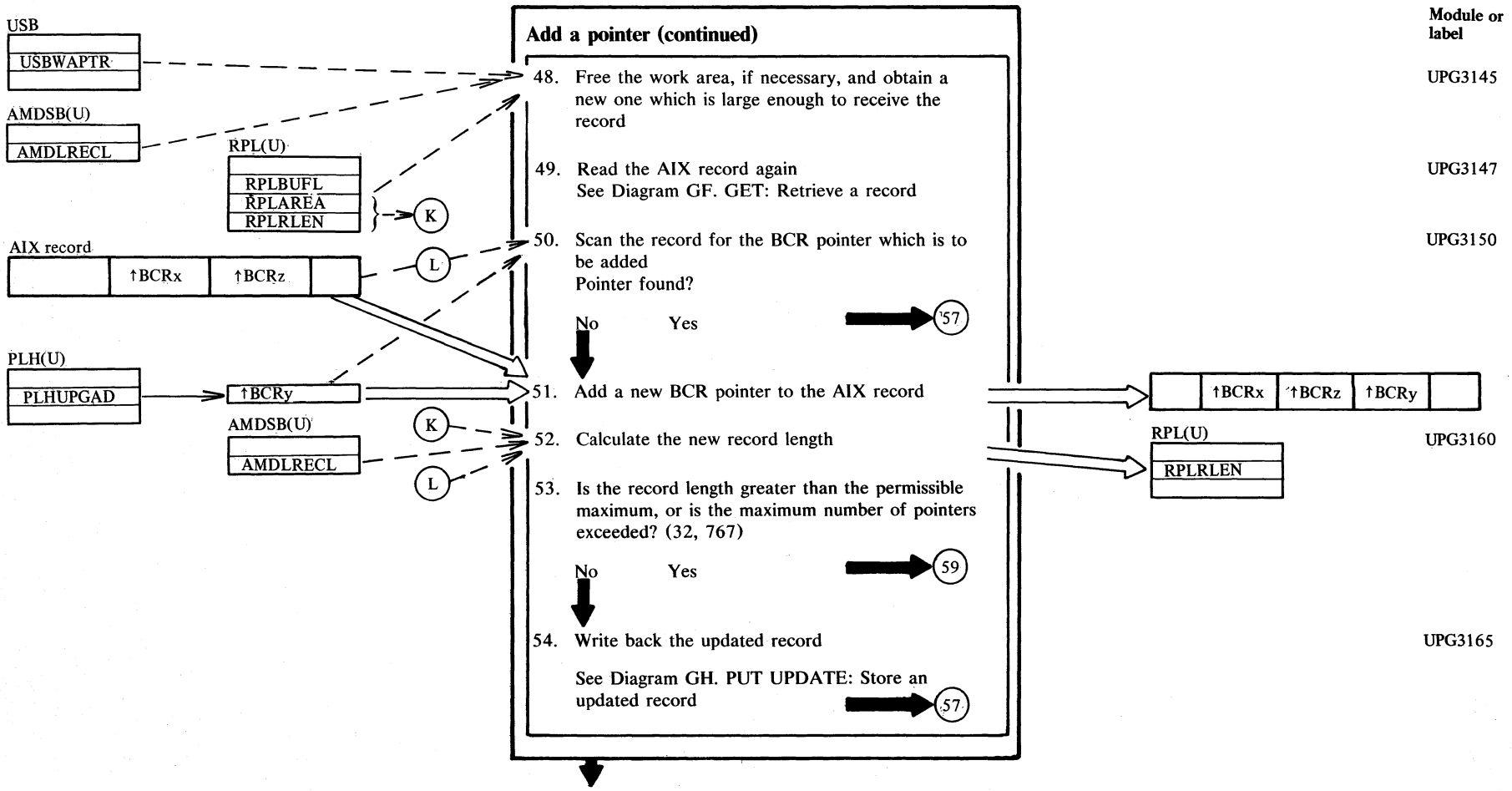


Diagram GD8. Alternate index upgrade

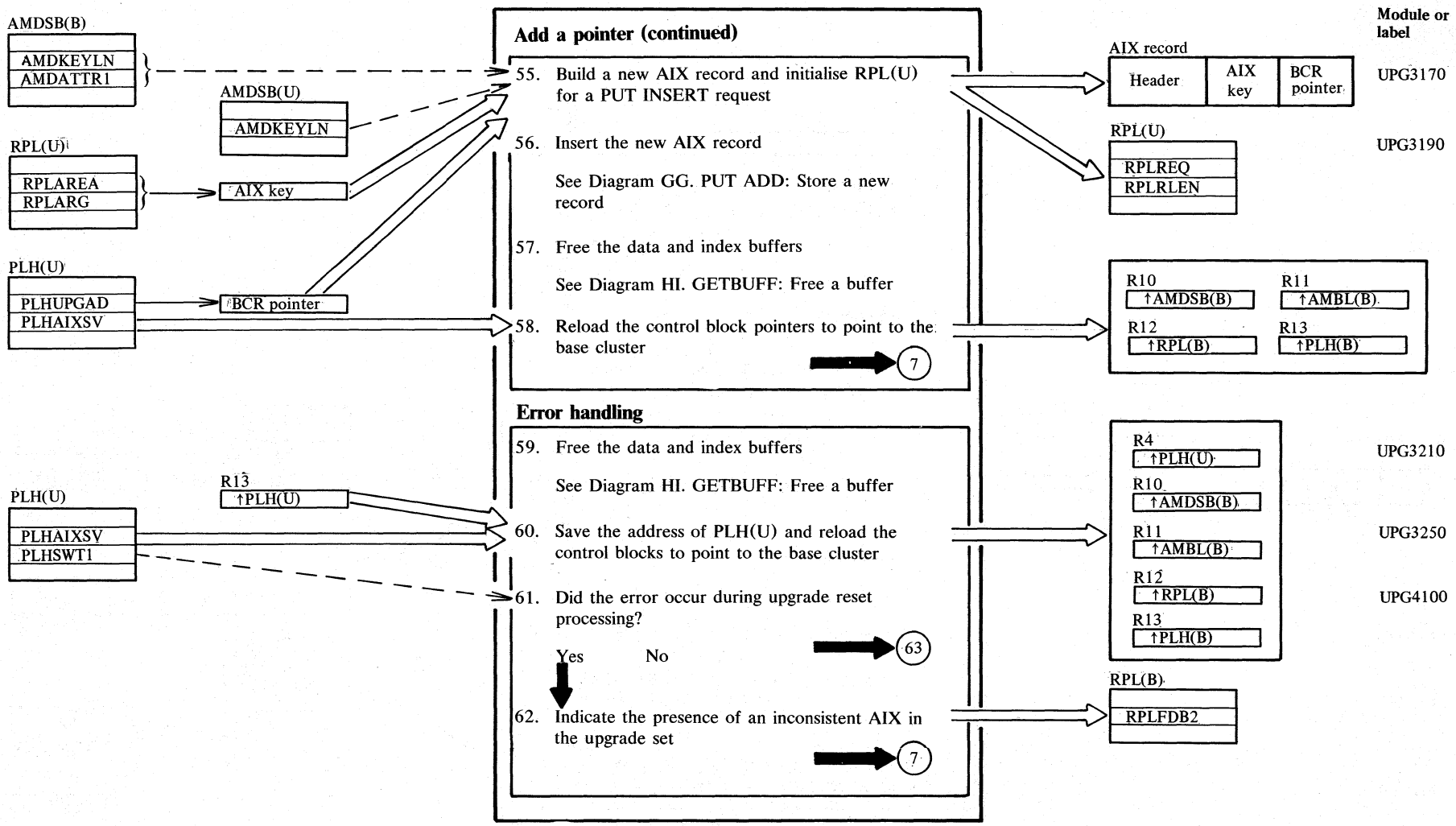


Diagram GD9. Alternate index upgrade

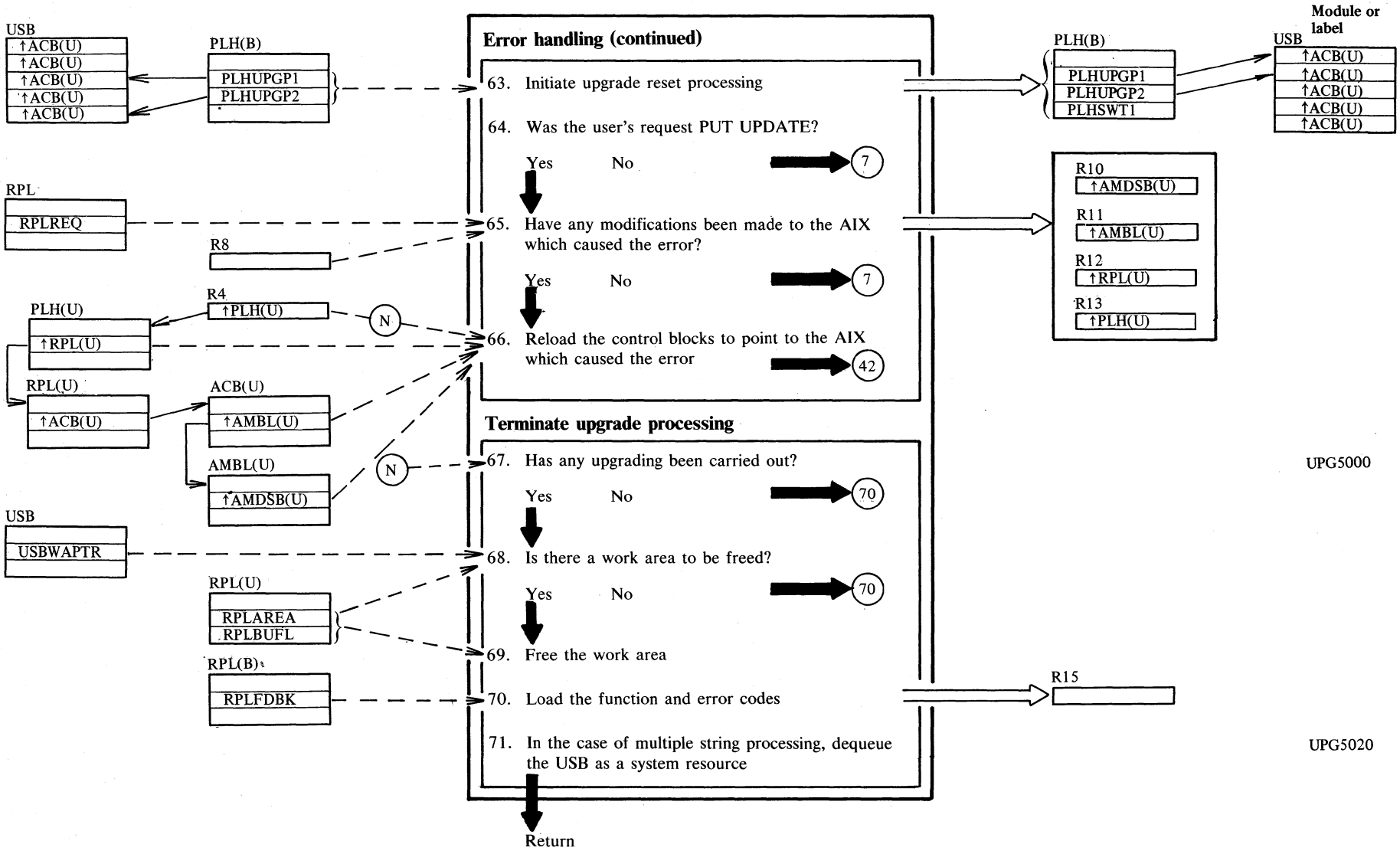
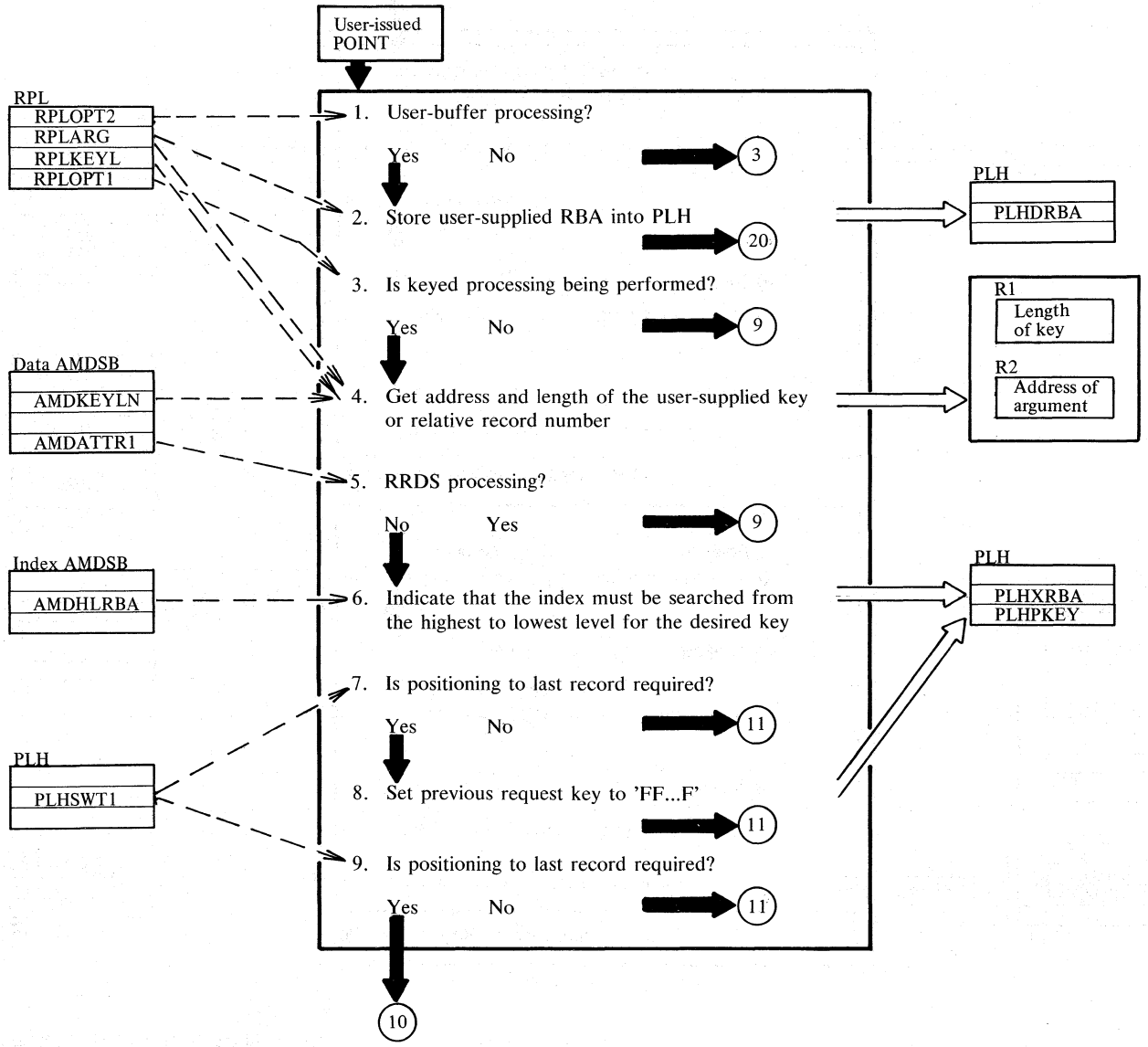


Diagram GE1. POINT: Position VSAM data record



Module

IKQGPT

GPT030

GPT060

GPT065

Diagram GE2. POINT: Position VSAM data record

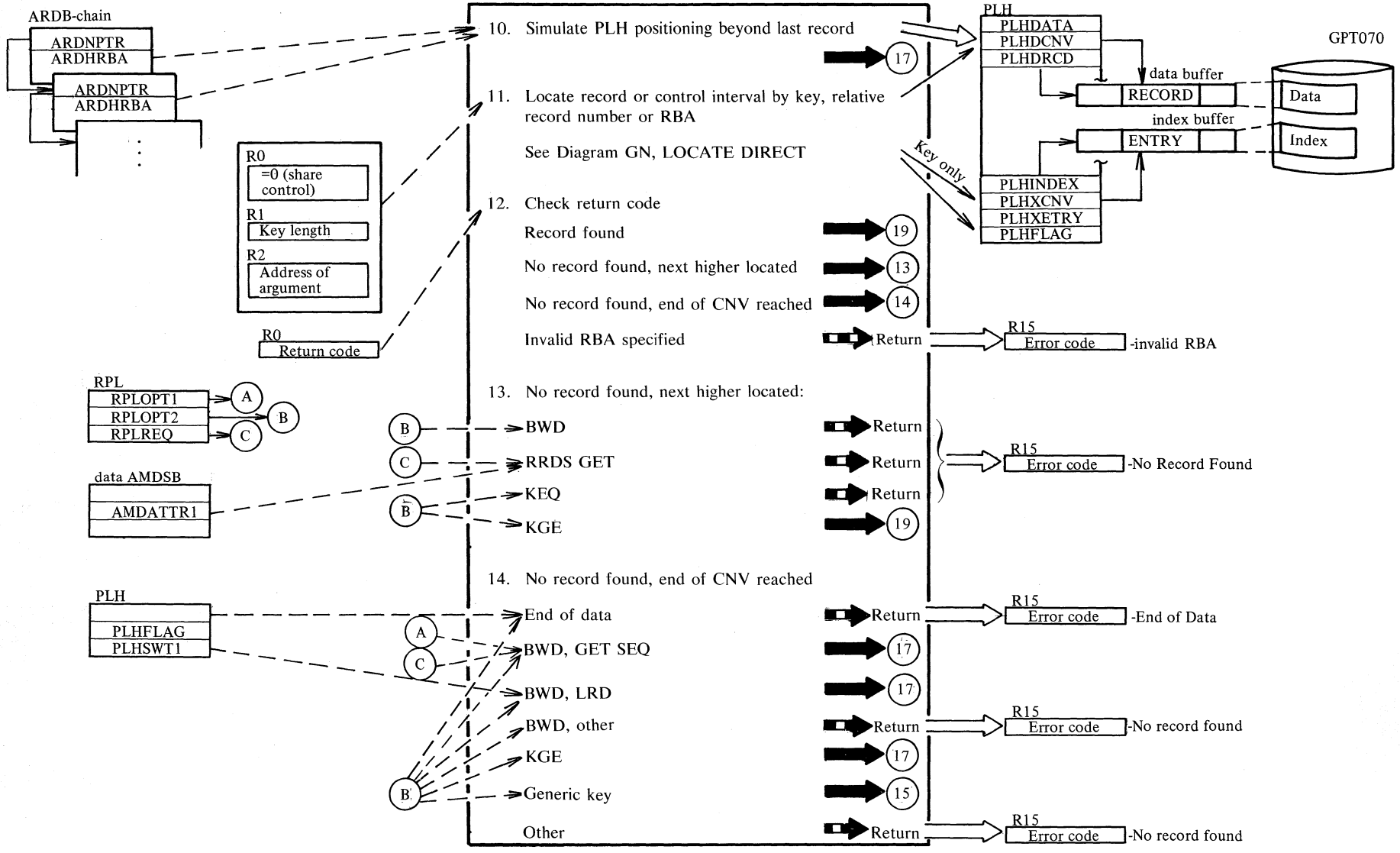


Diagram GE3. POINT: Position VSAM data record

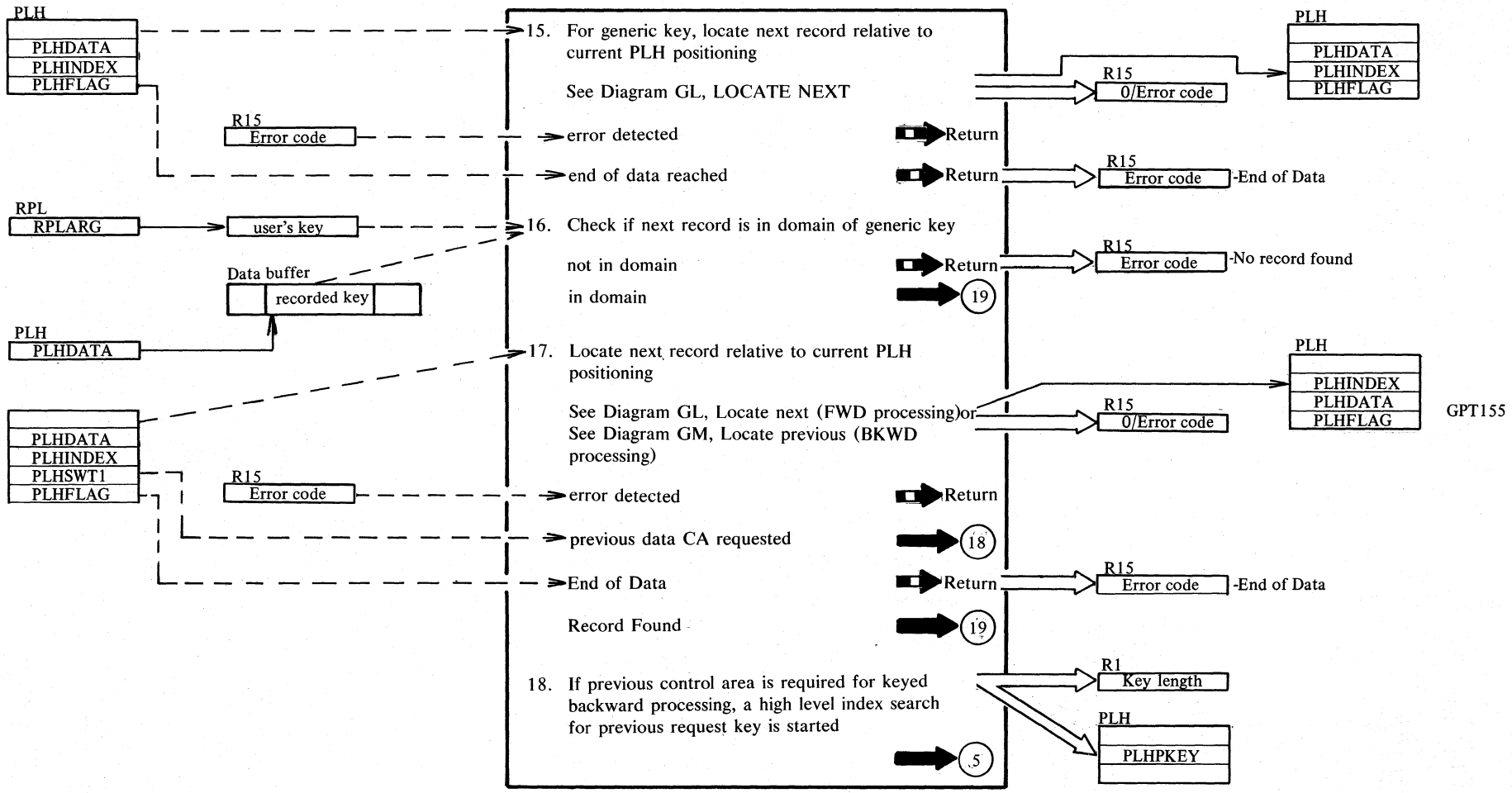
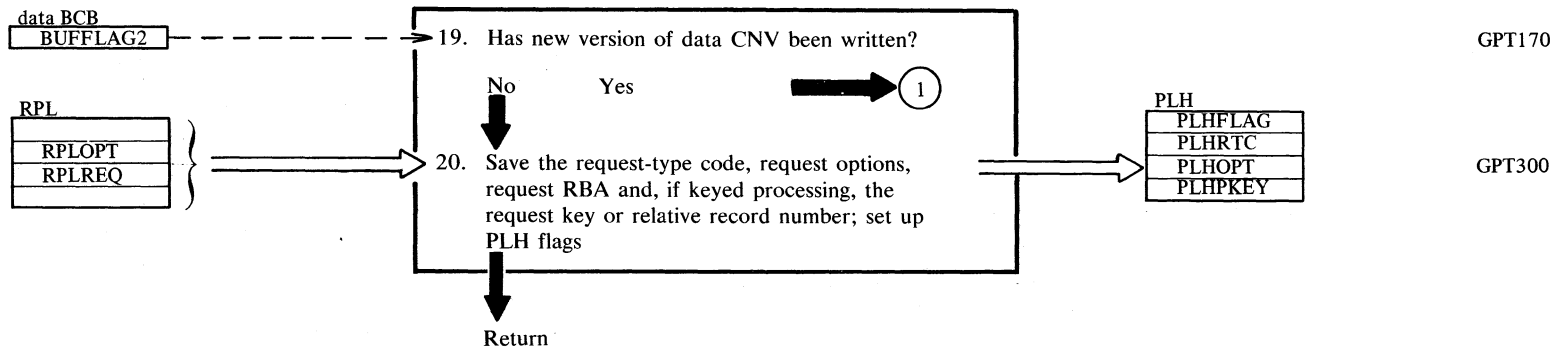


Diagram GE4. POINT: Position VSAM data record



Notes:

4 The user-supplied argument is a key for keyed processing or a 4-byte relative record number for relative record processing.

7-9 Positioning to last record is required for backward processing with the option LRD.

8 For keyed processing of a KSDS, positioning to the last record is achieved by searching the index for the maximum possible key (x'FF...FF')

9-10 For addressed or relative record processing, positioning past the last record is first carried out, using the high water mark of the highest key range as an RBA. The last record is then found by using LOCATE PREVIOUS.

17-18 During keyed backward processing of a KSDS, LOCATE PREVIOUS can step backwards only until it reached the beginning of a sequence set record, which corresponds to the start of a data CA. LOCATE PREVIOUS then returns a 'previous data CA required' condition, and stores the lowest key of the current CA in the previous request key field. LOCATE DIRECT and INDEX SEARCH will then carry out a top-down search and transition to the previous sequence set record.

Diagram GF1. GET: Retrieve a record

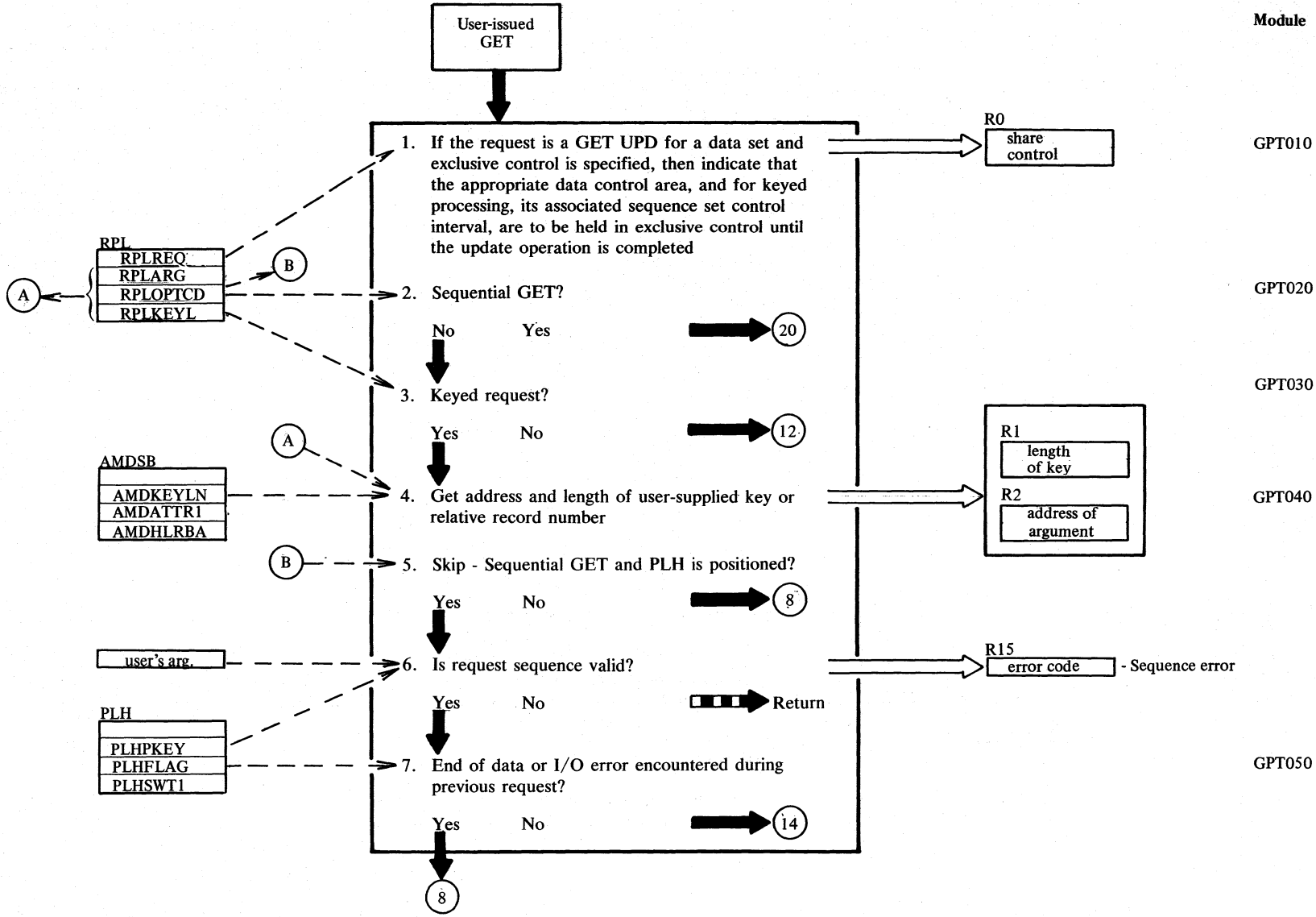


Diagram GF2. GET: Retrieve a record

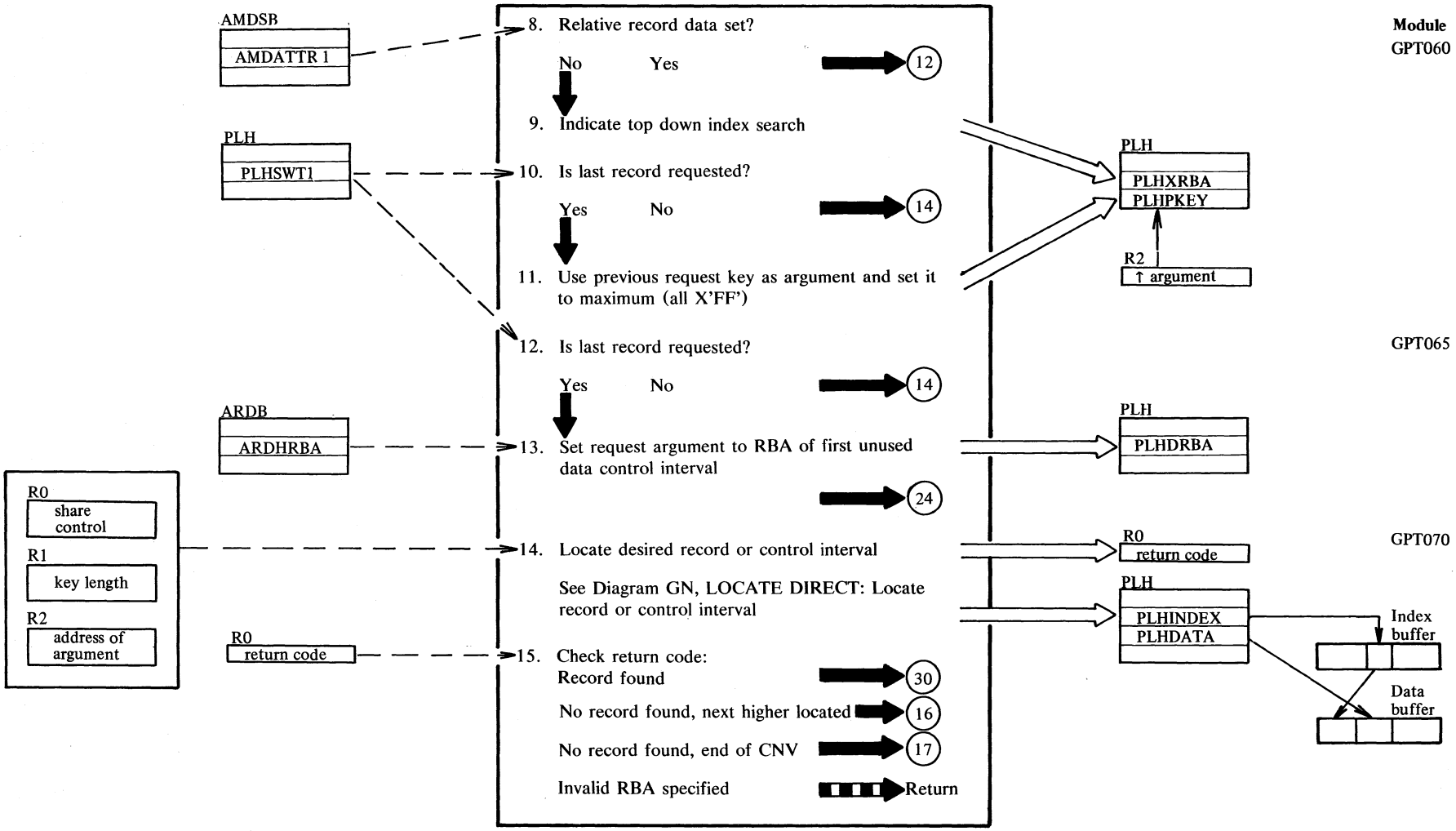


Diagram GF3. GET: Retrieve a record

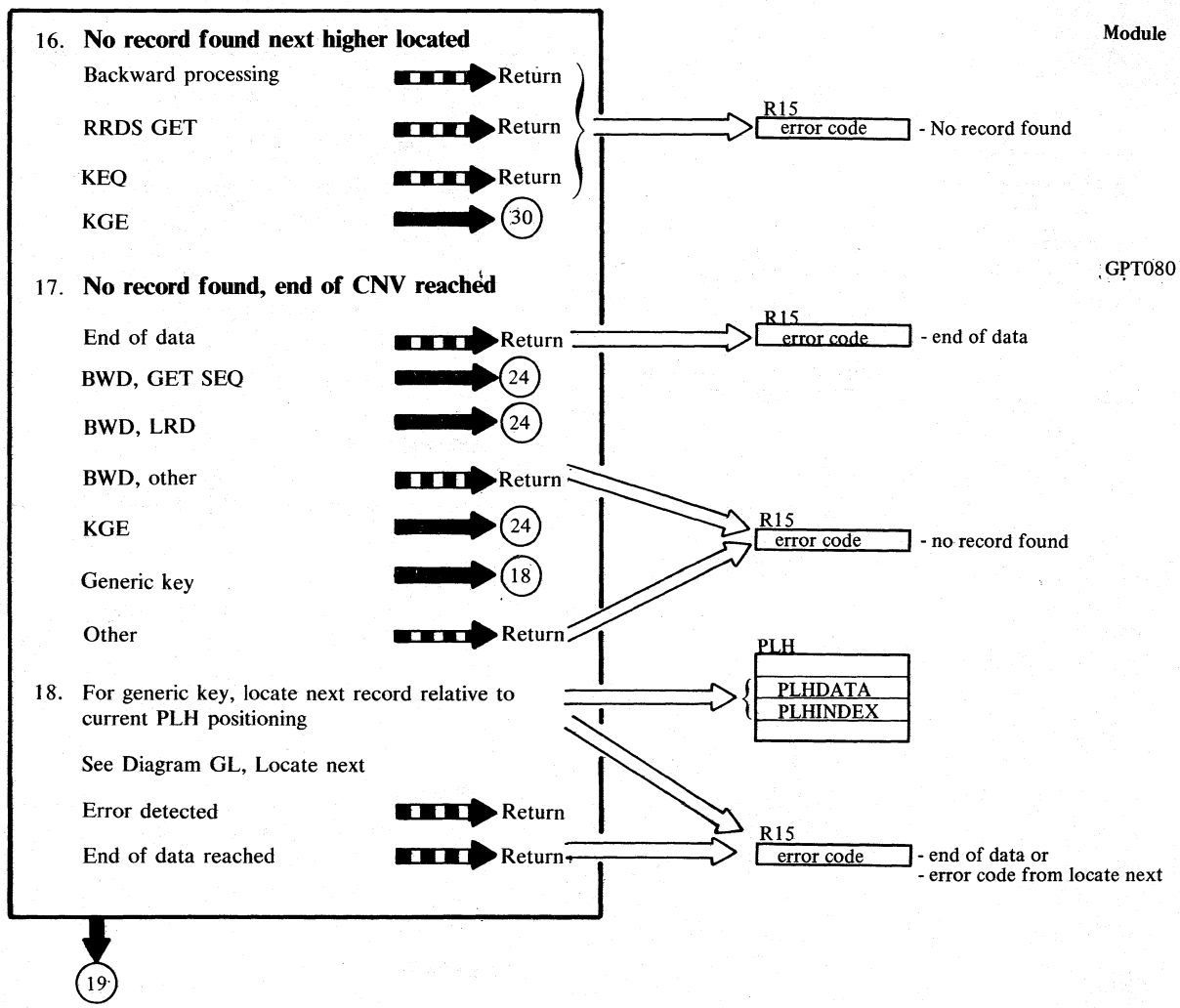


Diagram GF4. GET: Retrieve a record

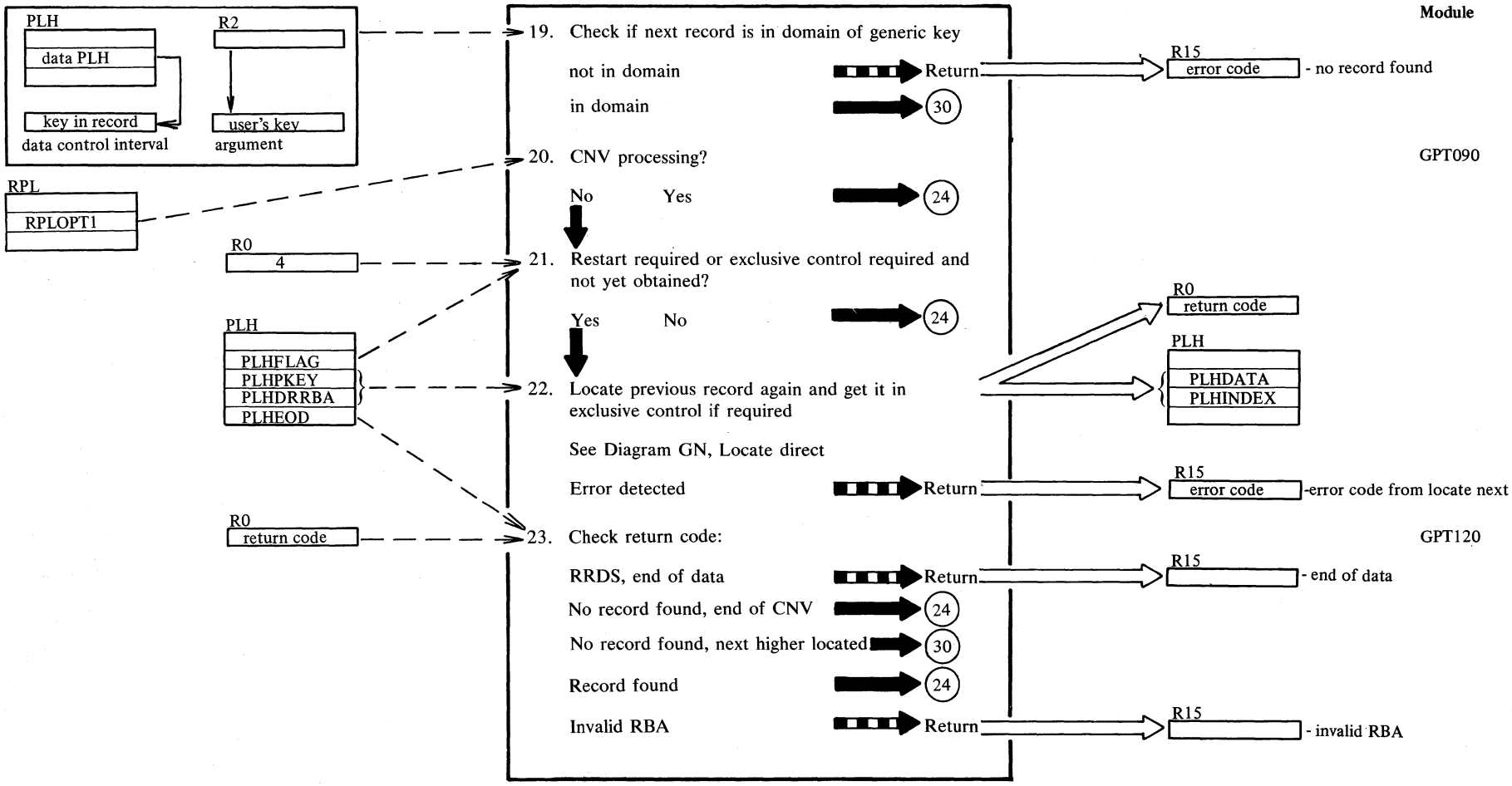


Diagram GF5. GET: Retrieve a record

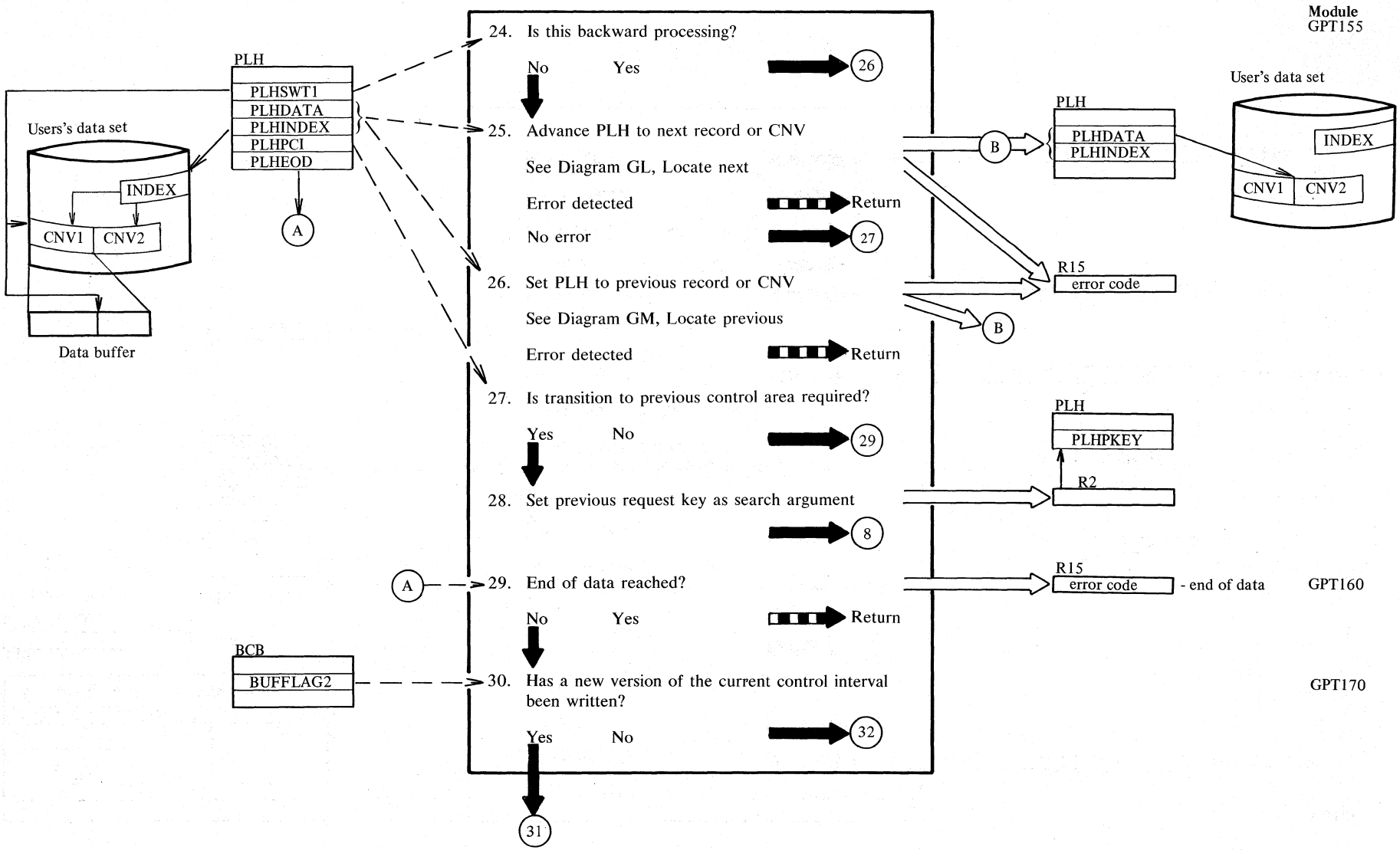


Diagram GF6. GET: Retrieve a record

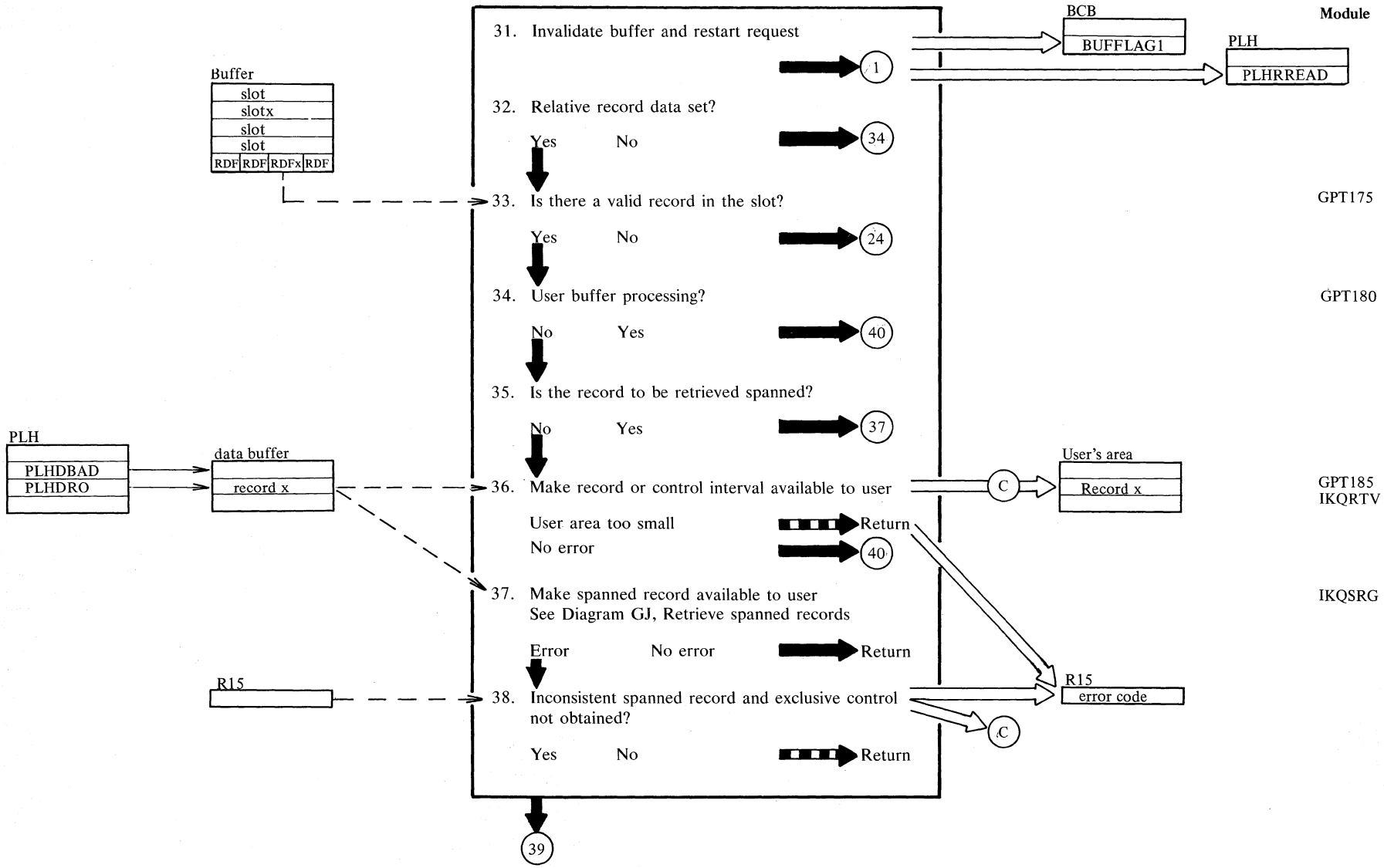
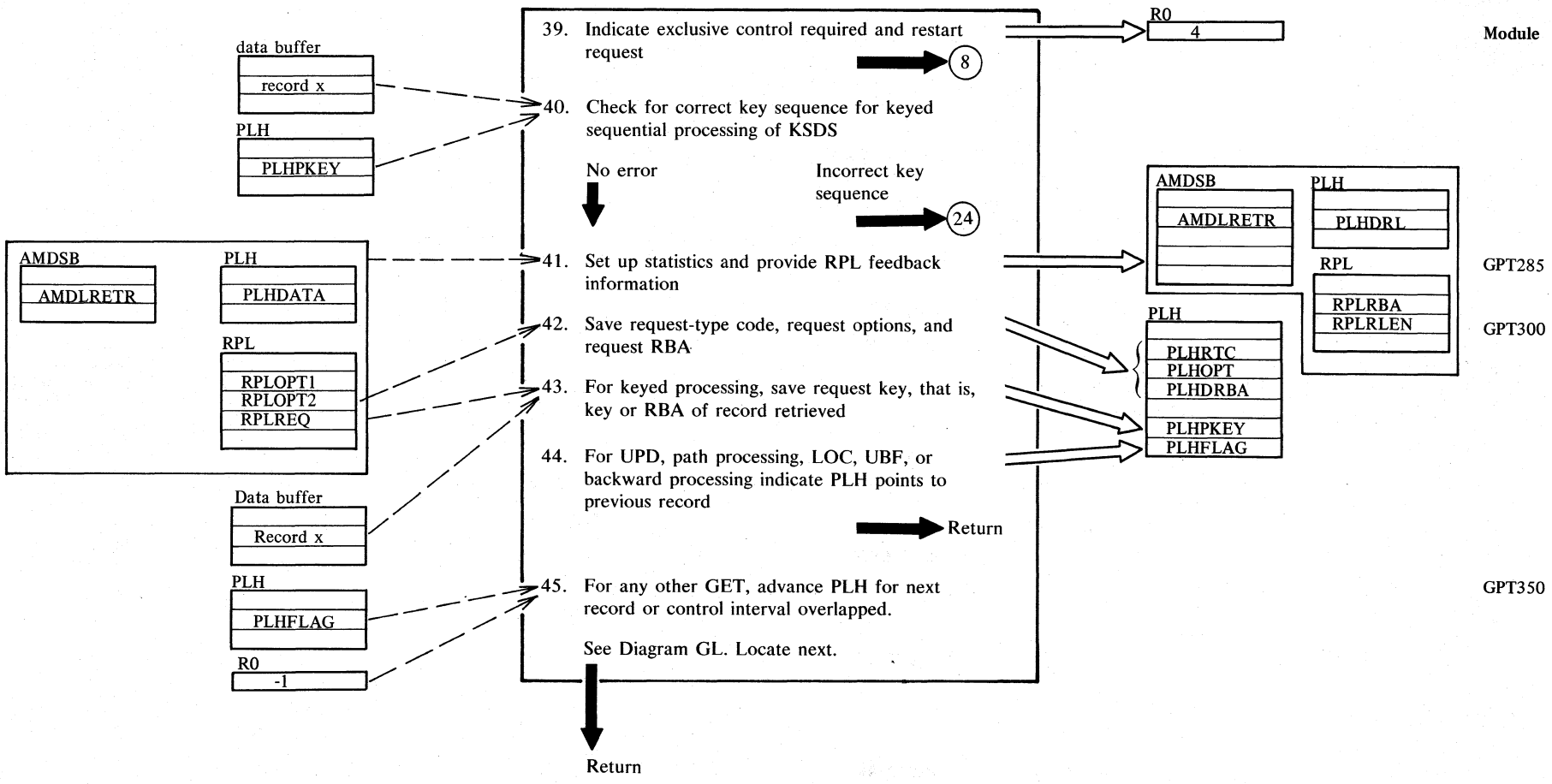


Diagram GF7. GET: Retrieve a record



Notes for Diagram GF

	Description	Module
2	A direct or skip sequential GET does not require prepositioning; it implicitly positions to the correct record or control interval before transferring it to the user's area.	IKQGPT
4	The key is supplied as argument. The length is the full key length (in AMDSB) for FKS and the KEYLEN supplied by the user for a generic key.	IKQGPT
6	For a positioned SKP request, the given key has to be in ascending key sequence with respect to the previous request.	IKQGPT
14	For a keyed request for KSDS other than a positioned SKP request, the search is top-down; for a positioned SKP request, the search is generally horizontal, starting with the sequence set control interval for the previous request. If the previous request had an I/O error or an end-of-data condition, however, the index search is top-down.	IKQLCD
27-28	During keyed backward processing of a KSDS, LOCATE PREVIOUS can step backwards only until it reaches the beginning of a sequence set record, which corresponds to the start of a data CA. LOCATE PREVIOUS then referens a qprevious data CA requiredq condition, and stores the lowest key of the current CA in the previous request key field. LOCATE DIRECT and INDEX SEARCH will then carry out a top-down search and transition to the previous sequence set record.	
36	If move mode is specified and not user buffer processing, the data record or control interval is transferred from the VSAM buffer to the user's area.	IKQRTV
37	Spanned record GET moves the first segment into the user's area, then reads the next control interval and moves the next segment into the user's area and so on until all segments have been moved.	
41	The RBA of the transferred data record or control interval is provided in the RPL as well as the length of the record or control interval retrieved. The number of records retrieved is the statistics updated in the AMDSB.	IKQGPT
45	VSAM will be positioned to the next control interval in an overlapped manner, that is, I/O operations (read ahead) are started but completion is not waited for, in order to overlap the I/O operation with the user's processing of the record or control interval just retrieved, so as to improve throughput.	IKQGPT

Diagram GG1. PUT ADD: Store a new record

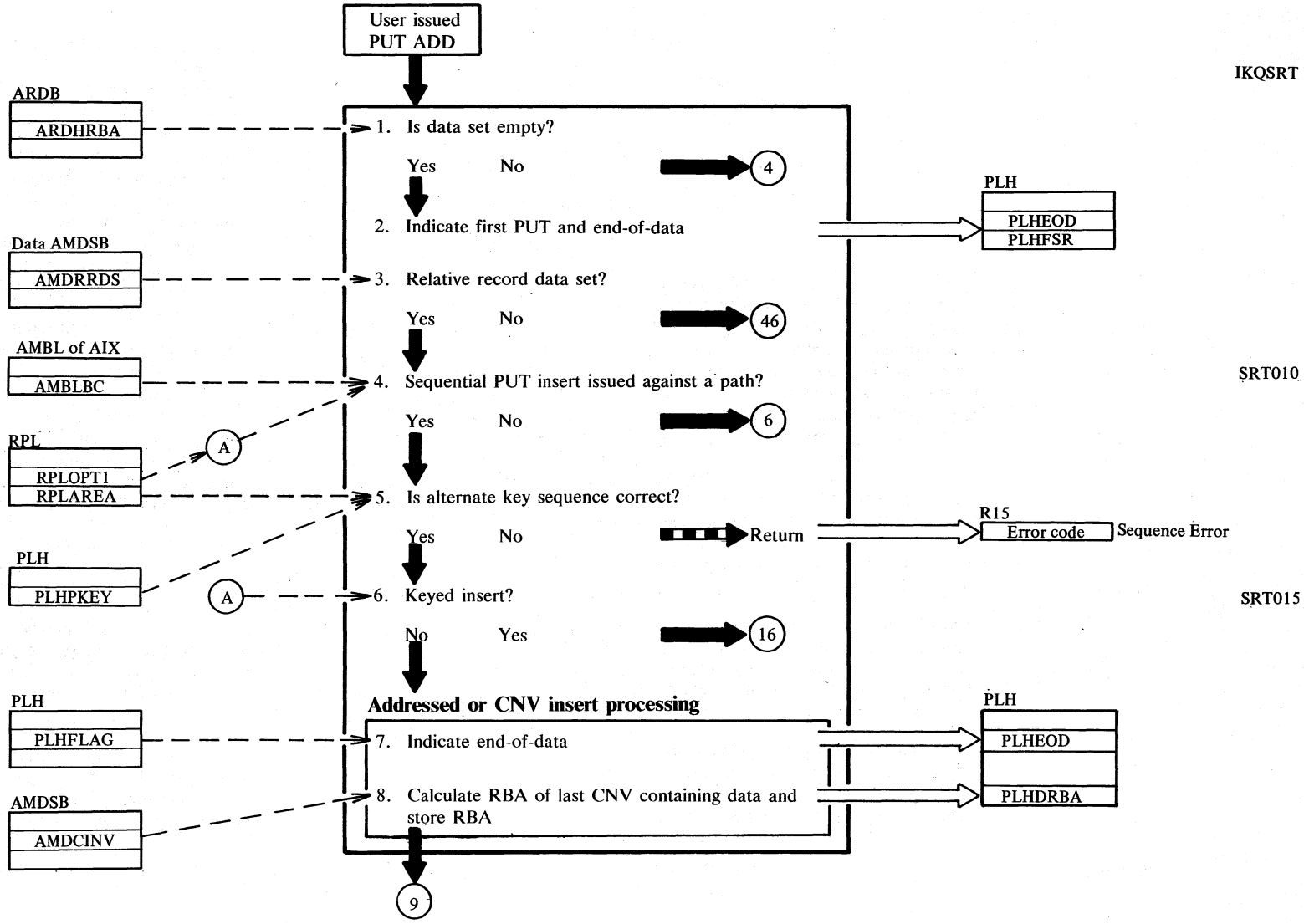


Diagram GG2. PUT ADD: Store a new record

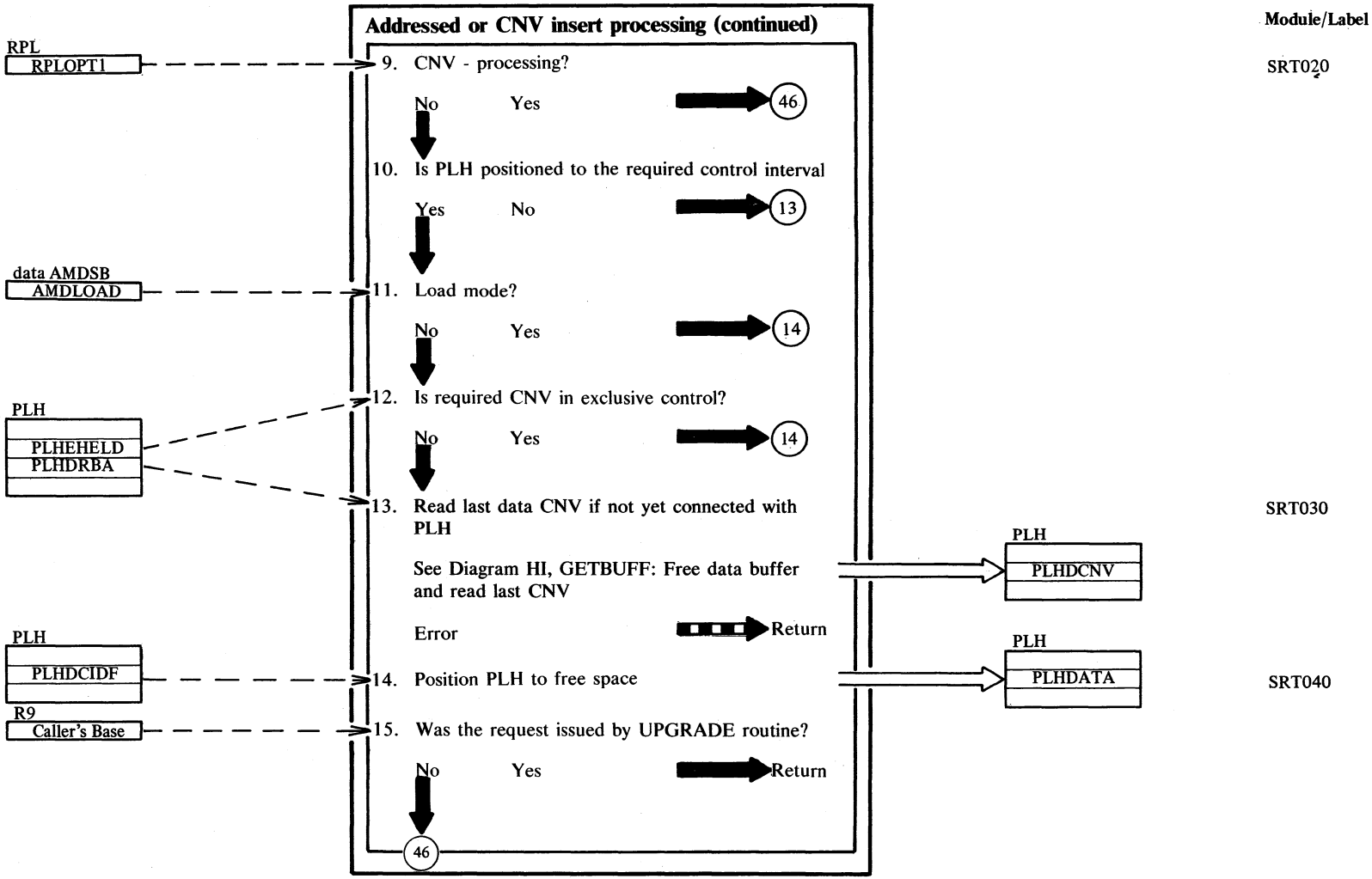


Diagram GG3. PUT ADD: Store a new record

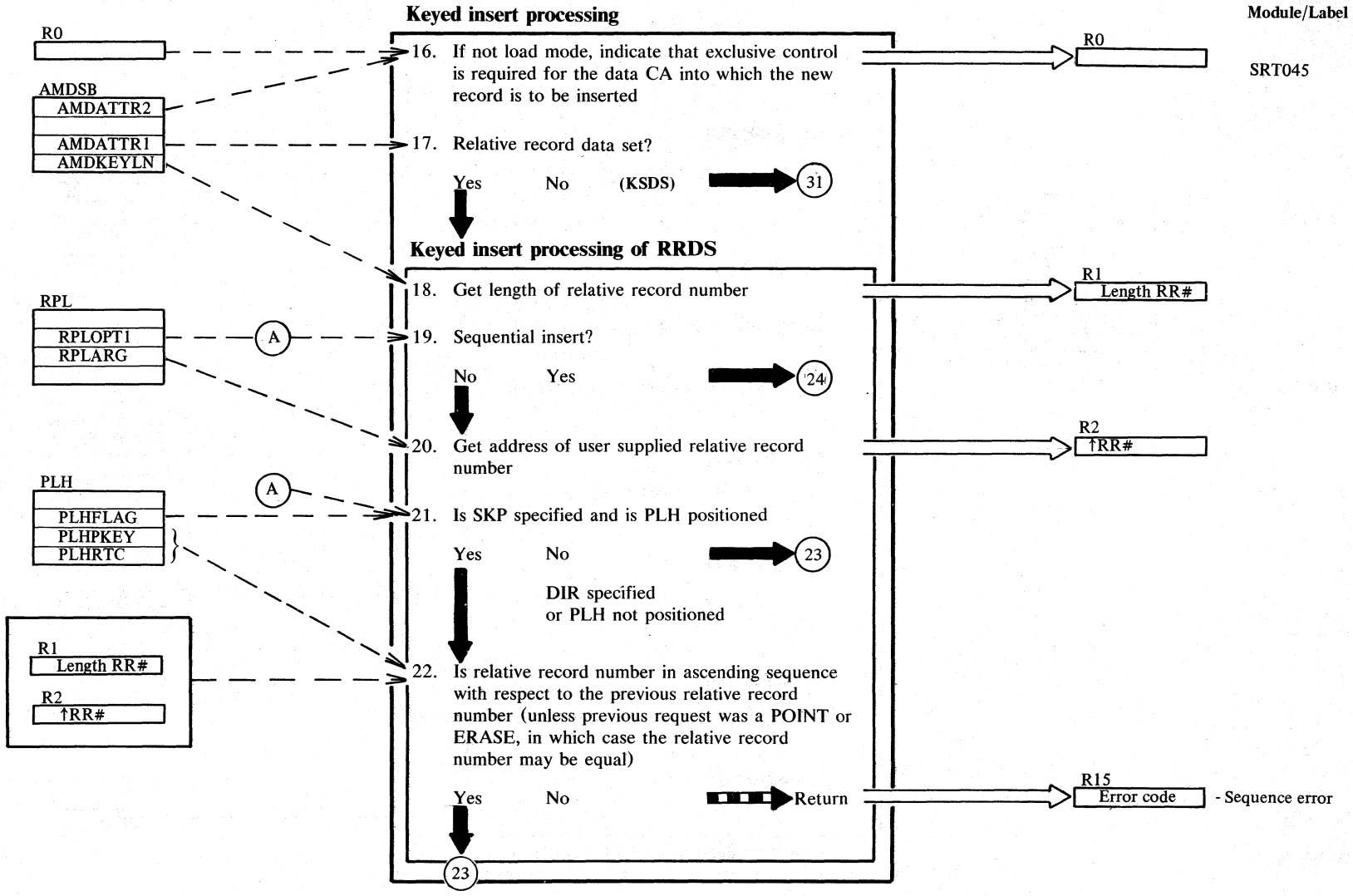


Diagram GG4. PUT ADD: Store a new record

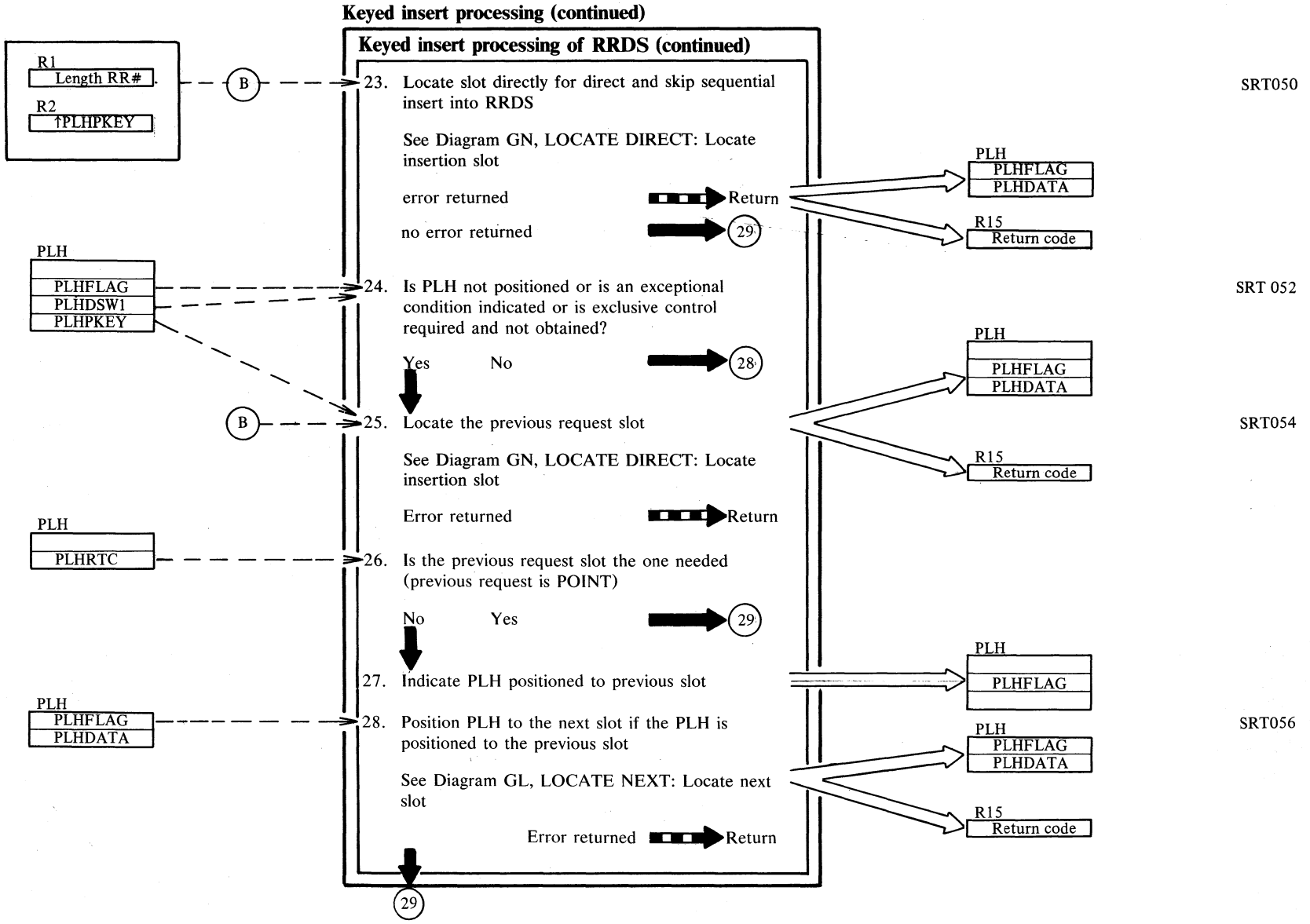


Diagram GG5. PUT ADD: Store a new record

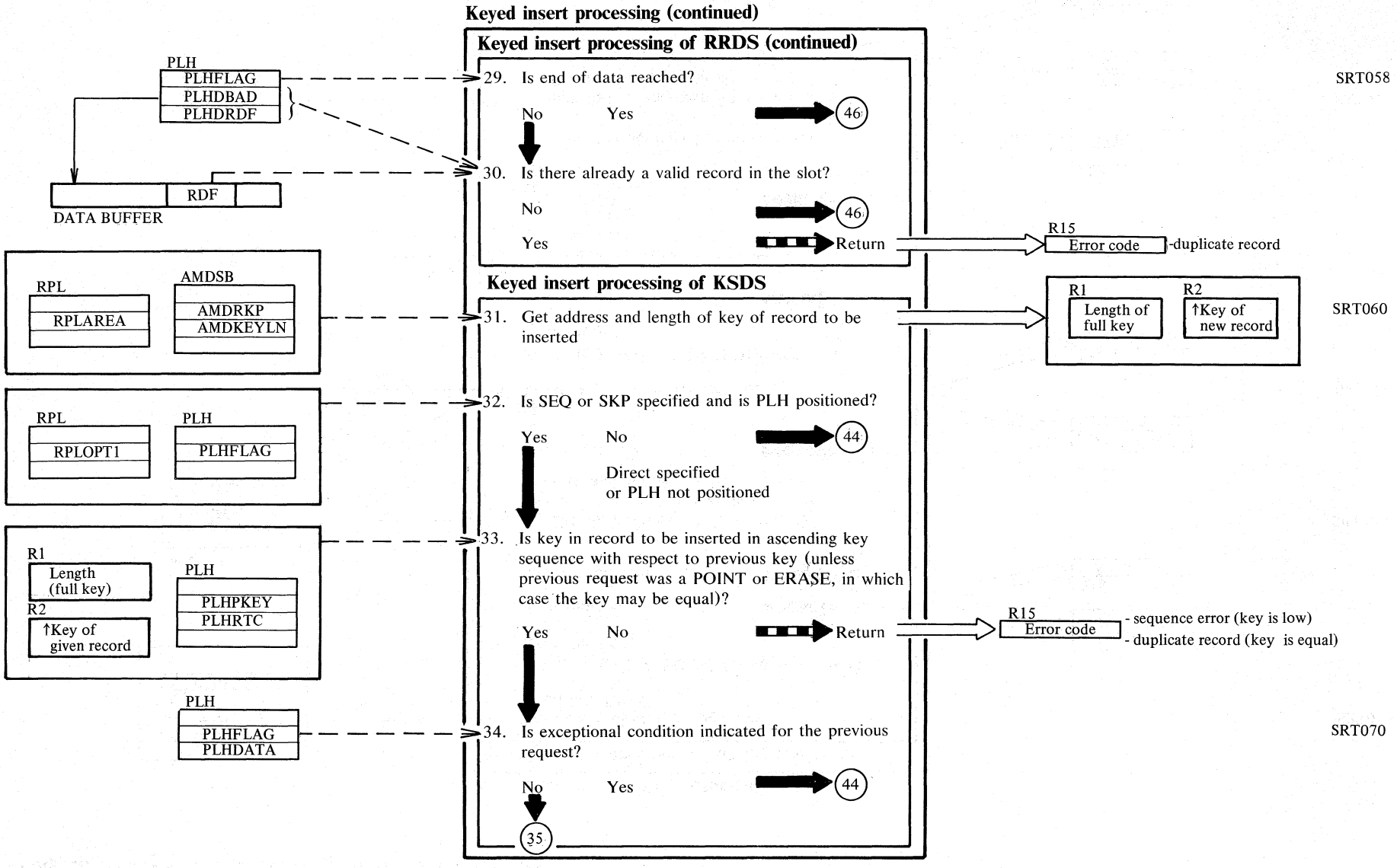


Diagram GG6. PUT ADD: Store a new record

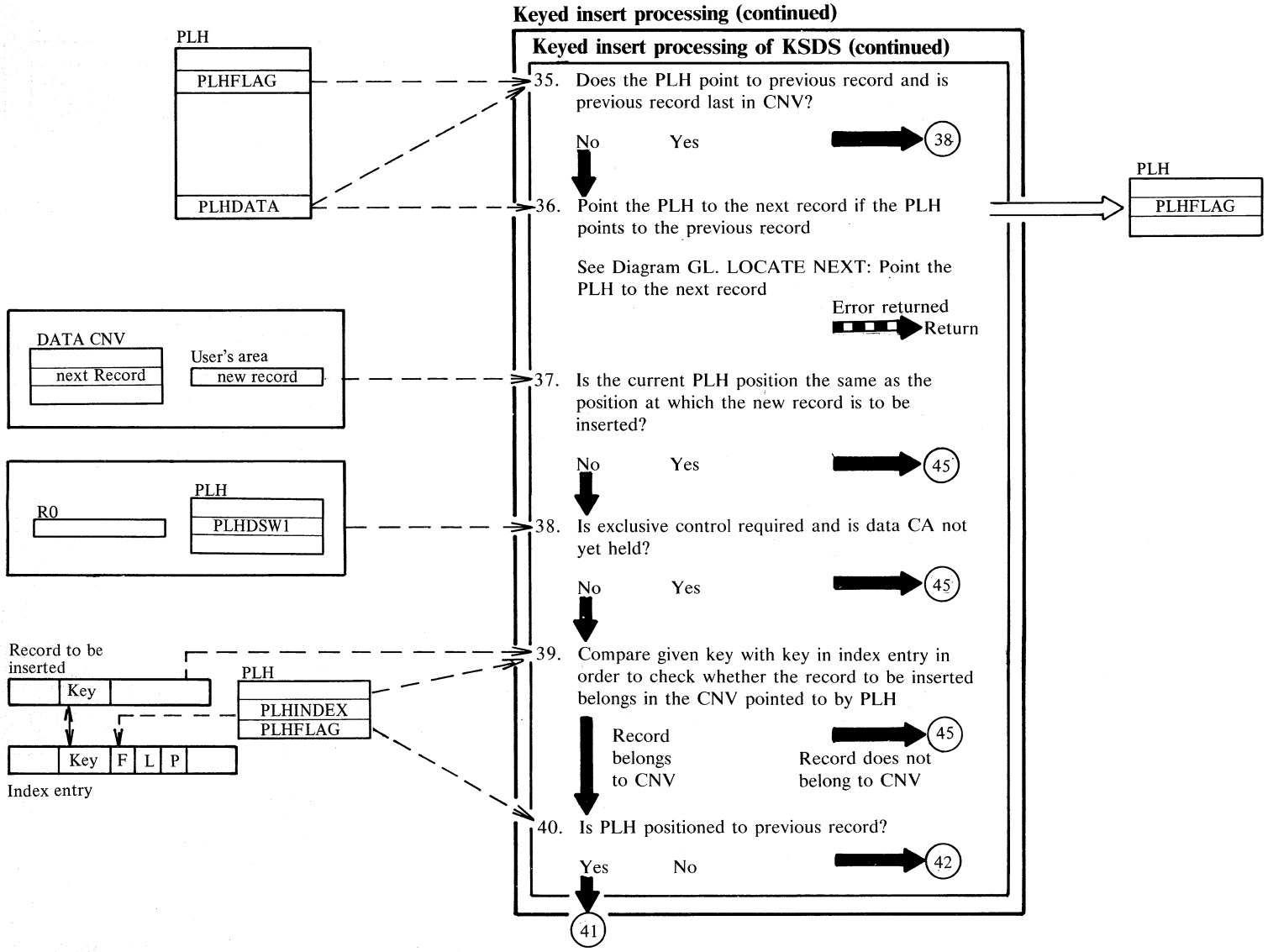


Diagram GG7. PUT ADD: Store a new record

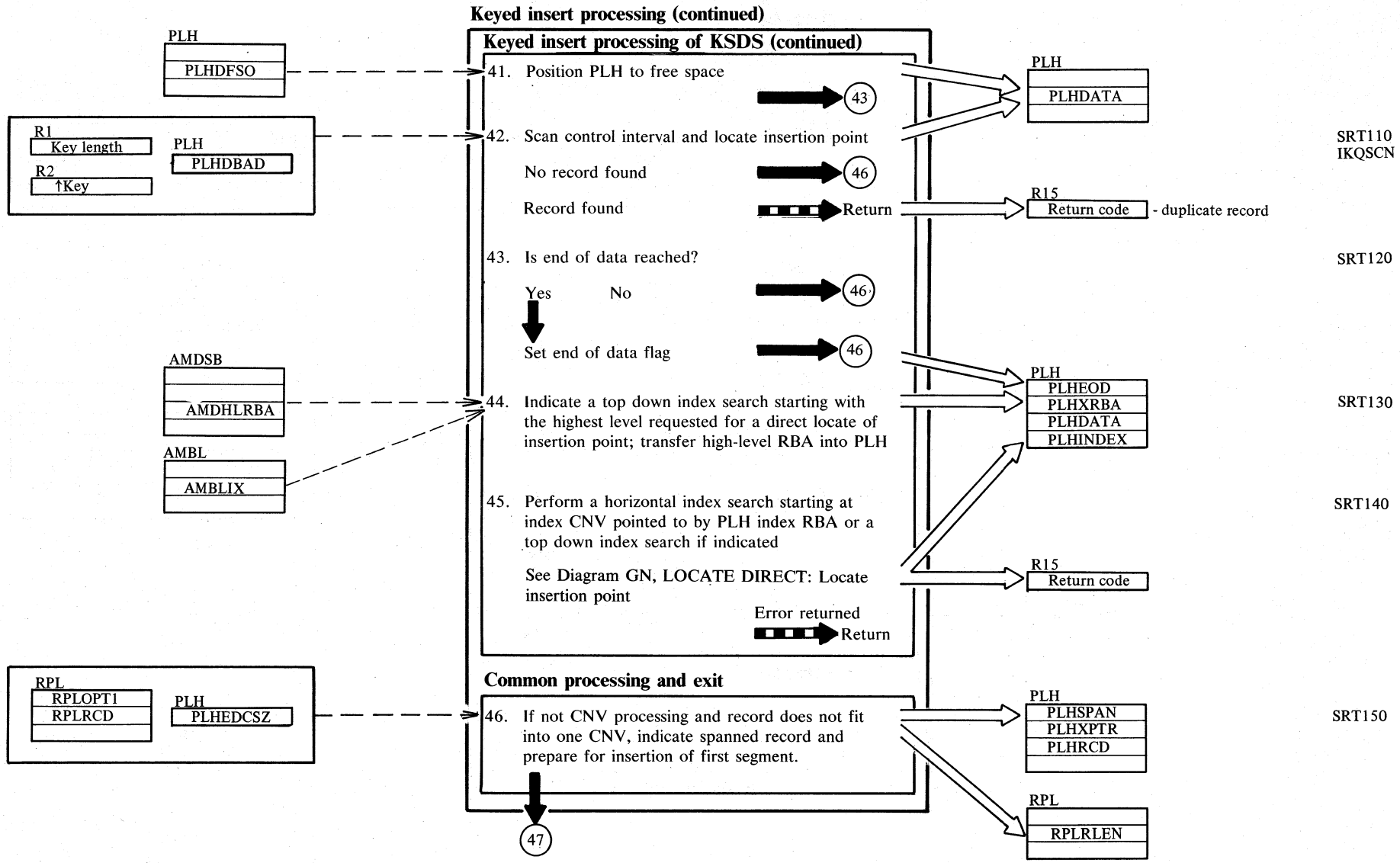


Diagram GG8. PUT ADD: Store a new record

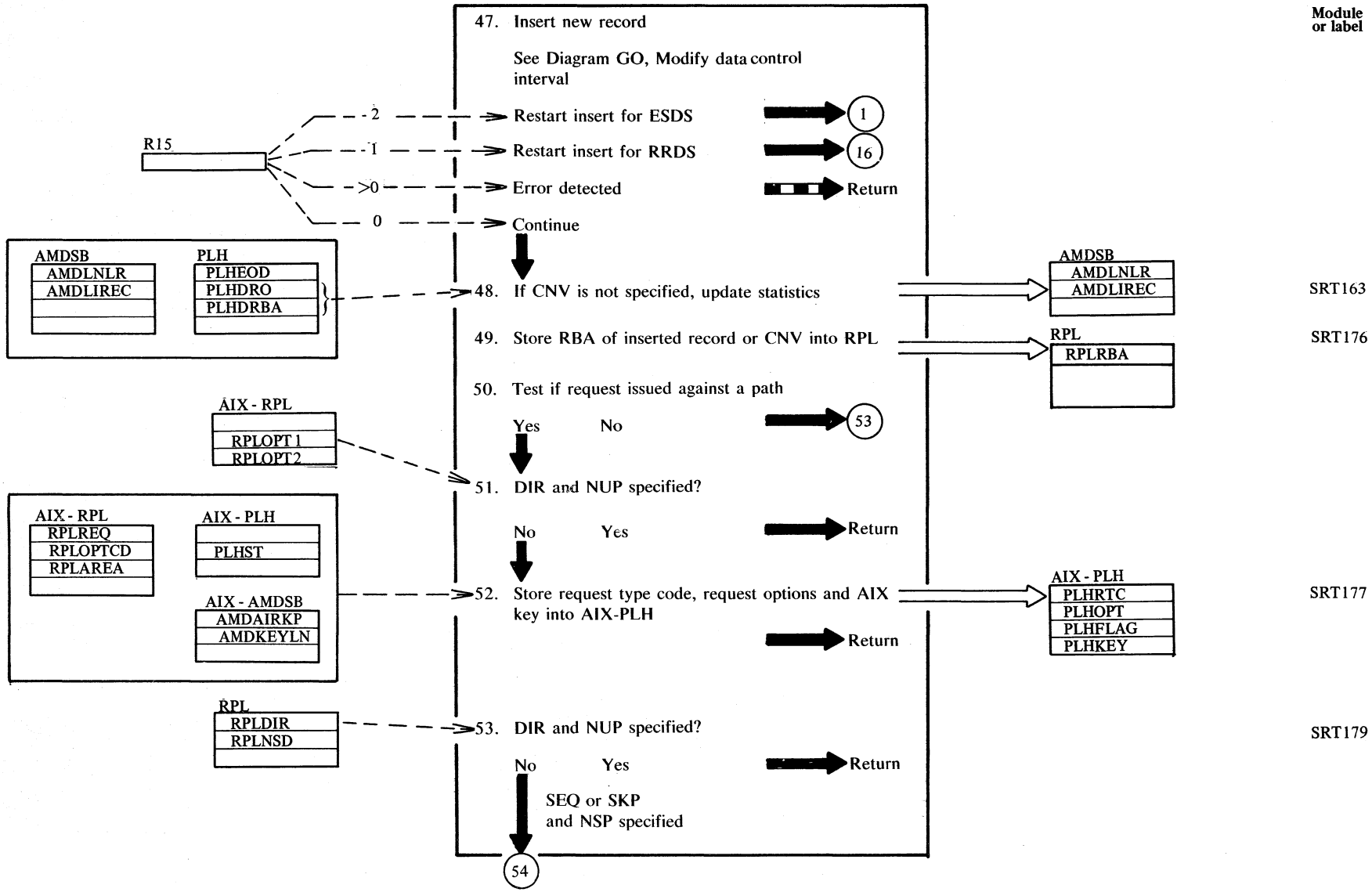
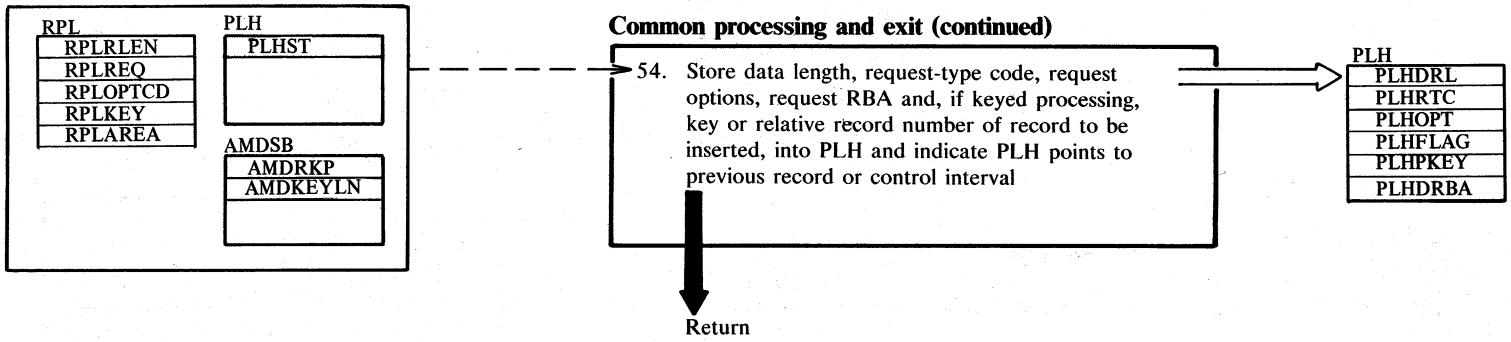


Diagram GG9. PUT ADD: Store a new record



SRT180

Notes:

47 **For relative record processing, the Modify Data CNV routine may indicate the PUT ADD processing has to be restarted.**

This occurs when larger portions of the data space had to be preformatted, and the connection between the PLH and the target data CI was lost.

Diagram GH1. PUT UPDATE or ISAM issued PUT in LOCATE mode: Store an updated record

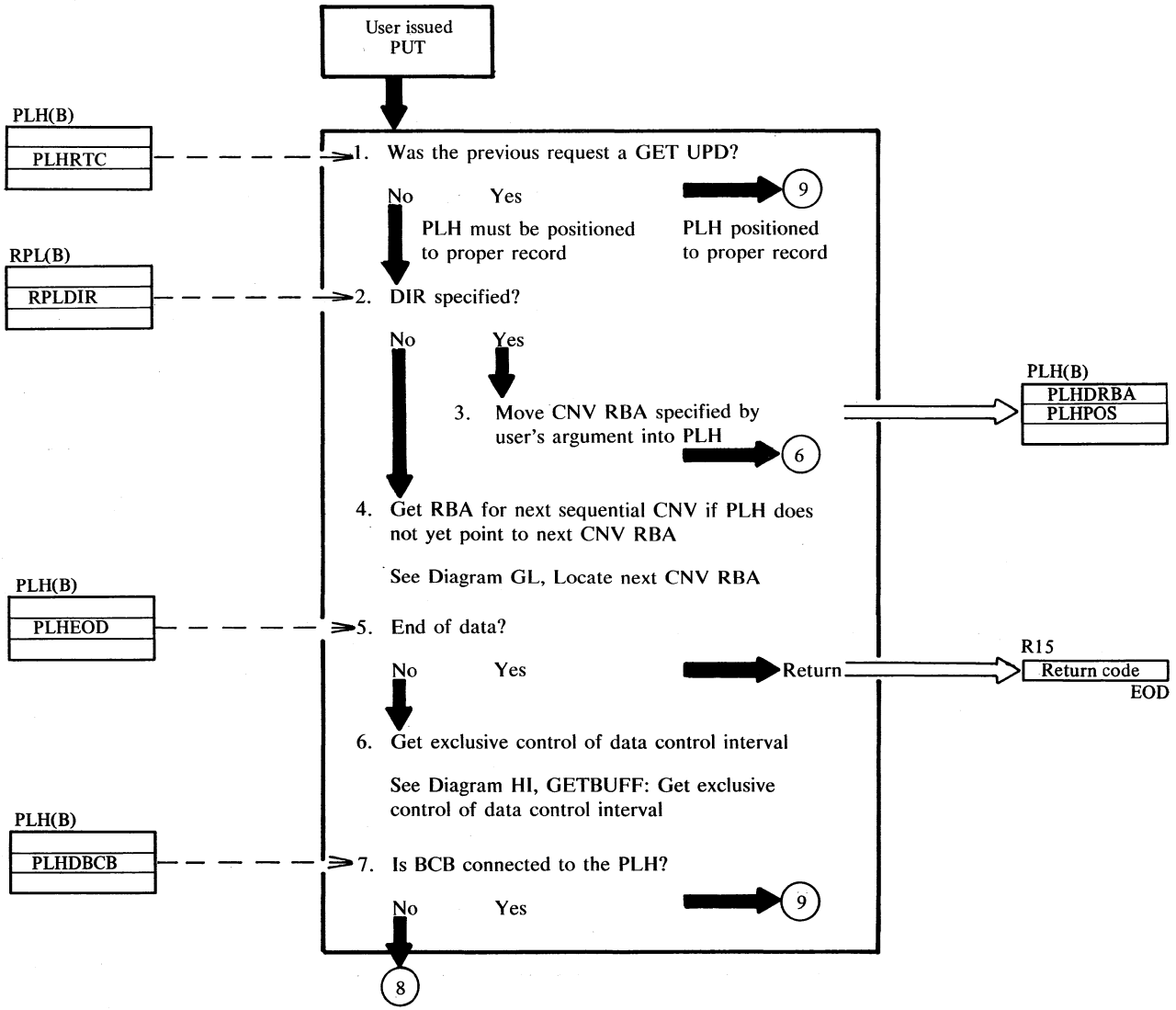


Diagram GH2. PUT UPDATE or ISAM issued PUT in LOCATE mode: Store an updated record

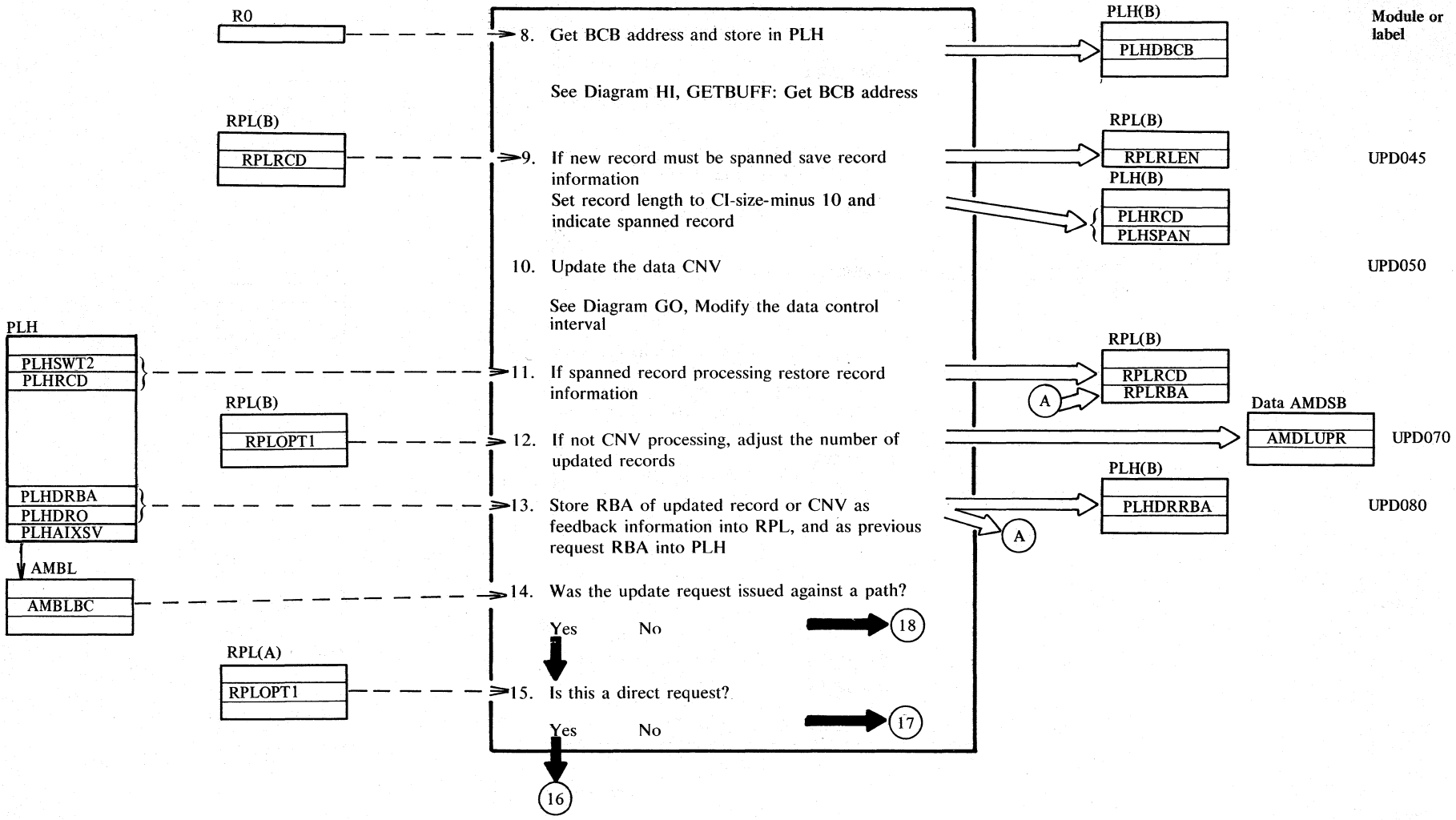
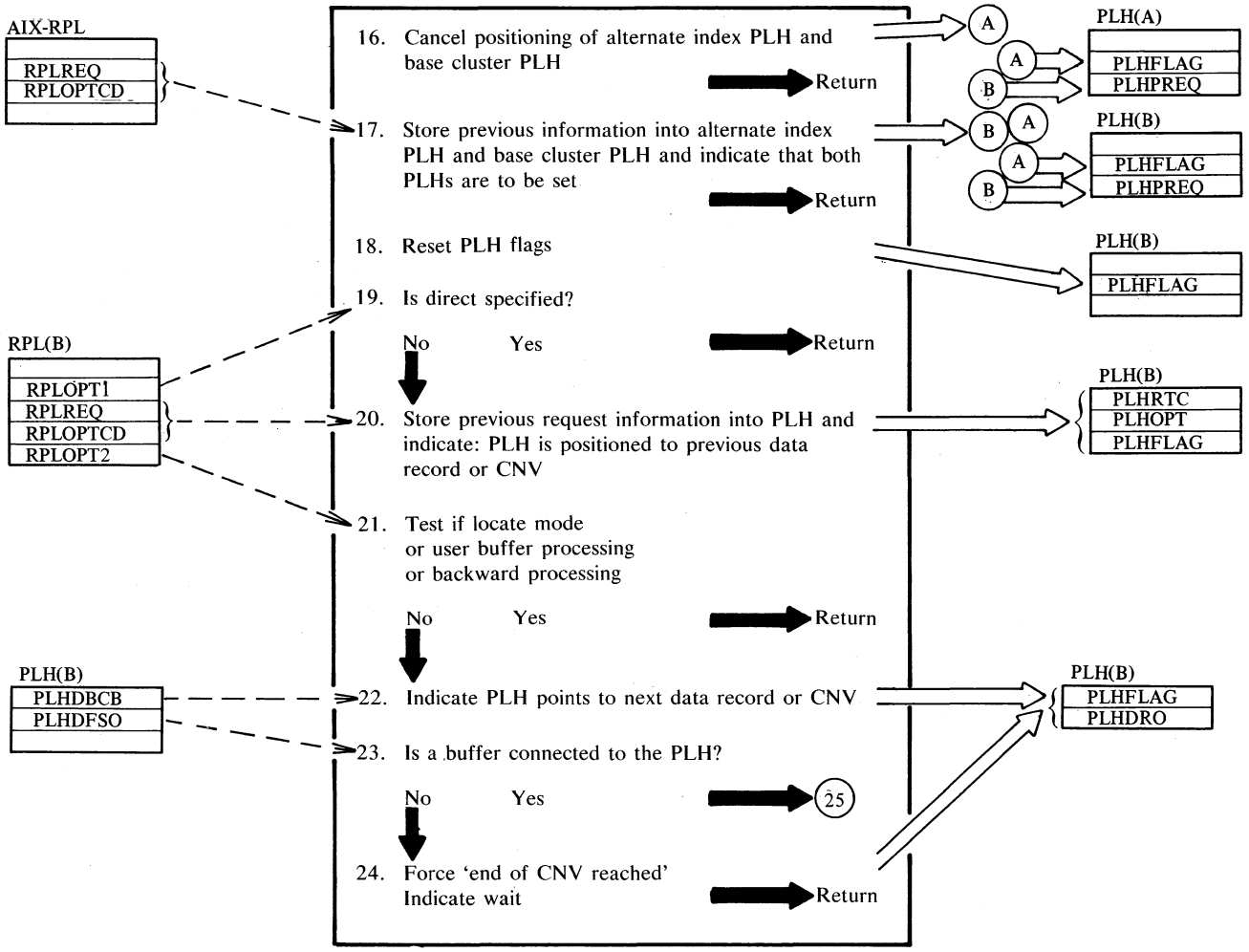


Diagram GH3. PUT UPDATE or ISAM issued PUT in LOCATE mode: Store an updated record



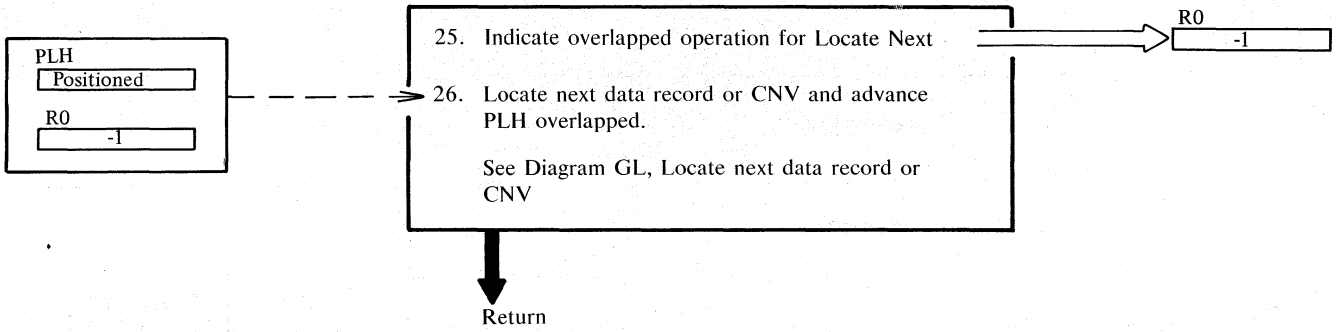
Module or label

UPD083

UPD085

Diagram GH4. PUT UPDATE or ISAM issued PUT in LOCATE mode: Store an updated record

Module or label
UPD090



Notes:

1 Stand-alone updates (updates without previous GETs) are allowed for user buffer processing.

26 PLH will be positioned to the next CI in an overlapped manner. This means that I/O operations (read-ahead) are started, but their completion is not waited for, in order to overlap the I/O operations with user processing.

Notation:

PLH(A), etc.: Control block associated with AIX
PLH(B), etc.: Control block associated with base cluster.

Diagram GI1. ERASE: Delete a record

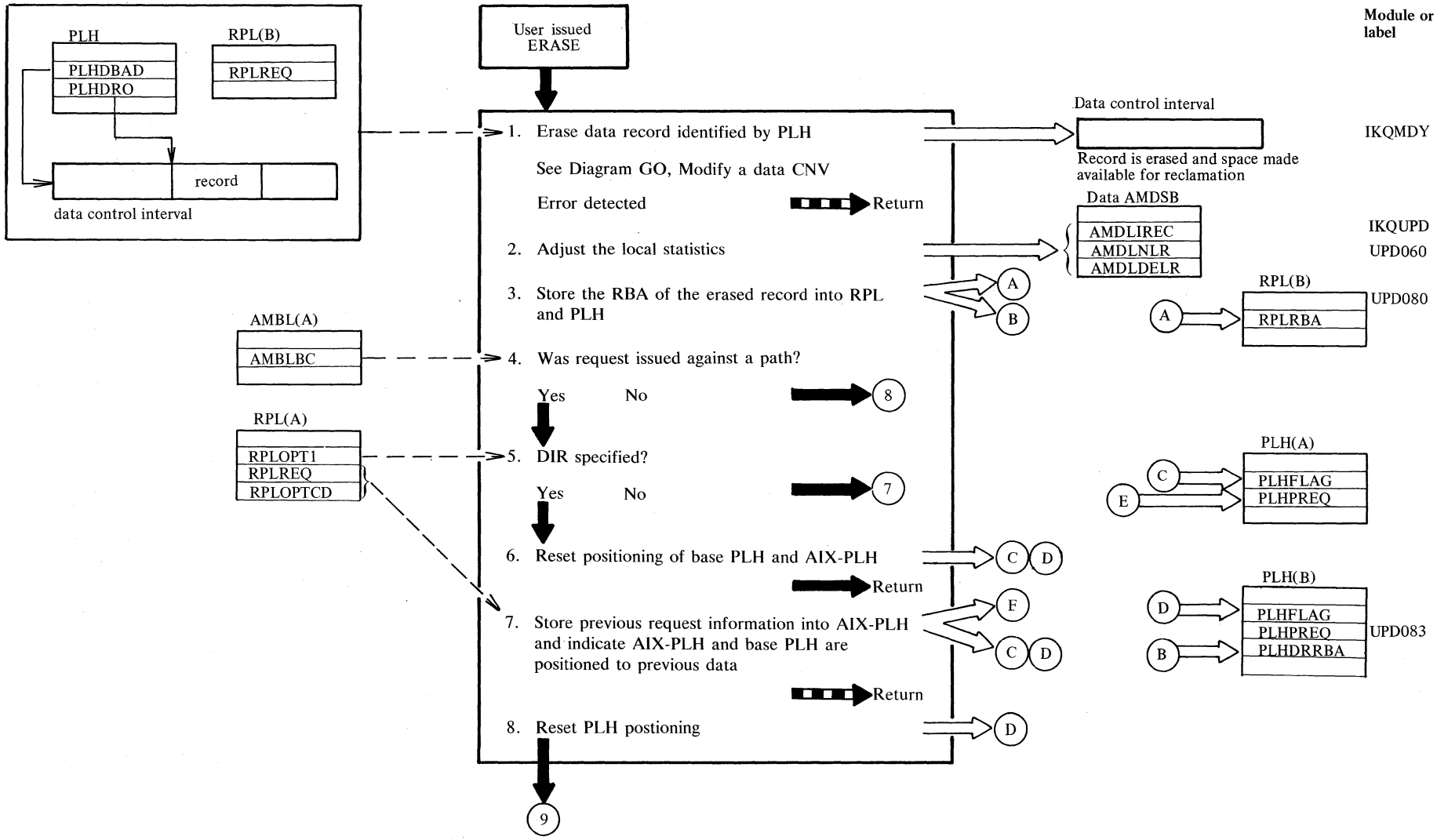
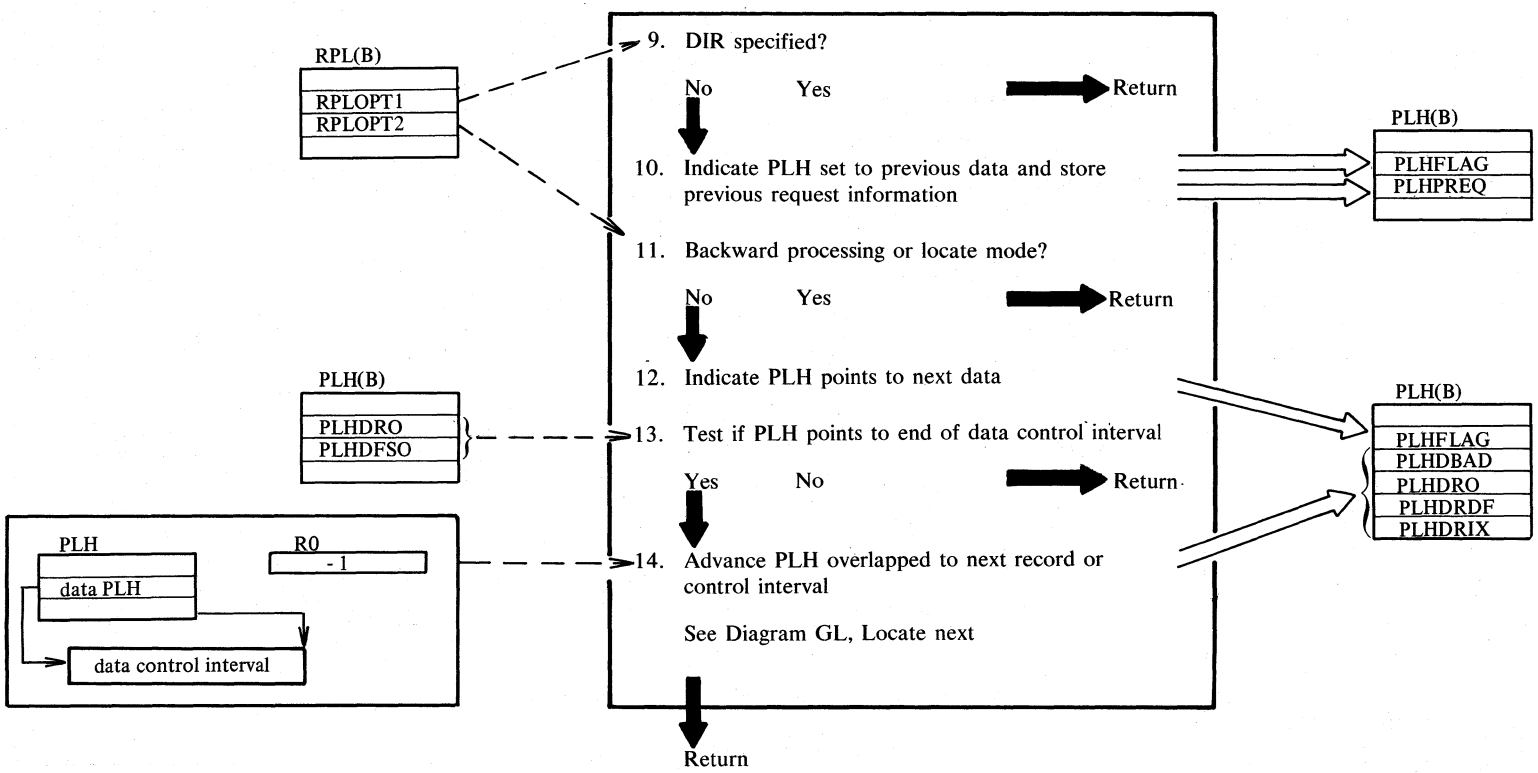


Diagram GI2. ERASE: Delete a record



Module or label
UPD085

UPD090

Notes:

Erase requests are allowed only for keyed or addressed processing of key-sequenced data sets or for relative record data sets. An ERASE must be preceeded by a GET for update, which positions the PLH to the record to be erased.

Notation:

PLH(B), etc.: Control block associated with base cluster

PLH(A), etc.: Control block associated with AIX

Diagram GJ1. Retrieve spanned records

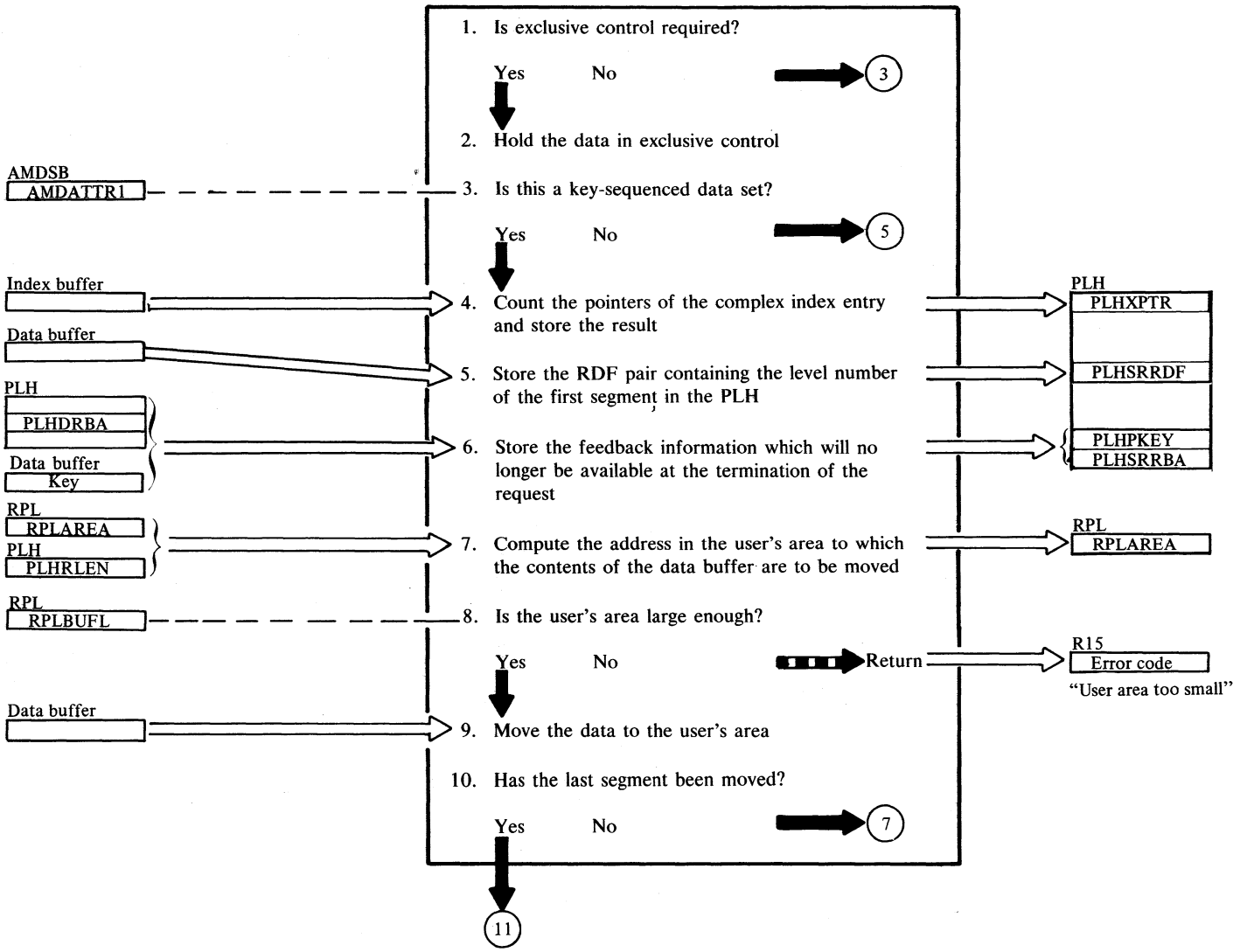


Diagram GJ2. Retrieve spanned records

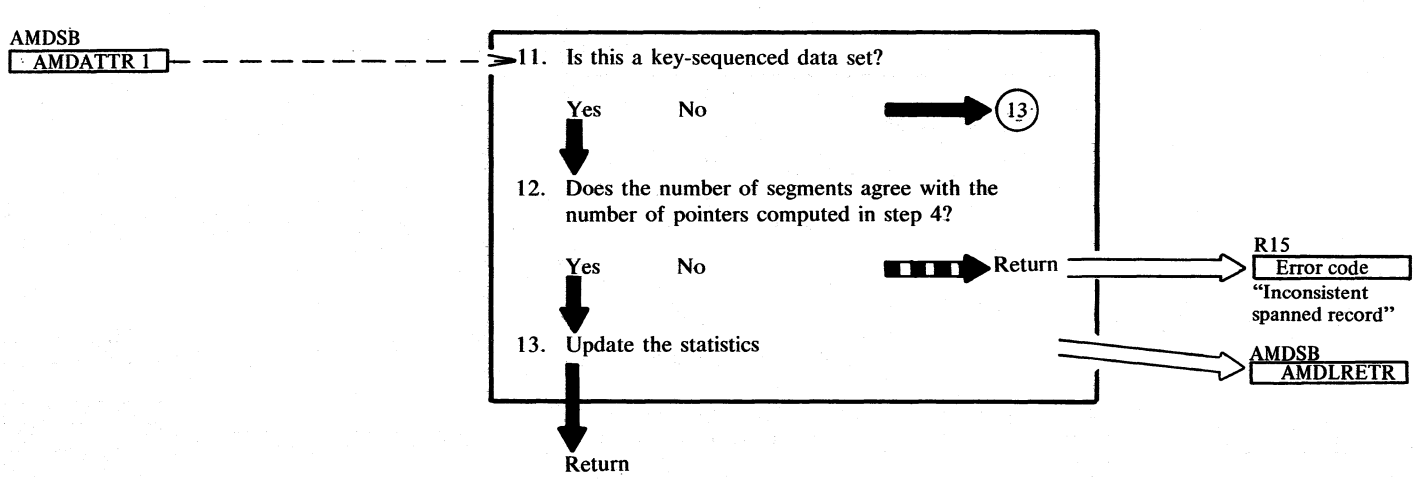


Diagram GK1. Store spanned records

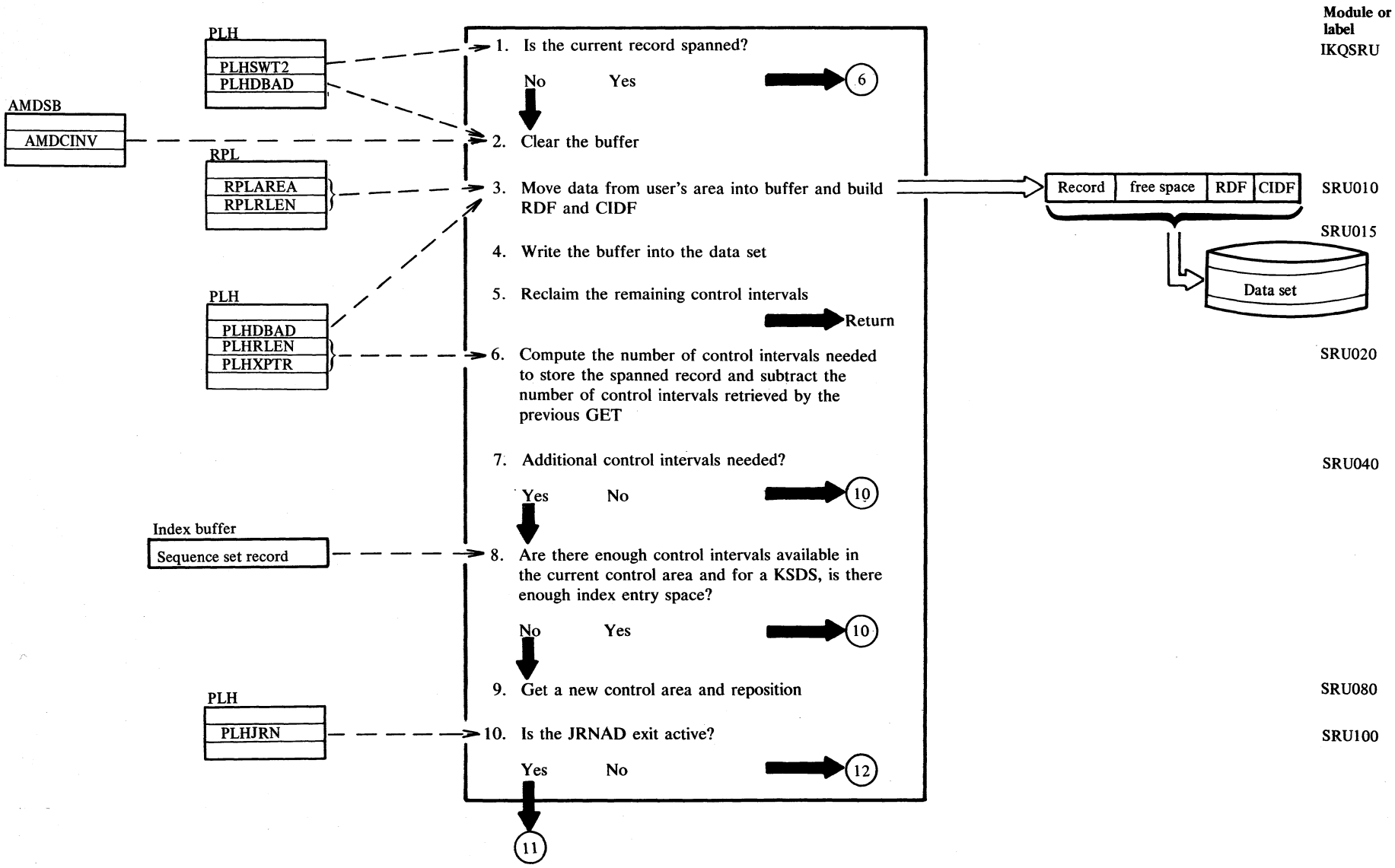


Diagram GK2. Store spanned records

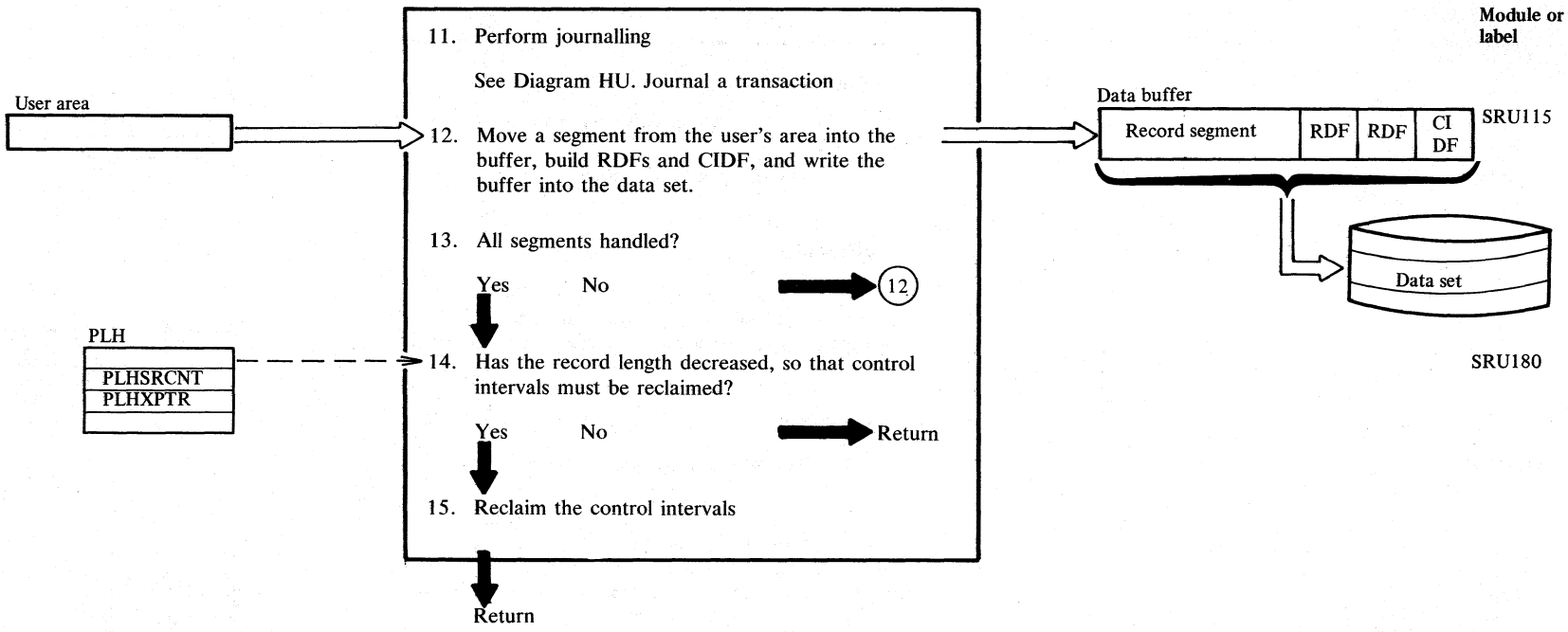
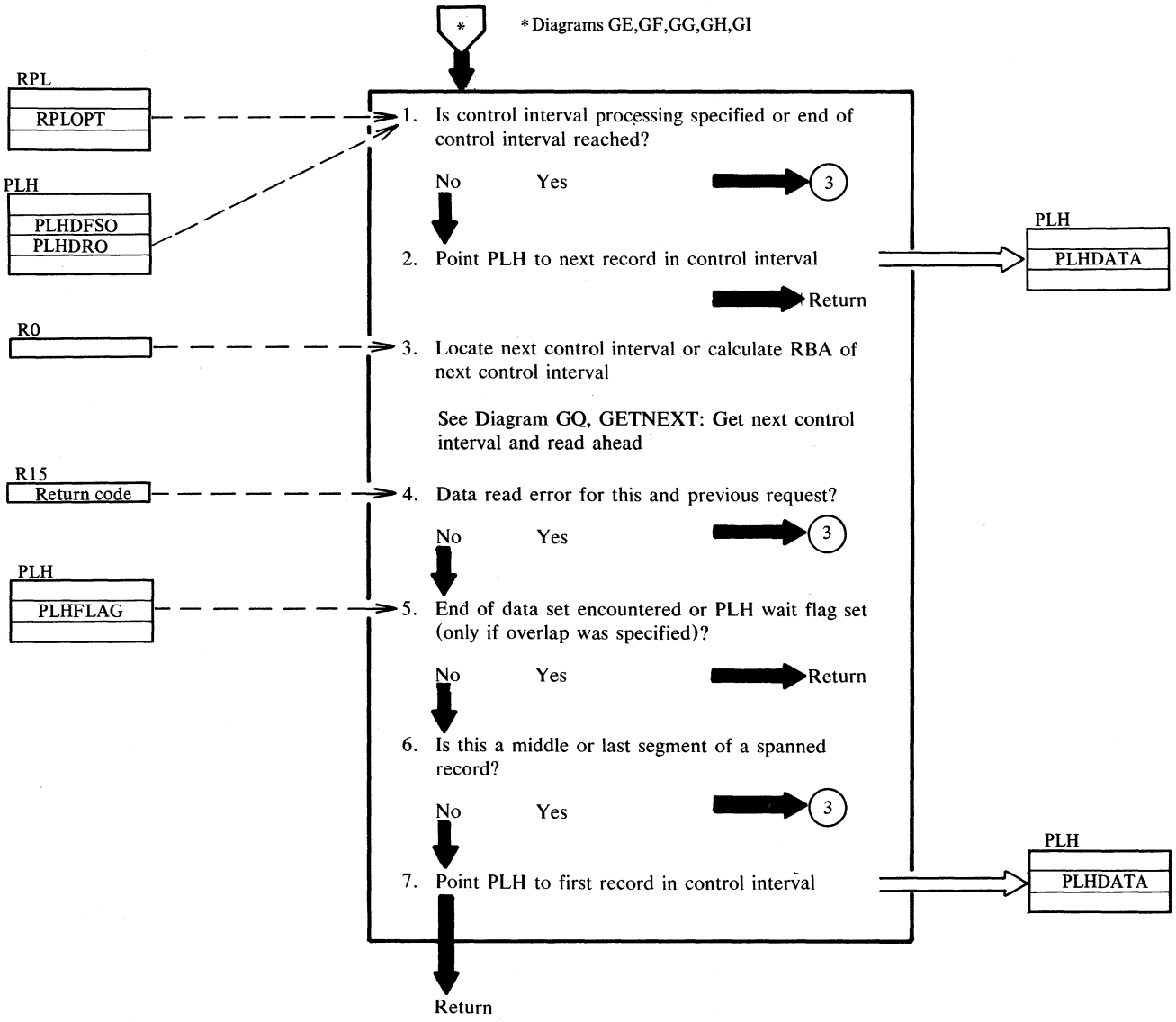


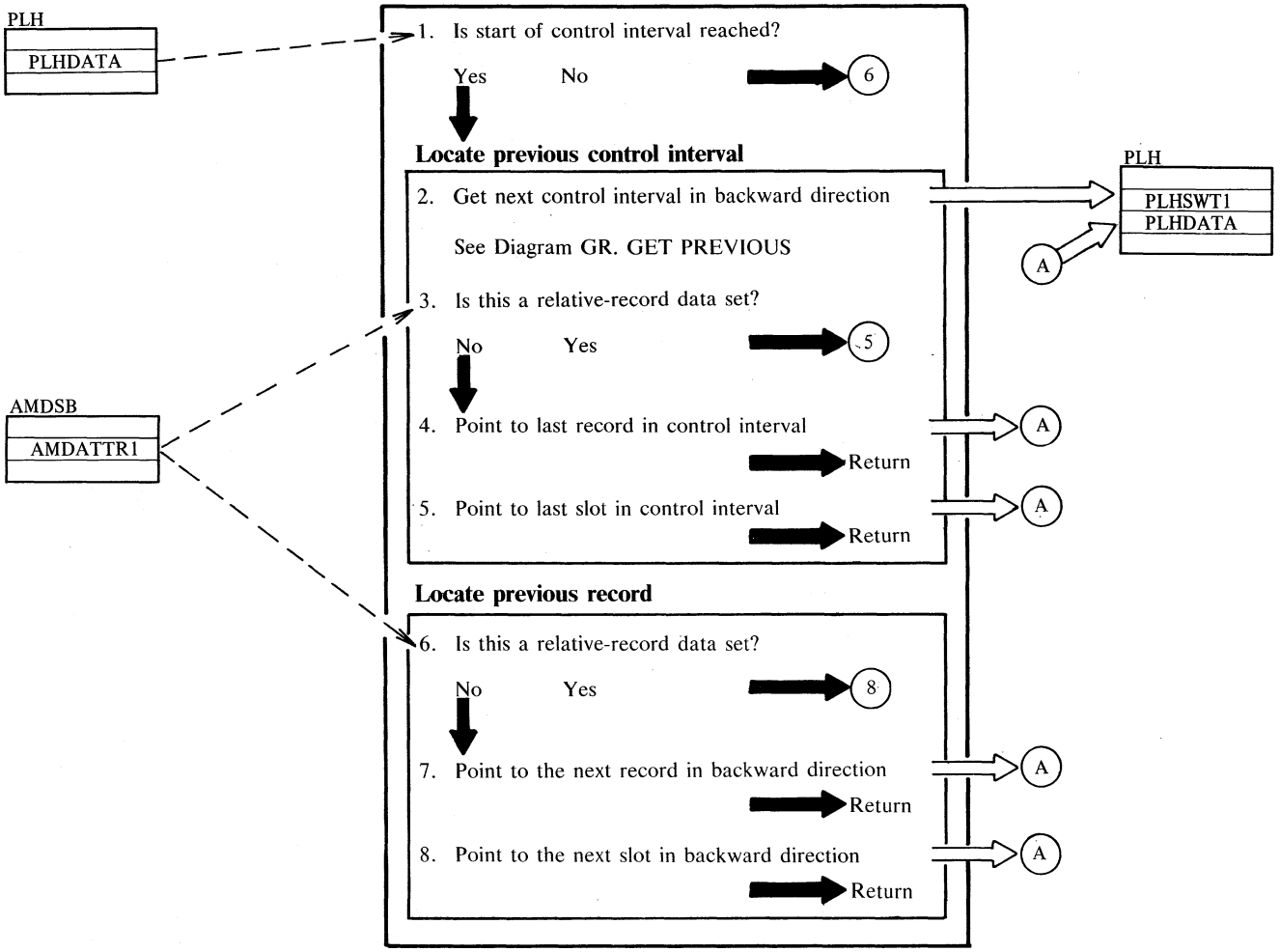
Diagram GL1. LOCATE NEXT: Locate next data record or control interval



Notes for Diagram GL

	Description	Label
1	A transition to a new control interval must take place if control interval processing is specified (CNV option in RPL specified), or if the end of a control interval is reached, or forced by a previous read error.	IKQLCN00
2	If a new control interval is not needed, the PLH is advanced to the next record within the current control interval and control is returned to the caller.	LCN010
3	GETNXT performs two types of operation: If register 0 = 0 or 4, an I/O operation is started and finished before further processing is done. If register 0 = 8 (CNV processing only), the RBA is advanced to the physically or logically next control interval but no reading is done. If register 0 holds a negative value, an overlap operation is indicated, causing read ahead to be started while processing continues. A second Locate Next operation with WAIT specified (register 0 = 0 or 4) will ensure that the data is read. This second Locate Next will connect the desired data buffer with the PLH. GETNXT overlap frees the data buffer and initiates read ahead (the index buffer is, however, retained).	LCN050
6	As the start of the next record is to be located, the middle and last segments of spanned records are skipped.	

Diagram GM1. LOCATE PREVIOUS: Locate previous data record or control interval



Module or label
IKQLCP LCP001
LCP020
LCP100
LCP150
LCP040
LCP160

Diagram GN1. LOCATE DIRECT: Locate data record or control interval by key or RBA

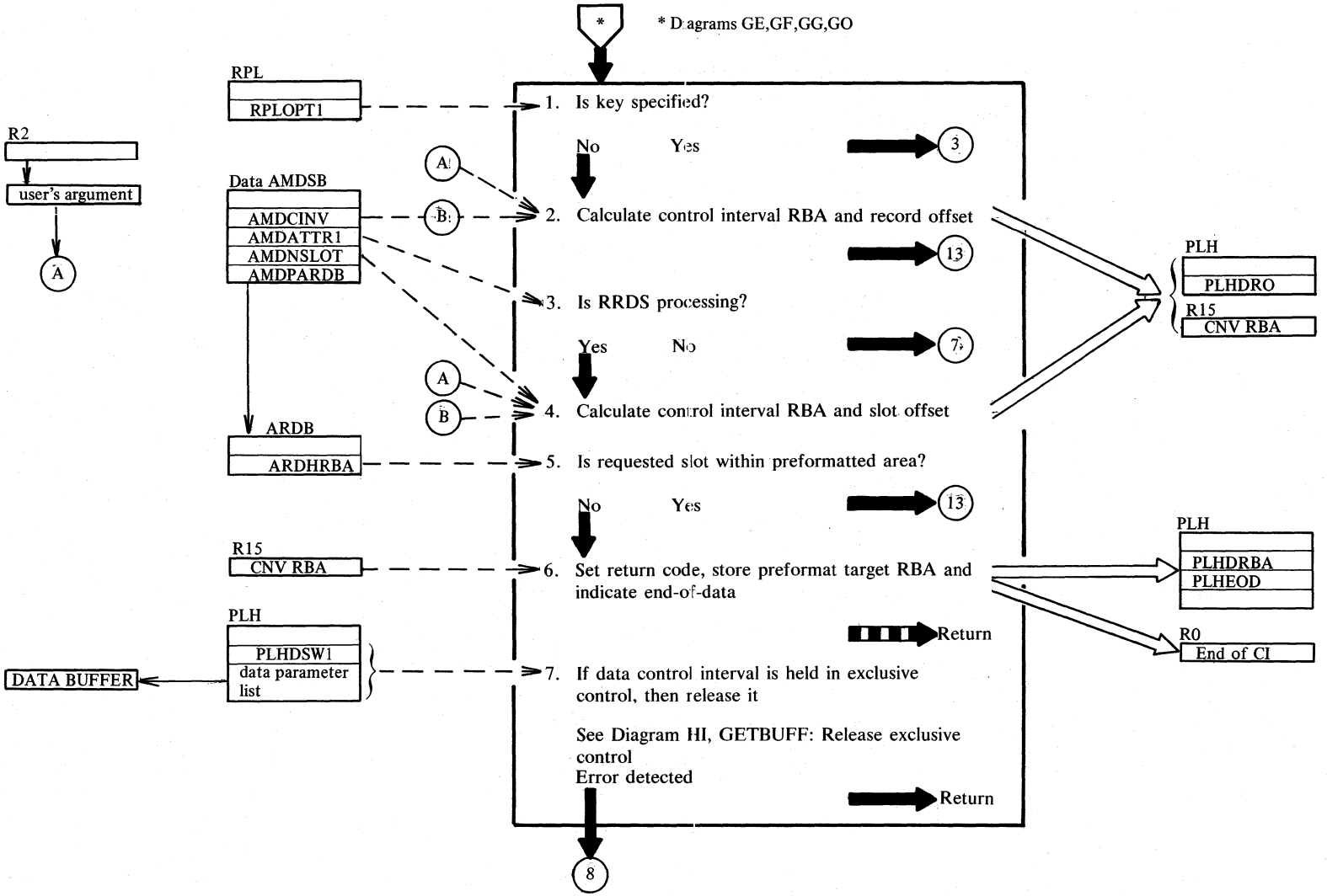


Diagram GN2. LOCATE DIRECT: Locate data record or control interval by key or RBA

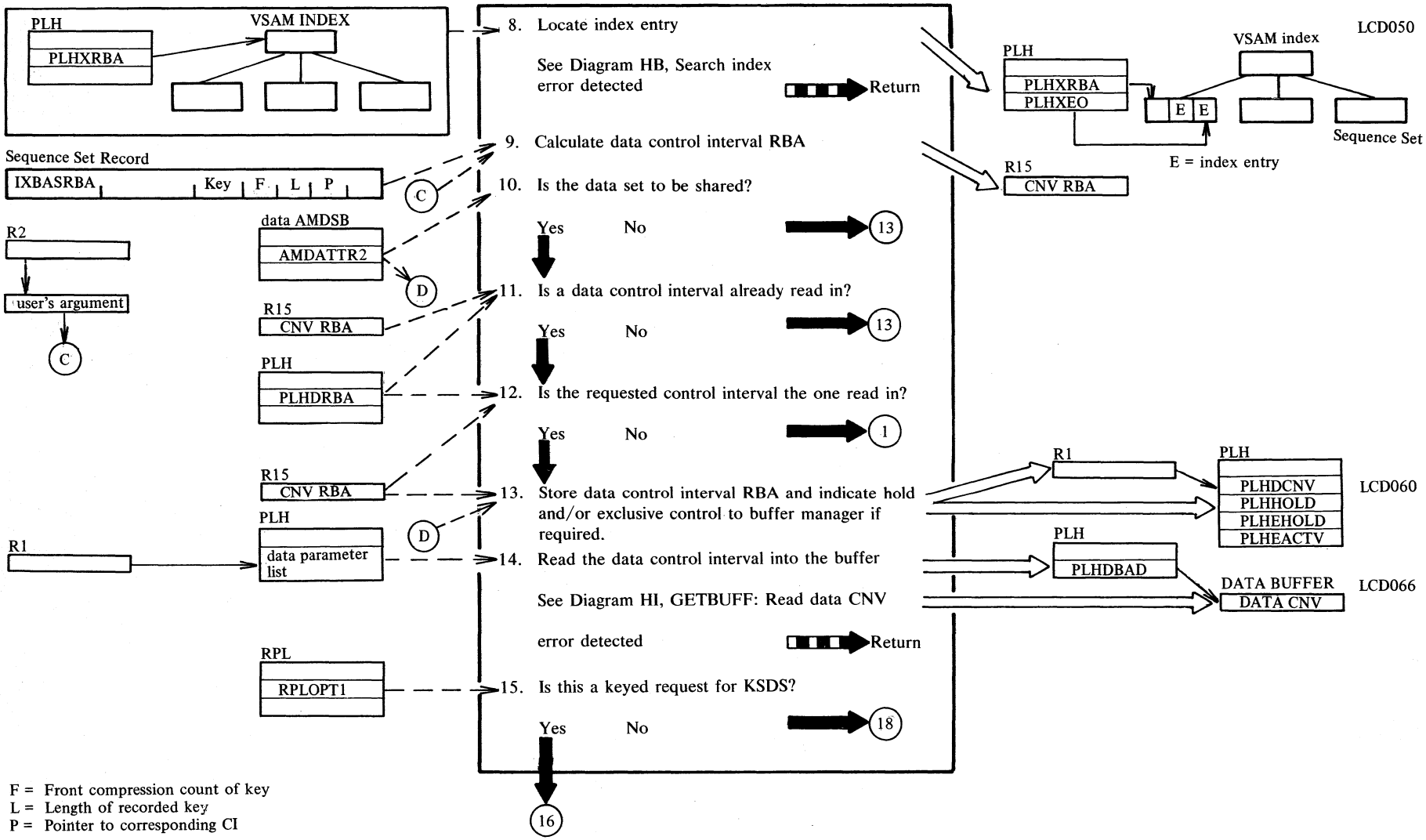
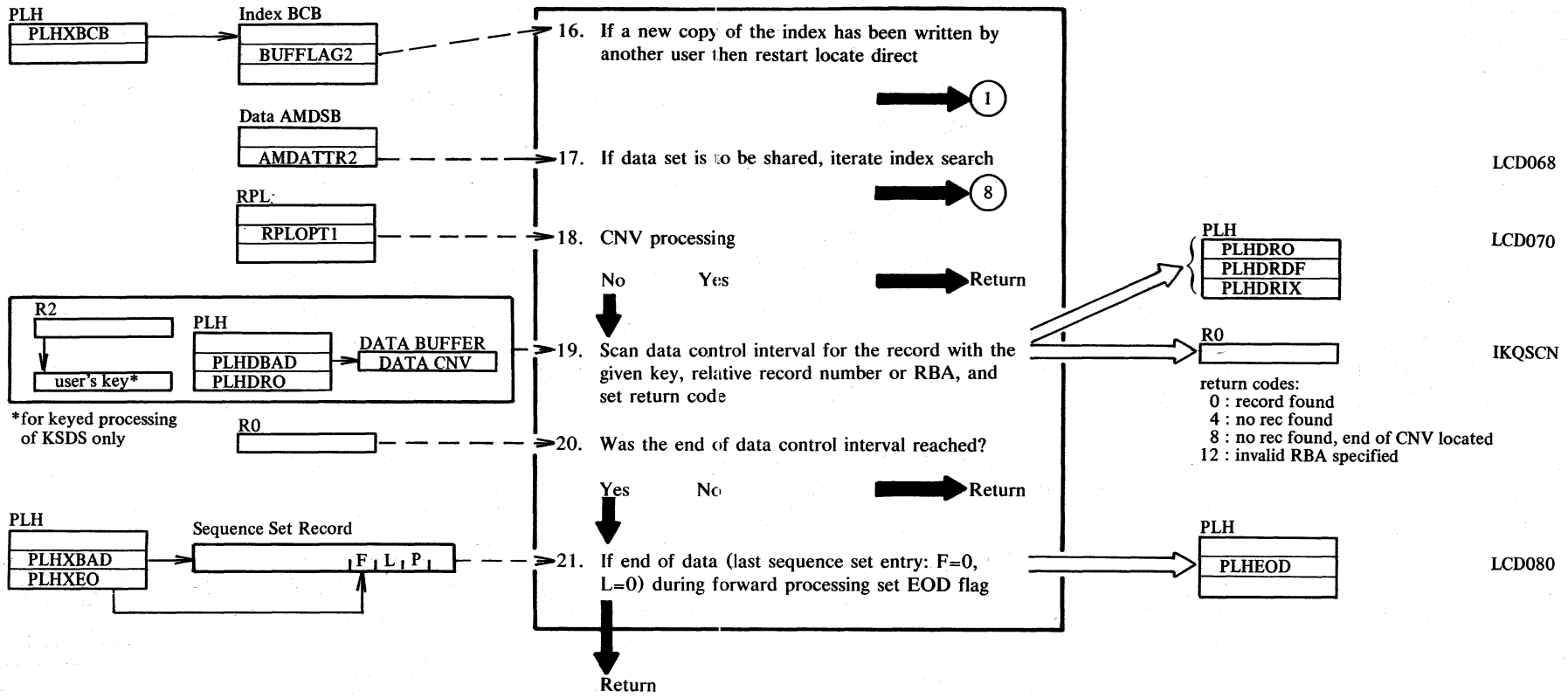


Diagram GN3. LOCATE DIRECT: Locate data record or control interval by key or RBA



F = Front compression count of key
L = Length of recorded key
P = Pointer to corresponding CI

Notes for Diagram GN

Description

10 If share option 4 was specified, and this is not the initial load of the data set, the data set is to be shared.

13 Hold:
If share option 4 is specified, buffer manager will issue a trackhold.

Exclusive control:
For multiple string processing, the buffer manager enters the RBA of the CI in an exclusive control list.

17 See note 10.

19 possible return codes for addressed processing
- record found
- invalid RBA

possible return codes for RRDS processing
- record found
- no record found
- no record found, end of CNV located (if requested slot is beyond preformatted area)

possible return codes for keyed processing of KSDS
- record found
- no record found
- no record found, end of CNV located

Diagram GO1. Modify a data control interval

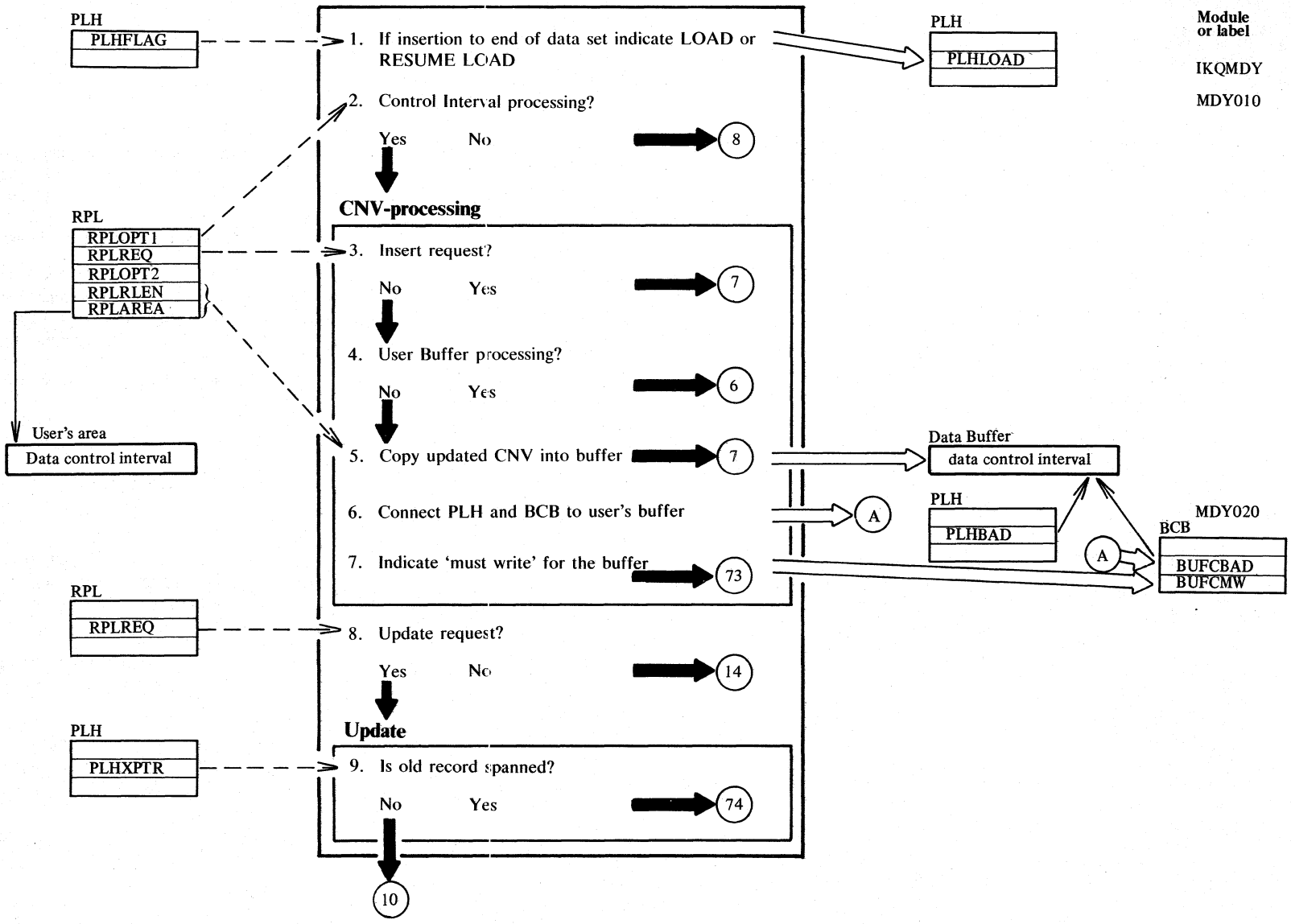


Diagram GO2. Modify a data control interval

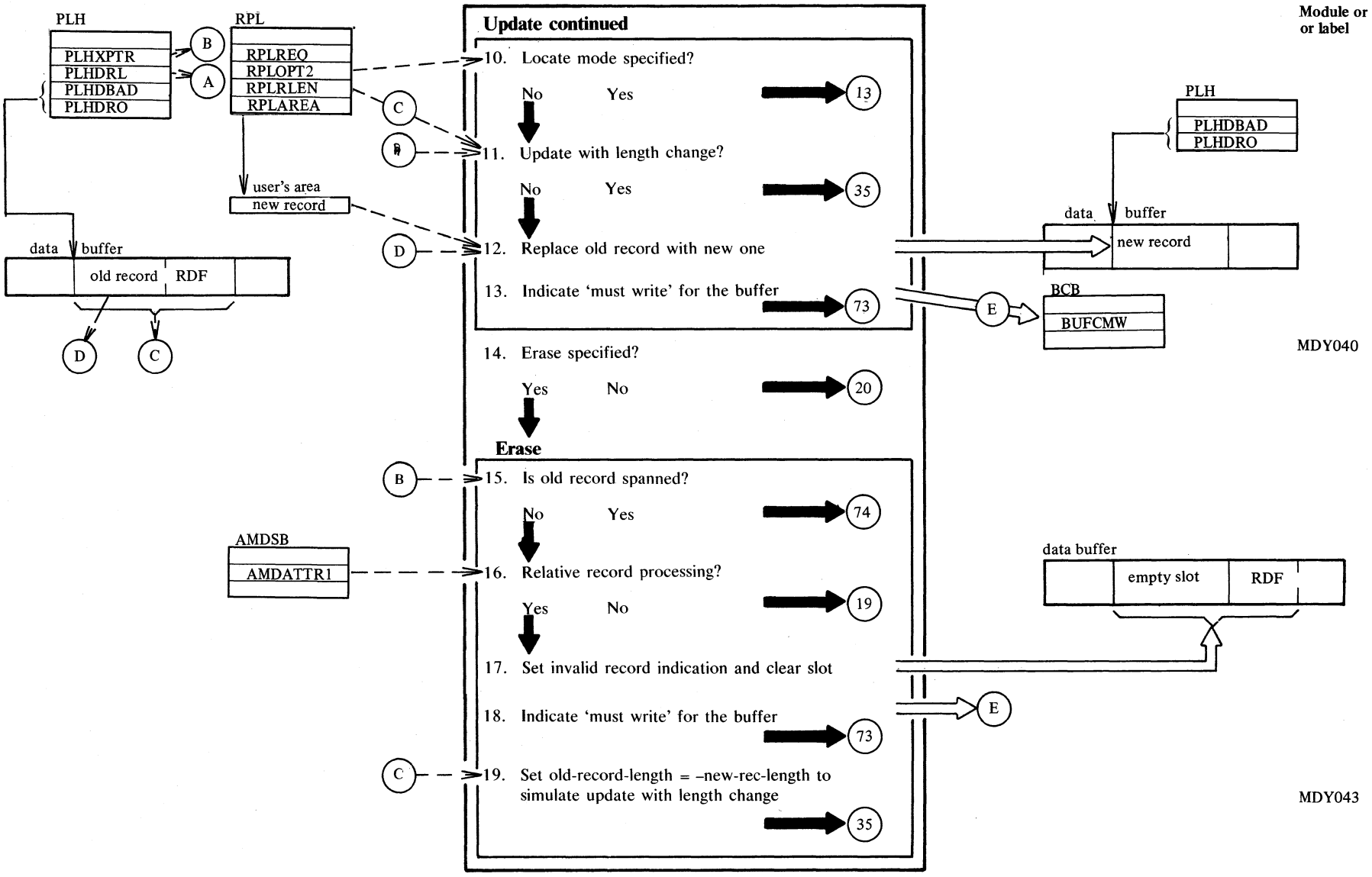


Diagram GO3. Modify a data control interval

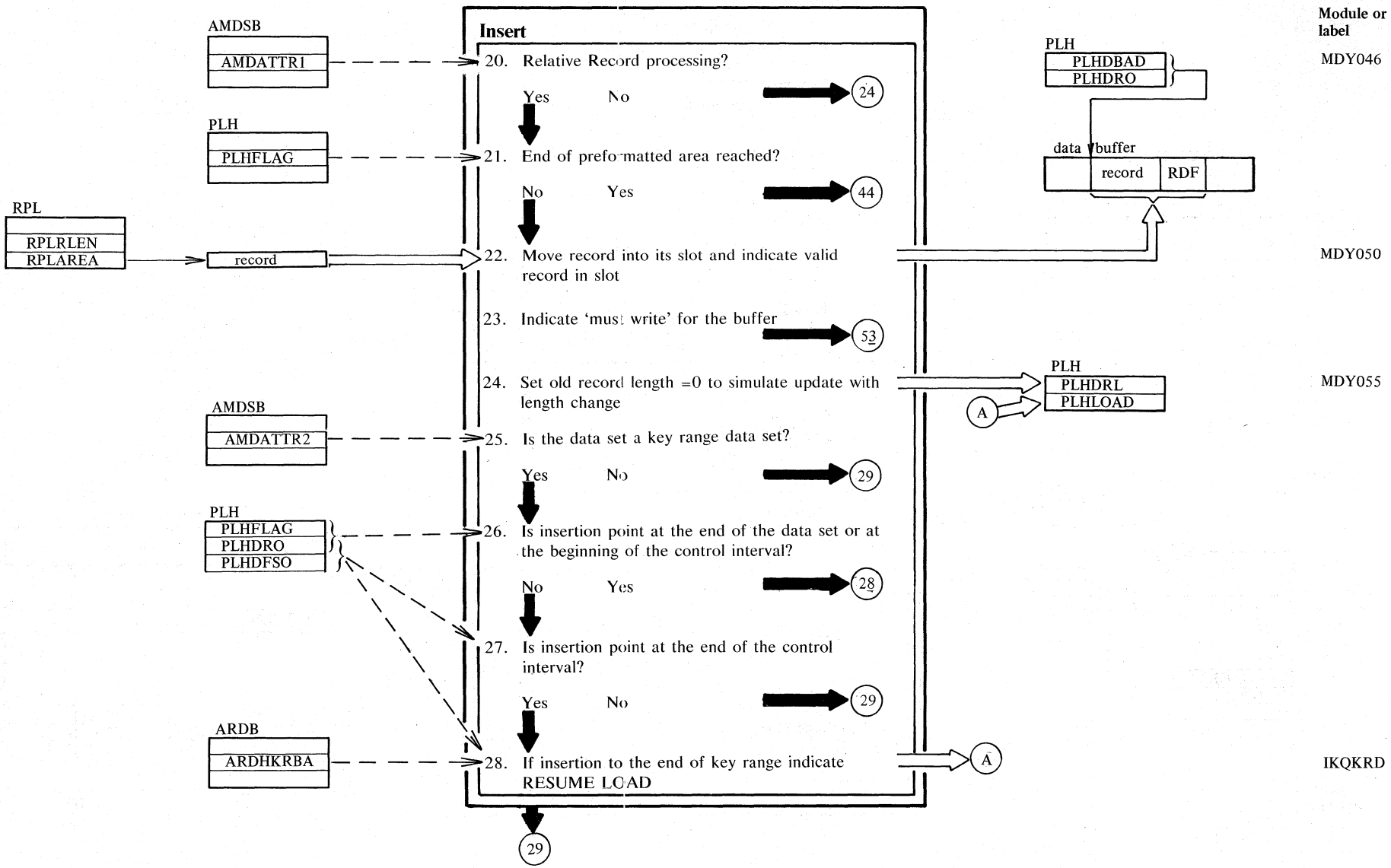


Diagram GO4. Modify a data control interval

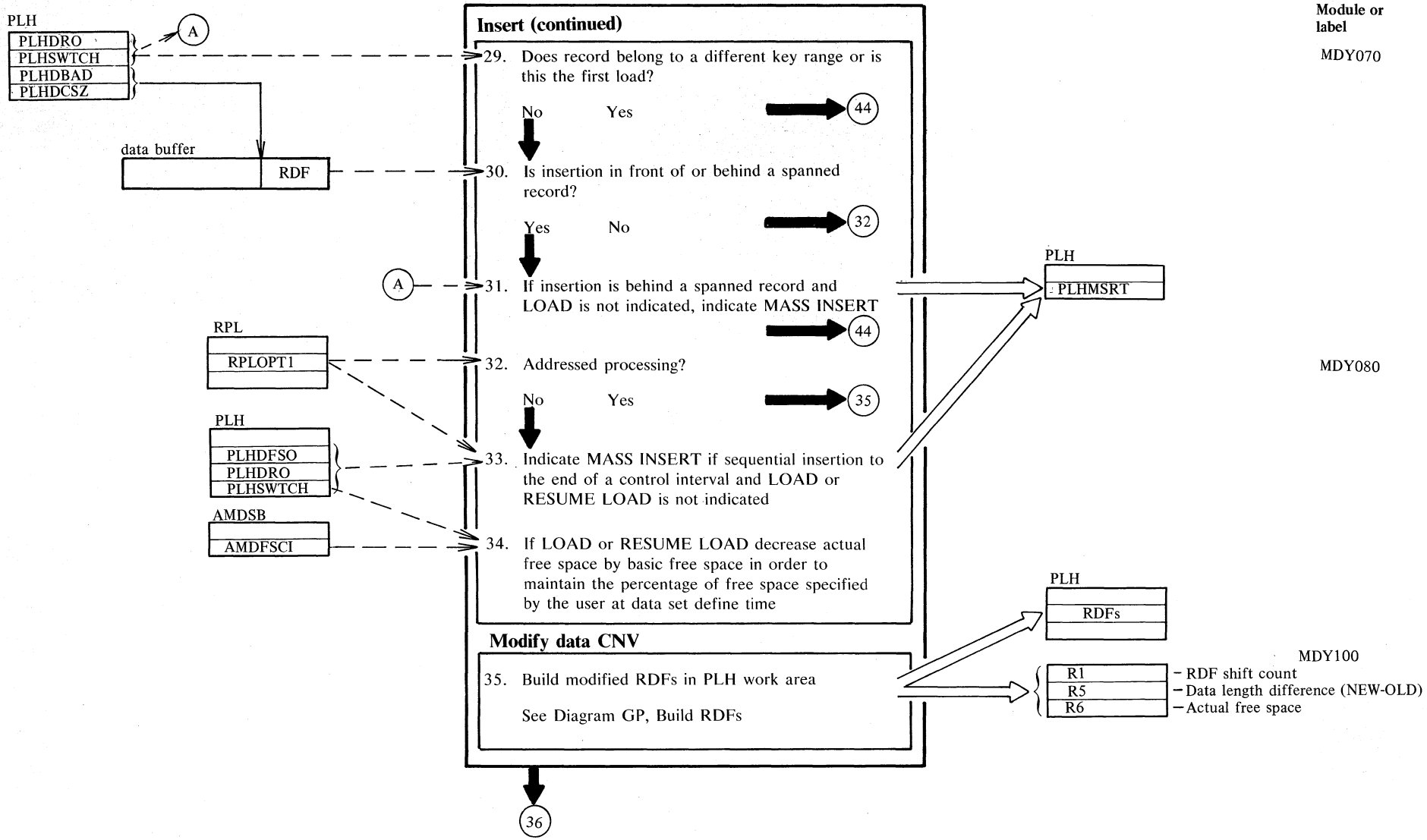


Diagram GO5. Modify a data control interval

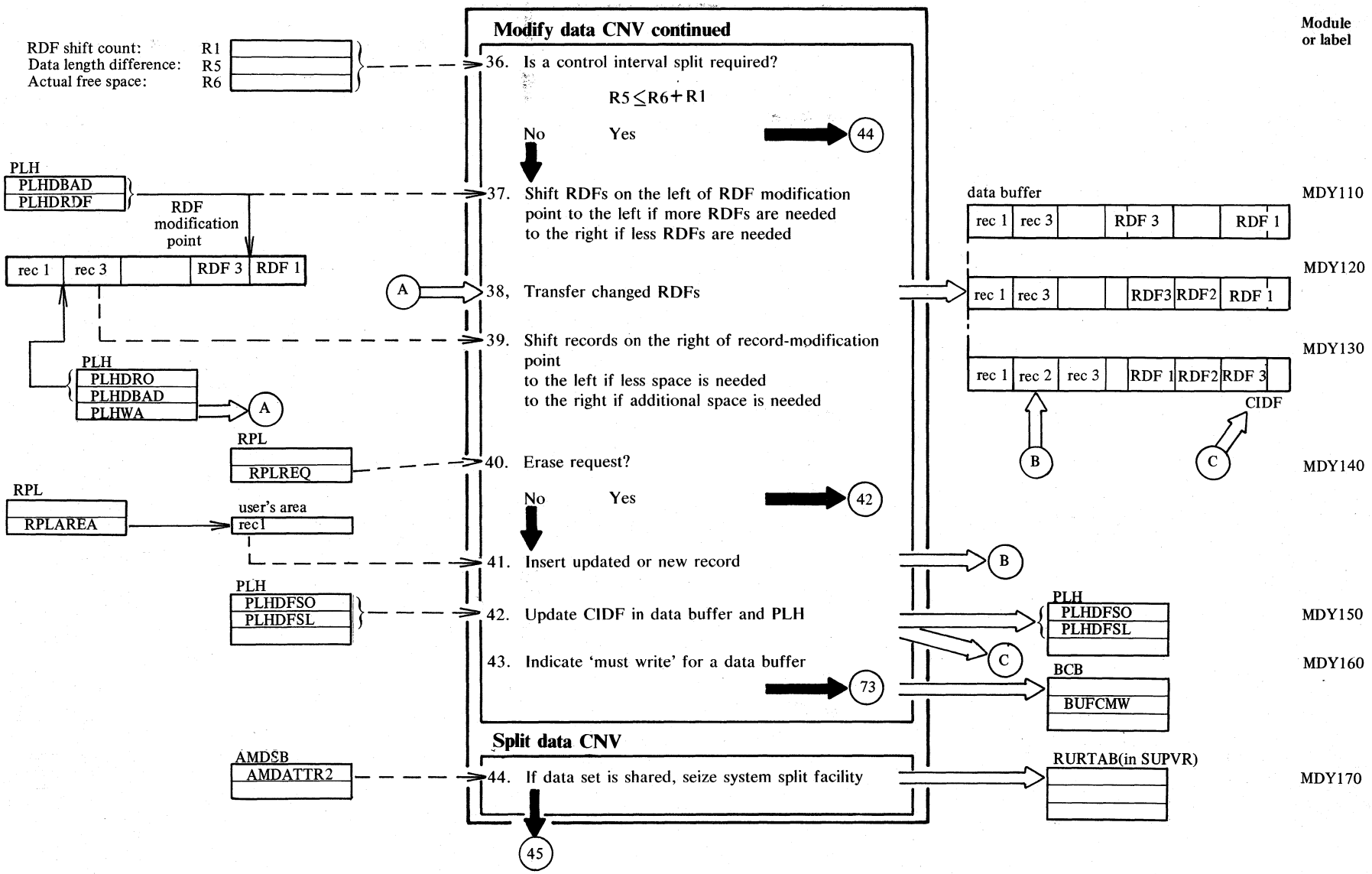


Diagram GO6. Modify a data control interval

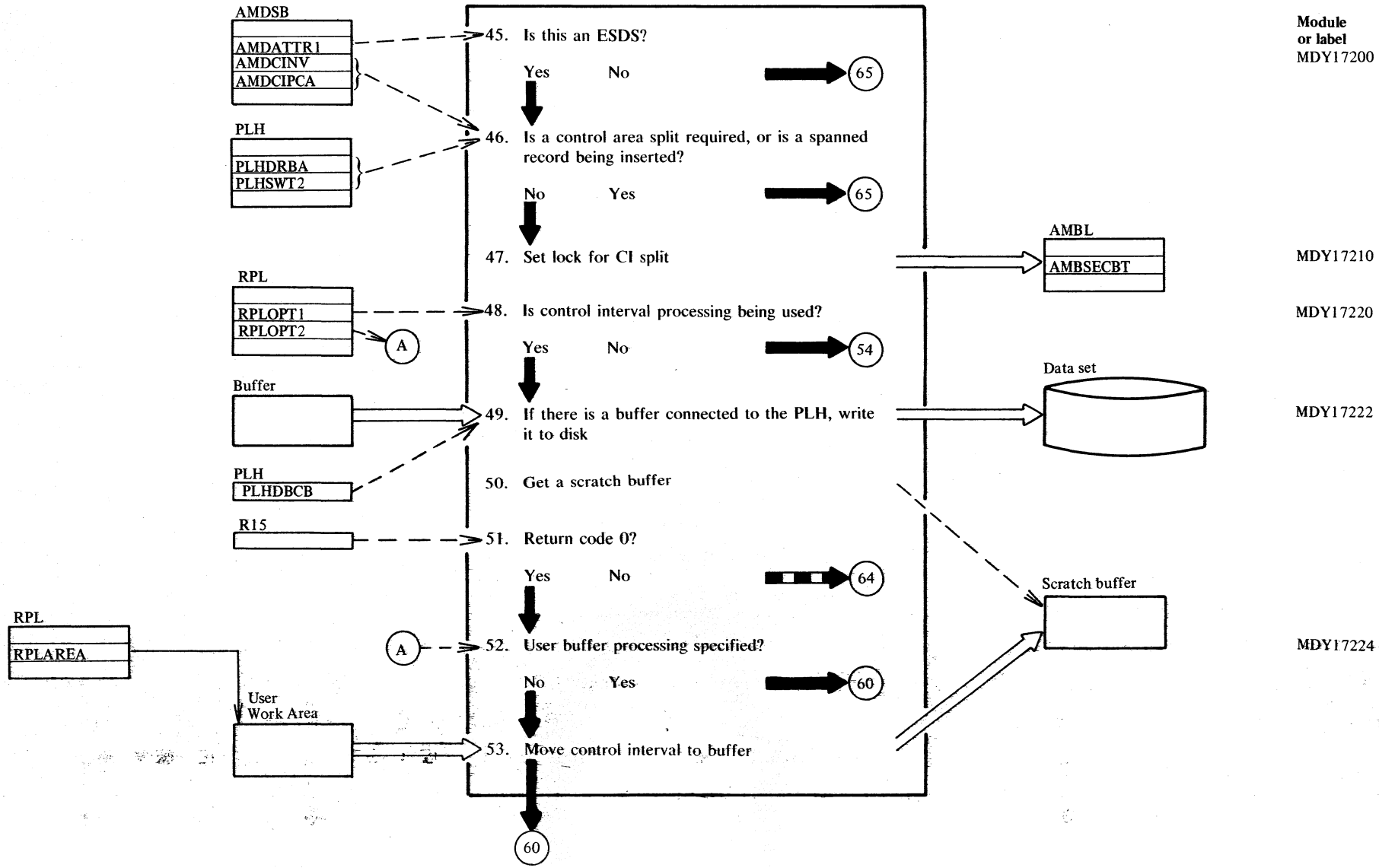
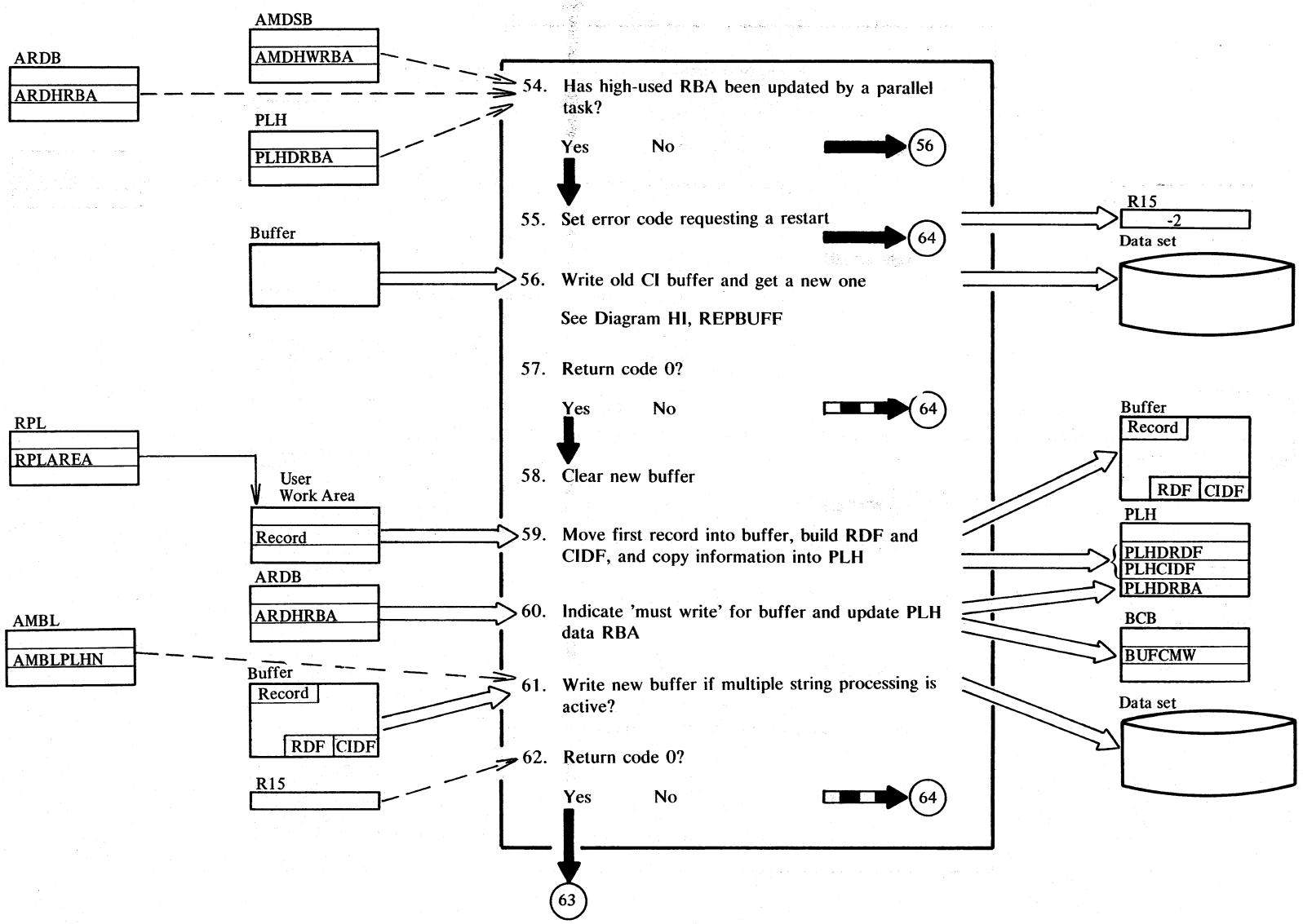


Diagram GO7. Modify a data control interval



Module or label
MDY17226
MDY17228
MDY17230

Diagram GO8. Modify a data control interval

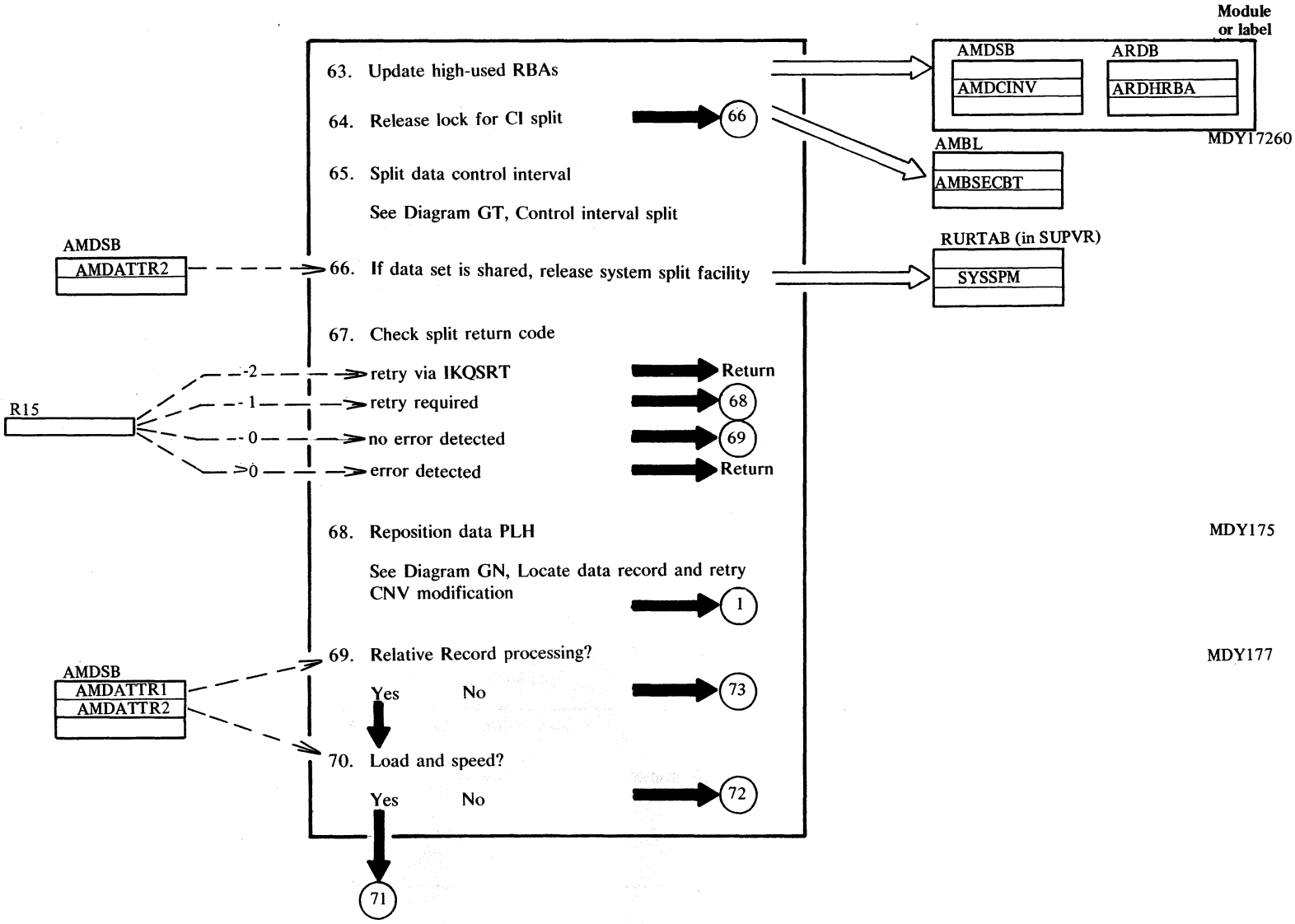


Diagram GO9. Modify a data control interval

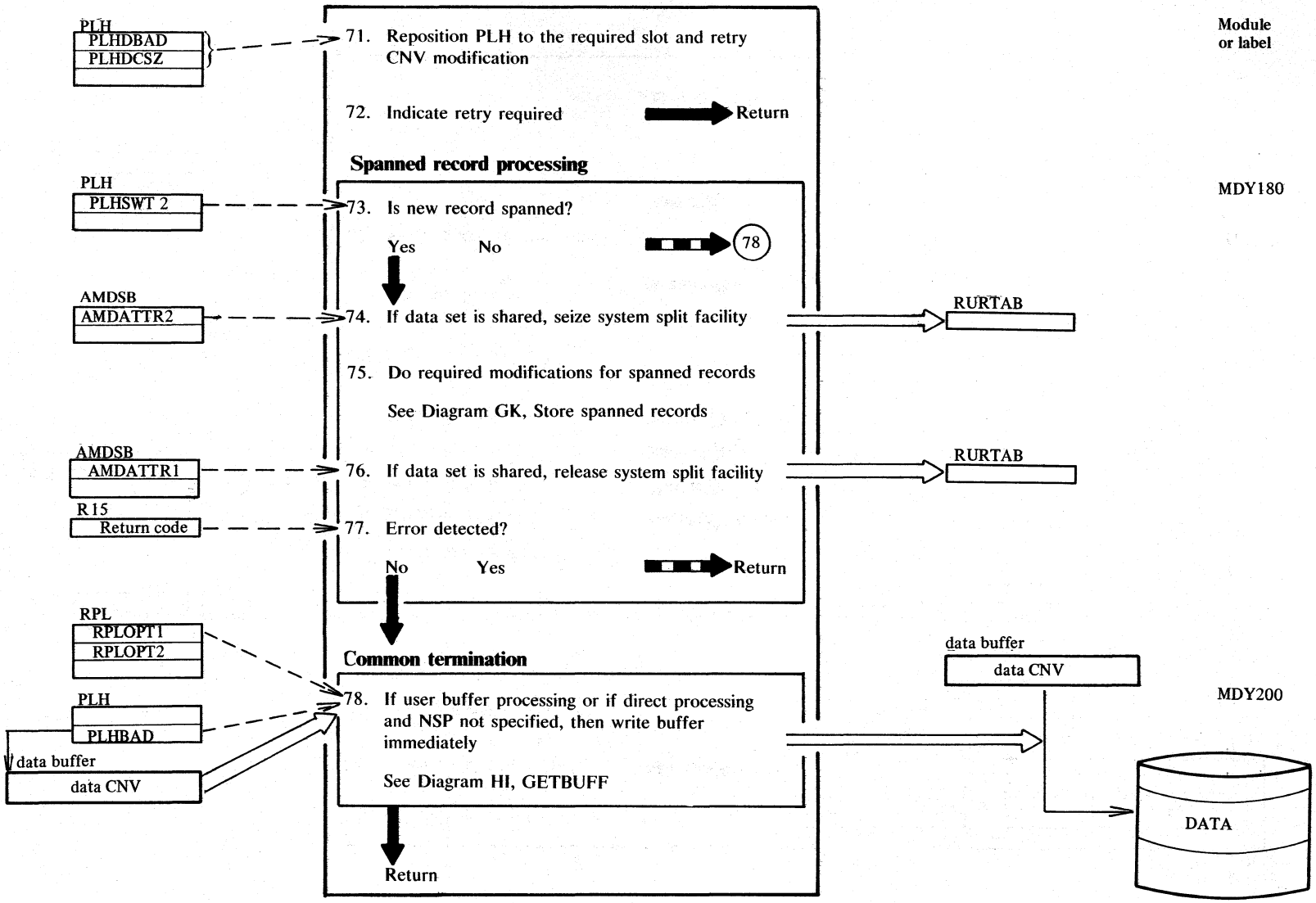
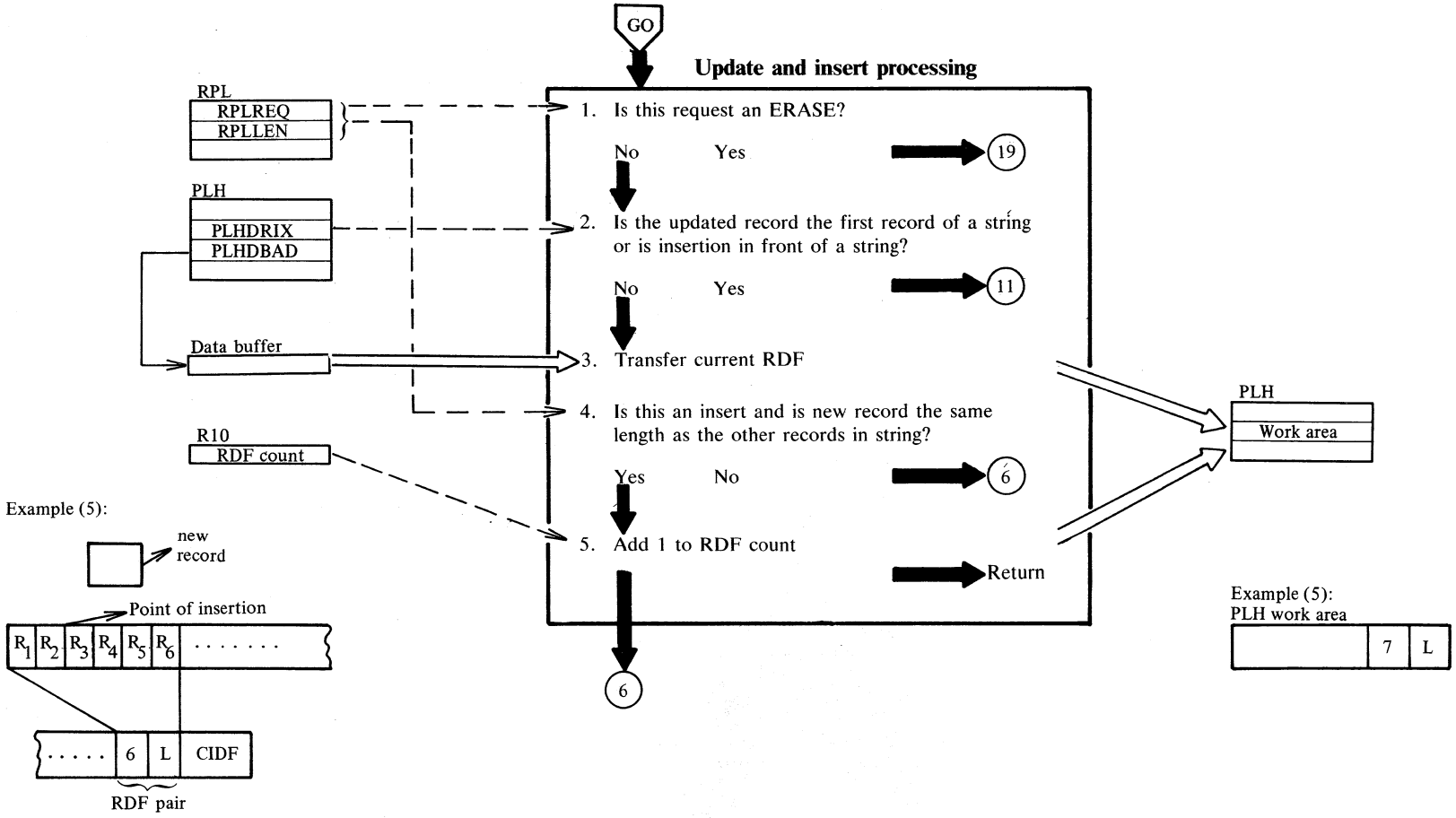


Diagram GP1. Build new and/or changed RDFs for non-spanned KSDS and ESDS

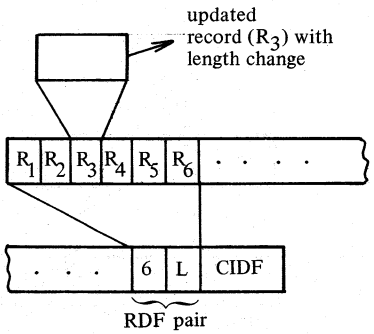


Module

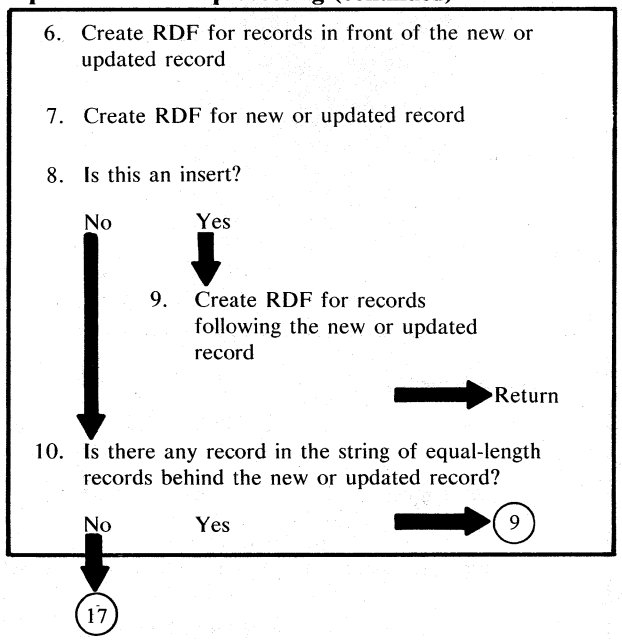
IKQBLD

Diagram GP2. Build new and/or changed RDFs for non-spanned KSDS and ESDS

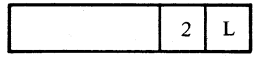
Example (6, 7, and 9):



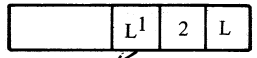
Update and insert processing (continued)



Example (6):
PLH work area



Example (7):
PLH work area



length of updated record

Example (9):
PLH work area

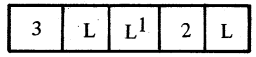
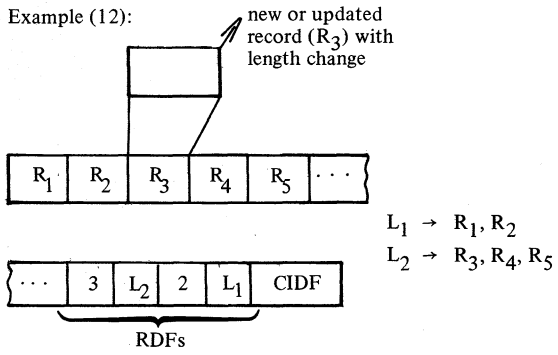
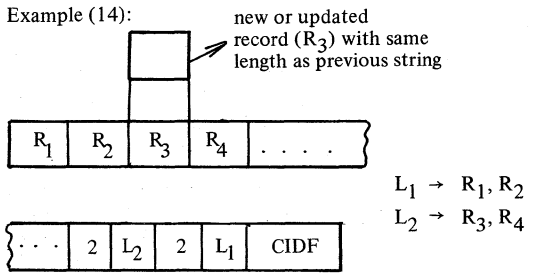


Diagram GP3. Build new and/or changed RDFs for non-spanned KSDS and ESDS

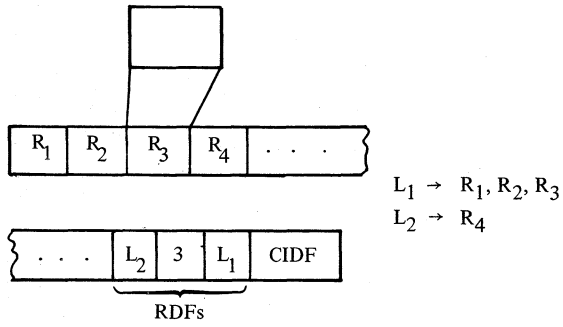
Example (12):



Example (14):



Example (18):



PLH work area

	L_2	2	L_1

Update and insert processing (continued)

11. Can the new or updated record be combined with the preceding string of records (if there is a preceding string of records)?
    - No
    - Yes → 14
12. Create a single-record-length-count RDF for the new or updated record
13. Is this an insert or an update?
    - Insert → 17
    - Update → 16
14. Attach new or updated record to previous string of equal-length records
15. Is this an insert or an update?
    - Update
    - Insert → Return
16. Does the current RDF have an RDF count?
    - No
    - Yes → 9
17. Can the last RDF in the work area be combined with the next RDF (if there is a next RDF)?
    - Yes
    - No → Return
18. Combine last RDF in work area with next RDF

Return

Module
IKQBLD

Example (12):
PLH work area

	L

Example (14):
PLH work area

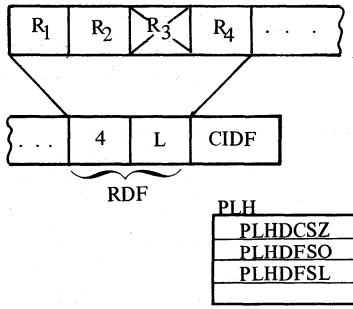
	3	L_1

Example (18):
PLH work area

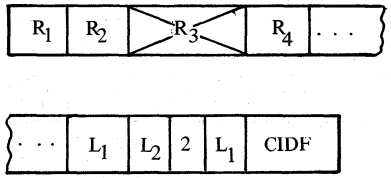
	2	L_2	2	L_1

Diagram GP4. Build new and/or changed RDFs for non-spanned KSDS and ESDS

Example (20):



Example (22):



$L_1 \rightarrow R_1, R_2$
$L_2 \rightarrow R_3$
$L_3 \rightarrow R_4$

Erase processing

19. Does the current RDF have a count?

    Yes — No → (21)

20. Build a replication-count RDF with the current RDF count reduced by 1 (for the record to be deleted) → Return

21. Can next RDF be combined with previous RDF (if there are next and previous RDFs)?

    Yes — No → Return

22. Combine previous and next RDFs

Return

Module

IKQBLD

Example (20):

PLH work area

	3	L

Example (22):

PLH work area

	3	L_1

Notes for Diagram GP

	Description	Label
1-5	RDF processing takes place if the updated record is not the first record of a string or if insert record is inserted into the middle of a string. In either case, the string must consist of equal-length records.	IKQBLD00
3	If the RDF index is equal to 1, the updated or new record may have the same length as the record(s) described by the previous RDF. For update, the RDF-build routine is entered only if the length of the updated record differs from the length of the original record.	
5	If the new record has the same length as the records described by the previous RDF and the RDF index in the PLH is equal to 1, and if the previous RDF does not have an RDF, an RDF count of 2 has to be created for the previous RDF.	
6-10	RDF processing takes place if the record to be updated is in the middle of a string or if the record to be inserted into a string has a length different from the length of the records in the string. In either case, the string must consist of equal-length records and the RDF must be split.	BLD010
11-18	RDF processing takes place if the first and possibly only record in a string is updated or if a record has to be inserted in front of the string. In either case, the string must consist of equal-length records.	BLD040
9	RDF processing takes place for records to the right of the new or updated record if the insertion is into the middle of a string of equal-length records and the length of the new record is different from the length of the records in the string or for any update with length change.	BLD160
19-20	For an ERASE request, the RDF-count of the RDF describing the record to be erased is reduced by 1 if the count was previously greater than 1. If the count was 2, the new RDF will not have a replication RDF.	BLD180
21-22	If the record to be erased was described by an RDF without count (the record being the only one of that length), and if the neighboring RDFs have the same length count (that is, represent records of the same length), the RDFs can be combined to a single RDF with replication count.	BLD210

Diagram GQ1. GETNXT: Get next buffer and read ahead

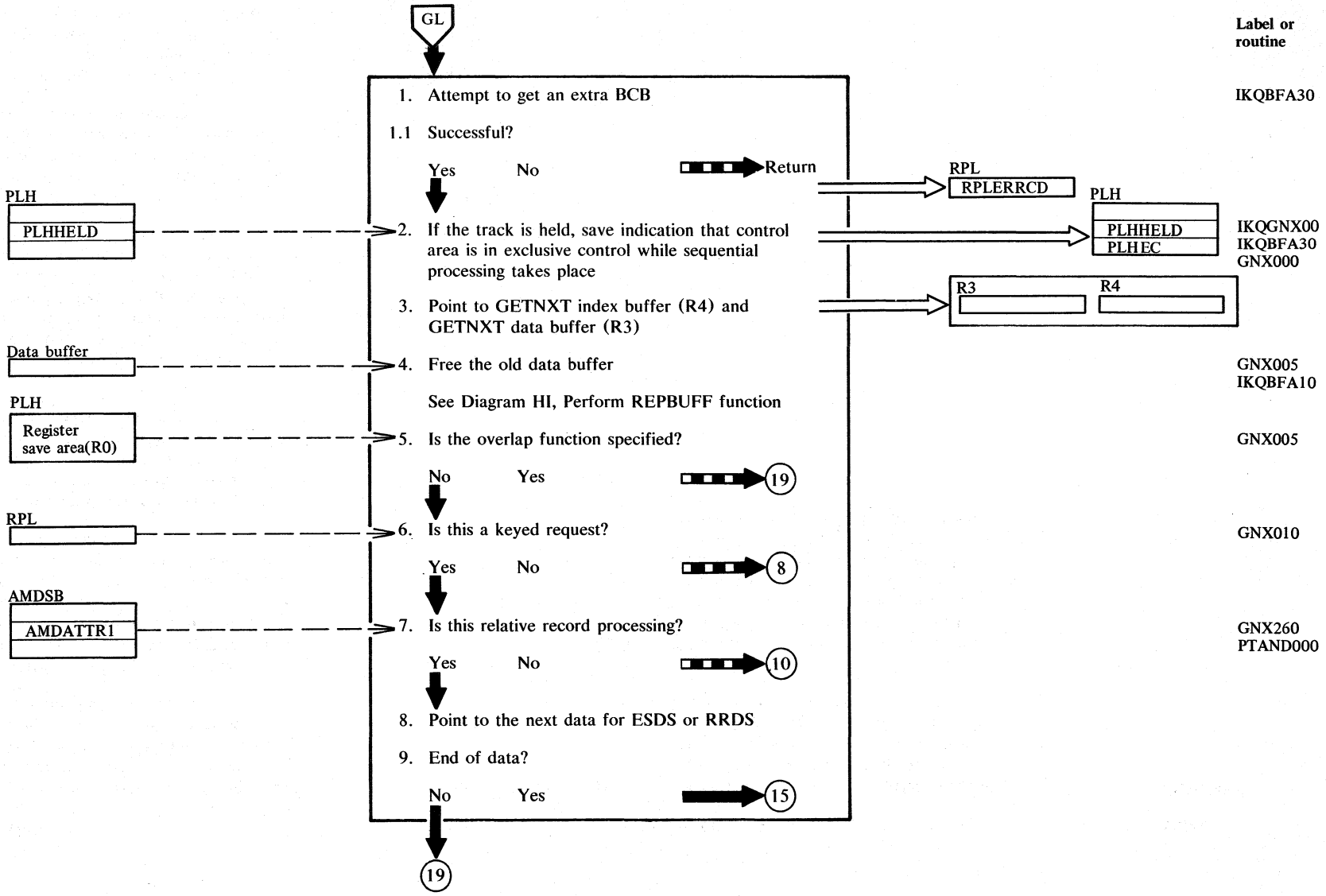


Diagram GQ2. GETNXT: Get next buffer and read ahead

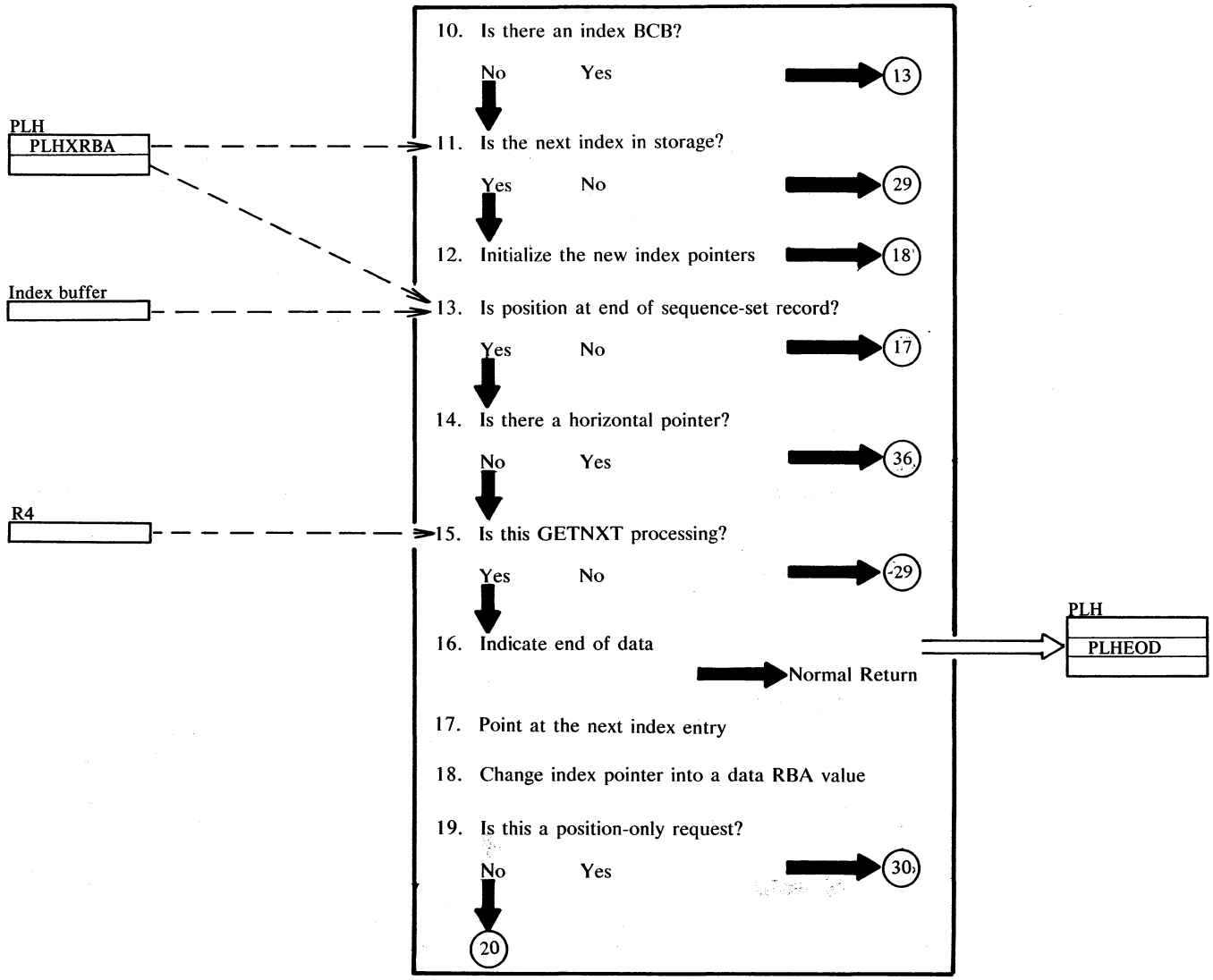


Diagram GQ3. GETNXT: Get next buffer and read ahead

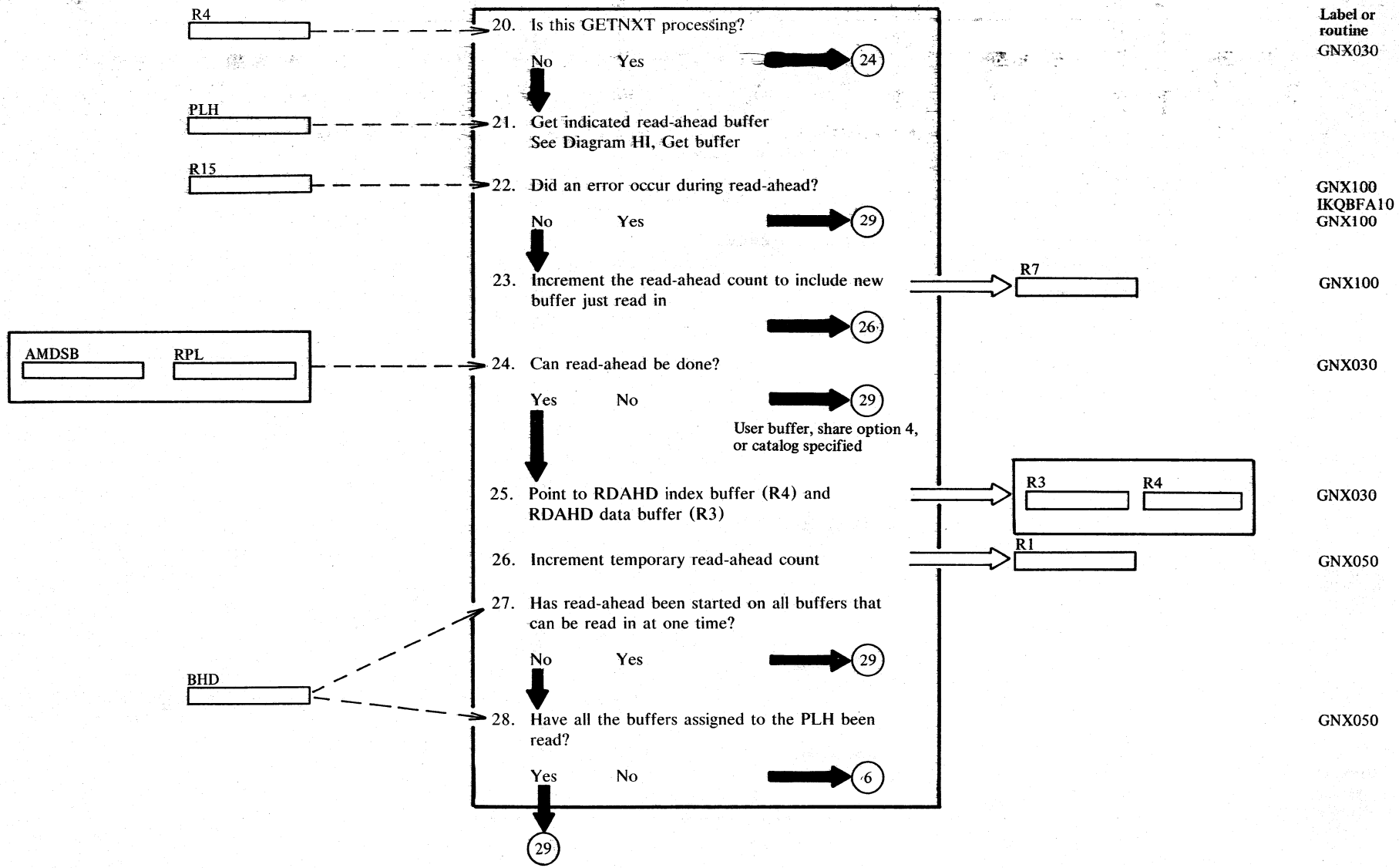


Diagram GQ4. GETNXT: Get next buffer and read ahead

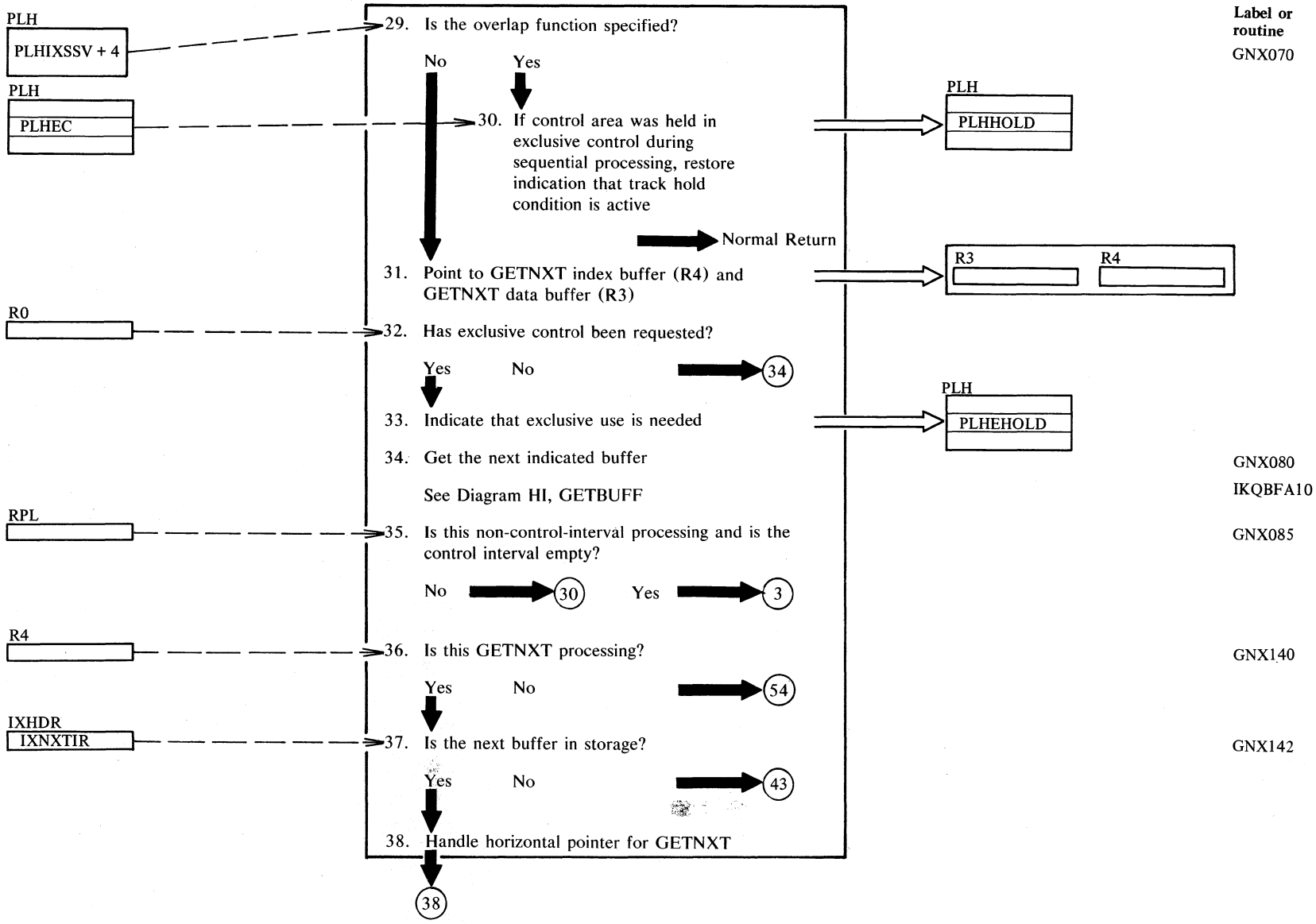


Diagram GQ5. GETNXT: Get next buffer and read ahead

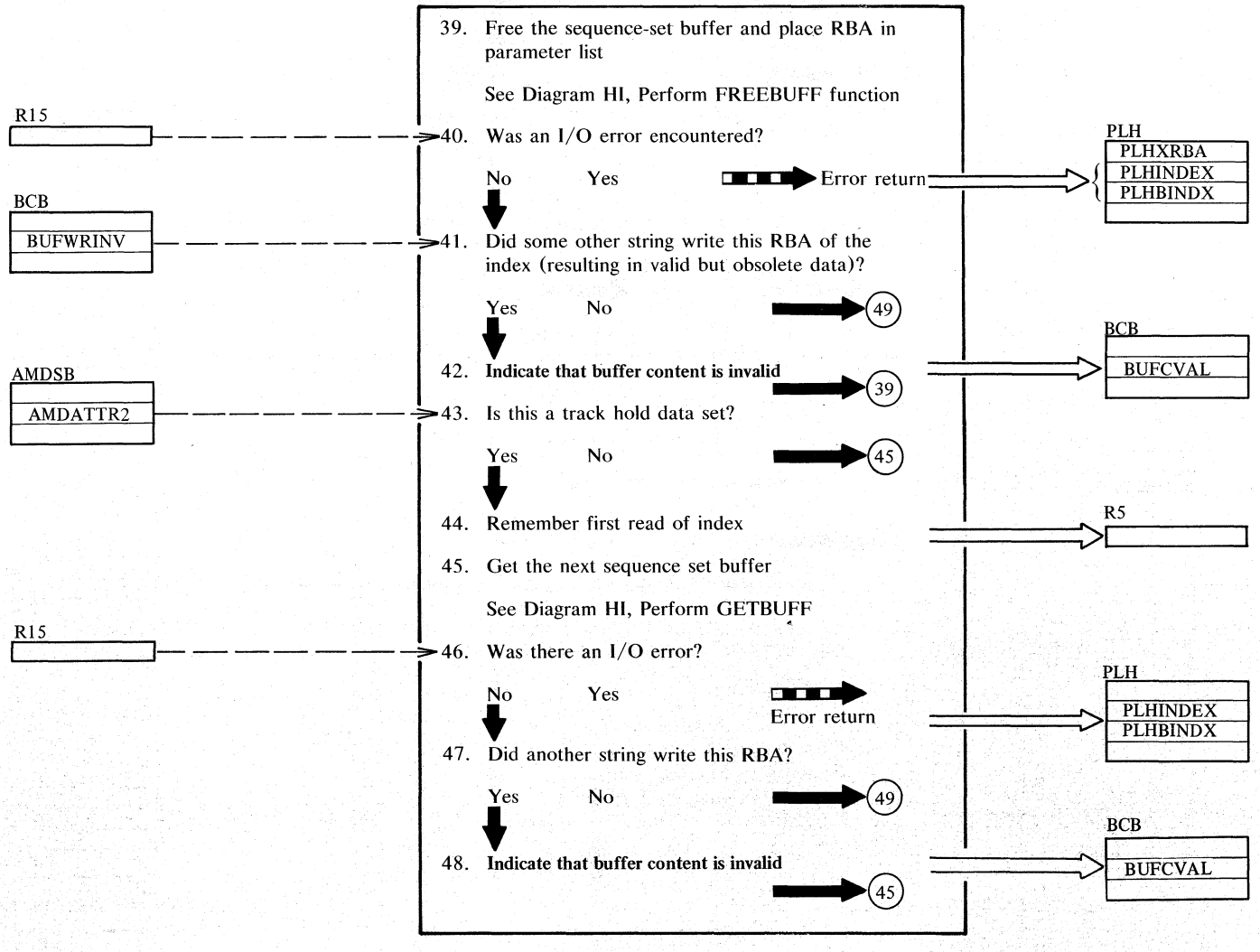
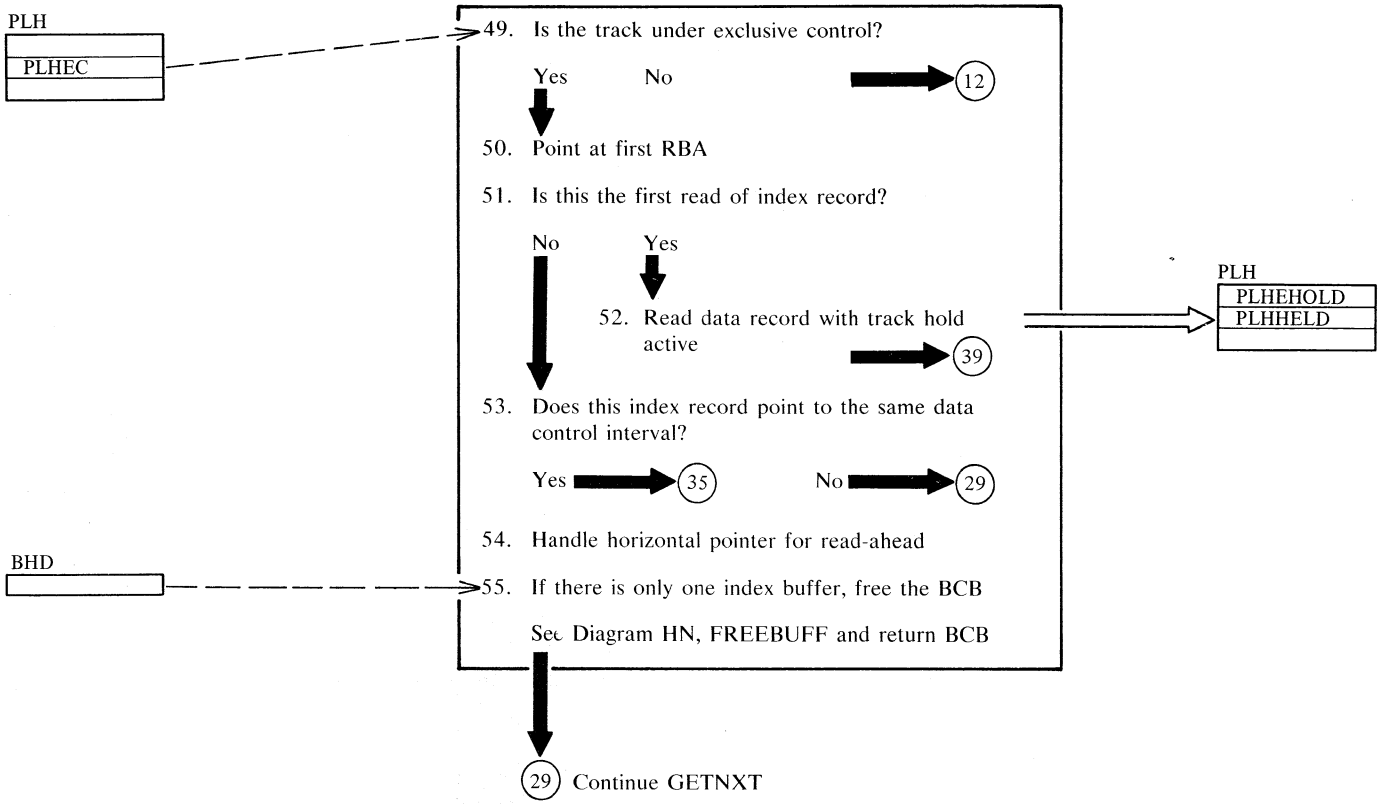


Diagram GQ6. GETNXT: Get next buffer and read ahead



Label or routine

GNX180

GNX200

GNX240
IKQBFA20

Diagram GR1. GET PREVIOUS: Retrieve previous record

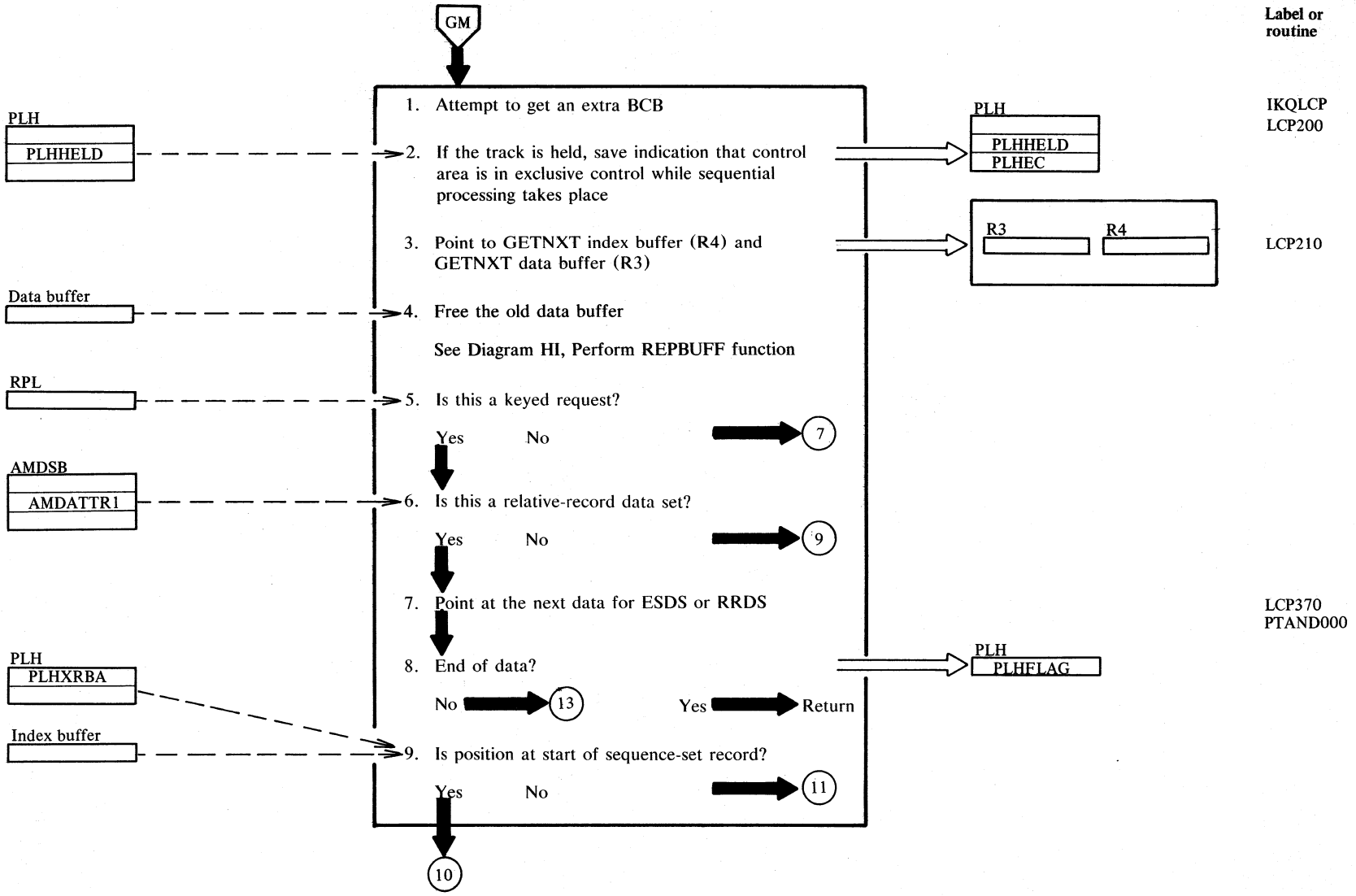


Diagram GR2. GET PREVIOUS: Retrieve previous record

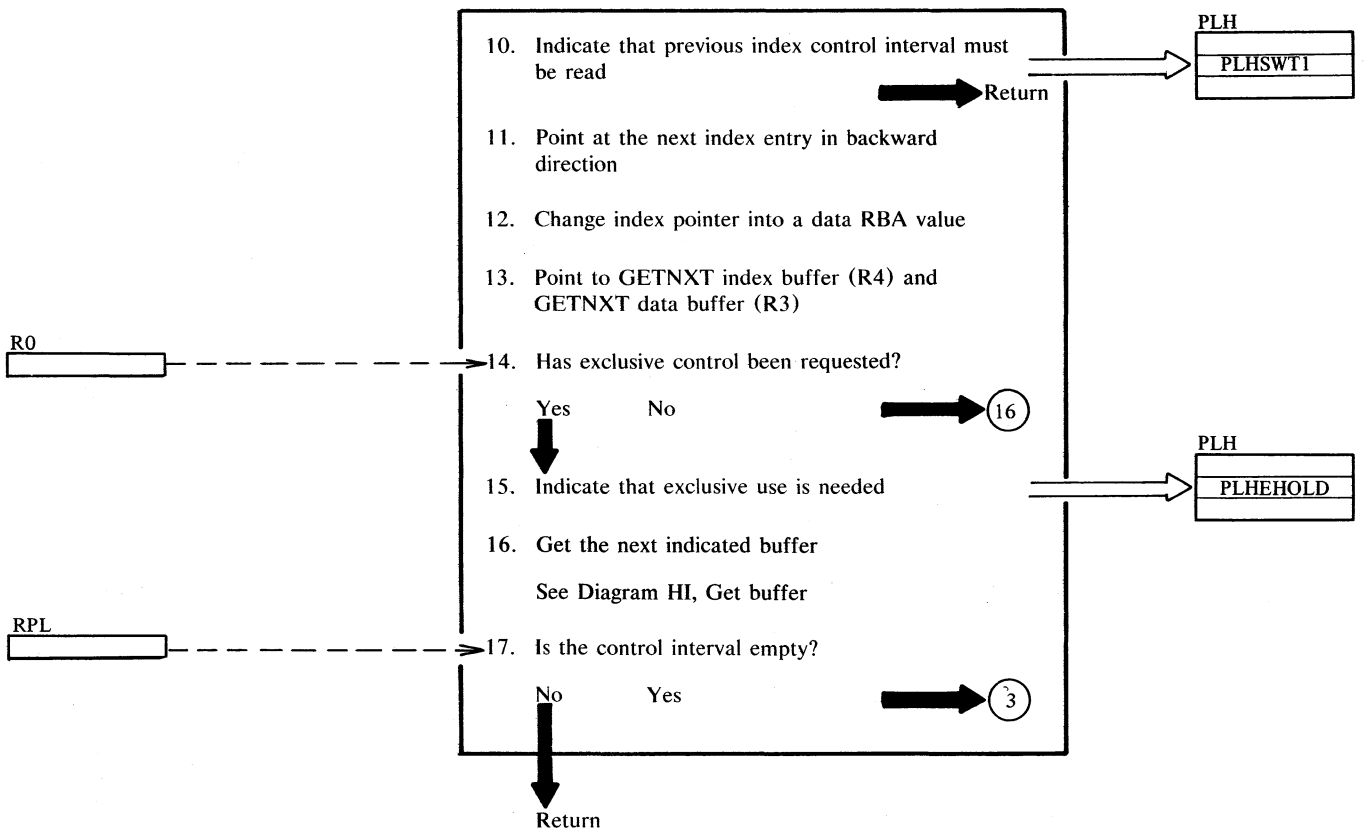


Diagram GS1. VERIFY: Reestablish high-used and high-key RBAs

Module

IKQVFY

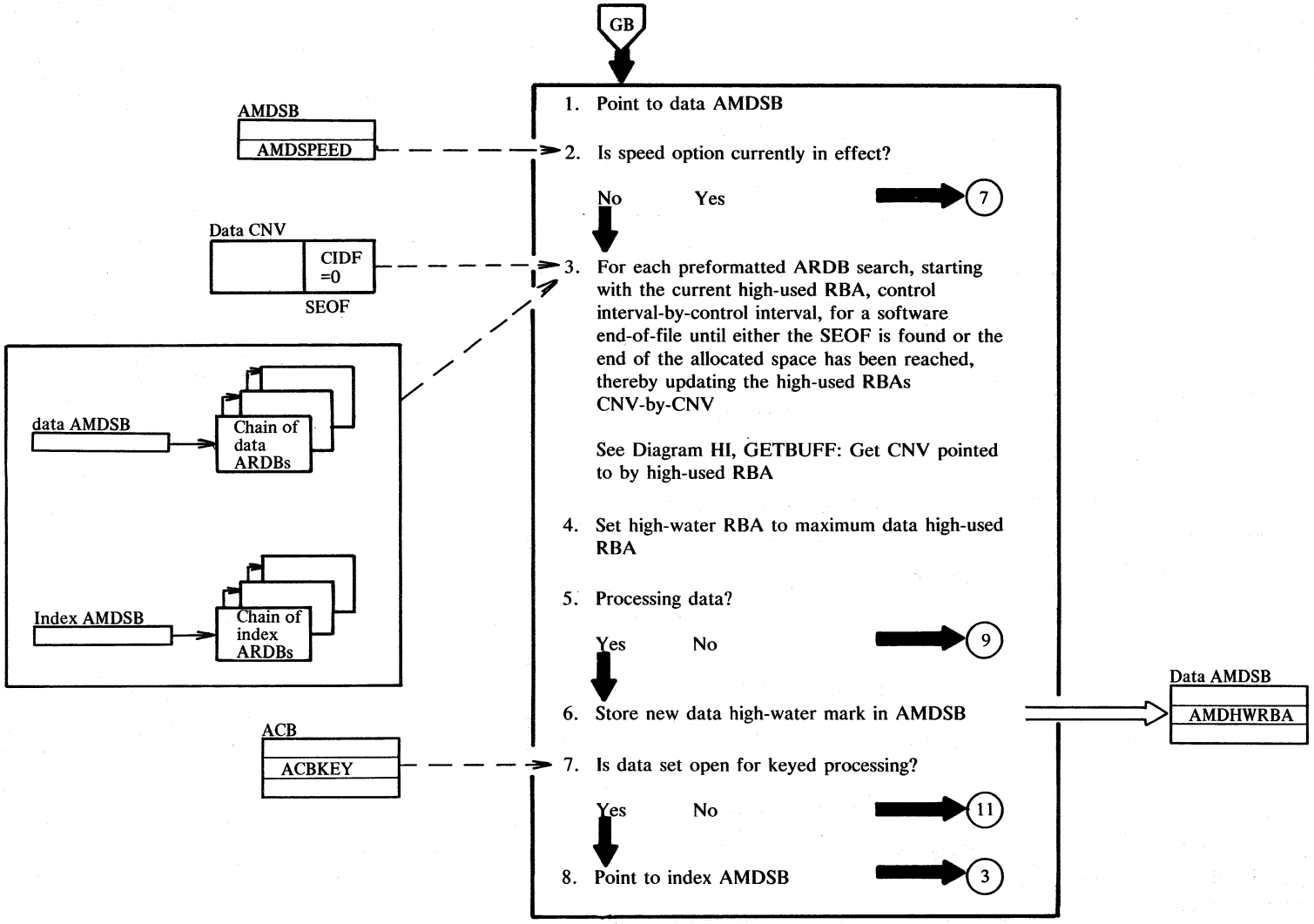


Diagram GS2. VERIFY: Reestablish high-used and high-key RBAs

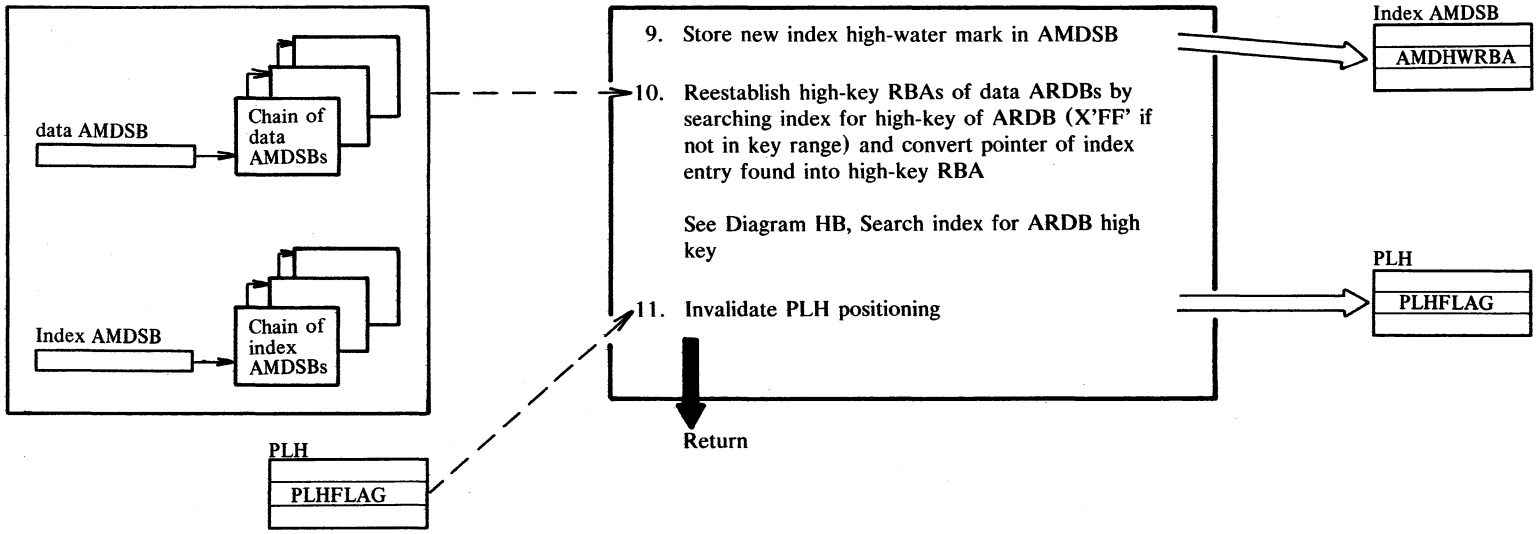
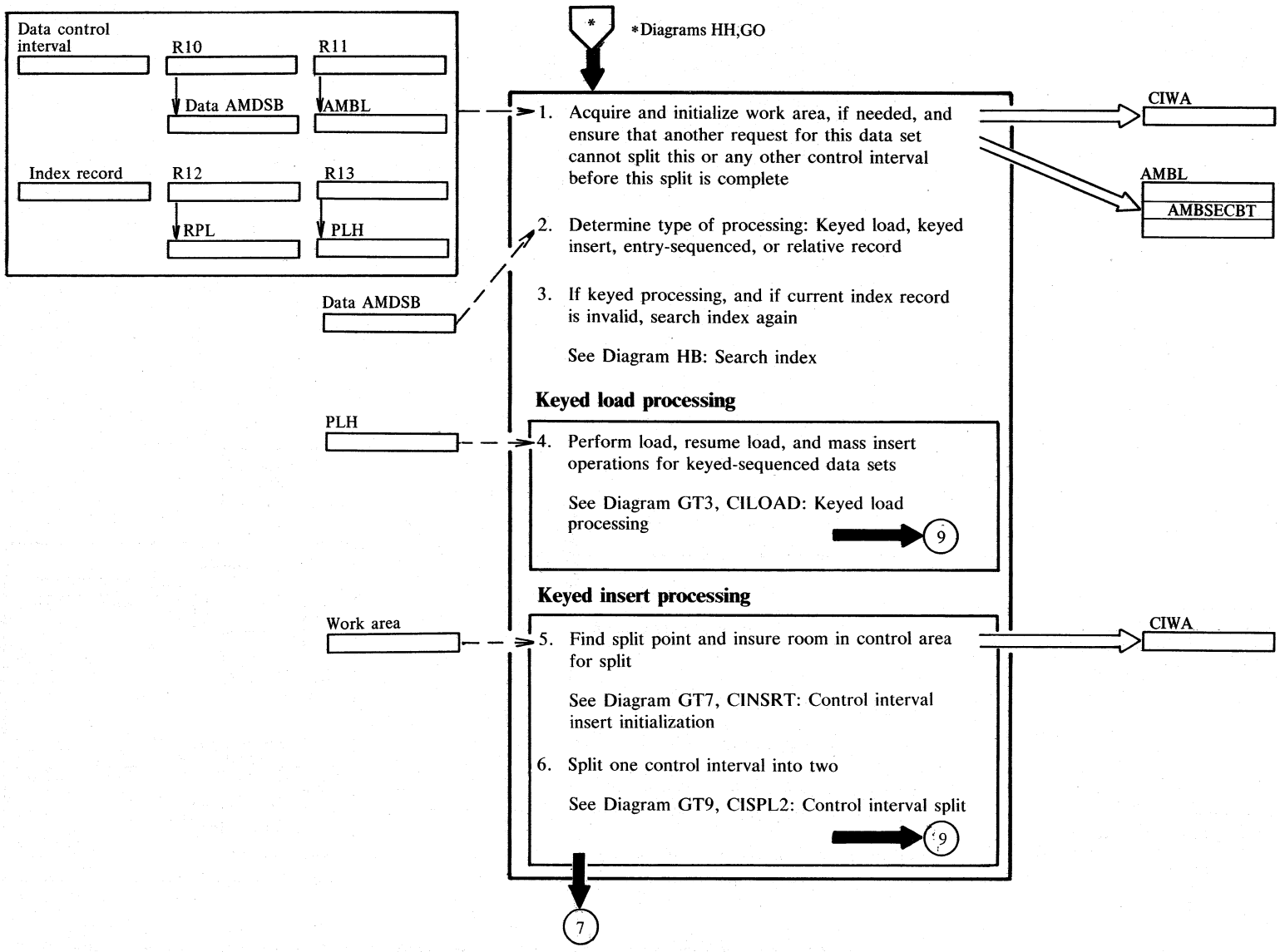
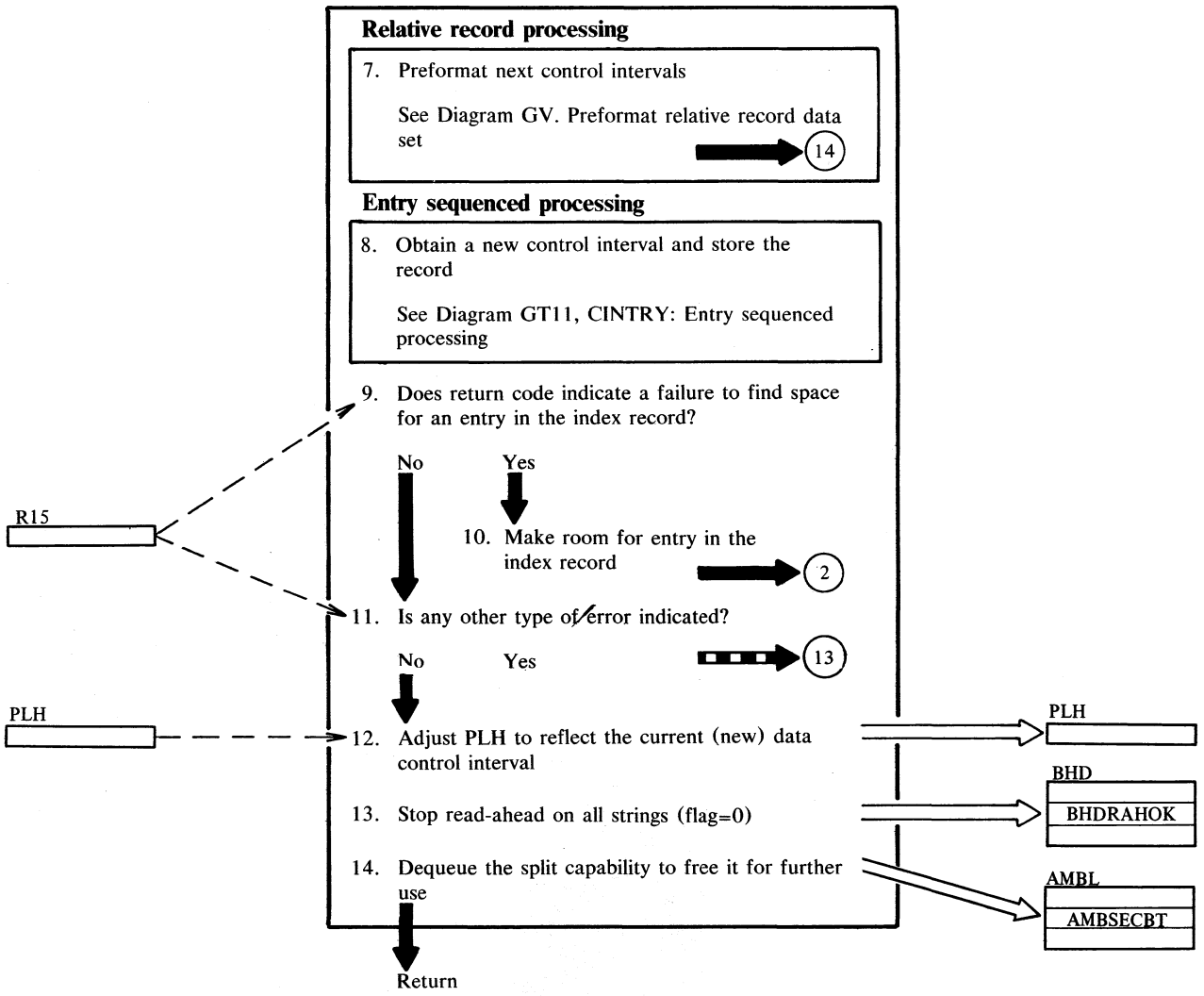


Diagram GT1. Control interval split



Routine or label
IKQCIS00
CIINIT
IKQCIS10
IKQCIS40
IKQCIS60
IKQCIS80

Diagram GT2. Control interval split



Notes for Diagram GT1 and GT2

	Description	Module or routine	Label
10	This routine is a recovery routine, called when control interval split processing must be interrupted after partial completion due to unavailability of index space in the first-level index record. Partially filled buffers are purged, the control area split switch is turned on, and control interval split is restarted.	CINXRM	IKQCIS94

Diagram GT3. CILOAD: Keyed load processing

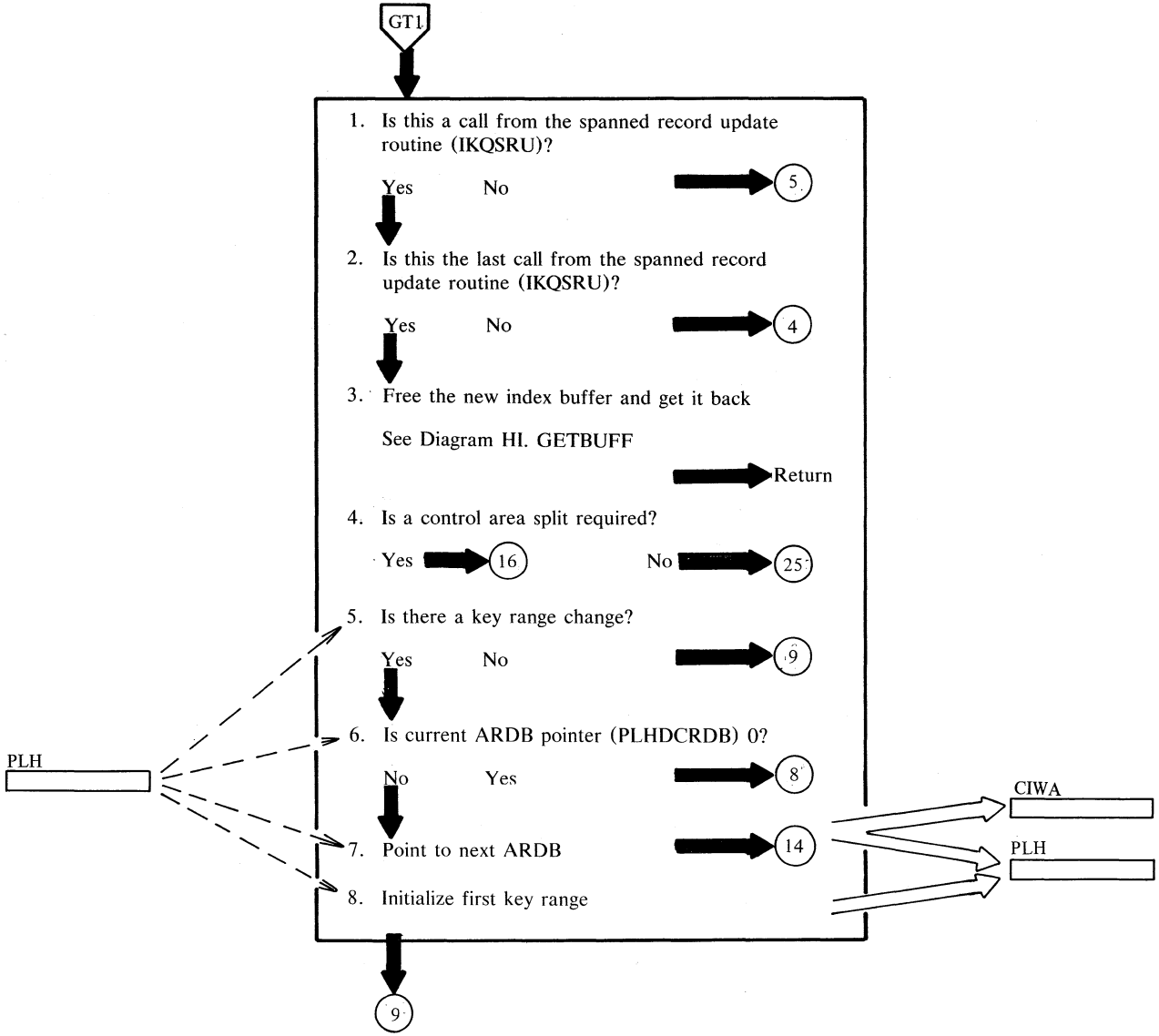


Diagram GT4. CILOAD: Keyed load processing

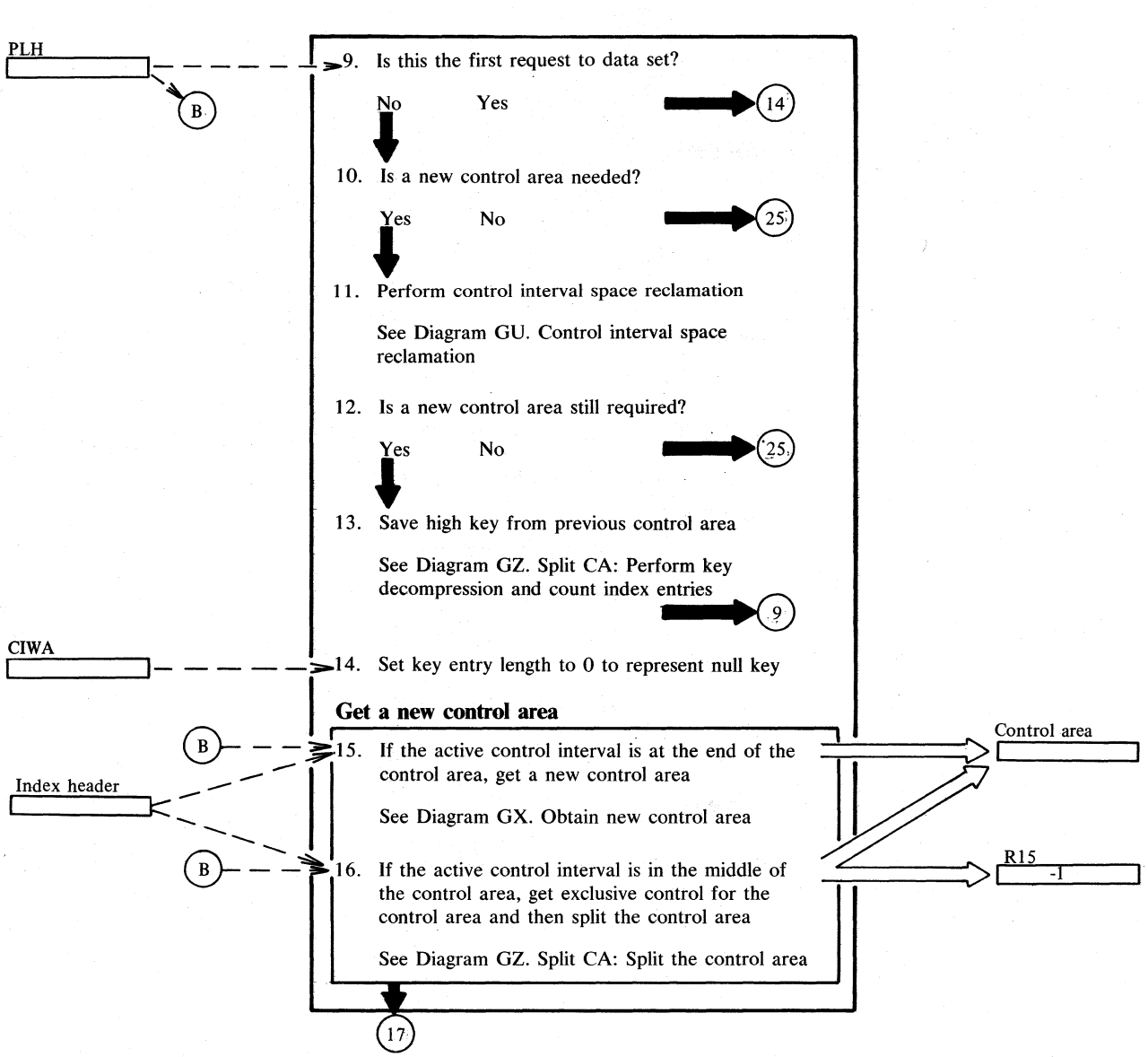


Diagram GT5. CILOAD: Keyed load processing

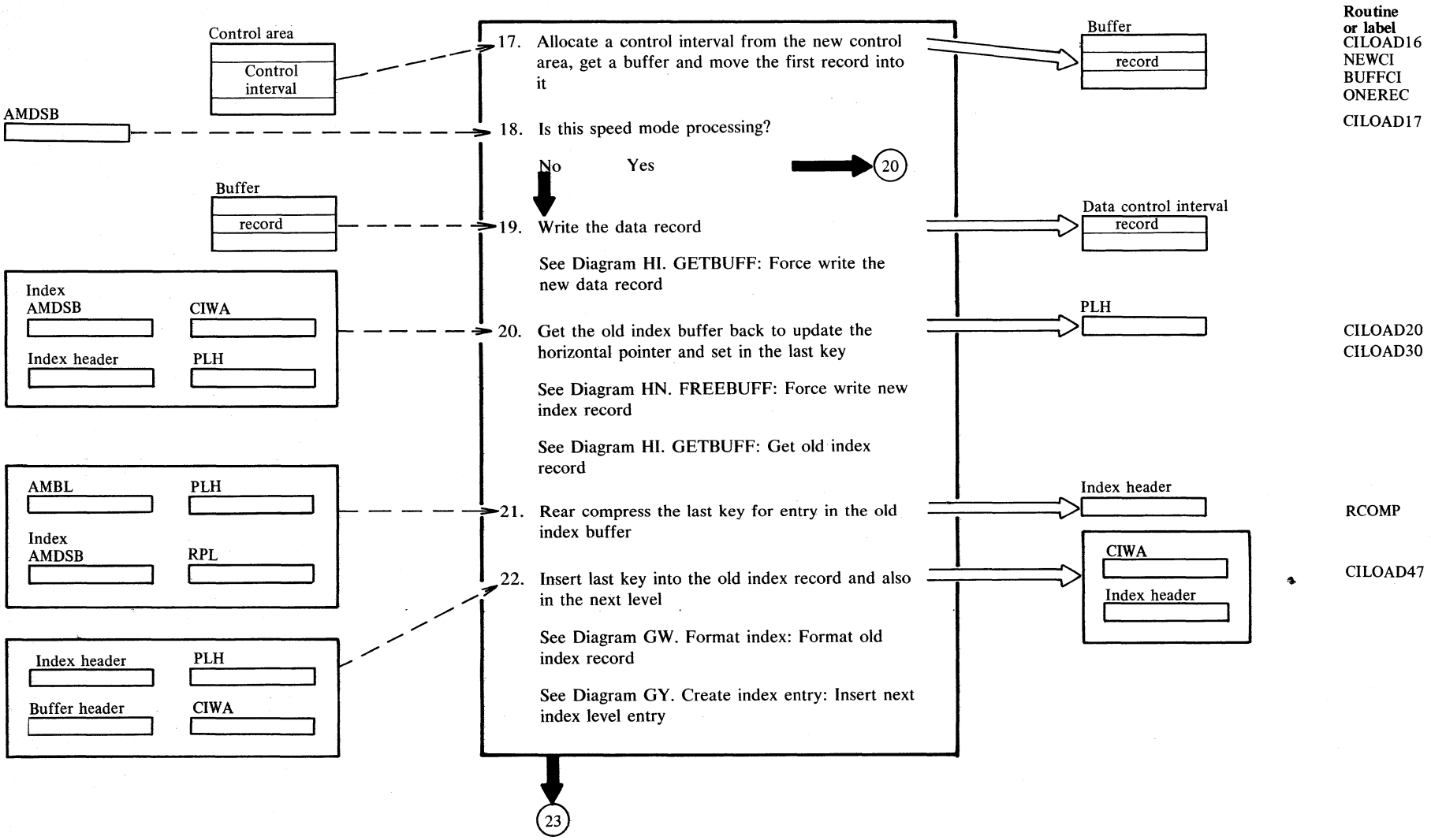
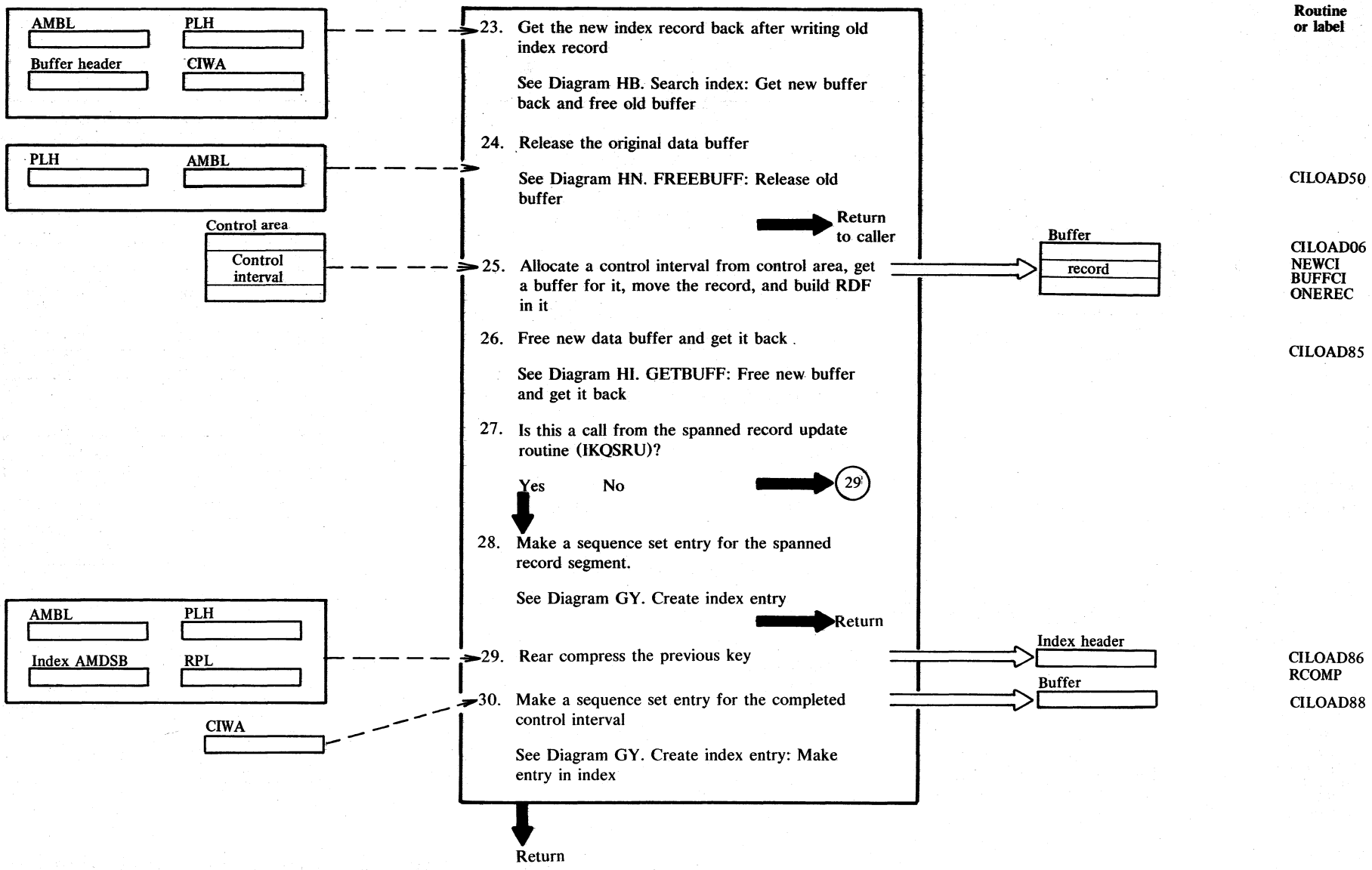


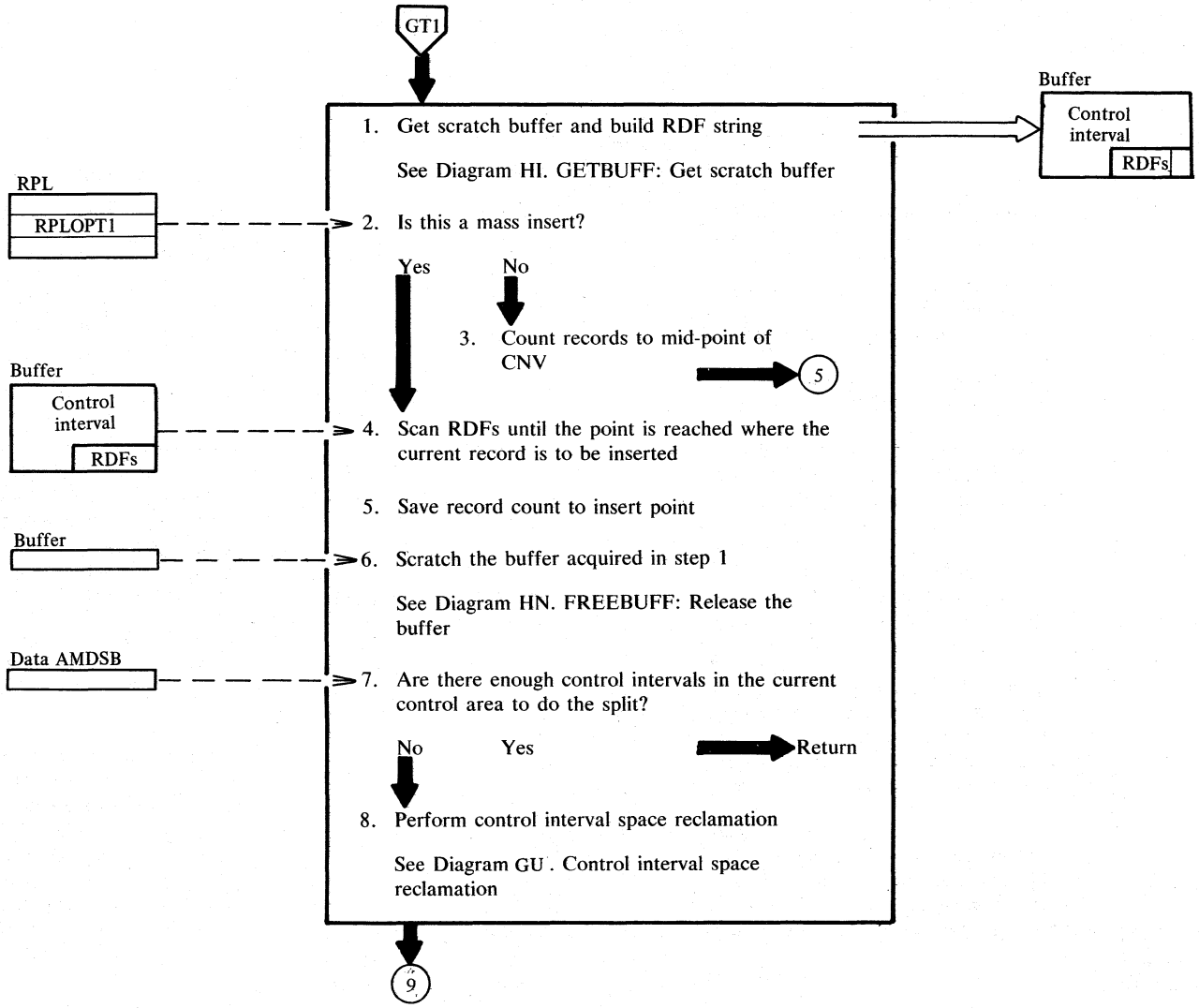
Diagram GT6. CILOAD: Keyed load processing



Notes for Diagrams GT3 - GT6

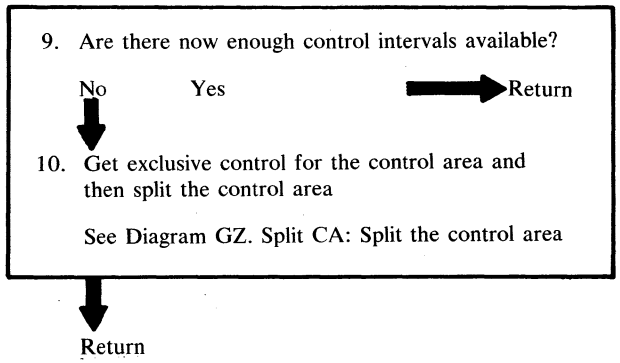
	Description	Module or routine	Label
5	For key range processing, pointers in the PLH are set for both the current ARDB and the ARDB for the key range in which the insert activity will take place. If these are not the same, one key range must be cleaned up before the next can be accessed. Any intervening key ranges must also be formatted and have index records built for them.	IKQCIS00	CILOAD03
7	The pointer to the current ARDB is saved in the work area and the pointer to the next ARDB is updated in the PLH.	IKQCIS00	
11	If the CI split routine determines that a CA split is necessary (as there are no free CIs available), it first calls IKQCIR, which searches for CIs whose contents have been erased. their space and informs IKQCIS, which then attaches a reclaimed CI. If there are no CIs which can be reclaimed, IKQCIR informs IKQCIS, which then continues with a CA split.	IKQCIR	CILOAD11
15	Key range processing is the same as normal except IKQNCA00 allocates only from the ARDB for the target key range. When a key range is exited, the dummy (F=L=0) index entry is replaced by the high key of the key range. If a key range is skipped, an index record must nevertheless be allocated and a control area formatted.	IKQNCA00	
16	If a mass insert is being made, the packing factor is ignored, and the control area is split at the end of the active control interval (unless it is the last in the control area - see step 8).	IKQCAS00	CILOAD12
20	If processing is being done in speed mode, the REPBUF step is skipped. If processing is being done in recovery mode, the write is forced by means of the combined FREEBUFF and GETBUFF, implied by REPBUF.	IKQCIS00	CILOAD20 CILOAD30
21	If there is not a key range change, the high key from the last control interval processed is picked up and rear compressed.	RCOMP	
22	If there is a key range change, the key range high key is picked up from the ARDB and replaces the key of the last entry in the record. The record is formatted and IKQIXE00 is called to make the next index level entry.	NEWCI	CILOAD60
25	In normal record processing, records are added to the end of a control interval. When the packing factor is reached, a new control interval is obtained. When the packing factor for a control area is reached, a new control area is obtained.	IKQCIS00	CILOAD47

Diagram GT7. CINSRT: Control interval insert initialization



Module, routine, or label
IKQCIS00
CINSRT
CINSRT30
CINSRT34
CINSRT40
CINSRT90
CINSRT95
IKQCIR

Diagram GT8. CINSRT: Control interval insert initialization



Module, routine, or label

CINSRT98
CASLNK

Notes for Diagrams GT7 - GT8

	Description	Module	Label
3	The halfway point in the control interval is calculated by taking into consideration not only the length of the data records, but also the length of the associated RDFs and the required CIDF.	IKQCIS00	CINSRT34
4	Compute the amount of space required for all records and their control fields up to the point of insertion. The length of the new record and its RDF are added to the prior total to give the point at which the split will take place. Both the record count and the total of all space needed up to the point of split are saved.	IKQCIS00	CINSRT40
11	If the CI split routine determines that a CA split is necessary (as there are no free CIs available), it first calls IKQCIR, which searches for CIs whose contents have been erased. If IKQCIR finds such CIs, it reclaims their space and informs IKQCIS, which then attaches a reclaimed CI. If there are not CIs which can be reclaimed, IKQCIR informs IKQCIS, which then continues with a CA split.	IKQCIR	

Diagram GT9. CISPL2: Control interval split

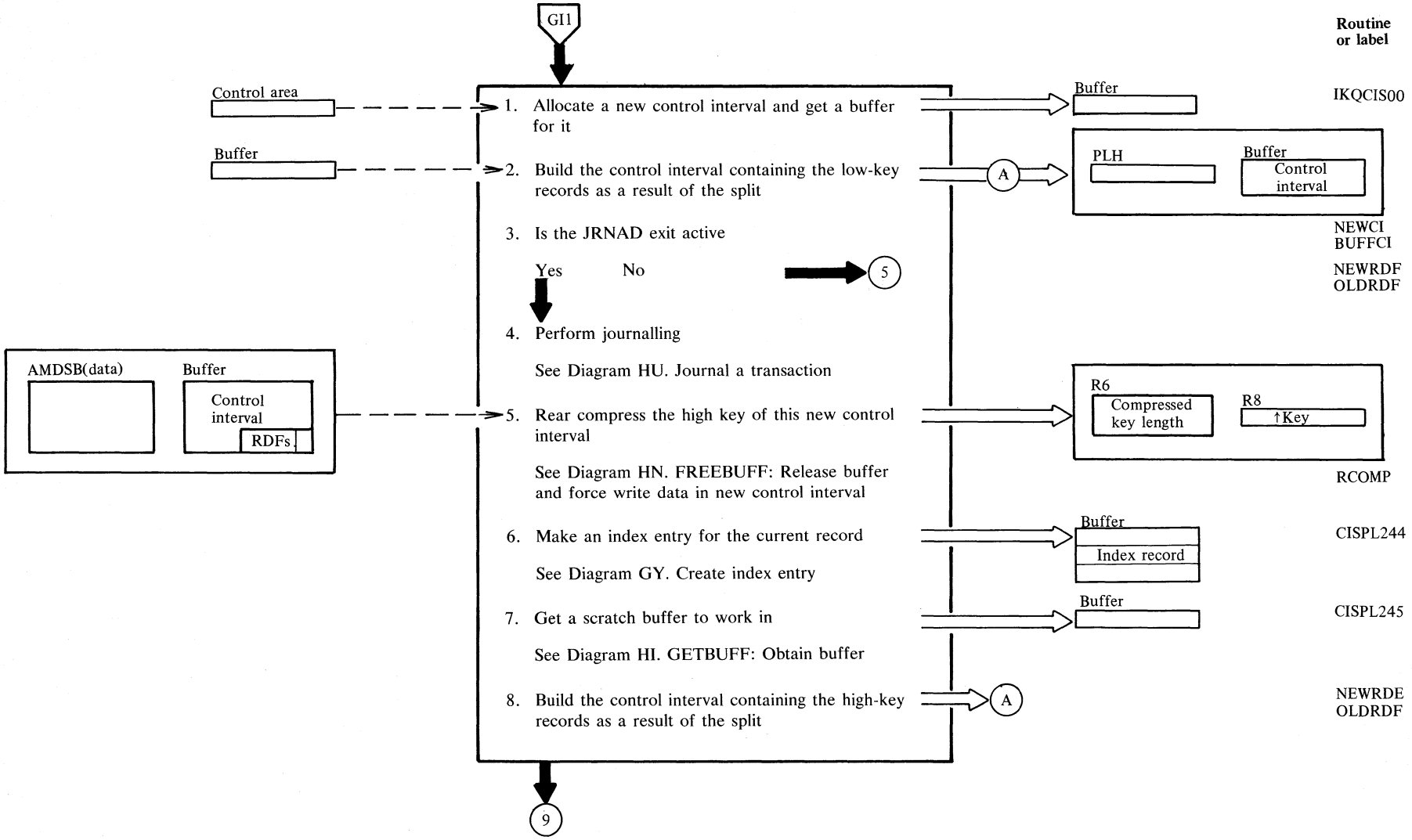
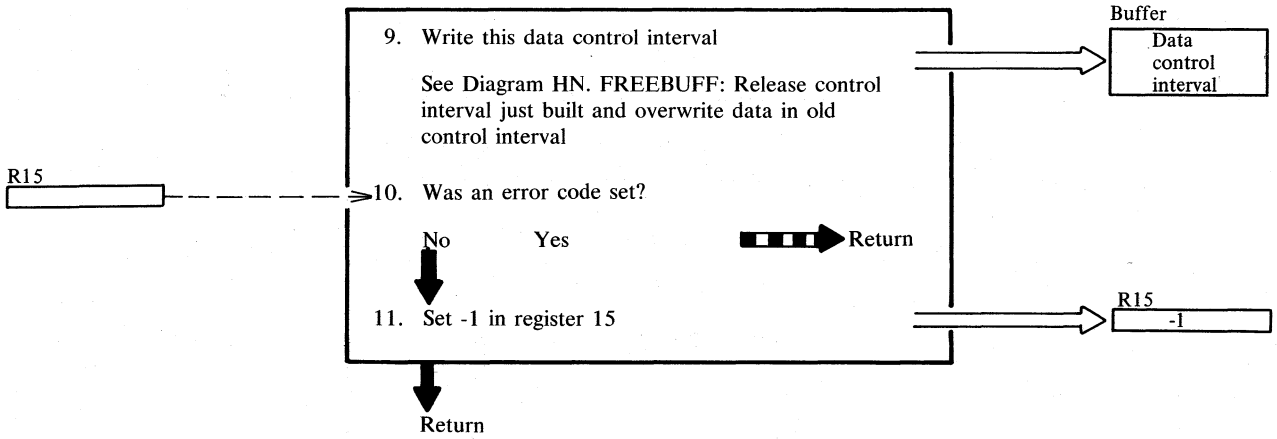


Diagram GT10. CISPL2: Control interval split



Routine or label

CISPL275

Notes for Diagram GT 10.

	Description	Module	Label
9.	Until this control interval is written, the low key records exist both in this control interval and in the new control interval written previously.		CISPL275
11.	The -1 code indicates to the caller (IKQMDY) that a split has taken place but no data has been inserted. IKQMDY must retry the insert or update operation that caused the original call to IKQCIS00.		

Diagram GT11. CINTRY: Entry-sequenced data set processing

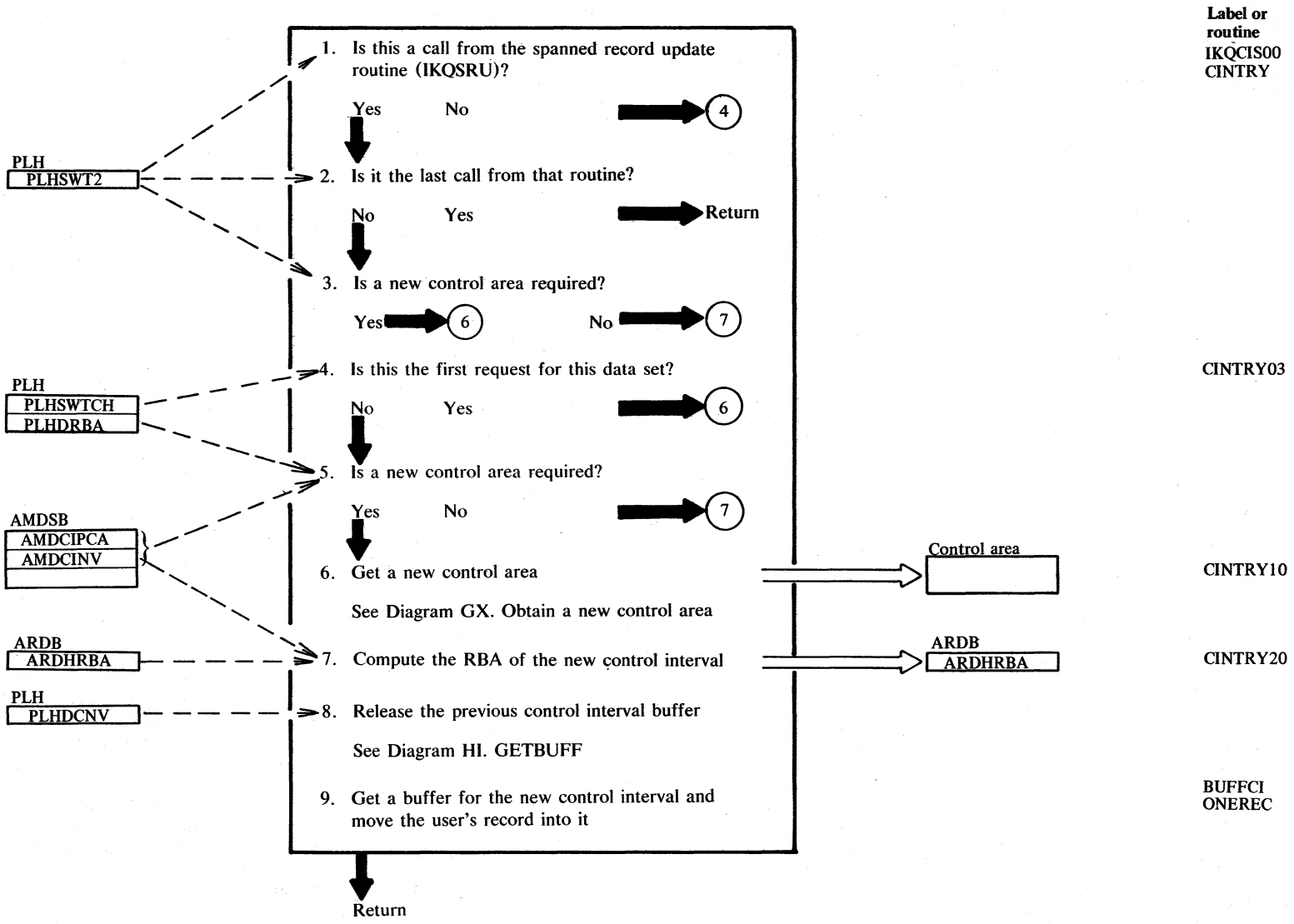


Diagram GU1. Control interval space reclamation

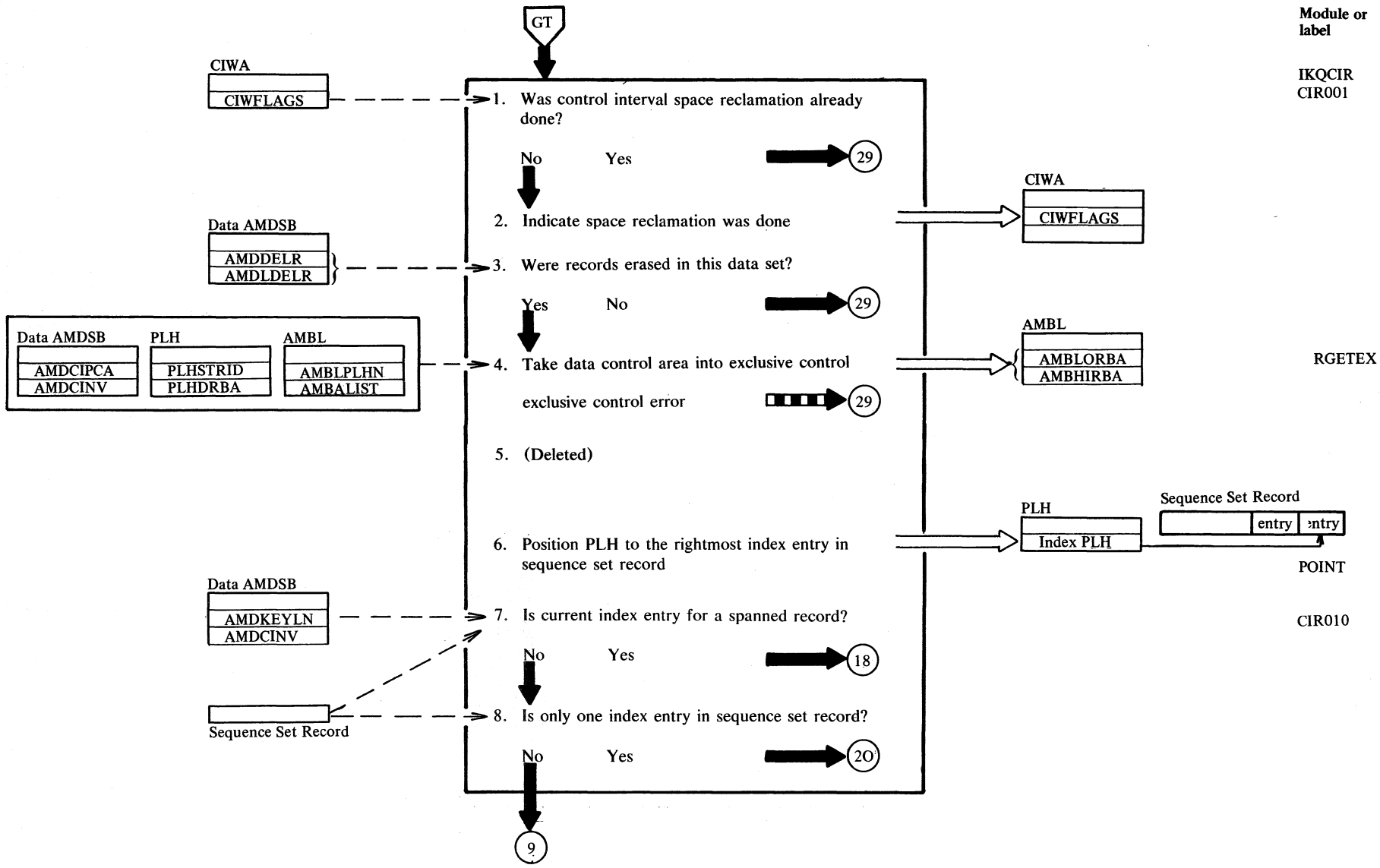


Diagram GU2. Control interval space reclamation

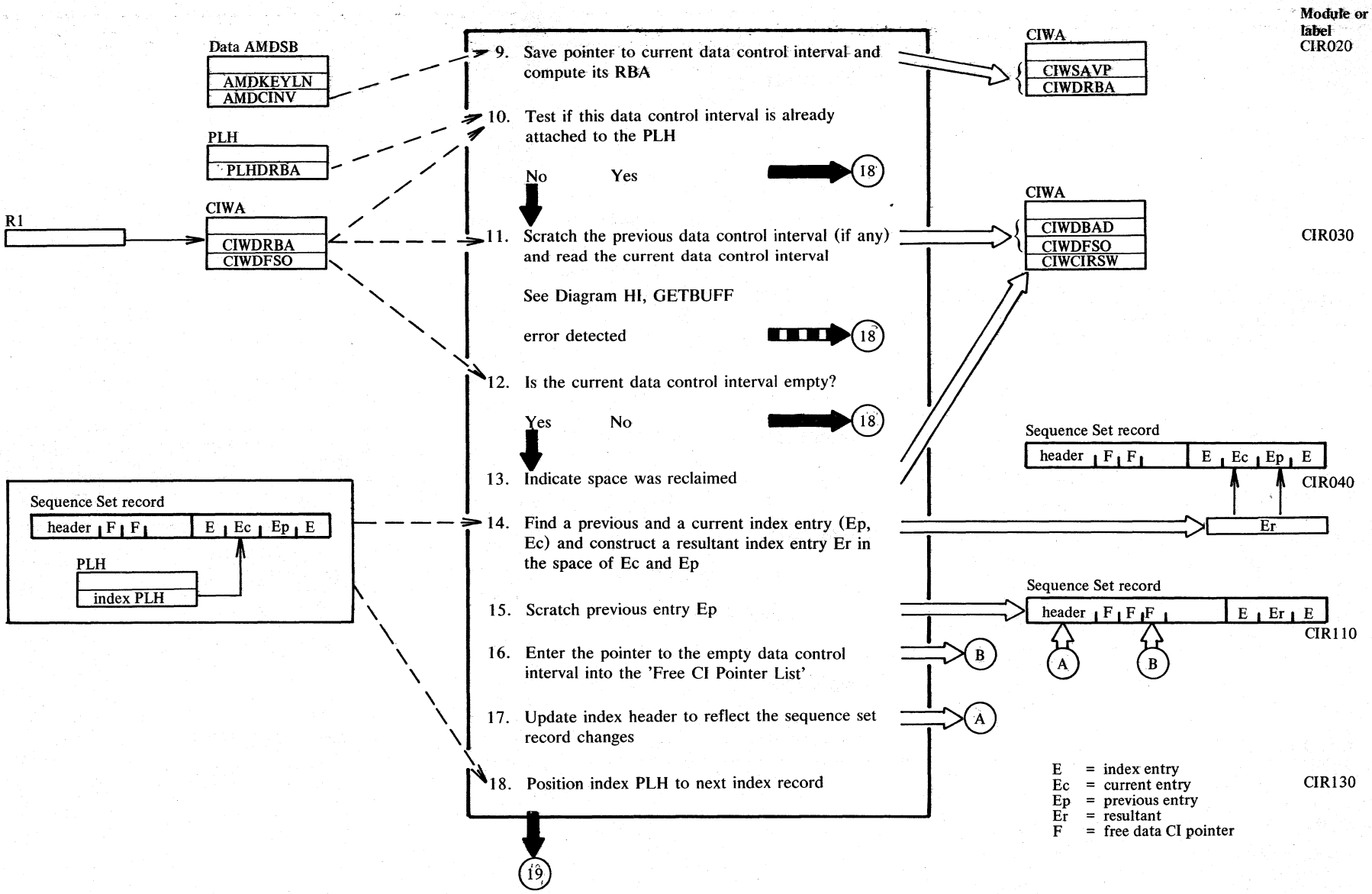


Diagram GU3. Control interval space reclamation

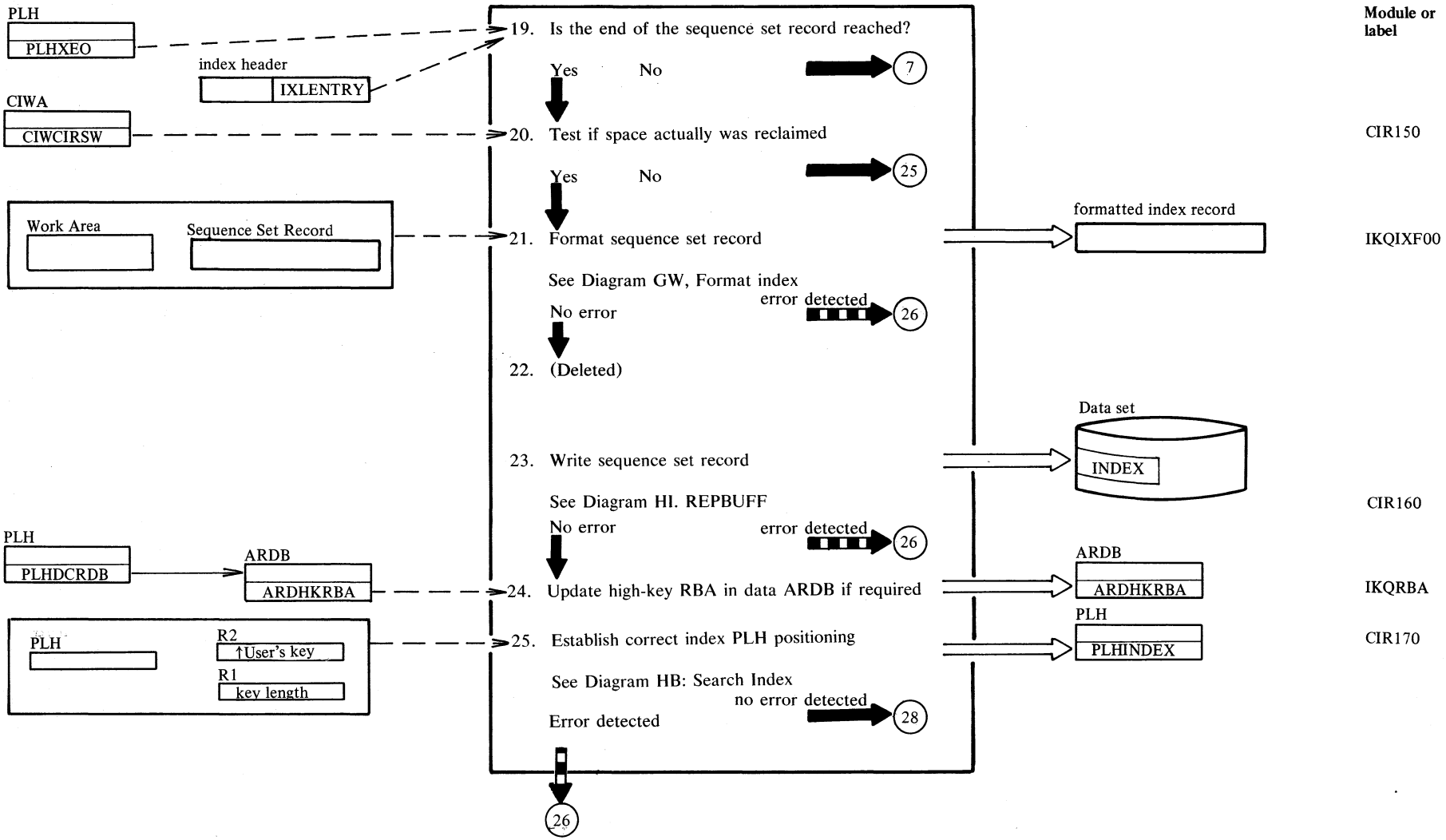
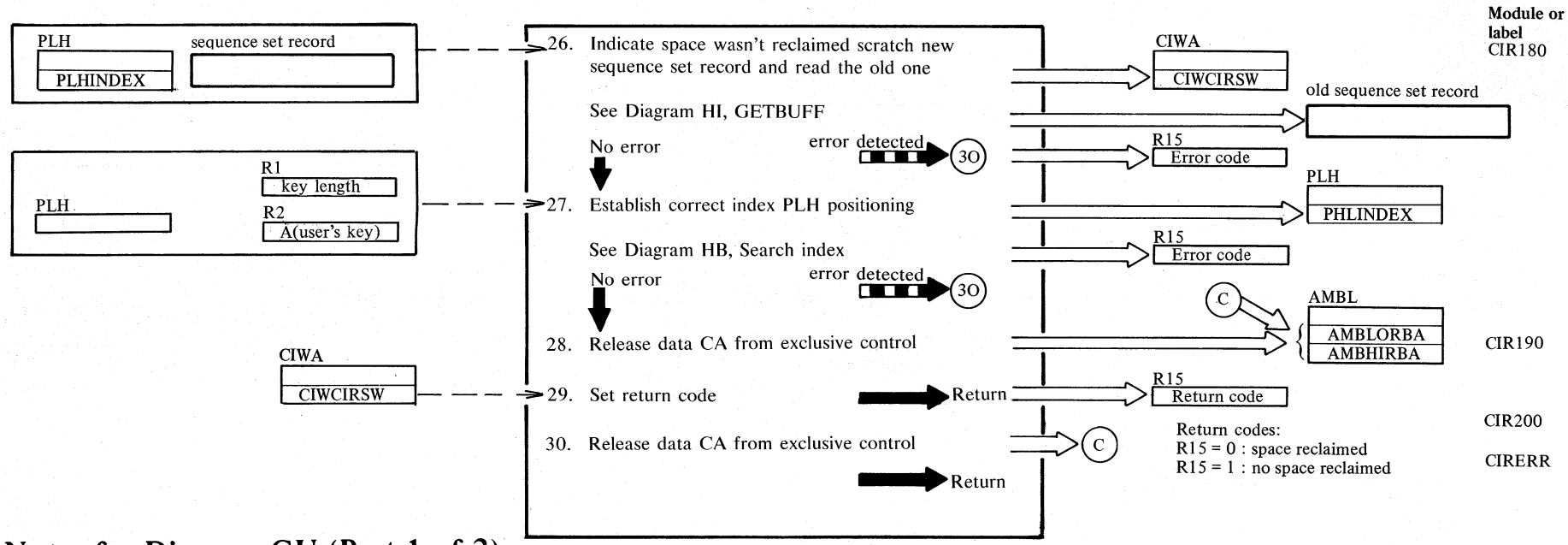


Diagram GU4. Control interval space reclamation



Notes for Diagram GU (Part 1 of 2)

IF IKQCIS00 determines that there are no CIs available for a split (IKQCAS00 is required), IKQCIR is first called. IKQCIR ensures that records have been deleted from the data set and then reads every CI within the appropriate CA, looking for all CIDFs that indicate that all records in the CI have been deleted. If such a CI is not found, then the CA must be split. If one or more deleted CIs are found within the CA, the index sequence set is adjusted to indicate that CIs are available. Control returns to IKQCIS00 to process the now-freed data CI.

Notes for Diagram GU (Part 2 of 2)

Description

14 The current index entry is the one pointing to the empty data CI, and the previous entry is the one to the right of this in the index record, except where the current entry is the first (rightmost) entry in the index record. In this case, the pointer from the next entry (to the left) is transferred to this entry and the next entry is used as the current entry, with the entry which points to the empty data CI now acting as the previous entry. This modification is necessary in order to allow the use of uniform processing for both cases.

The resulting index entry Er is formed from the current Ec and previous Ep entries in the following manner:

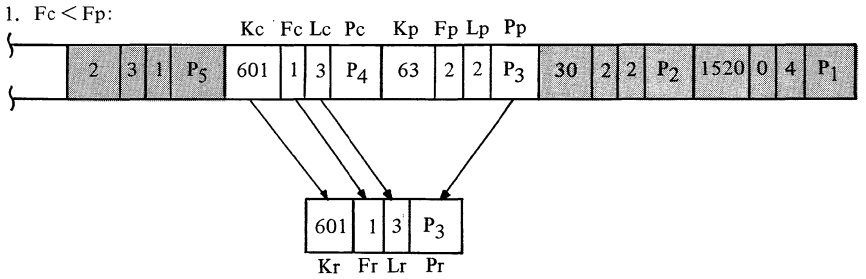
If the front compression count Fp of the previous entry is greater than, or equal to, the front compression count Fc of the current entry, the resultant entry consists of the key Kc, front compression count Fc, and key length Lc of the current entry, and the pointer Pp of the previous entry. (See examples 1 and 2.)

If the front compression count Fp of the previous entry is less than the front compression count Fc of the current entry, the resultant entry consists of:

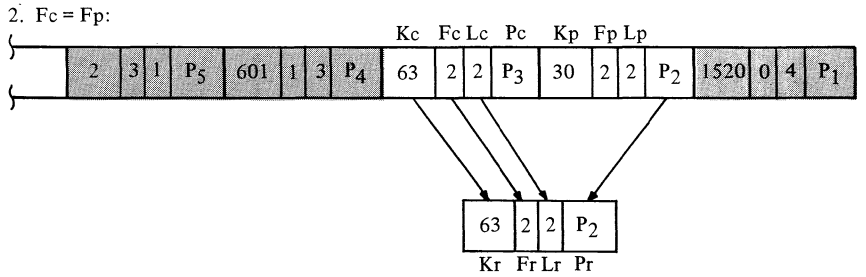
Kr: The first (Fc-Fp) characters of the previous key Kp, followed by the current key Kc.
Fr: The front compression count Fp of the previous entry.
Lr: The key length Lc of the current entry plus the difference between the front compression counts (Fc-Fp).
Pr: The pointer Pp from the previous entry.
This is also shown in Example 3.

Examples:

1. Fc < Fp:



2. Fc = Fp:



3. Fc > Fp:

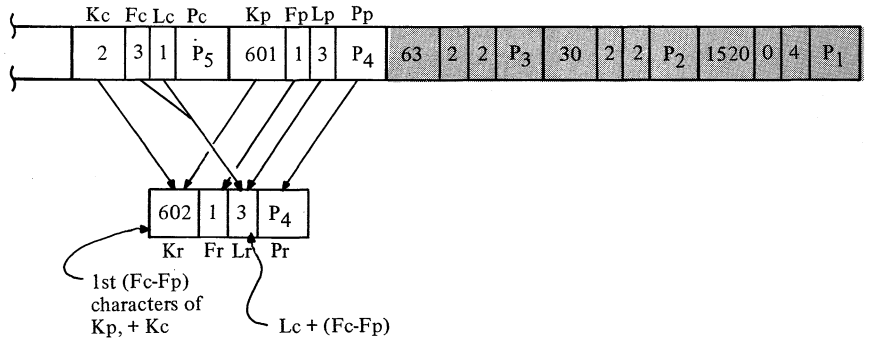


Diagram GV1. Preformat relative record data set

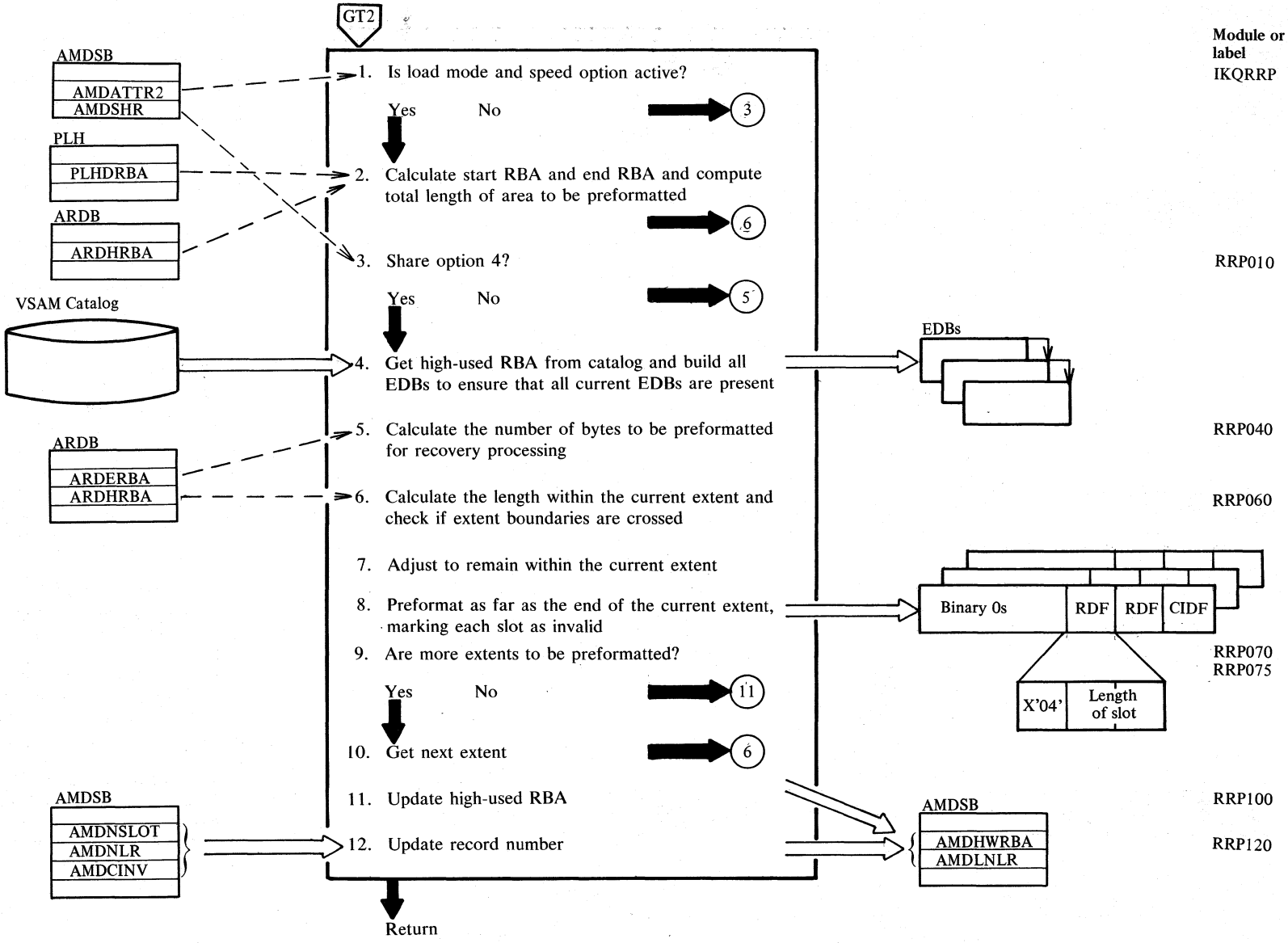
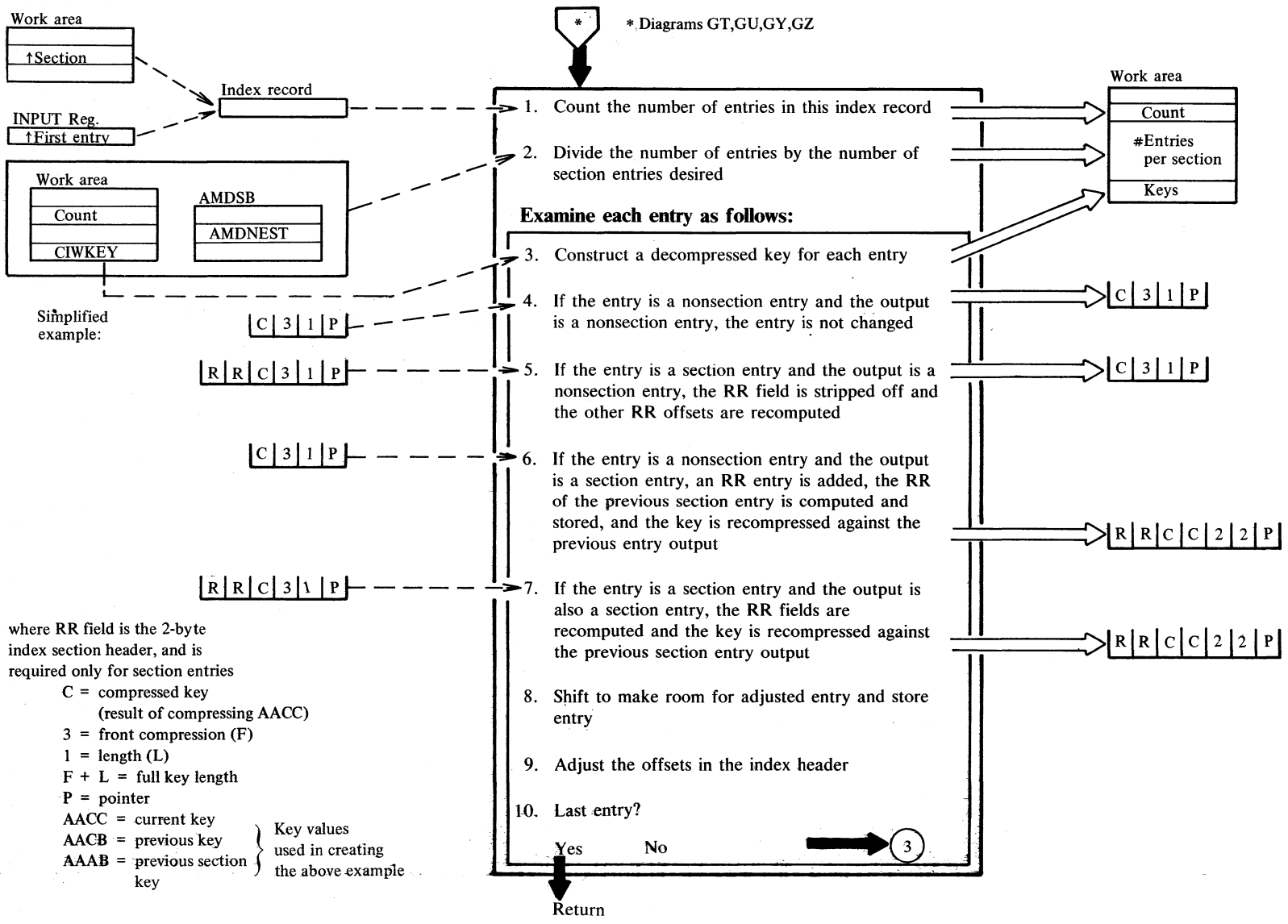


Diagram GW1. Format index



Module

IKQIXF00

IKQSFT

Diagram GX1. Obtain new control area

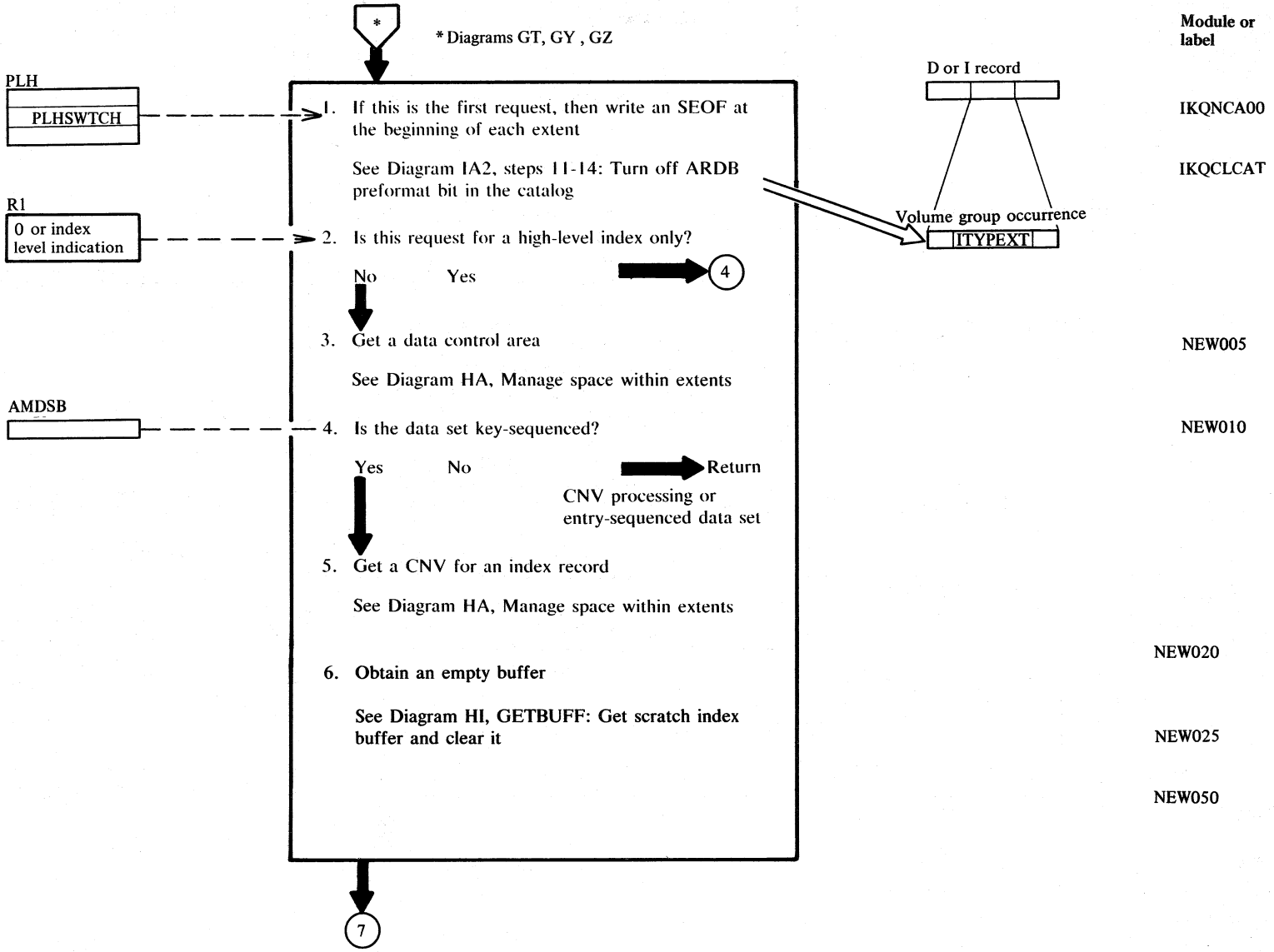


Diagram GX2. Obtain new control area

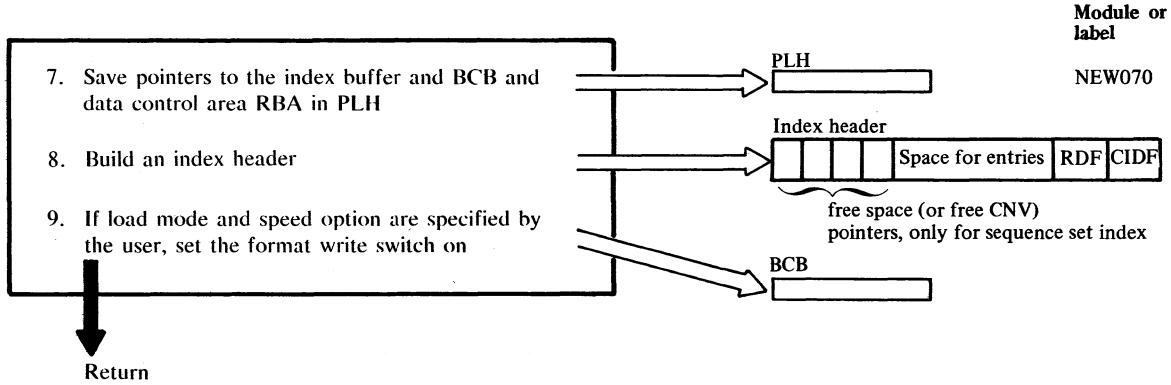


Diagram GY1. Create index entry

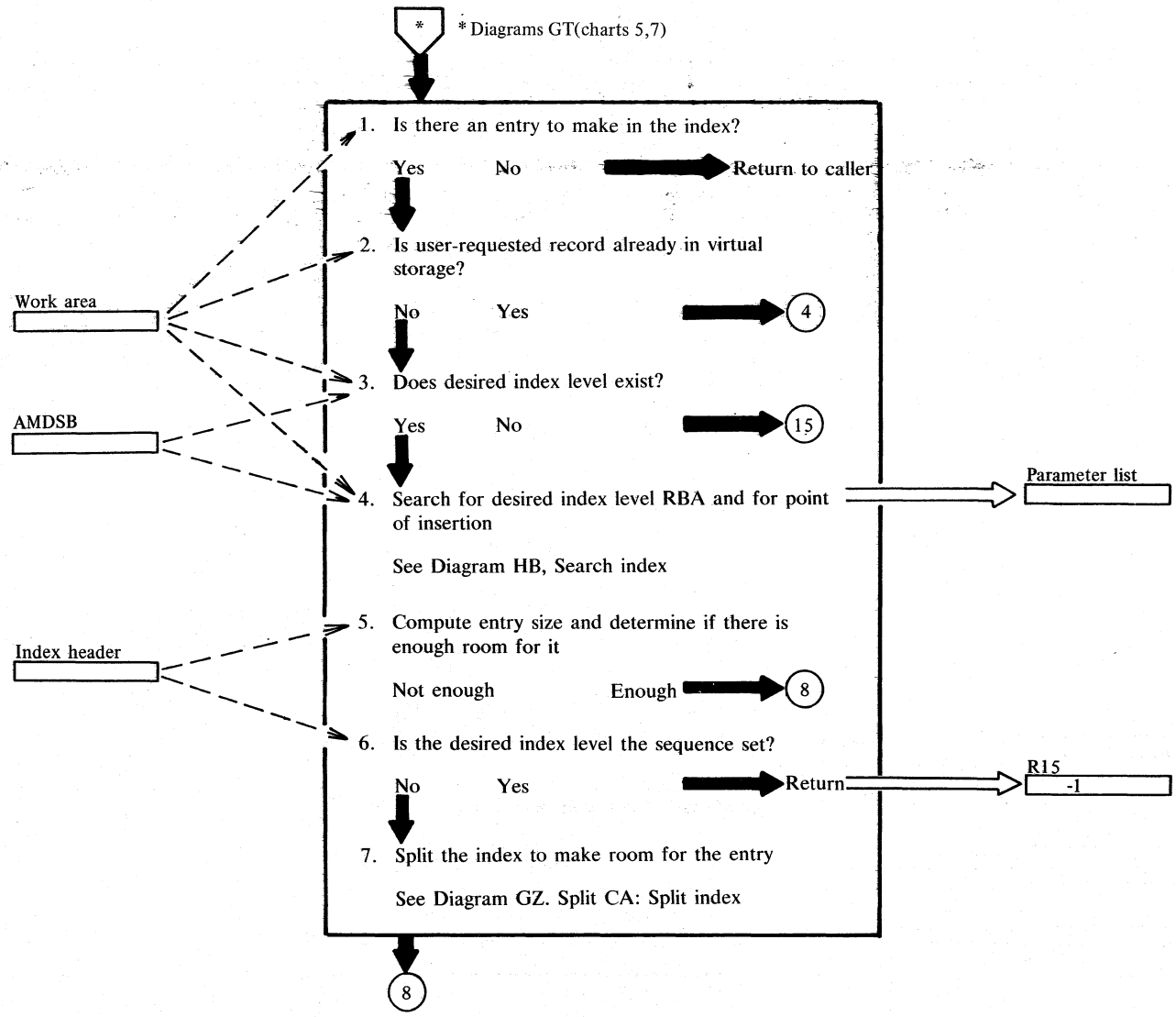


Diagram GY2. Create index entry

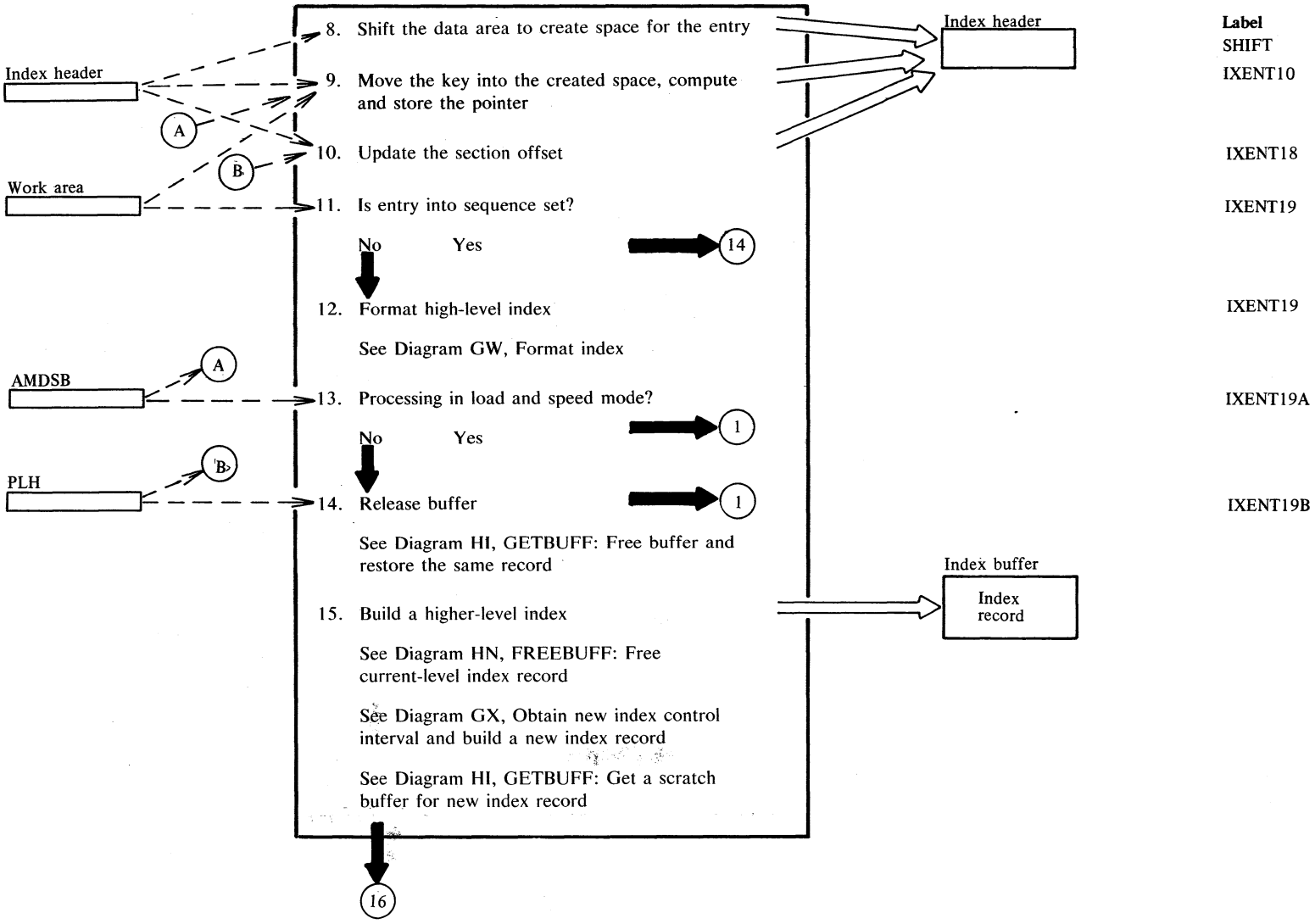
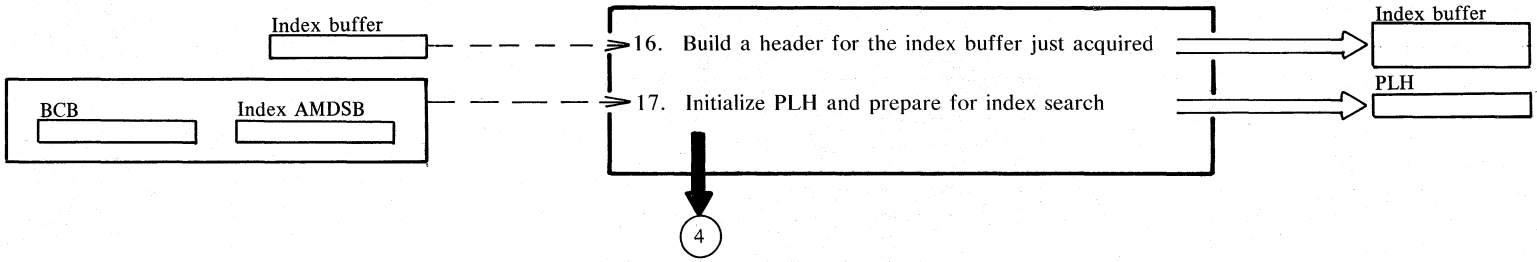
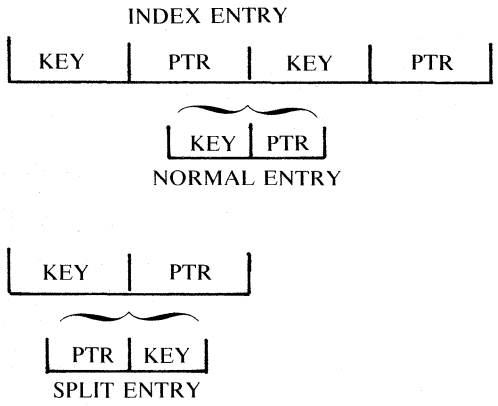


Diagram GY3. Create index entry



Notes for Diagram GY1 - GY3

	Description	Module
8-14	There are two types of index entries, normal and split (see examples below). The only difference between a normal entry and a split entry is in the point of insertion. An index entry is composed of a key and pointer. A normal entry is inserted after the pointer; a split entry, however, is inserted after the key and the new entry is composed of a pointer and key. The result is two entries composed of the old key and new pointer, and the new key and old pointer.	IKQIXE00



Notes for Diagram GY4

Description

The index entry for a spanned record is a special case of the split entry mentioned in the notes for Diagrams GY1 - GY3. When a spanned record is stored, the first index entry is created by the normal method (GY1 - GY3). Index entries for segments after the first one are created by the routine shown in Diagram GY4.

The index entry for the first segment originally holds the key and its F byte contains the actual key compression count. Index entries for further segments are inserted in the middle of the first entry, thus causing the key to be moved to the left and to remain in the entry for the last segment of the spanned record.

The entries for the second and subsequent segments do not contain a key, and their F byte contains a compression count equal to the keylength, thus indicating a keylength (in the entry) of zero.

Diagram GY4. Create index entry for spanned records

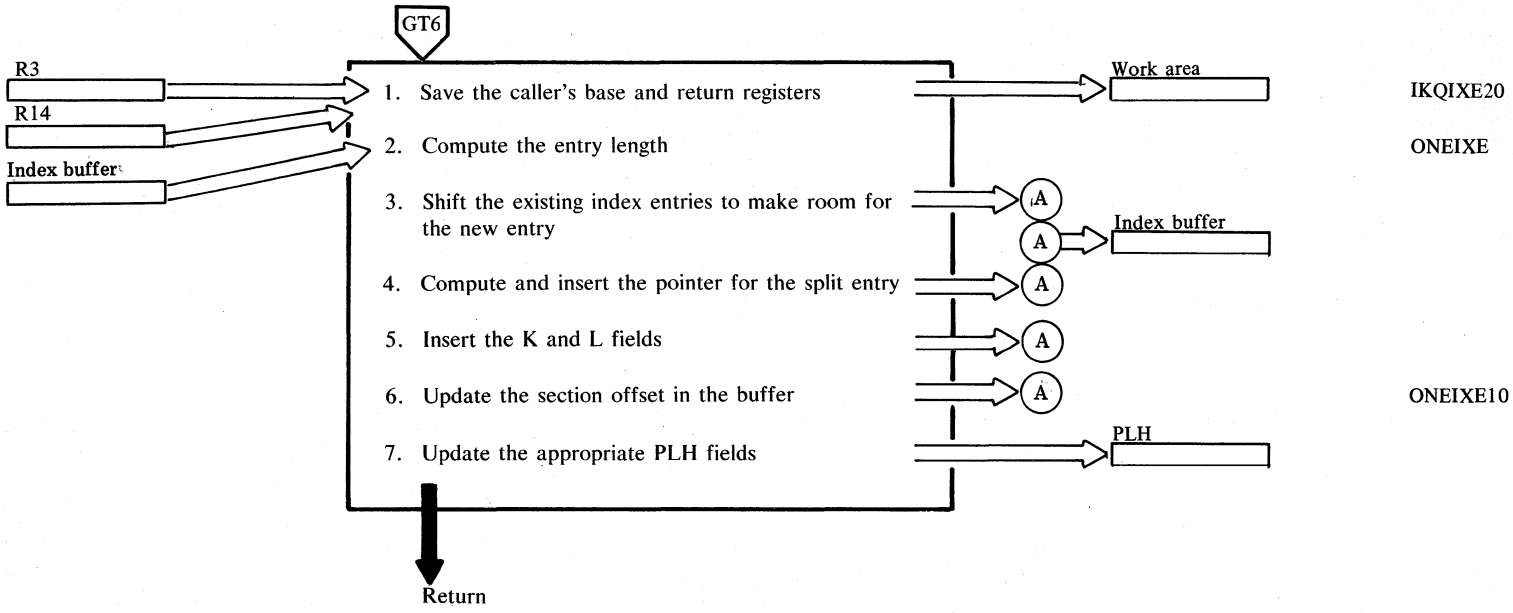


Diagram GZ1. Split control area

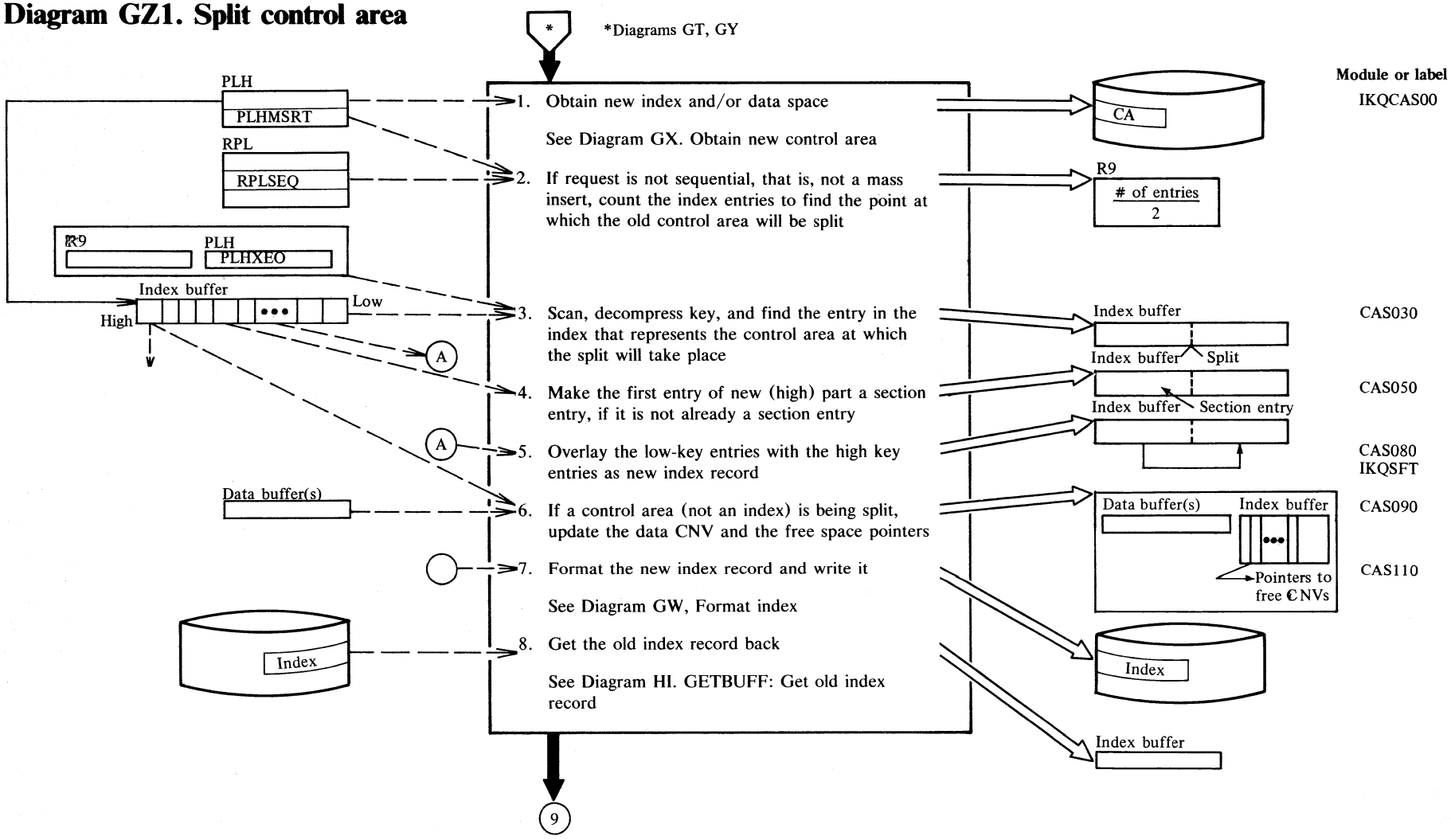


Diagram GZ2. Split control area

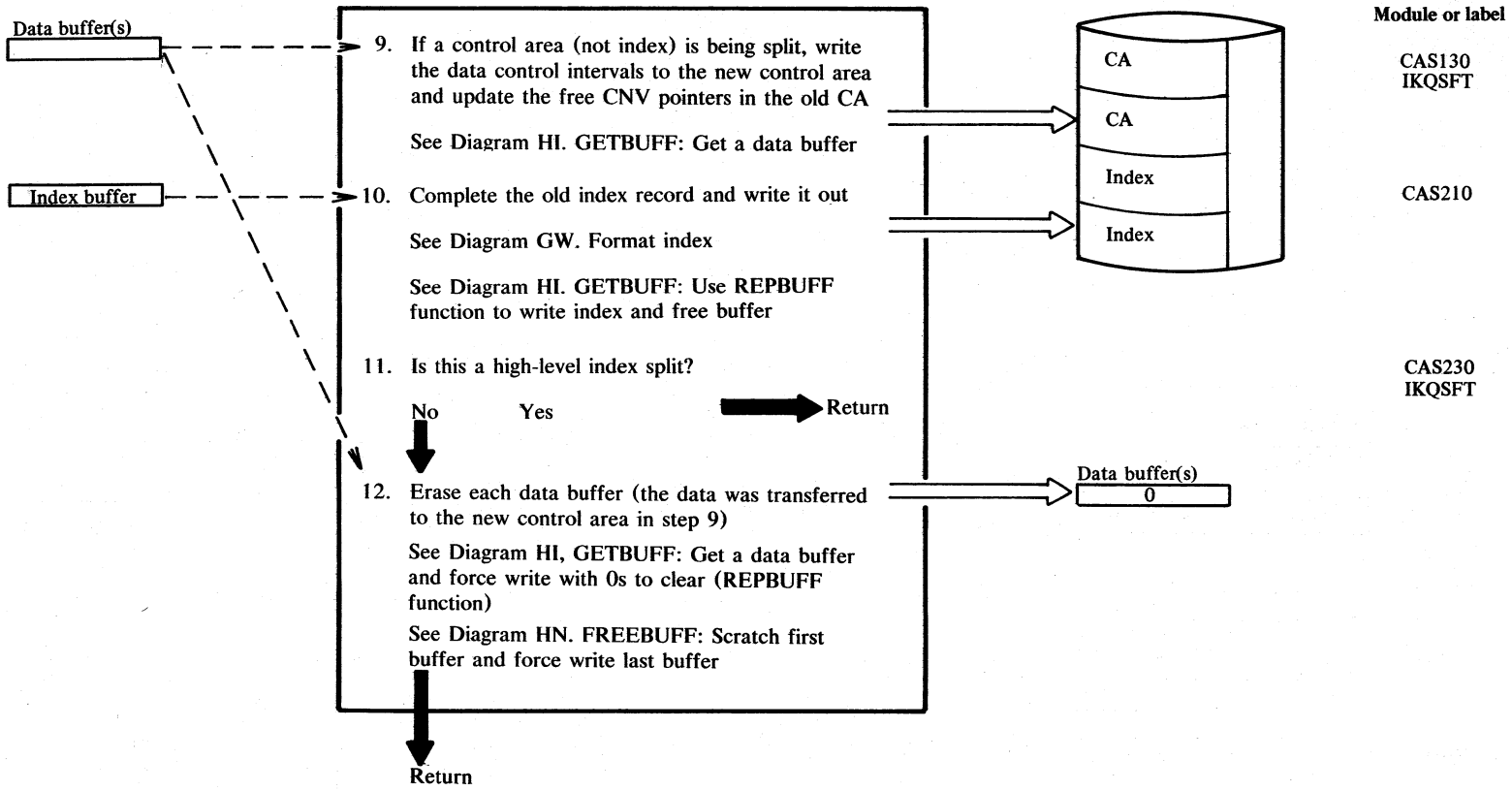
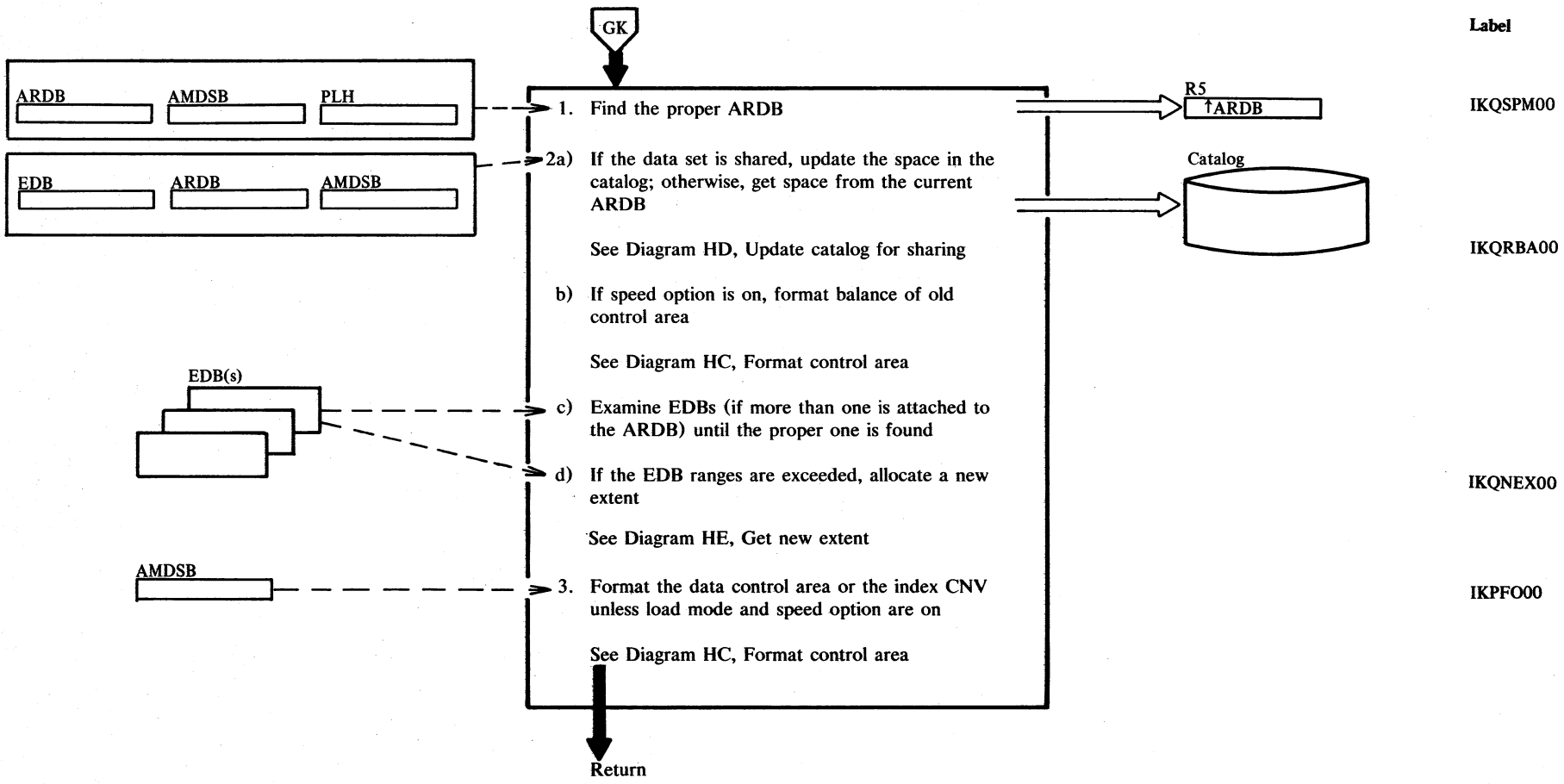


Diagram HA1. Manage space within extents



Notes for Diagram HA

Description

1 The logic of finding the proper ARDB is determined by using the following decision table where certain conditions result in certain operations taking place. For example, looking at the rightmost column, if there is an index, a key range and a change of key range, and the sequence set is embedded with data, then the ARDB is pointed to from the PLH and the ARDPREL, a field in the data ARDB, points to the associated index ARDB.

Data or index	D	D	D	I	I	I	I	I	I	I
Key range	N	Y	Y	N	N	N	Y	Y	Y	Y
Key range change	–	Y	N	–	–	–	–	–	N	Y
Sequence set or high level	–	–	–	–	HL	SS	–	HL	SS	SS
Sequence set with data	–	–	–	N	Y	Y	N	Y	Y	Y
The ARDB is found as follows:										
ARDB from AMDSB	X		X	X		X	X		X	
ARDB from PLH		X								X
Search ARDBs for key range			X			X			X	
Use ARDPREL*										X
Search ARDBs for high-level					X			X		

* ARDPREL is a field in the data ARDB that points to the associated index ARDB

Legend:
N = no
Y = yes
D = data
I = index
– = don't care
X = indicates how the proper ARDB is found
SS = sequence set
HL = high level

Description

2 'Space' in this context means the serial apportionment of a single control interval (for an index) or control area (for data) for immediate use. This space is a subdivision of an extent as defined by DADSM allocation.

Diagram HB1. Search index

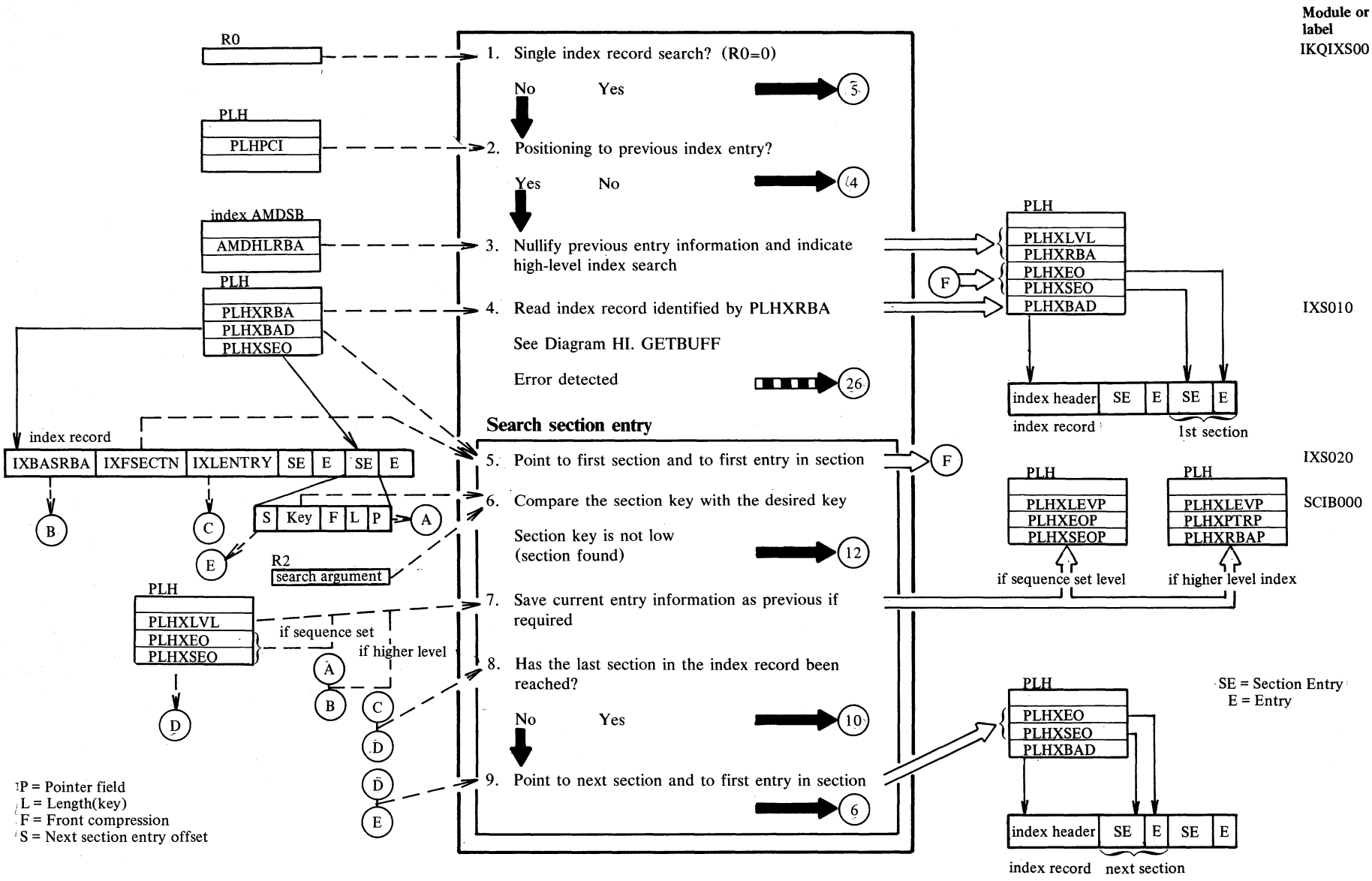


Diagram HB2. Search index

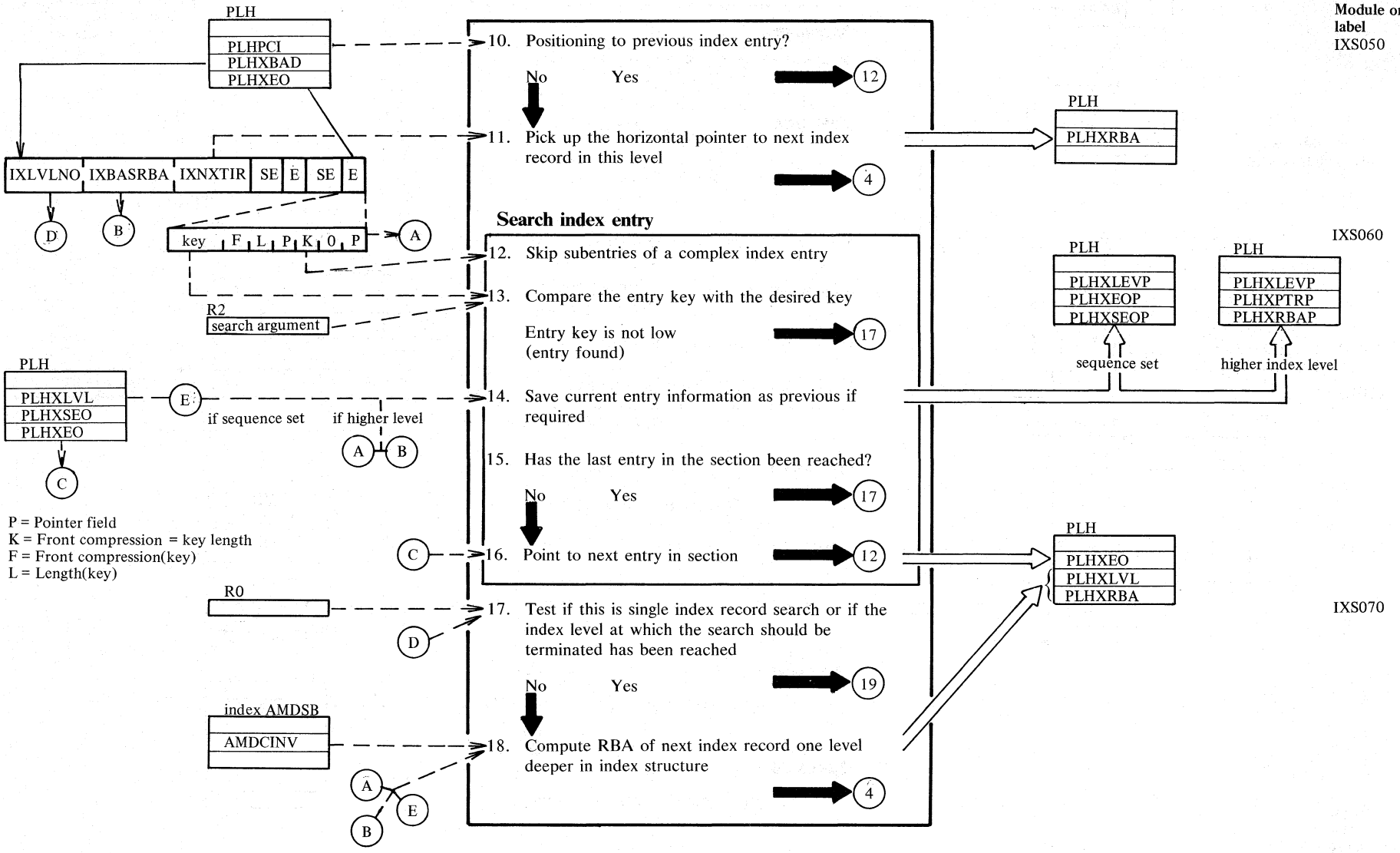
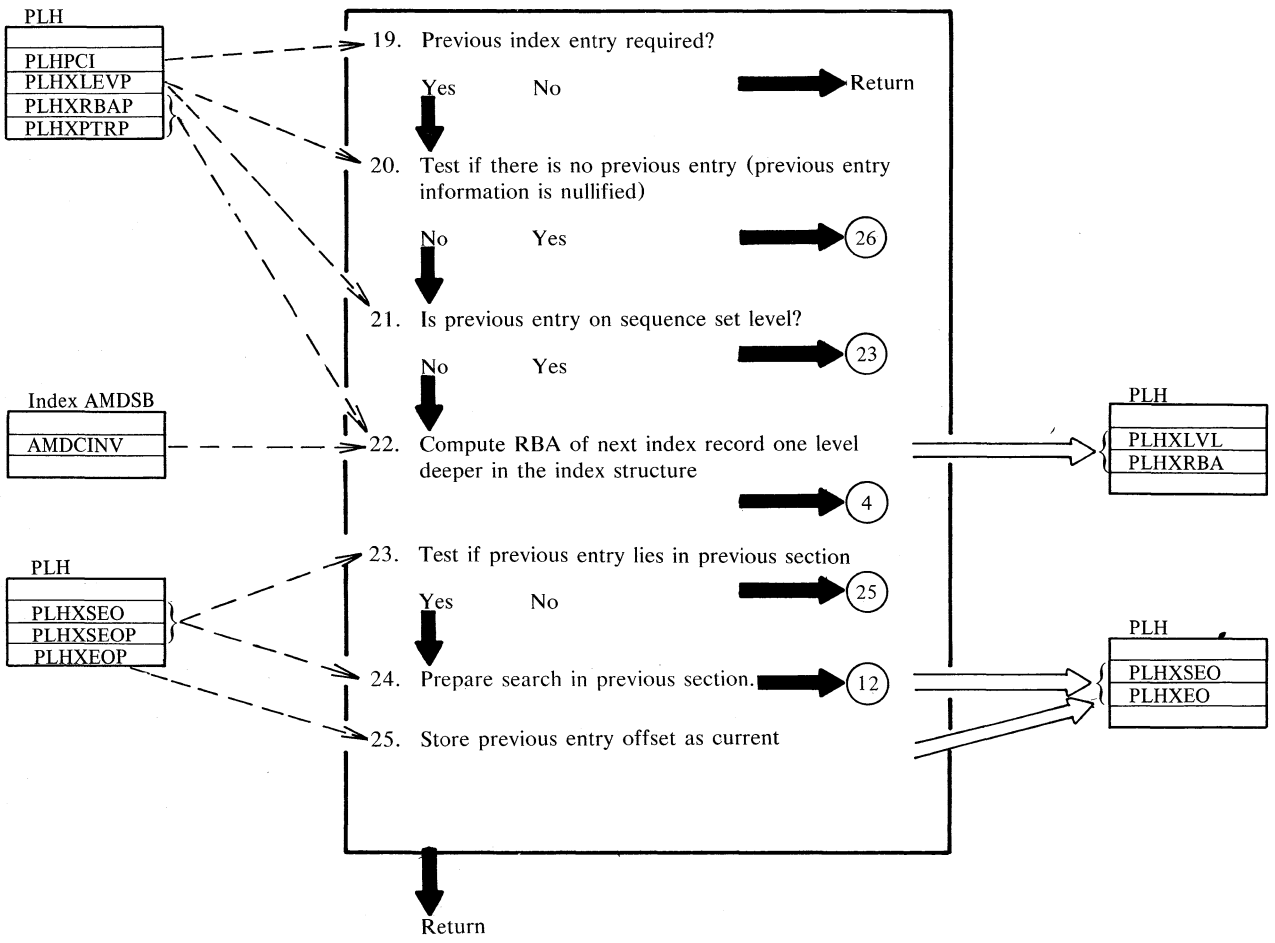


Diagram HB3. Search index



Module or label

IXS080

IXS085

IXS086

IXS100

Notes for Diagram HB

Description

Normal index search:
An index record is searched for the entry containing a key which is not low compared with the search argument. The search starts with the lowkey entry and ends when the required entry is found. A search is done in two steps: first the section containing the desired entry is located (step 5 - 9), then the entry within the section is identified (steps 12 - 16).

If R0=0 only the index record attached to the PLH (PLHXBAD) is searched. No I/O is done,

If R0>0 its value identifies the index level to stop on, and PLHXRBA contains the RBA of the index record to start with. The 'desired index entry' in high-level index records contains a relative pointer to the index record one level deeper which must be searched next. If R0=1 the index hierachy is searched down to the sequence set level, which is the deepest index level. The 'desired entry' in sequence set contains a relative pointer to the data control interval which contains (if inserted) the data record whith the desired key.

The high-key entry is identified by a key with length 0.

Output of index search is the PLH positioned to the 'desired index entry' on the desired level. If the 'desired entry' is complex, (i.e. for a spanned record) PLH is positioned to the leftmost subentry.

Description

Index search for the previous index entry
During sequential backward processing, whenever the low-key entry in a sequence set record was reached the previous index entry in the next (low-key direction) sequence set record must be located.

This is achieved by a normal top-down index search for the previous request key during which previous entry information is saved whenever the inspected index entry is not the 'desired entry' (steps 7, 14).

Normally the saved previous index entry at the end of the normal top-down index search for the previous request key will lie in a higher level index; so a secondary index search is started (steps 21, 22) with the index record identified by the previous entry information. The secondary search locates the high-key index entry in the next lower sequence set record, which is the previous key.

The end of data set (in backward direction) is reached when no previous entry information was stored during the normal index search for the previous request key.

Diagram HC1. Format data CA or index CNV

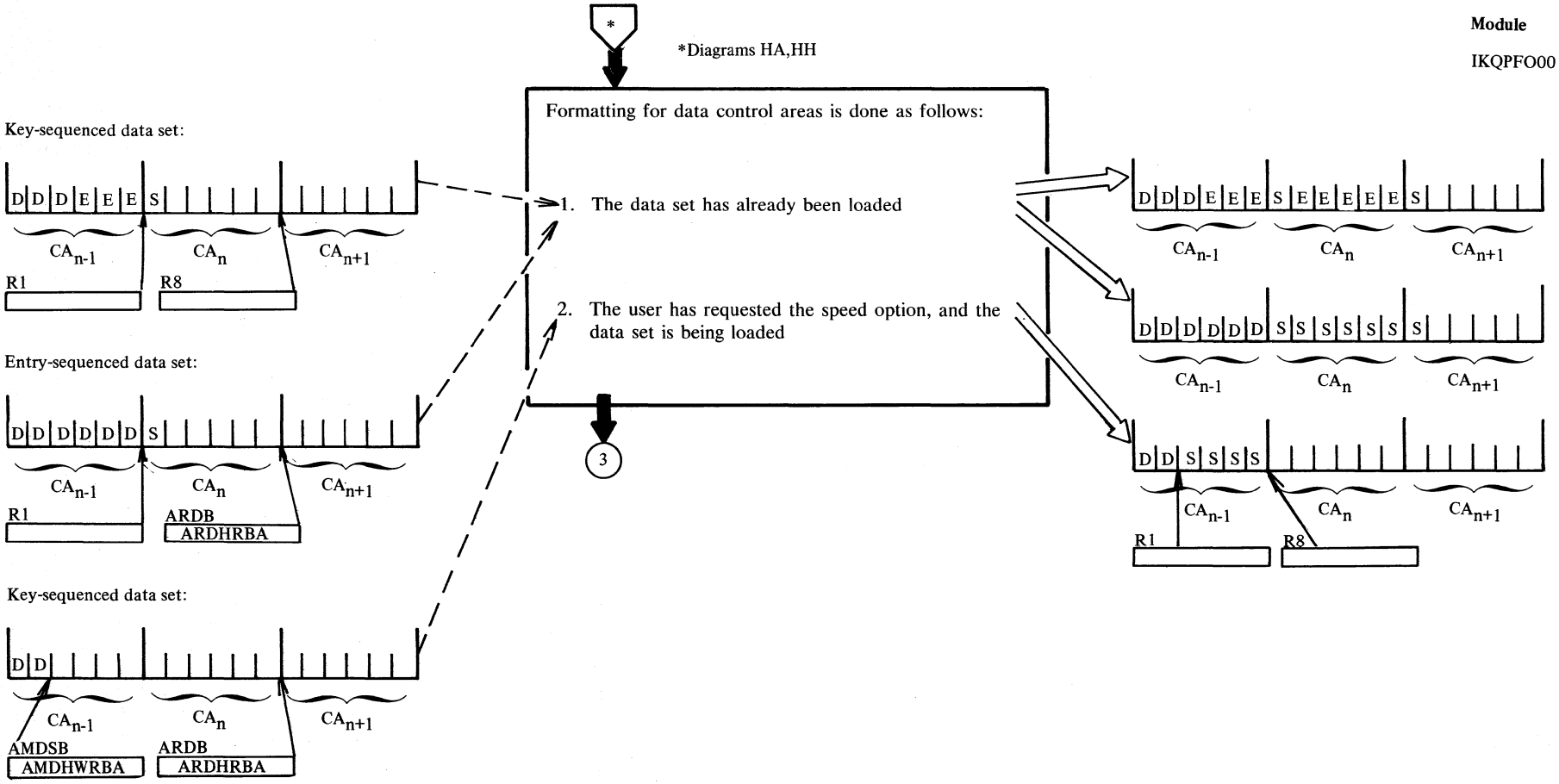
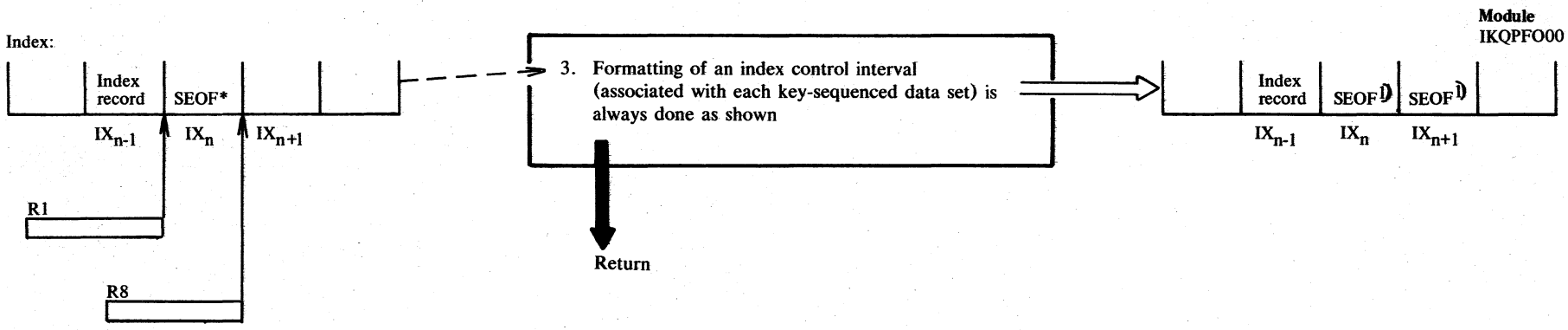


Diagram HC2. Format data CA or index CNV



*Does not exist for first control interval, that is, is not written.

1) Both SEOFs (at n and n + 1) are written, even though redundant.

where:

D = data control interval
E = empty control interval (a CNV filled with 0 plus CIDF)
S = software end-of-file (a CNV filled with binary 0s)
CA = control area
CNV = control interval
IX = index CNV (for index record)
n = number

Note: The three steps in this diagram are not sequential. They simply show 'before' and 'after' examples of the various types of preformatting.

Notes for Diagram HC

	Description	Module
1-3	IKQBFA00 and IKQSFT are called to help process each of the operations described.	IKQPFO00
†	The speed option can be specified by the user, but load mode will automatically be invoked by VSAM if the data set is empty or if the end of a key range or data set has been reached.	
2	Once loading has been completed, that is, one or more records have been loaded into the data set and the data set has been closed, load mode may still be invoked automatically, but the speed option will be ignored.	
1 and 3	IKQRBA is called to turn off the ARDB preformat bit in the catalog after any extend operation.	

Diagram HD1. Update catalog

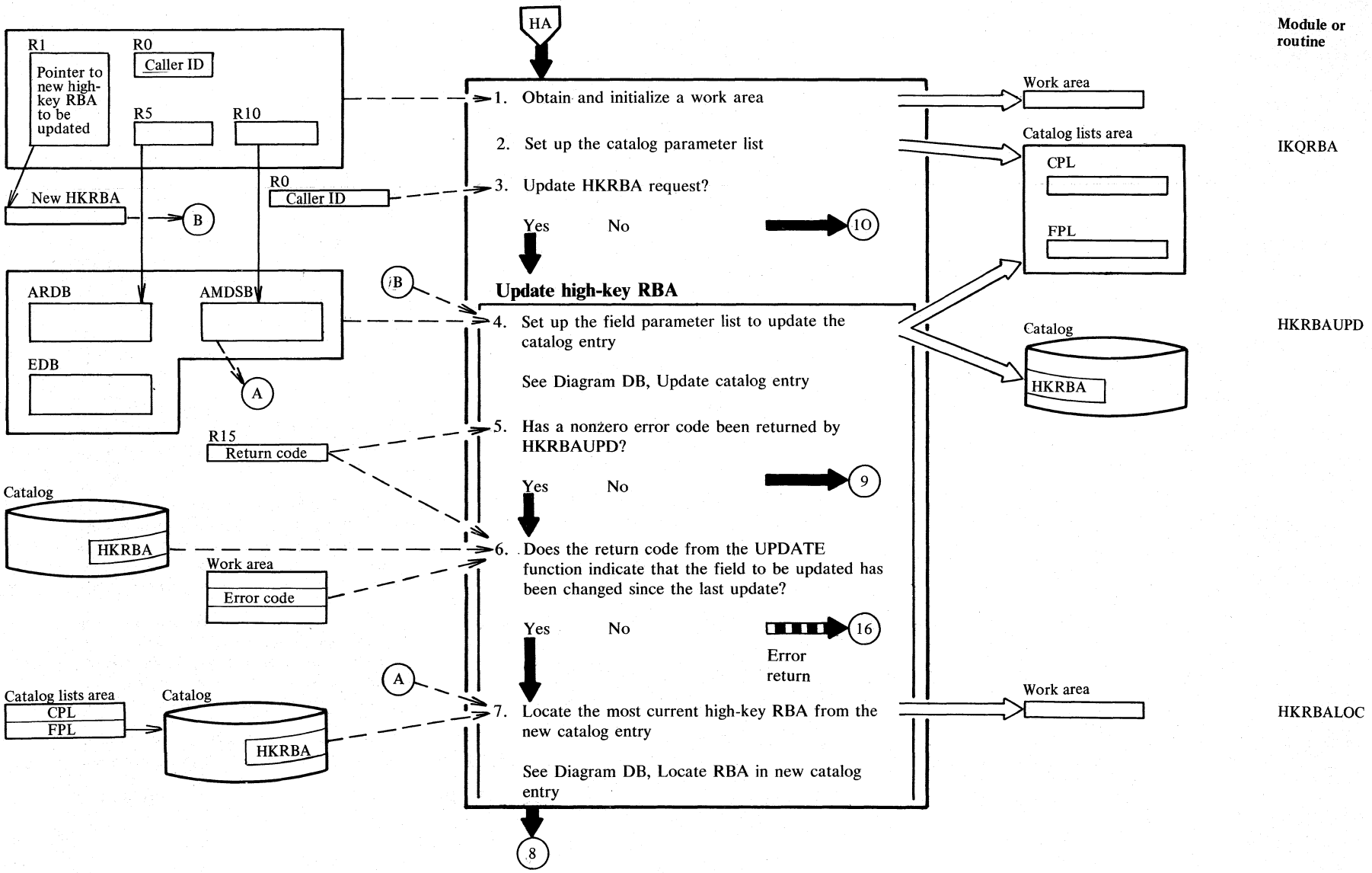


Diagram HD2. Update catalog

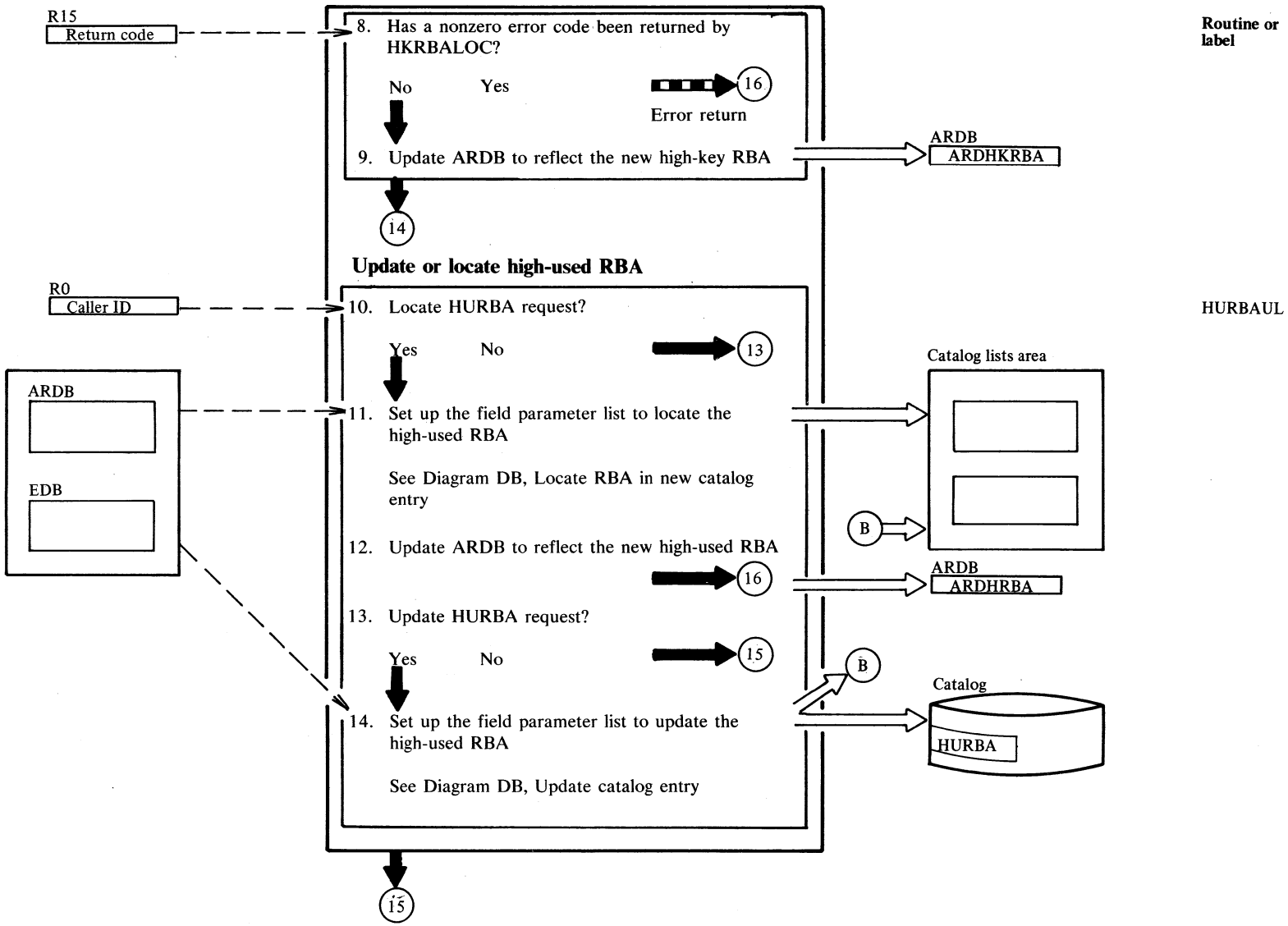


Diagram HD3. Update catalog

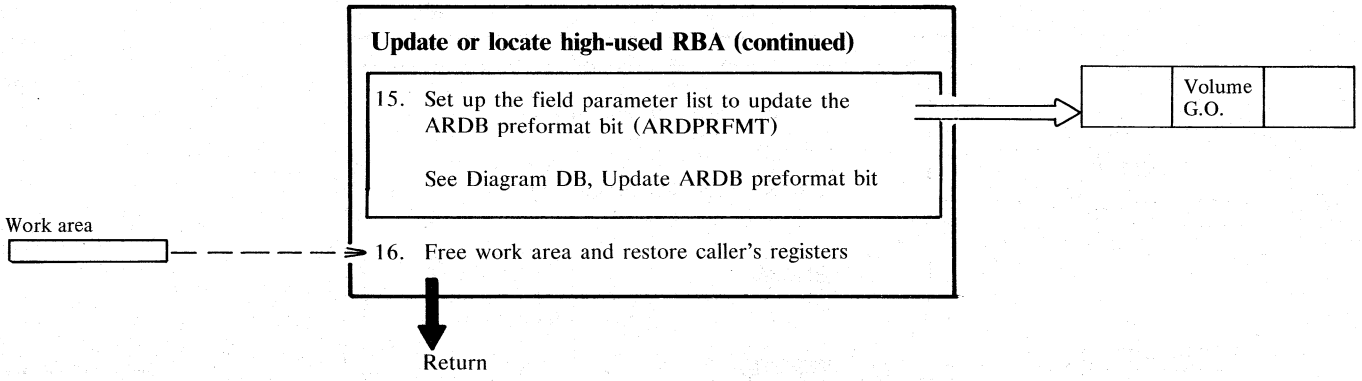


Diagram HE1. Get new extent

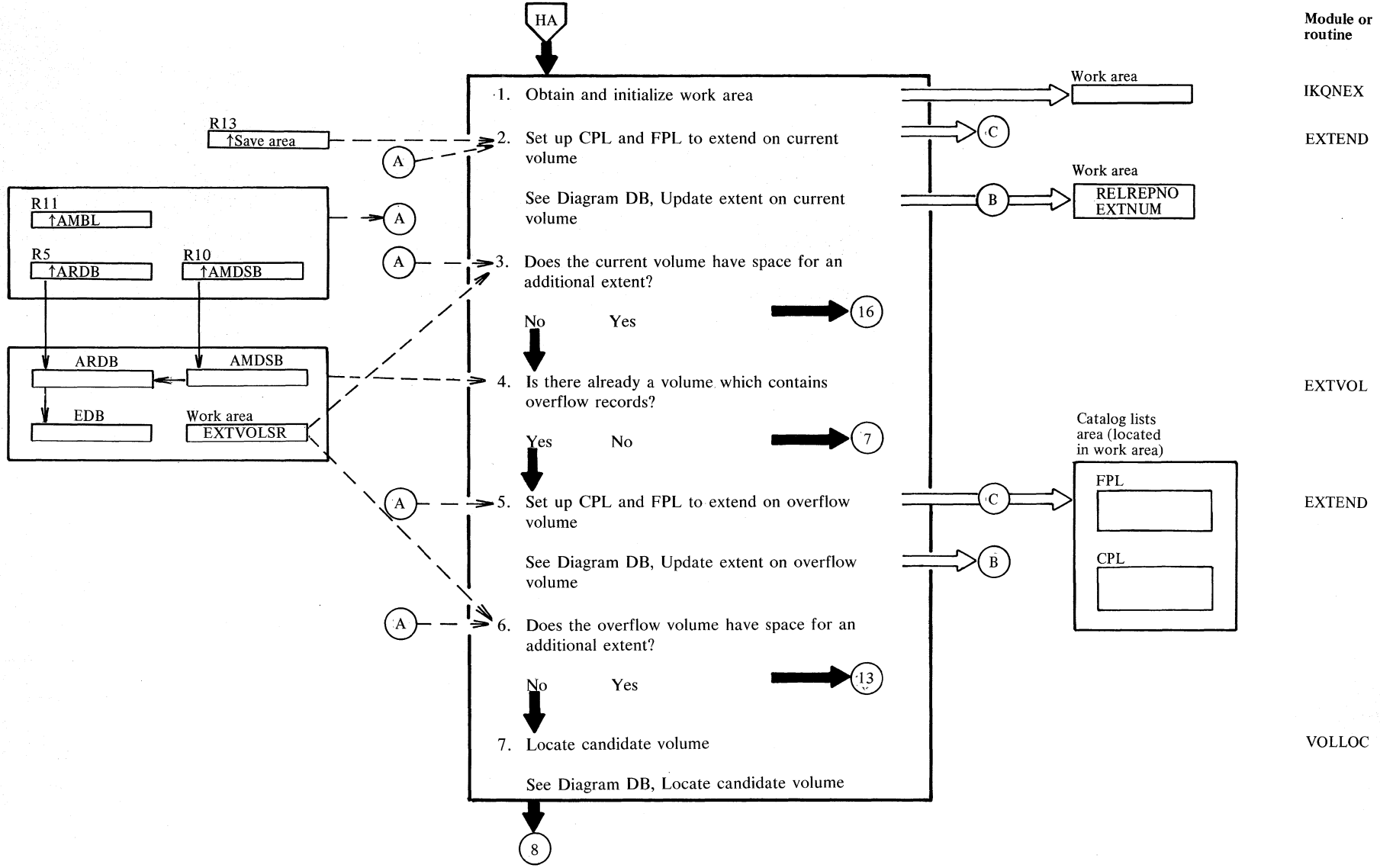


Diagram HE2. Get new extent

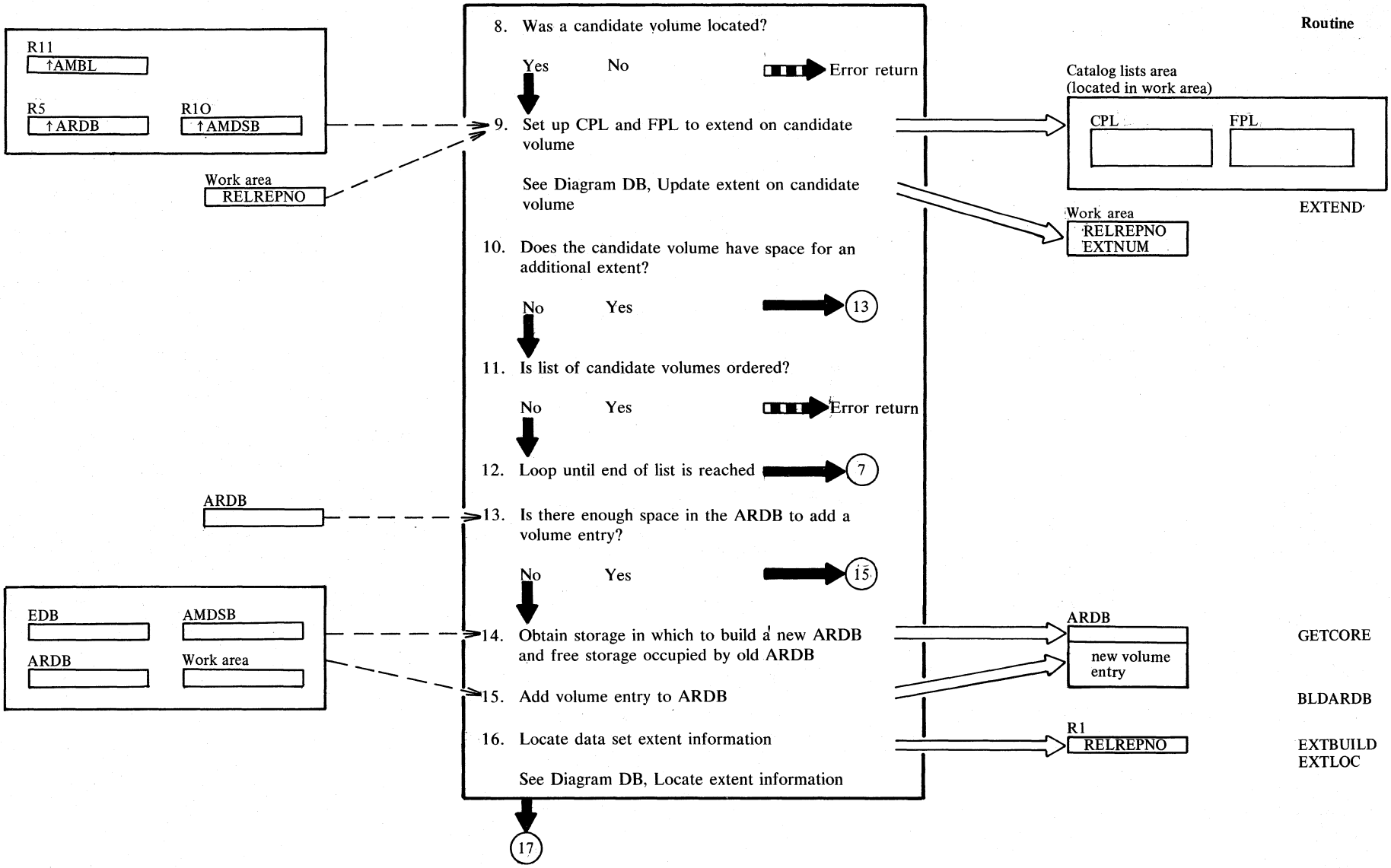
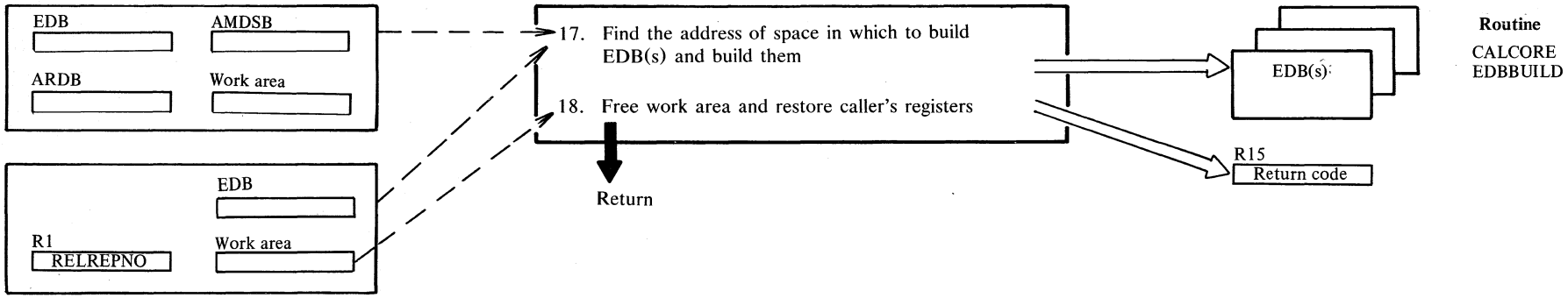


Diagram HE3. Get new extent



Notes for Diagram HE

Description	Module
Note: This module falls into four logical groupings: • Steps 1-2 where an attempt is made to extend on the current volume • Steps 3-5 where an attempt is made to extend on an overflow volume, if the current volume did not have enough space for an additional extent • Steps 6-11 where an attempt is made to extend on a candidate volume, if the overflow volume did not have enough space for an additional extent • Steps 12-16 where the data set extent information is located on the proper volume, a volume entry is made to the ARDB, and EDB(s) are built.	IKQNEX

Diagram HF1. Error handler

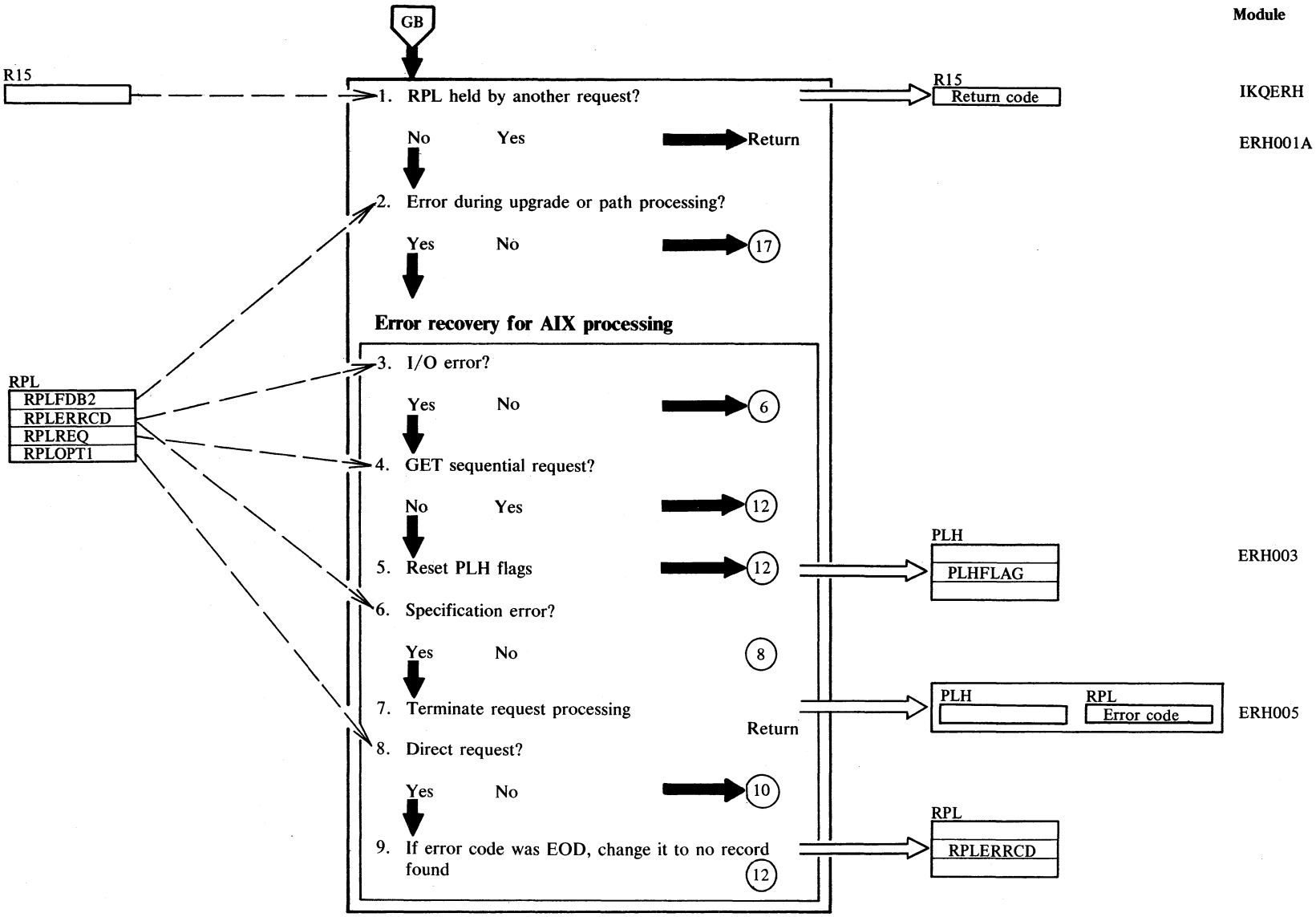


Diagram HF2. Error handler

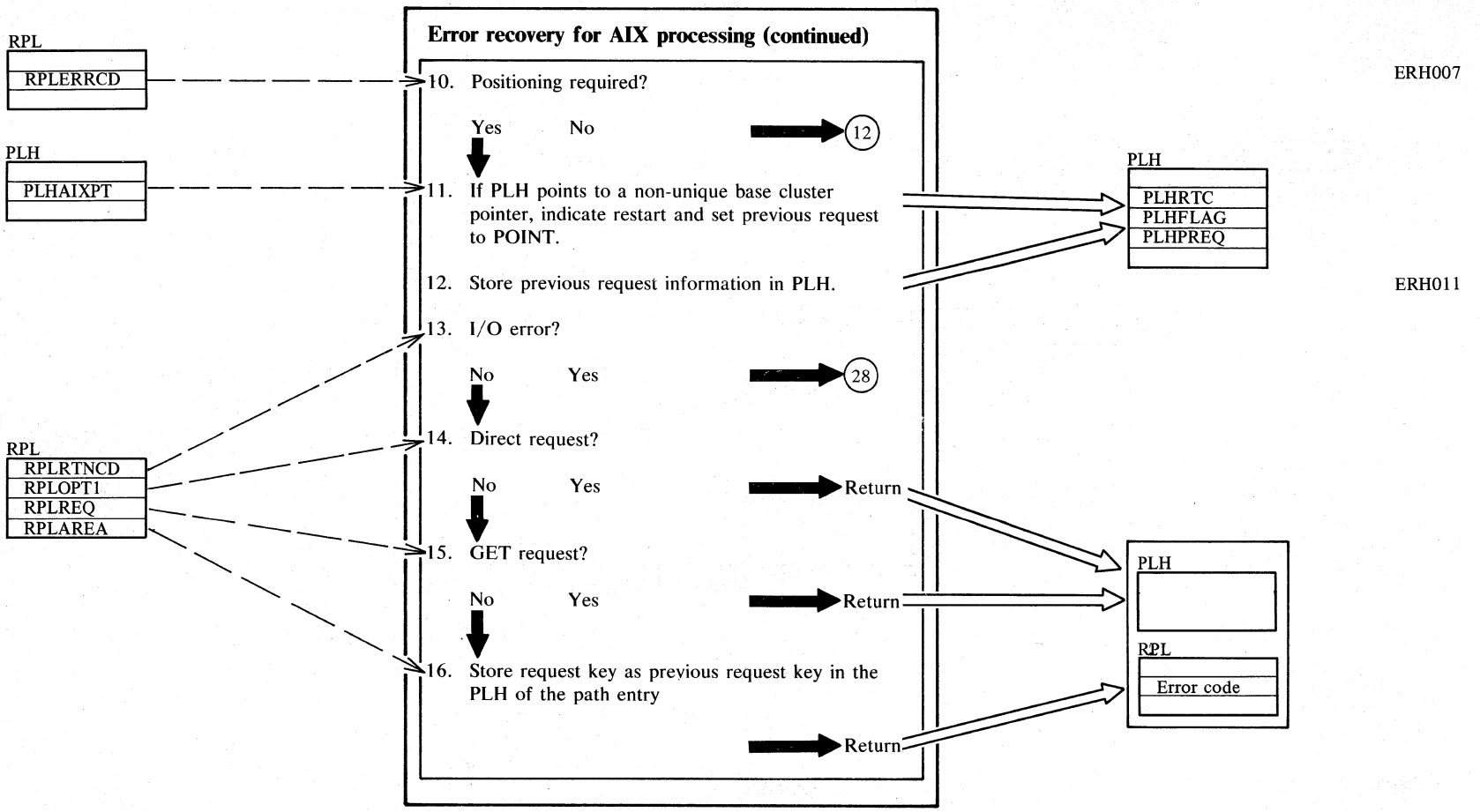


Diagram HF3. Error handler

Module

IKQERH

ERH019

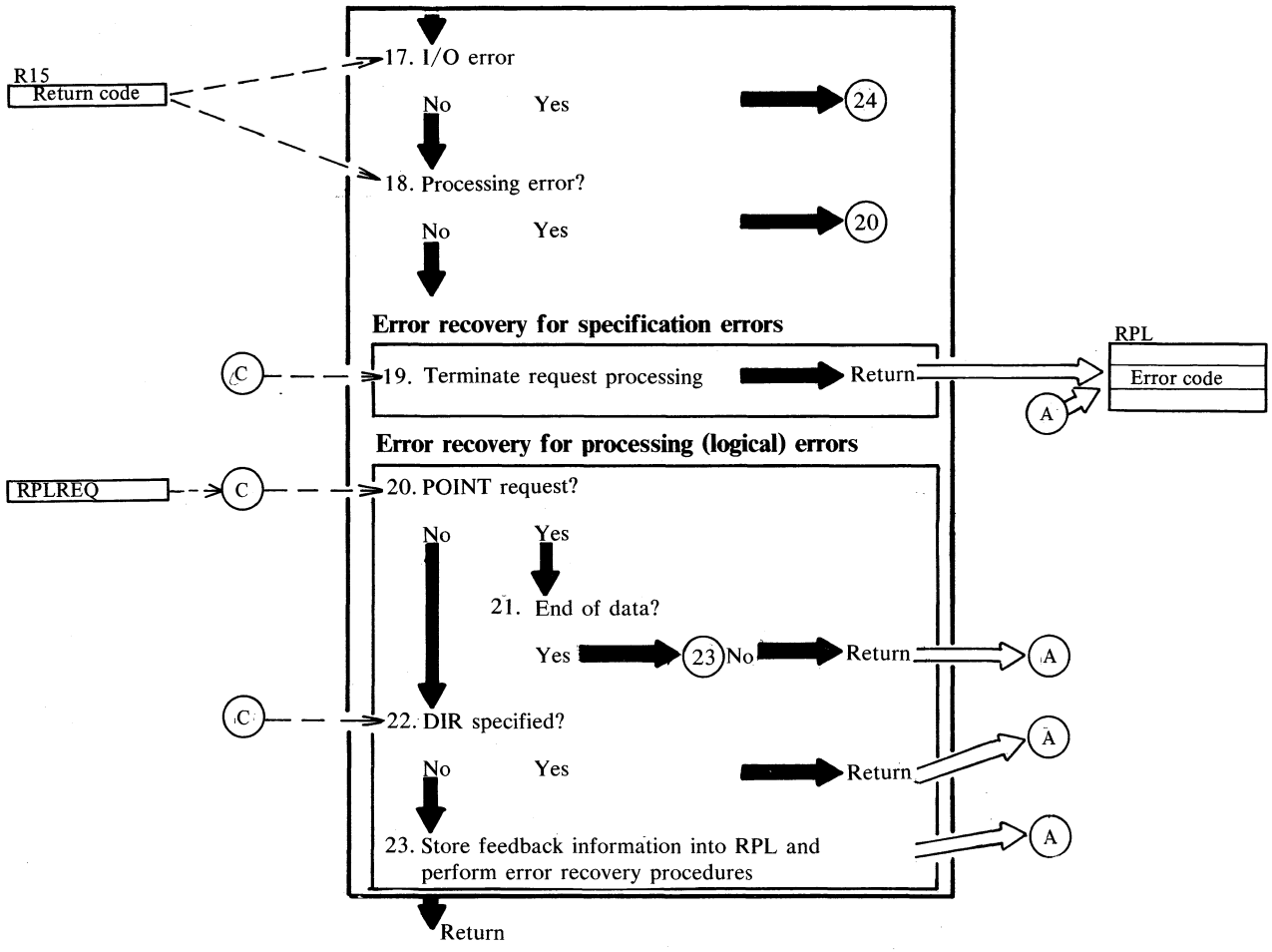
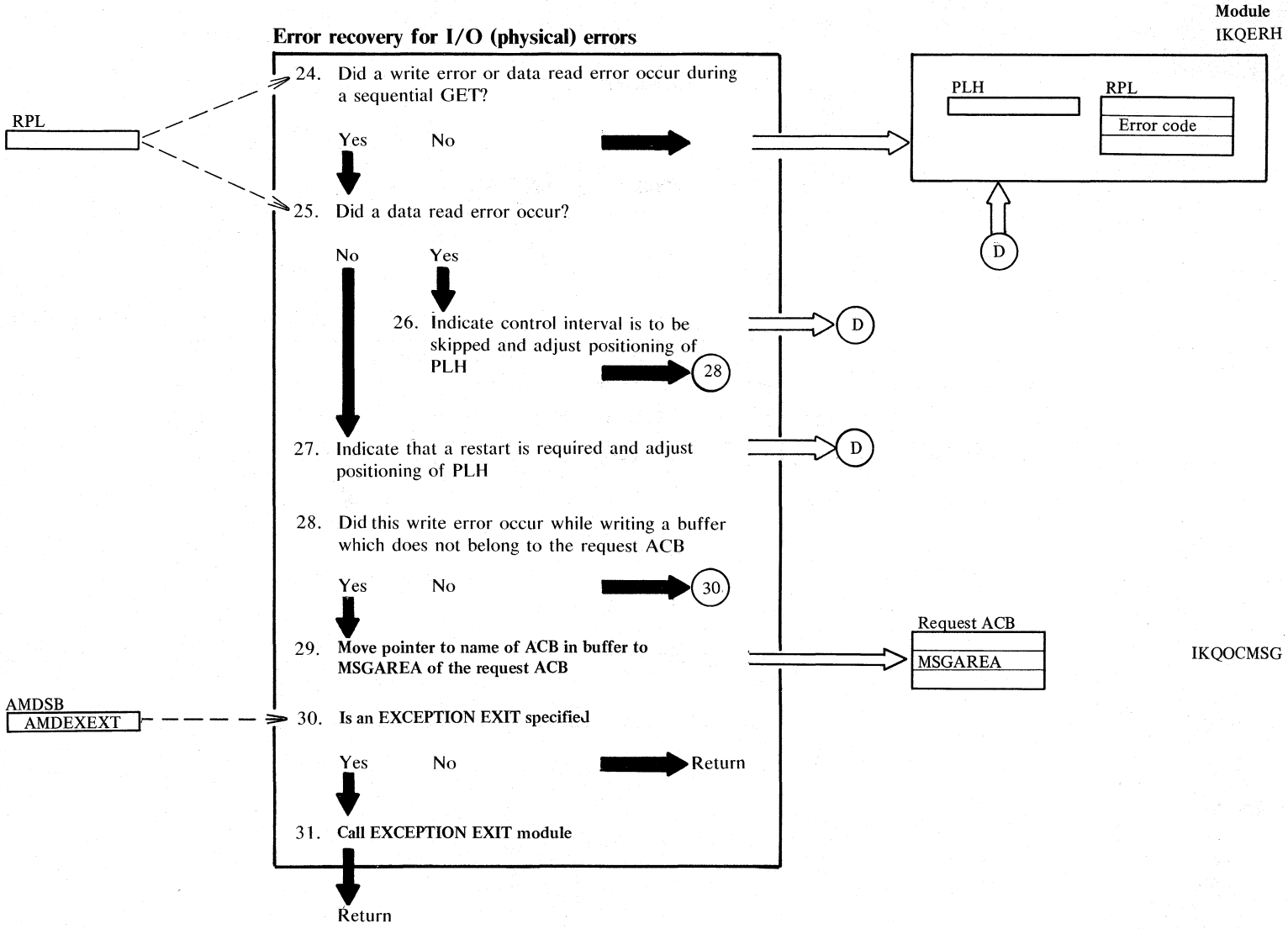


Diagram HF4. Error handler



Notes for Diagram HF

Description

1. If the user did not point to a valid RPL or the RPL is in use (used by another task), no feedback information can be stored into the RPL.

24. **A write error can occur on a GET when buffers (that have not been written) are written because more are needed for the GET.**

26. For a data read error occurring during a sequential GET, the user may continue processing. The next GET issued will skip the erroneous data control interval.

    If necessary, IKQBFA is called to release exclusive control and track hold.

27. For a write error occurring during a sequential GET, positioning will automatically be reestablished for the next request, that is, a restart is performed. To the user's program, the erroneous operation looks like a no-operation except that an error exit is taken or an error code is returned. A subsequent sequential GET will cause processing to continue.

28. If the write error occurs while writing a buffer which does not belong to the request ACB, field PLHACB points to the ACB to which the buffer belongs. Otherwise PLHACB is 0.

29. **A pointer to the ACB name is moved into the MSGAREA of the request ACB for user information.**

Diagram HG1. Error exit

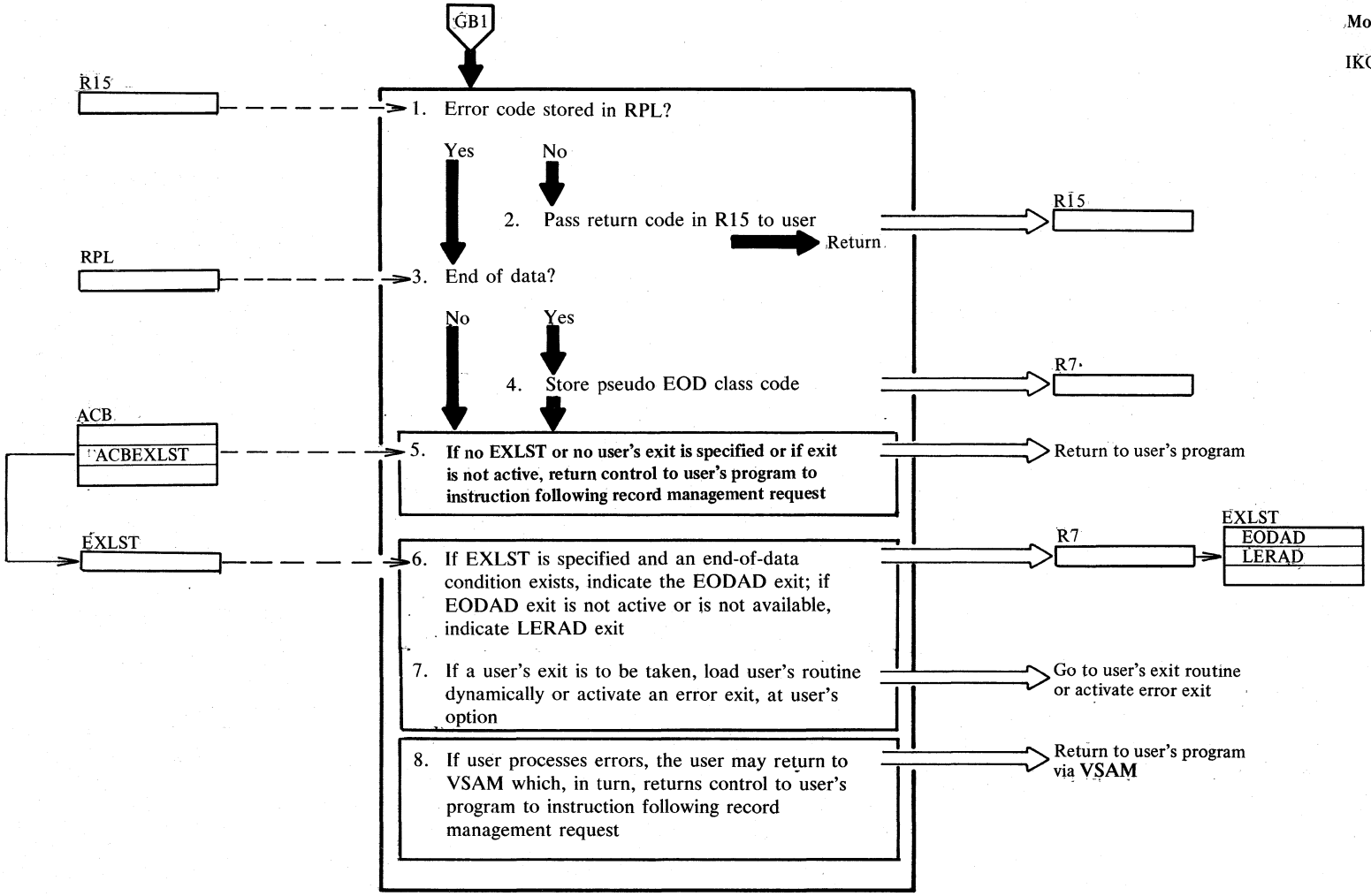


Diagram HH1. Record management close

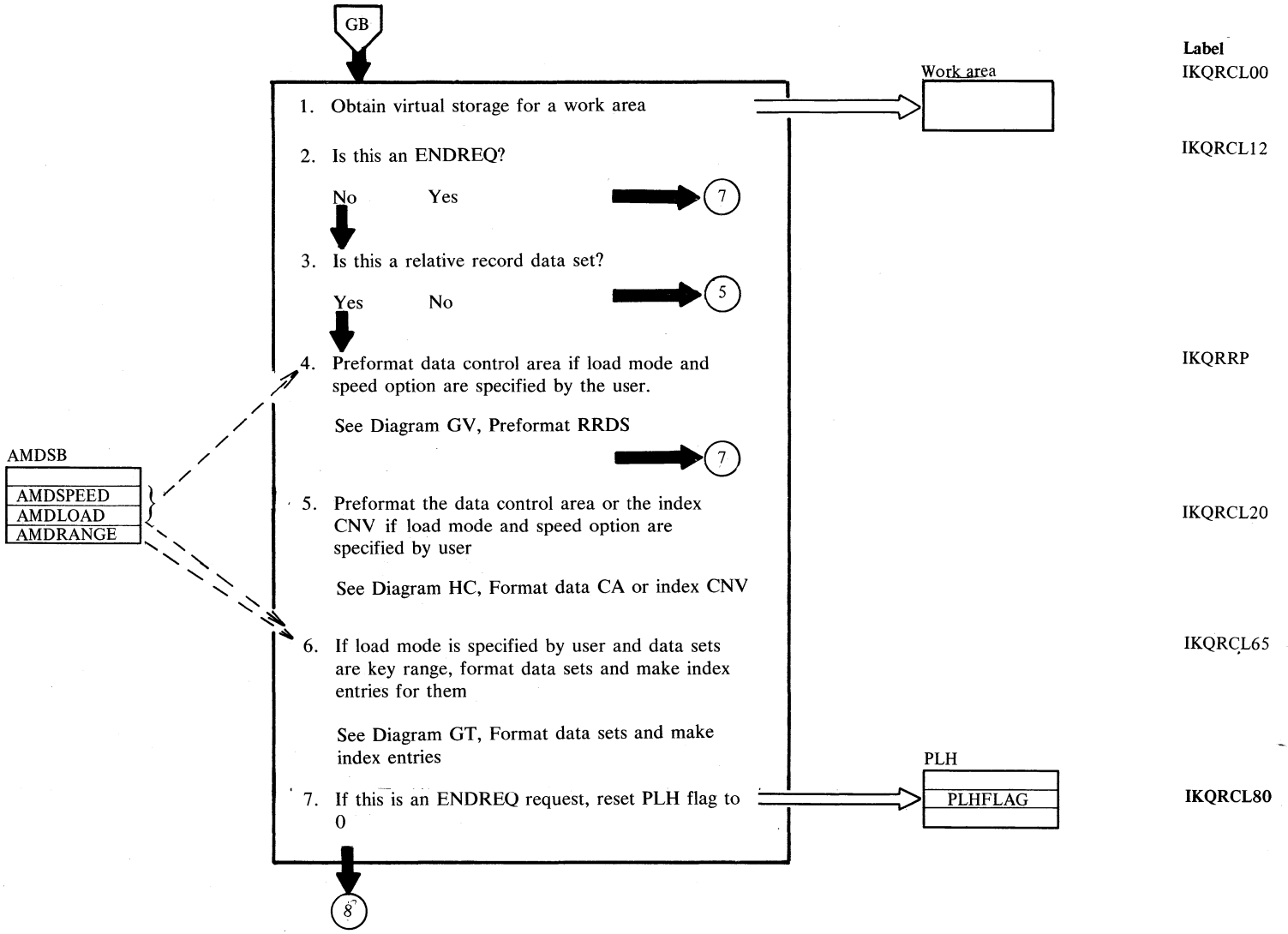
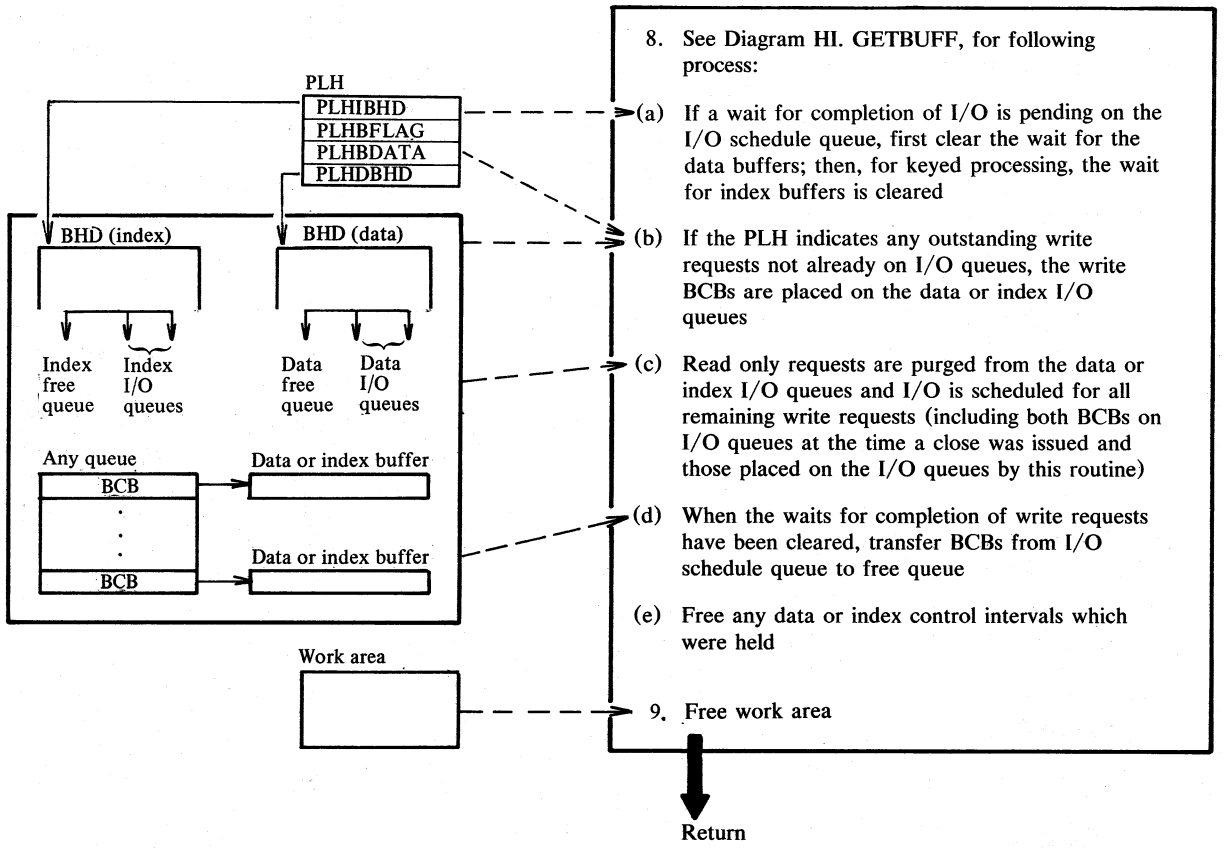


Diagram HH2. Record management close



Label
CBF020

IKQRCL90

Diagram HI1. Buffer manager: GETBUFF

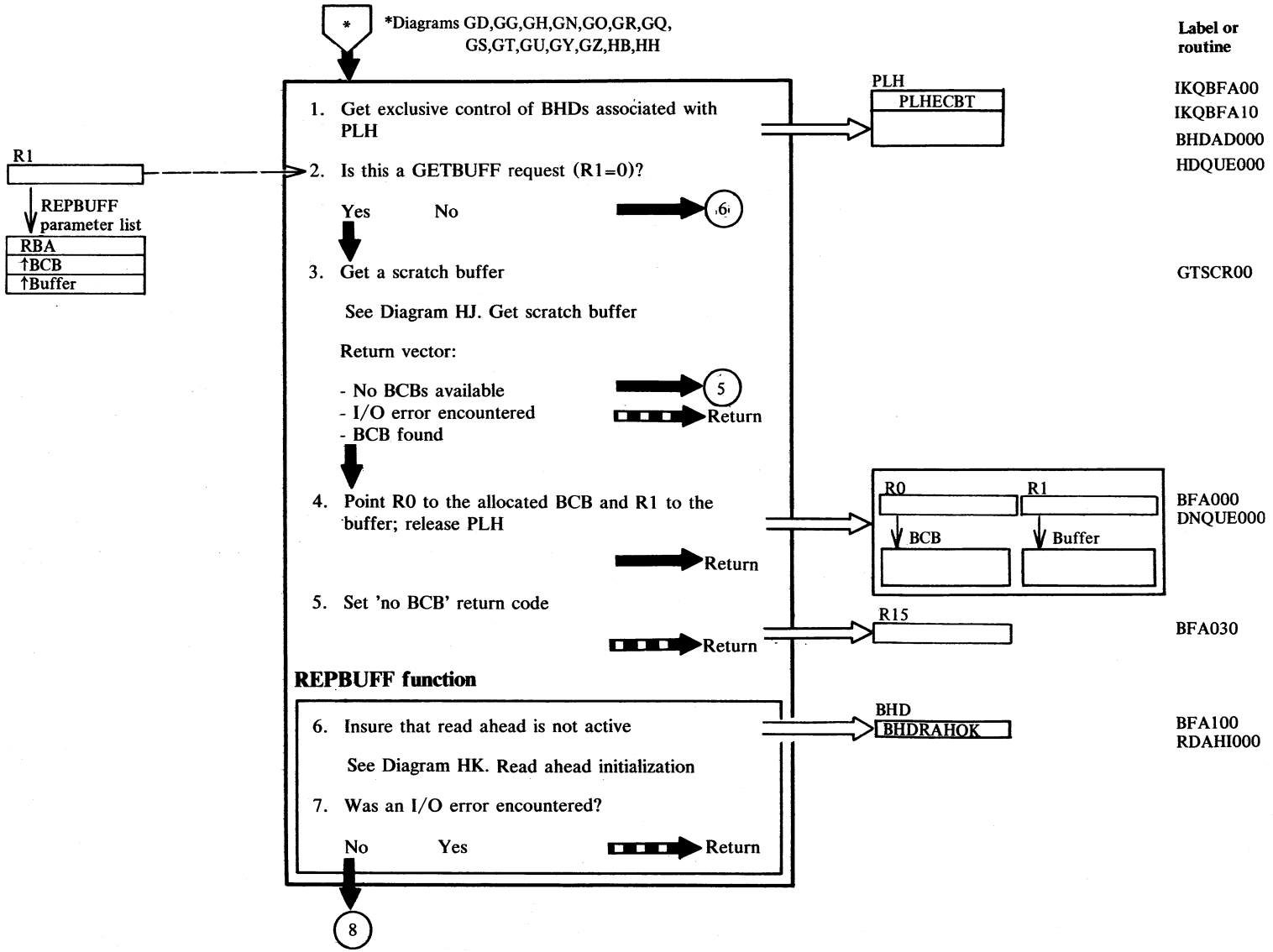


Diagram HI2. Buffer manager: GETBUFF

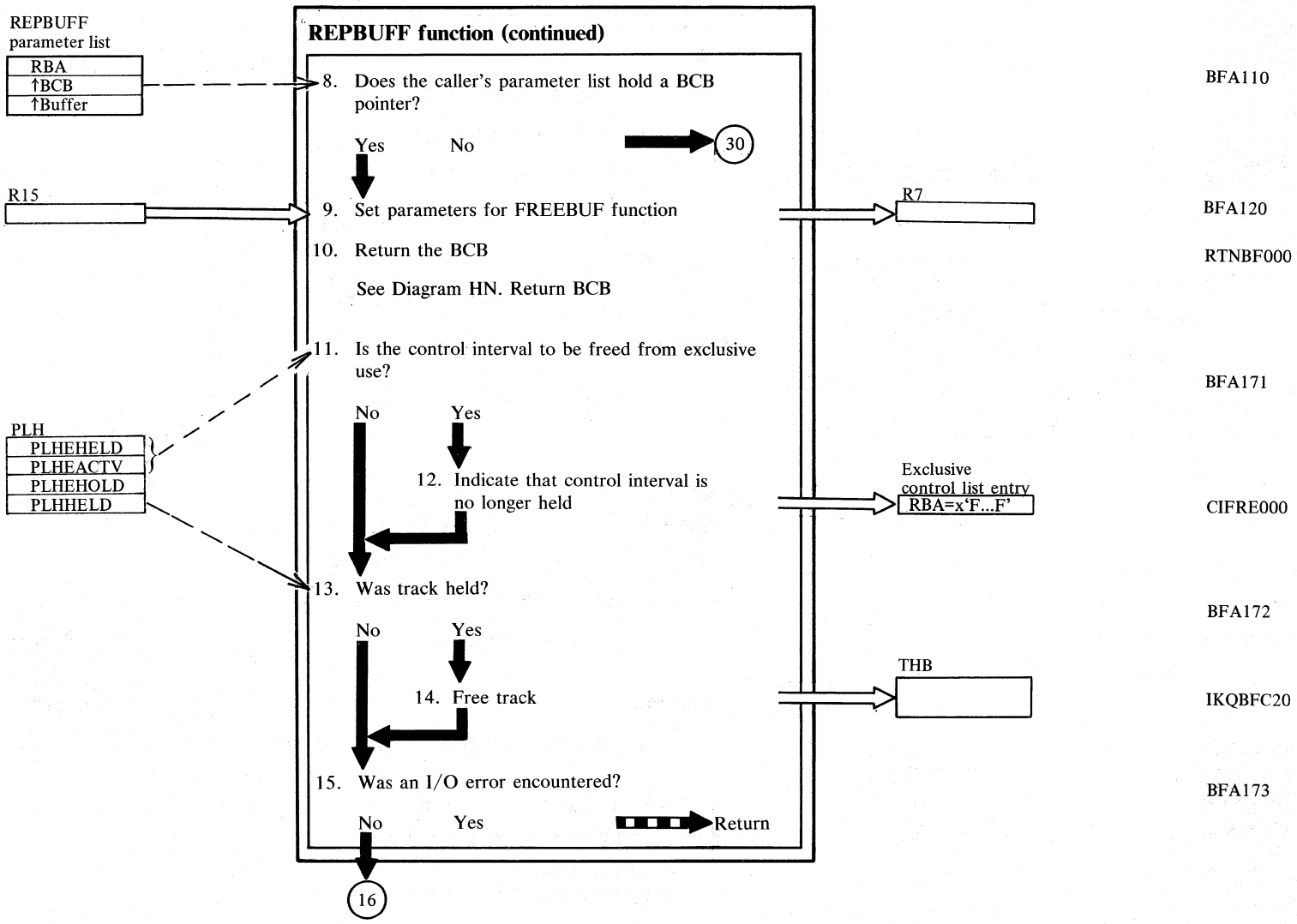


Diagram HI3. Buffer manager: GETBUFF

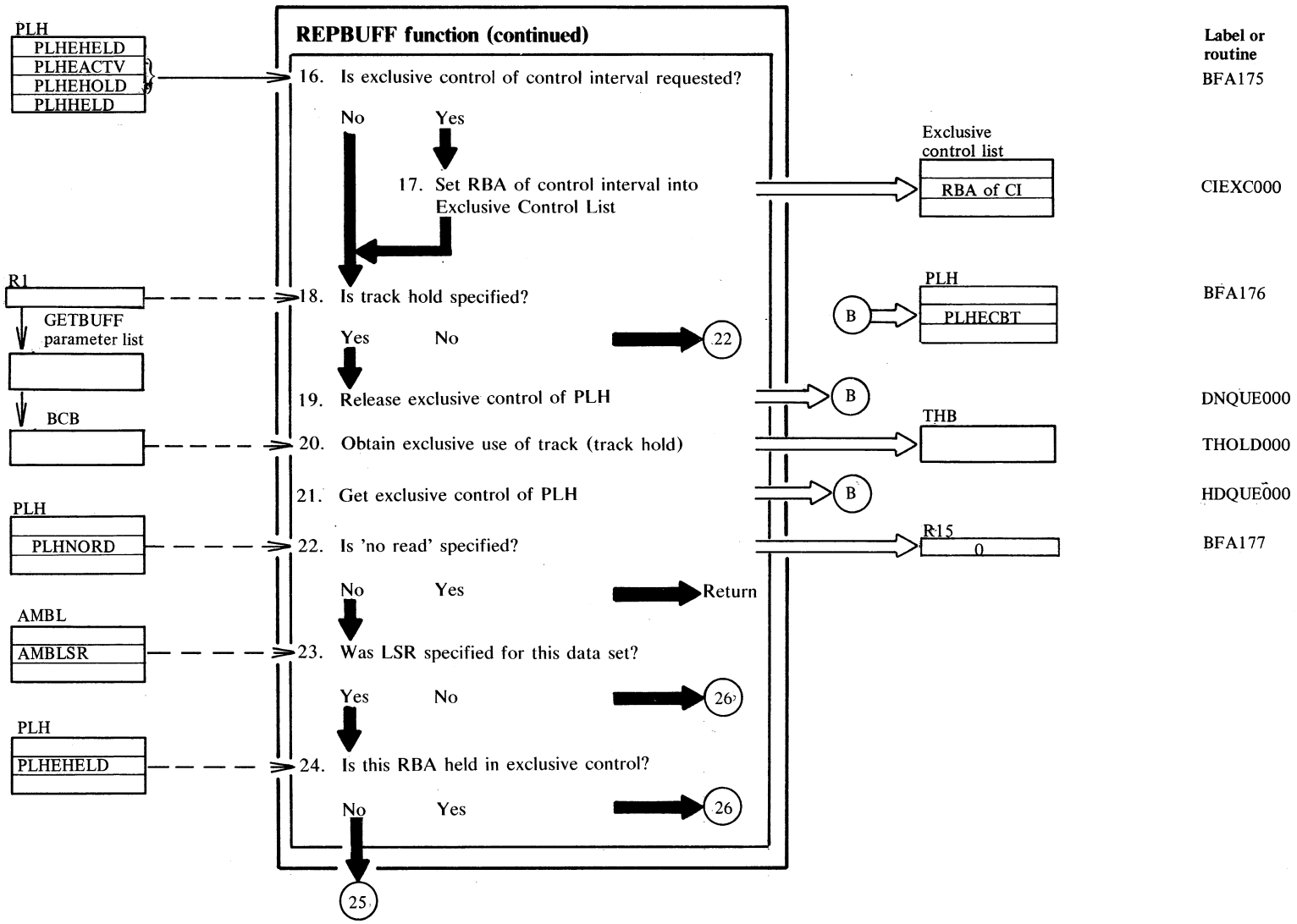


Diagram HI4. Buffer manager: GETBUFF

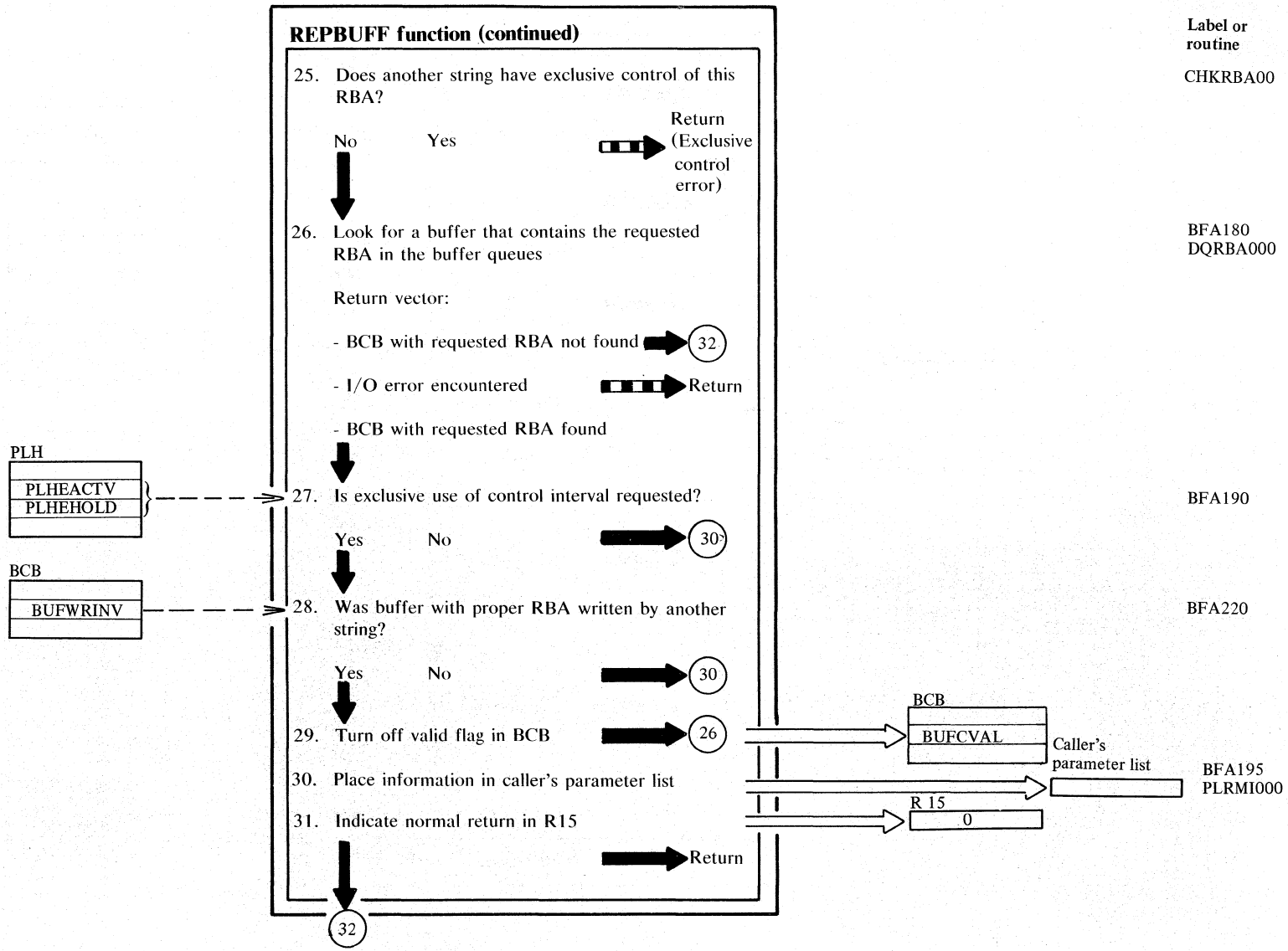


Diagram HI5. Buffer manager: GETBUFF

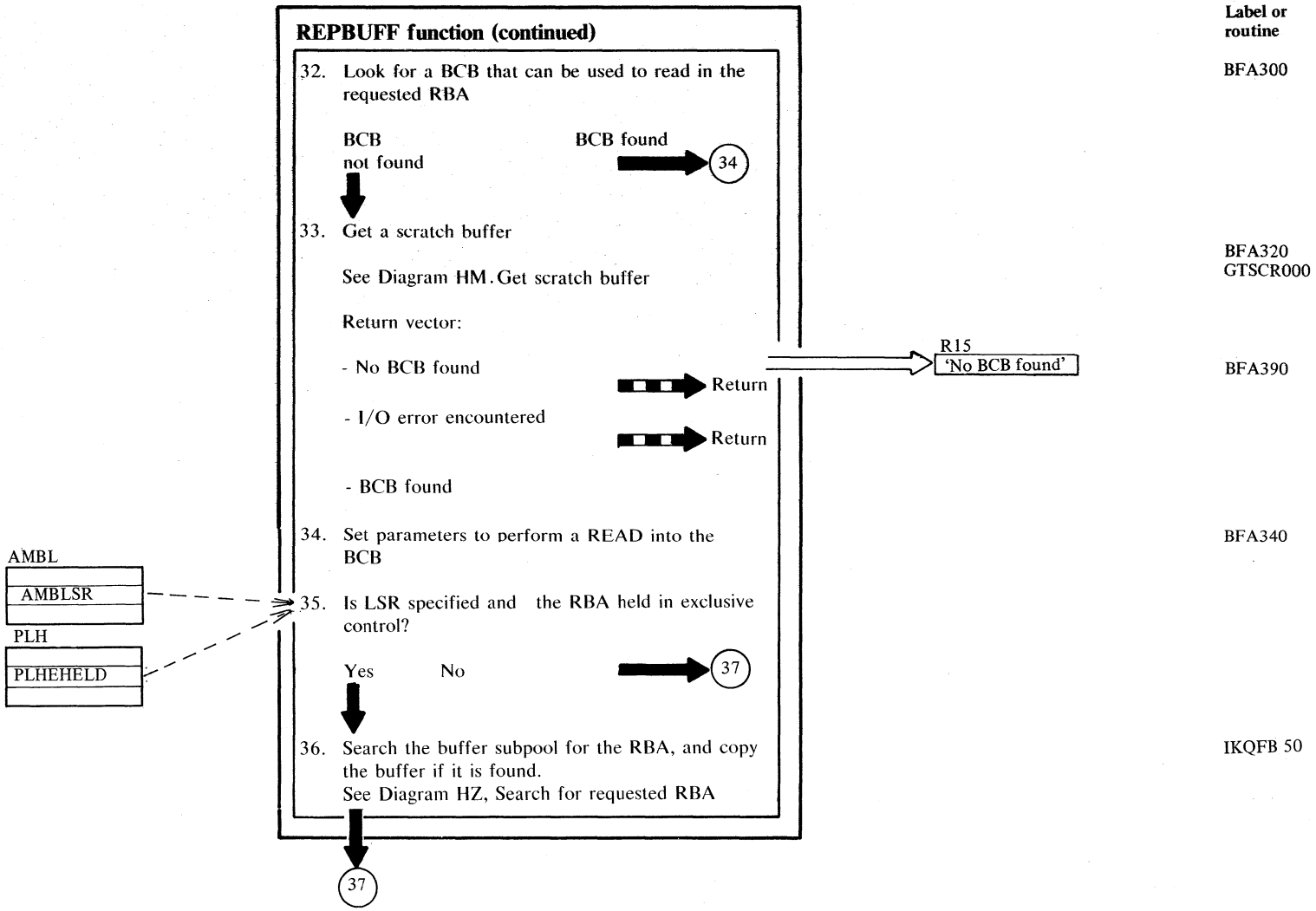
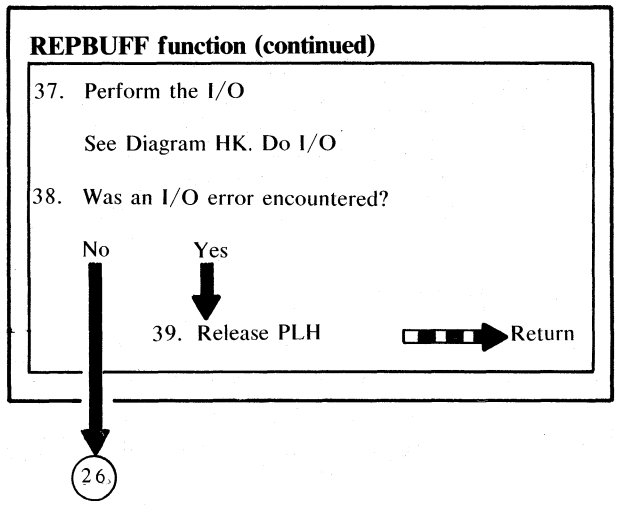


Diagram HI6. Buffer manager: GETBUFF



Label or Routine	
BFA370	37. Perform the I/O
DOIO000	See Diagram HK. Do I/O
BFA210	38. Was an I/O error encountered?
DNQUE000	39. Release PLH

Notes for Diagram HI

The two main functions of this module are FREEBUFF (free a buffer) and GETBUFF (get a buffer). The REPBUFF function is a combination of the two, except when 'no read' is specified. In this case, the function is basically a FREEBUFF.

FREEBUFF:
Free the input BCB contained in the parameter list:
1. Do any necessary I/O operations.
2. Place the BCB on the free queue.
Note: The first step may force additional I/O on the queues.

GETBUFF:
Get a buffer with the requested RBA:
1. Search the queues for the requested RBA, in order to determine if it is already in storage. If so, detach the buffer from the queue and complete the parameter list for return to the caller.

2. If the RBA is not located on the free queue, as indicated by the BCB 'buffer contents valid' flag, but is found on the schedule or no-schedule queue, force completion of the I/O, thus forcing the BCB into the free queue, where it can be used to satisfy the request.

Note for LSR processing:
If the RBA is not found in one of the queues, the subpool of the VSAM shared resources pool is searched for the requested RBA.

3. If the RBA is not in storage, get a scratch buffer and read in the RBA. Then repeat step 1, above.

Note: Once the request has been satisfied, the BCB/buffer is unknown to the buffer manager, as it has been detached from the queues.

NO READ:
If the 'no read' flag (PLHNORD) is set in the parameter list, the request is really a FREEBUFF request via the REPBUFF interface. This means that the 'requested' RBA is not read in. Only the BCB in the parameter list is freed.

Definitions of the queues

The are four queues which can hold BCBs. These are shown below, with their pointers from the BHD and their contents.

Schedule queue:
Located by BHDSKDQ. Contains BCBs for which I/O has been started without wait.

Non-schedule queue:
Located by BHDNSKDQ. Contains BCBs for which I/O needs to be done but has not been started yet.

I/O queue:
Located by BHD1STW. Contains BCBs on which the I/O manager is presently working (to build channel programs). This is the primary input from the buffer manager to the I/O manager.

Free queue:
Located by BHD1STF. Contains BCBs (with and without buffers with valid contents) which are available to record management.

Diagram HJ1. Buffer manager: Get scratch buffer

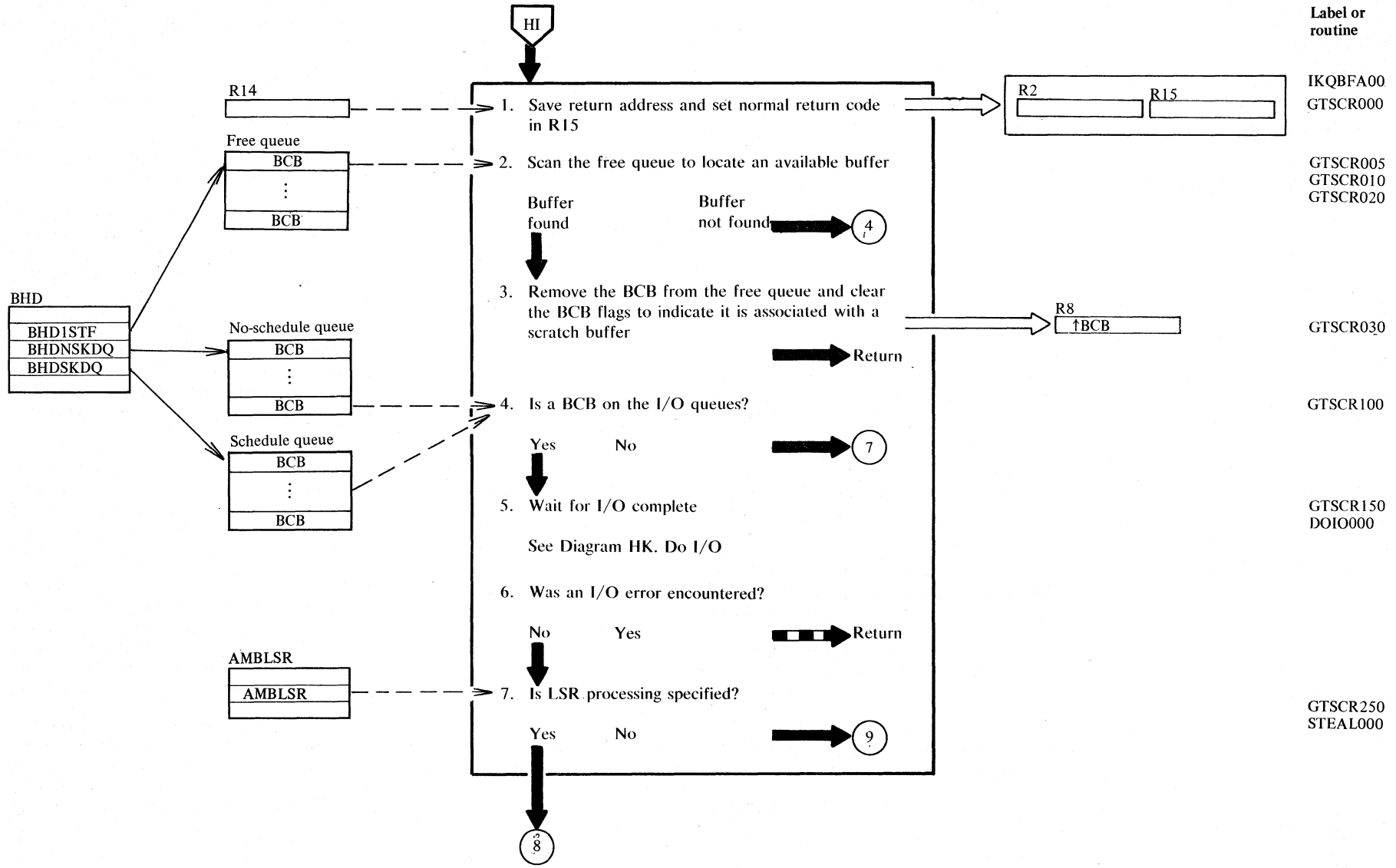
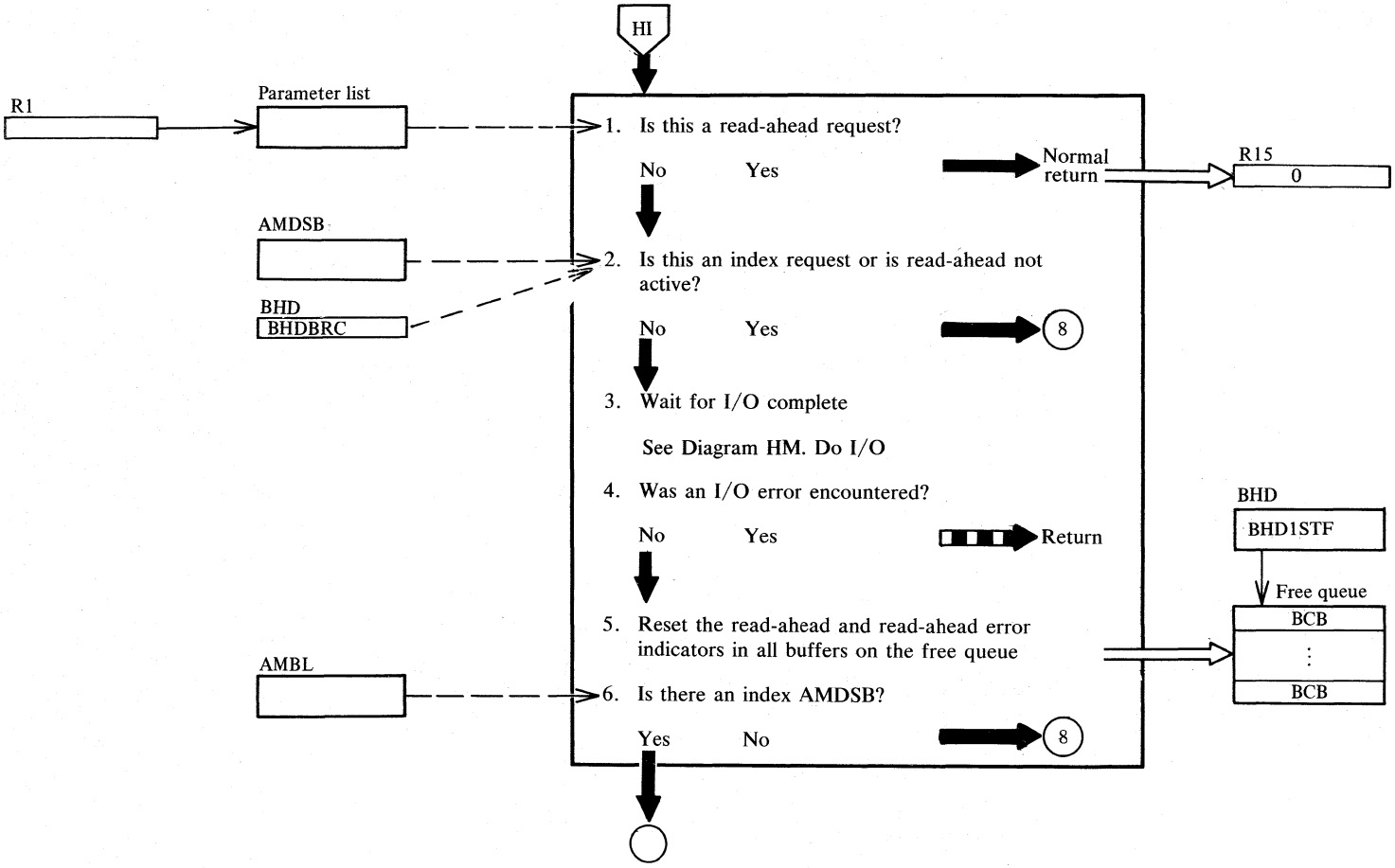


Diagram HJ2. Buffer manager: Get scratch buffer

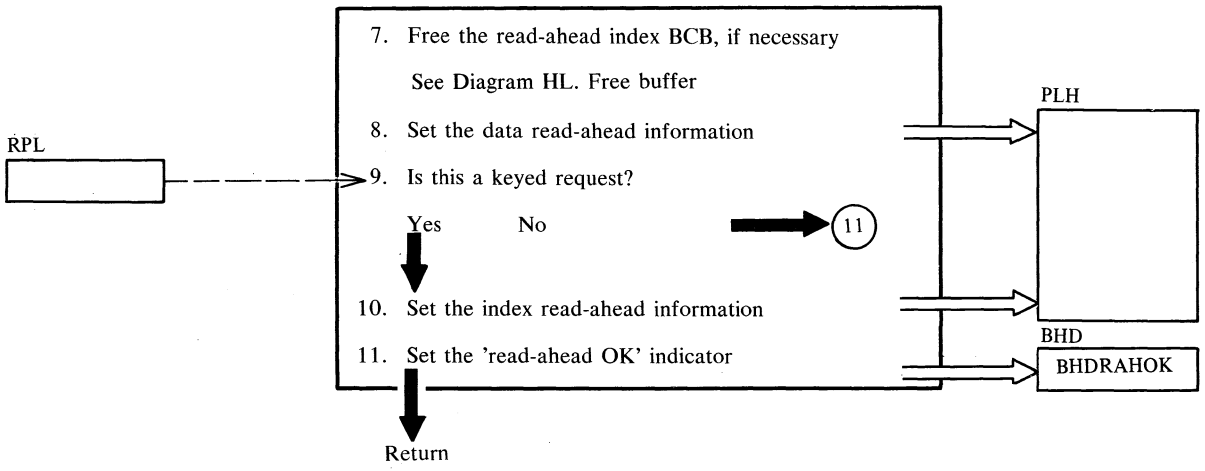
Label or Routine	
IKQBFB30	8. Search buffer subpool for a scratch buffer See Diagram HX. Get a sratch buffer
IKQBFA20 STEAL020	9. Must get a buffer from another string Return vector: - Buffer found → 3 - Buffer not found → Return

Diagram HK1. Buffer manager: Read-ahead interface



Label or routine
IKQBFA00
RDAHI000
DOIO000
RDAHI100

Diagram HK2. Buffer manager: Read-ahead interface

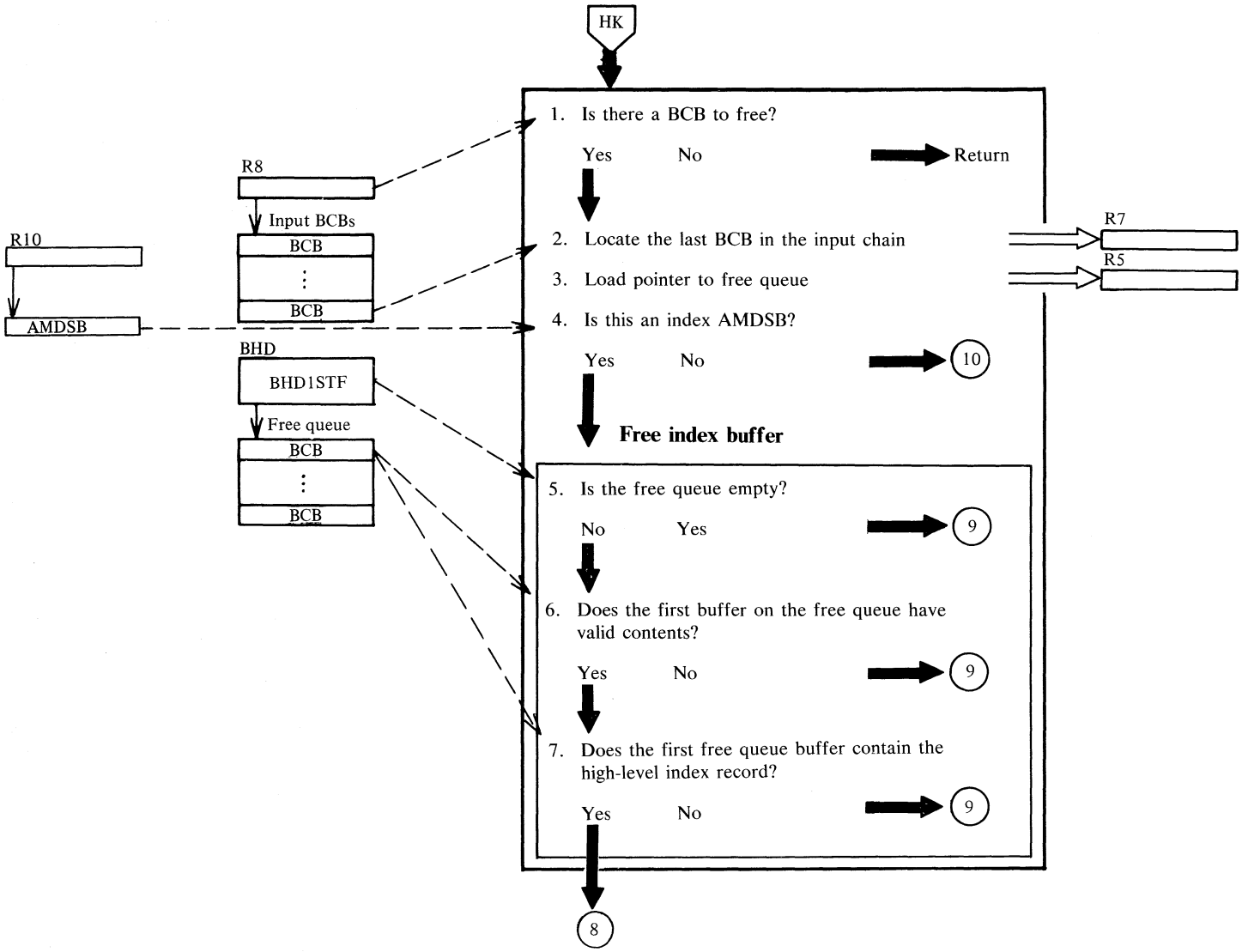


Notes for Diagram HK

This routine handles any non-sequential request that occurs while read-ahead is active. If a non-sequential request, such as GET DIRECT, is received after read-ahead has been started, this interrupts the read-ahead operations, and the read-ahead routine in deactivated, and the read-ahead control information is reset. Read-ahead can be activated on a subsequent GET SEQUENTIAL when a control interval is read (GETNXT processing).

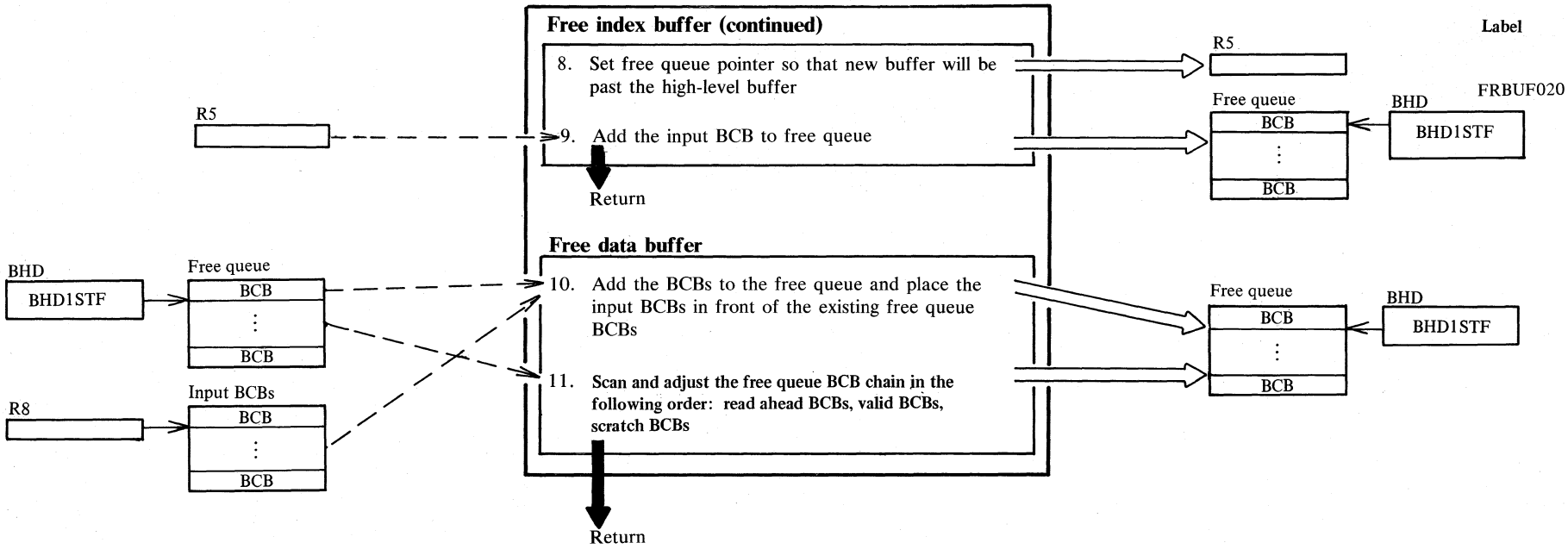
5. If a read-ahead error occurs, the buffer is flagged invalid.

Diagram HL1. Buffer manager: Free buffer



Label	
FRBUF000	
IKQBFA40	
	FRBUF010

Diagram HL2. Buffer manager: Free buffer



Notes for Diagram HL

8. Make sure a buffer containing the high-level index control interval is lowest priority to be scratched for reuse.

11. Make sure the free queue BCB chain is in the following order of increasing priority to be scratched for reuse:

- Read-ahead BCBs (valid contents)
- Other BCBs with valid contents
- Scratch BCBs (no valid contents)

Diagram HM1. Buffer manager: Do I/O

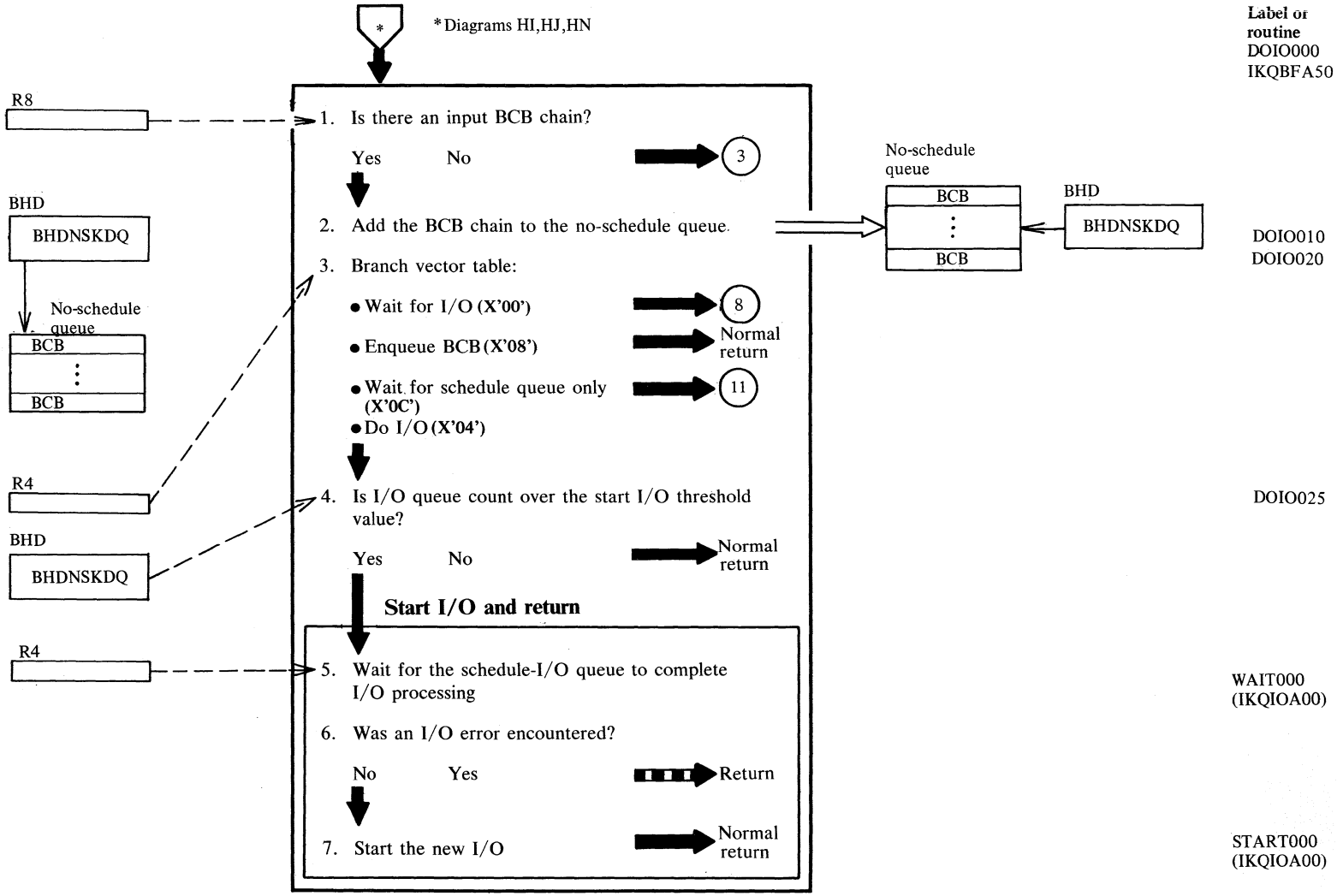
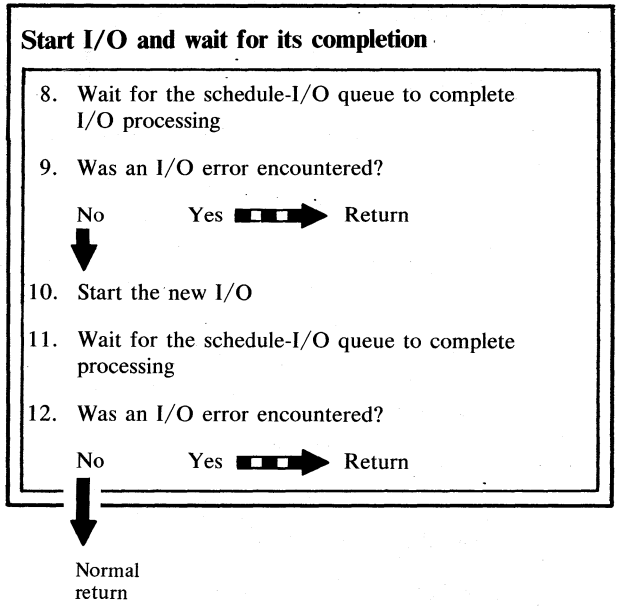


Diagram HM2. Buffer manager: Do I/O



Notes for Diagram HM

2. Add BCB to no-schedule queue, and increment count of I/Os to be done.

3. On entry to the Do I/O routine, R4 contains one of the following codes:

   0 - Wait for all I/O to complete. (I/O started for user.)

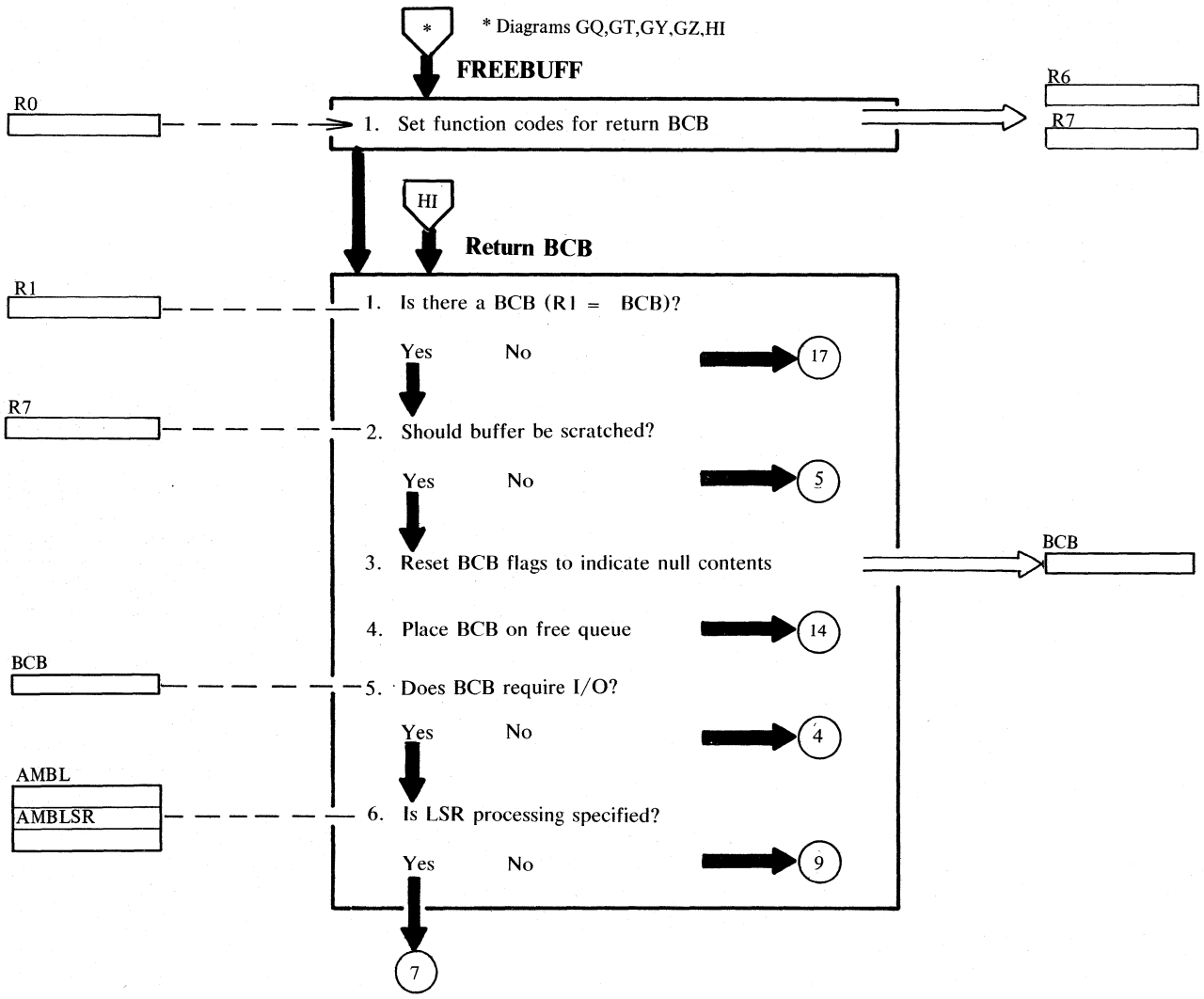
   4 - If threshold (see below) has been reached, wait for previously started I/O to complete, and start I/O not previously started.

   8 - Enqueue the BCB.

   12 - Wait for previously started I/O to complete.

4. The threshold test consists of comparing the count of I/O operations to do (number of BCBs on the no-schedule queue) to a fixed threshold value. For non-shared resources, the value is either:
   - The total number of BCBs owned by this PLH, if that number is less than 4.
   - One-half the total number of BCBs owned by this PLH (rounded high if an odd number), if that number is greater than or equal to 4.

Diagram HN1. Buffer manager: FREEBUFF and return BCB



Label

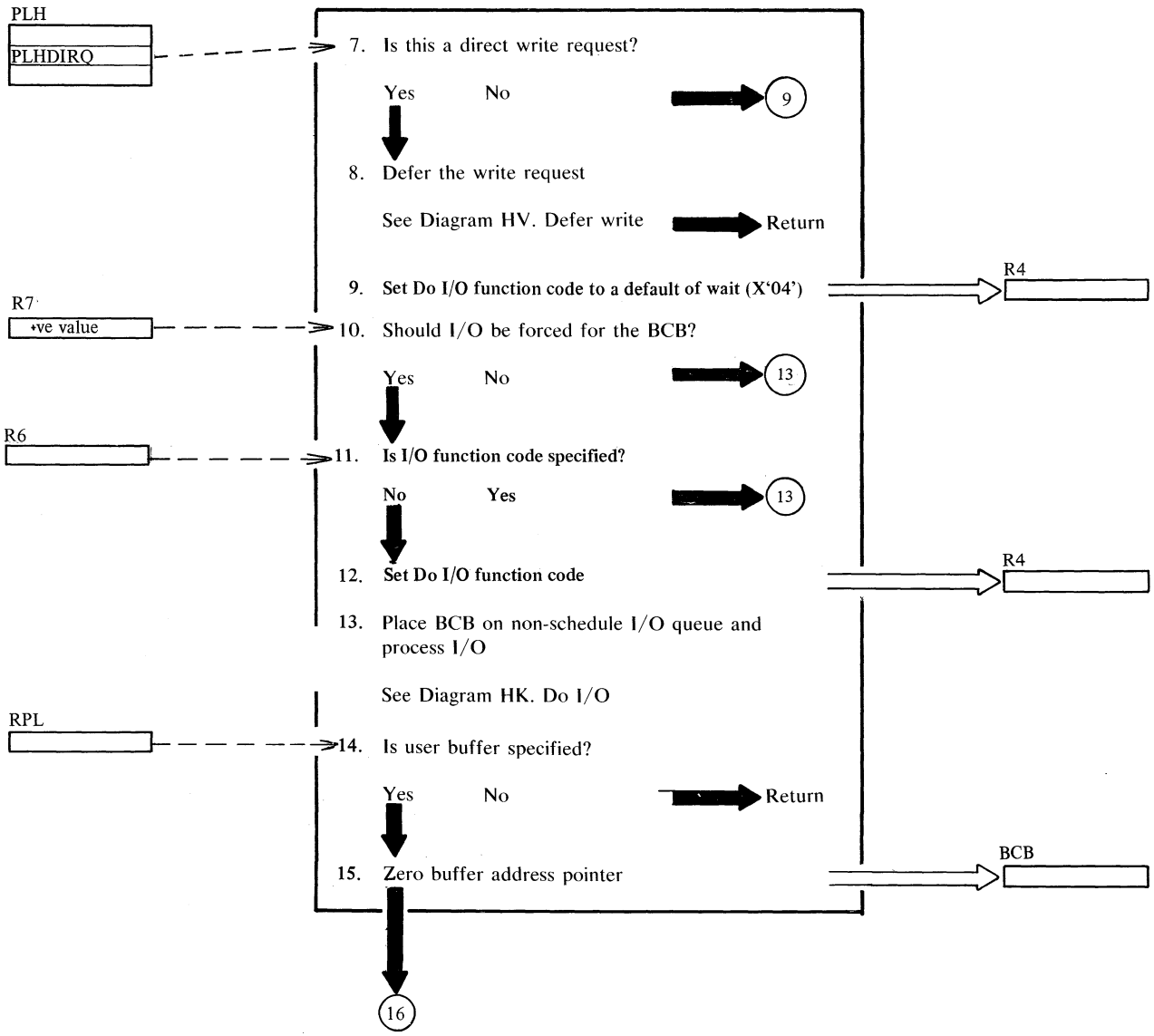
IKQBFA20
FREEBUFF

IKQBFA80
RTNBF000

RTNBF010

RTNBF020

Diagram HN2. Buffer manager: FREEBUFF and return BCB



Label

IKQBFB10

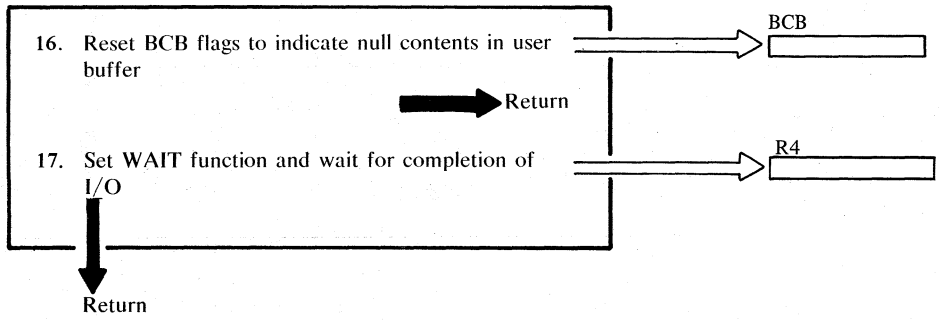
RTNBF025

RTNBF030
DOIO000

RTNBF040

RTNBF045

Diagram HN3. Buffer manager: FREEBUFF and return BCB



Label
RTNBF050
RTNBF100
DOIO000

Notes for Diagram HN

1. A BCB address is passed in R1. If this address is zero, FREEBUF must wait on all I/O. (For all references to the Do I/O routine and Do I/O function codes, see Diagram HM.) If this routine is entered at IKQBFA20, the function code is passed in R0 and moved to R7, and R6 is set to zero before proceding.

   Regardless of entry point, R7 contains one of the following function codes:

   negative - BCB is to be scratched.

   zero - if I/O flags are set on in the BCB, go to the Do I/O routine to start I/O but not wait.

   positive - If I/O flags are set on in the BCB, go to the Do I/O routine to start I/O and wait.

   R6 may also contain a code:

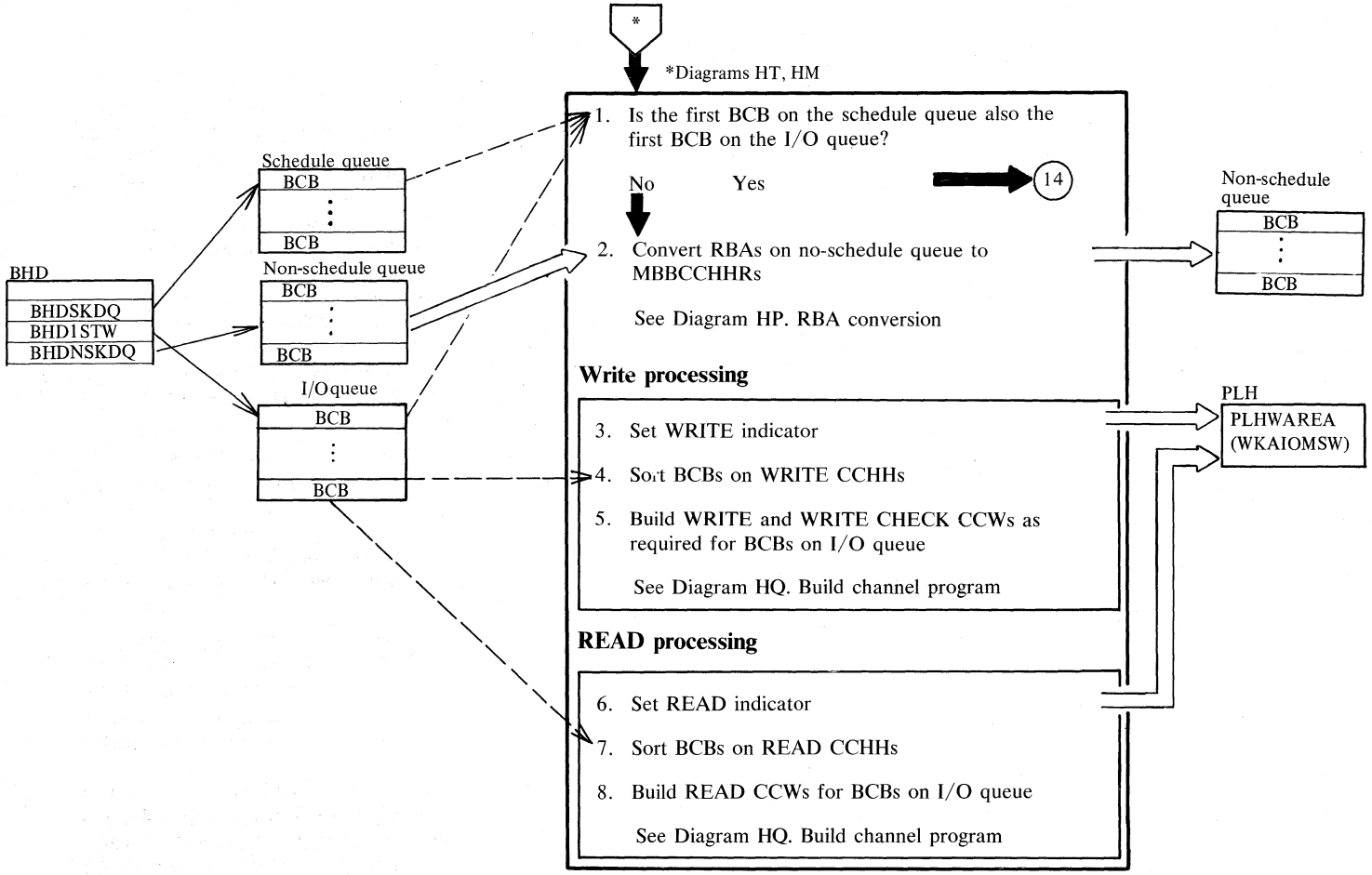
   zero - Use R7 function codes as described above.

   non-zero - Override R7 function code if it was zero or positive. The override consists of decrementing by 1 the code in R6, and passing the result to the Do I/O routine as its function code.

10. I/O is forced if the BCB is for a catalog, or if a share option 4 data set is being processed.

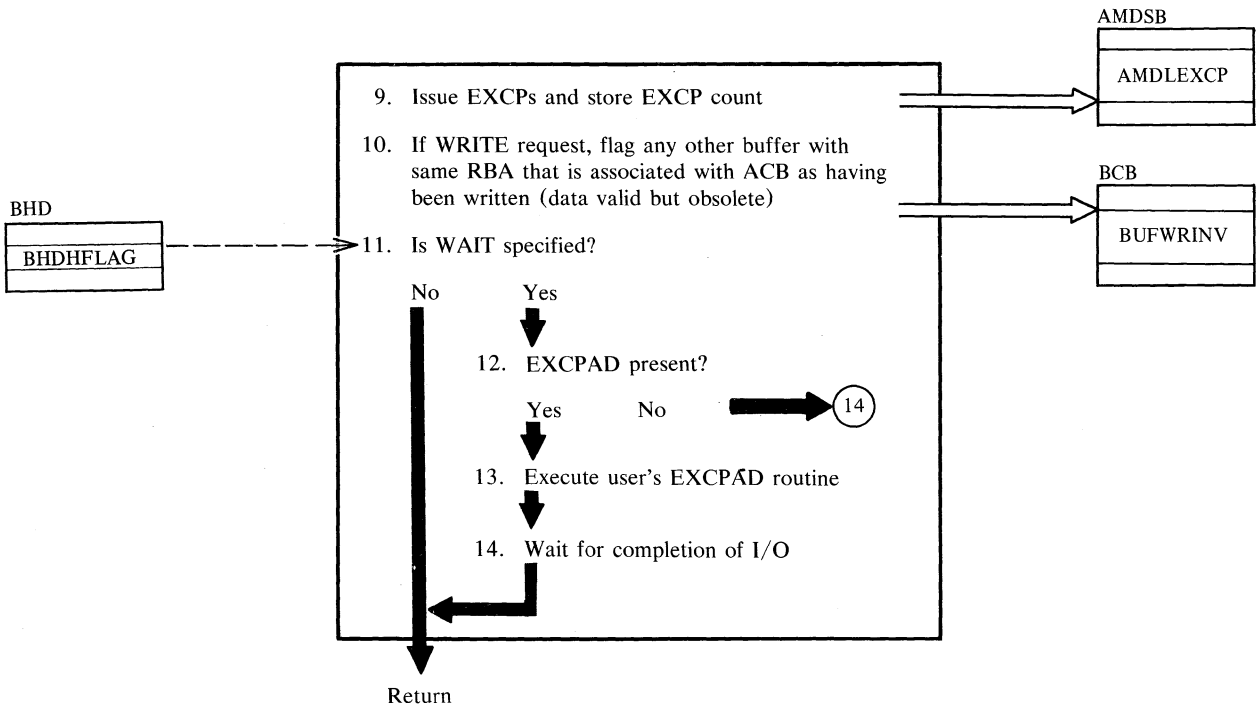
12. The I/O function code is set to X'04' (start all I/O) if R7 contains zero; the code is set to X'00' (wait for all I/O to complete) if R7 is positive.

Diagram HO1. I/O manager



Label or routine
IKQIOA00
RBACN000
IOA250
SORT000
IOA420
BLDCP000
IOA210
SORT000
IOA520
BLDCP000

Diagram HO2. I/O manager

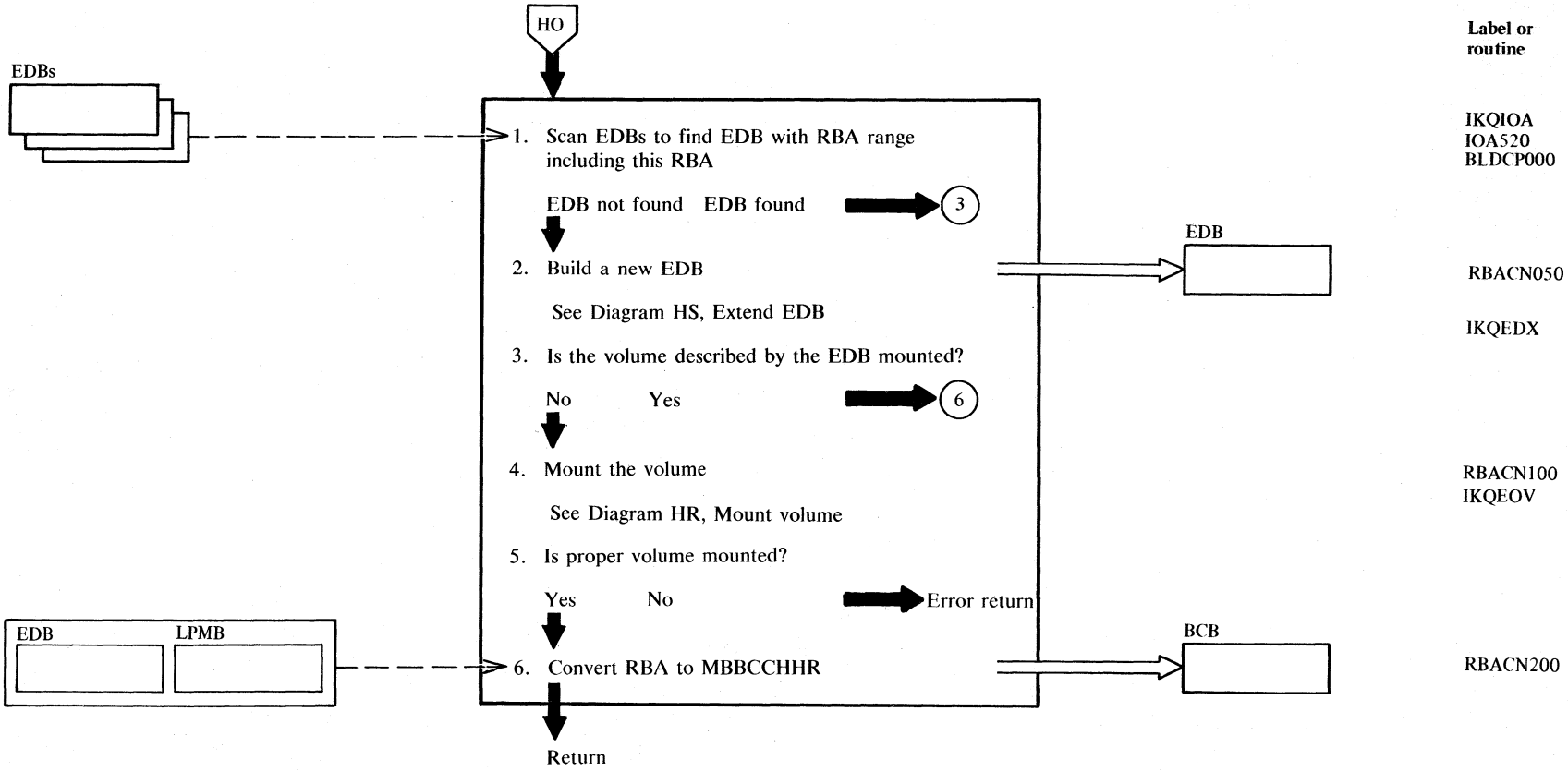


Label or routine

EXCP000

WAIT000

Diagram HP1. I/O manager: RBA conversion

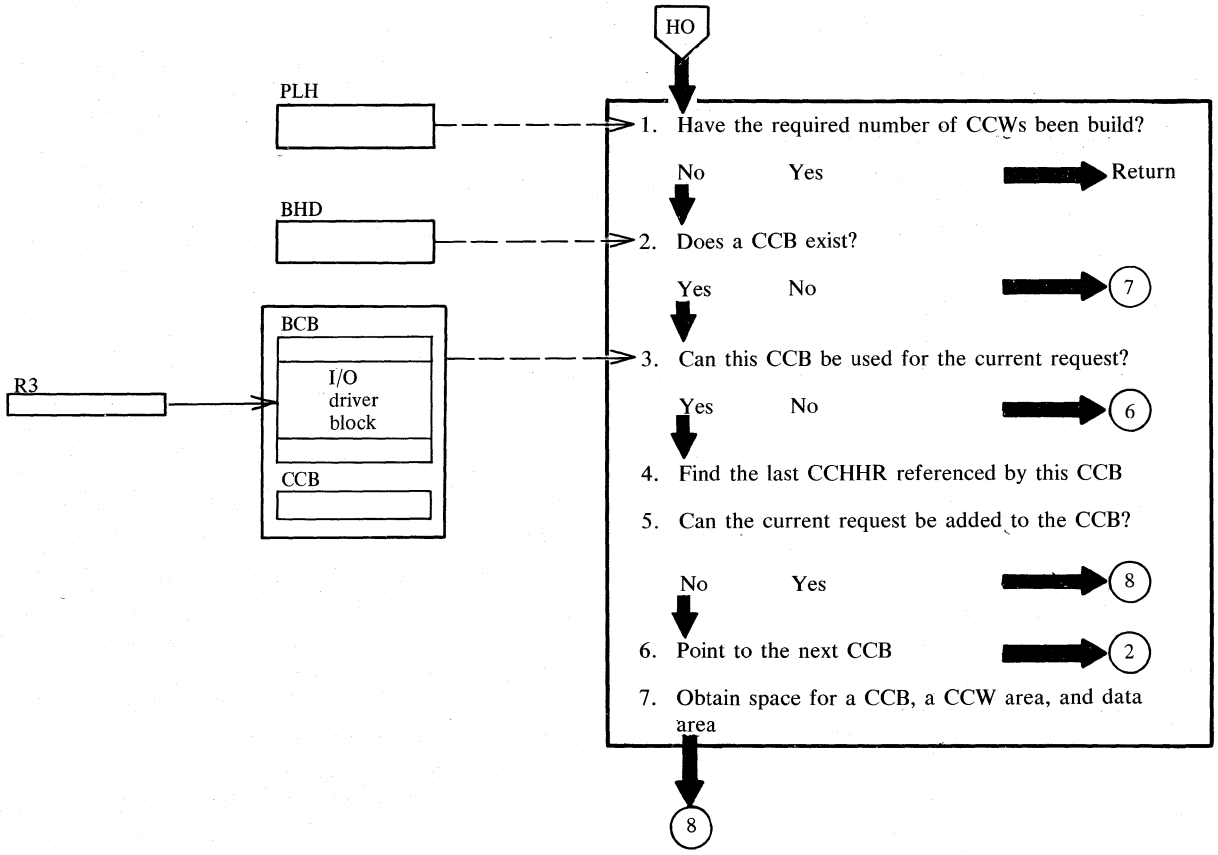


Notes for Diagram HP

6. The RBA conversion formula is:

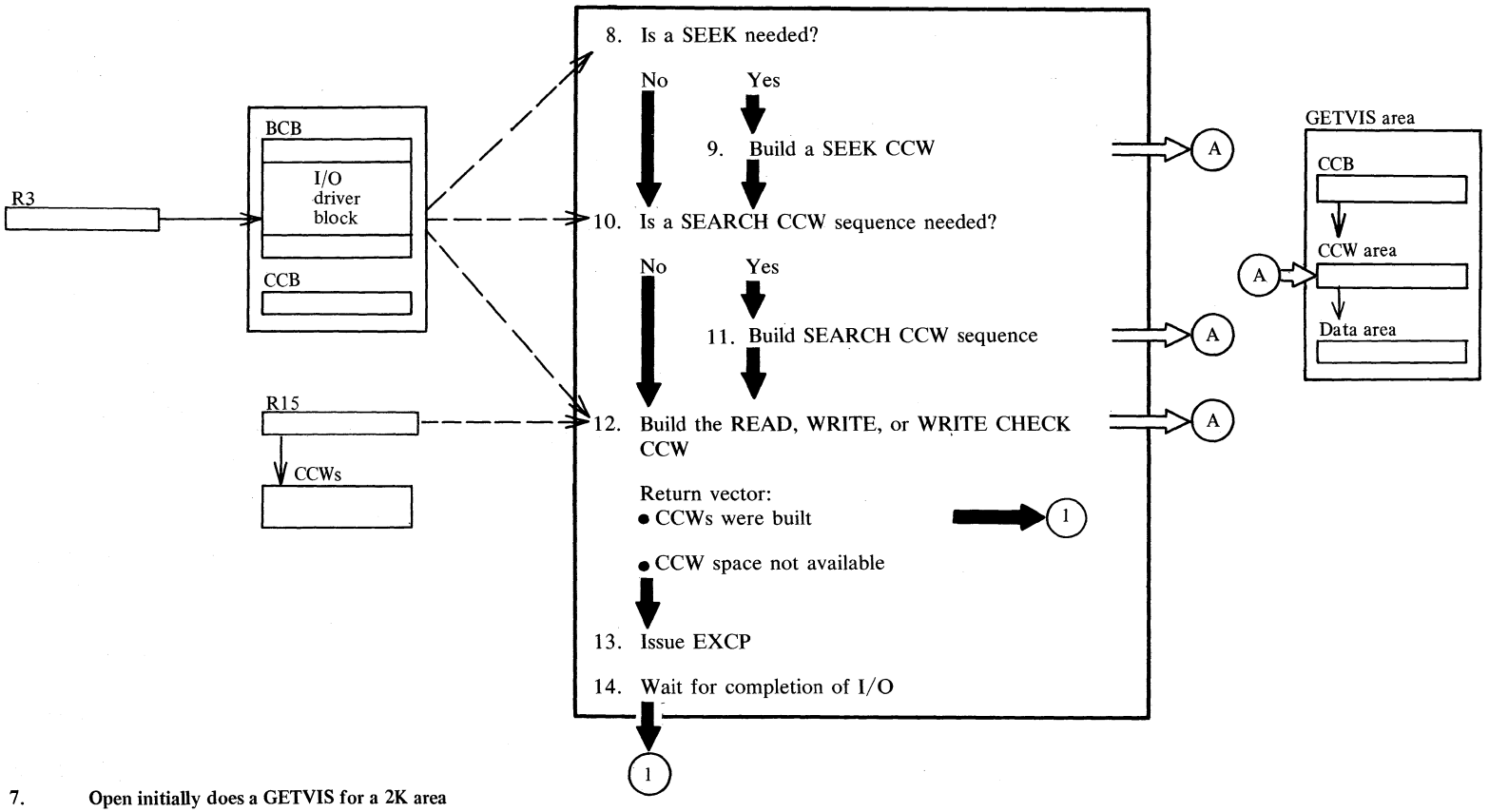
- Low RBA = RBA displacement within extent
- Relative control area number within extent (RAN) = $\frac{\text{low RBA}}{\text{bytes per control area}}$
- Relative track within control area (RTT) = remainder of $\left(\frac{\text{RAN}}{\text{bytes per track}}\right)$
  (Add 1 to RTT if this is a data RBA and the sequence set is imbedded in the data area.)
- Record number (R) = remainder of $\left(\frac{\text{RTT}}{\text{record size}}\right) + 1$
- Absolute track (ATT) = RAN x (tracks per control area) + RTT + start track of extent.
- CC = $\frac{\text{ATT}}{\text{tracks per cylinder}}$
- HH = remainder of $\left(\frac{\text{ATT}}{\text{tracks per cylinder}}\right)$

Diagram HQ1. I/O manager: Build channel program



Module or routine
IKQIOA
BLDCP000
BLDCP005
BLDCP020
BLDCP040
BLDCP100

Diagram HQ2. I/O manager: Build channel program



7. **Open initially does a GETVIS for a 2K area to be used for building CCBs, channel programs, and the DASD address arguments for the channel programs. This 2K area is suballocated into 64-byte blocks by the ALLBK000 subroutine.**

Diagram HR1. Mount volume

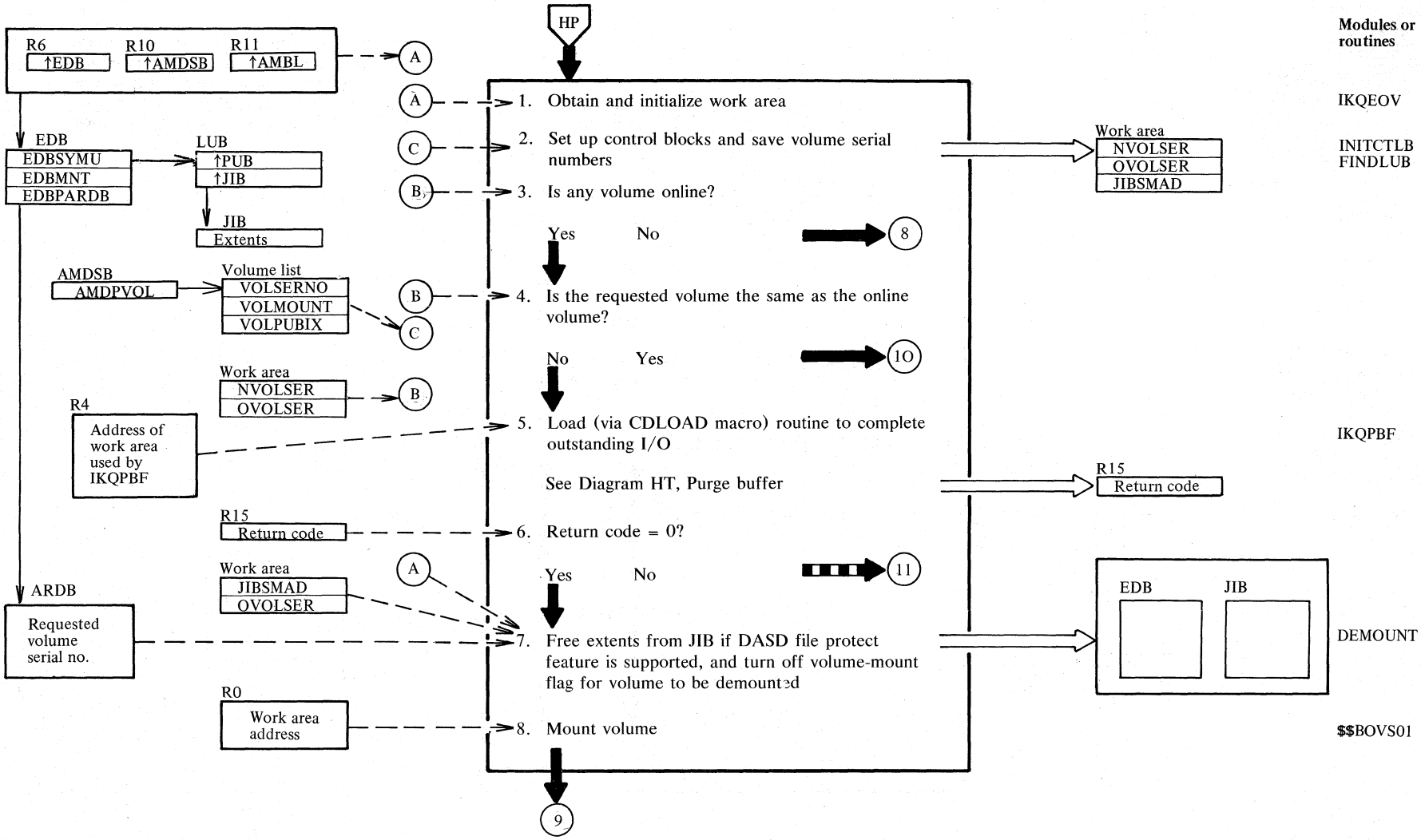


Diagram HR2. Mount volume

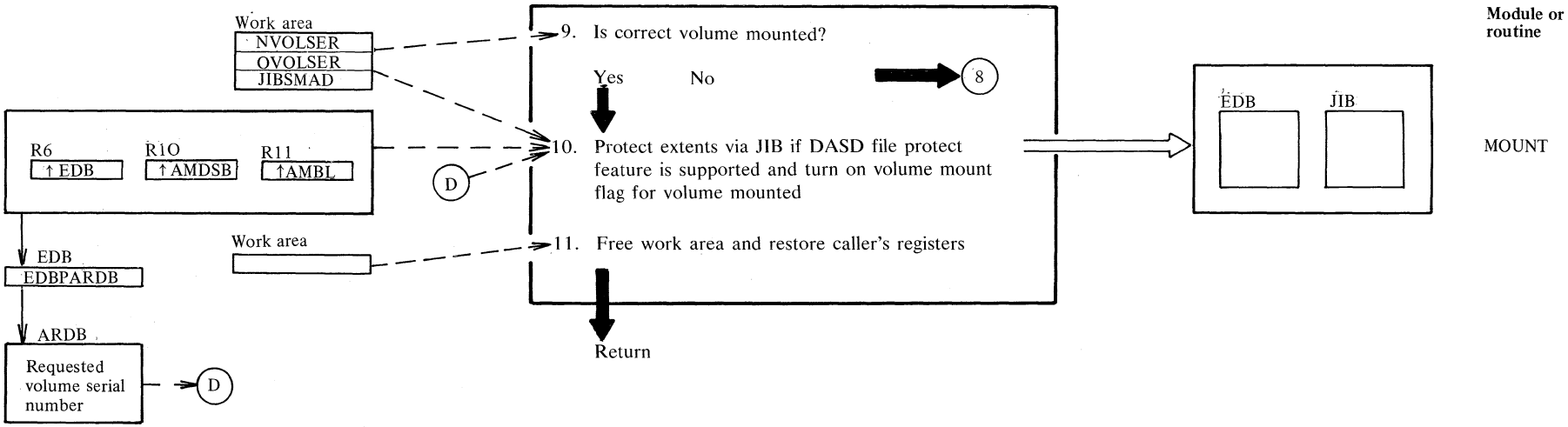


Diagram HS1. Extend EDB

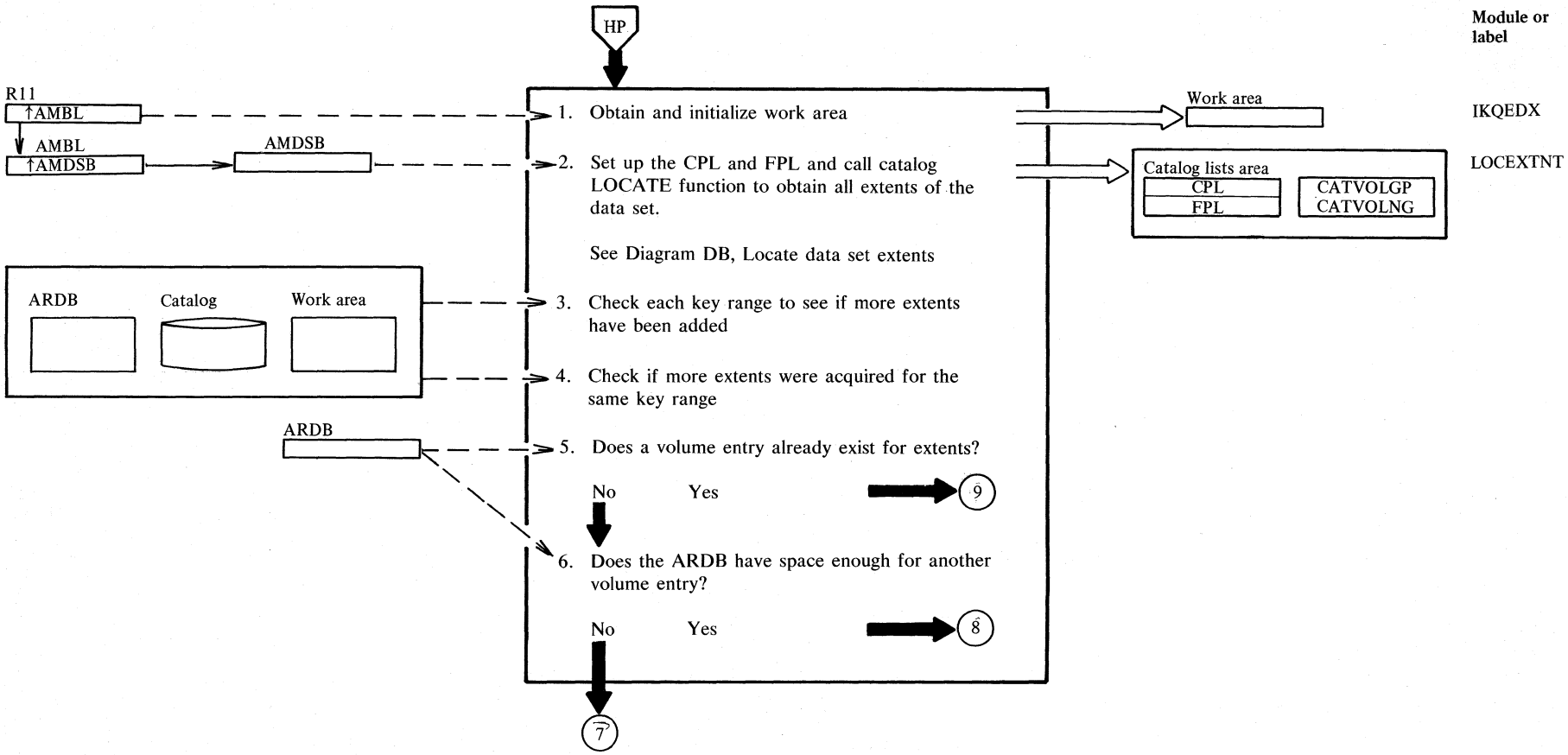
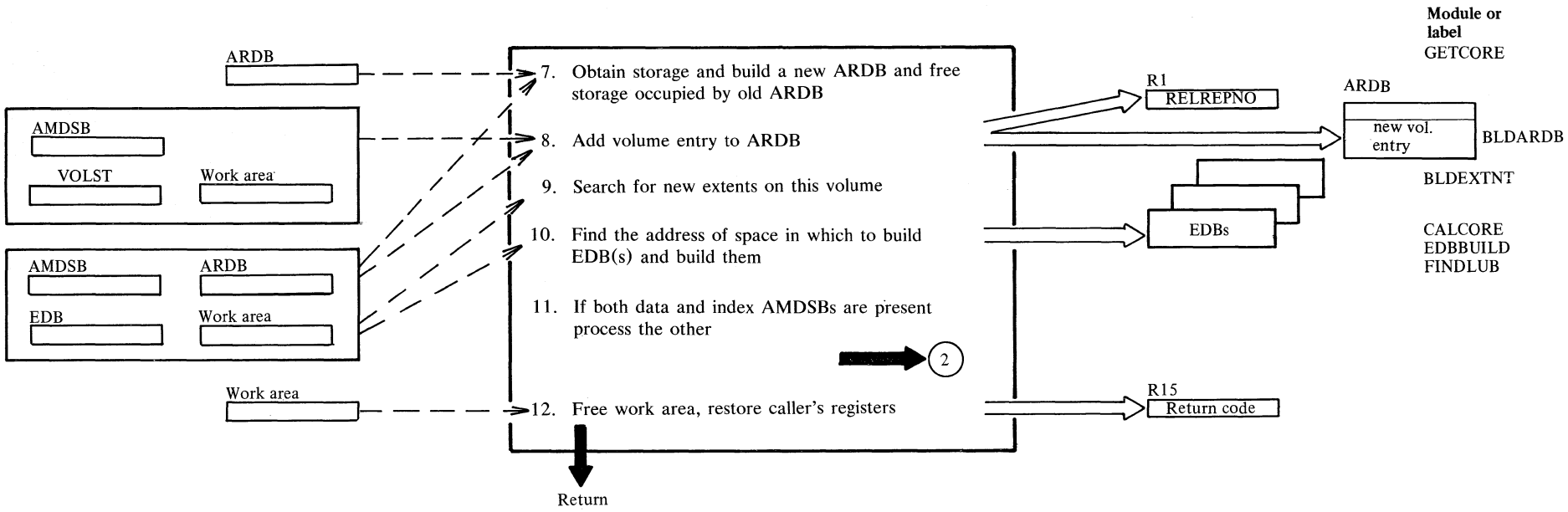


Diagram HS2. Extend EDB



Notes for Diagram HS

	Description	Module	Routine
2	LOCEXTNT is called and, in turn, calls the catalog LOCATE function to obtain all extents of the data set. If a nonzero code is returned in register 15 by the catalog LOCATE routine, an error code is set in the work area by LOCEXTNT to specify what error condition occurred before exiting to the mainline routine.	IKQEDX	LOCEXTNT
3-4	Each of these steps constitutes a small loop within the major loop (i.e., steps 2 through 11 are processed twice if both data and index AMDSBs exist). After each of the checks in steps 3 and 4, steps 5 through 10 are processed. That is, whenever extents are found that belong to the key range being processed, associated ARDBs, EDBs, and volume entries must be created for them, if they do not already exist.	IKQEDX	
7	If there isn't enough space in the ARDB to add a volume entry, GETCORE is called to obtain storage, copy the old ARDB, and free the storage occupied by the old ARDB.	IKQEDX	GETCORE
10	To locate and build EDB(s) for the new extents, BLDEXTNT is called and, in turn, calls CALCORE to get the address of space in which to build the EDBs, EDBBUILD to build the EDBs for the new extents, and FINDLUB to find the LUB index in order to turn on the mount flag.	IKQEDX	BLDEXTNT CALCORE EDBBUILD FINDLUB

Diagram HT1. Purge buffer

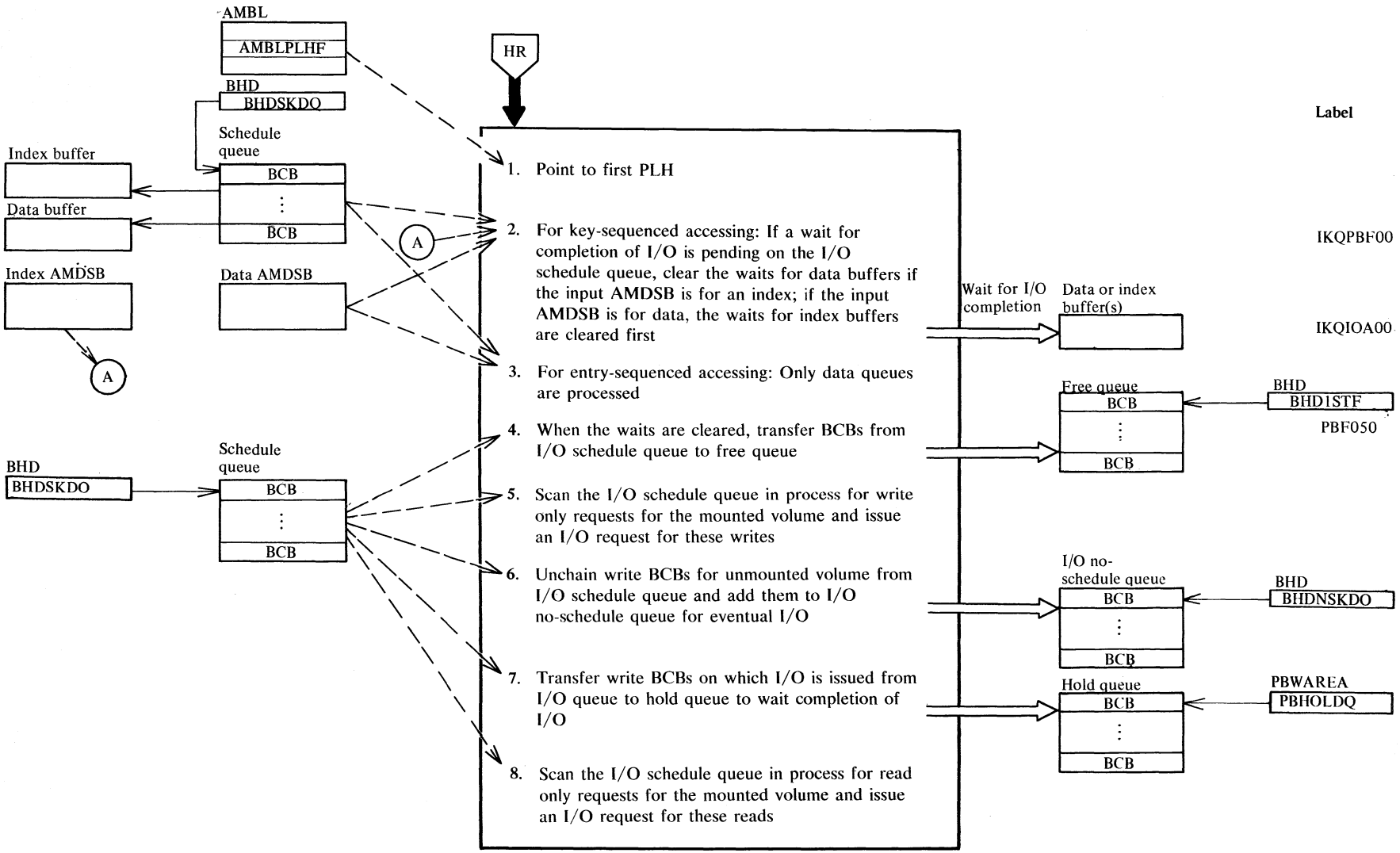
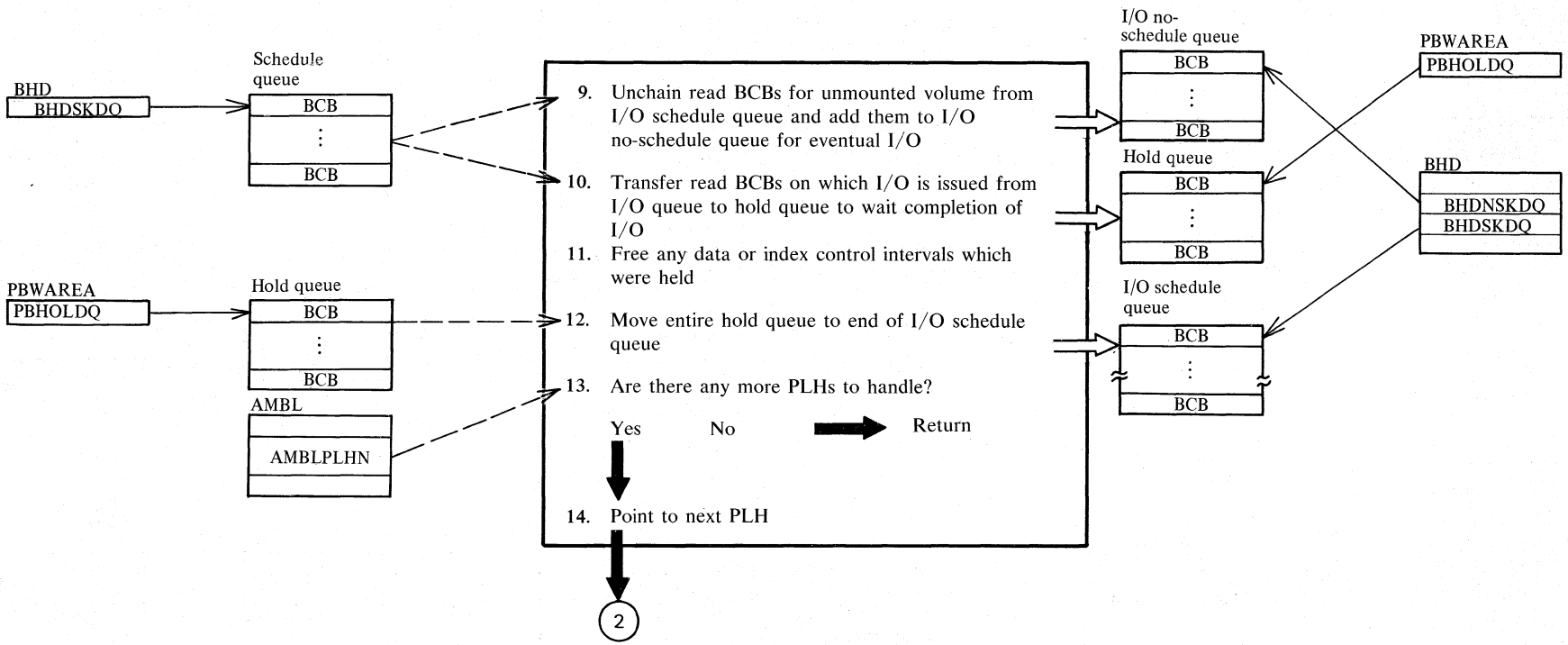


Diagram HT2. Purge buffer



Note: The hold queue is a temporary queue built and used by IKQPBF. The purge buffer work area PBWAREA contains a pointer to the first entry on the queue.

Diagram HU1. JRNAD exit: Journal a transaction

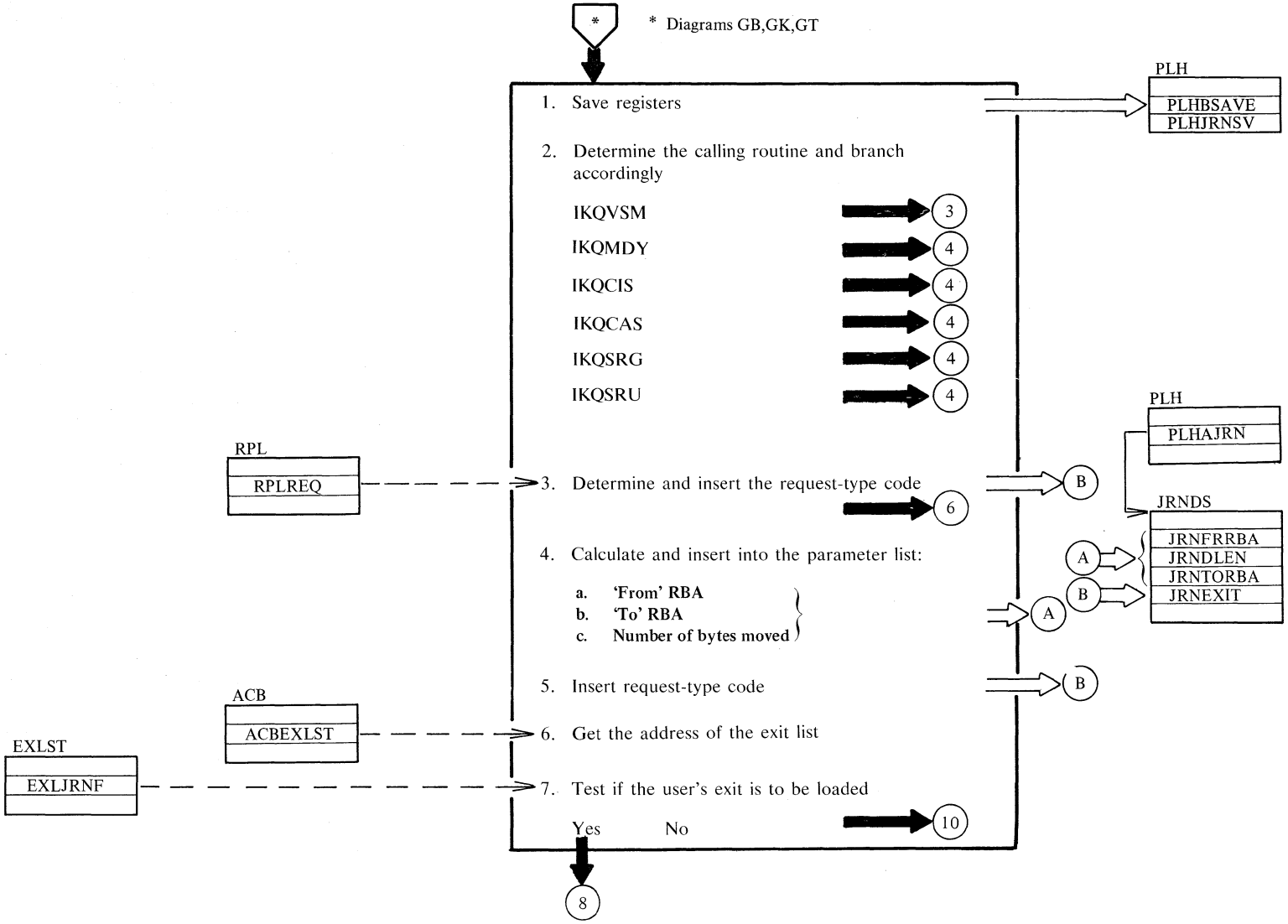
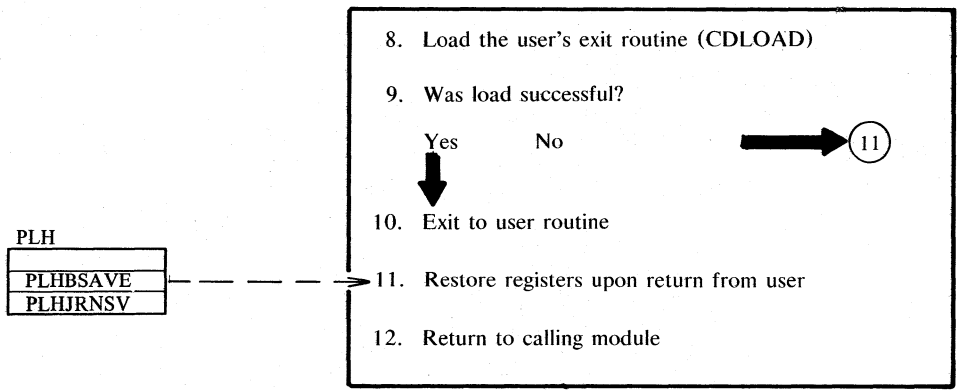


Diagram HU2. JRNAD exit: Journal a transaction



Module or label

JRN160

JRNRTN

Note for step 1: Registers 14, 0, and 1 are saved in field PLHJRNSV
Registers 2 - 12 are saved in field PLHBSAVE

Notes for Diagram HU

2. **IKQAIX also calls JRNAD, but it uses the IKQVSM calling interface.**

   Depending on the caller, the following parts of step 4 are executed:

IKQMDY	**4a, 4b, and 4c**
IKQCIS	**4a and 4b**
IKQCAS	**4a (first pass)**
	4b (second pass)
IKQSRG	**4a**
IKQSRU	**4c**

Diagram HV1. Defer writing of buffers

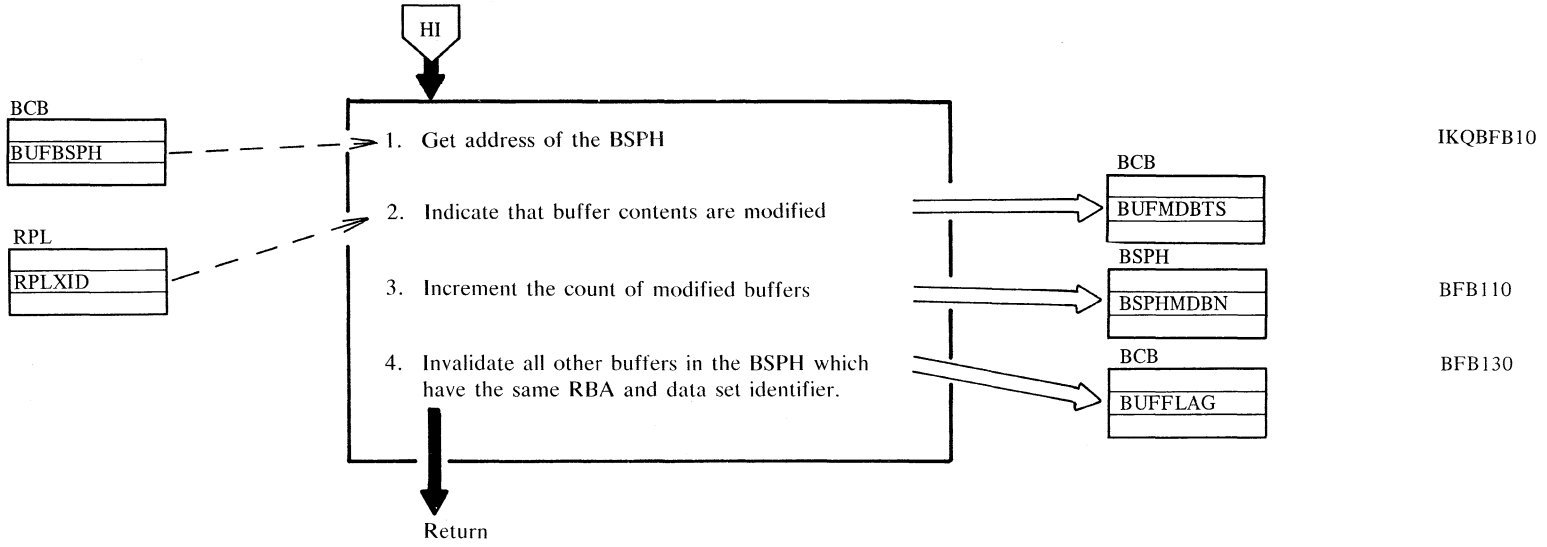


Diagram HW1. WRTBFR: Write deferred buffers

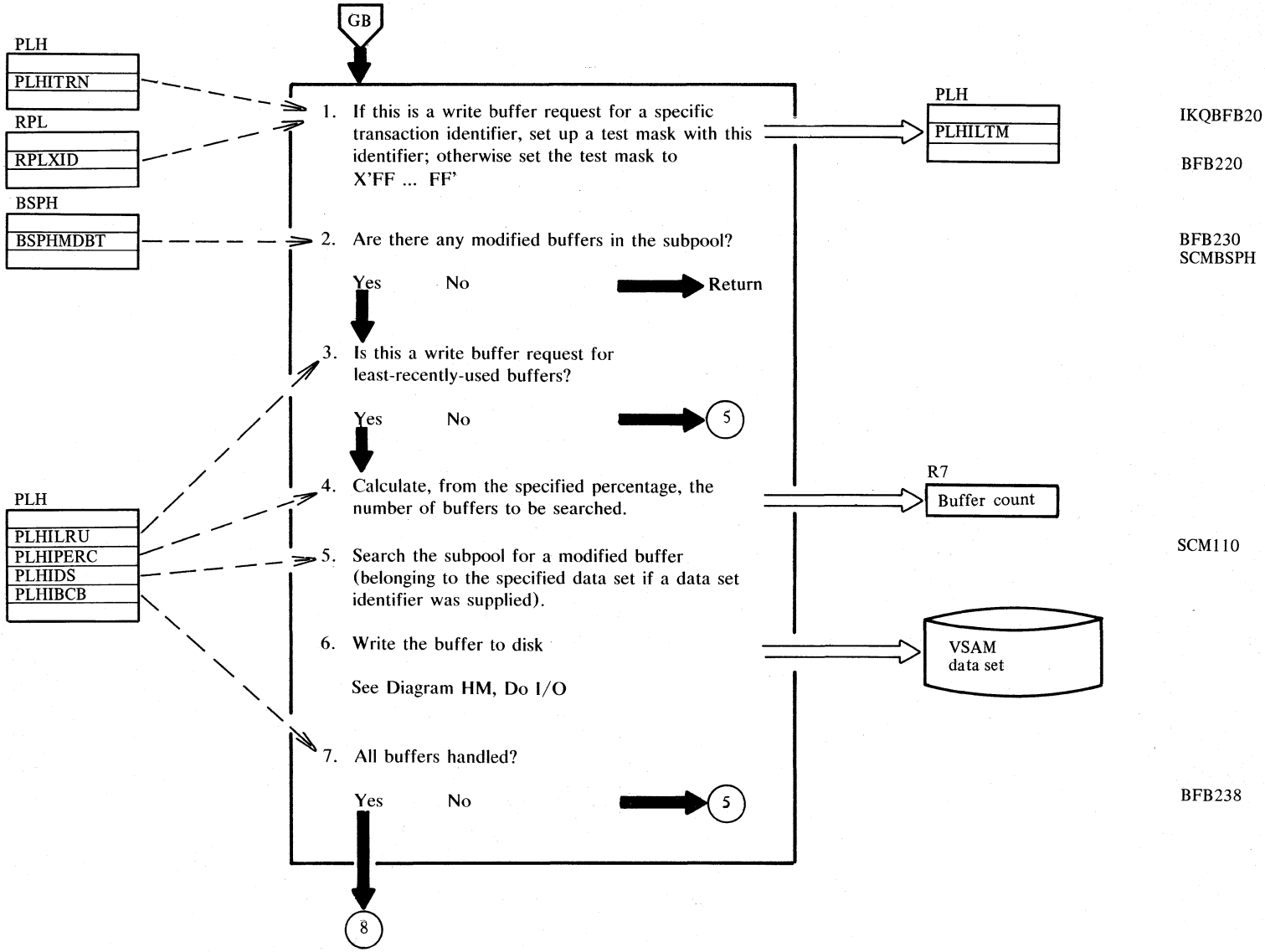
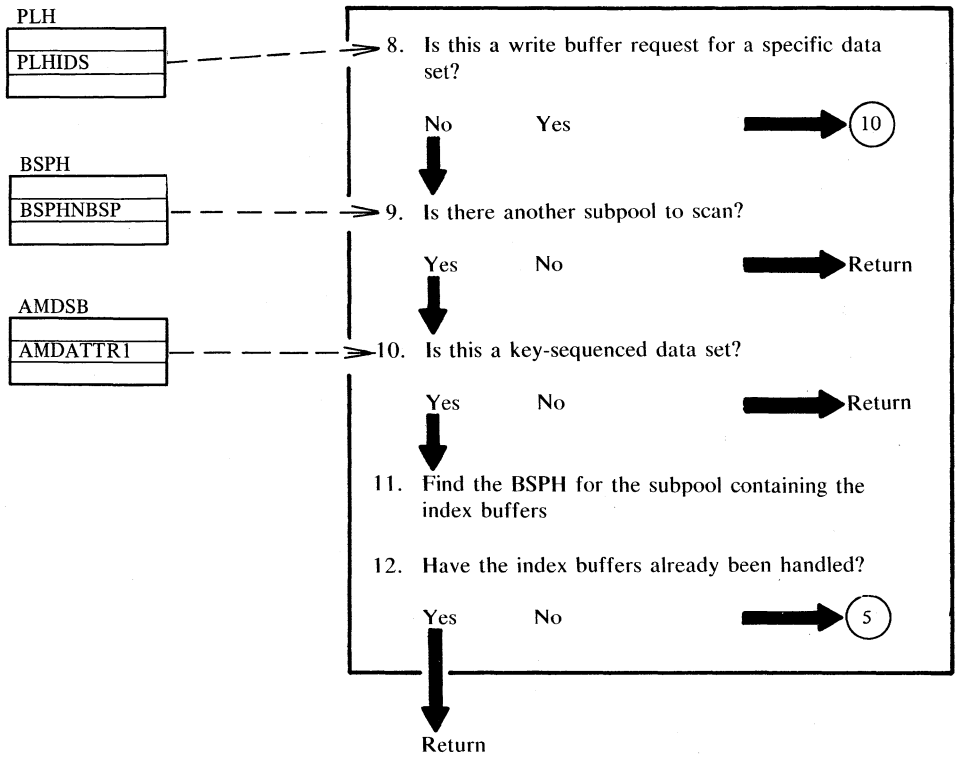


Diagram HW2. WRTBFR: Write deferred buffers



BFB240

Diagram HX1. Get a scratch buffer from the resource pool

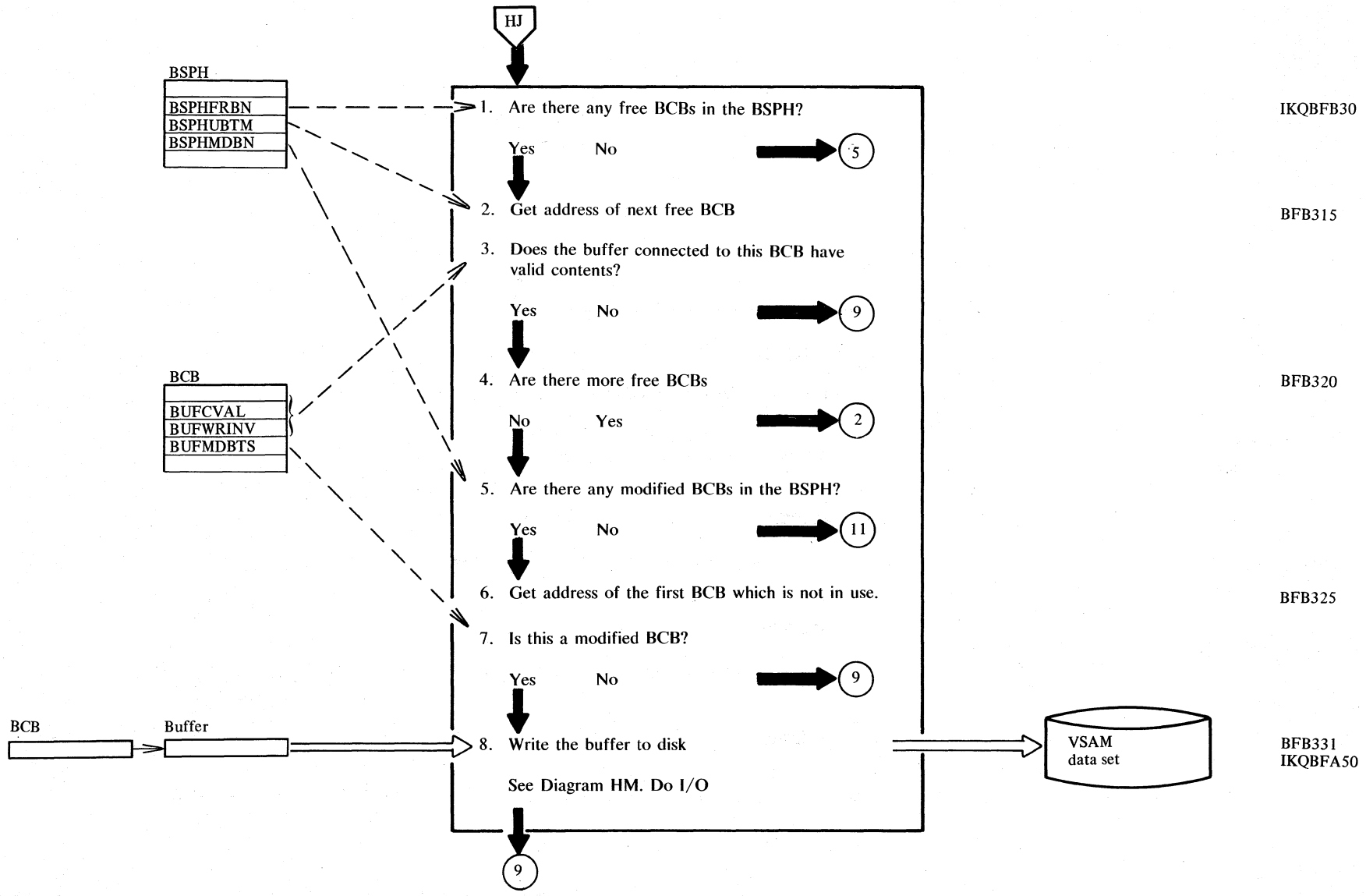


Diagram HX2.Get a scratch buffer from the resource pool

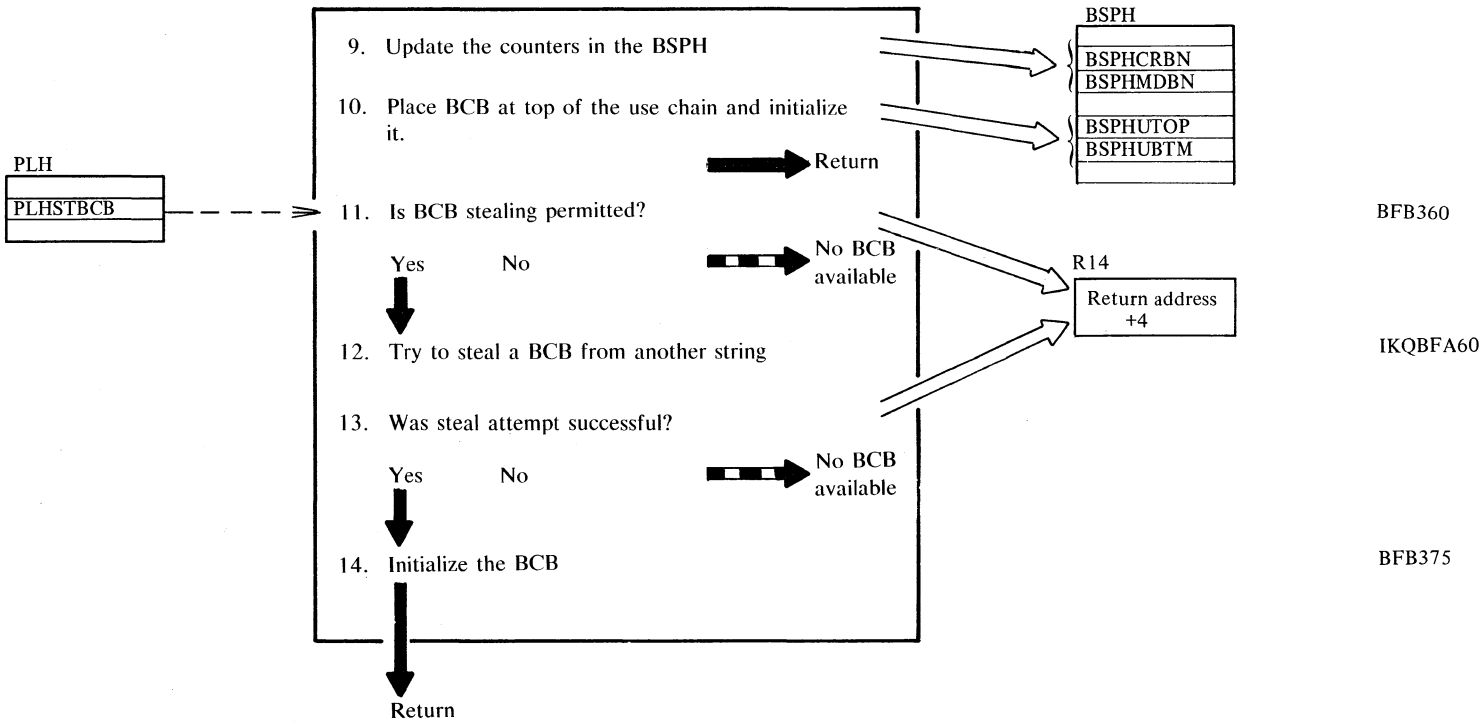


Diagram HY1. Return a buffer to the resource pool

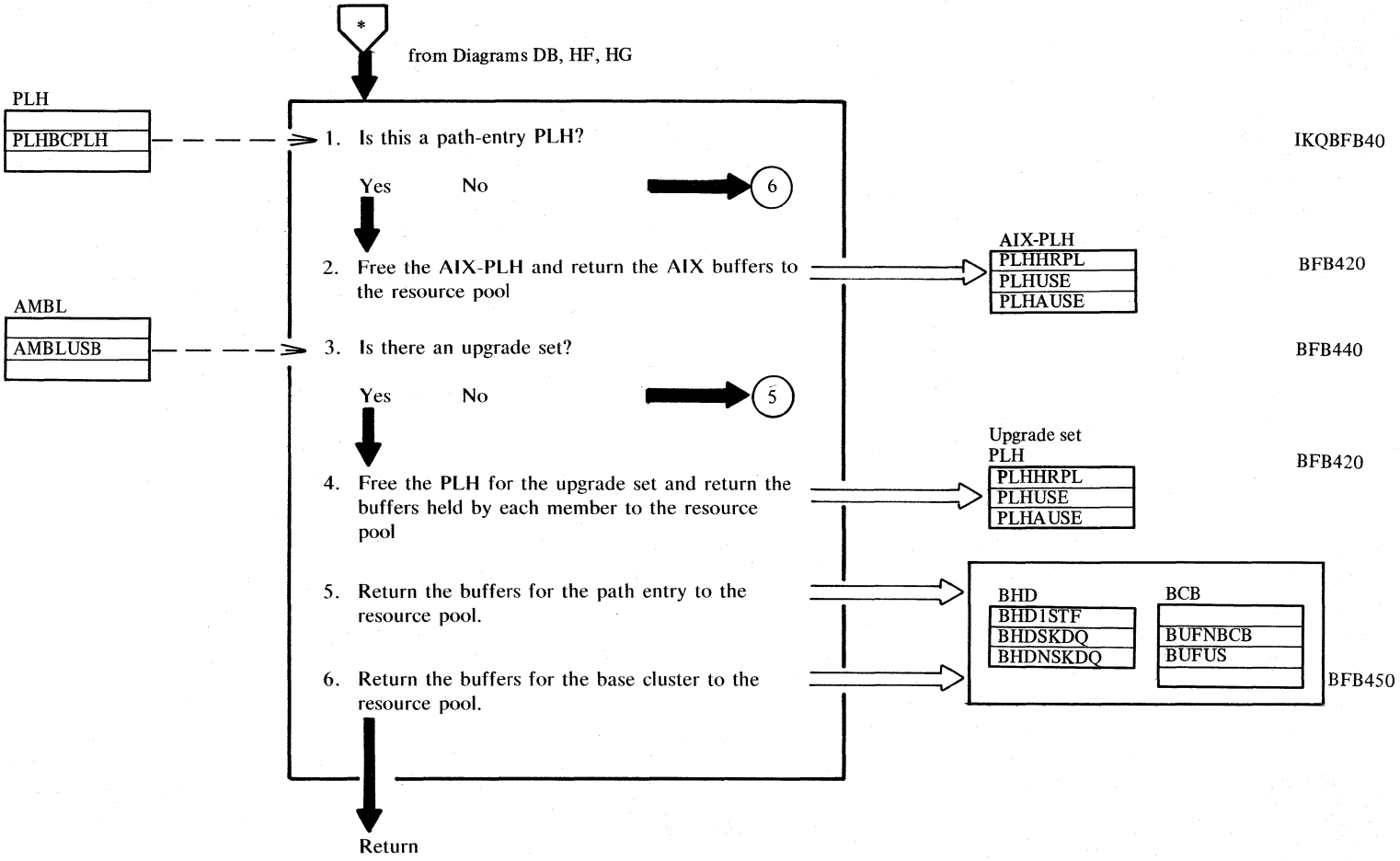


Diagram HZ1. Search resource pool for requested RBA

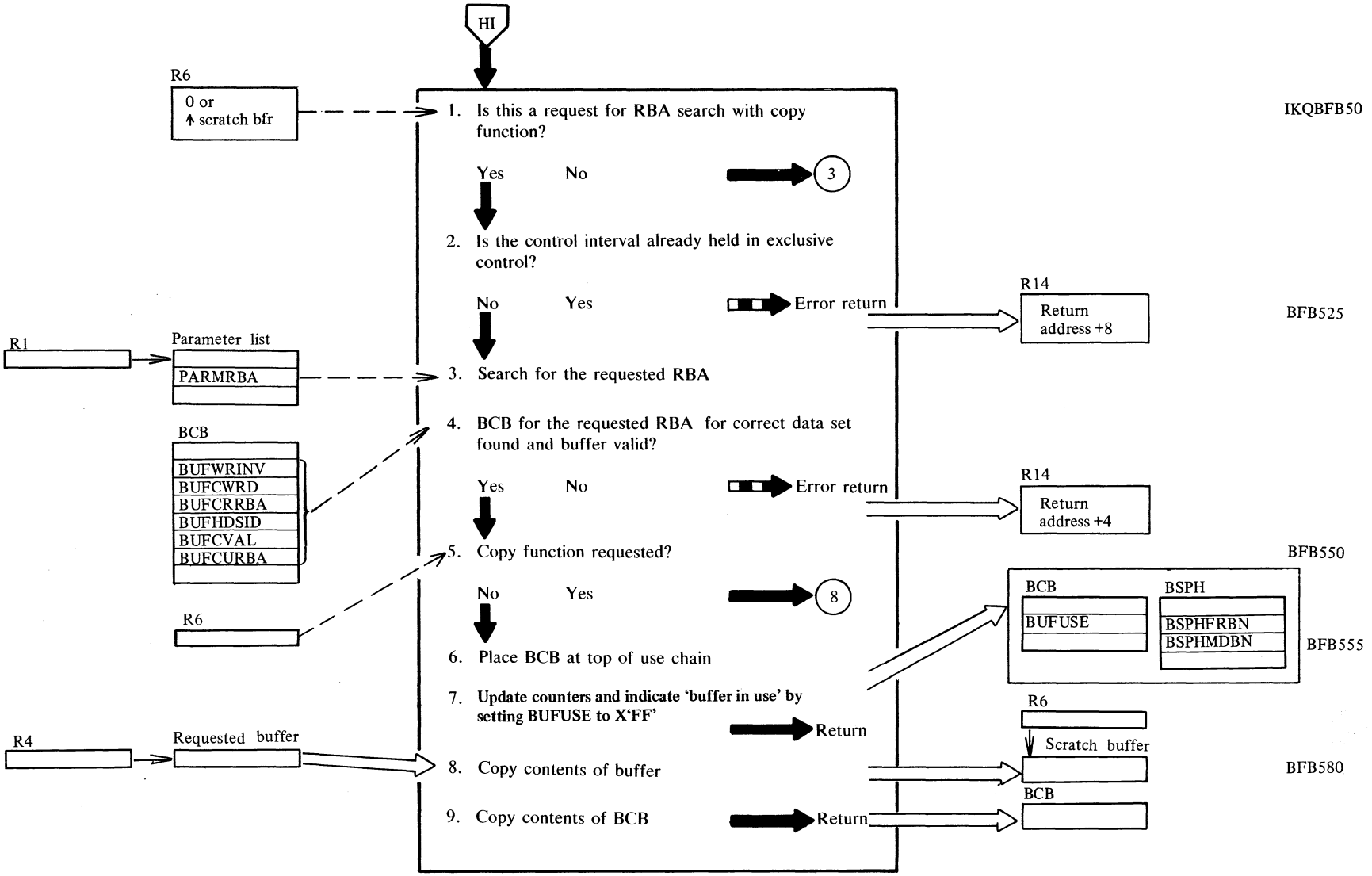


Diagram IA1. CLOSE or TCLOSE: Disconnect a user's program from a VSAM data set

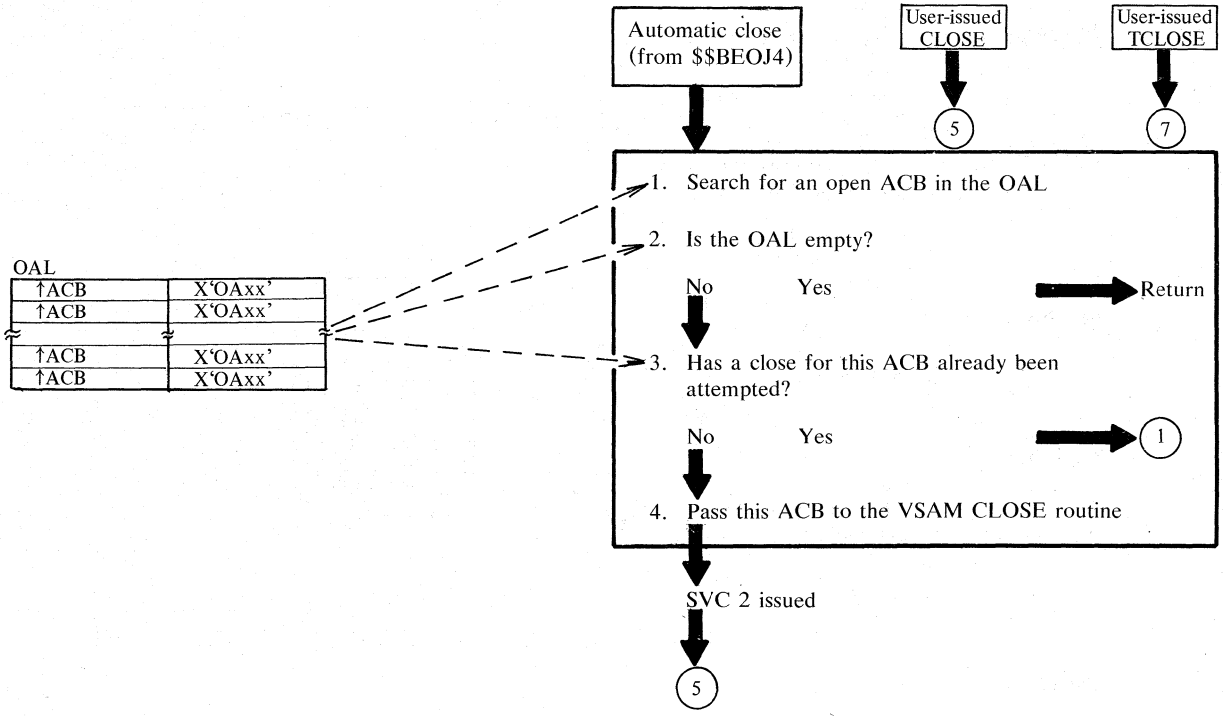


Diagram IA2. CLOSE or TCLOSE: Disconnect a user's program from a VSAM data set

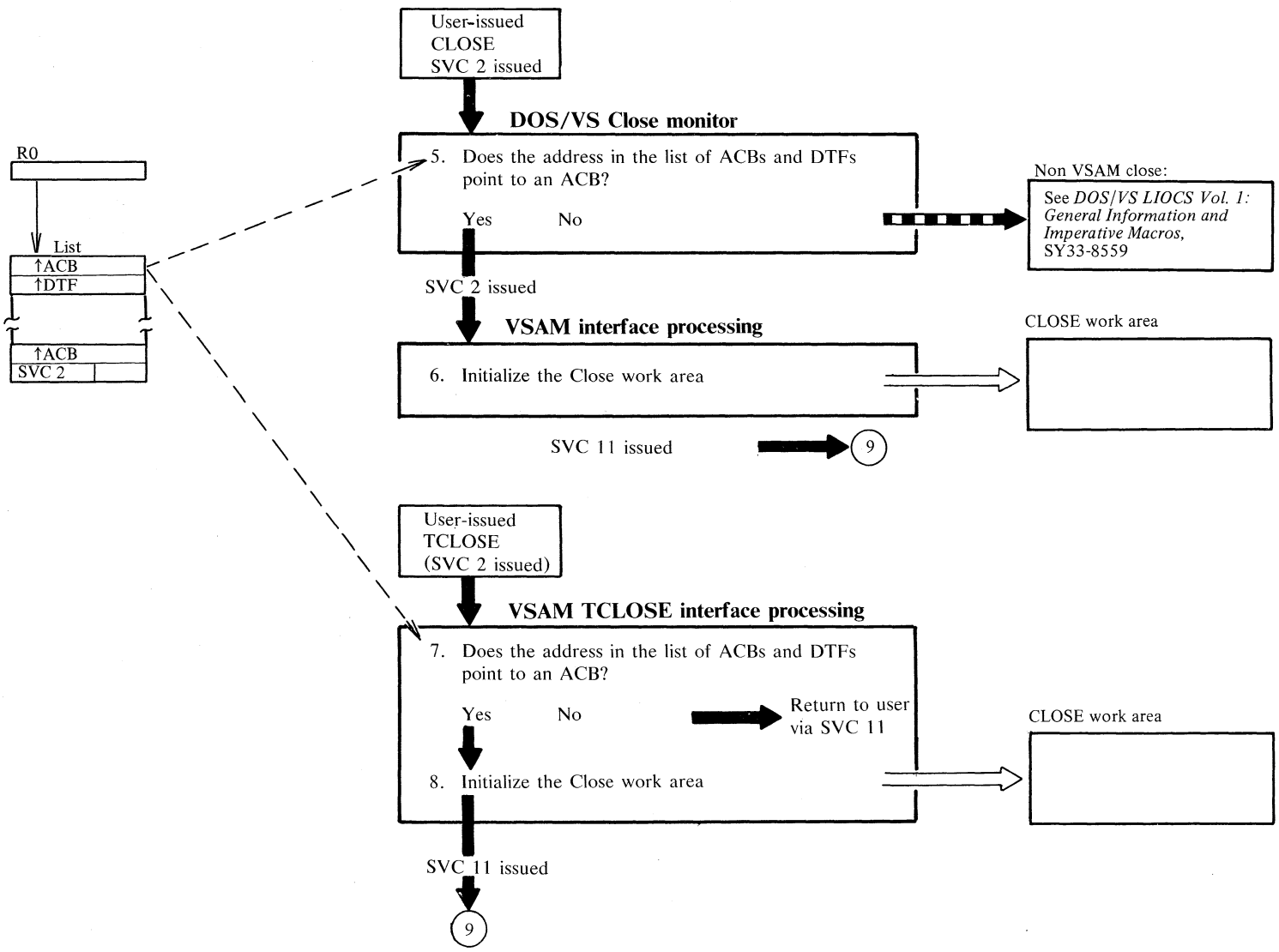
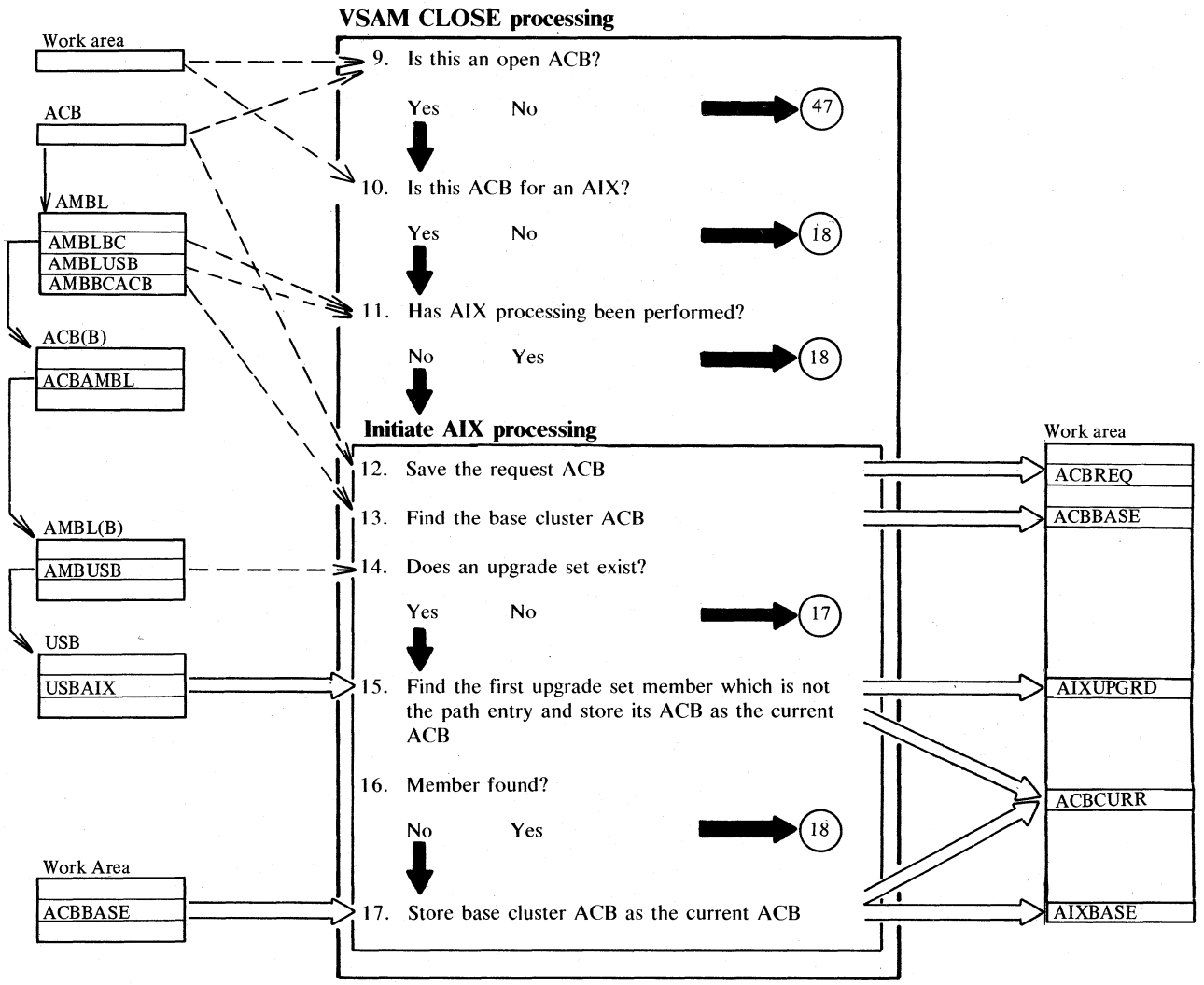


Diagram IA3. CLOSE or TCLOSE: Disconnect a user's program from a VSAM data set



IKQCLOVY

Diagram IA4. CLOSE or TCLOSE: Disconnect a user's program from a VSAM data set

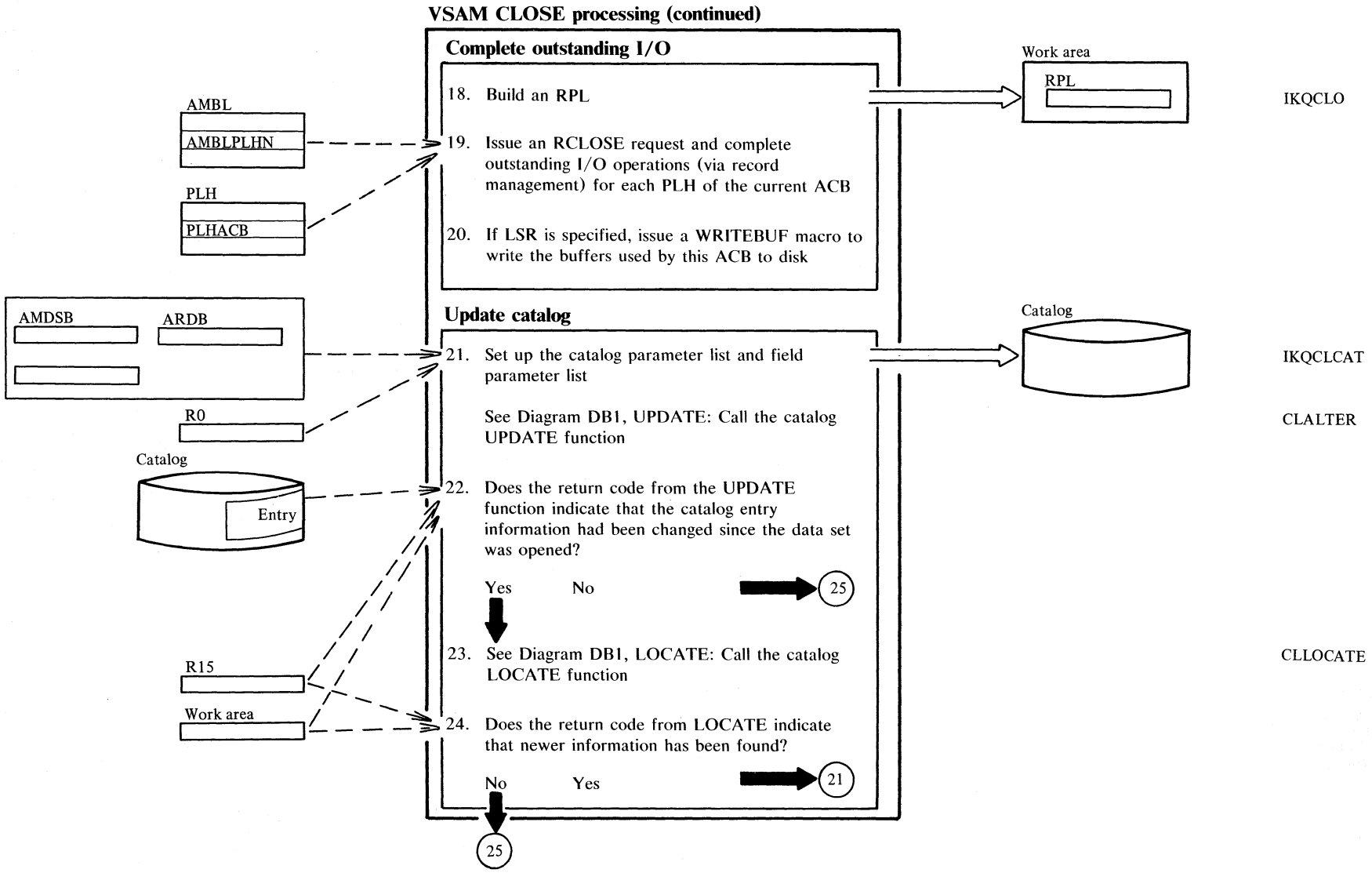


Diagram IA5. CLOSE or TCLOSE: Disconnect a user's program from a VSAM data set

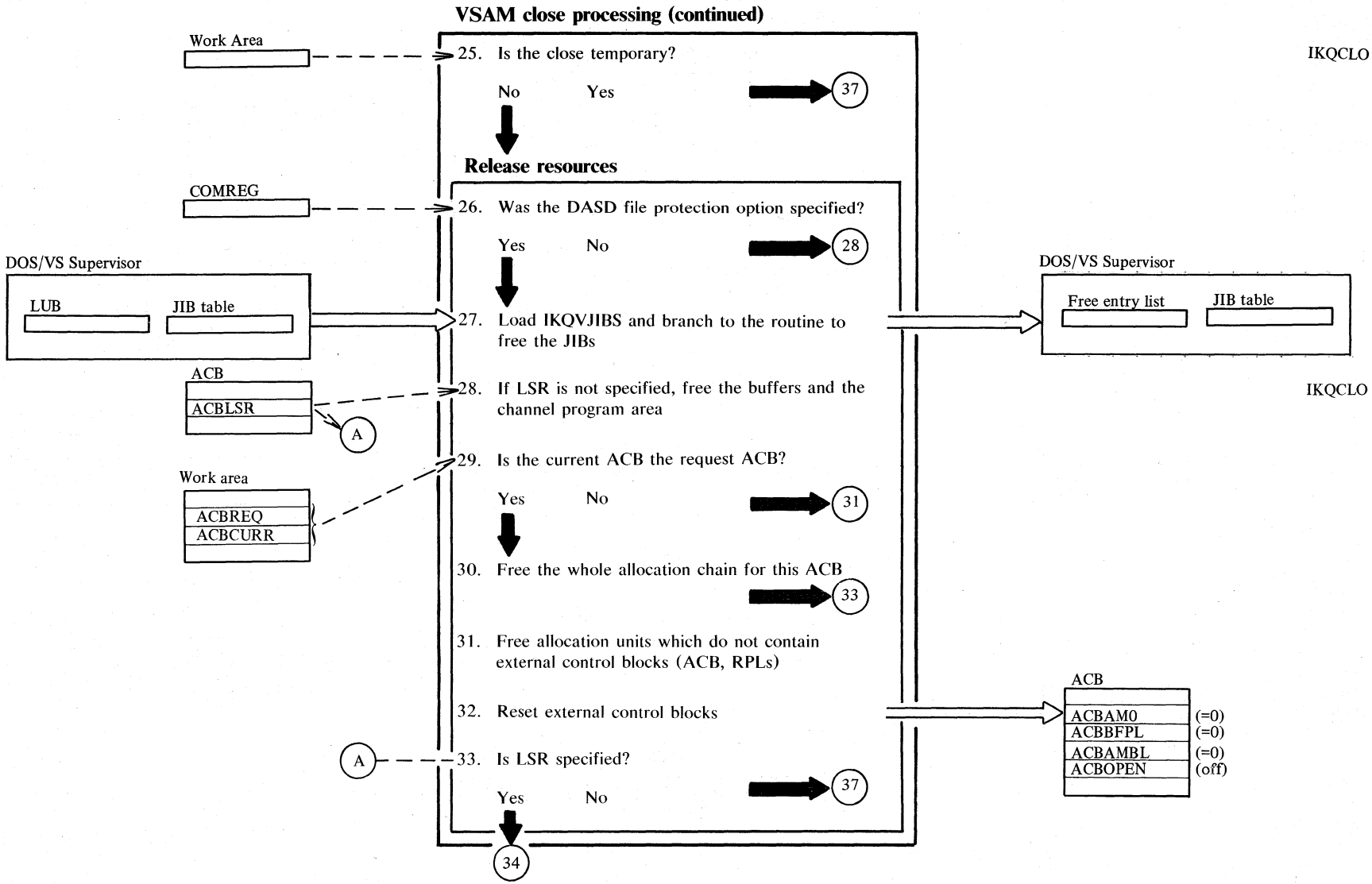
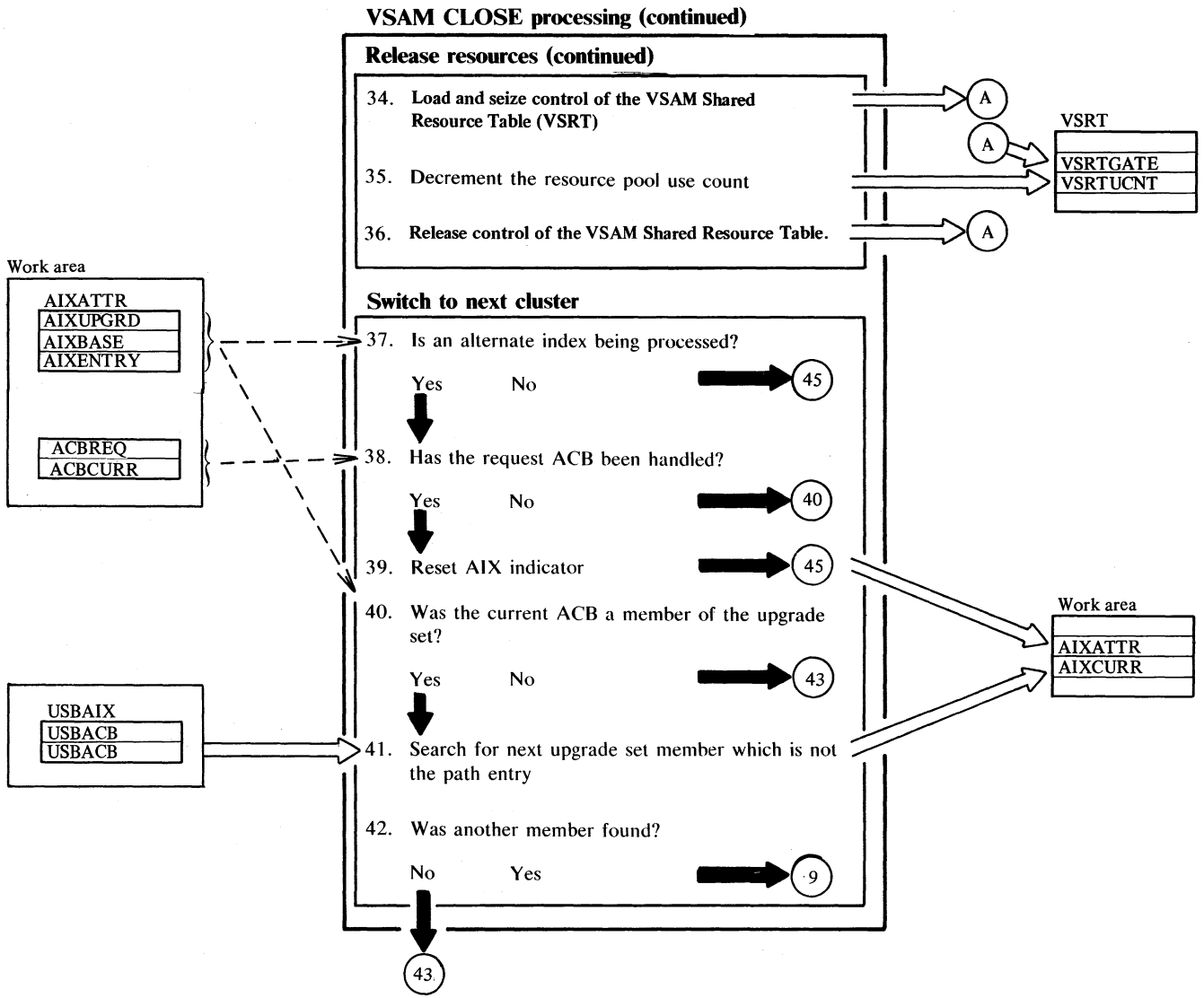
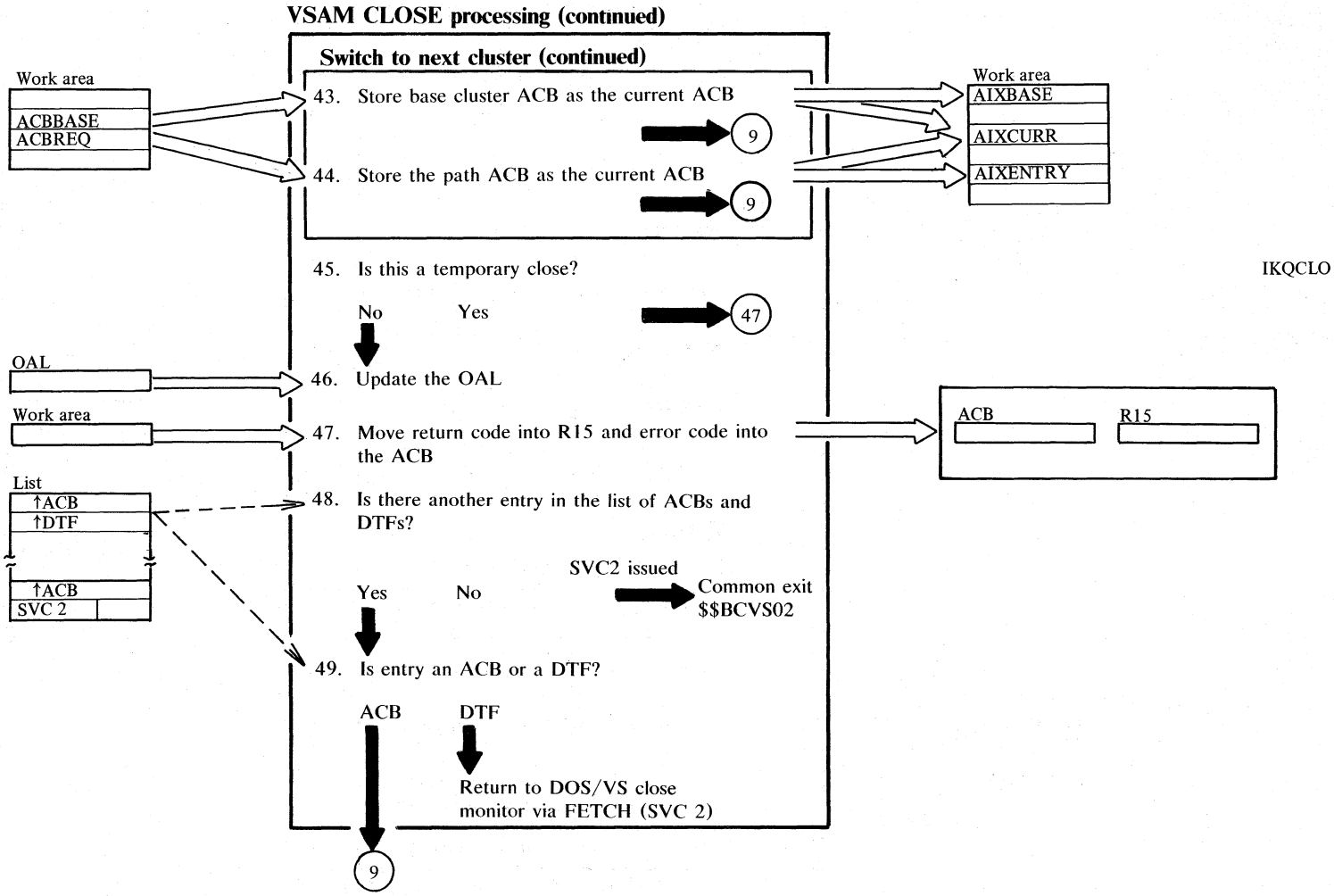


Diagram IA6. CLOSE or TCLOSE: Disconnect a user's program from a VSAM data set



IKQCLOCL

Diagram IA7. CLOSE or TCLOSE: Disconnect a user's program from a VSAM data set



Notes for Diagram IA (part 1 of 3)

	Description	Module	Routine
Note:	When the user program issues a CLOSE or TCLOSE macro against an ACB or a DTF, or the user program SYNAD routine issues a CLOSE macro against an ACB, or the DOS/VS end-of-job routines initiate an automatic close, an SVC2 is generated.		
1	$$BACLOS is called by $$BEOJ4 (DOS/VS end-of-job routine) to initiate automatic closing of ACBs which were not closed by the user's program.	$$BACLOS	
2	If the OAL is empty, control is returned to $$BEOJ4, which continues end-of-job processing.	$$BACLOS	
3	A flag in the OAL entry indicates whether an attempt has already been made to close this ACB. If this is the case, the ACB is skipped, to avoid recurring attempts to close the same ACB, which would lead to a program loop.	$$BACLOS	
4	The ACB is passed to the close routine as an ACB list, containing only one ACB.	$$BACLOS	
5	The DOS/VS Close Monitor examines the DTF-type field (offset 20 from the address passed in the list) of the ACB or DTF. If the byte indicates an ACB (X'28'), an SVC 2 is issued and $$BCVSAM is fetched into the B-transient area. The list may consist of all DTFs, all ACBs, or a mixture. It is passed to the DOS/VS Close Monitor by the user program via a pointer in register 0.	$$BCLOSE	
6	The VSAM interface module obtains and initializes a work area in which it sets a flag to indicate a Close macro was issued. Pointers are saved to the current list entry, the user's save area and the DOS/VS communication region in the work area. It copies the user's PSW and registers into the work area. It loads the VSAM Close module and then issues an SVC 11 to branch to it.	$$BCVSAM	
7-8	If the user program issued a TCLOSE (temporary close) macro against an ACB, the temporary close module is fetched into the B-transient area.	$$BTCLOS	
	The VSAM TCLOSE interface module examines the DTF-type field (offset 20 of the address passed in the list) of the ACB or DTF. If the byte indicates an ACB (X'28'), this module obtains and initializes a work area in which it sets a flag to indicate a TCLOSE macro was issued. Pointers are saved to the current list entry, the user's save area and the DOS/VS communication region in the work area. It copies the user's PSW and registers into the work area. It loads the VSAM Close module and then issues an SVC 11 to branch to it.		
9	The ACB identifier field is checked for an X'A0'. The ACB open flag is also checked. If the ID is incorrectly specified or the open flag is off, an error code is set in the work area.	IKQCLO	CHECKACB
18	BUFMGT issues an RCLOSE to IKQVSM for each string. Dummy RPLs are built and passed by BUFMGT to IKQVSM, which then passes control to IKQRCL00.	IKQCLO IKQVSM	BUFMGT
	If load mode with speed option has been specified by the user's program (which means that a control interval is written to the disk only when the control interval has	IKQRCL00 IKQPFO00	

Notes for Diagram IA (part 2 of 3)

	Description	Module	Routine
	been filled with logical records), the remainder of the control area must be formatted and an SEOF record (all 0s) placed at the end of the area.		
	For a key-range data set, IKQCIS00 formats all remaining unformatted key ranges.	IKQCIS00	
	Note: In recovery mode, each control area is formatted with empty control intervals and the control area is terminated with an SEOF prior to loading any records into the control area. Hence, in this case, it is not necessary to call IKQPFO00.		
	IKQBFA00 is always entered from IKQRCL00 to complete outstanding I/O requests. All current read operations are finished and all pending update or write operations are done. If an error occurs while I/O is being completed, IKQVSM returns a non-zero error code in register 15, and BUFMGT sets an error code in the work area.	IKQBFA00	
21	If R0 = 0, IKQCLCAT has been called by Close; if R0 = 4, it has been called by Record Management.	IKQCLCAT	
22	To update the permanent data set information in the catalog, Close utilizes the work area for the catalog parameter and field parameter list.	IKQCLCAT	CLCATLG CLALTER
22-24	If the information in the catalog entry had been changed since the data set had been opened (that is, between Open and Close another user had processed the data set and information had been altered), the latest copy of the entry must be located. If the data set has an index, repeat steps 7-20 for the index.	IKQCLCAT	CLCATLG CLLOCATE
25	A temporary Close does not release the JIB extents.	IKQCLO	
27	If the DASD file-protection option has been specified, the JIB entries for each extent of the data set are removed from the JIB chain and returned to the free entry list. The IKQJIBSM routine is called to release the extents. This processing is repeated for all extents. If the data set has an index, this step processes the index also.	IKQJIBSM	
28	Storage obtained by Open and/or end-of-volume for LPMBs (other than the first LPMB), EDBs (other than the first EDB), ARDBs, BCBs, and buffers is released. If the data set has an index, this step processes the index also.	IKQCLO	RELSCORE
29-31	Allocation units containing the user's external control blocks may not be released. The whole allocation chain may be released if the first allocation unit contains only internal control blocks ('AMBL allocation chain') or if the external control blocks were created by OPEN (base cluster ACB, upgrade set members ACBs, etc.).		
46	The OAL entry for this data set is set 'inactive' and the count of open ACBs in the OAL is decremented.	IKQCLO	

Notes for Diagram IA (part 3 of 3)

	Description	Module
47	The work area error code field is checked and the error return code is moved into the ACB error flag field. An error indicator is also set in register 15 of the user's register save area.	IKQCLO
48	If there are no more entries in the list, control is passed to $$BCV02, the VSAM common exit module, via an SVC 2.	
49	For a normal (user-issued) CLOSE or TCLOSE, $$BCVS02 returns control to the user. For an automatic close, it returns control to $$BACLOS.	

Section 3. Program Organization

VSAM program listings are the key to VSAM's organization. You get into the listings from the method of operation diagrams. Once you have located the module or routine name that interests you in the diagrams, you are ready to turn to the listing to find the additional information you require.

Module Prologues

Each VSAM module listing begins with a description of the module, called the module prologue. The information contained in VSAM prologues is described in the topics that follow.

Module name: The external procedure name of the module (for example, IKQOPN).

Descriptive name: The English name of the module (for example, VSAM Open).

Status: The version and release level of the module.

Function: A brief step-by-step explanation of the functions performed by this module. Function is divided into steps so that you may more easily locate the routine responsible for each step.

Notes: A generalized heading that includes (1) any dependencies, for example, CPU model or features, that will affect the operation of this module, (2) any restrictions that apply to this module, (3) symbols used to represent registers and register usage, (4) symbolic name of the maintenance area for this module and whether the maintenance area is used or reserved, and (5) any special terms and acronyms that are used within this module that are not necessarily used elsewhere in the documentation.

Module Type: A description of the type of this module (for example, procedure or macro) the name of the compiler used/required to create this module, the amount of storage required by this module for executable code and associated data, and the attributes of the module (for example, reentrant or read-only).

Entry point: The name of the point at which control can enter this module, the conditions of entry, the calling sequence by which control was given, including any parameters passed and the names of modules that may enter at this entry point.

Input: A description of anything this module gets or references, such as registers, control blocks, or data. The means by which this module gains access to the input is included.

Output: A description of registers, control blocks, and data areas at output; any messages issued as a result of this module's processing are included.

Exit-normal: A description of conditions at and reasons for normal exit from this module, including the names of modules called by this module.

Exit-error: A description of conditions at and reasons for any error exit from this module.

External references: A list of modules, data areas, etc., defined outside of or accessible outside of this module.

Tables: A list of all local tables and work areas, that is, data areas built and used only within this module.

Macros: A description of system macros used by this module.

Change activity: A list of any change activity to this module.

Routine Prologues

The numbered steps in the module prologue FUNCTION heading are your link to the routine prologues. Routine prologues contain (1) an expanded description of the processing steps shown in the module prologues, (2) input to the routine, and (3) output from the routine.

Program Structures and Catalog Program Flowcharts

The following group of program structures and, for the catalog modules, program flowcharts, shows how the VSAM program is organized. These structures link modules together from the time a macro instruction is issued by the user program to the time that control exits from VSAM. The structures are ordered by user-issued macro instructions and the verify function in a way similar to the organization of method of operation diagrams. In addition, program structures are also shown for significant subfunctions required to complete processing of a macro instruction. These subfunctions are the ISAM interface, catalog management, DADSM, and buffer and I/O management.

The flowcharts are arranged in alphabetical order according to the last two alphameric characters of the module name. The title of the flowchart also has a number, in the third position, which is the page number within the flowchart. Module IGG0CLAF is thus shown on two pages — Chart AF1 and Chart AF2. Off-page connectors between the pages contain the page number and the block location. For example, the off-page connector at block CI in chart AF1 contains "02 J1", which refers to block J1 on chart AF2.

As the flowcharts are intended to show the calling sequence rather than the internal logic, not all procedures are documented. Only those procedures which call other procedures are shown. This leads to two different types of cross-references in the flowcharts. Look, for example, at Chart AF2, blocks H1 and H2. In H1, the procedure IGGPF4WR is called, with the cross-reference BU1A2. This means that the procedure is located in module IGG0CLBU and documented in Chart BU1, starting at block A2. In H2, in contrast, the cross-reference for procedure IGGPDLER is simply AF. This means that the procedure is located in module IGG0CLAF, but is not documented.

Figure 3.1 shows the symbols used on the structures and describes their meanings.

———	Indicates that a module is called and returns to calling module
———→	Indicates that a module does not return to calling module
- - - - -	Indicates that a module is called under certain conditions and then returns to calling module
- - - - - →	Indicates that a module is called under certain conditions and does not return to calling module
UPPER CASE	Indicates that a module is executed and calls one or more modules before returning
lower case	Indicates that a module is executed and then returns to the calling module

Figure 3.1 **Graphic symbols used in program structures**

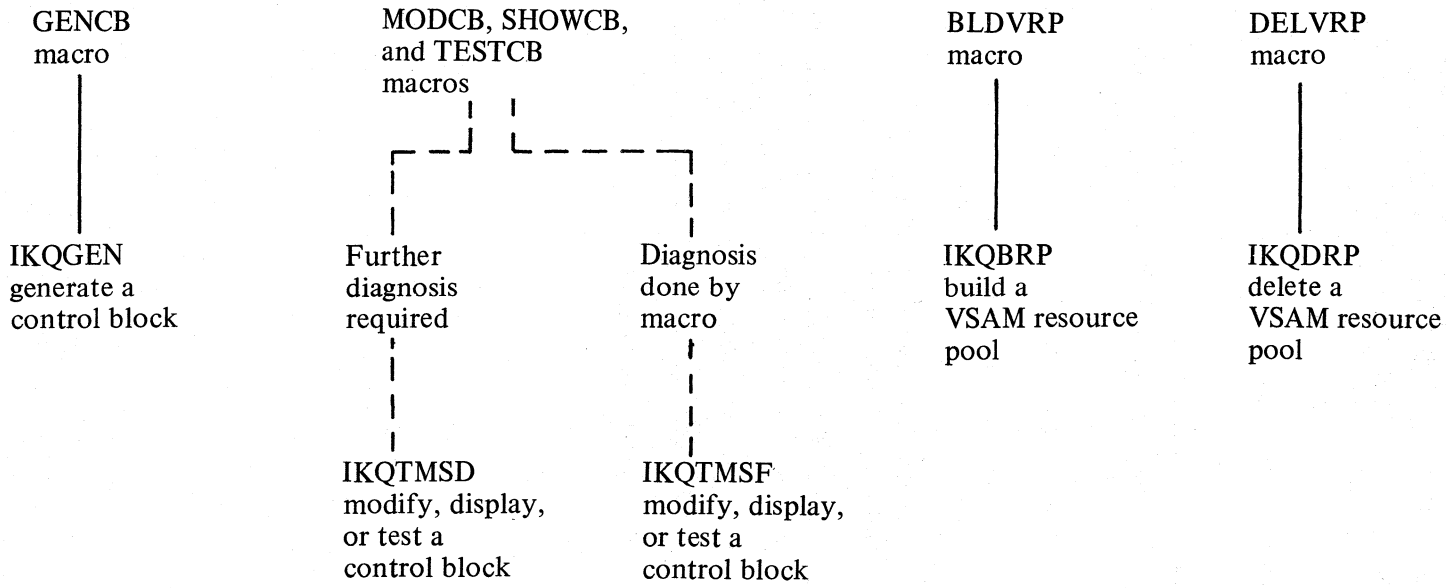


Figure 3.2 **Program structure to process control block manipulation macros**

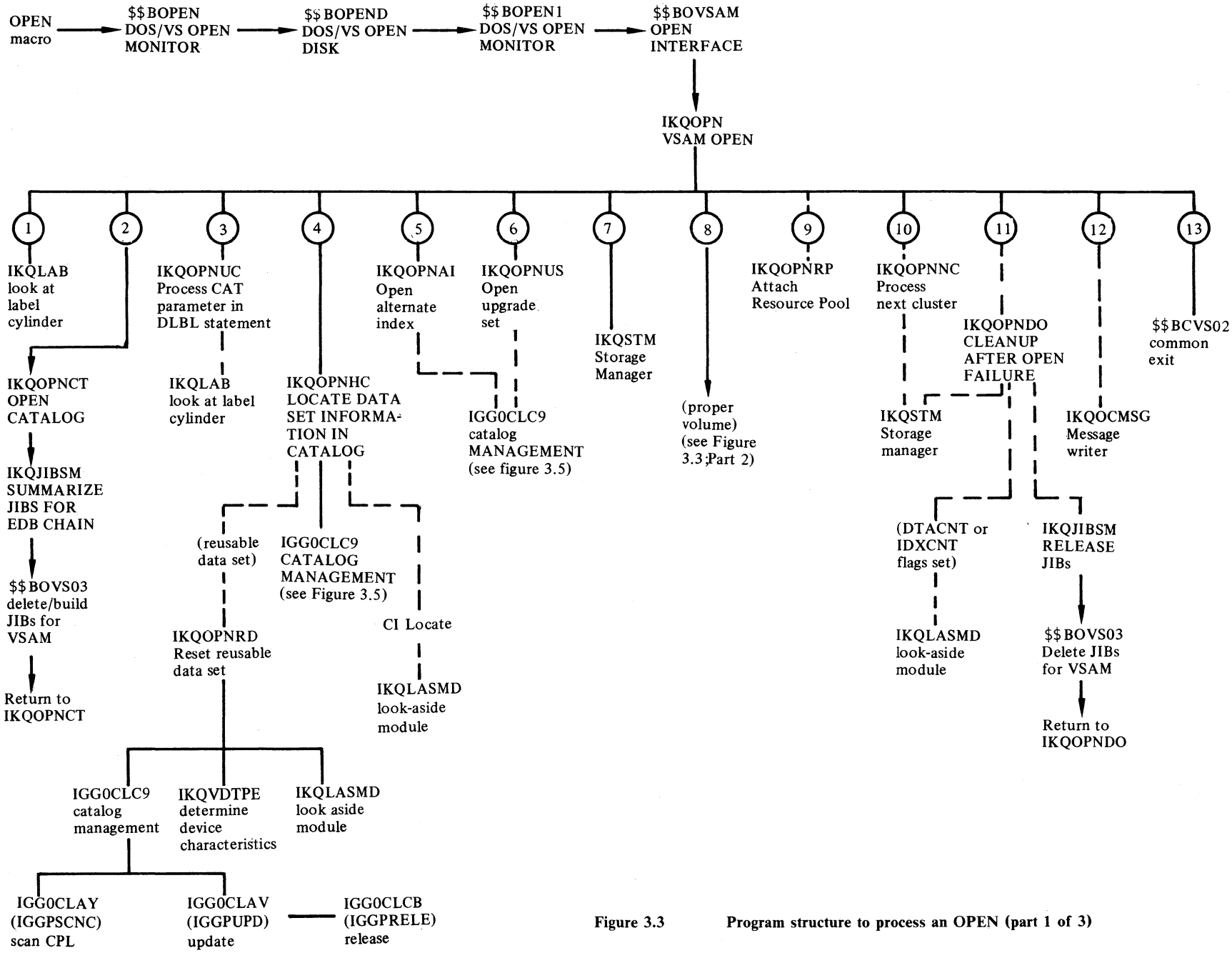


Figure 3.3 **Program structure to process an OPEN (part 1 of 3)**

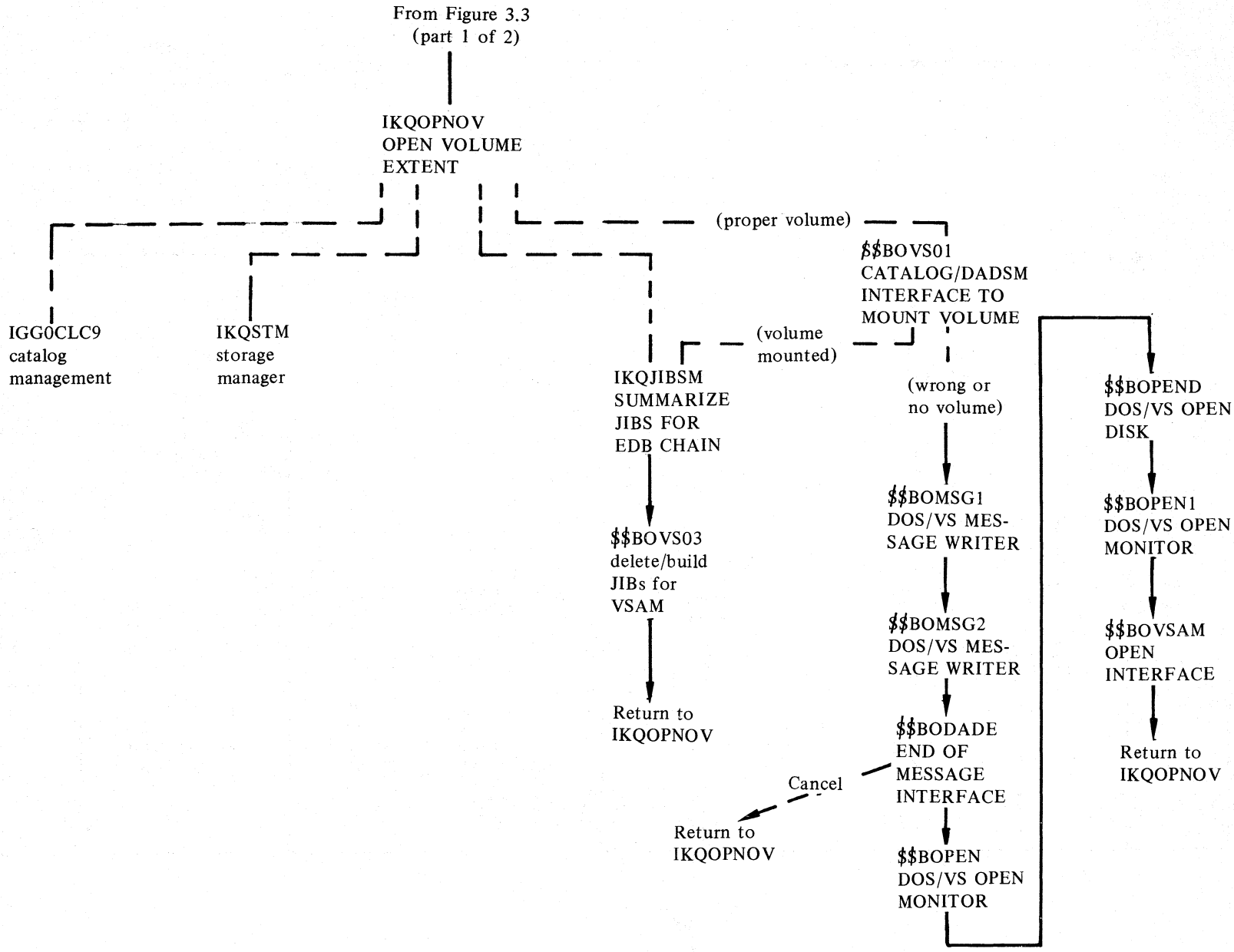


Figure 3.3 **Program structure to process an OPEN (part 2 of 3)**

(1) IKQLAB reads the label cylinder and establishes the connection between data set name and file name.

(2) IKQOPNCT is called if a catalog or a catalog recovery area is to be opened. IKQOPNCT finds in VTOC the address of the catalog cluster record or the CRA cluster record and reads it.

(3) IKQOPNCT checks if a user catalog is needed to open the cluster. It obtains user catalog label information via IKQLAB if required.

(4) IKQOPNHC is called to locate information in the catalog concerning the cluster to be opened.
It resets reusable data sets via IKQOPNRD.
It checks whether the cluster can be opened according to the sharing conditions via IKQLASMD

(5) IKQOPNAI is called if the cluster to be opened is an alternate index cluster
IKQOPNAI retrieves the cluster record of the base cluster identified by the AIX record from catalog.

(6) IKQOPNUS is called whenever a possible base cluster is processed.
IKQOPNUS retrieves information concerning the upgrade set from the catalog.

(7) IKQSTM is called to allocate VSAM record management control blocks.

(8) IKQOPNOV is called to process the cluster's extent information.
It retrieves the extent information from the catalog, obtains JIB's via IKQJIBSM, checks if the proper volumes are mounted, and and builds control blocks via IKQSTM.

(9) IKQOPNRP is called when resource sharing is requested. It attaches the cluster's control block structure to the resource pool.

(10) IKQOPNNC is called when an alternate index structure is to be opened.
It decides which cluster is to be opened next and creates ACB and RPL via IKQSTM.

(11) IKQOPNDO is called whenever an Open error occured. It reduces the open count via IKQLASMD.
It resets the open indication in catalog.
It releases the JIB's via IKQJIBSM.
It frees the allocated storage via IKQSTM.

(12) IKQOCMSG is called whenever a message has to be written.

(13) $$BCVS02 returns control to the user.

Figure 3.3 Program structure to process an OPEN (part 3 of 3)

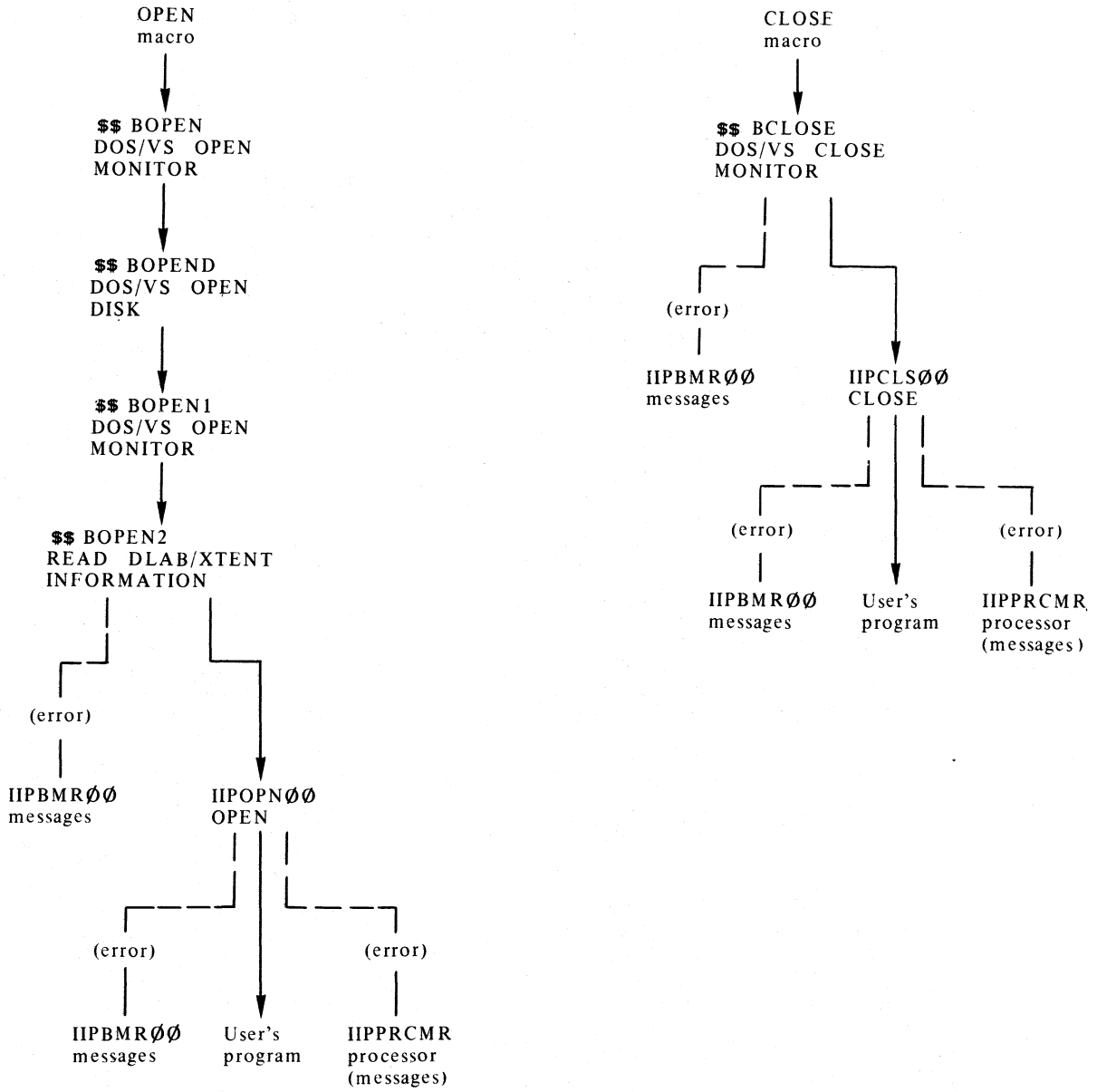


Figure 3.4 **Program structure to process ISAM interface macros (part 1 of 2)**

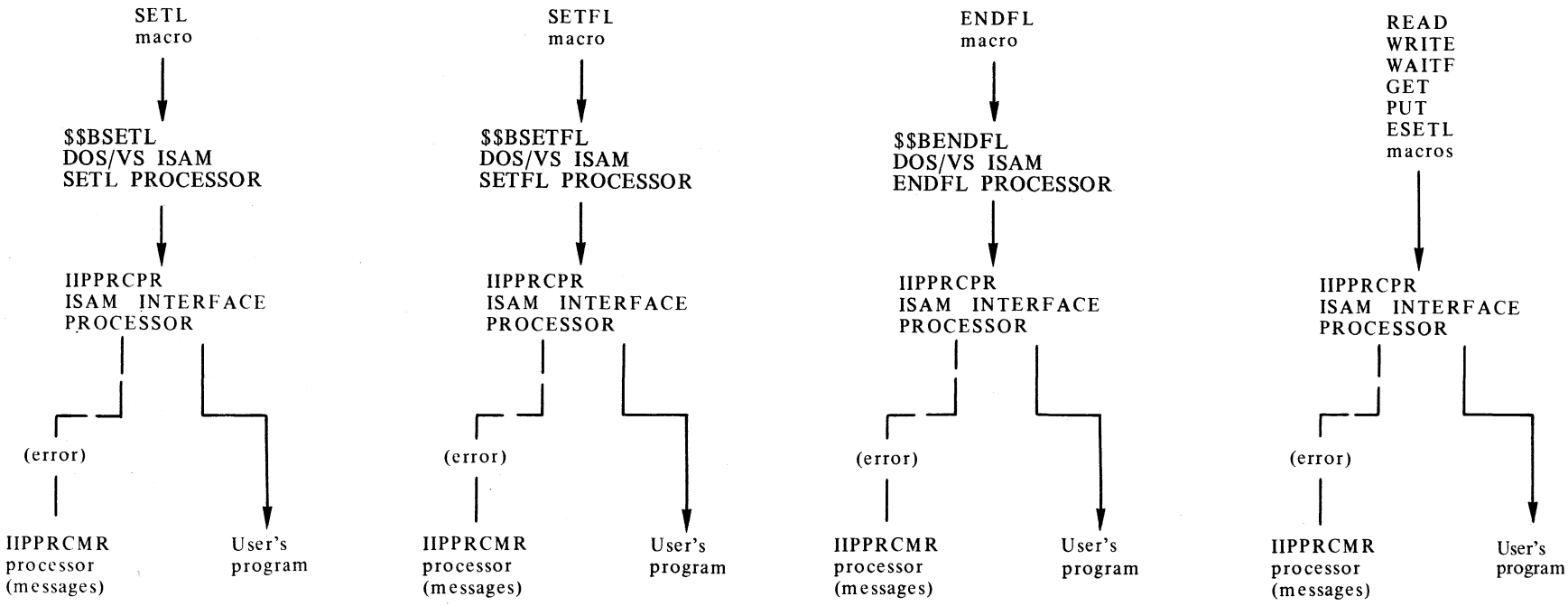


Figure 3.4 **Program structure to process ISAM interface macros (part 2 of 2)**

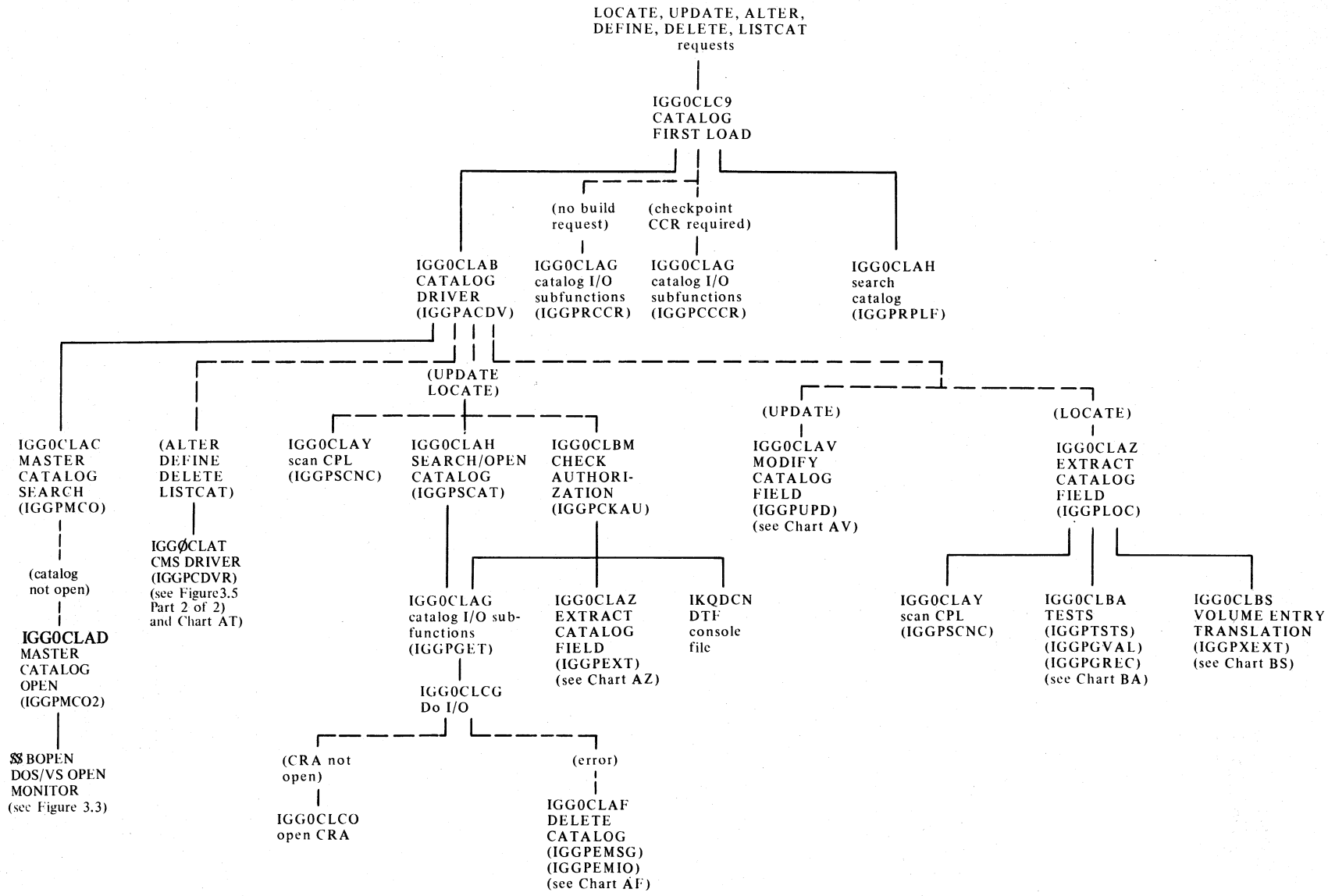


Figure 3.5 **Program structure to process catalog management requests (part 1 of 2)**

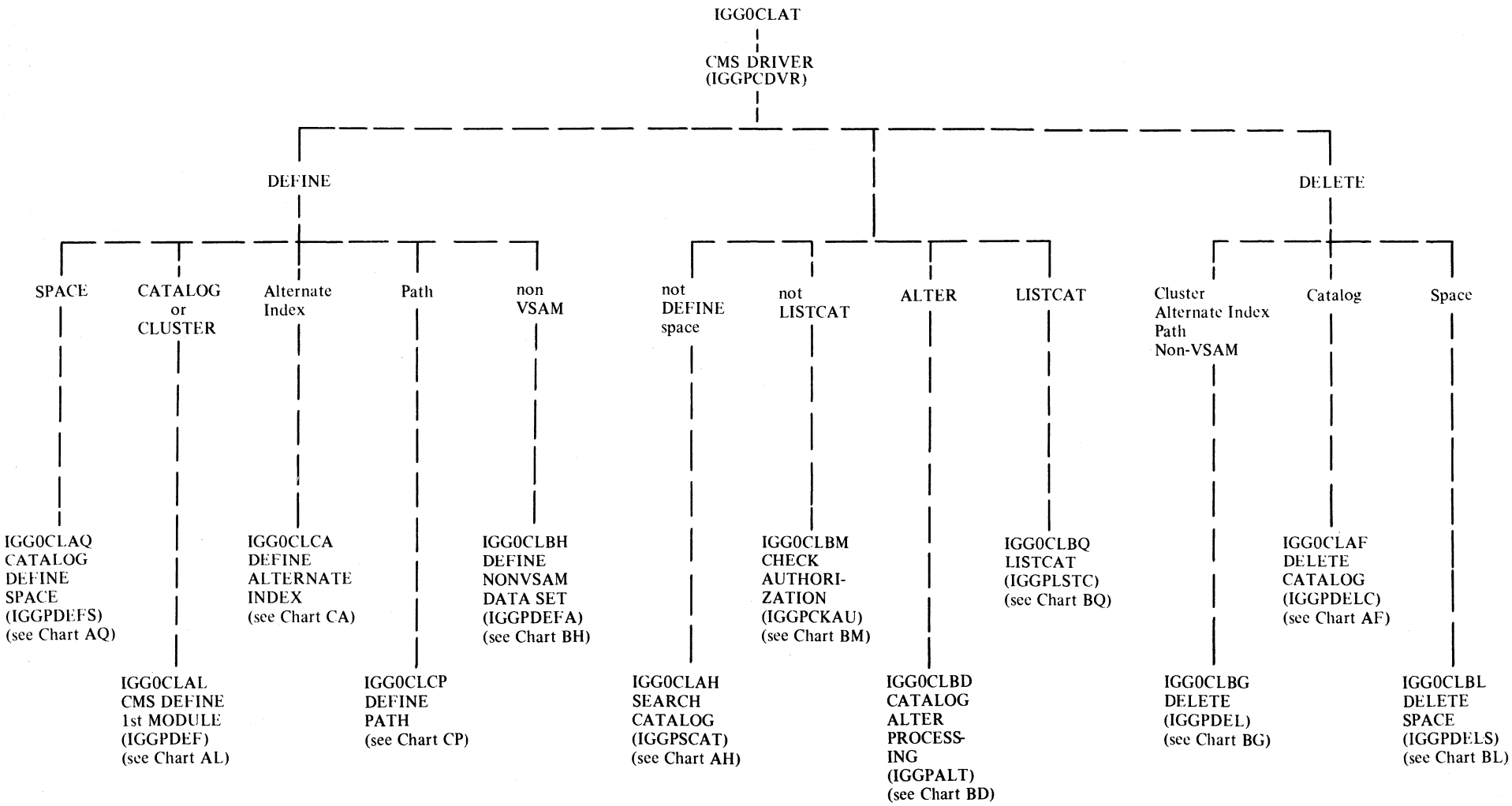


Figure 3.5 **Program structure to process catalog management requests (part 2 of 2)**

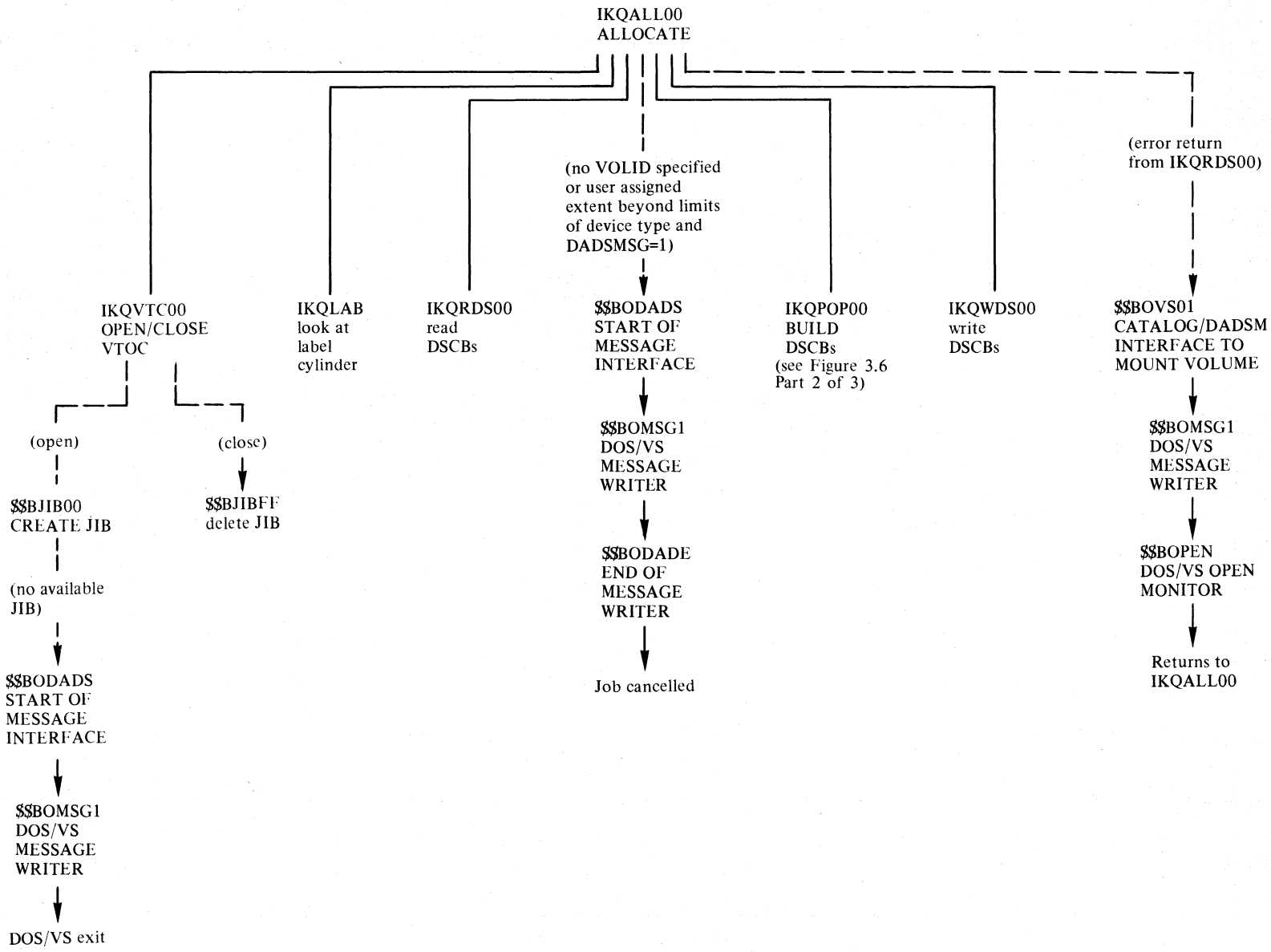


Figure 3.6 **Program structure to process DASDM (part 1 of 3)**

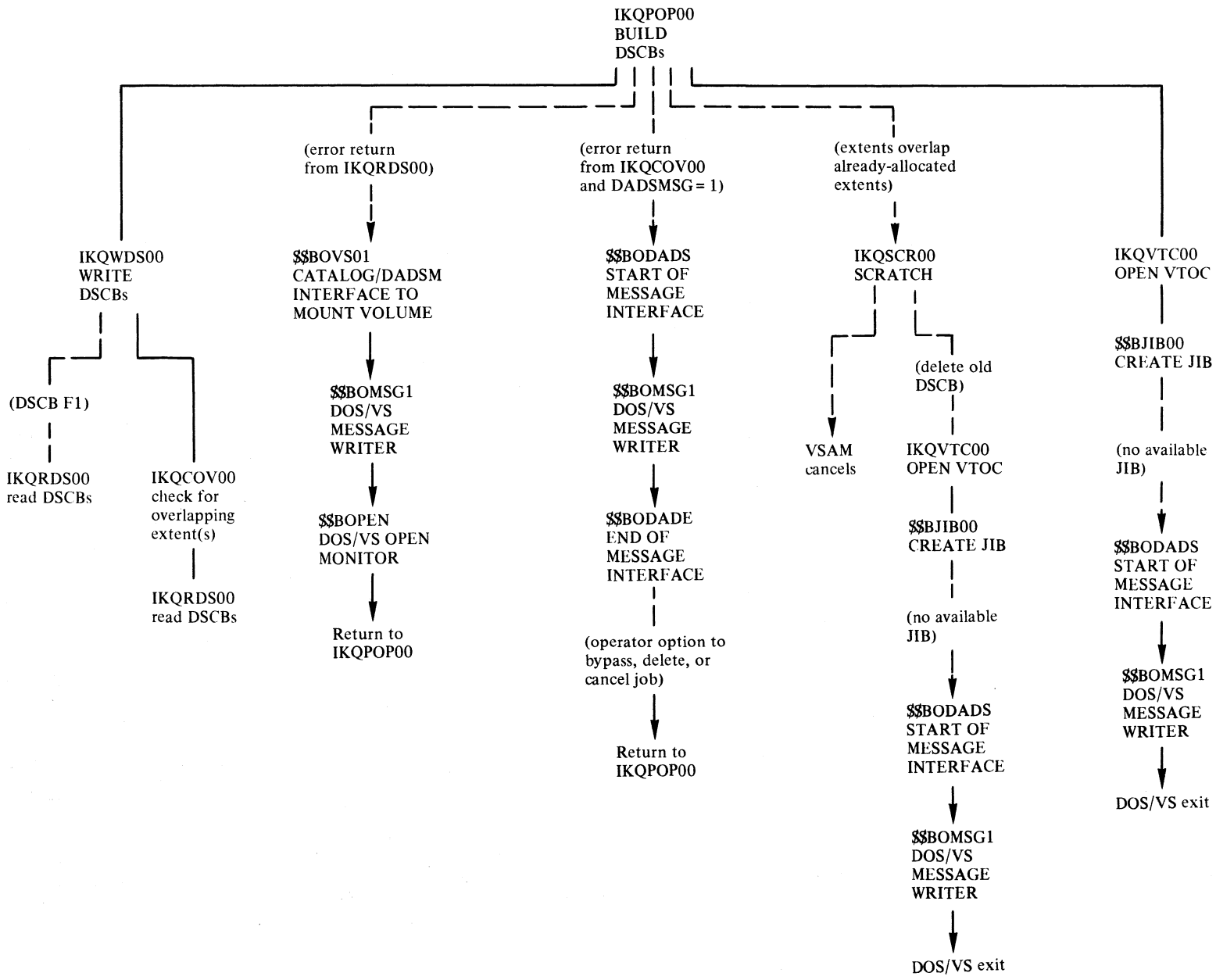
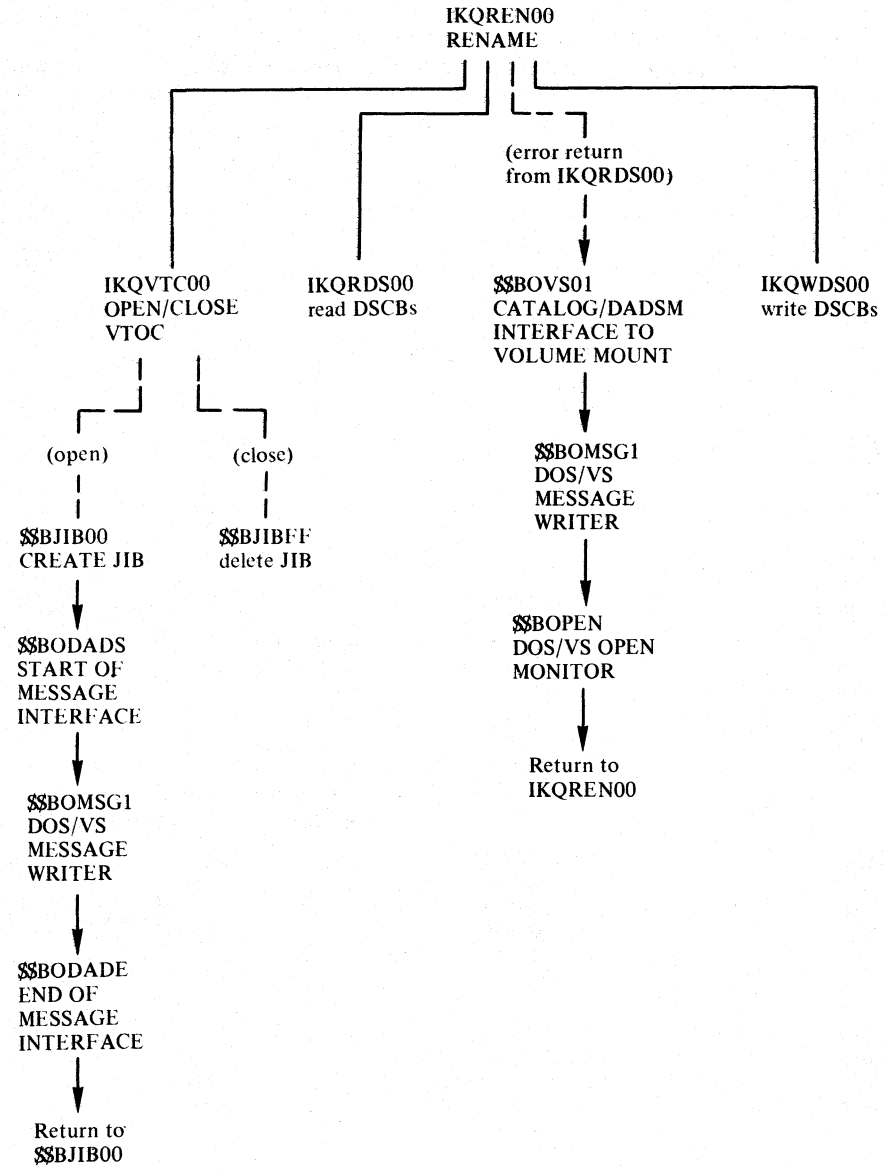


Figure 3.6 **Program structure to process DASDM (part 2 of 3)**



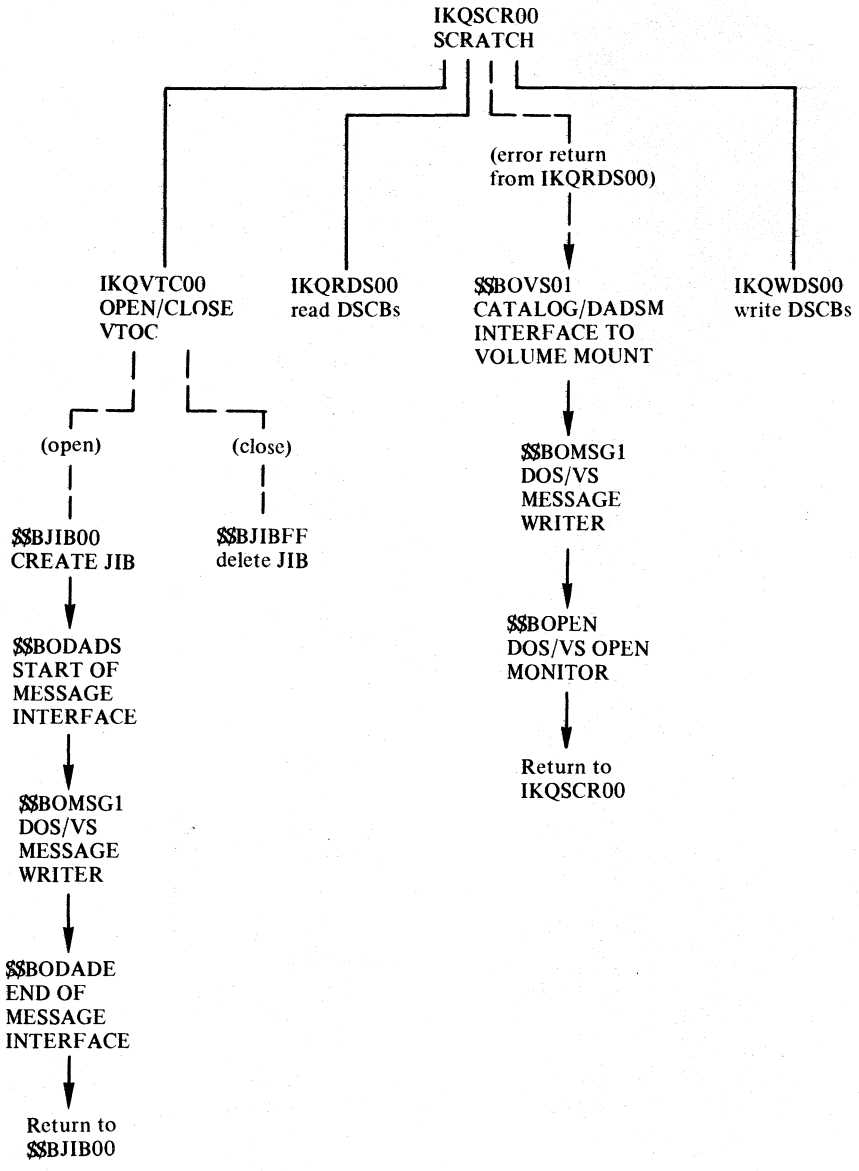


Figure 3.6 **Program structure to process DASDM (part 3 of 3)**

This page has been left blank intentionally

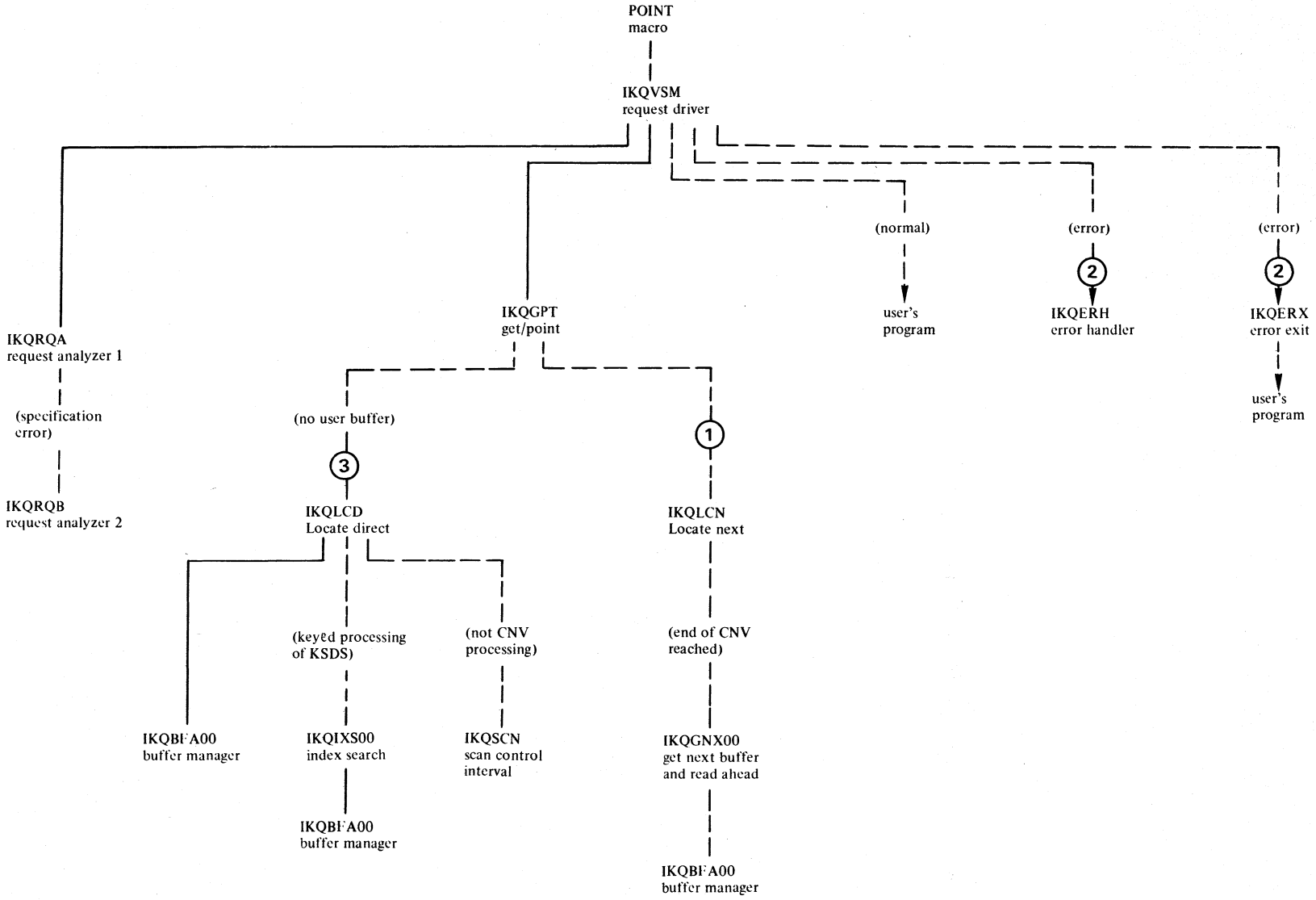


Figure 3.7 **Program structure to process a POINT (part 1 of 2)**

(1.) IKQLCN is called - if no user buffer and
a) IKQLCD didn't find the record and reached end of Control interval whenever: - FWD, KGE or
- FWD, GEN or
- BWD, LRD
b) BWD, LRD with ADR processing or keyed processing of RRDS

(2.) Possible logical errors are:
- ADR: Invalid RBA
- Keyed: no record found
end of data

(3.) IKQLCD is not called - if user buffer
- if BWD, LRD, ADR processing
- if BWD, LRD keyed processing of RRDS

Figure 3.7 Program structure to process a POINT (part 2 of 2)

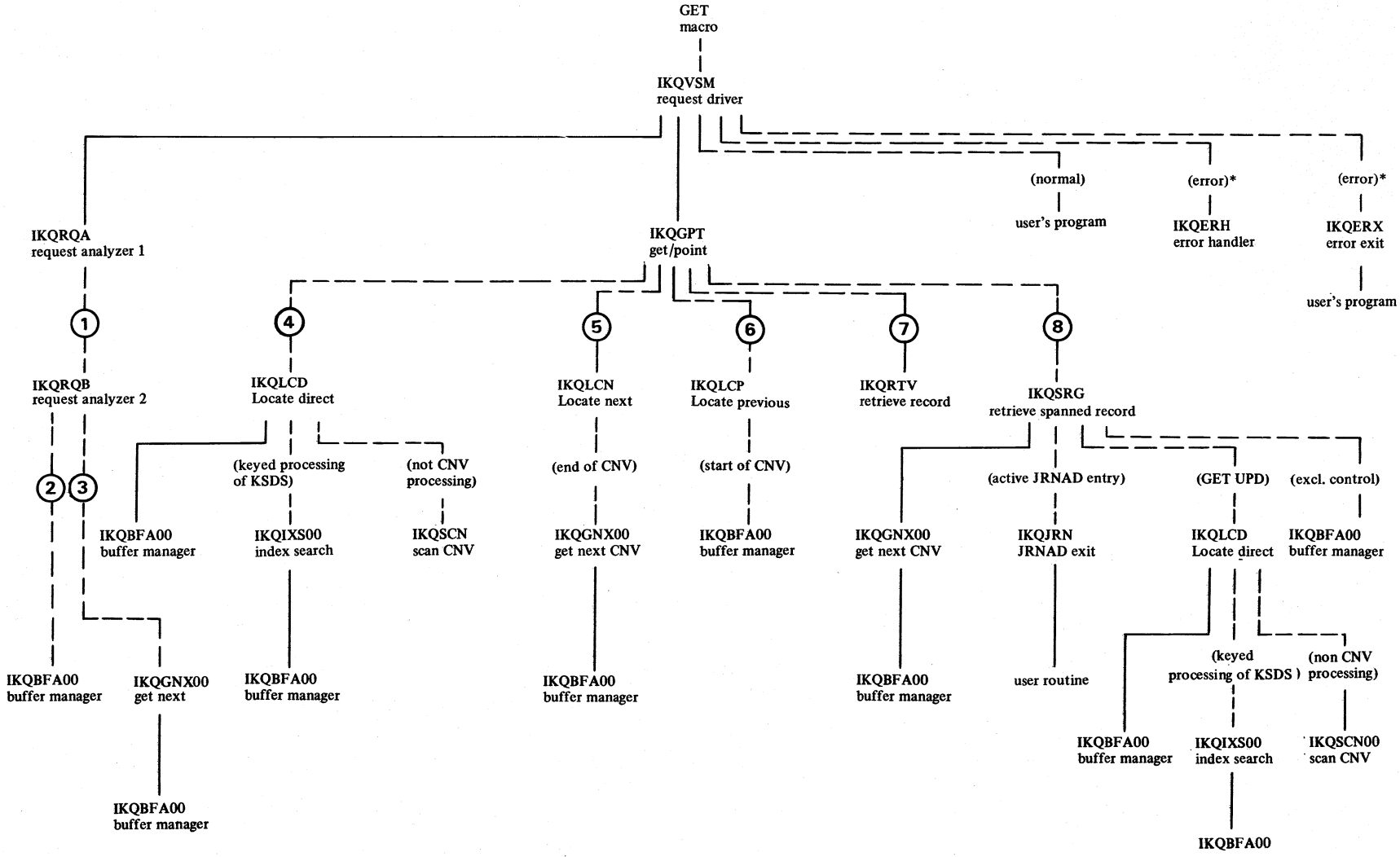


Figure 3.8 **Program structure to process a GET (part 1 of 2)**

GET macro

(1.) IKQRQB is called
- for initial positioning of the PLH for sequential forward processing
- if specification errors were detected

(2.) IKQBFA00 is called
- to read the first control interval of a data set

(3.) IKQGNX is called
- if the first control interval is empty

(4.) IKQLCD is called
- for direct and skip sequential processing, except for the retrieval of the last record (LRD) during addressed or keyed processing of an RRDS
- for sequential processing
  - . if restart is required
  - . if exclusive control was required but not obtained
  - . if previous request resulted in an error or was to end of data
- for user buffer processing
- during sequential backward processing, of a KSDS, if a transition to the previous sequence set record is required

(5.) IKQLCN is called
- for sequential forward processing, if the PLH is positioned to the last record
- for skip sequential or direct forward processing, if the record could not be found by IKQLCD and the end of a control interval was reached and KGE or GEN was specified
- for overlapped advance of the PLH, if the request is not for update and not LOC and UBF, or not BWD

(6.) IKQLCP is called
- for sequential backward processing, if the PLH is positioned
- for direct backward LRD processing or keyed processing of an RRDS
- for direct backward LRD keyed processing of a KSDS, if IKQLCD located an empty control interval

(7.) IKQRTV is called
- for retrieval of non-spanned records, if user buffer processing is not specified

(8.) IKQSRG is called
- for retrieval of spanned records

* Possible logical errors
- E 32 invalid RBA
  E 33 no record found
  E 34 end of data
  E 35 user area too small
  E 36 sequence error
  E 44 exclusive control error
  E 46 locate mode for spanned record GET
  E 47 inconsistent spanned record

Figure 3.8 Program structure to process a GET (part 2 of 2)

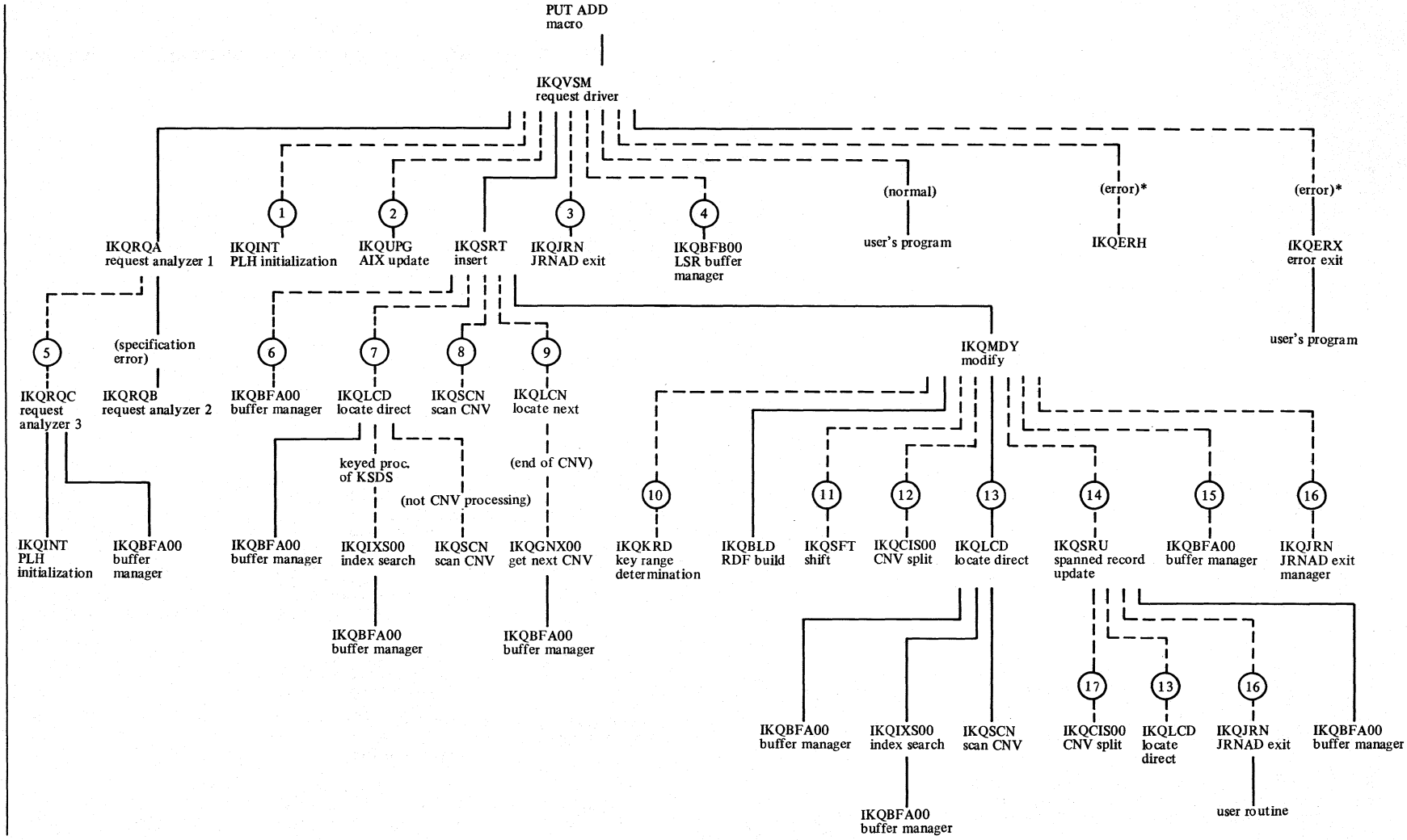


Figure 3.9 **Program structure to process a PUT (part 1 of 4)**

PUT ADD

(1) IKQINT is called – if LSR is specified, to initialize a PLH

(2) IKQUPG is called – if an upgrade set exists

(3) IKQJRN is called – if the JRNAD exit is active, to inform the user of data set changes

(4) IKQBFB is called – if LSR is specified, to return control blocks to the buffer pool

(5) IKQRQC is called
- if path processing and LSR are specified, to assign a PLH to the base cluster
- if an upgrade set exists and LSR is specified, to assign BCBs

(6) IKQBFA00 is called – for addressed processing

(7) IKQLCD is called – for keyed processing of an RRDS
  - if direct
  - if skip sequential
  - if sequential, after exceptional conditions or to obtain exclusive control
- for keyed processing of a KSDS
  - if direct
  - after exceptional condition
  - if PLH is not positioned
  - to obtain exclusive control

(8) IKQSCN is called – in order to find the correct insertion point for keyed processing of a KSDS

(9) IKQLCN is called – whenever the PLH is positioned to the previous record for keyed sequential processing of a KSDS or an RRDS

(10) IKQKRD is called – to determine keyrange changes for a keyrange data set

(11) IKQSFT is called – in order to make room for the record when no CNV split is necessary

(12) IKQCIS00 is called – if a control interval split (or pseudo split) is necessary
  - if actual CNV free space is too short for changes (real split)
  - for CNV insert processing (pseudo split) except for an ESDS (see Note)
  - if first data load request (pseudo split)
  - if keyrange change (pseudo split)
  - if RRDS and insertion beyond preformatted limit (pseudo split)

Note: For CNV insert processing of an ESDS, IKQMDY carries out the pseudo-split internally.

(13) IKQLCD is called – in order to reposition the PLH when insertion is to be retried after a CNV split. For keyed non-load processing only.

(14) IKQSRU is called – in order to insert spanned records

(15) IKQBFA00 is called – if immediate writing is required for user buffer processing or direct requests without the option NSP

(16) IKQJRN is called – if the JRNAD exit is active, to inform the user of data set changes

(17) IKQCIS00 is called
- if there are not enough CNVs in the control area to accept the spanned record
- for the insertion of each segment

*Logical errors – E 36 sequence error
E 37 duplicate record
E 43 VSAM internal logic error

Figure 3.9 Program structure to process a PUT (part 2 of 4)

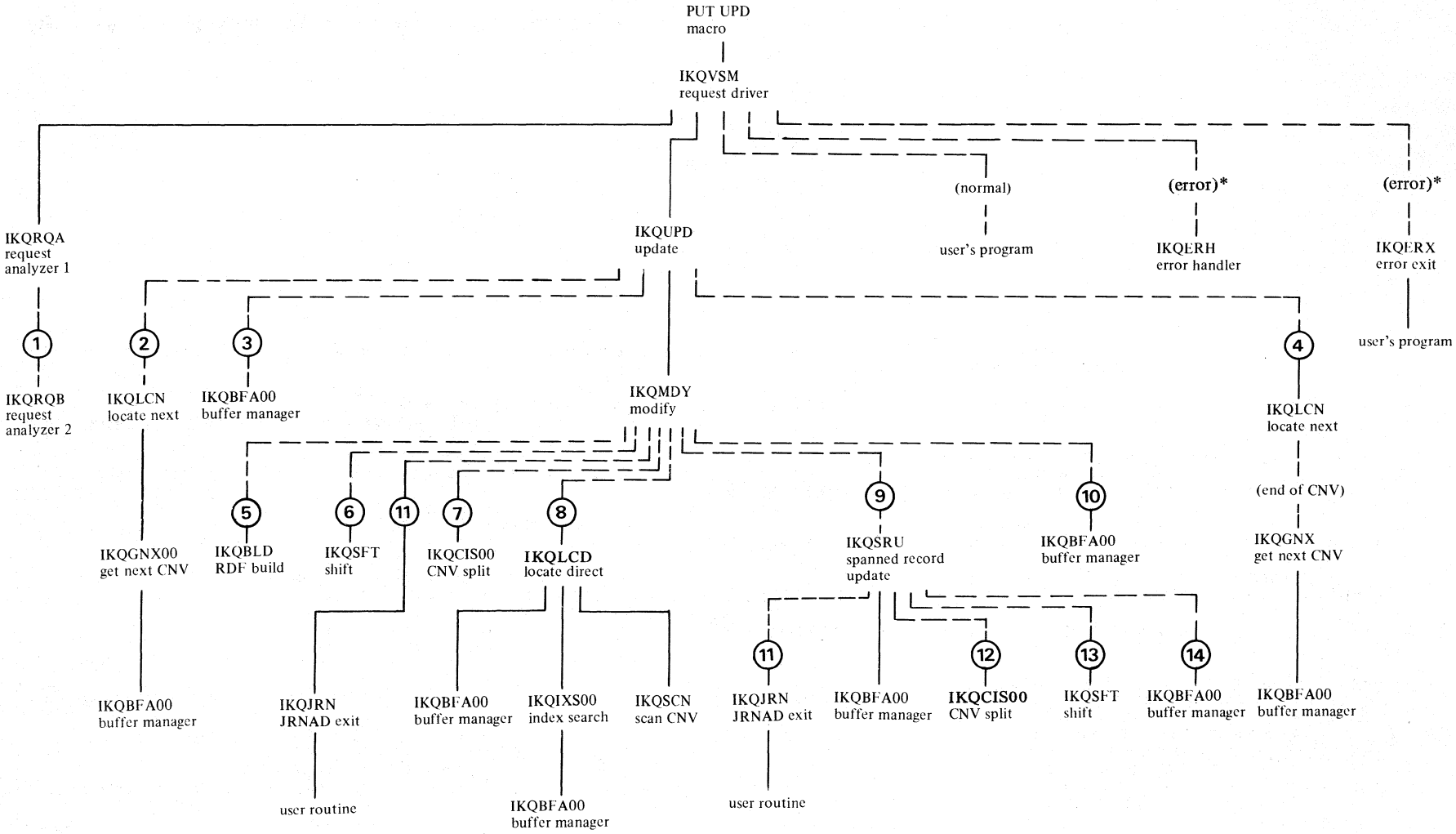


Figure 3.9 **Program structure to process a PUT (part 3 of 4)**

PUT UPD

(1) IKQRQB is called – for the initial positioning of the PLH for a sequential standalone update (user buffer processing only)
– if a specification error was detected

(2) IKQLCN is called – for sequential standalone update (user buffer processing only), if the PLH is positioned to the previous CNV

(3) IKQBFA00 is called – in order to get the CNV to be updated for a direct standalone update

(4) IKQLCN is called – in order to advance the PLH overlapped for a sequential forward request without UBF or LOC specified

(5) IKQBLD is called – if an update to a non-spanned record causes a length change

(6) IKQSFT is called – if a record with changed length fits into the CNV

(7) IKQCIS00 is called – if a record with changed length does not fit into the CNV

(8) IKQLCD is called – in order to reposition the PLH when the update is to be retried after a CNV split–for keyed processing only

(9) IKQSRU is called – if the old record or the new record is spanned

(10) IKQBFA00 is called – if immediate writing is required for user buffer processing or for direct requests without the option NSP

(11) IKQJRN is called – if the JRNAD exit is active, in order to inform the user of changes to the data set

(12) IKQCIS00 is called – if more control intervals are needed than are allocated

(13) IKQSFT is called – in order to reorganize the sequence set record during CI space reclamation, if control intervals become free during update with length change

(14) IKQBFA00 is called – to write freed data CNVs and changed sequence set records during CI space reclamation

*Logical error – End of data (for sequential standalone update)

Figure 3.9 Program structure to process a PUT (part 4 of 4)

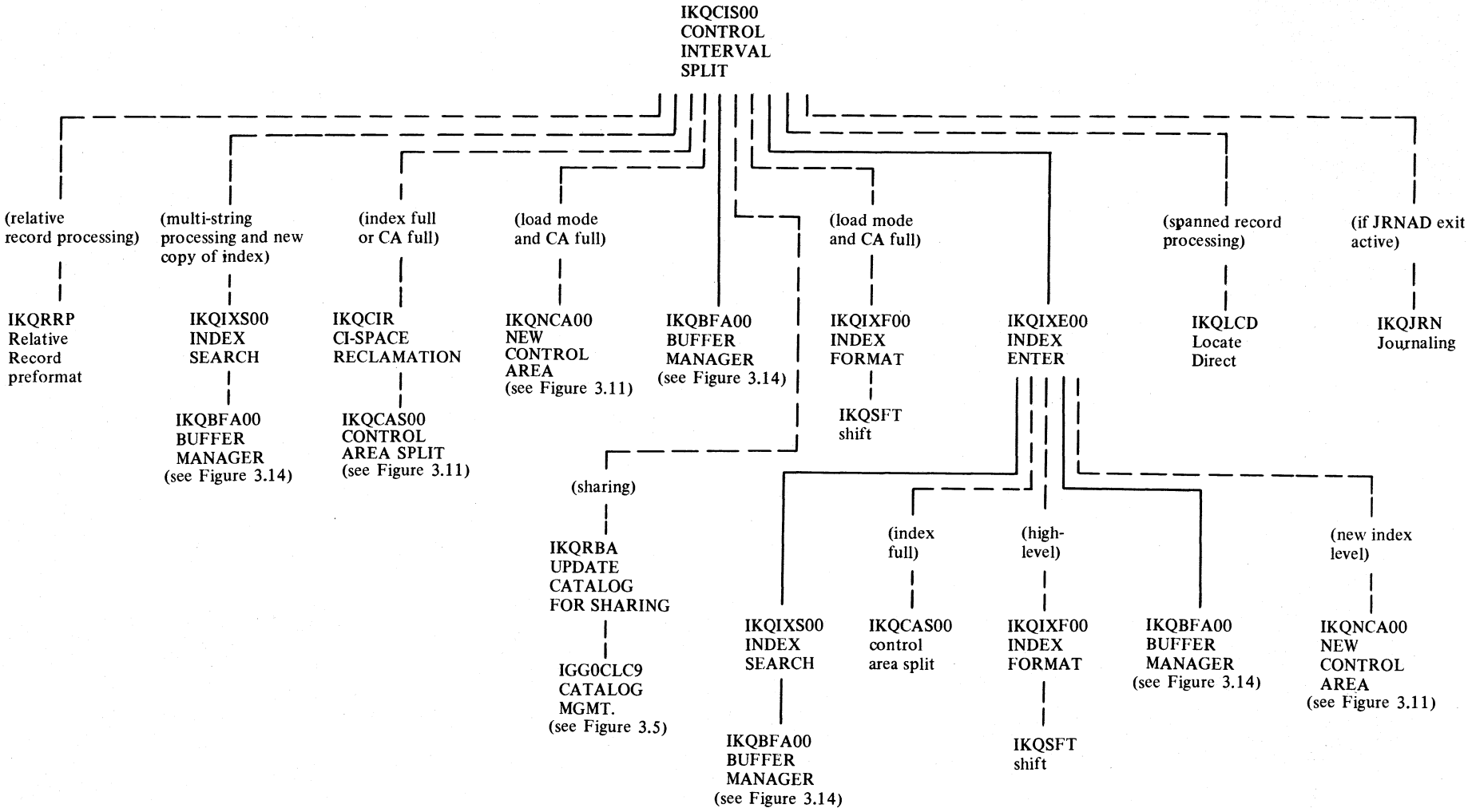


Figure 3.10 **Program structure to process a control interval split**

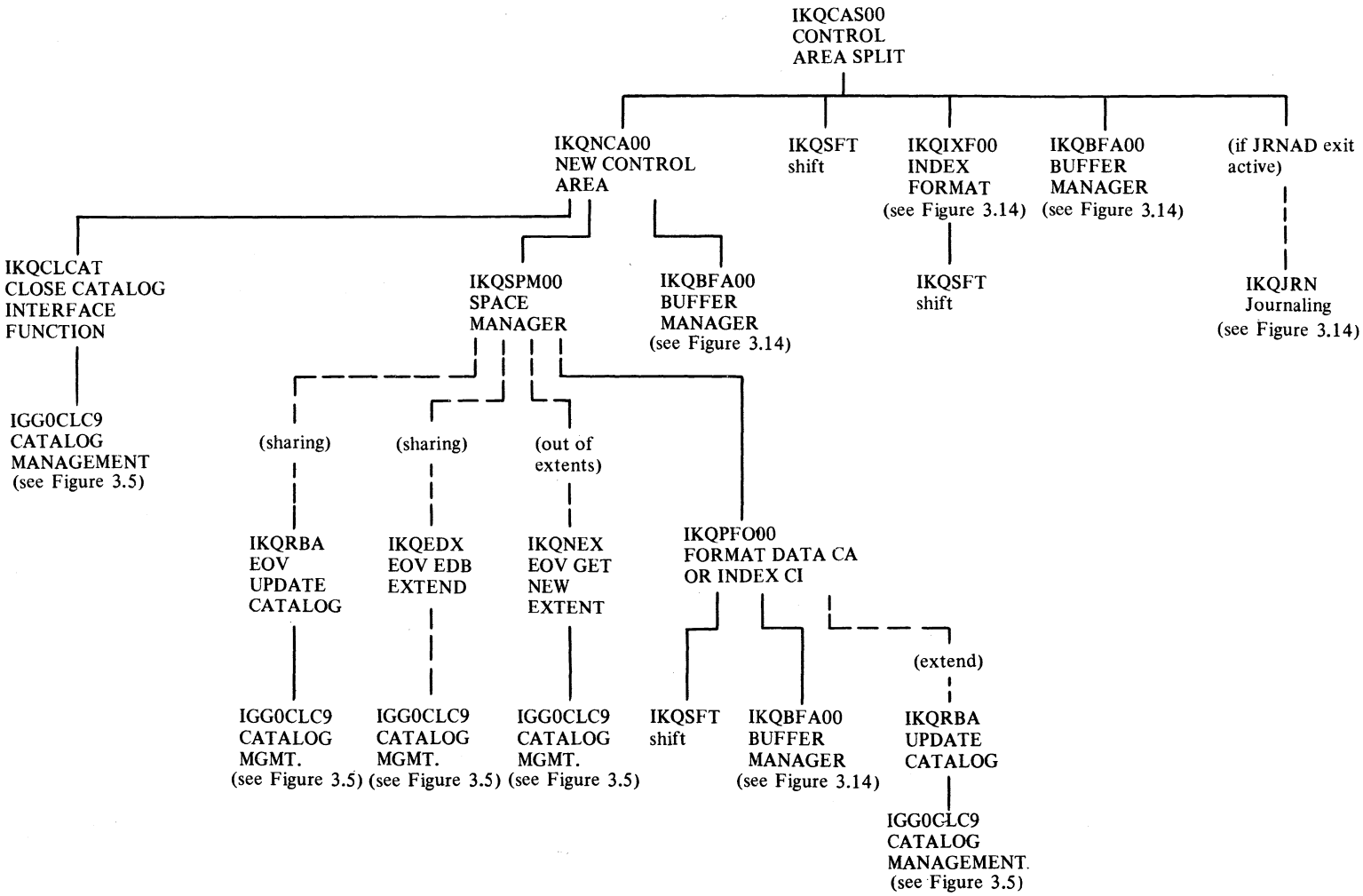


Figure 3.11 **Program structure to process a control area split**

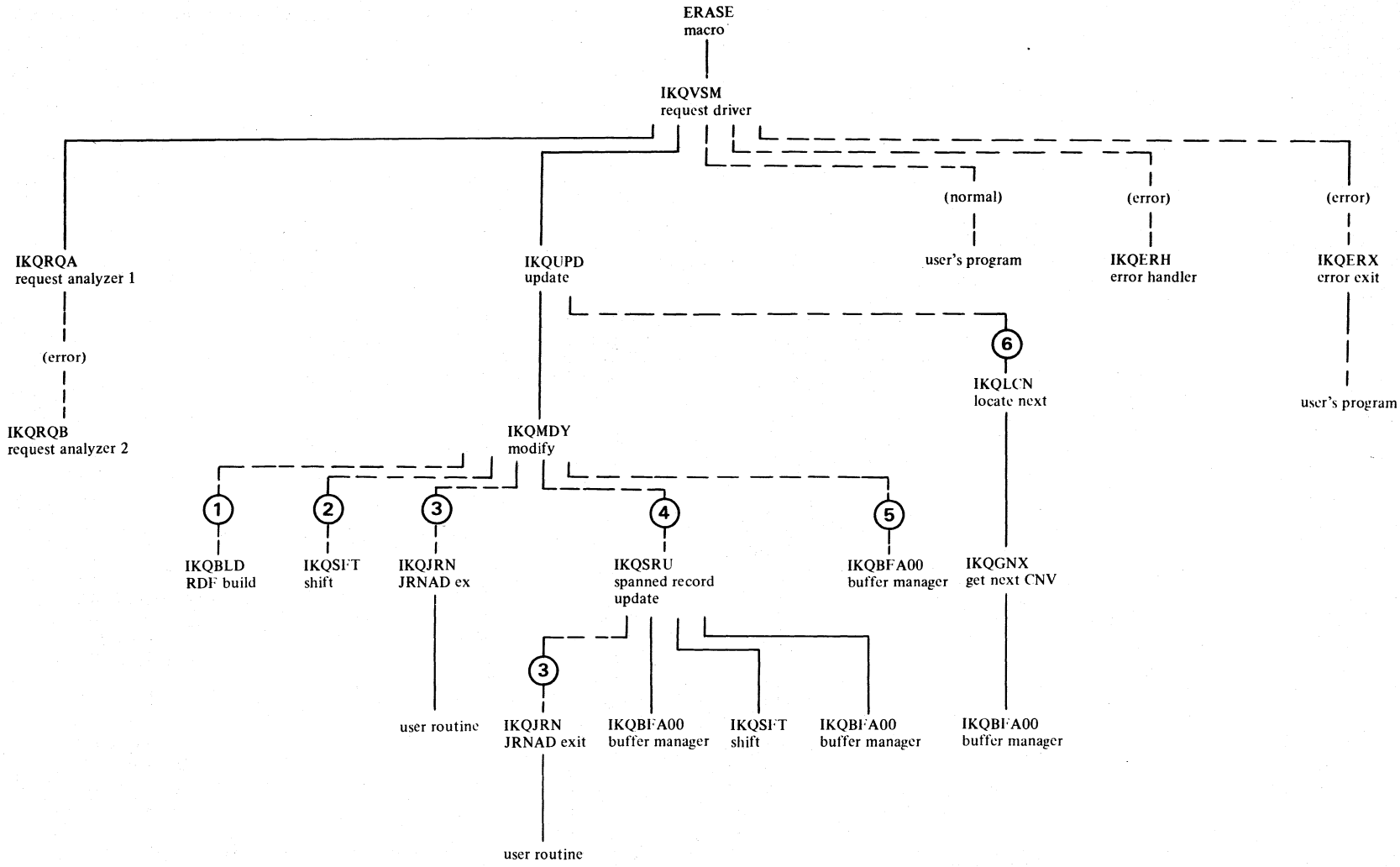


Figure 3.12 **Program structure to process an ERASE (part 1 of 2)**

ERASE

1. IKQBLD is called - if the ERASE request is for a KSDS
2. IKQSFT is called - if the ERASE request is for a KSDS
3. IKQJRN is called - if the JRNAD exit is active
4. IKQSRU is called - if a spanned record is to be erased
5. IKQBFA00 is called - for direct requests without the NSP option
6. IKQLCD is called - for sequential and skip sequential requests during forward processing, whenever the end of a control interval is reached

Figure 3.12 Program structure to process an ERASE (part 2 of 2)

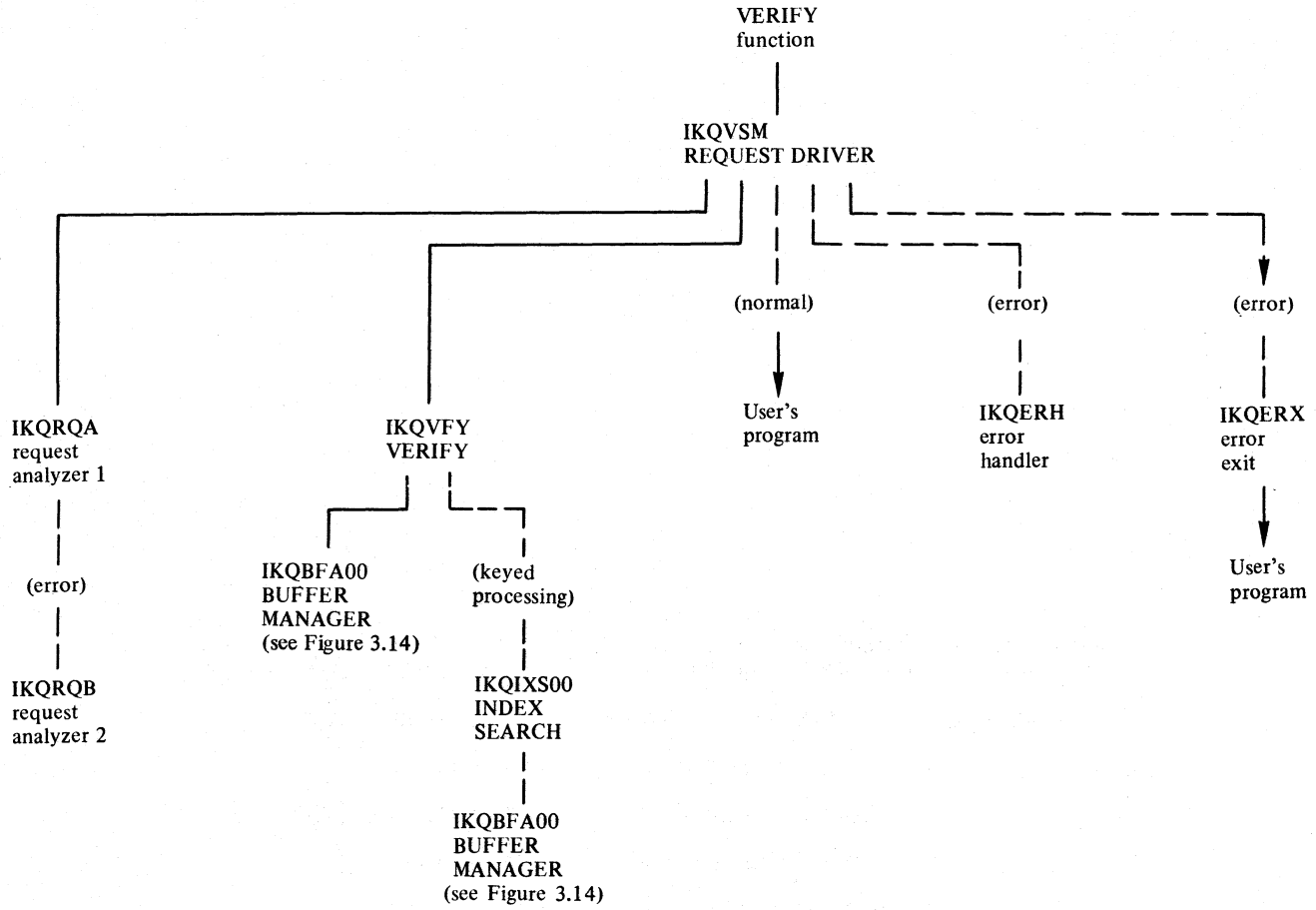


Figure 3.13 **Program structure to process a VERIFY**

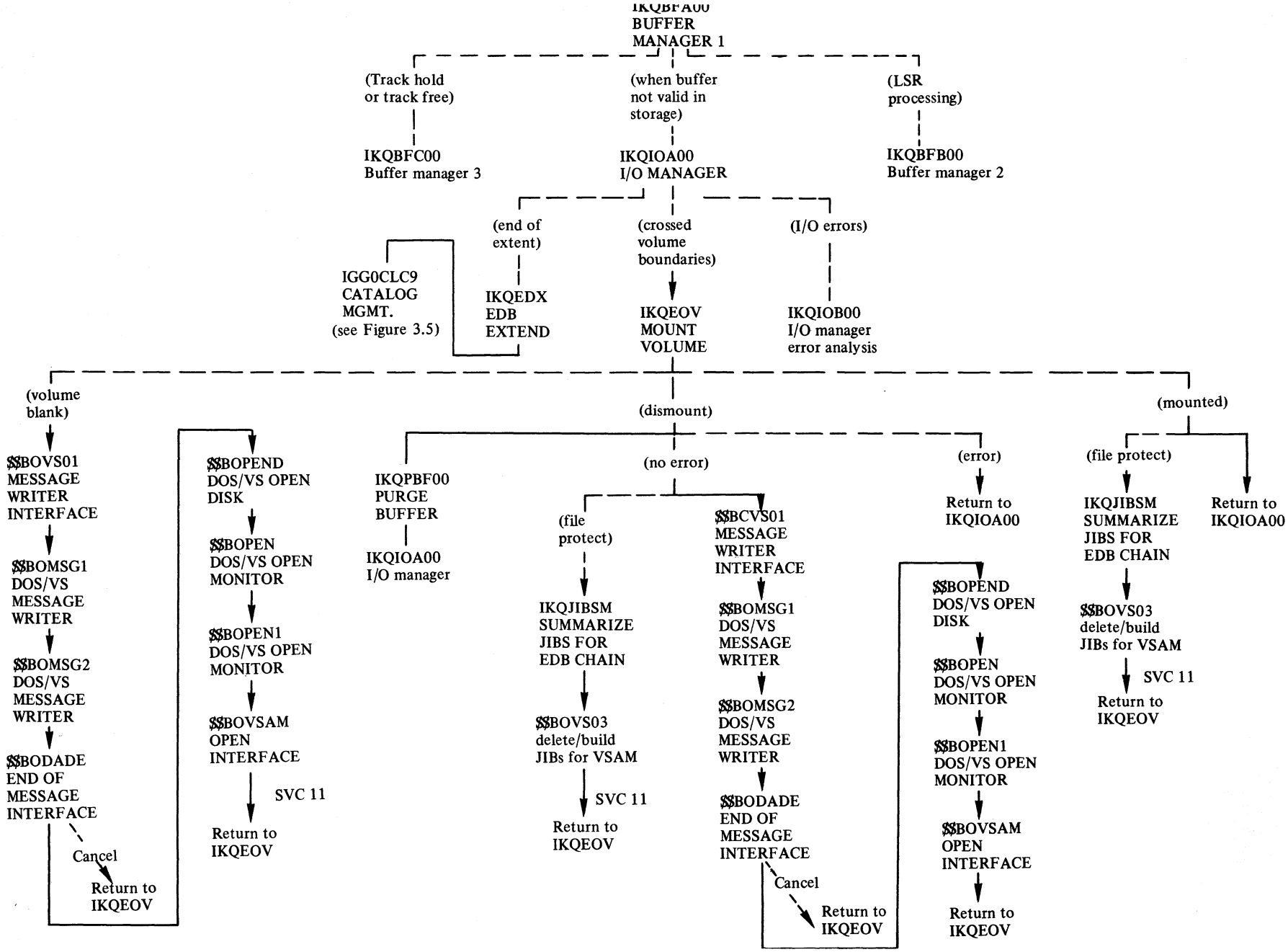


Figure 3.14 **Program structure to process buffer and I/O management**

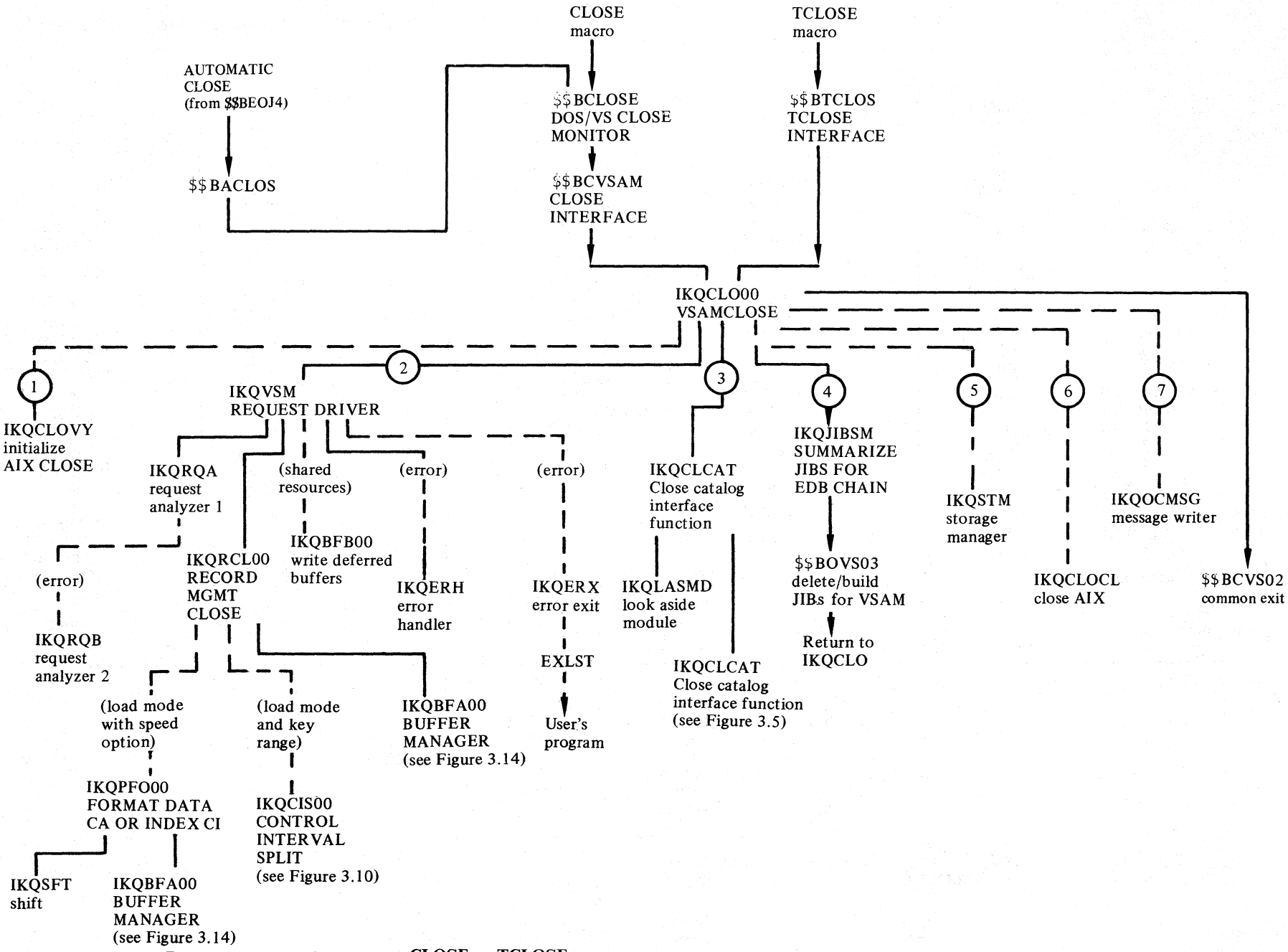


Figure 3.15 **Program structure to process a CLOSE or TCLOSE**

(1) IKQCLOVY is called if an Alternate Index structure is to be closed: i.e. a path entry or an upgrade set exists. IKQCLOVY determines which cluster is to be closed first.

(2) IKQVSM is called to complete outstanding I/O. For LSR (local shared resources) deferred I/O is completed.

(3) IKQCLCAT is called to update the high water marks and the statistics for the data set to be closed in catalog.

(4) IKQJIBSM is called if the data set was shared across partitions (share option 4). JIB's (Job information blocks) are deleted.

(5) IKQSTM is called for CLOSE Request to free the virtual storage occupied by the control blocks of the data set to be closed.

(6) IKQCLOCL is called if an alternate index structure is to be closed. IKQCLOCL determines the next cluster to be closed.

(7) IKQOCMSG is called whenever an error turns up during close. IKQOCMSG writes a message to the user.

Figure 3.15 Program structure to process a CLOSE or TCLOSE (part 2 of 2)

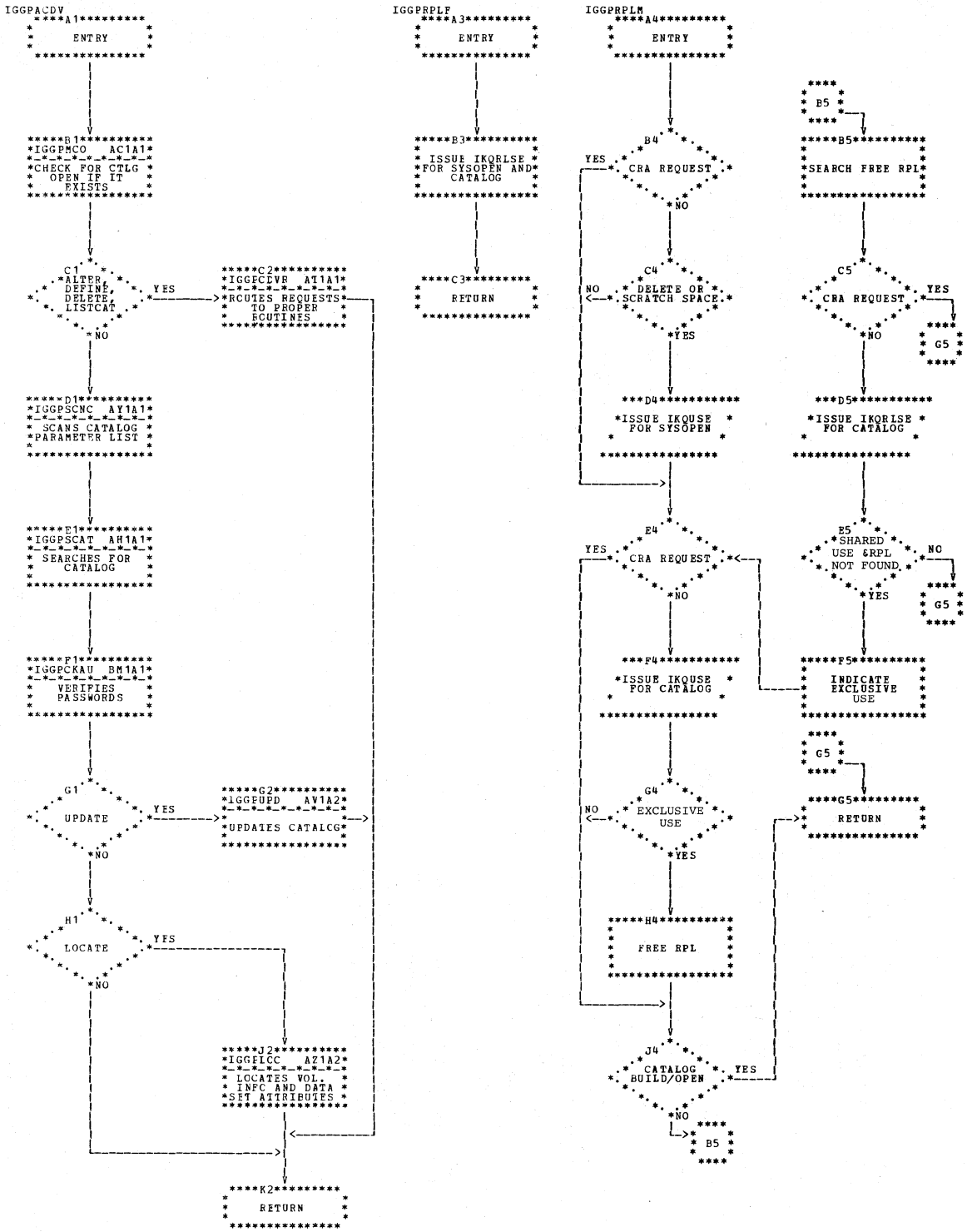


Chart AB1. Catalog driver (IGG0CLAB)

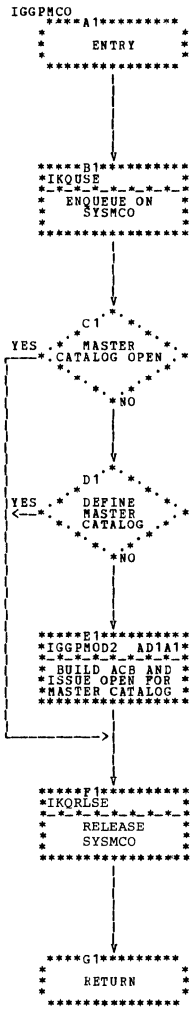


Chart AC1. Master catalog search (IGG0CLAC)

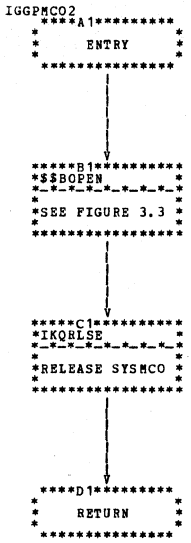


Chart AD1. Master catalog open (IGG0CLAD)

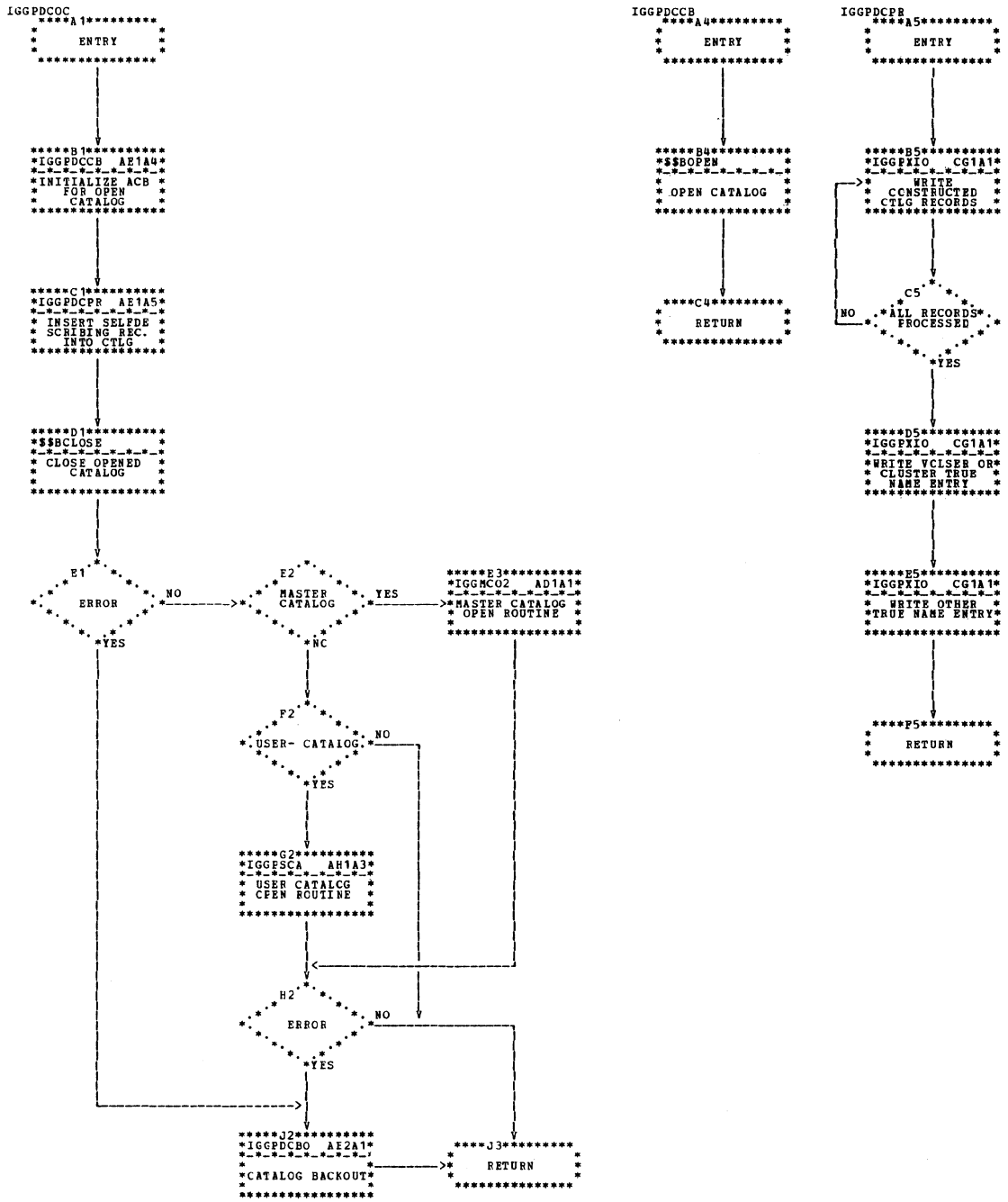


Chart AE1. Catalog build and open (IGG0CLAE)

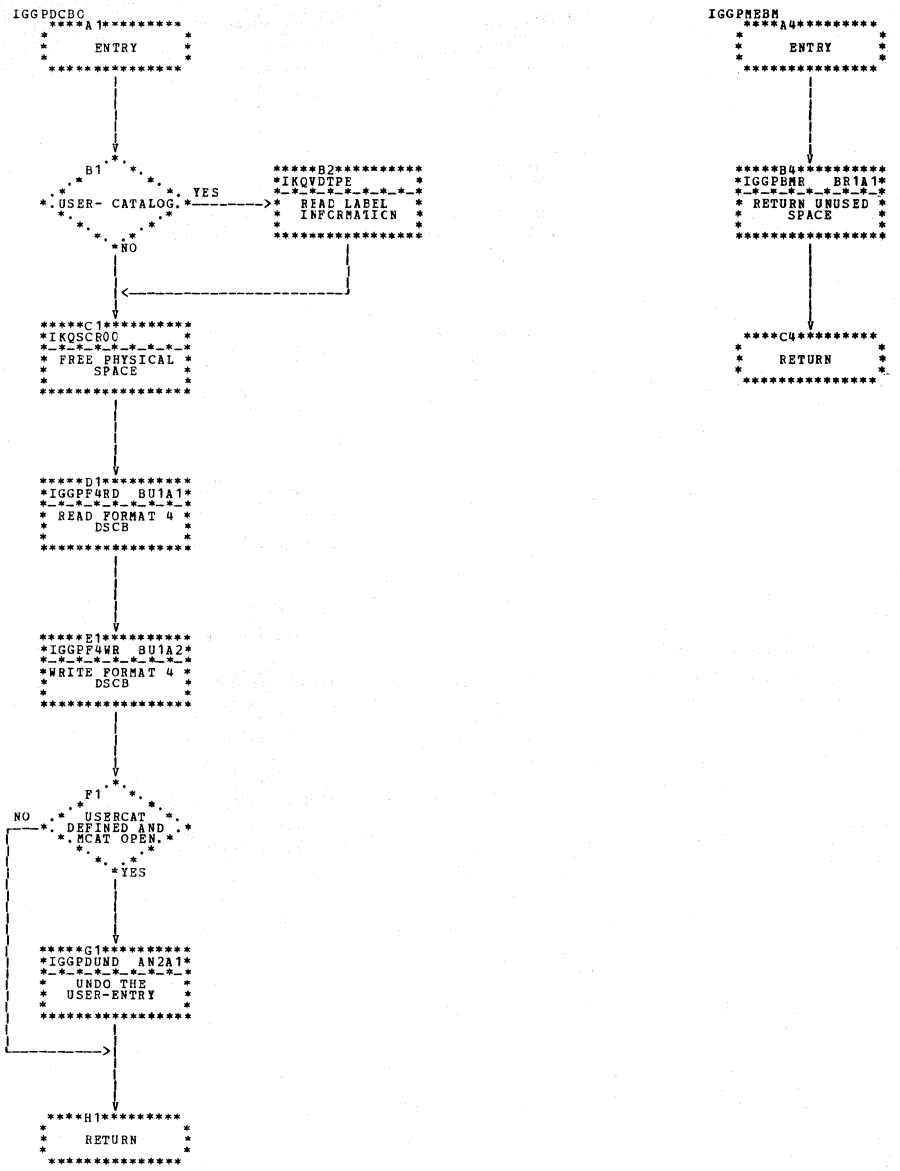


Chart AE2. Catalog build and open (IGG0CLAE)

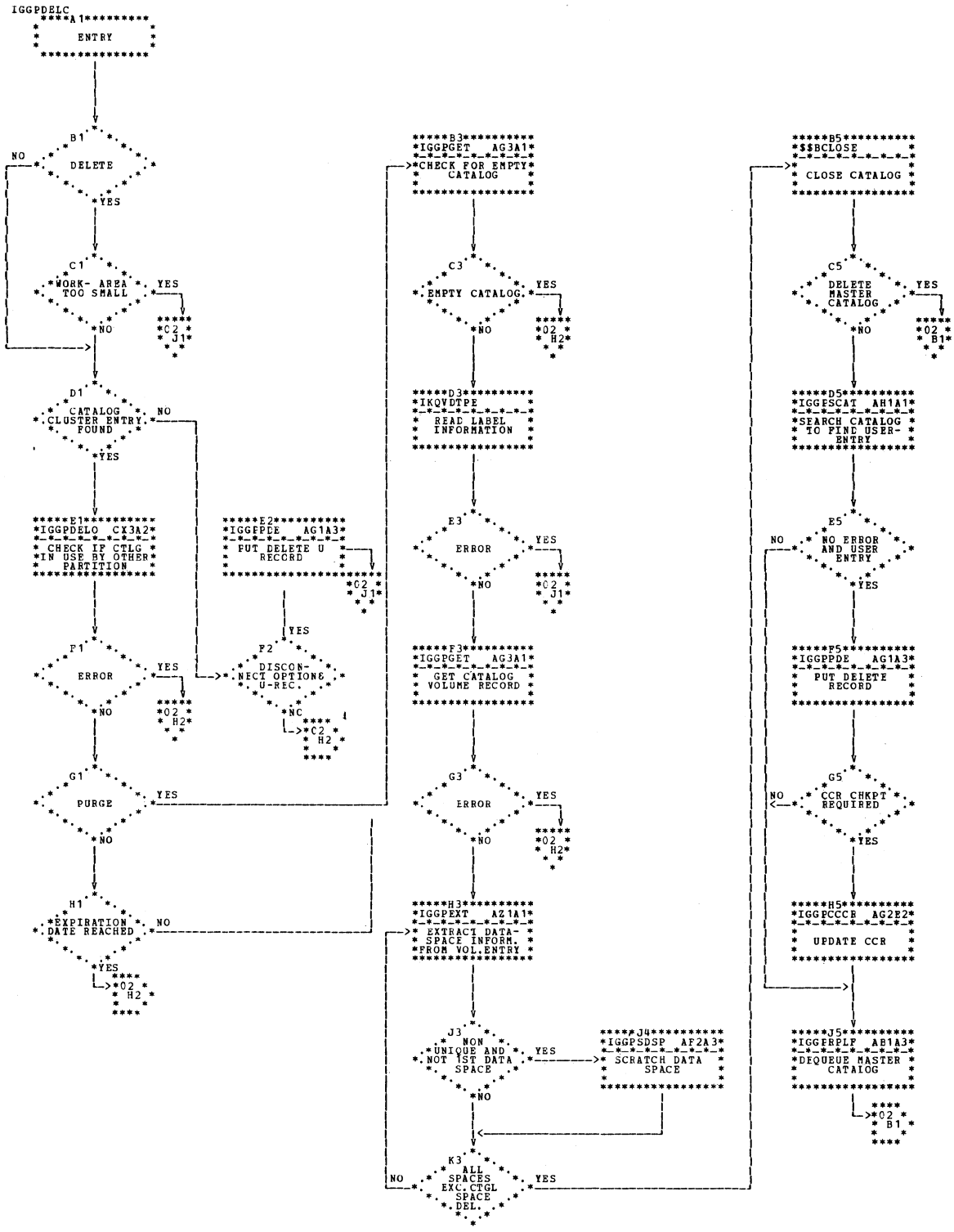


Chart AF1. CMS delete catalog (IGG0CLAF)

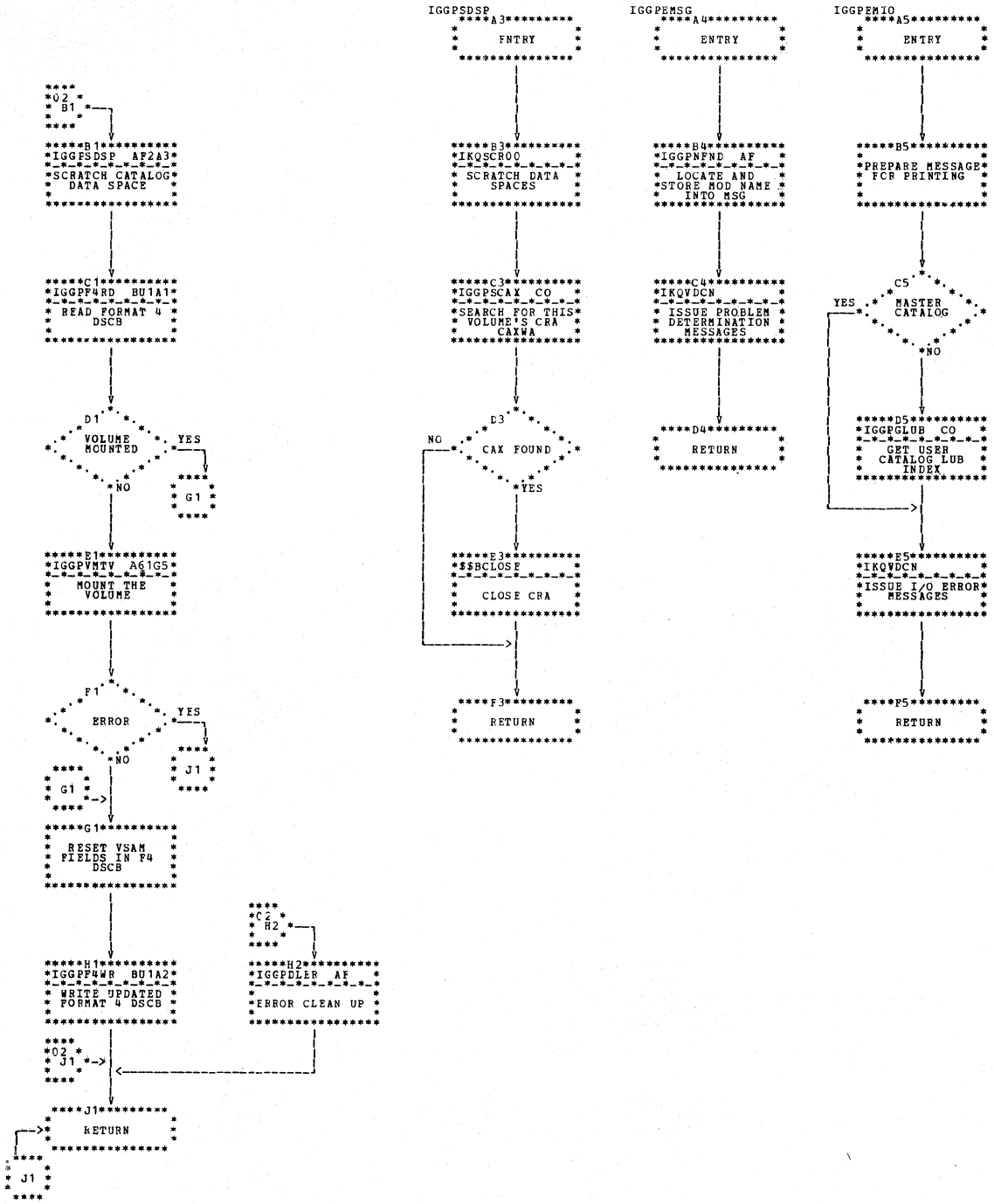


Chart AF2. CMS delete catalog (IGG0CLAF)

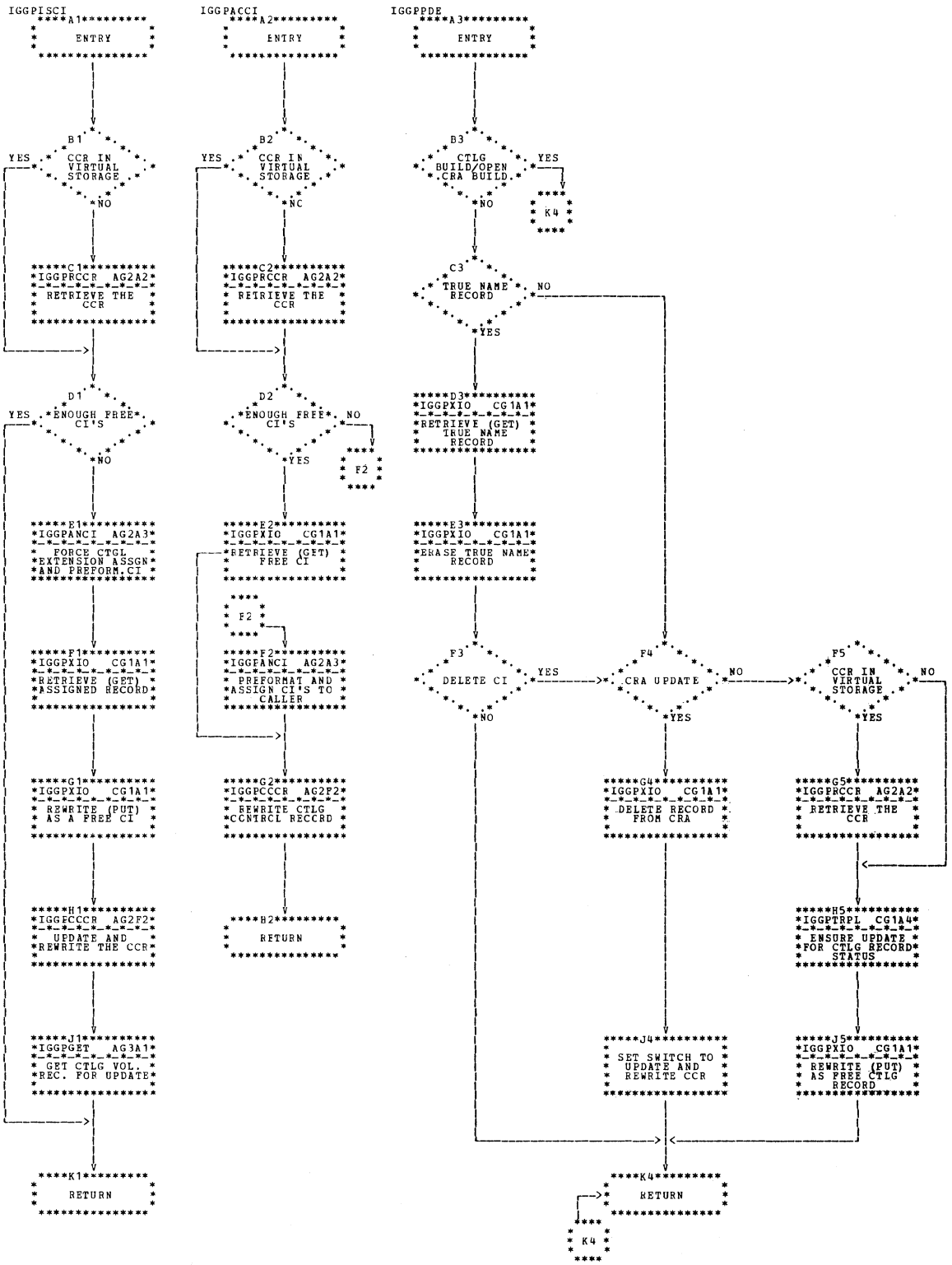


Chart AG1. Catalog I/O Subfunction (IGG0CLAG)

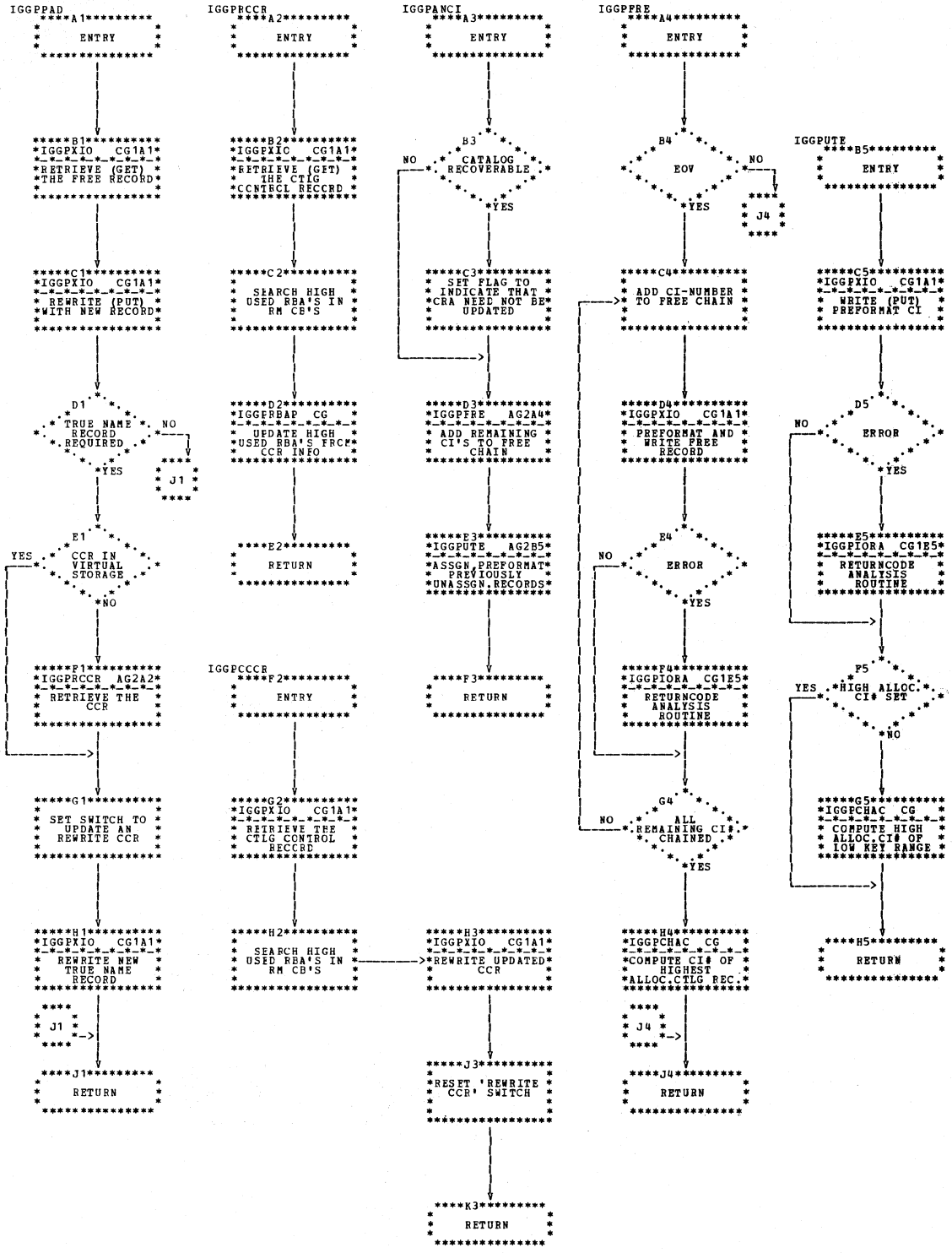


Chart AG2. Catalog I/O Subfunction (IGG0CLAG)

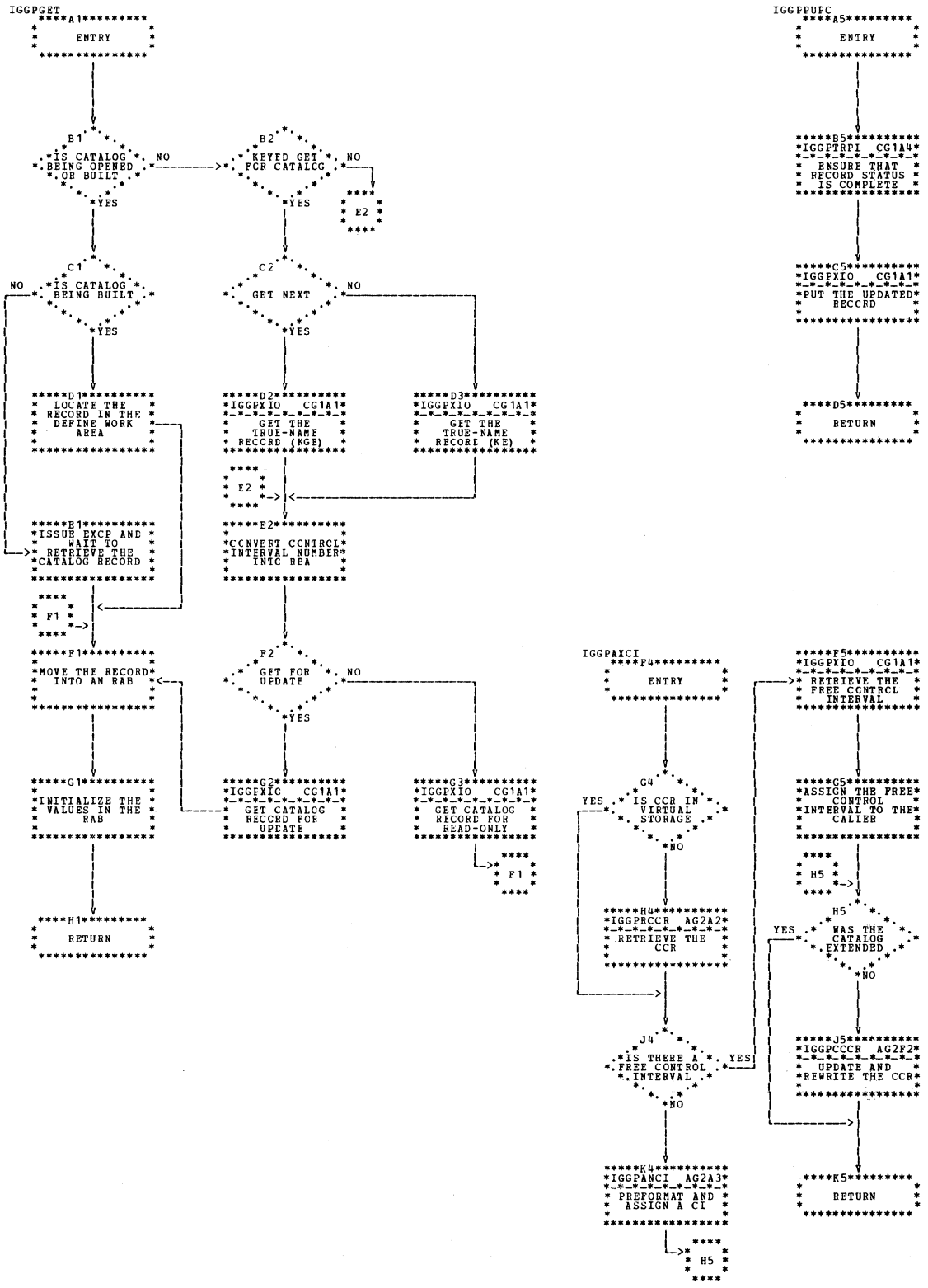


Chart AG3. Catalog I/O Subfunction (IGG0CLAG)

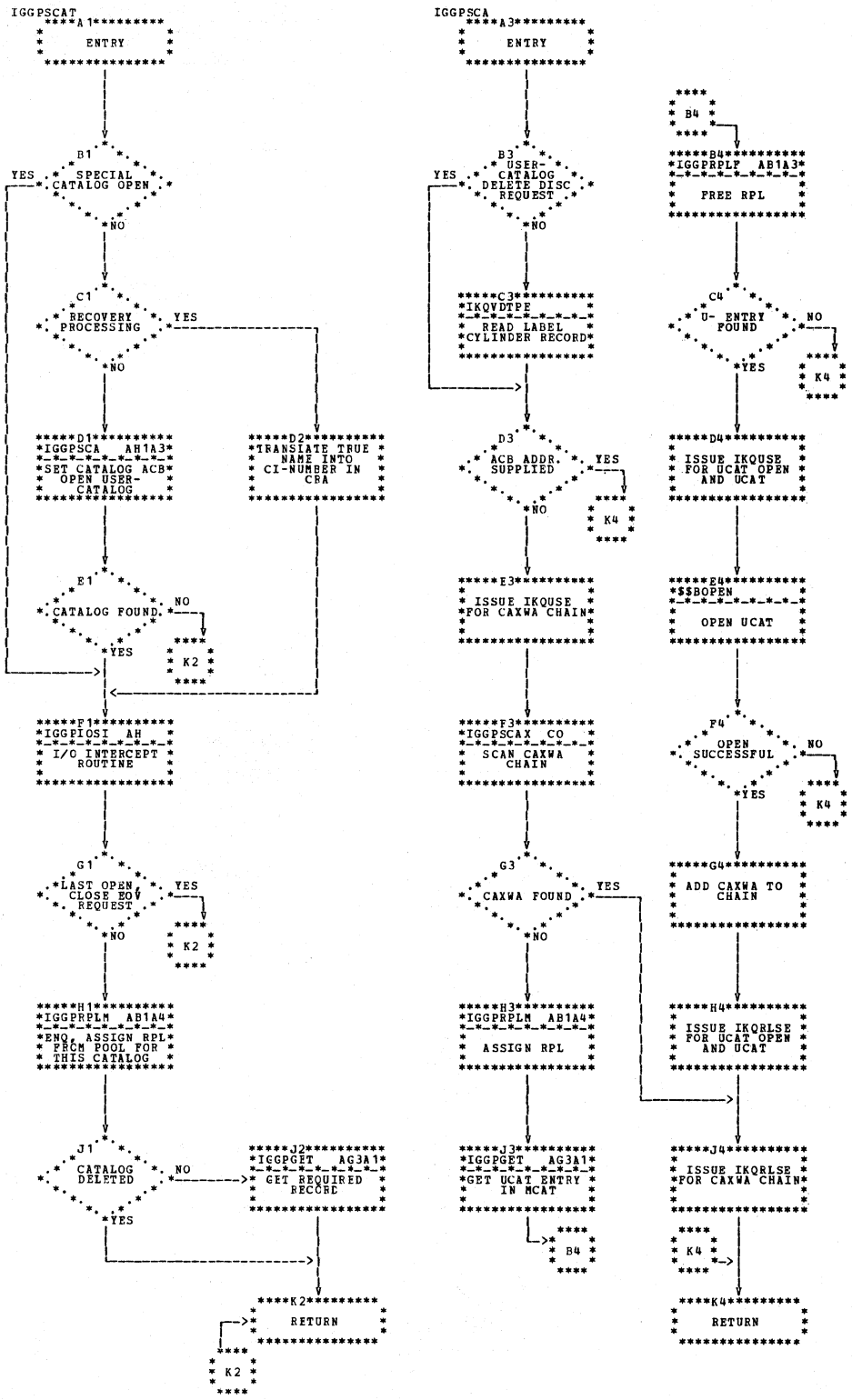


Chart AH1. Search/Open catalog (IGG0CLAH)

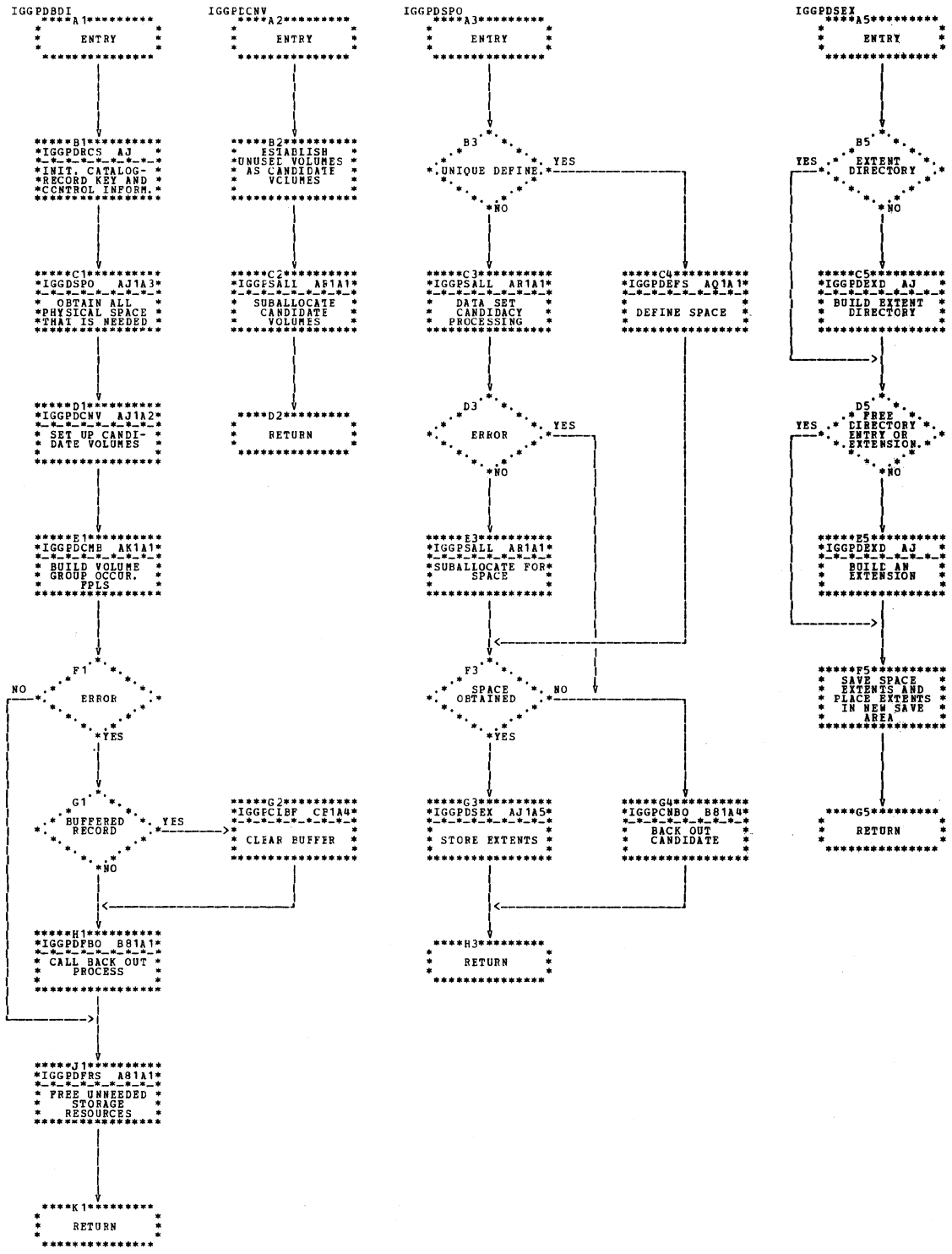


Chart AJ1. Define - build data and index entries (IGG0CLAJ)

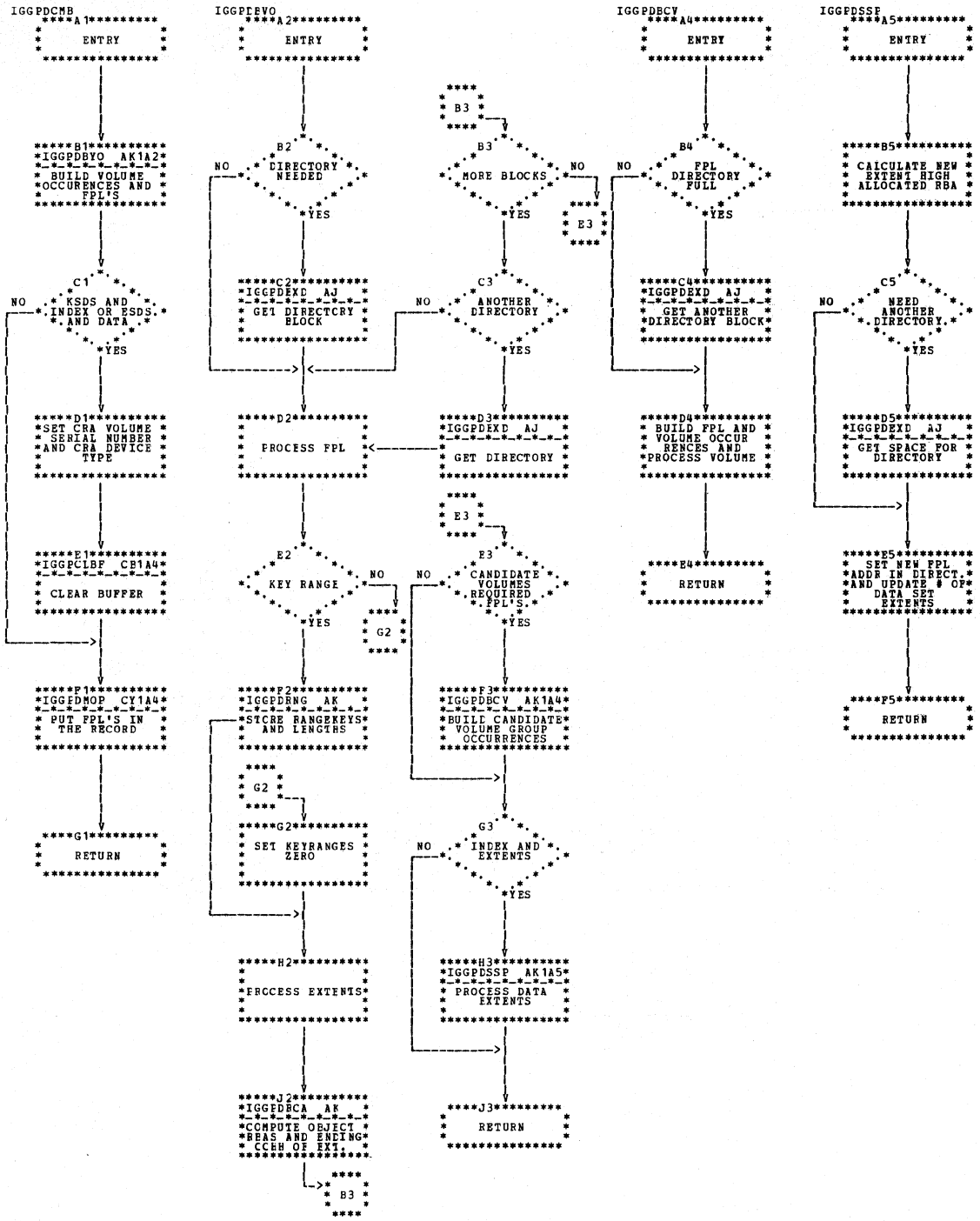


Chart AK1. Complete define of an entry (IGG0CLAK)

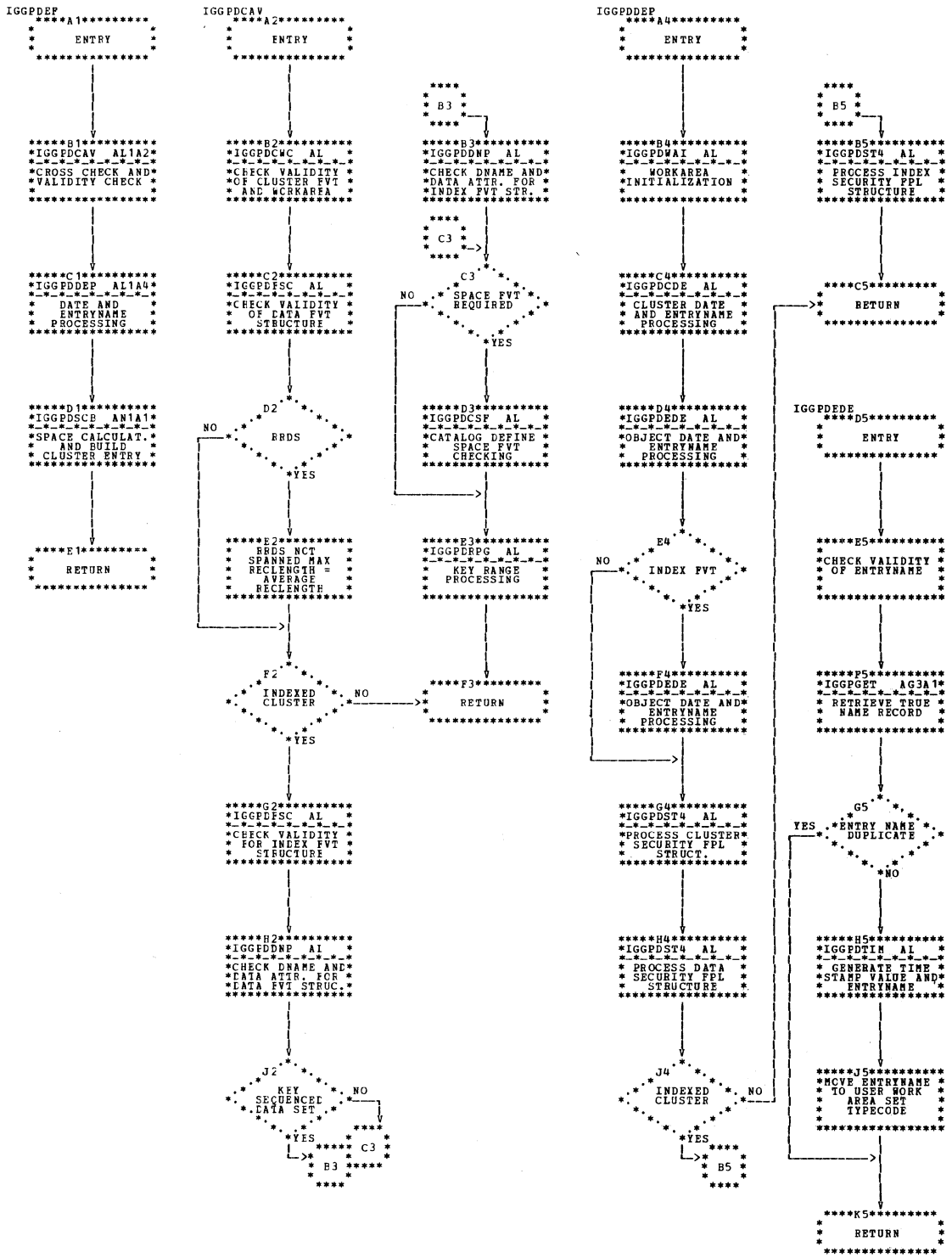


Chart AL1. CMS define - first module (IGG0CLAL)

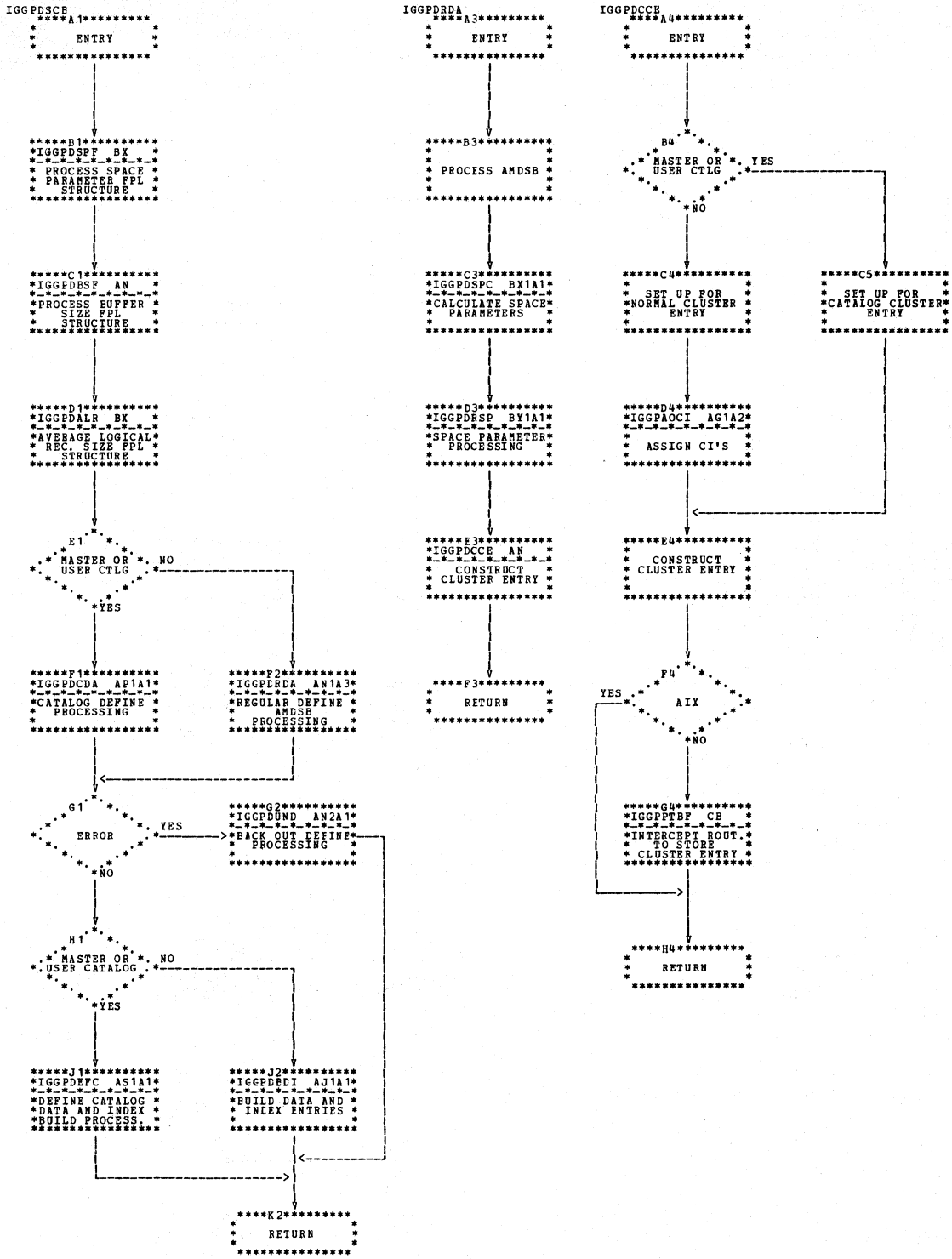


Chart AN1. CMS define - second module (IGG0CLAN)

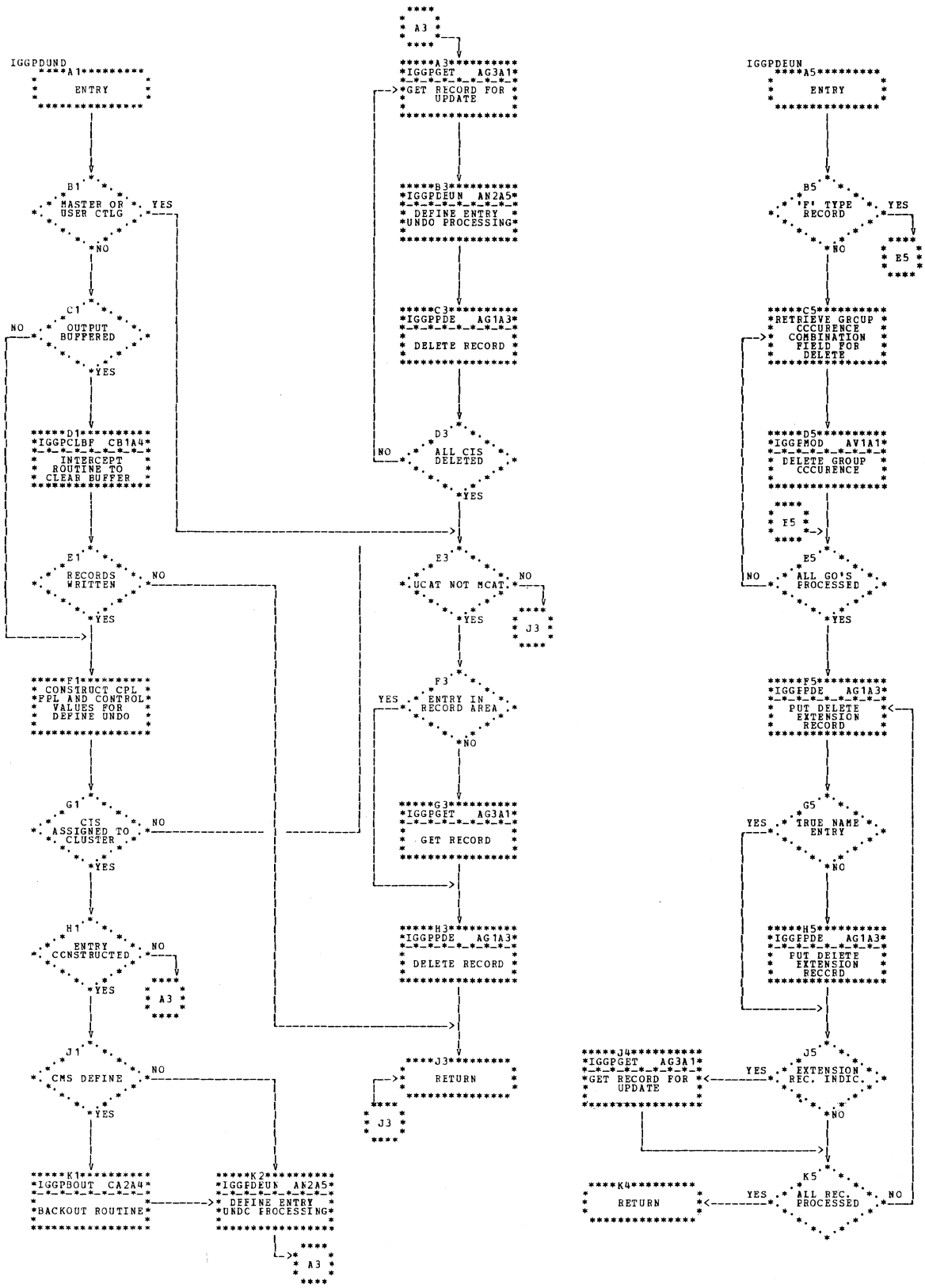


Chart AN2. CMS define - second module (IGG0CLAN)

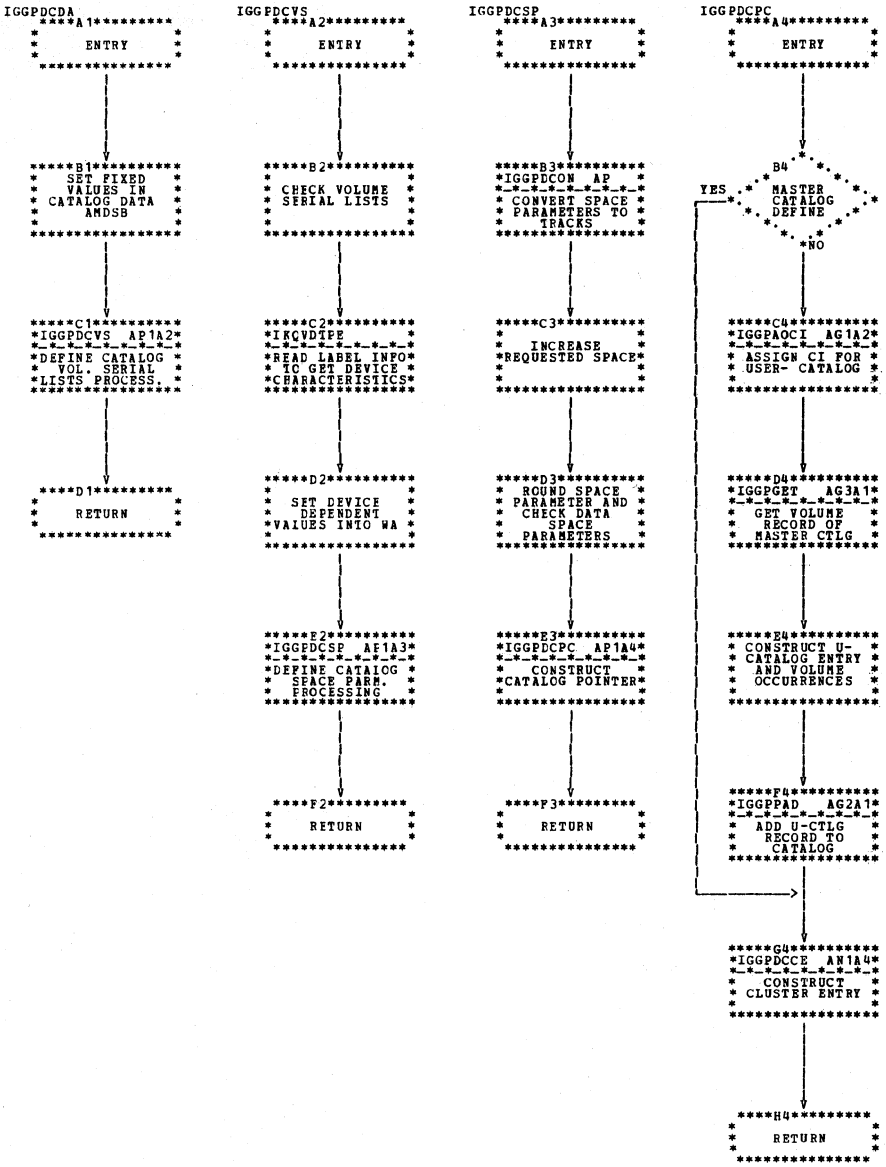


Chart AP1. CMS define - third module (IGG0CLAP)

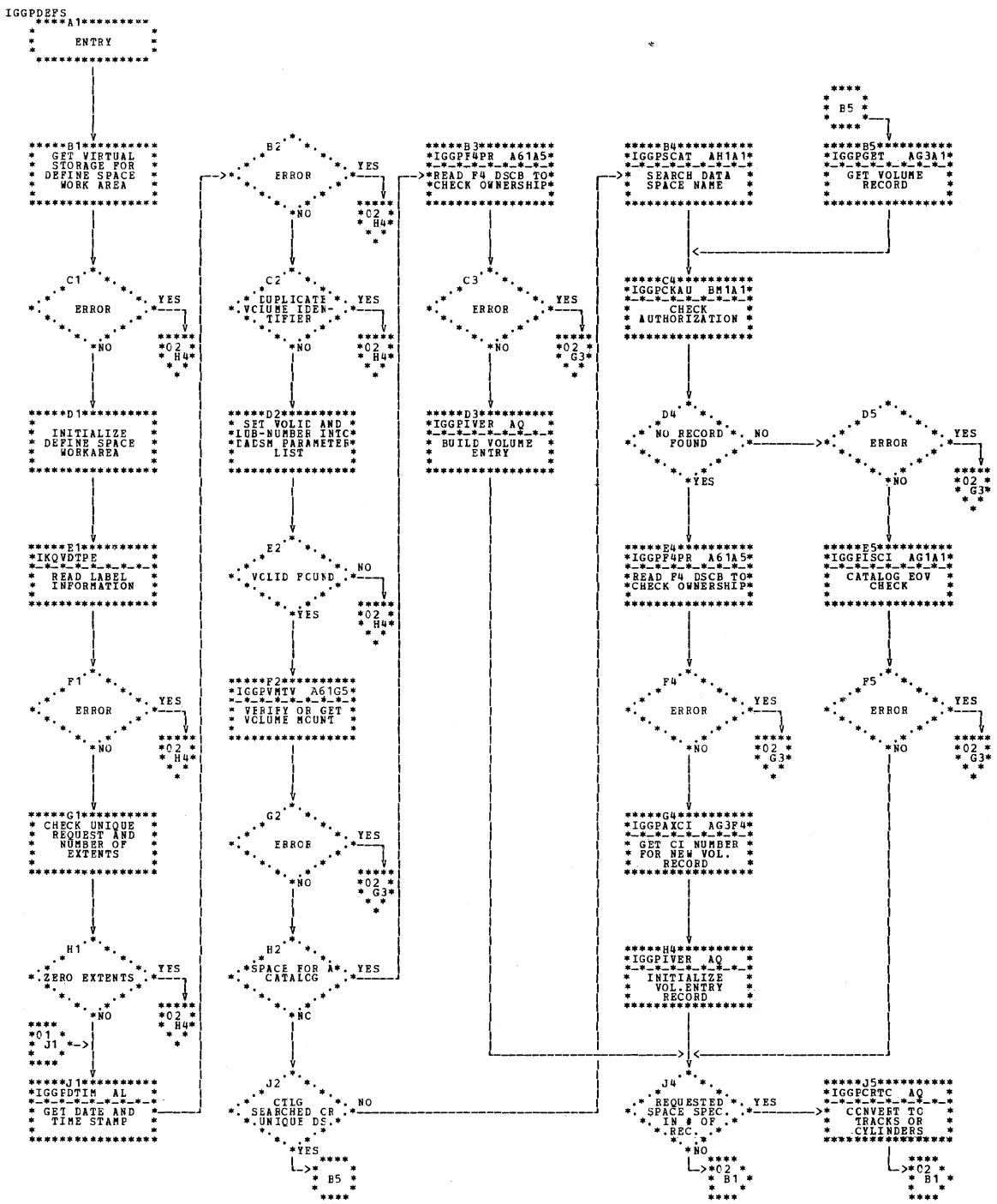


Chart AQ1. CMS define space - first load (IGG0CLAQ)

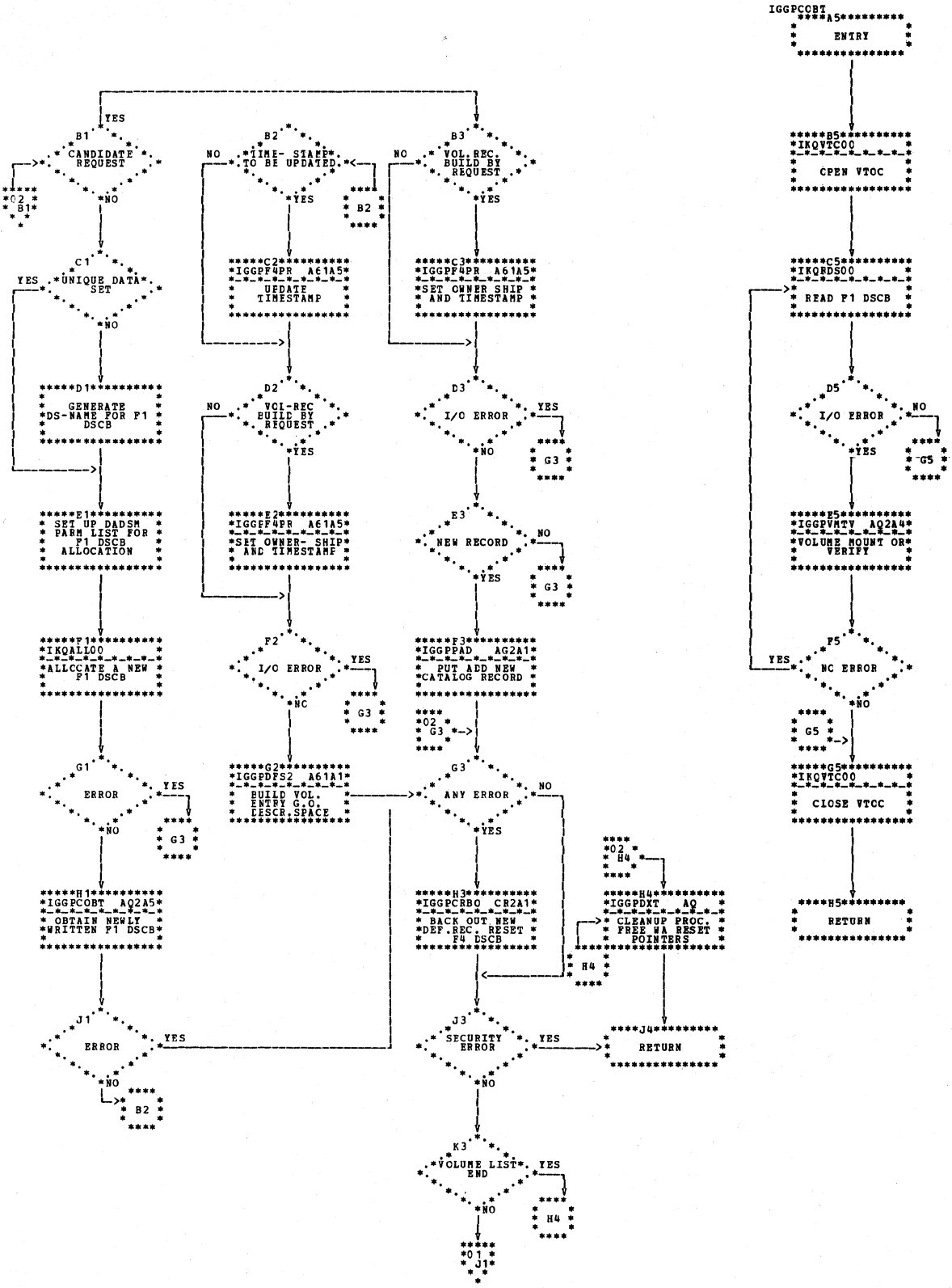


Chart AQ2. CMS define space - first load (IGG0CLAQ)

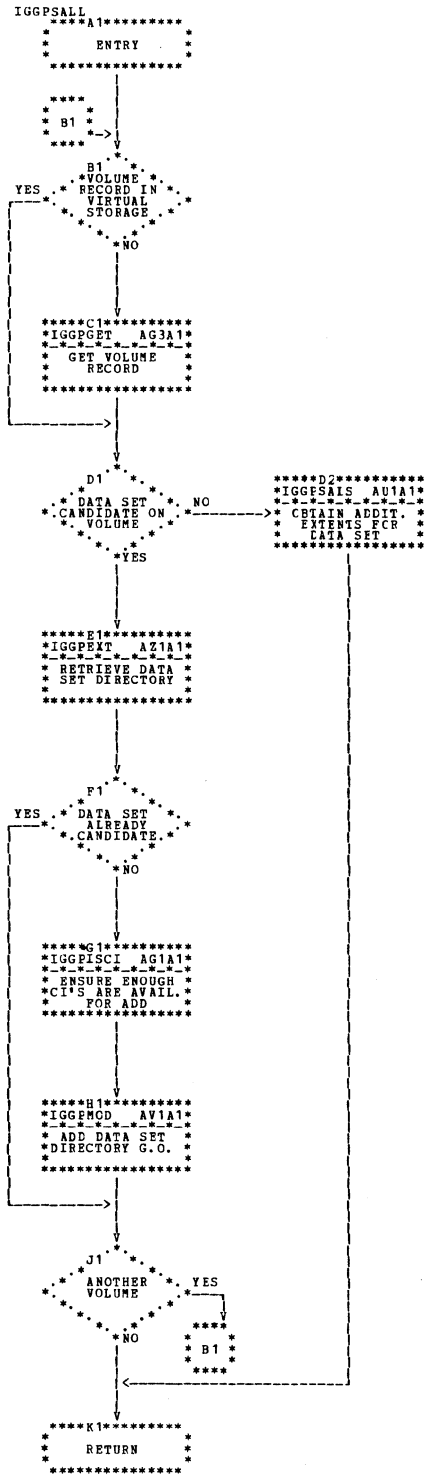


Chart AR1. Space suballocation (IGG0CLAR)

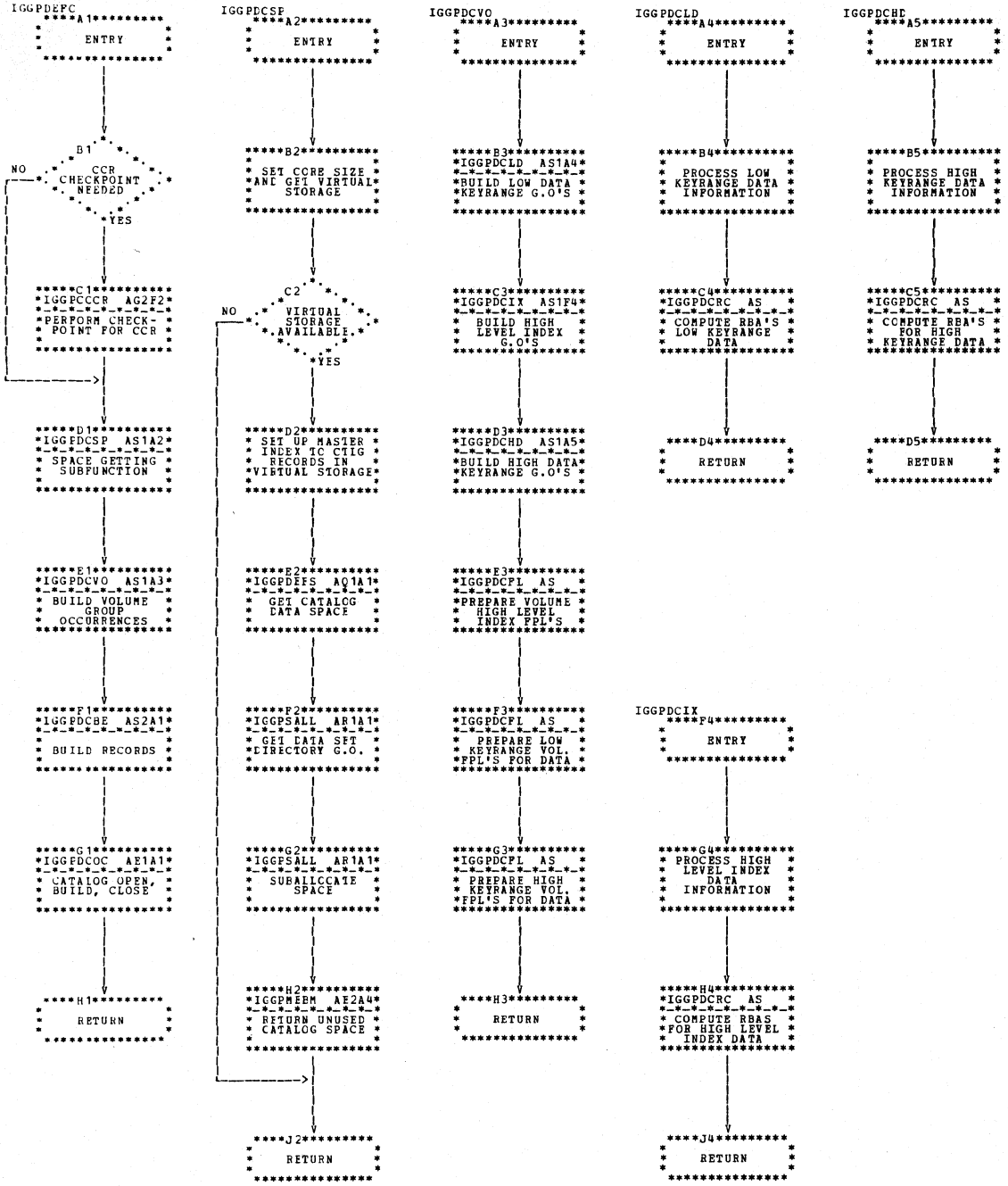


Chart AS1. Catalog definition processing (IGG0CLAS)

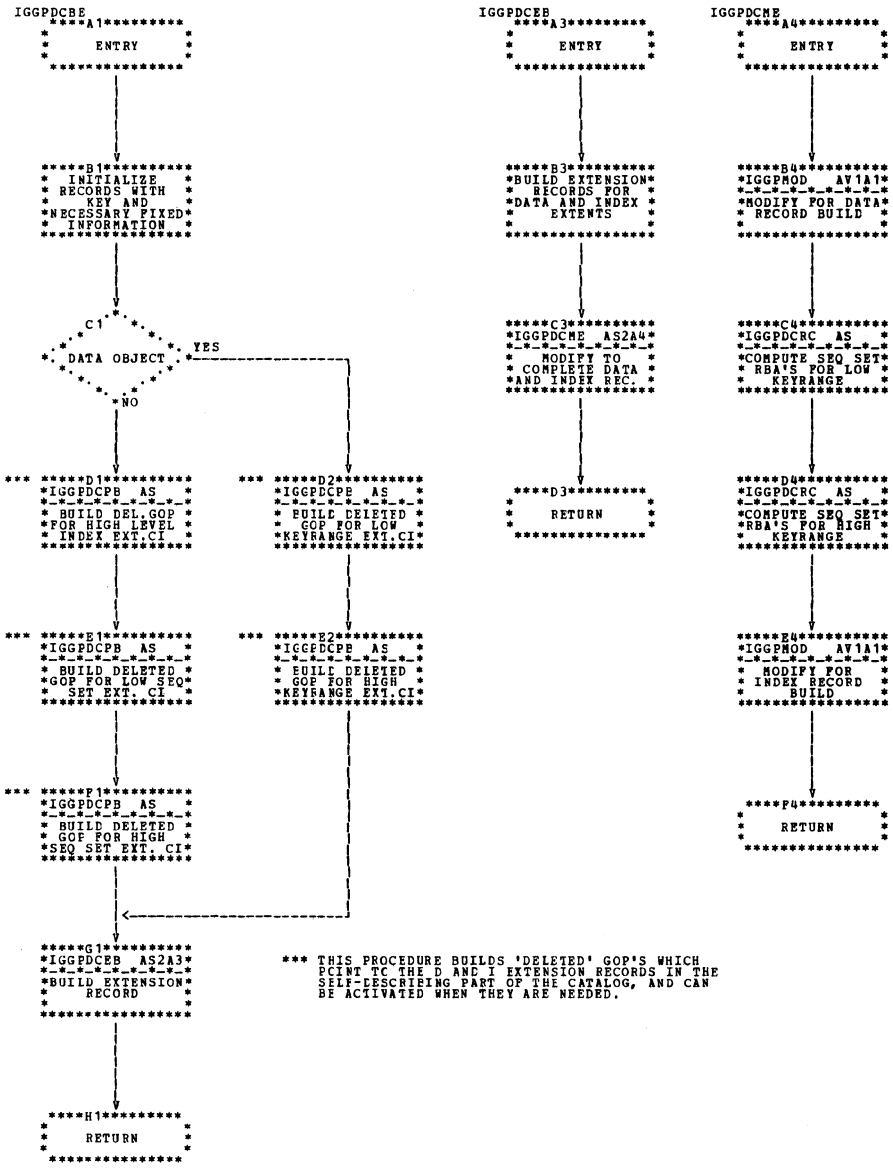


Chart AS2. Catalog definition processing (IGG0CLAS)

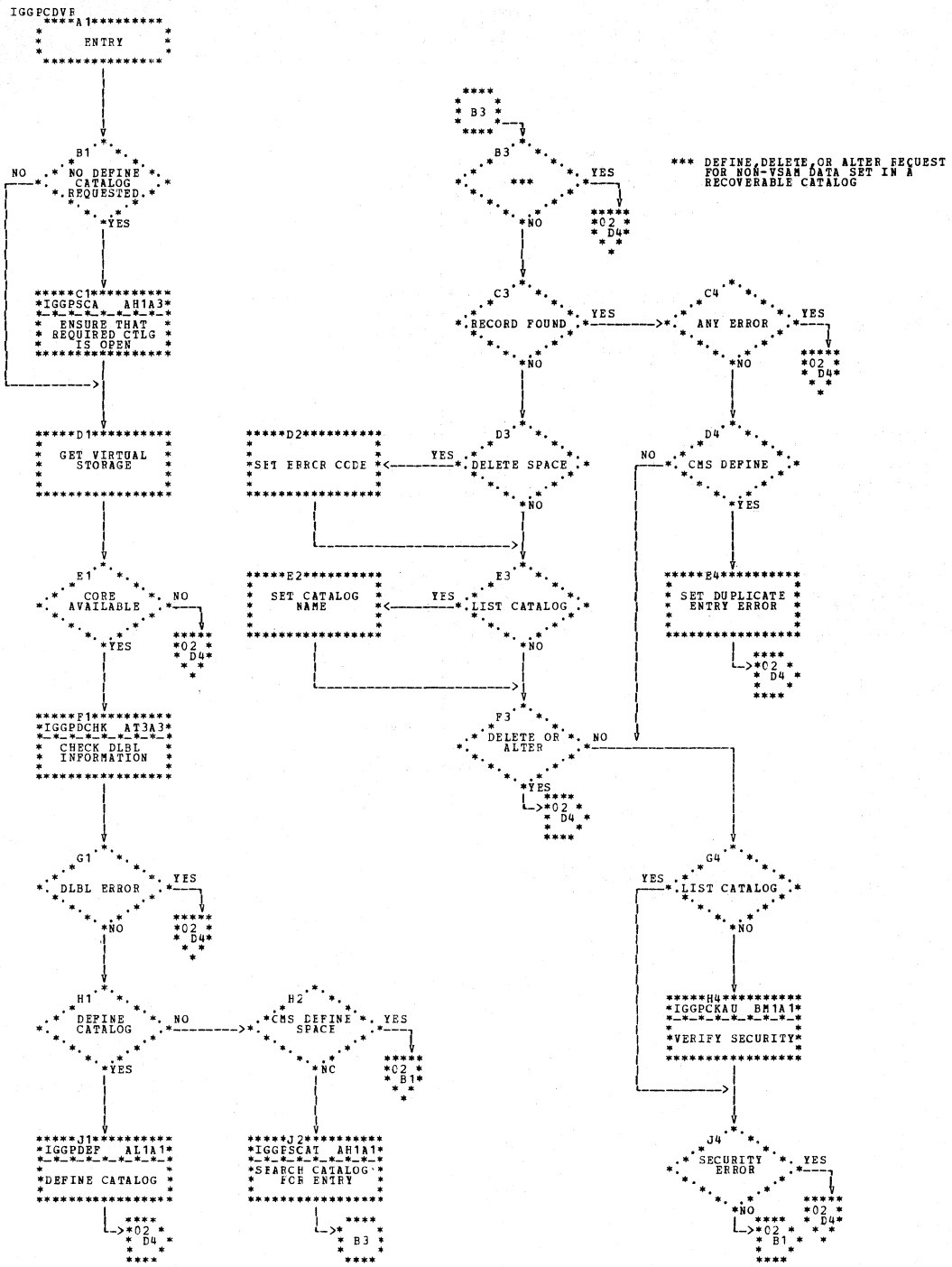


Chart AT1. CMS driver (IGG0CLAT)

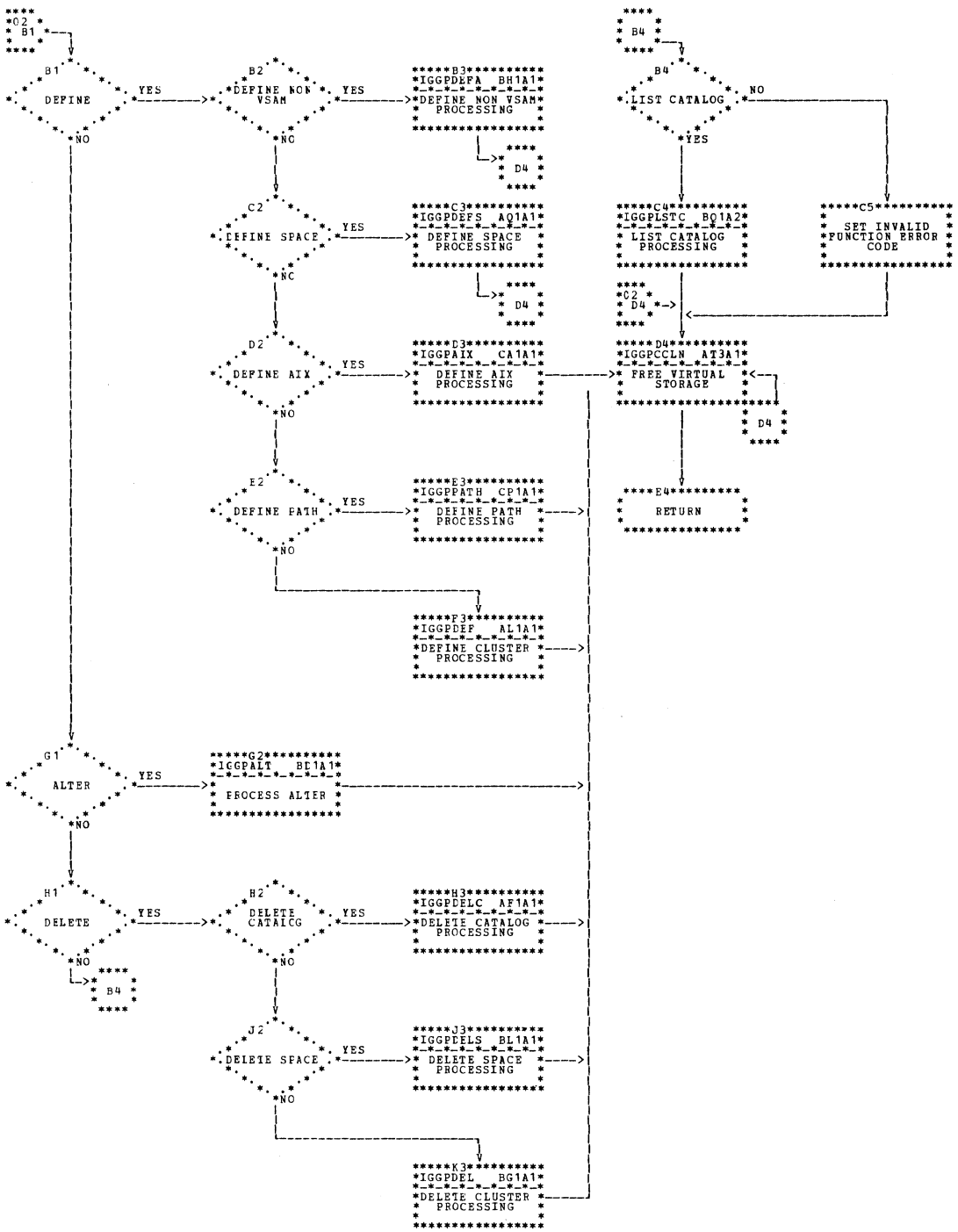


Chart AT2. CMS driver (IGG0CLAT)

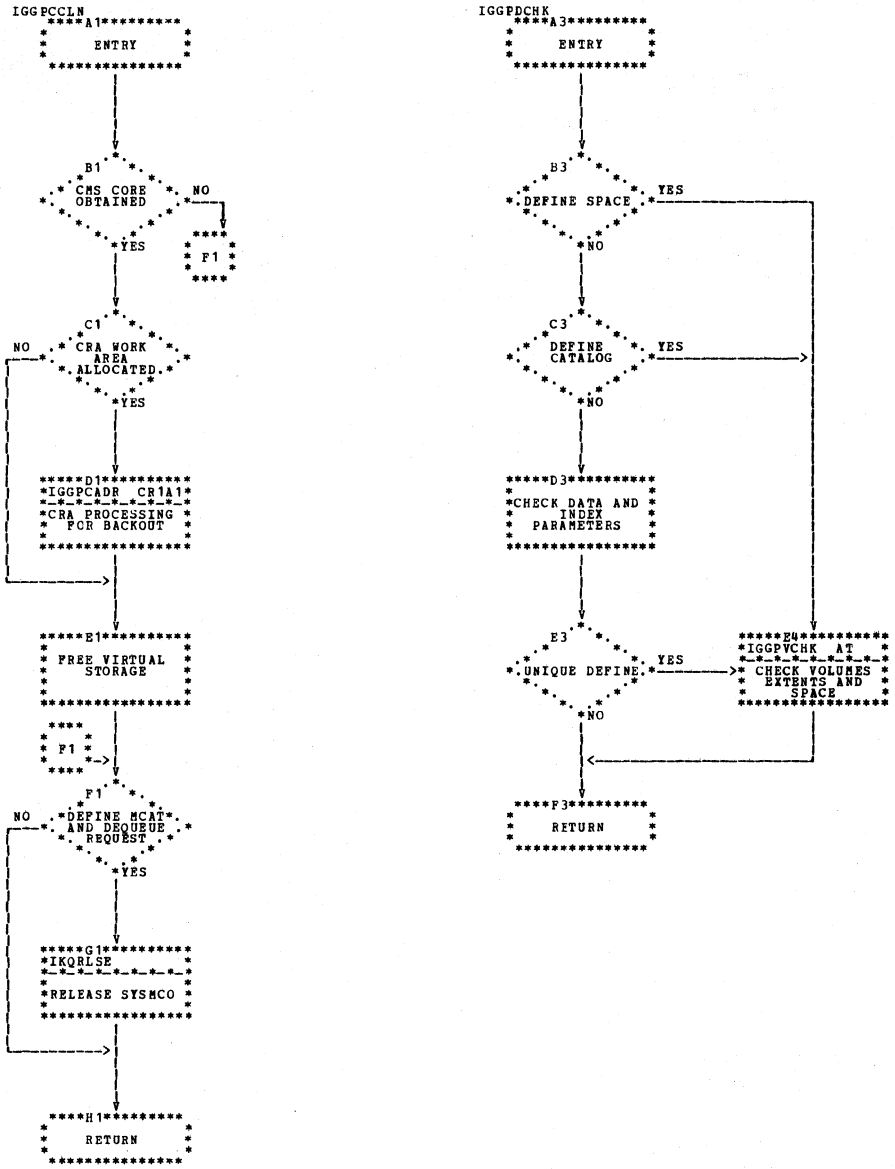


Chart AT3. CMS driver (IGG0CLAT)

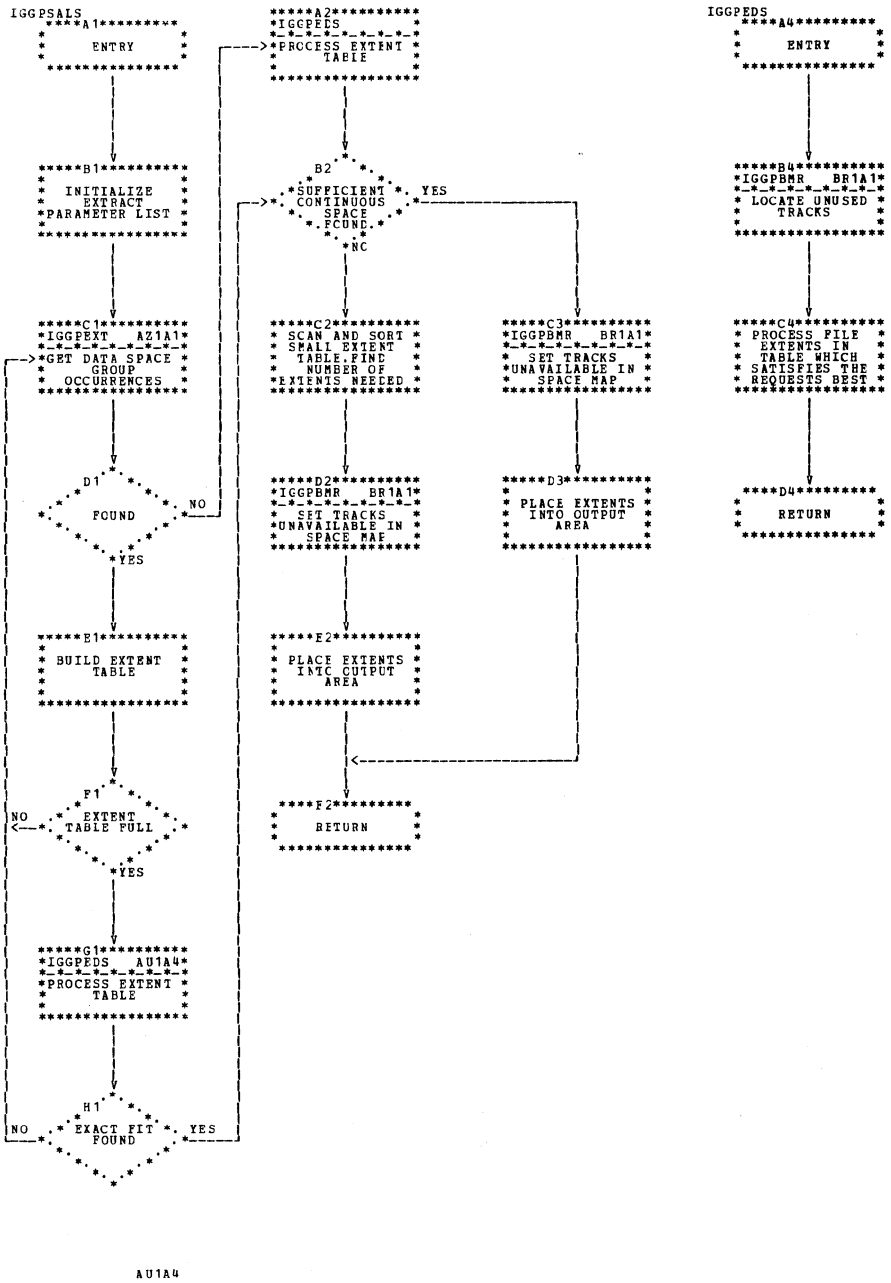


Chart AU1. Suballocation routine (IGG0CLAU)

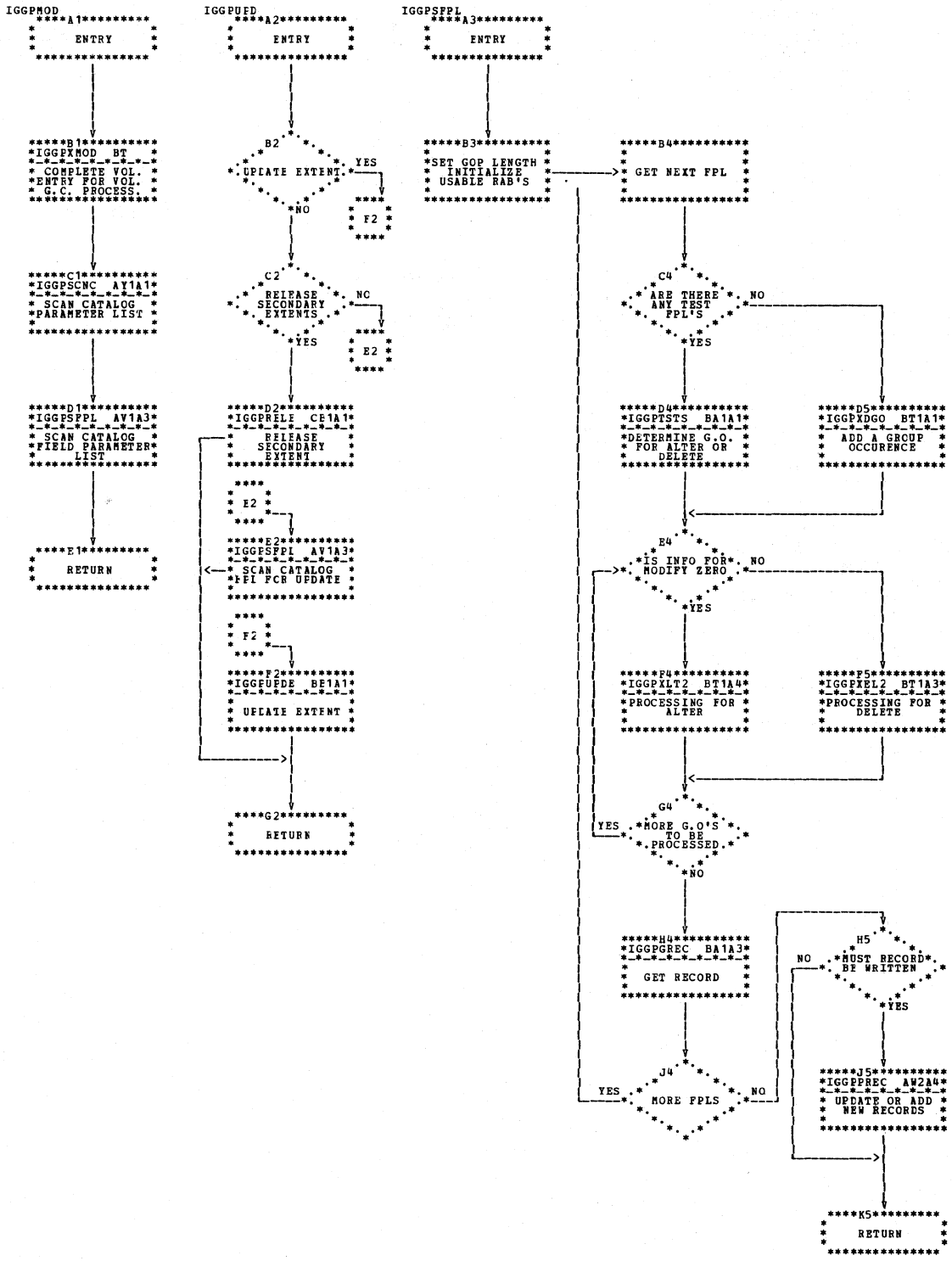


Chart AV1. Modify catalog field (IGG0CLAV)

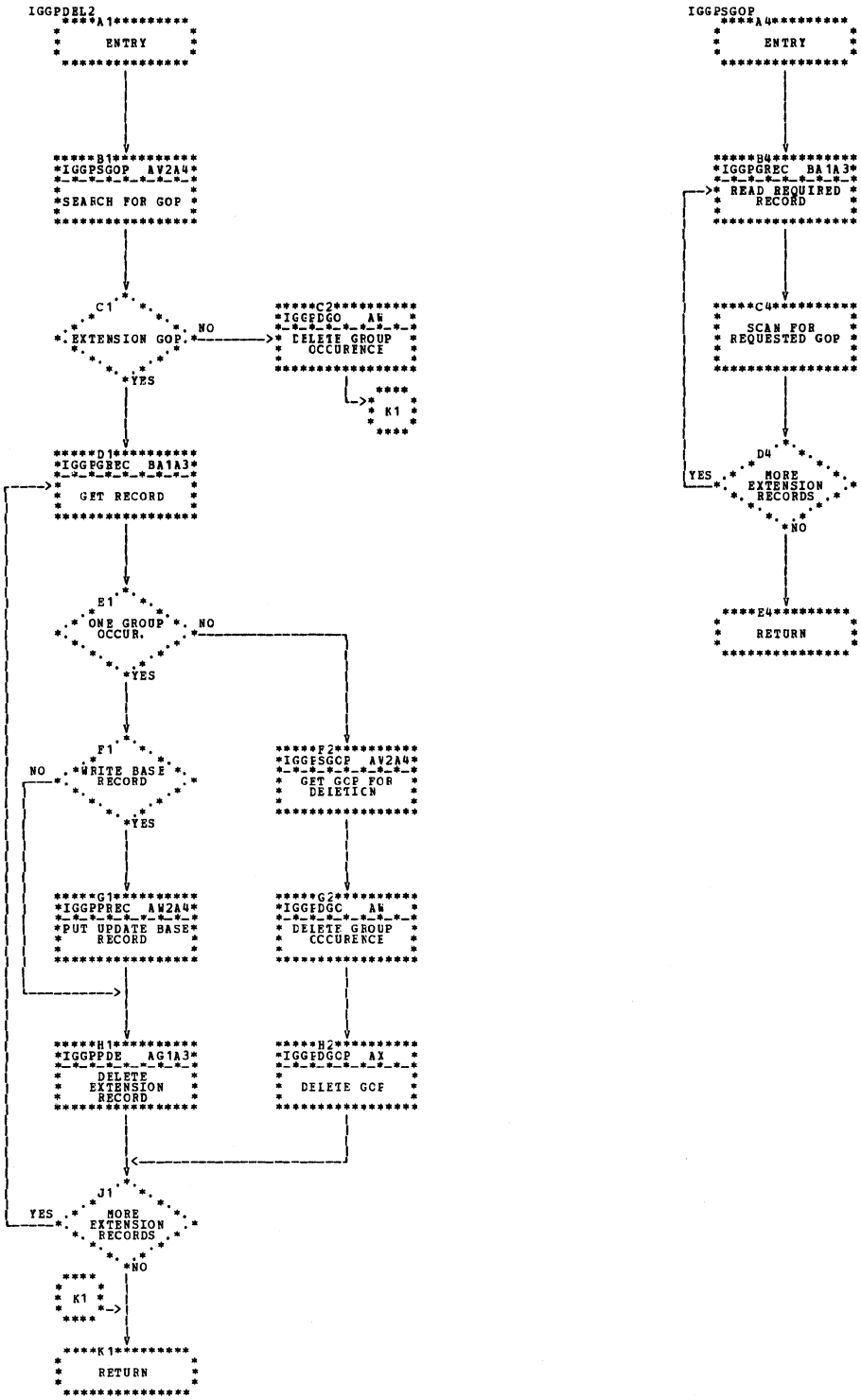


Chart AV2. Modify catalog field (IGG0CLAV)

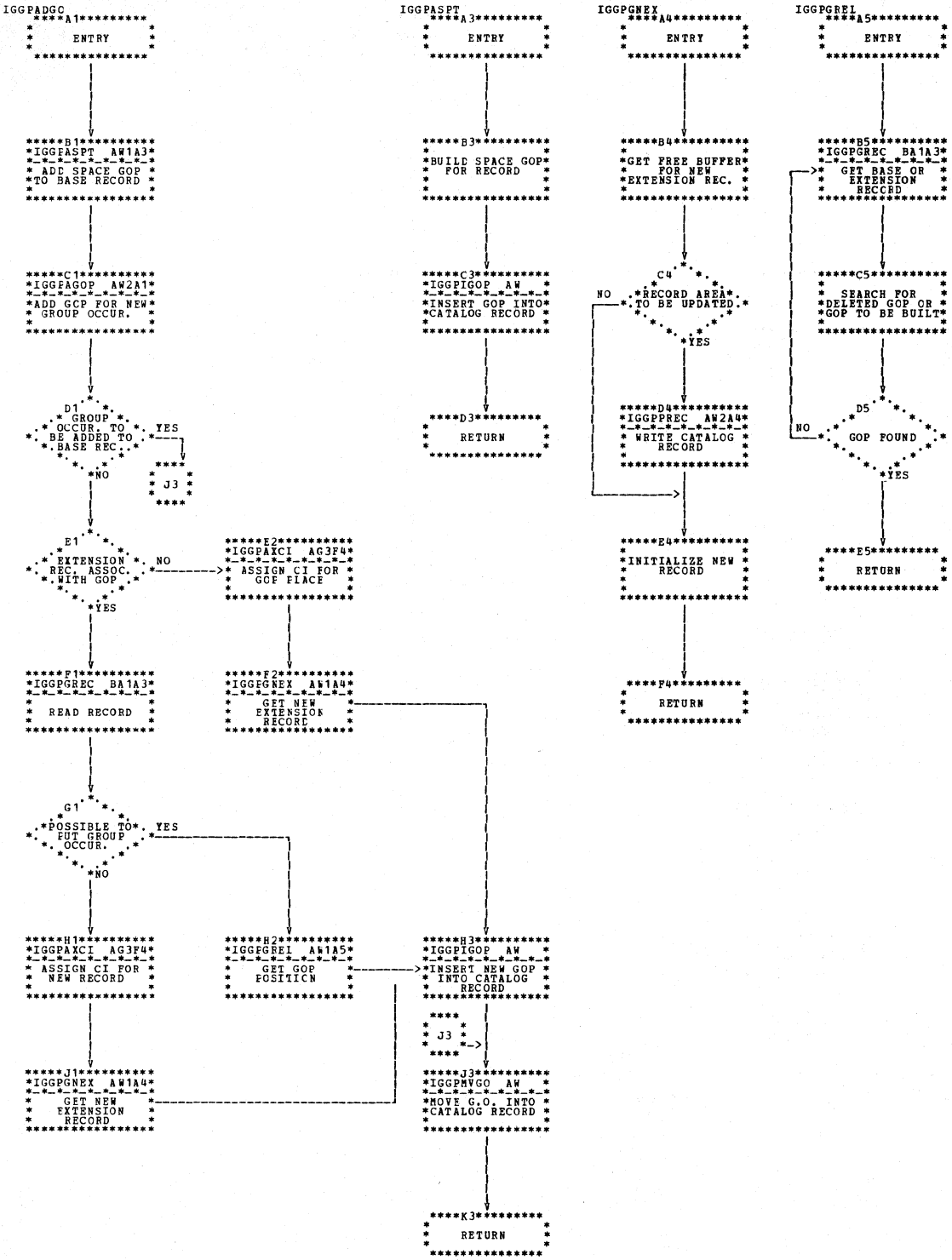


Chart AW1. Add group occurrence (IGG0CLAW)

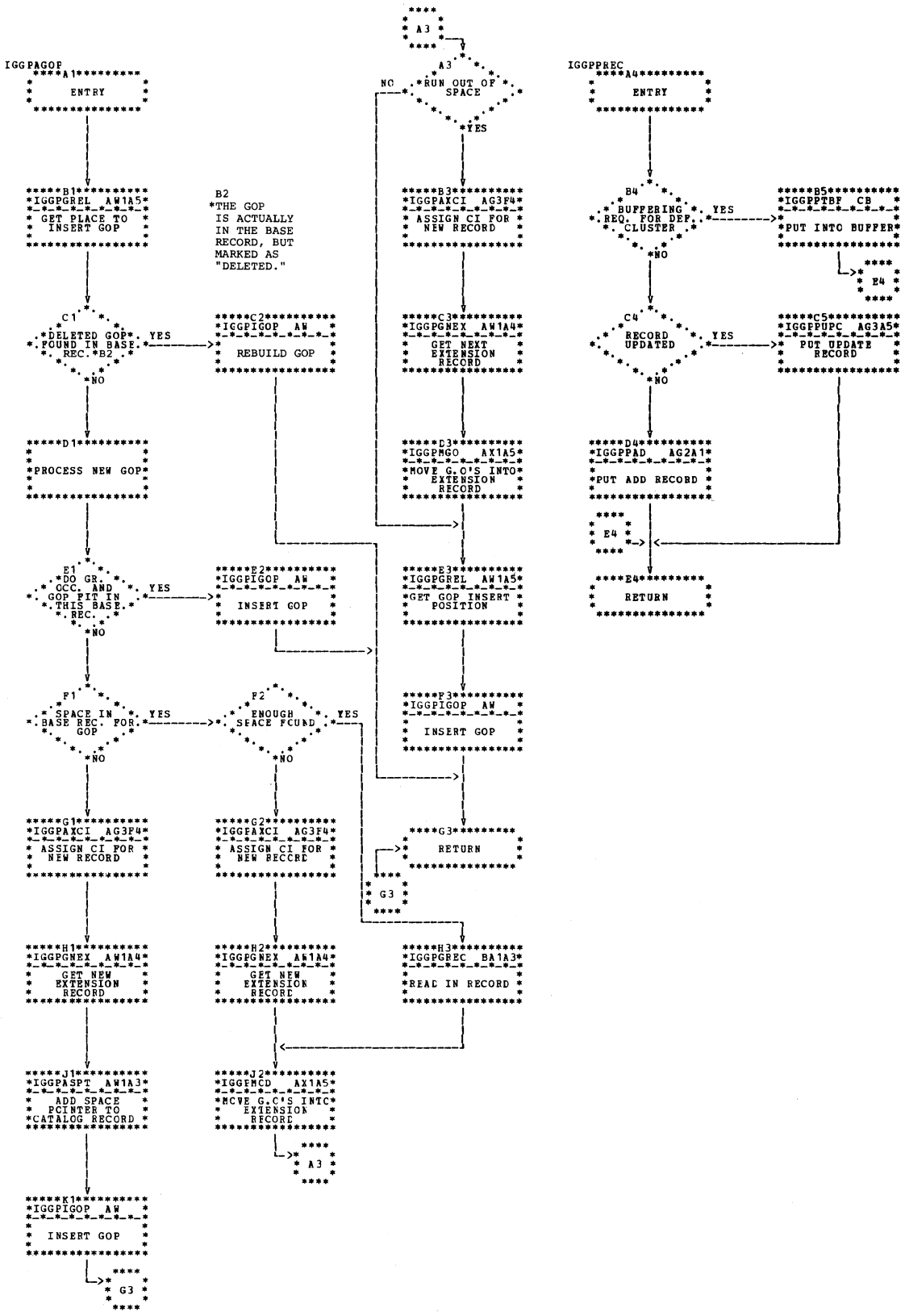


Chart AW2. Add group occurrence (IGG0CLAW)

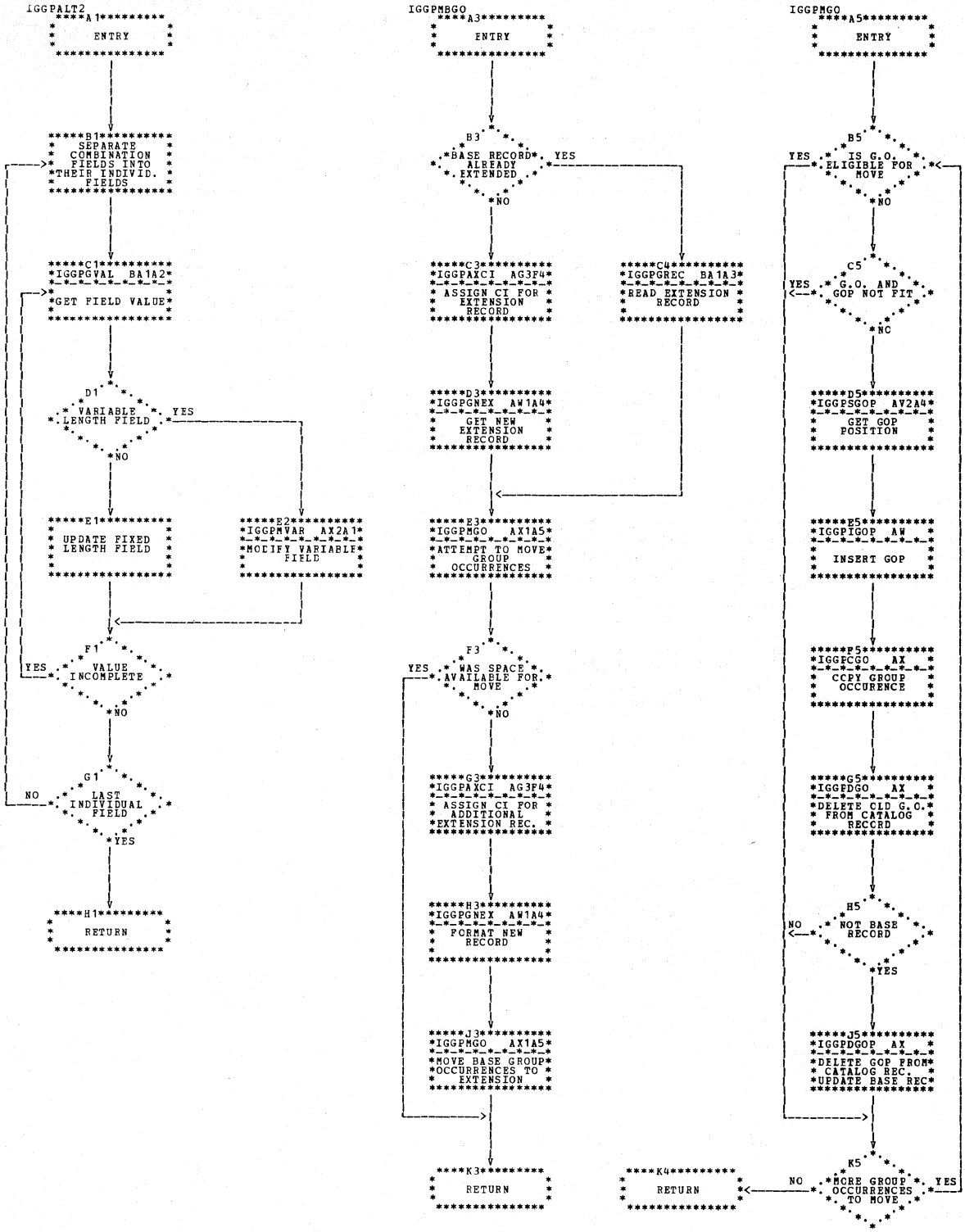


Chart AX1. Alter catalog field (IGG0CLAX)

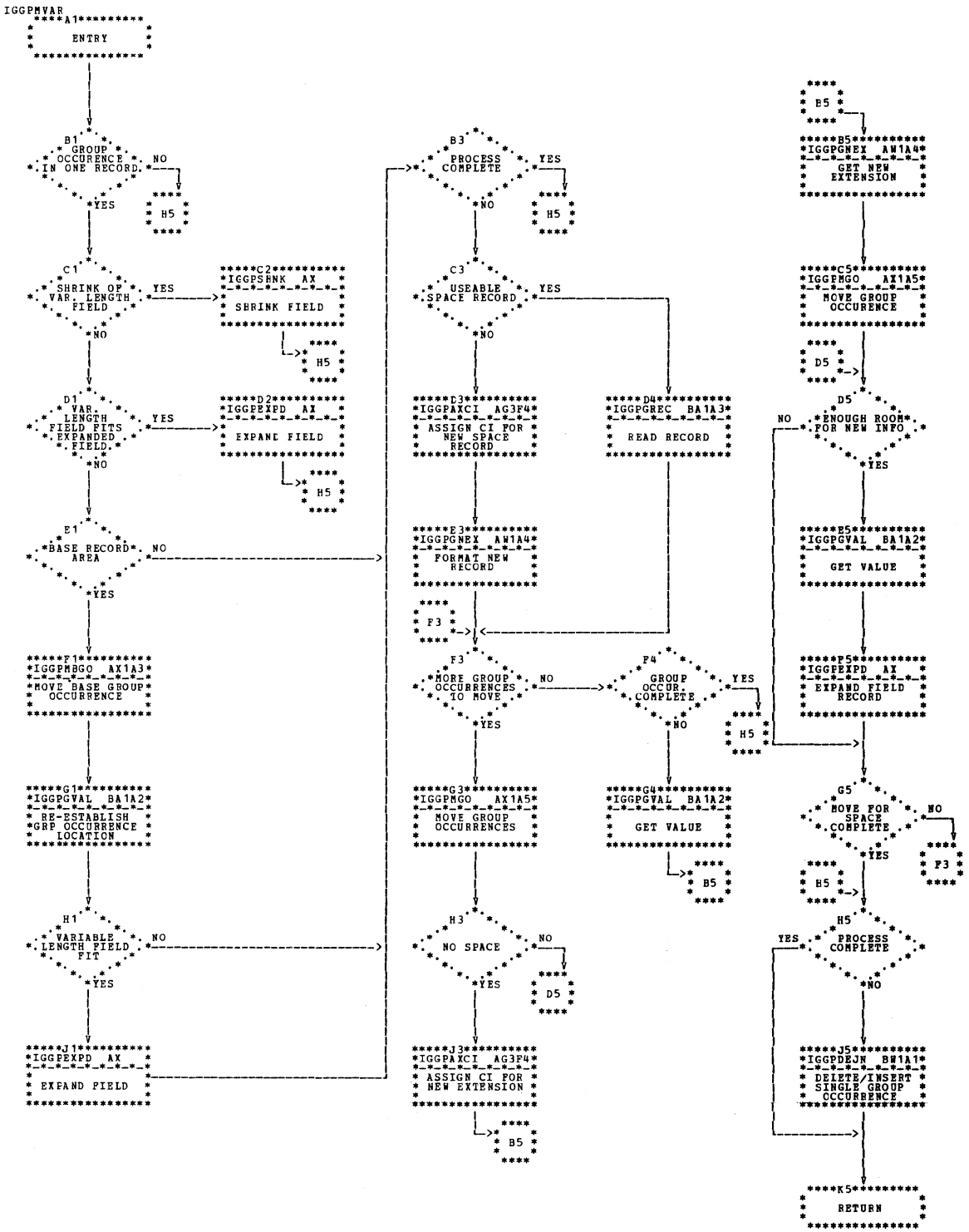


Chart AX2. Alter catalog field (IGG0CLAX)

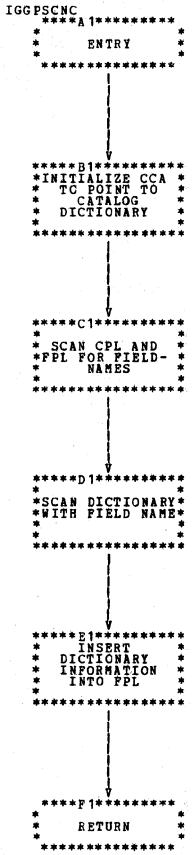


Chart AY1. Scan catalog parameter list (IGG0CLAY)

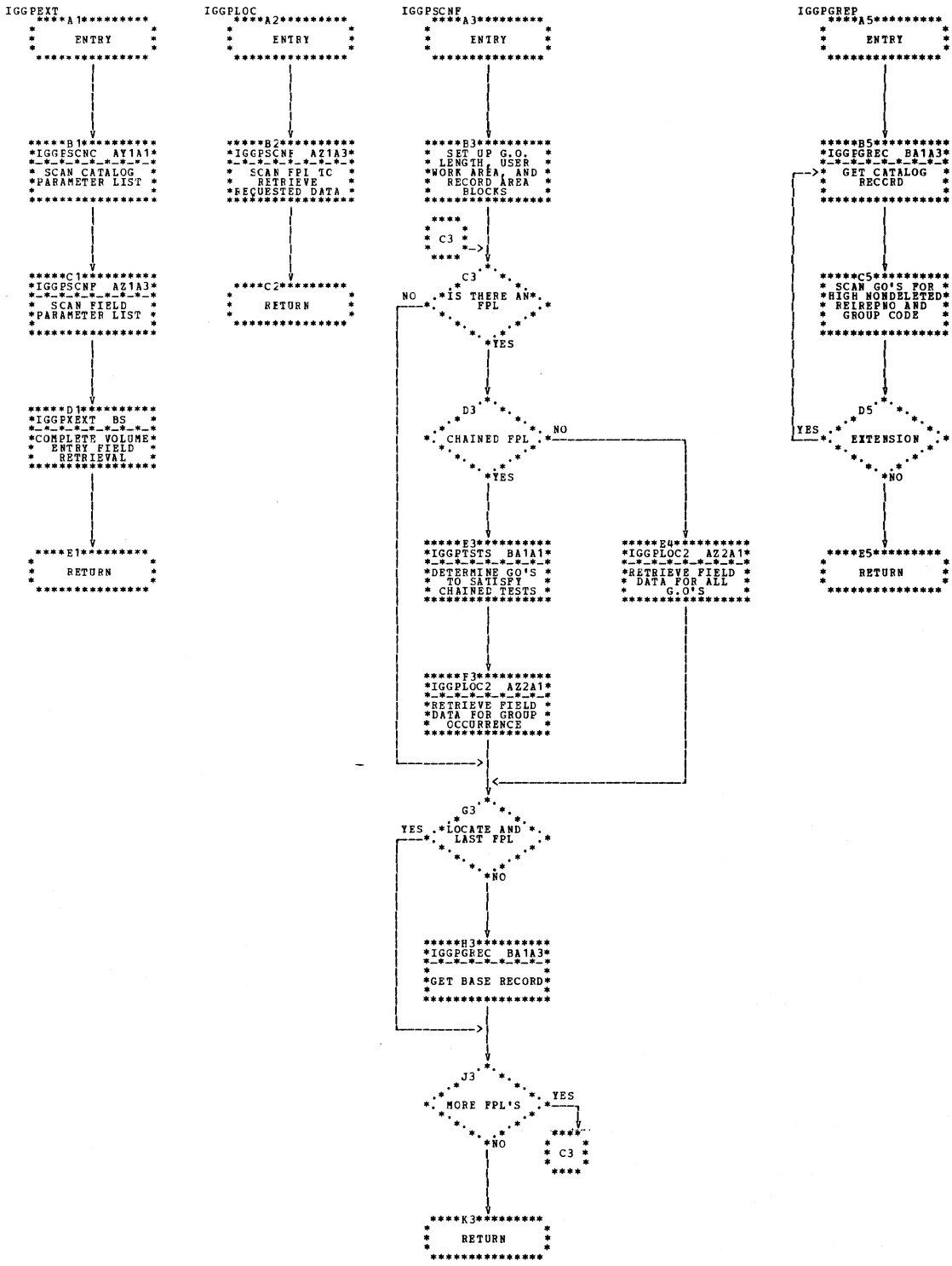


Chart AZ1. Extract catalog field (IGG0CLAZ)

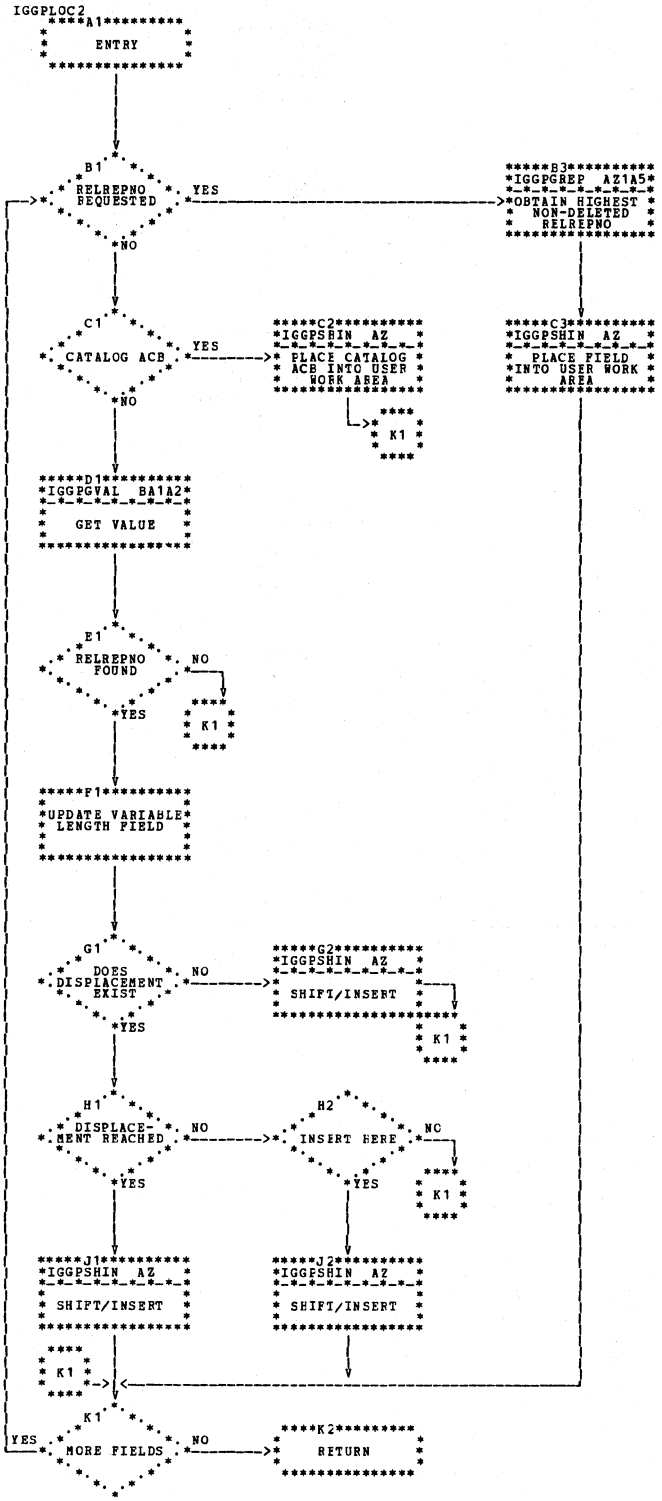


Chart AZ2. Extract catalog field (IGG0CLAZ)

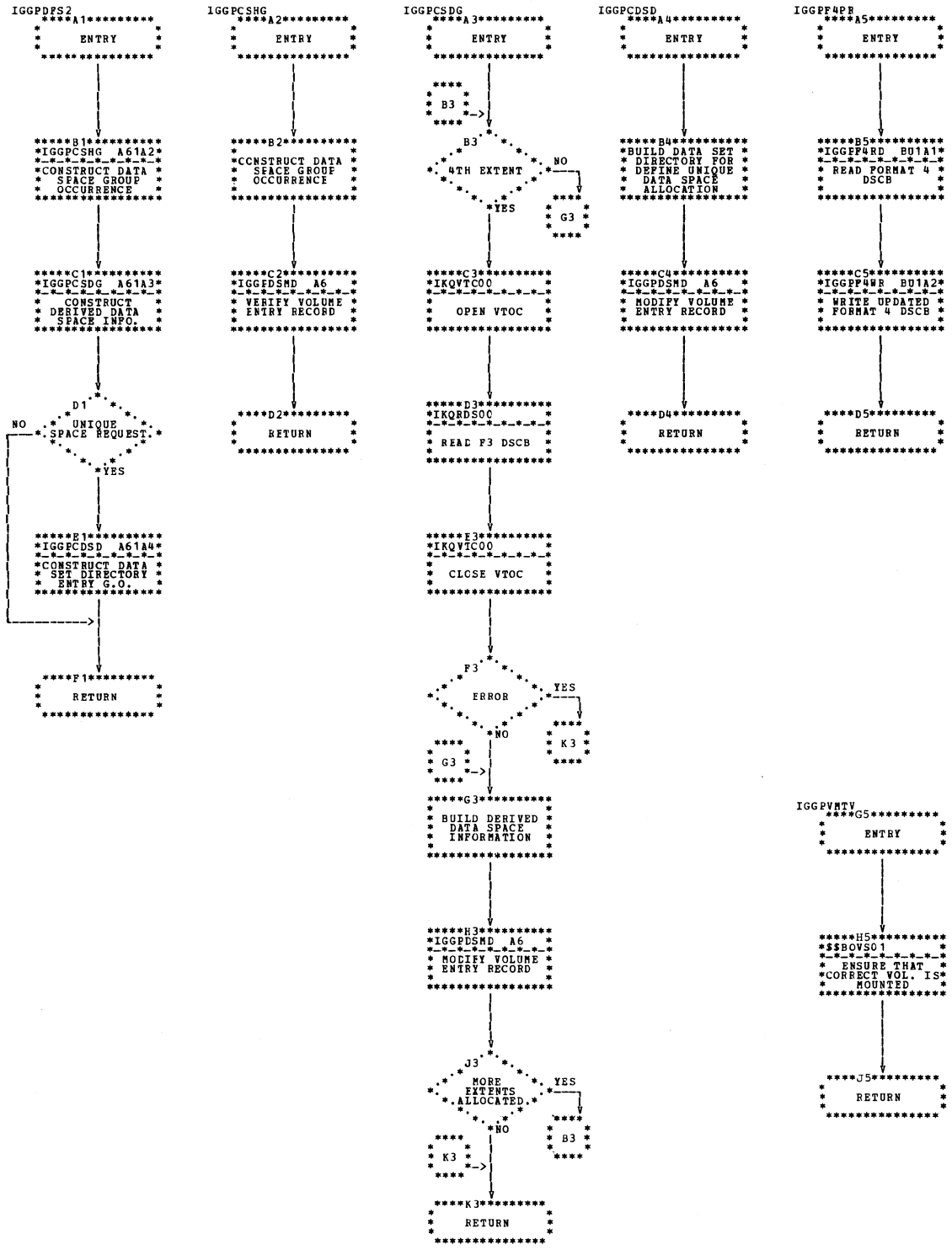


Chart A61. Define space - second load (IGG0CLA6)

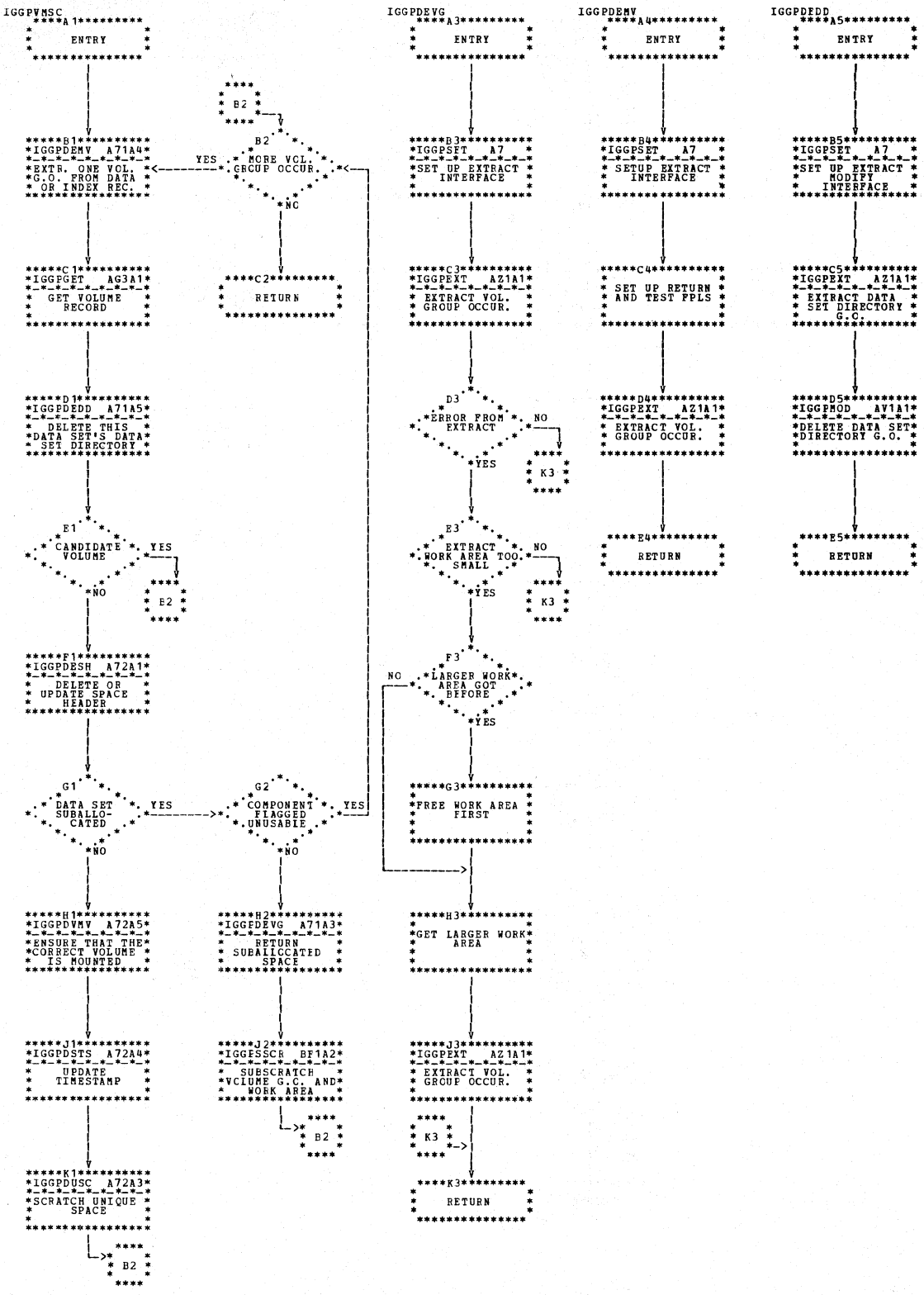


Chart A71. CMS delete - second module (IGG0CLA7)

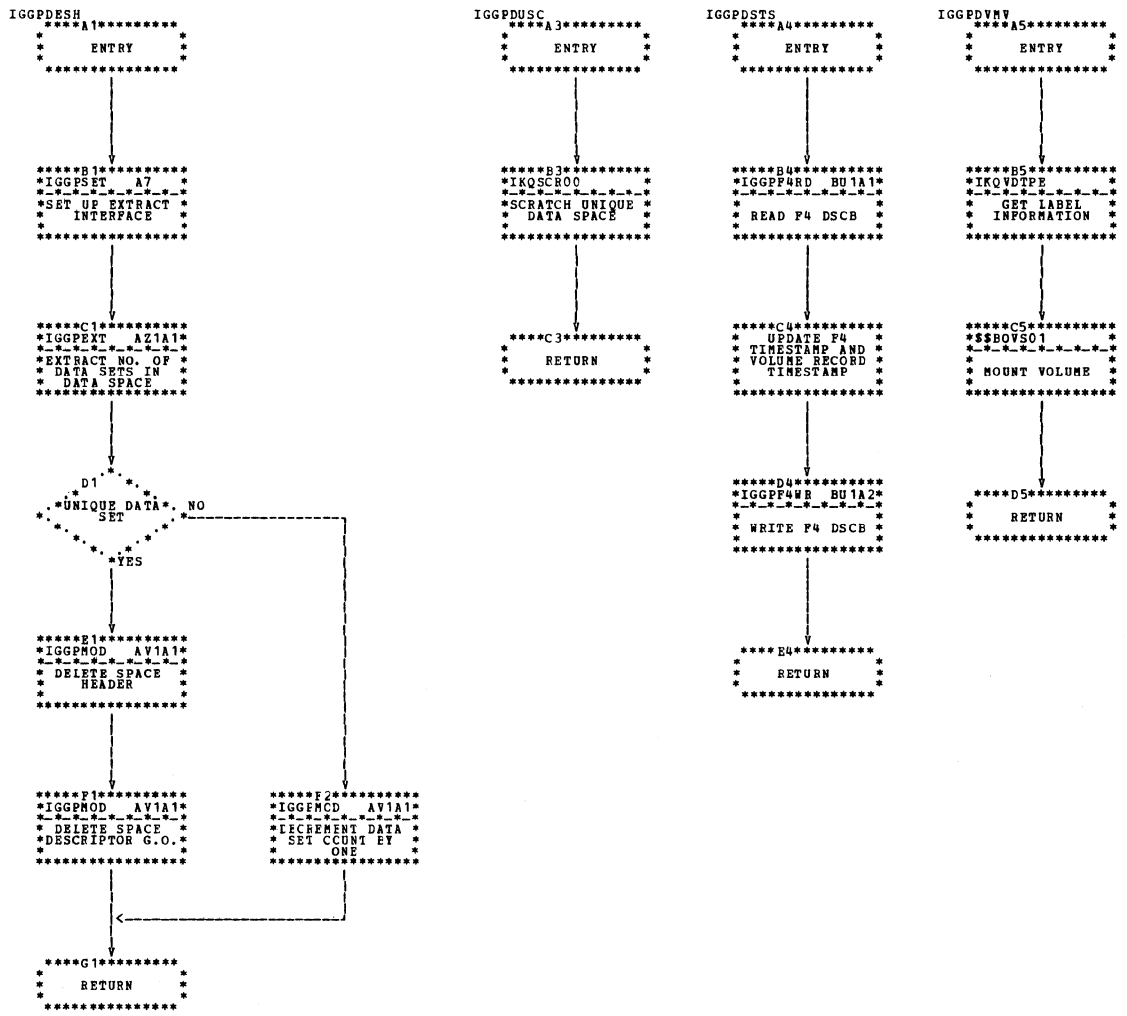


Chart A72. CMS delete - second module (IGG0CLA7)

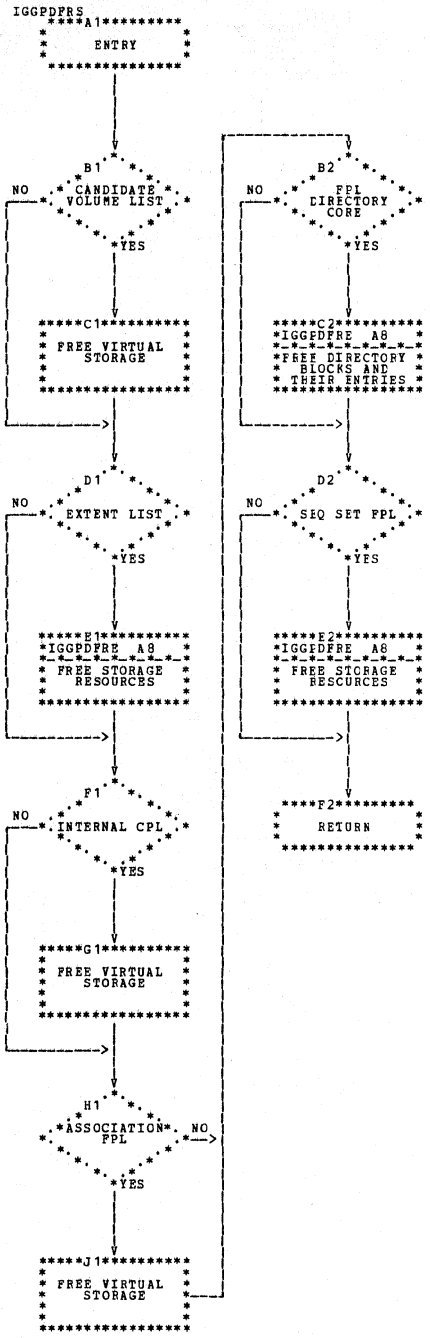


Chart A81. Clean up of storage resources (IGG0CLA8)

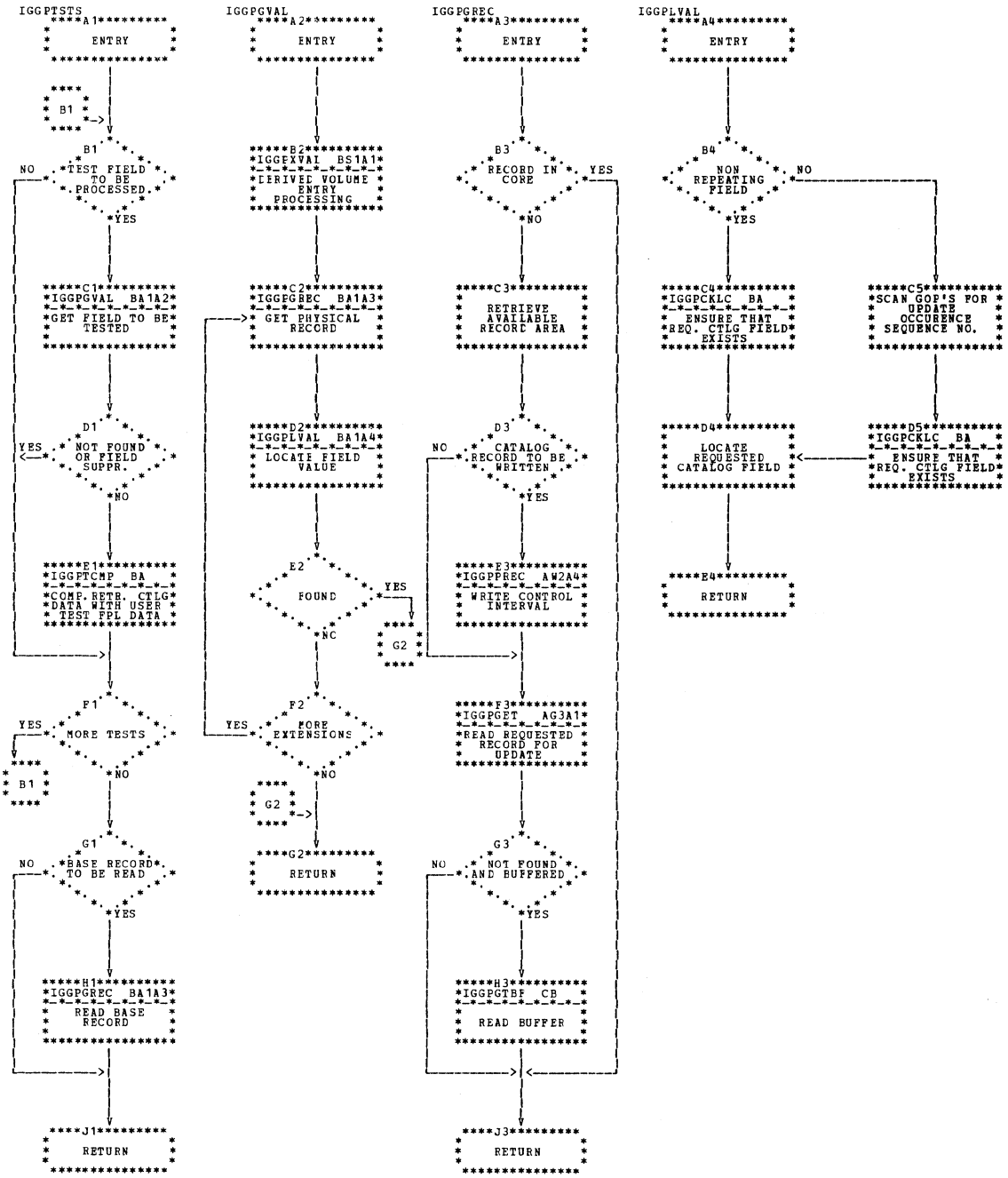


Chart BA1. Tests (IGG0CLBA)

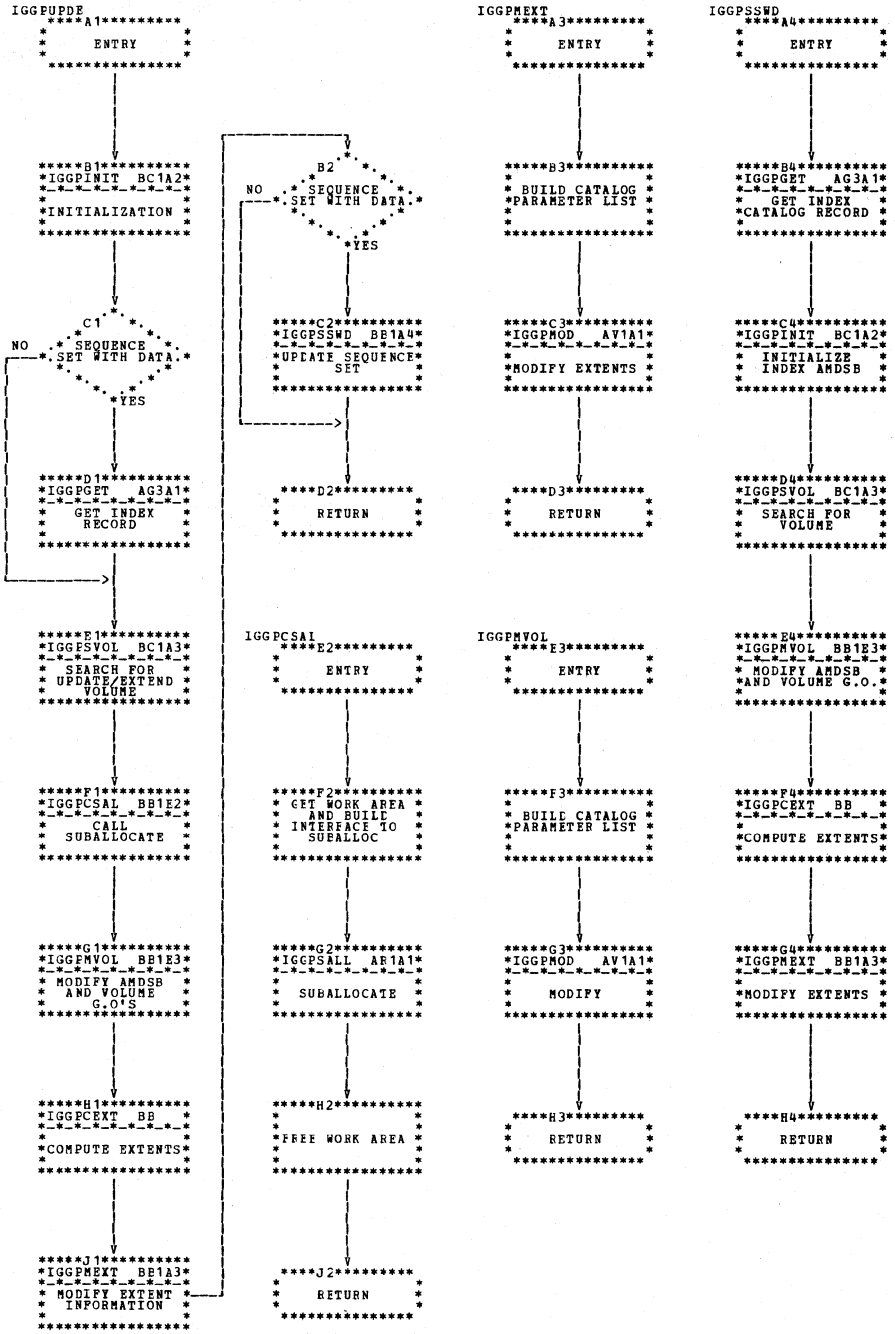


Chart BB1. Update extend (IGG0CLBB)

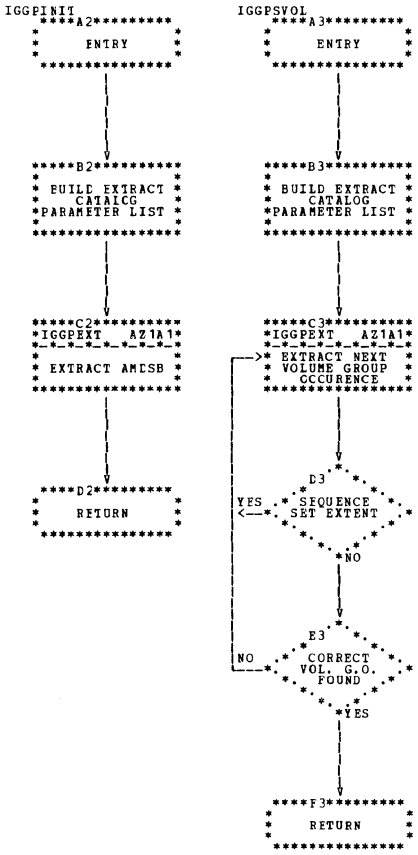


Chart BC1. Update extend initialization (IGG0CLBC)

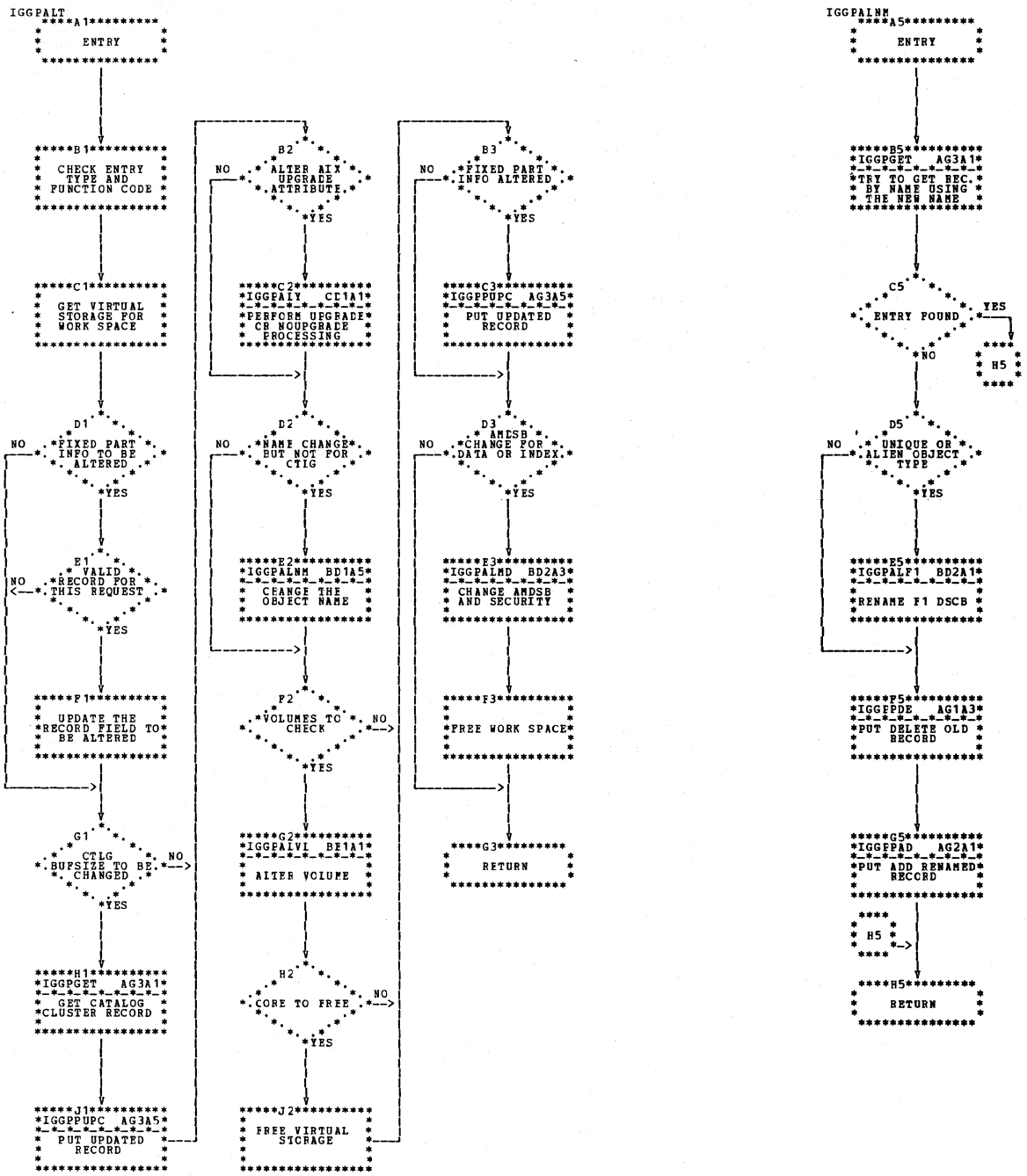


Chart BD1. CMS alter - first module (IGG0CLBD)

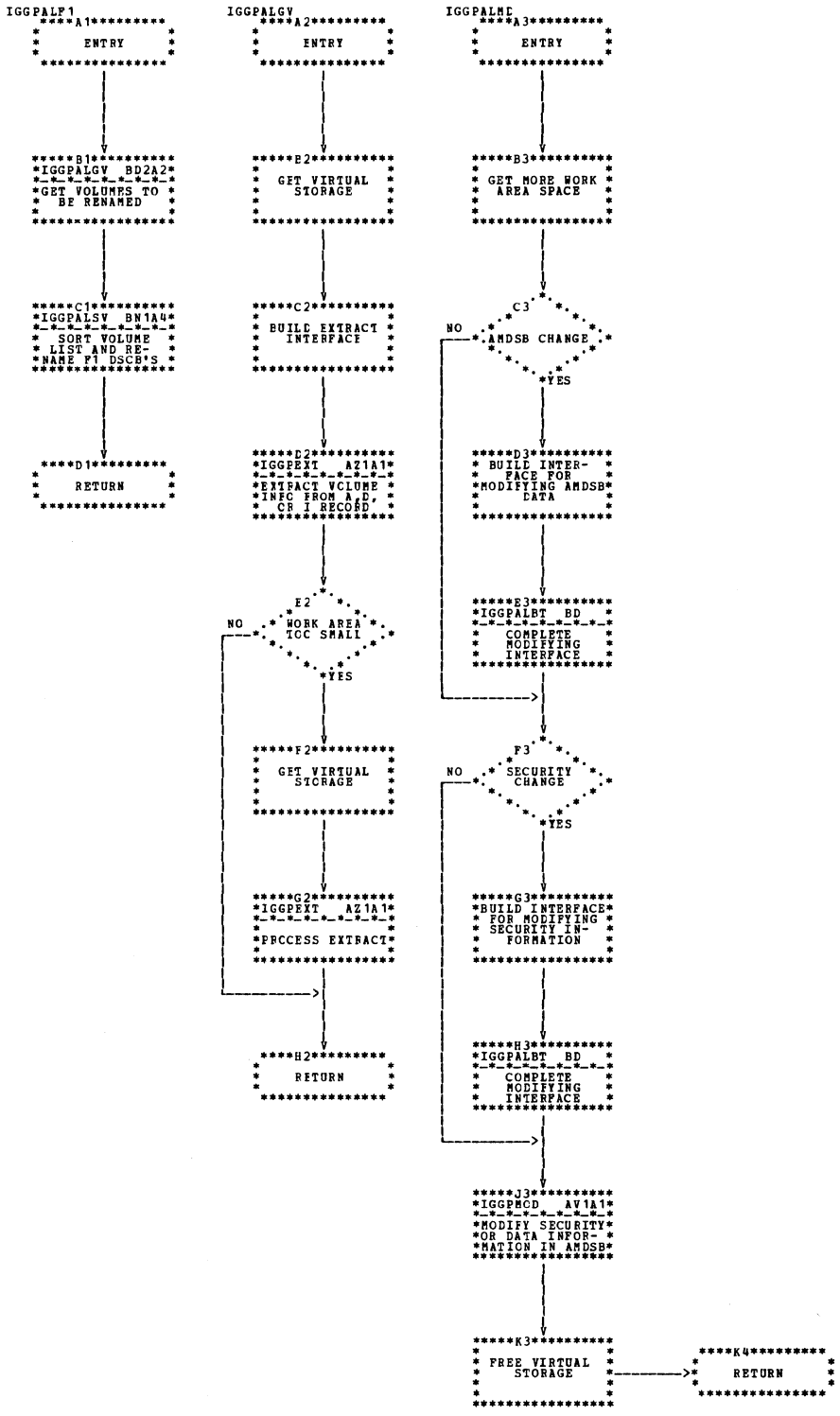


Chart BD2. CMS alter - first module (IGG0CLBD)

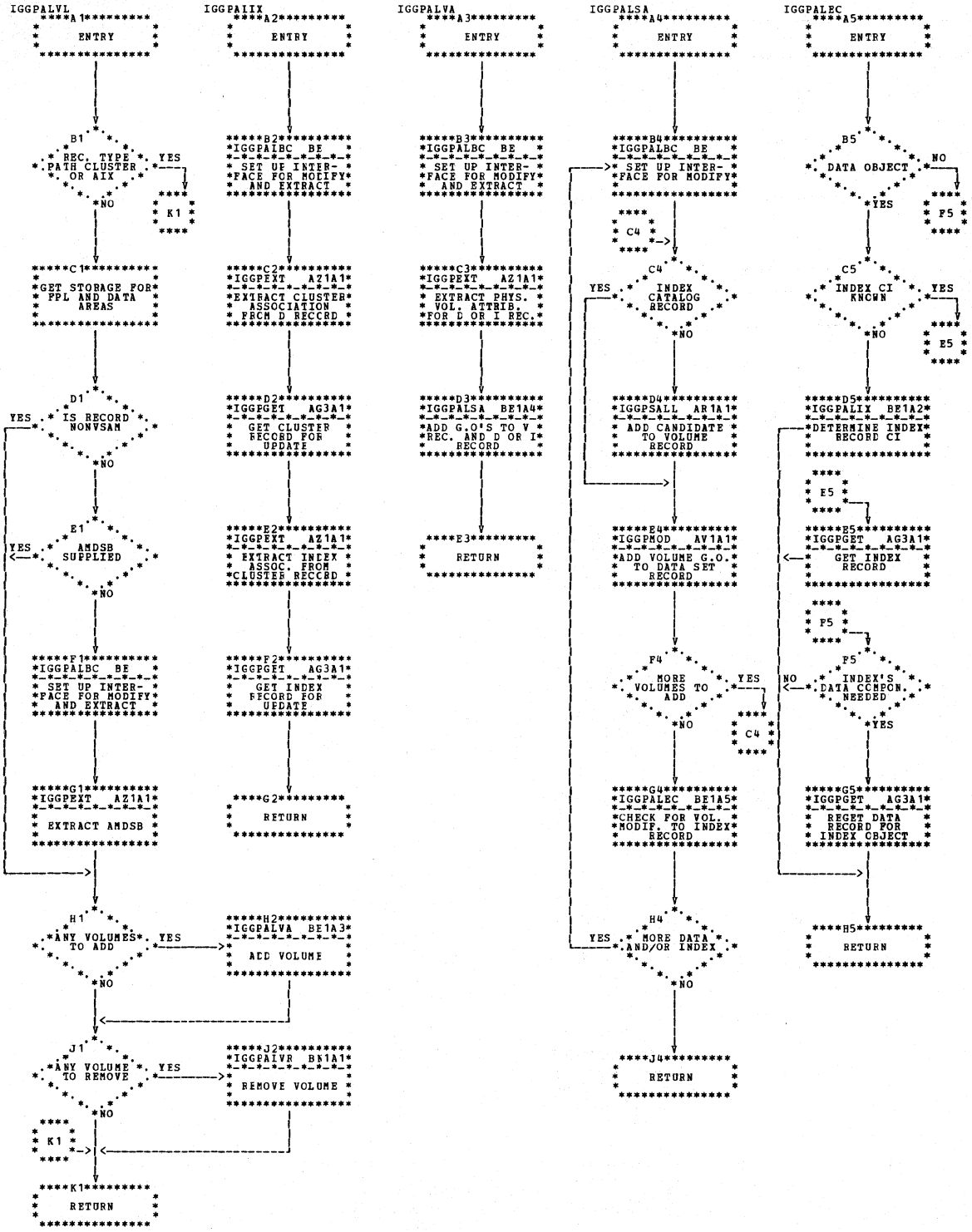


Chart BE1. CMS alter - third module (IGG0CLBE)

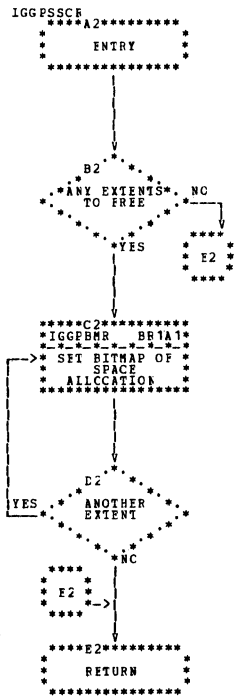


Chart BF1. Subscratch routine (IGG0CLBF)

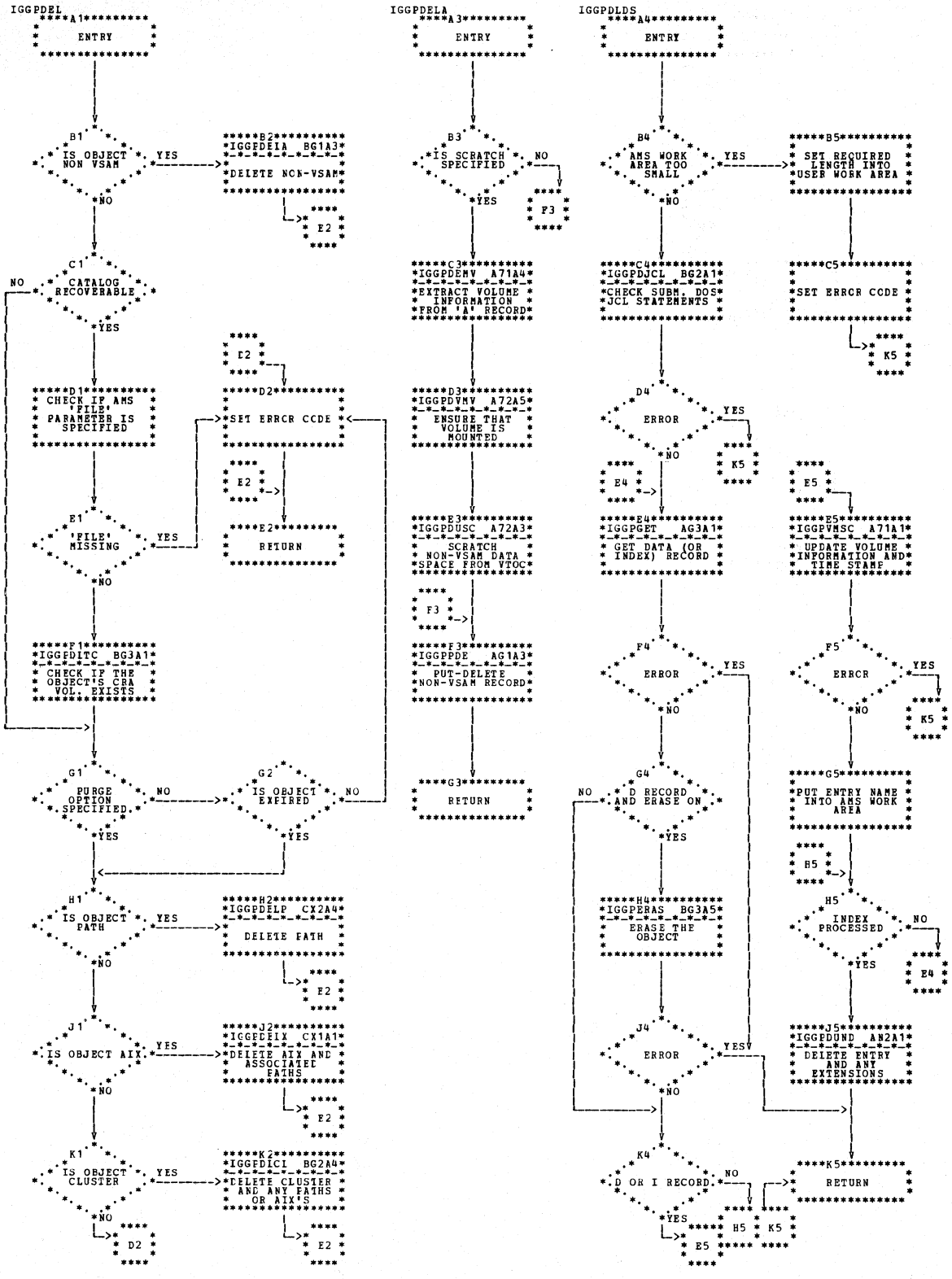


Chart BG1. CMS delete - first module (IGG0CLBG)

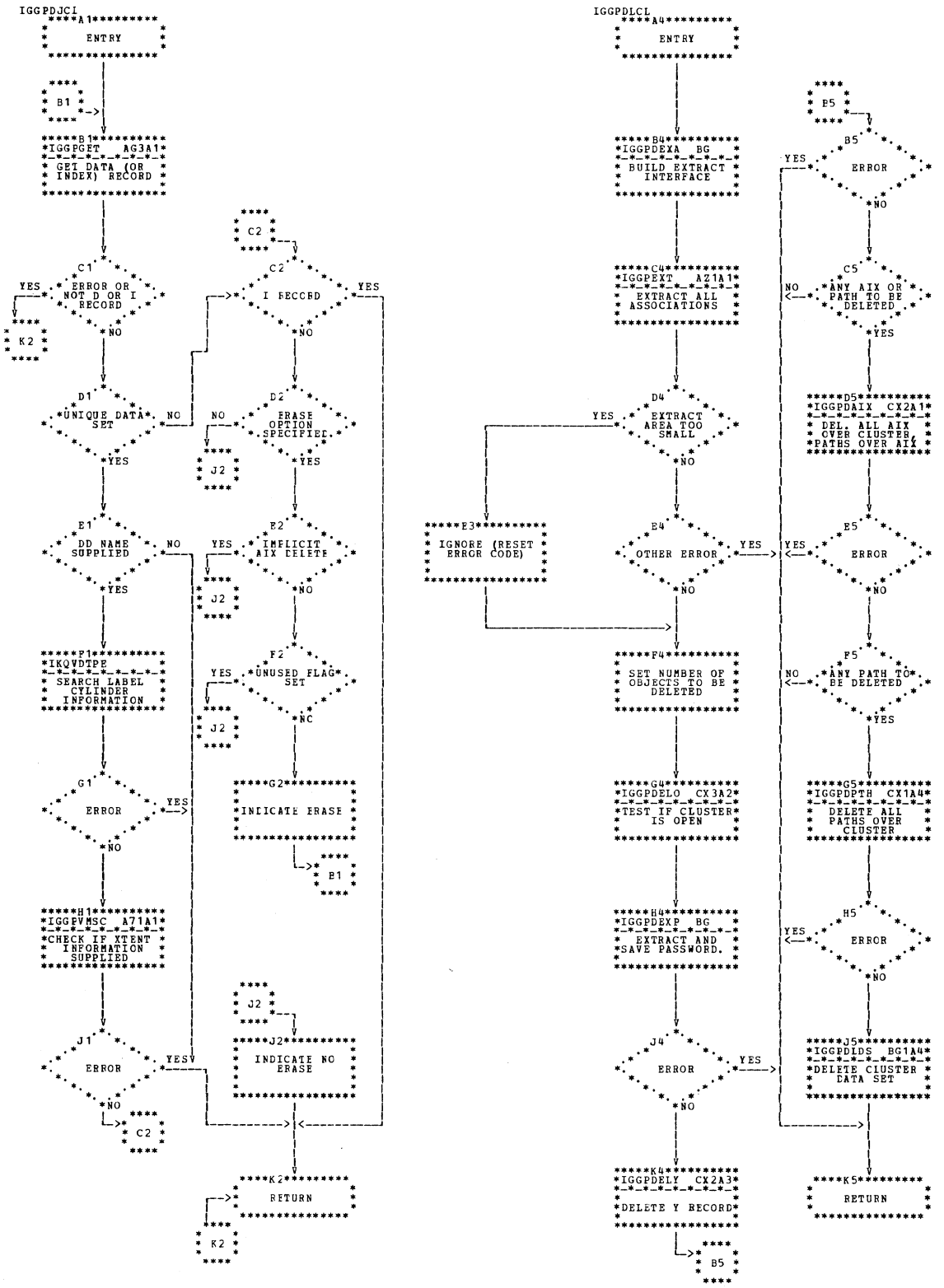


Chart BG2. CMS delete - first module (IGG0CLBG)

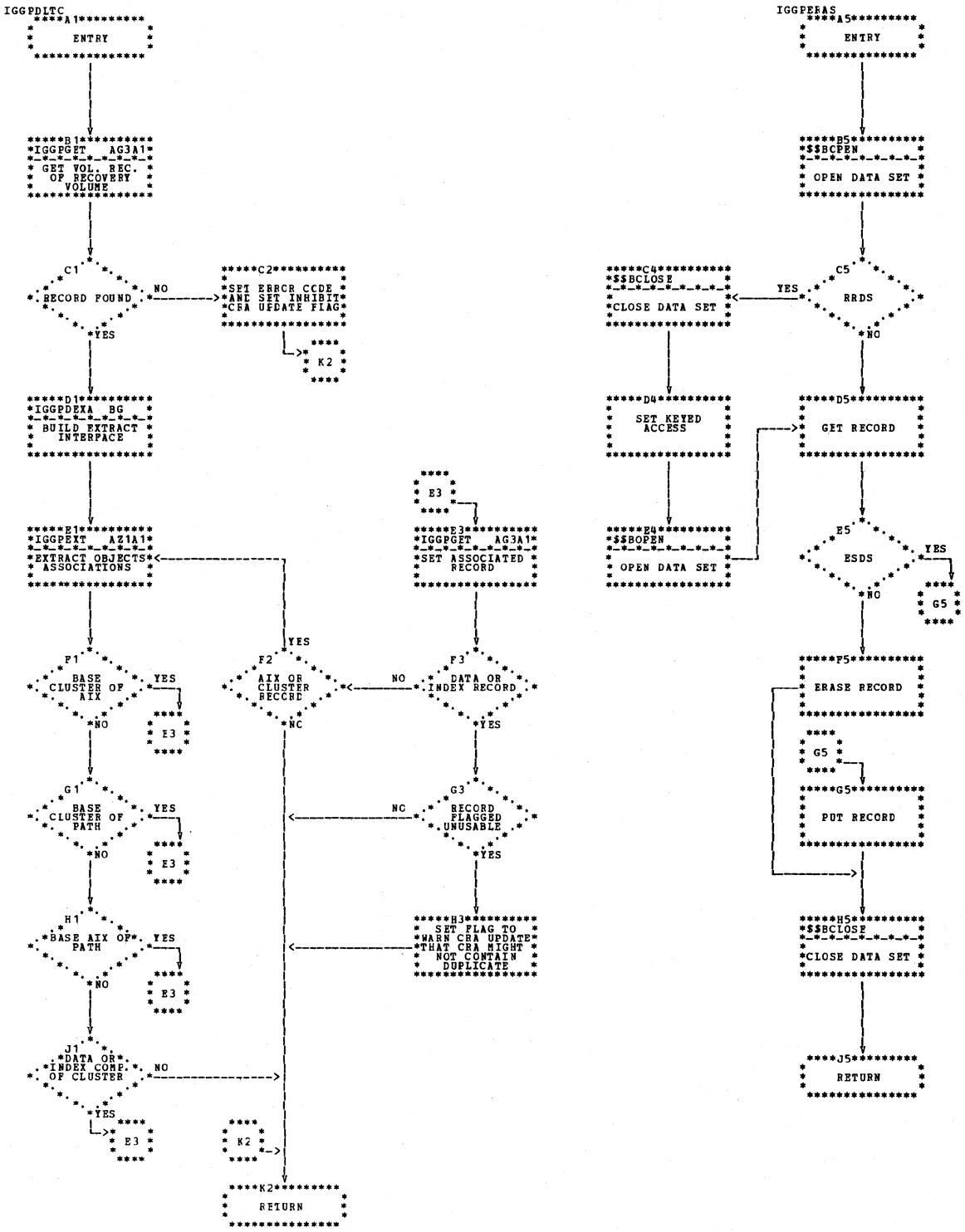


Chart BG3. CMS delete - first module (IGG0CLBG)

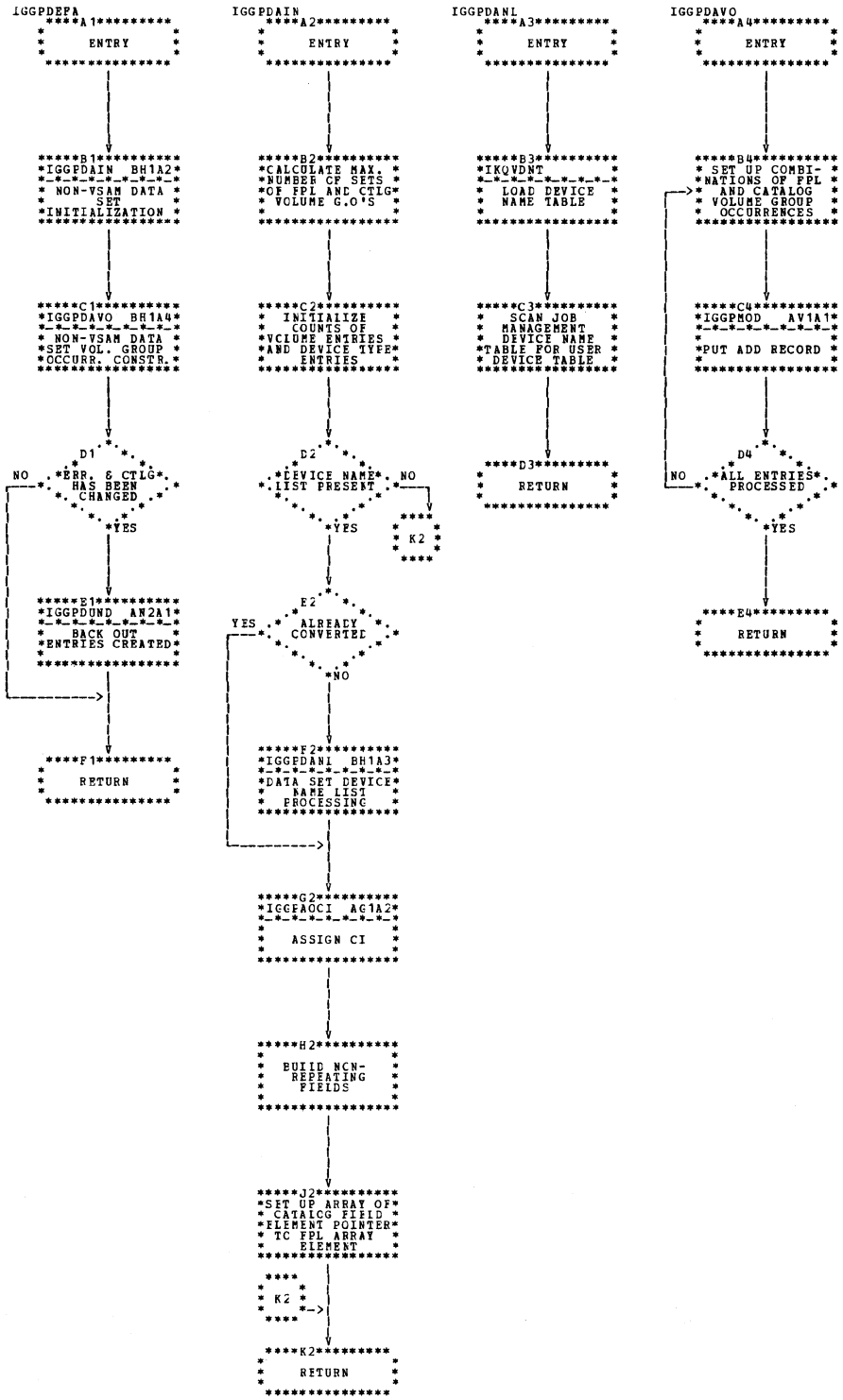


Chart BH1. Define non-VSAM data set (IGG0CLBH)

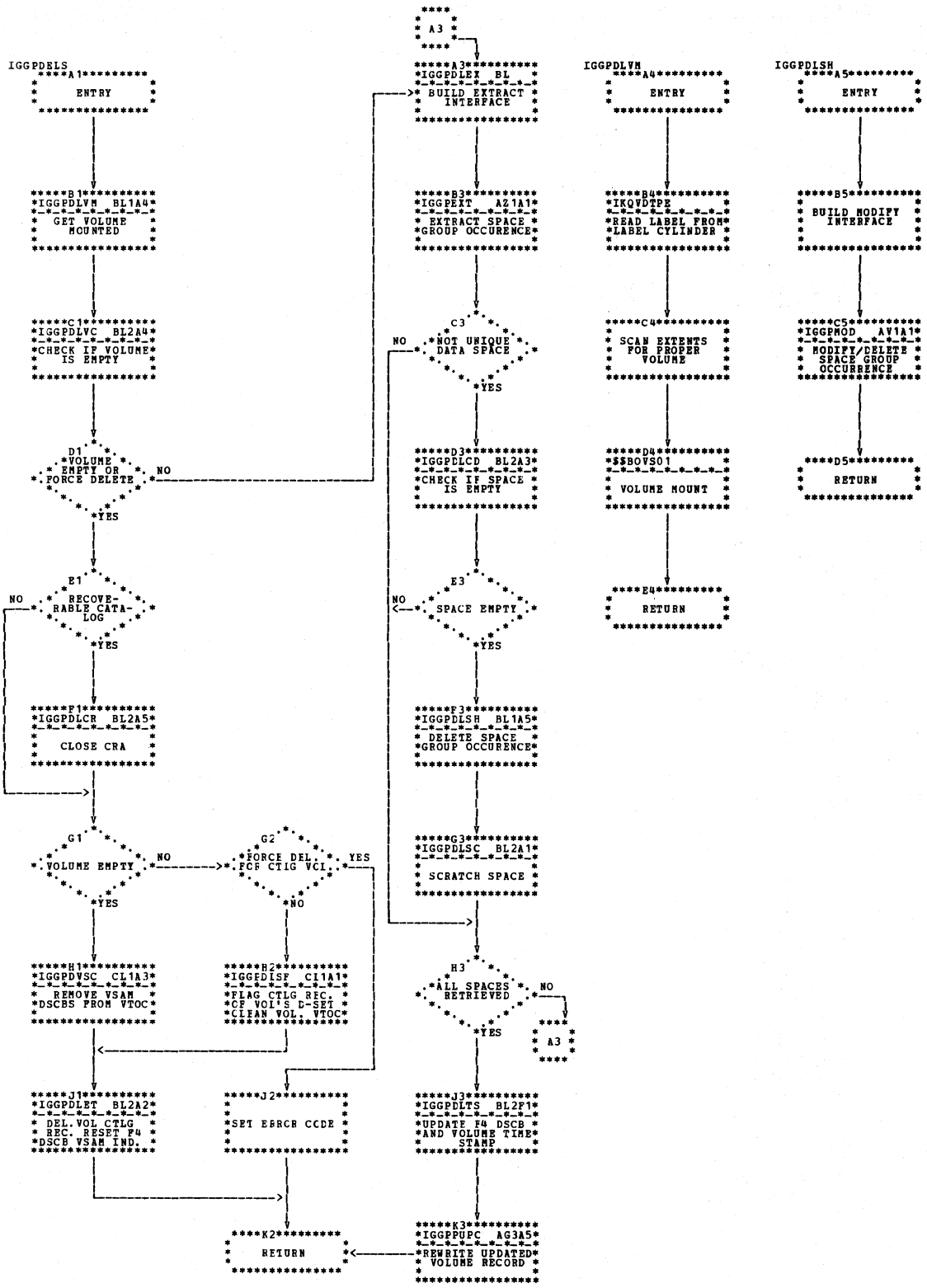


Chart BL1. CMS delete space - first module (IGG0CLBL)

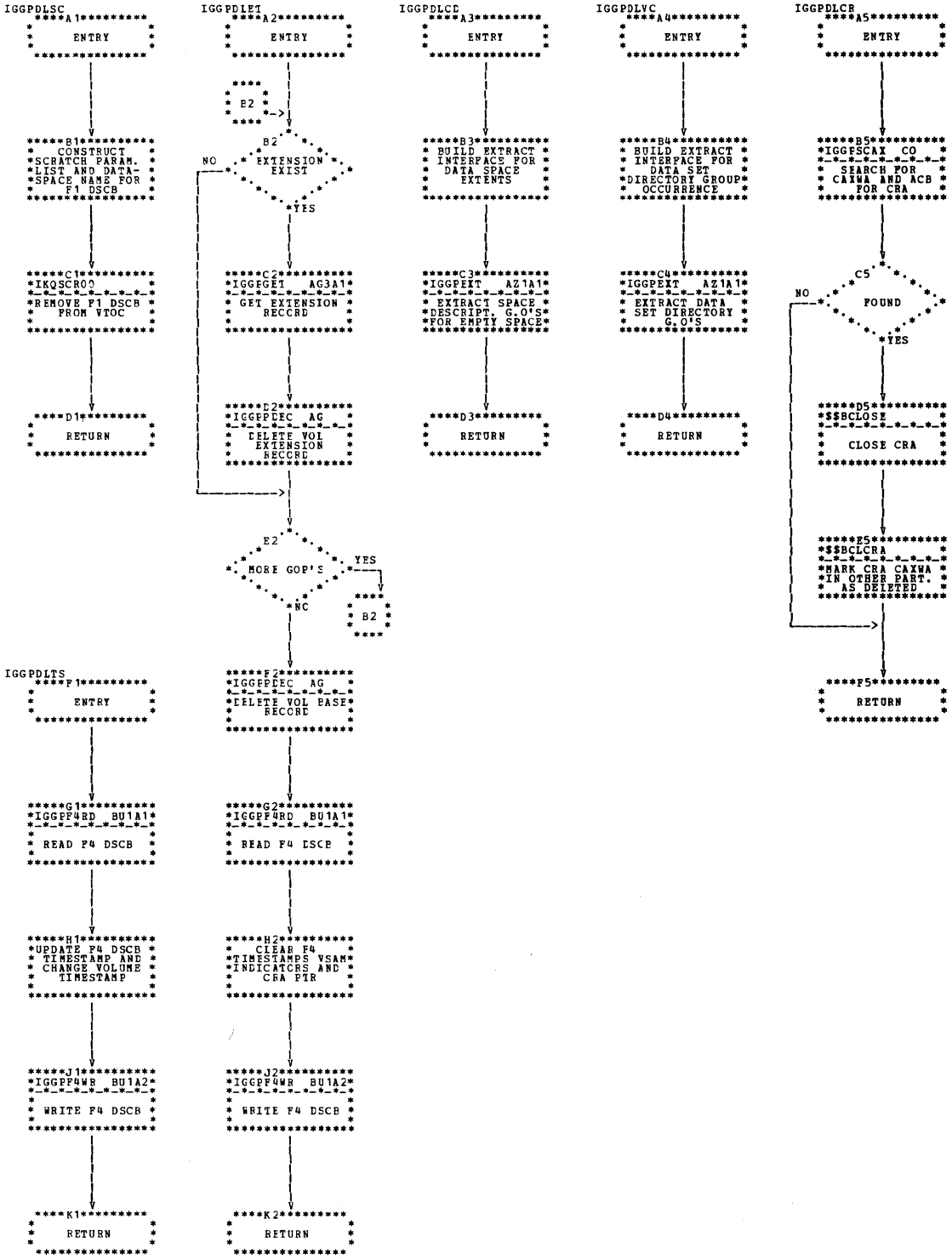


Chart BL2. CMS delete space - first module (IGG0CLBL)

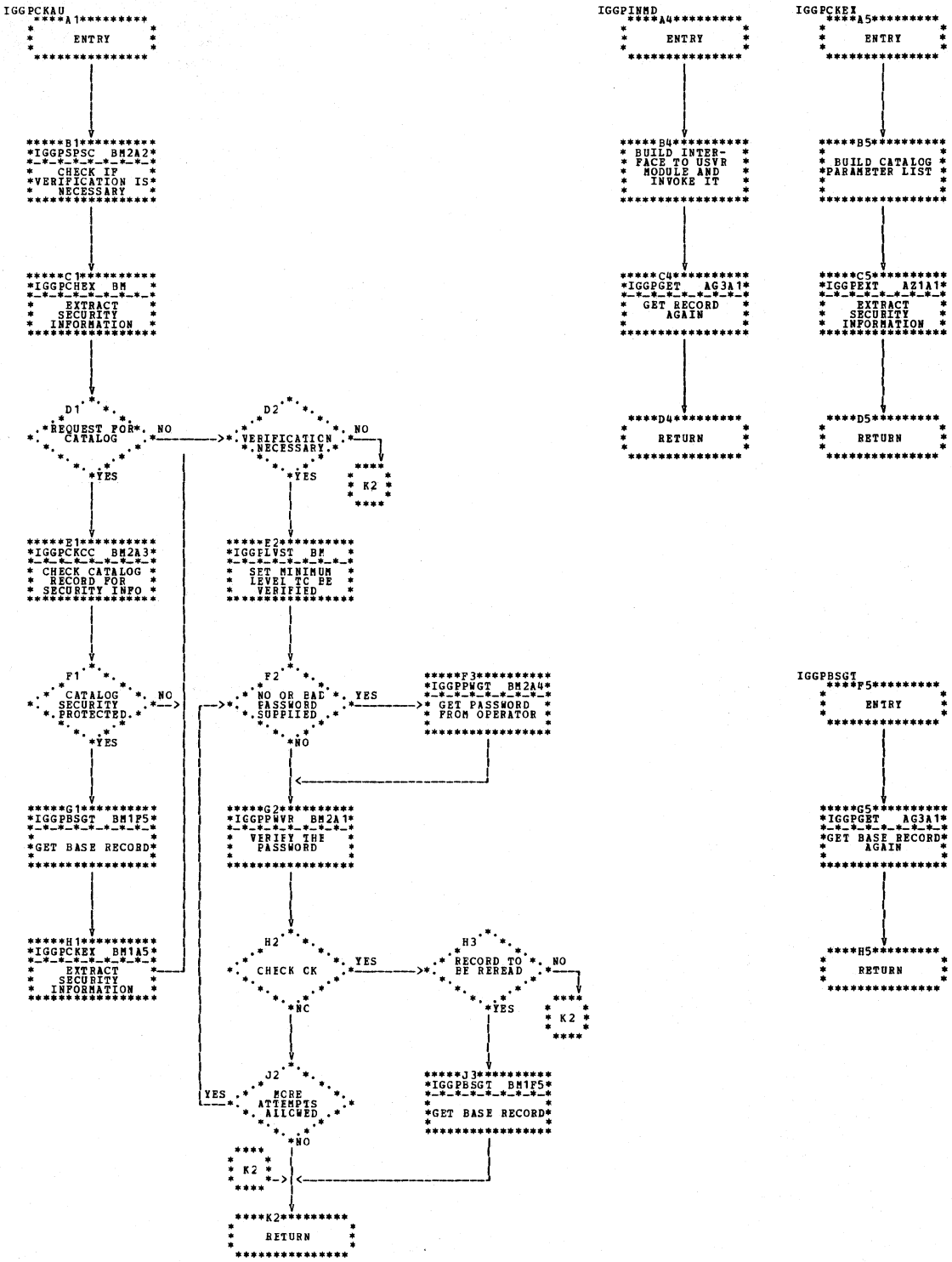


Chart BM1. Check authorization (IGG0CLBM)

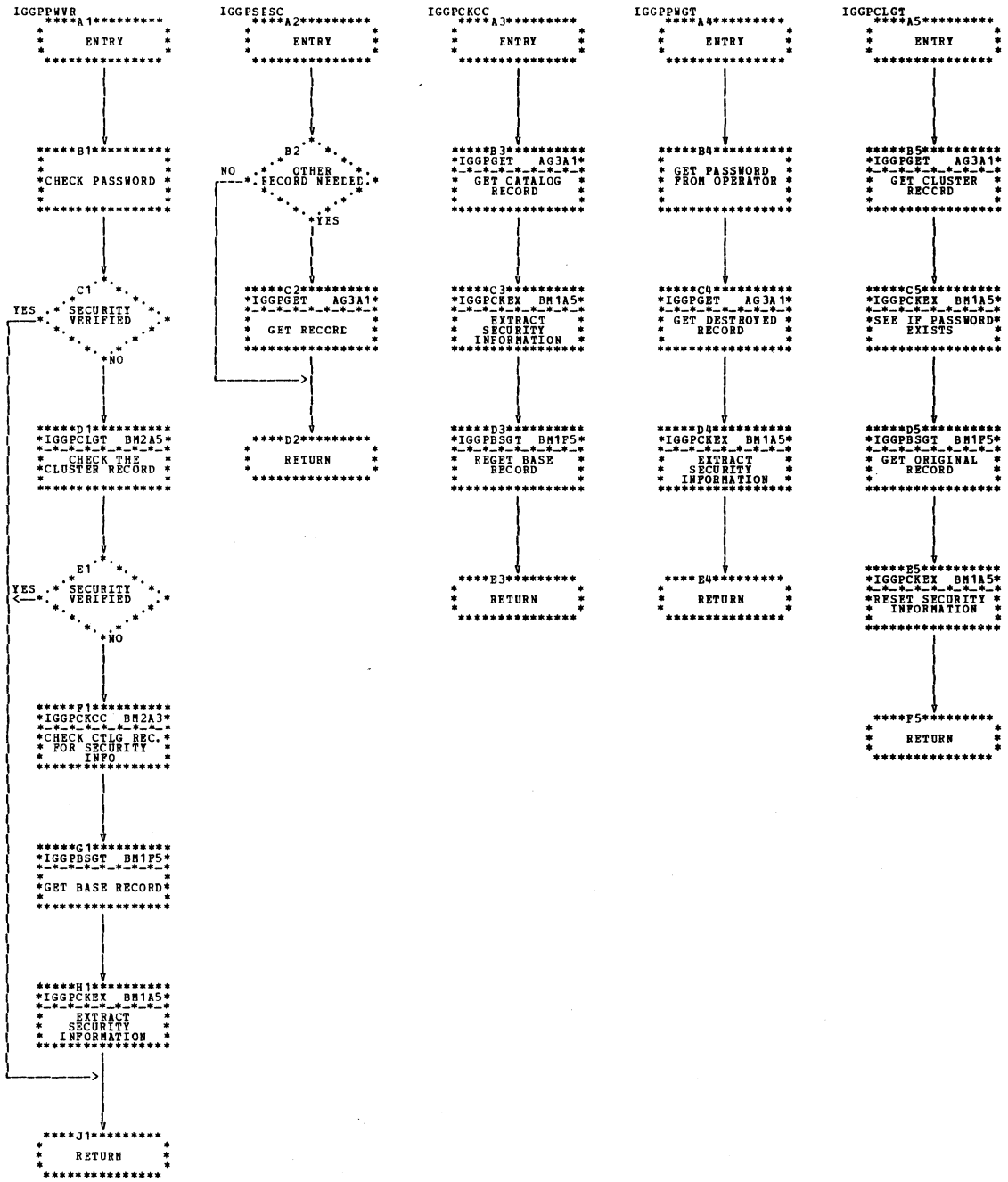


Chart BM2. Check authorization (IGG0CLBM)

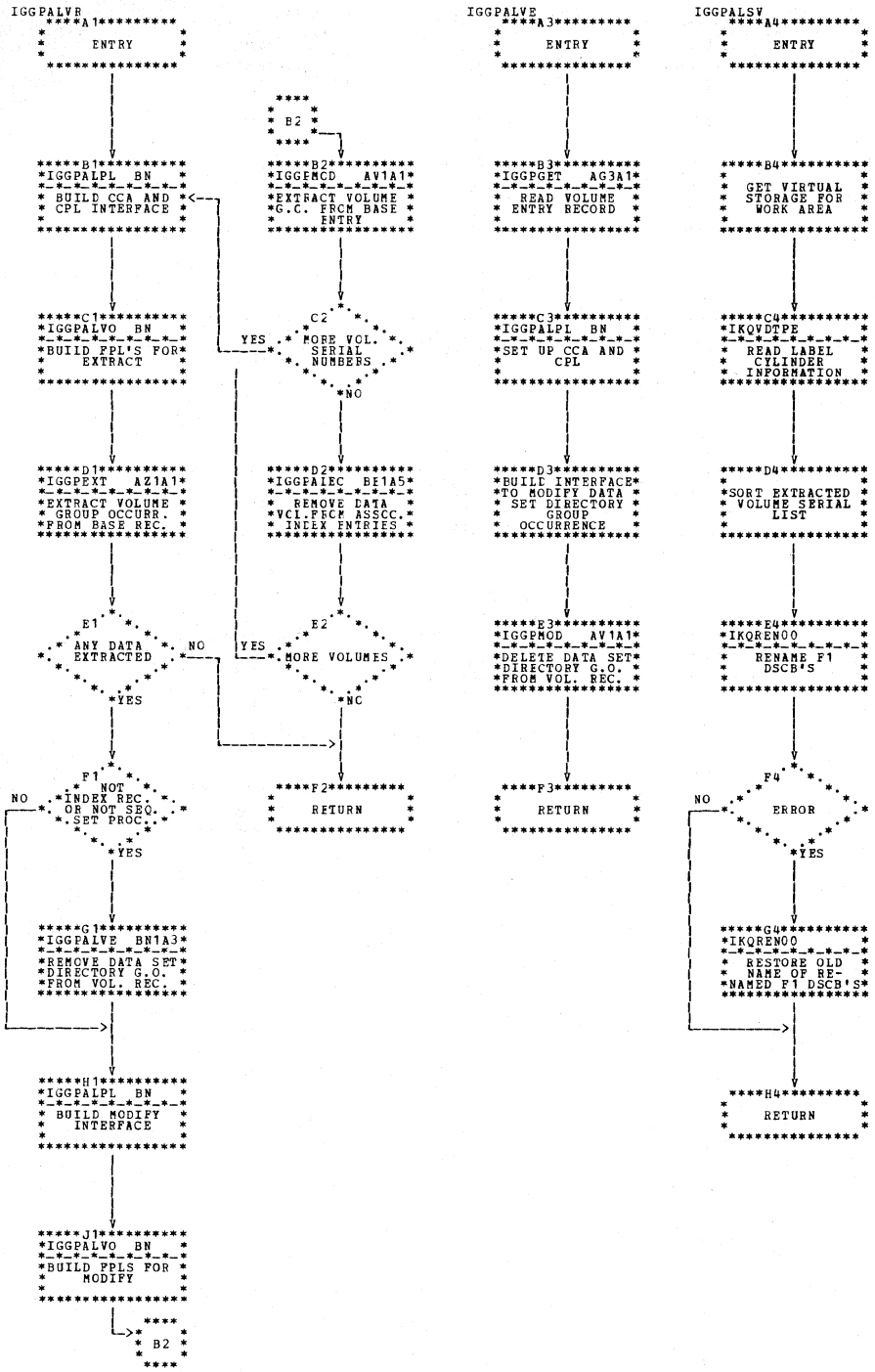


Chart BN1. CMS alter - second module (IGG0CLBN)

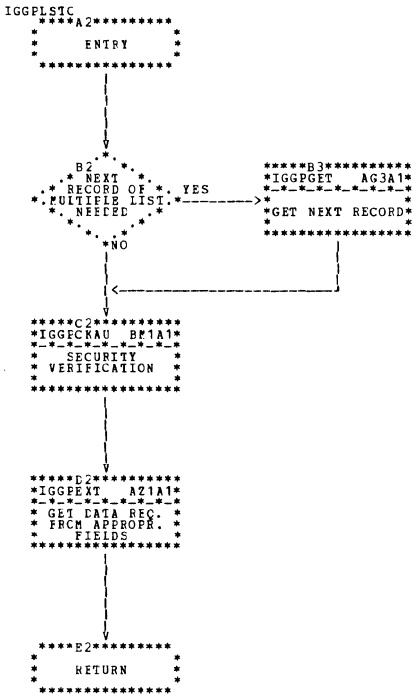


Chart BQ1. List catalog (IGG0CLBQ)

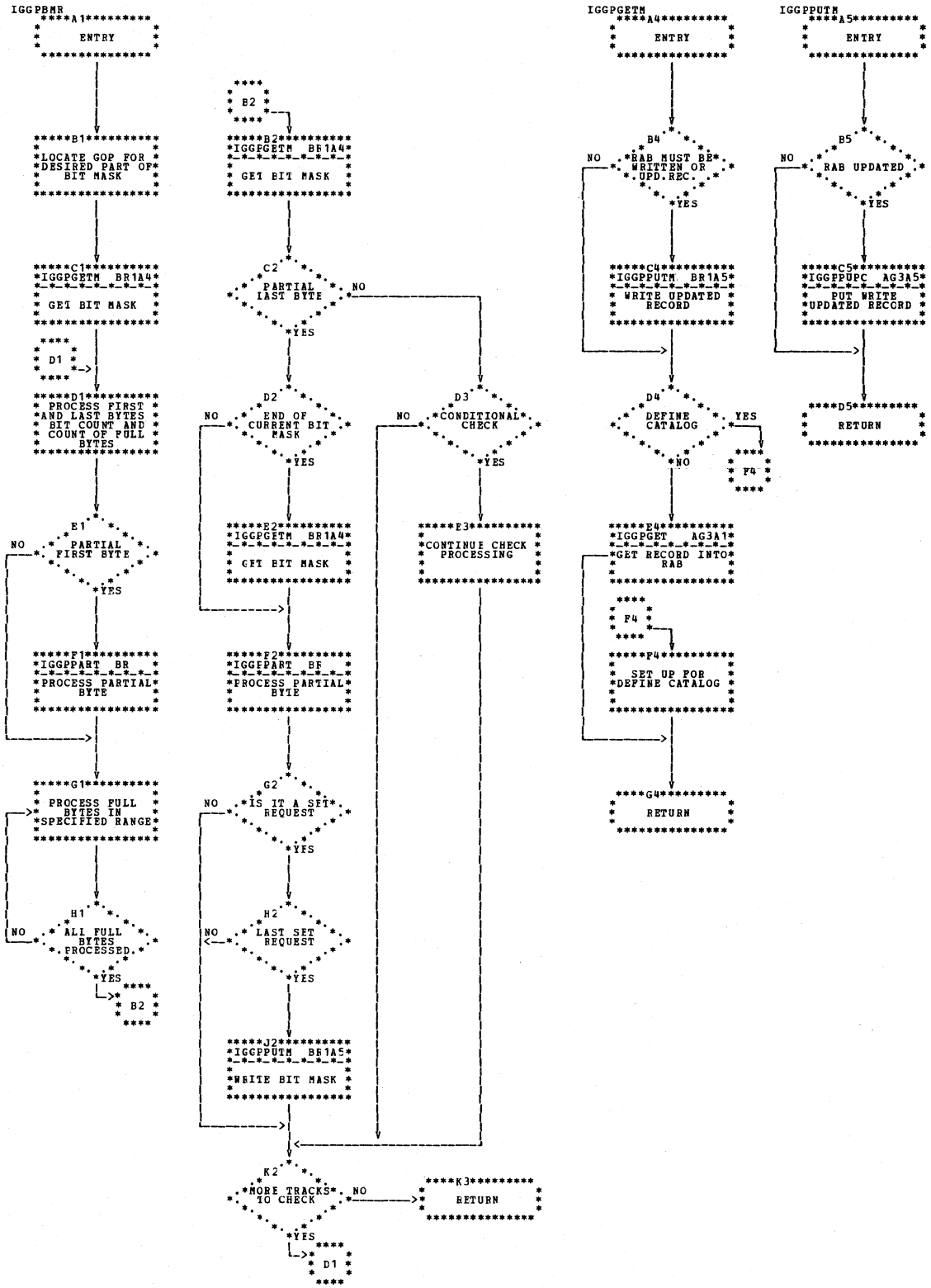


Chart BR1. Suballocate bit map handler (IGG0CLBR)

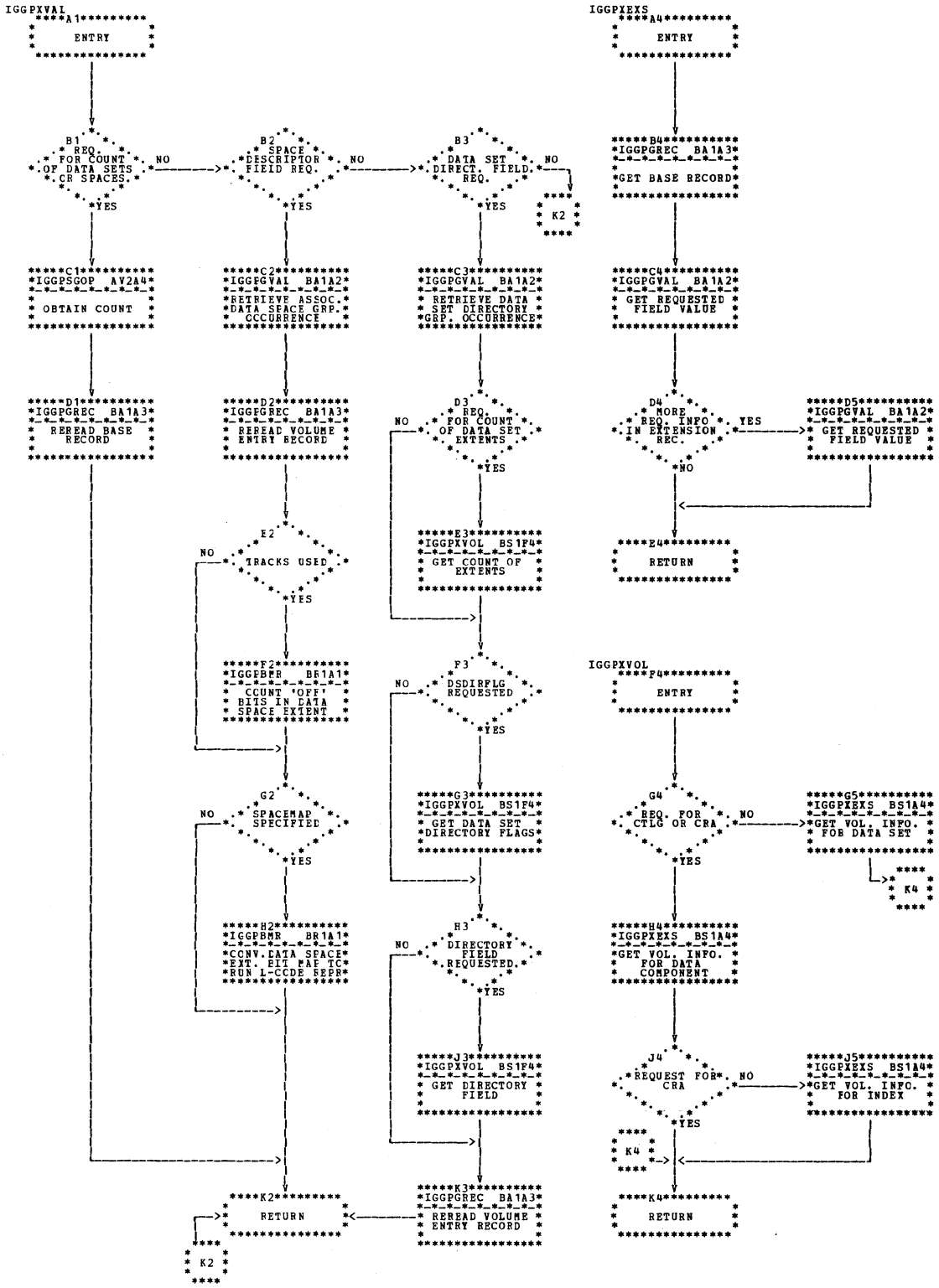


Chart BS1. Volume entry translation (IGG0CLBS)

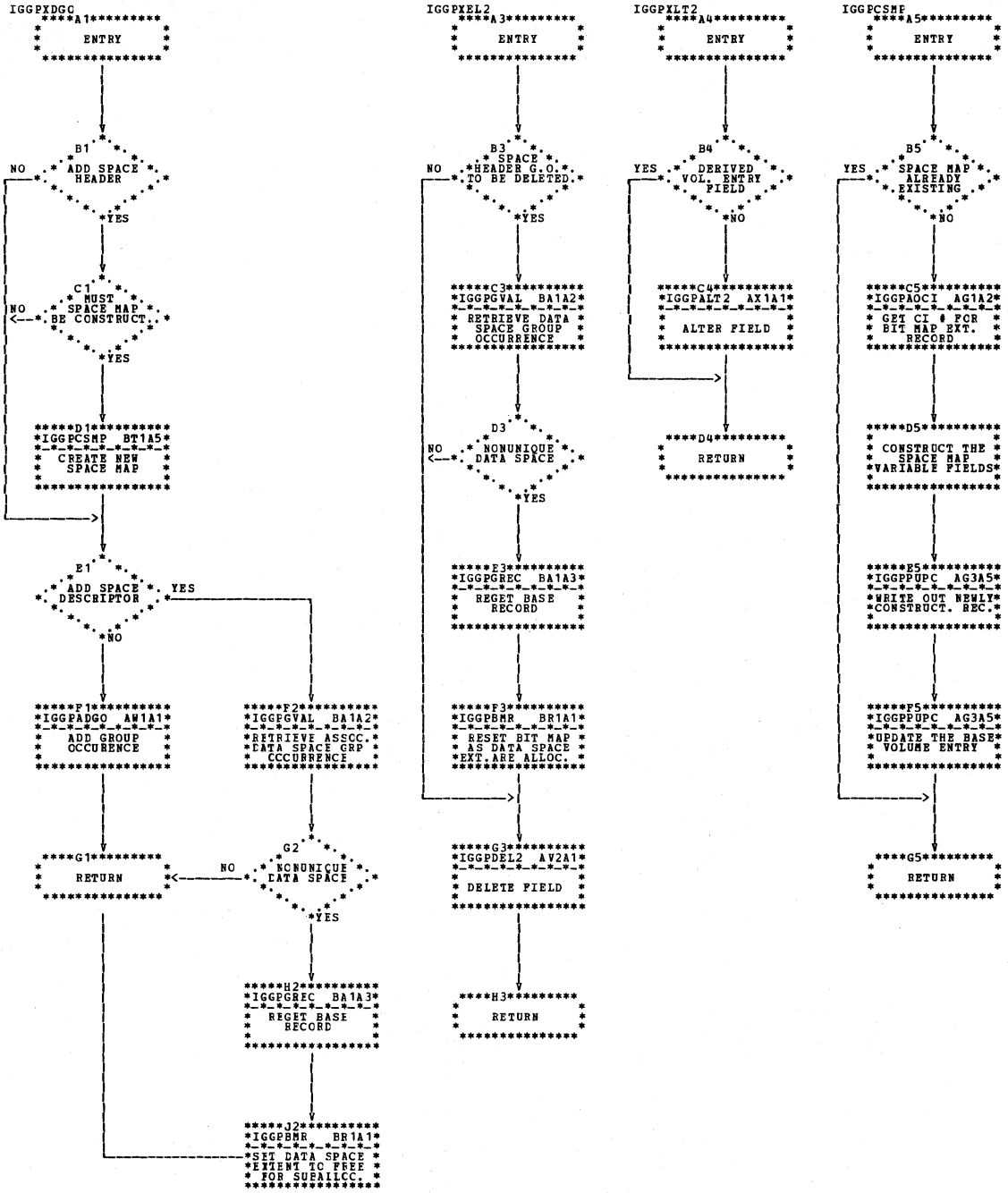


Chart BT1. Modify volume entry translation (IGG0CLBT)

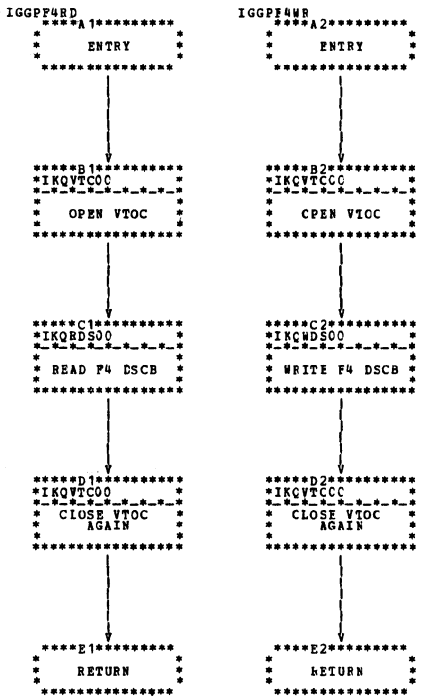


Chart BU1. Catalog read/write format 4 DSCB (IGG0CLBU)

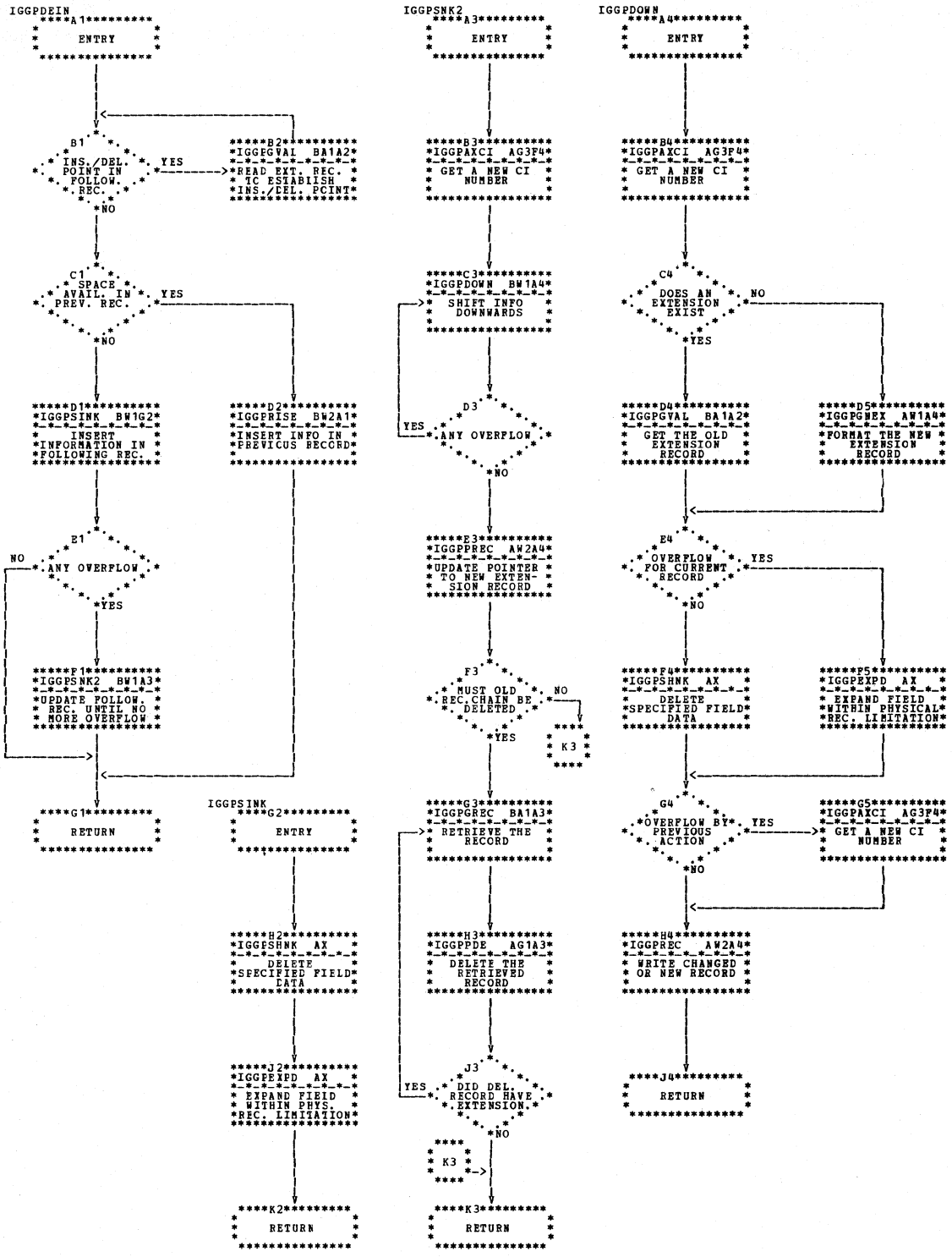


Chart BW1. Delete/insert subfunction IGG0CLBW)

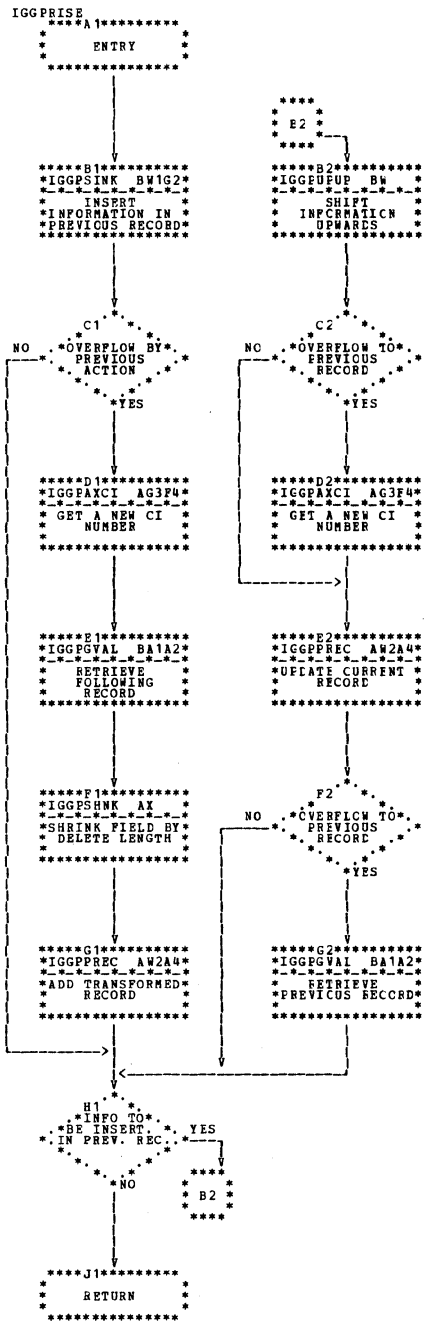


Chart BW2. Delete/insert subfunction (IGG0CLBW)

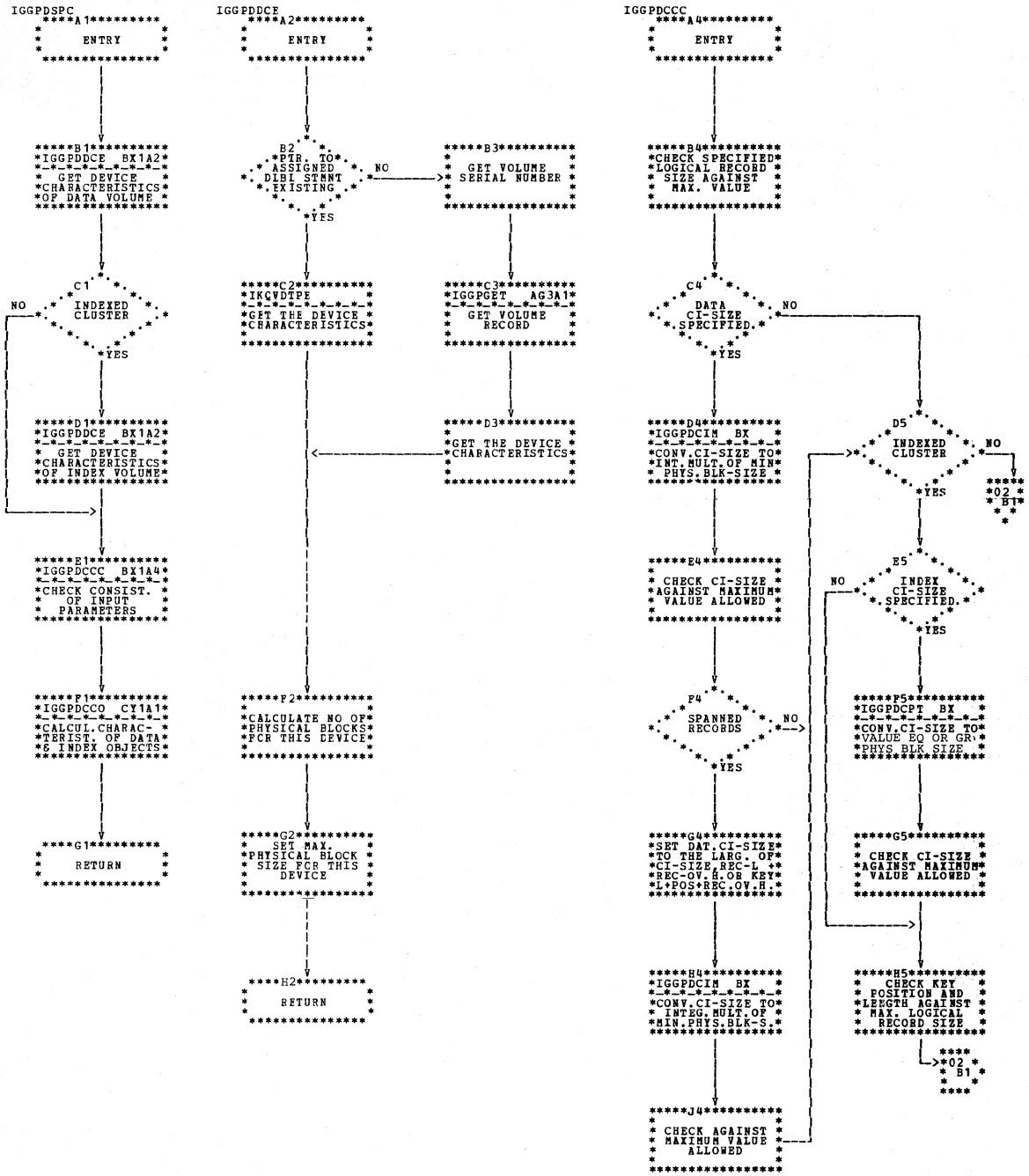


Chart BX1. CMS define - fourth module (IGG0CLBX)

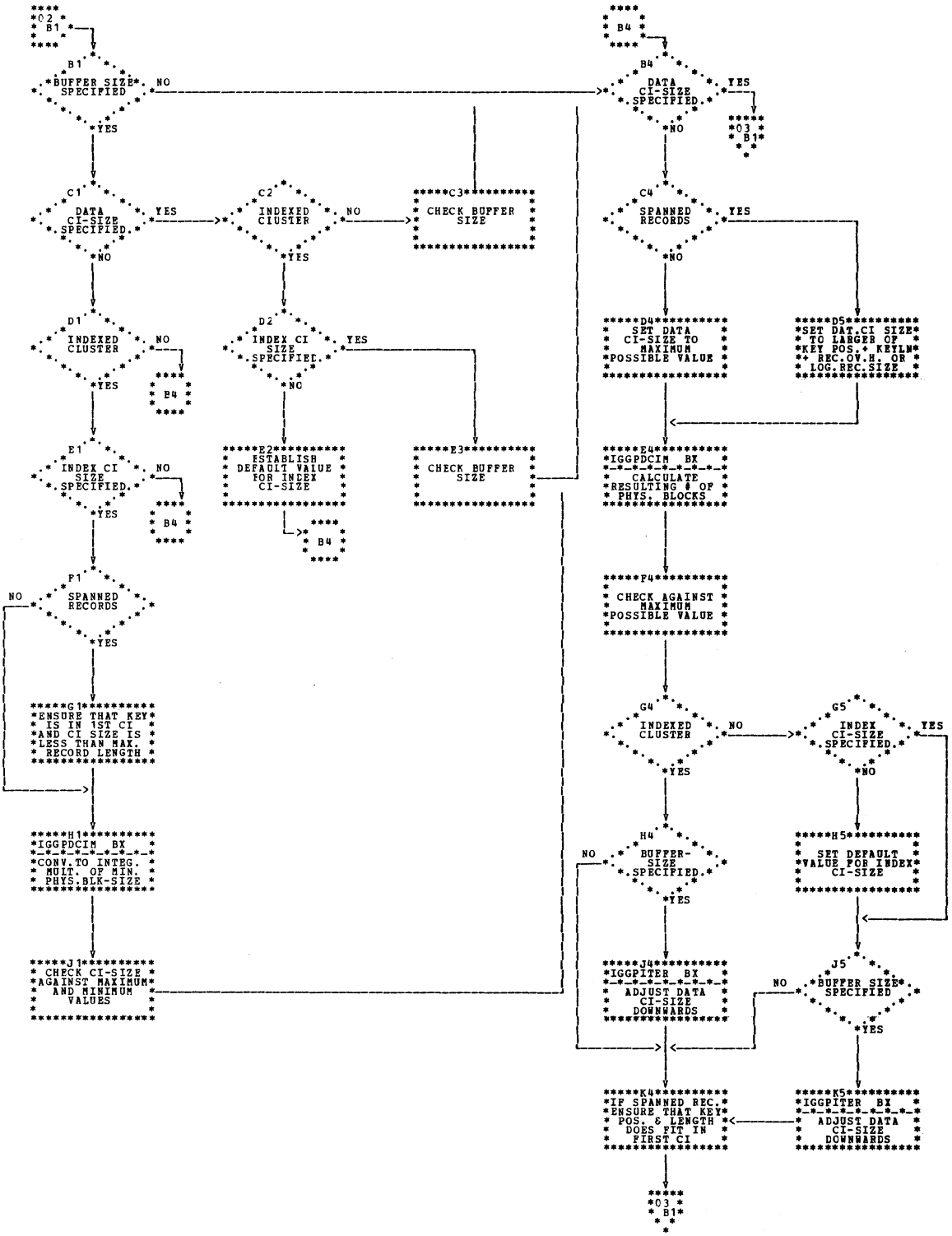


Chart BX2. CMS define - fourth module (IGG0CLBX)

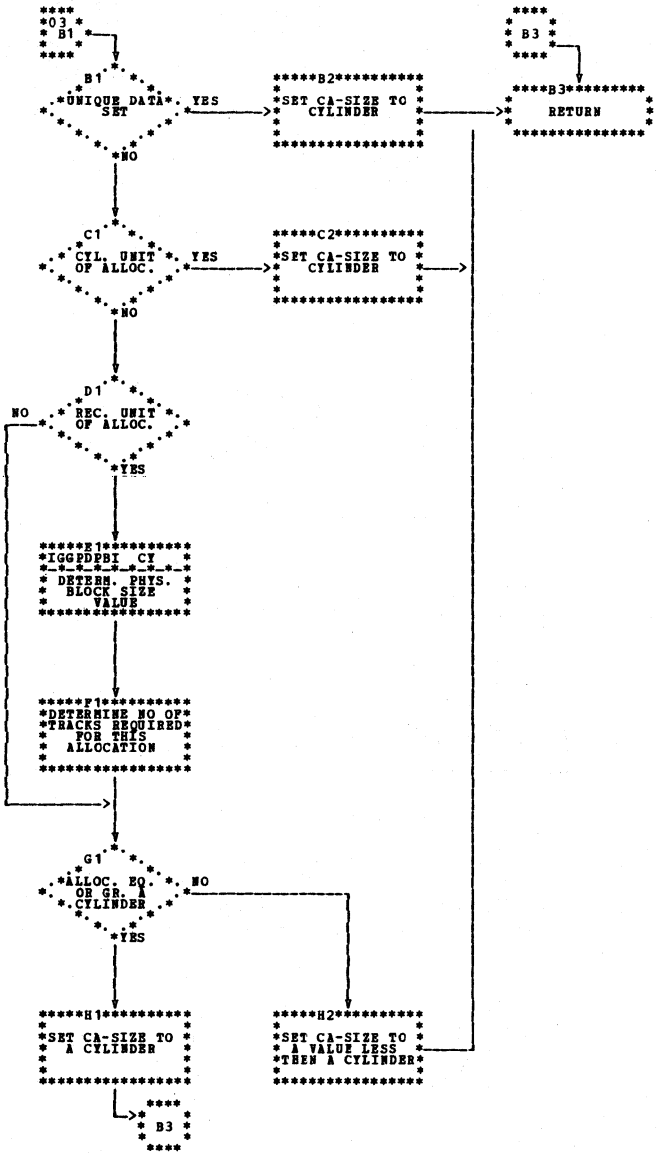


Chart BX3. CMS define - fourth module (IGG0CLBX)

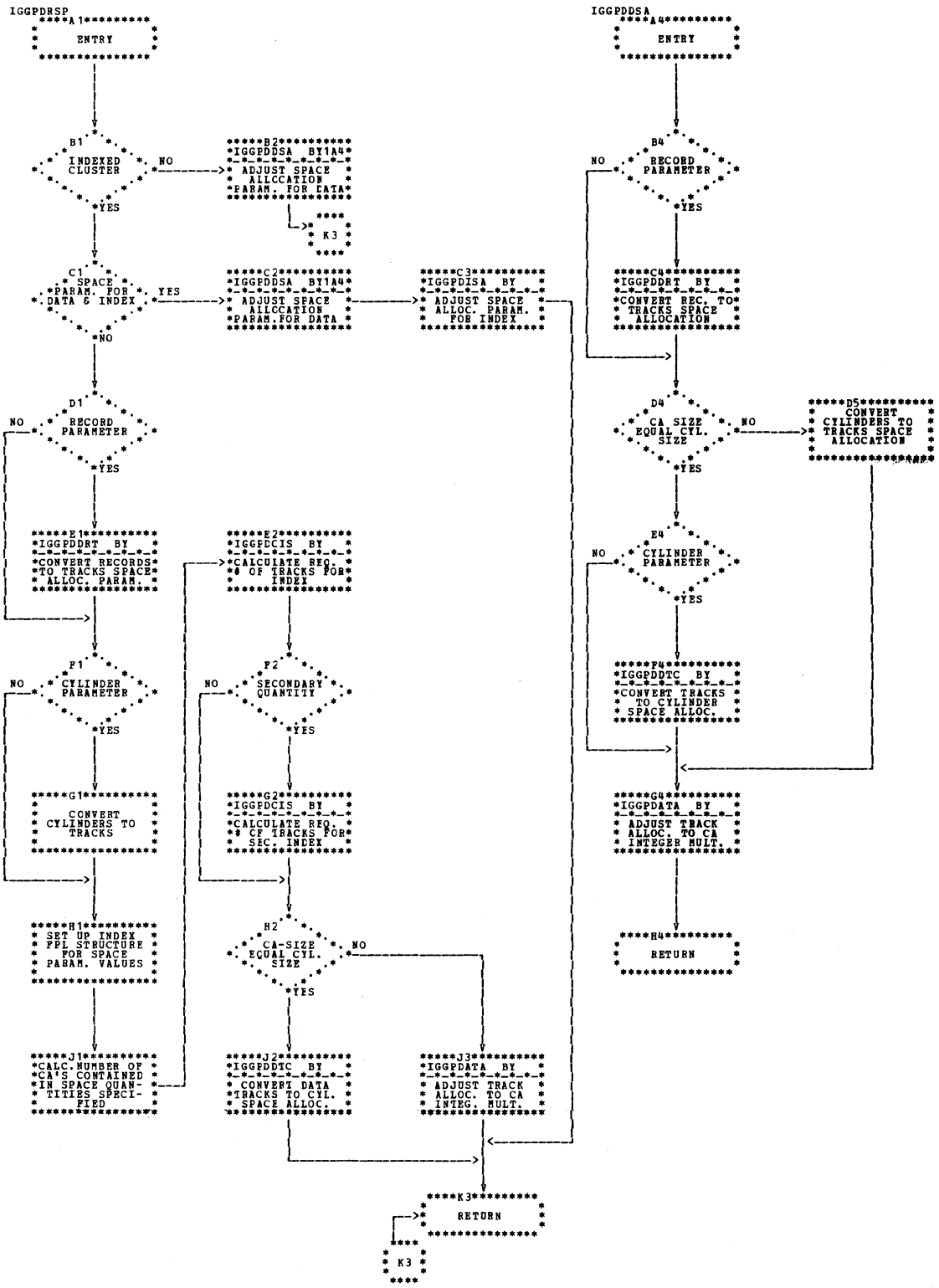


Chart BY1. CMS define - fifth module (IGG0CLBY)

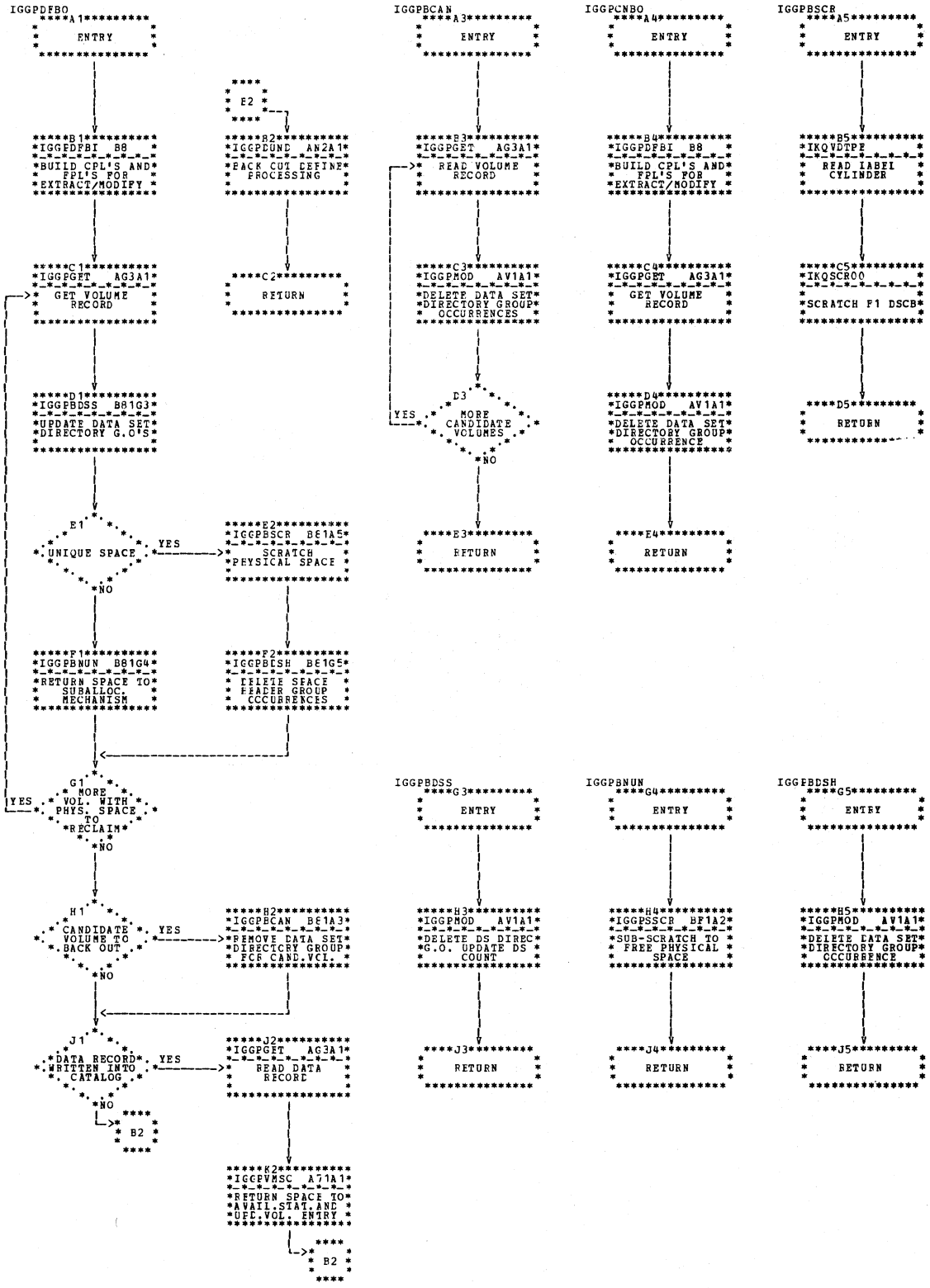
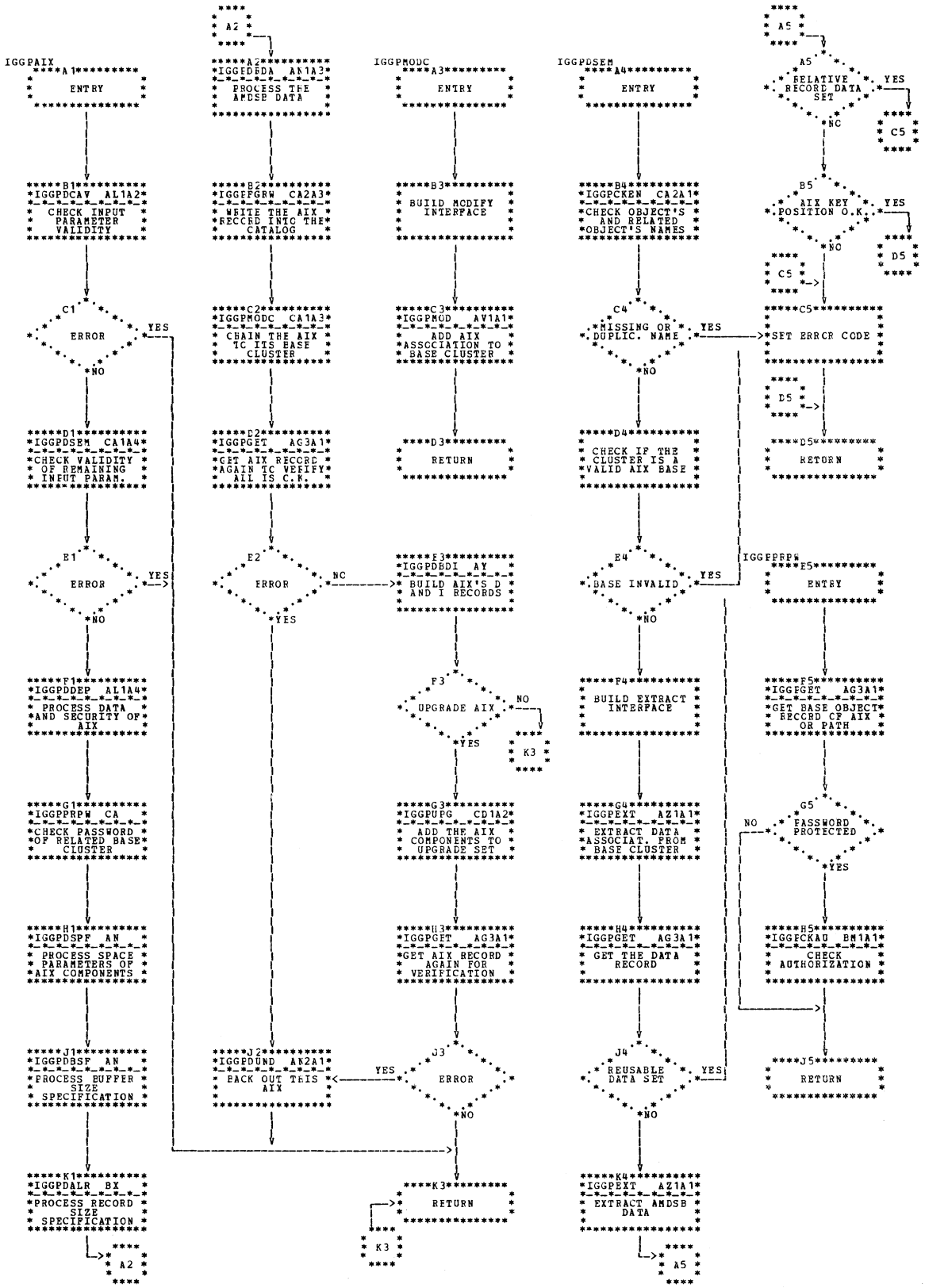


Chart B81. Define space recovery (IGG0CLB8)


IGGPAIX
A1
ENTRY
B1
IGGPDCAV AL1A2
CHECK INPUT PARAMETER VALIDITY
C1
ERROR
YES
NO
D1
IGGPDSEM CA1A4
CHECK VALIDITY OF REMAINING INPUT PARAM.
E1
ERROR
YES
NO
F1
IGGPDDEP AL1A4
PROCESS DATA AND SECURITY OF AIX
G1
IGGPPRPW CA
CHECK PASSWORD OF RELATED BASE CLUSTER
H1
IGGPDSPF AN
PROCESS SPACE PARAMETERS OF AIX COMPONENTS
J1
IGGPDBSP AN
PROCESS BUFFER SIZE SPECIFICATION
K1
IGGPDALR BX
PROCESS RECORD SIZE SPECIFICATION
A2
A2
IGGEDEDA AN1A3
PROCESS THE AMDSB DATA
B2
IGGPFGRW CA2A3
WRITE THE AIX RECORD INTO THE CATALOG
C2
IGGPMODC CA1A3
CHAIN THE AIX TO ITS BASE CLUSTER
D2
IGGPGET AG3A1
GET AIX RECORD AGAIN TO VERIFY ALL IS O.K.
E2
ERROR
NO
YES
J2
IGGPDUND AN2A1
BACK OUT THIS AIX
IGGPMODC
A3
ENTRY
B3
BUILD MODIFY INTERFACE
C3
IGGPMOD AV1A1
ADD AIX ASSOCIATION TO BASE CLUSTER
D3
RETURN
F3
IGGPDBDI AY
BUILD AIX'S D AND I RECORDS
F3
UPGRADE AIX
NO
YES
K3
G3
IGGPUPG CD1A2
ADD THE AIX COMPONENTS TO UPGRADE SET
H3
IGGPGET AG3A1
GET AIX RECORD AGAIN FOR VERIFICATION
J3
ERROR
YES
NO
K3
RETURN
K3
IGGPDSEM
A4
ENTRY
B4
IGGPCKEN CA2A1
CHECK OBJECT'S AND RELATED OBJECT'S NAMES
C4
MISSING OR DUPLIC. NAME
YES
NO
D4
CHECK IF THE CLUSTER IS A VALID AIX BASE
E4
BASE INVALID
YES
NO
F4
BUILD EXTRACT INTERFACE
G4
IGGPEXT AZ1A1
EXTRACT DATA ASSOCIAT. FROM BASE CLUSTER
H4
IGGPGET AG3A1
GET THE DATA RECORD
J4
REUSABLE DATA SET
YES
NO
K4
IGGPEXT AZ1A1
EXTRACT AMDSB DATA
A5
A5
RELATIVE RECORD DATA SET
YES
NO
C5
B5
AIX KEY POSITION O.K.
YES
NO
D5
C5
C5
SET ERROR CODE
D5
D5
RETURN
IGGPPRPW
E5
ENTRY
F5
IGGPGET AG3A1
GET BASE OBJECT RECORD OF AIX OR PATH
G5
PASSWORD PROTECTED
NO
YES
H5
IGGPCKAU BM1A1
CHECK AUTHORIZATION
J5
RETURN

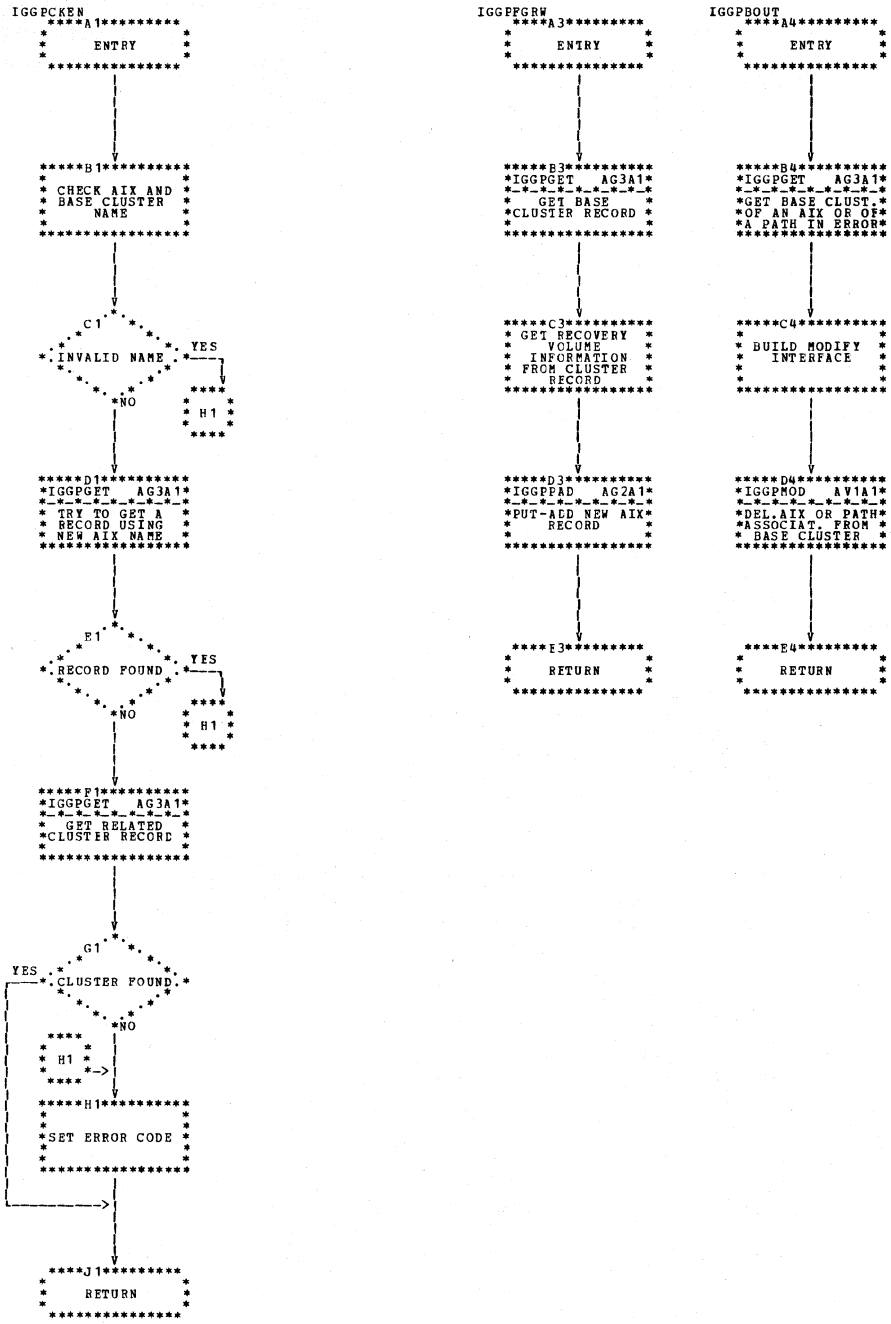


Chart CA2. Define alternate index (IGG0CLCA)

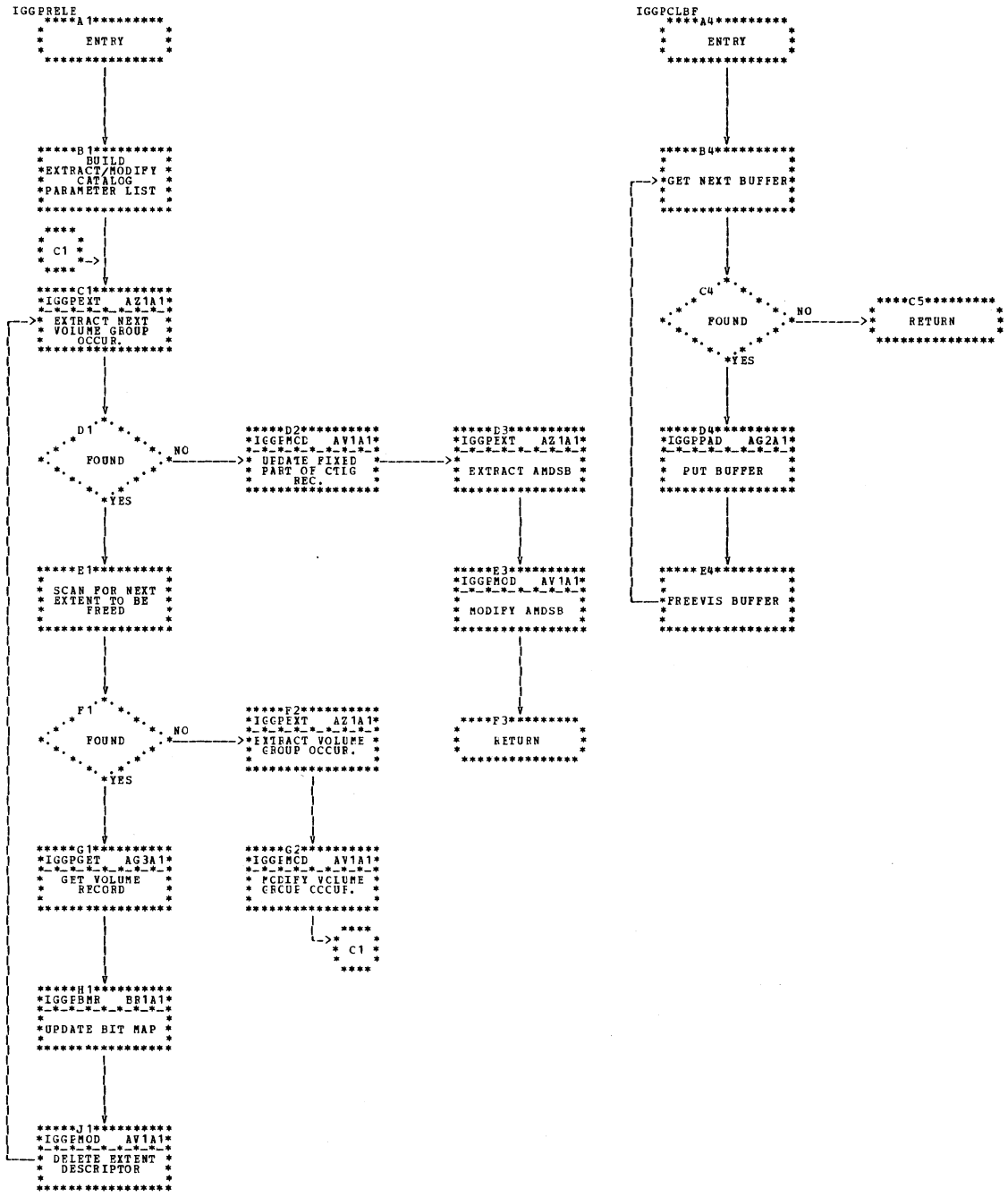


Chart CB1. Release function (IGG0CLCB)

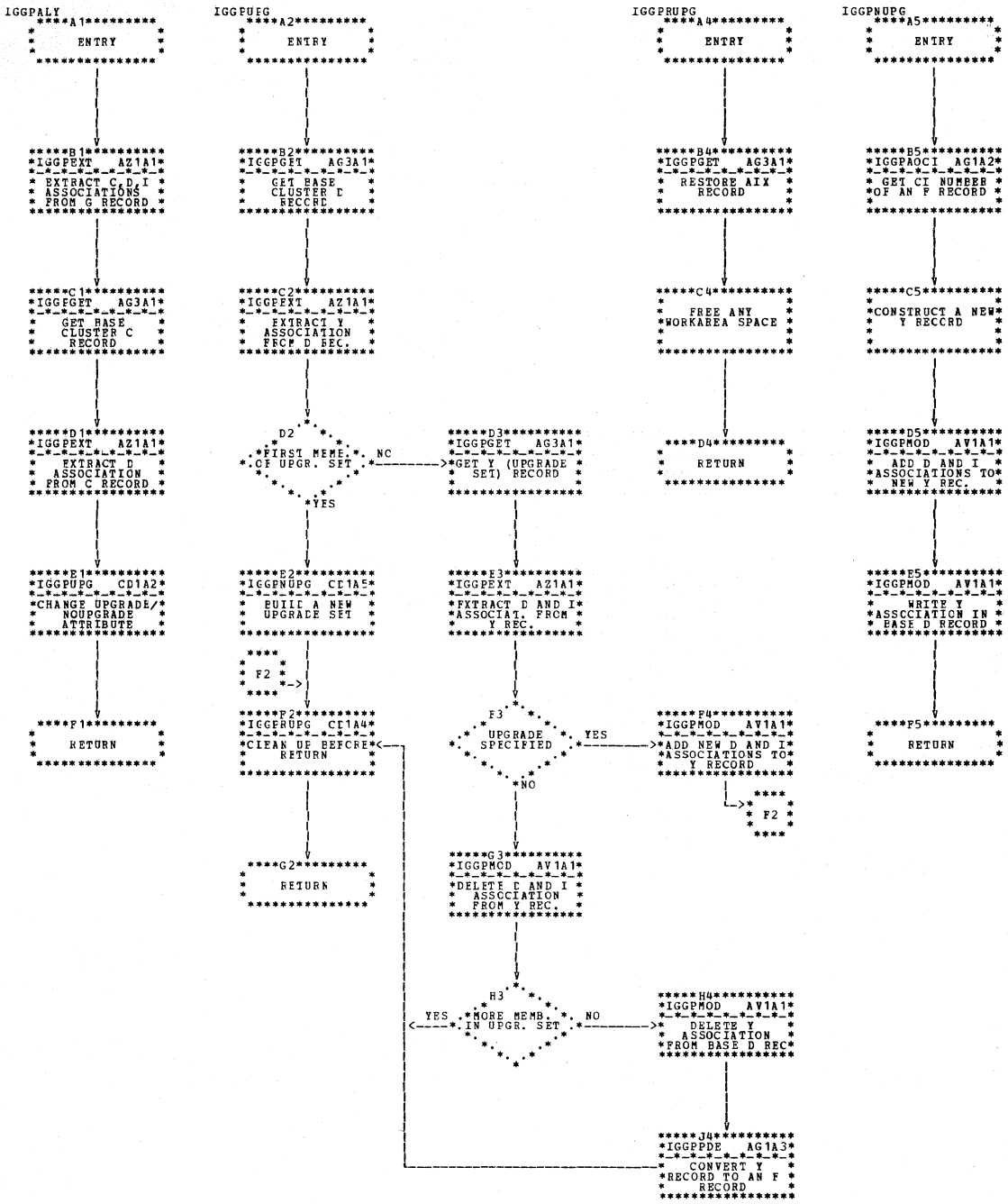


Chart CD1. CMS alter - fourth module (IGG0CLCD)

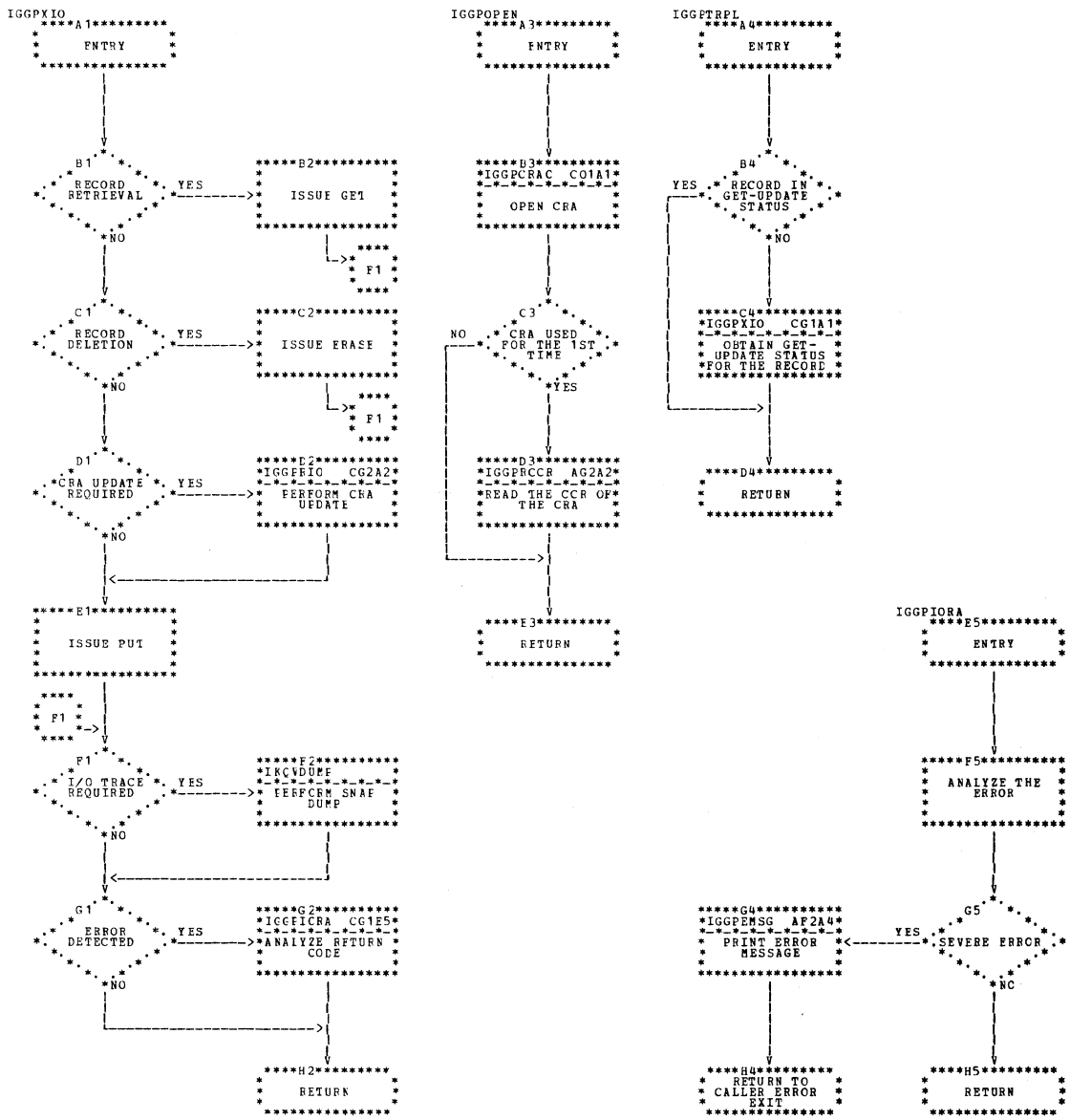


Chart CG1. Catalog I/O subroutine - second module (IGG0CLCG)

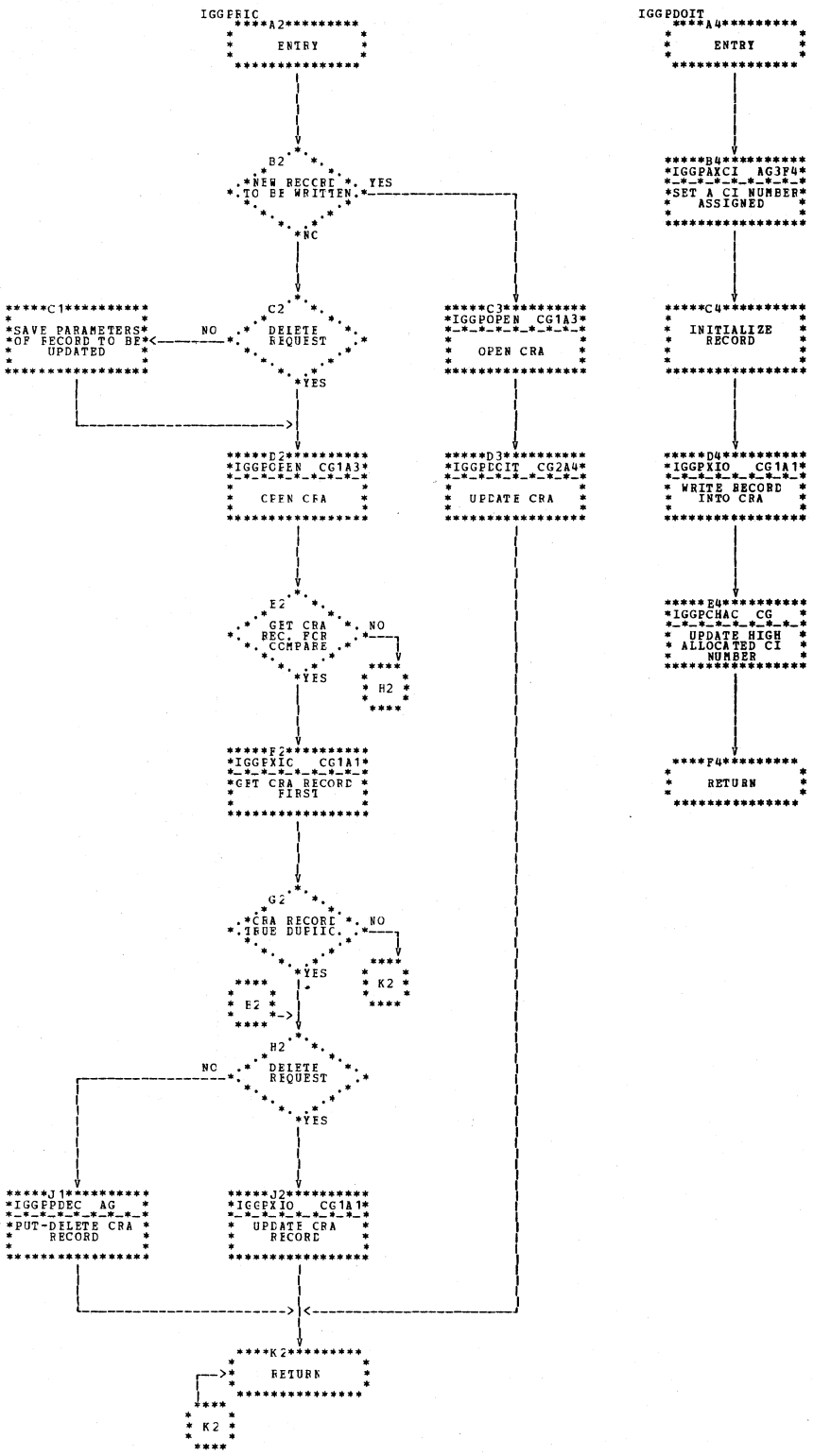


Chart CG2. Catalog I/O subroutine - second module (IGG0CLCG)

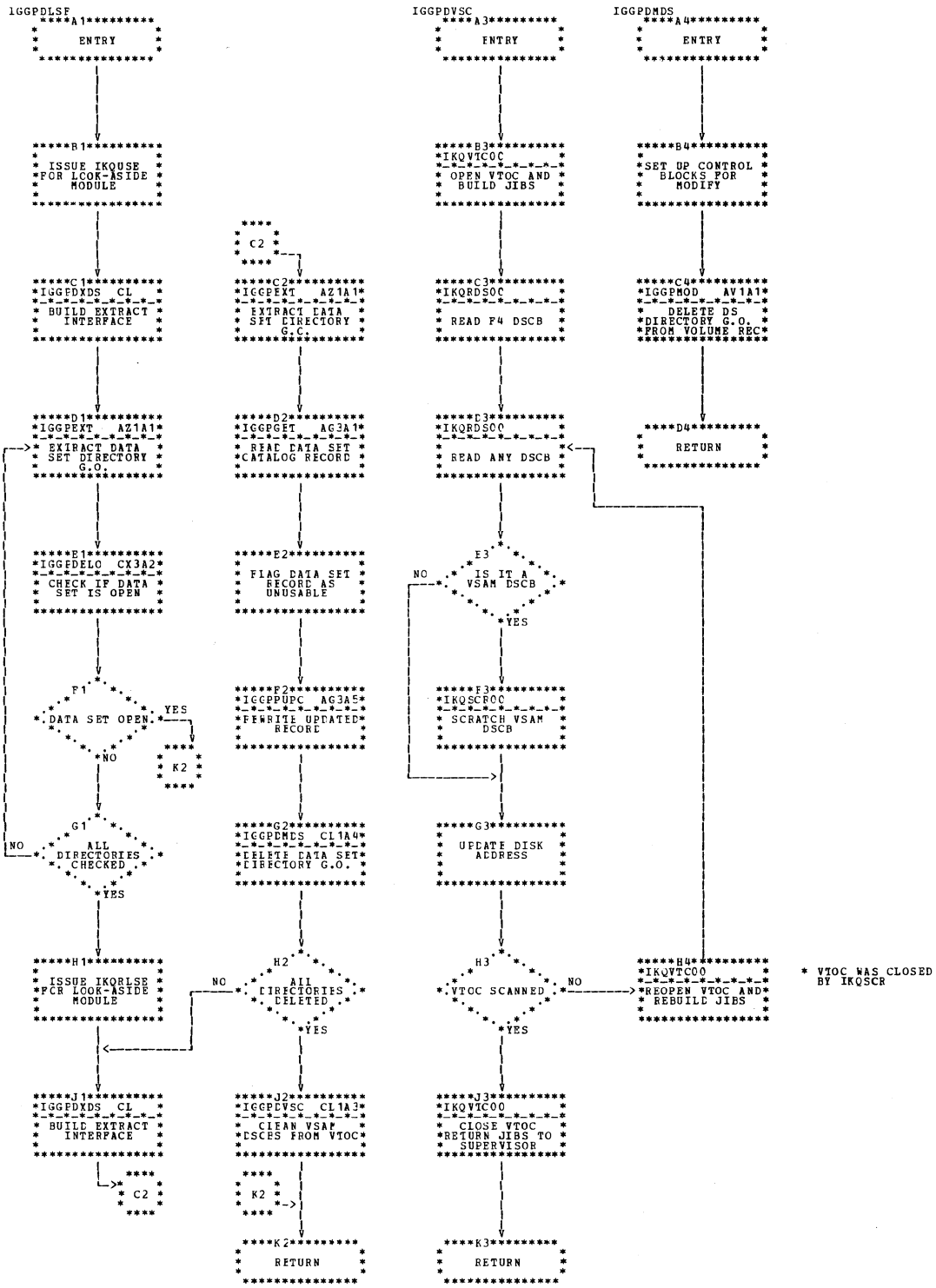


Chart CL1. CMS delete space - second module (IGG0CLCL)

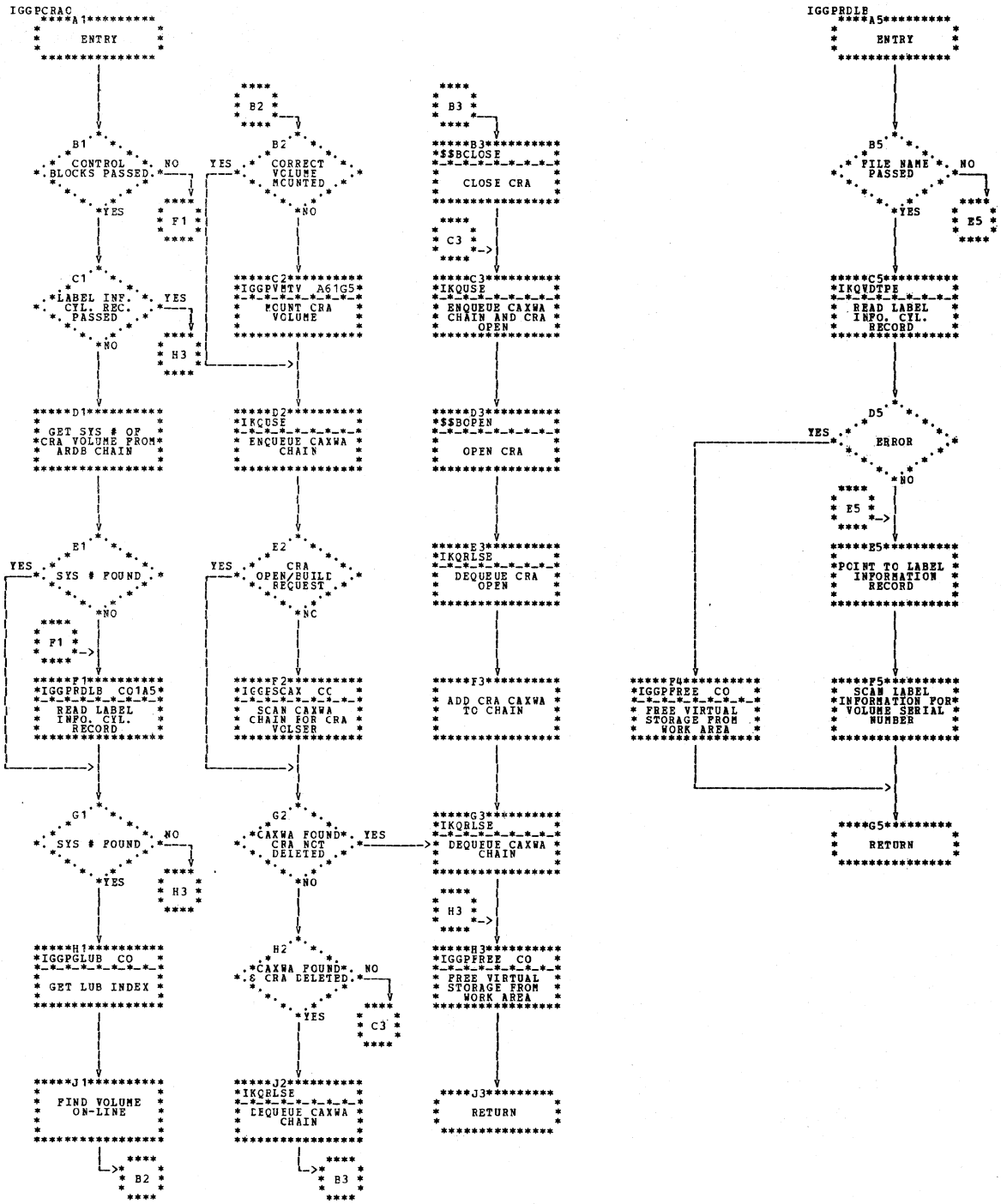


Chart CO1. Open catalog recovery area (IGG0CLCO)

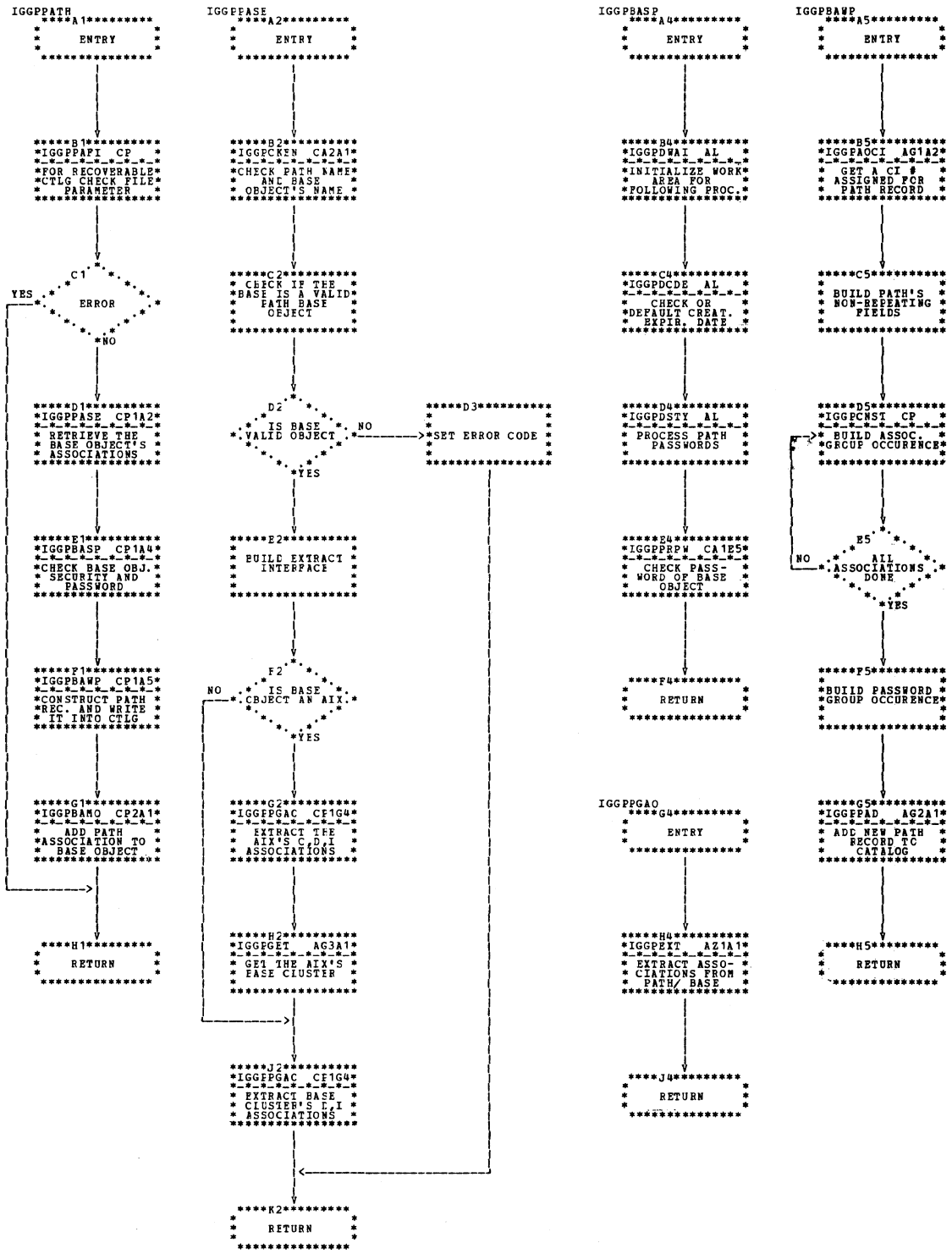


Chart CP1. Define path (IGG0CLCP)

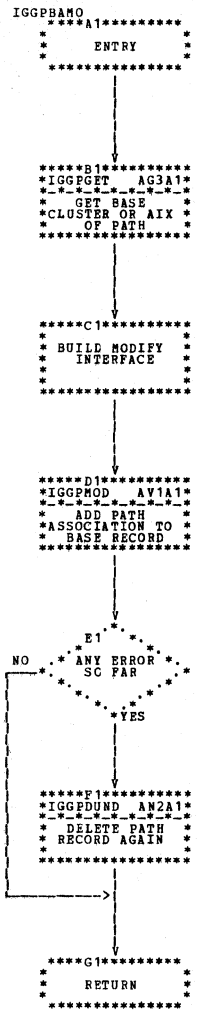


Chart CP2. Define path (IGG0CLCP)

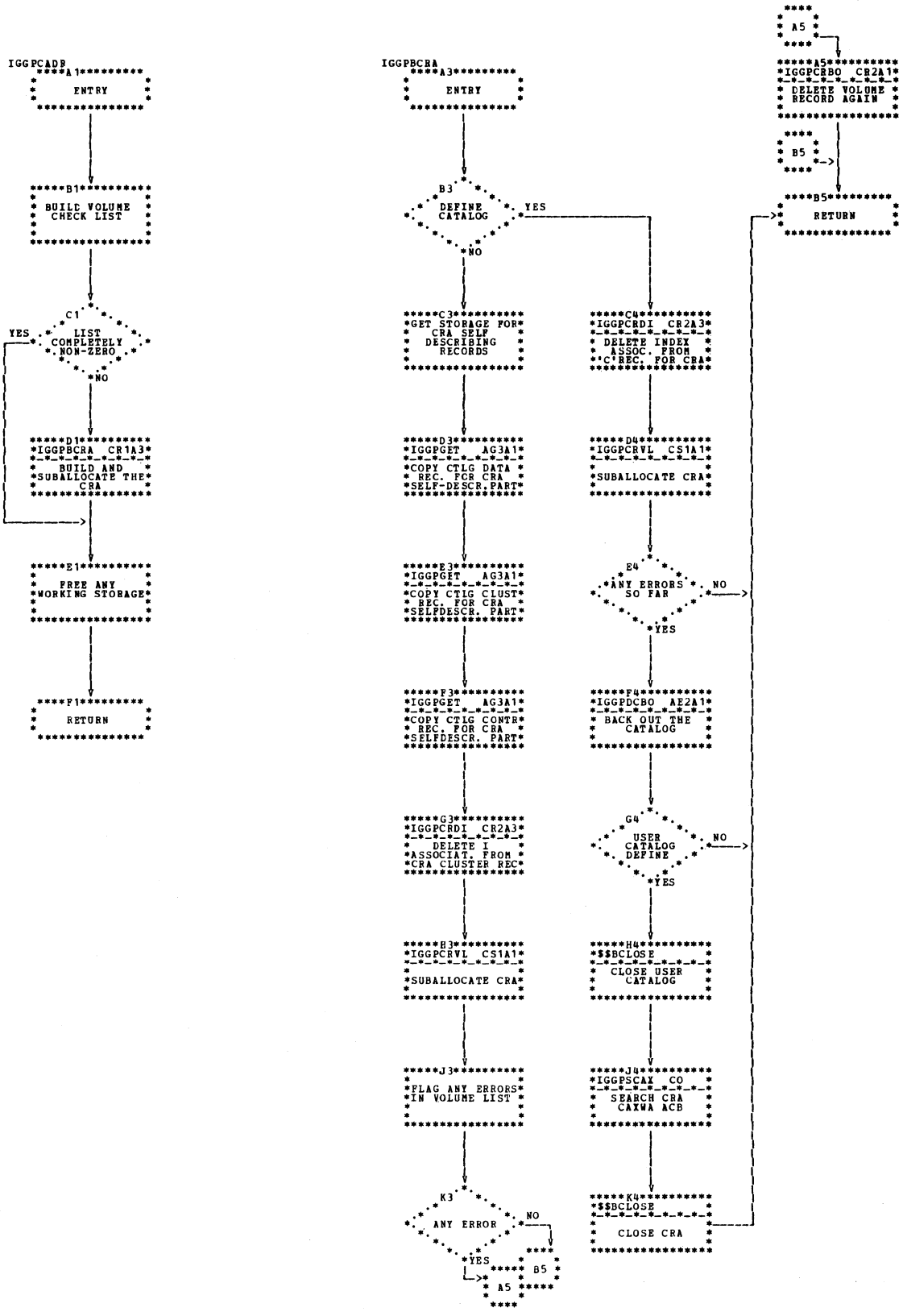


Chart CR1. Define catalog recovery area (IGG0CLCR)

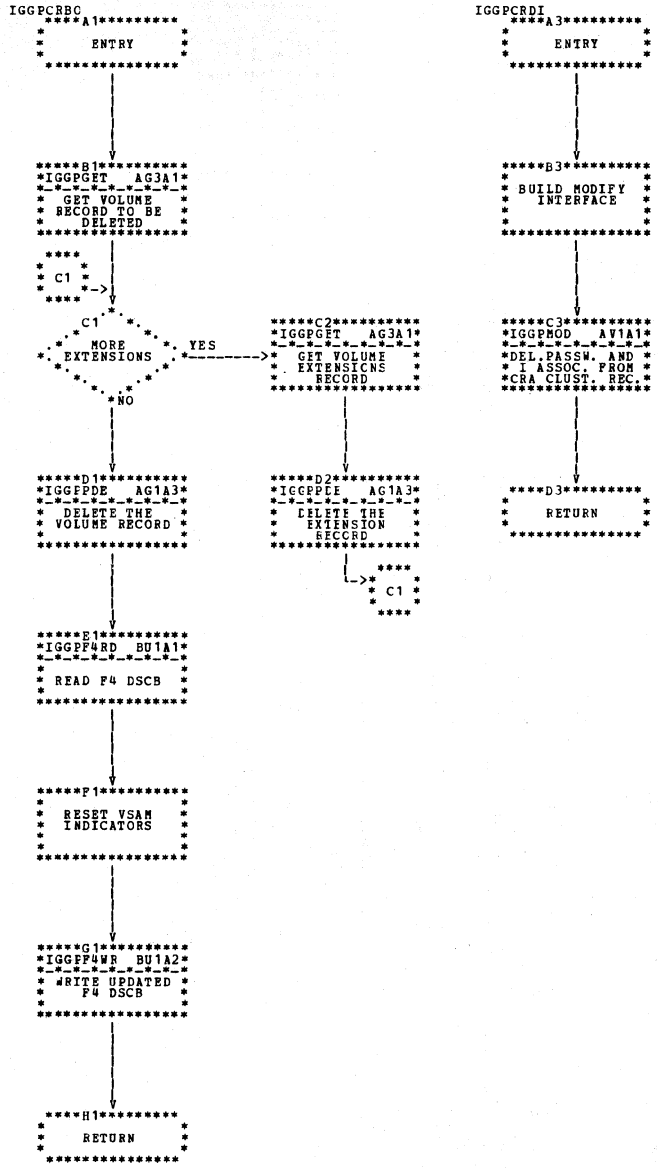


Chart CR2. Define catalog recovery area (IGG0CLCR)

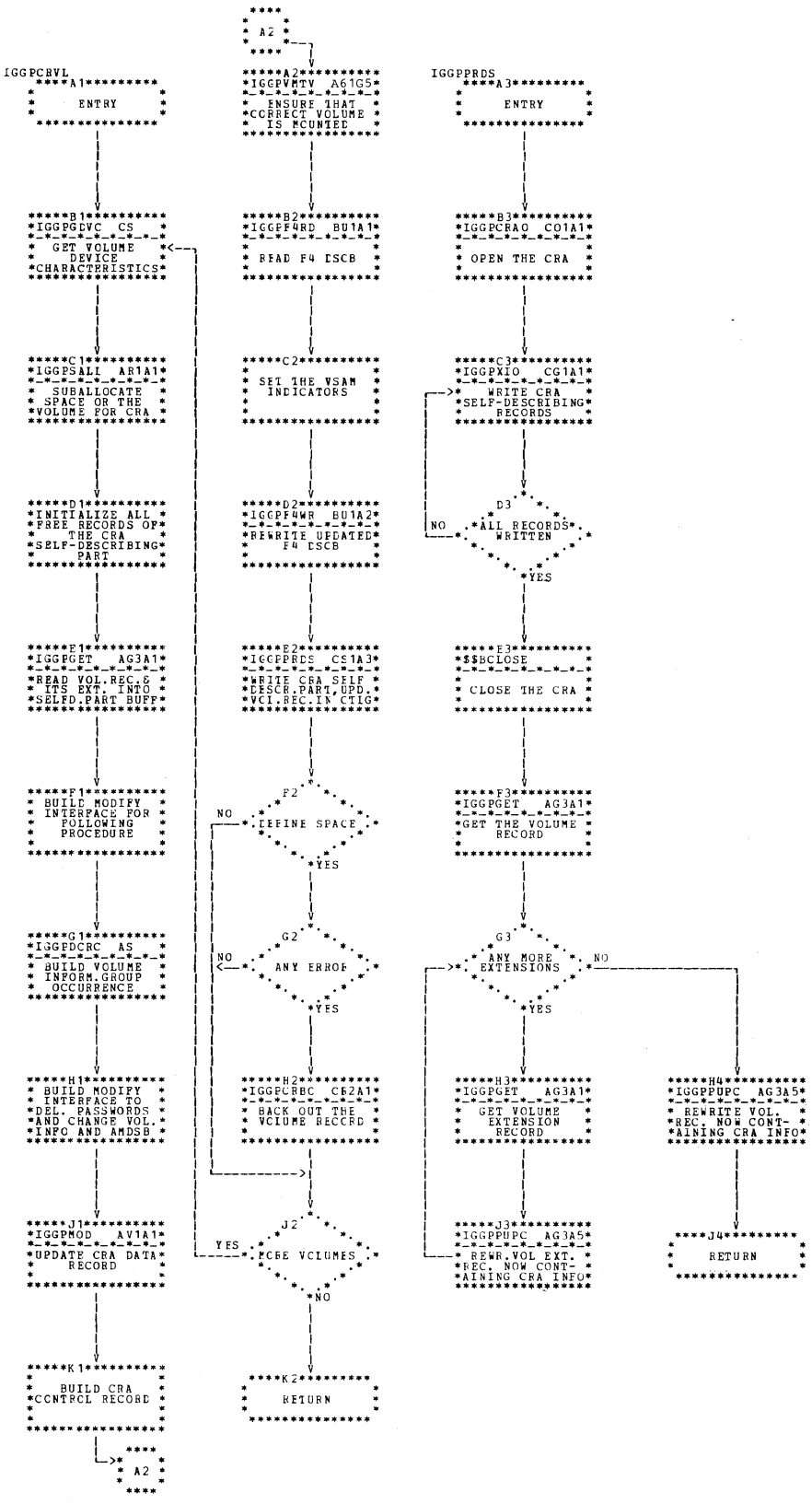


Chart CS1. Define catalog recovery area - second module (IGG0CLCS)

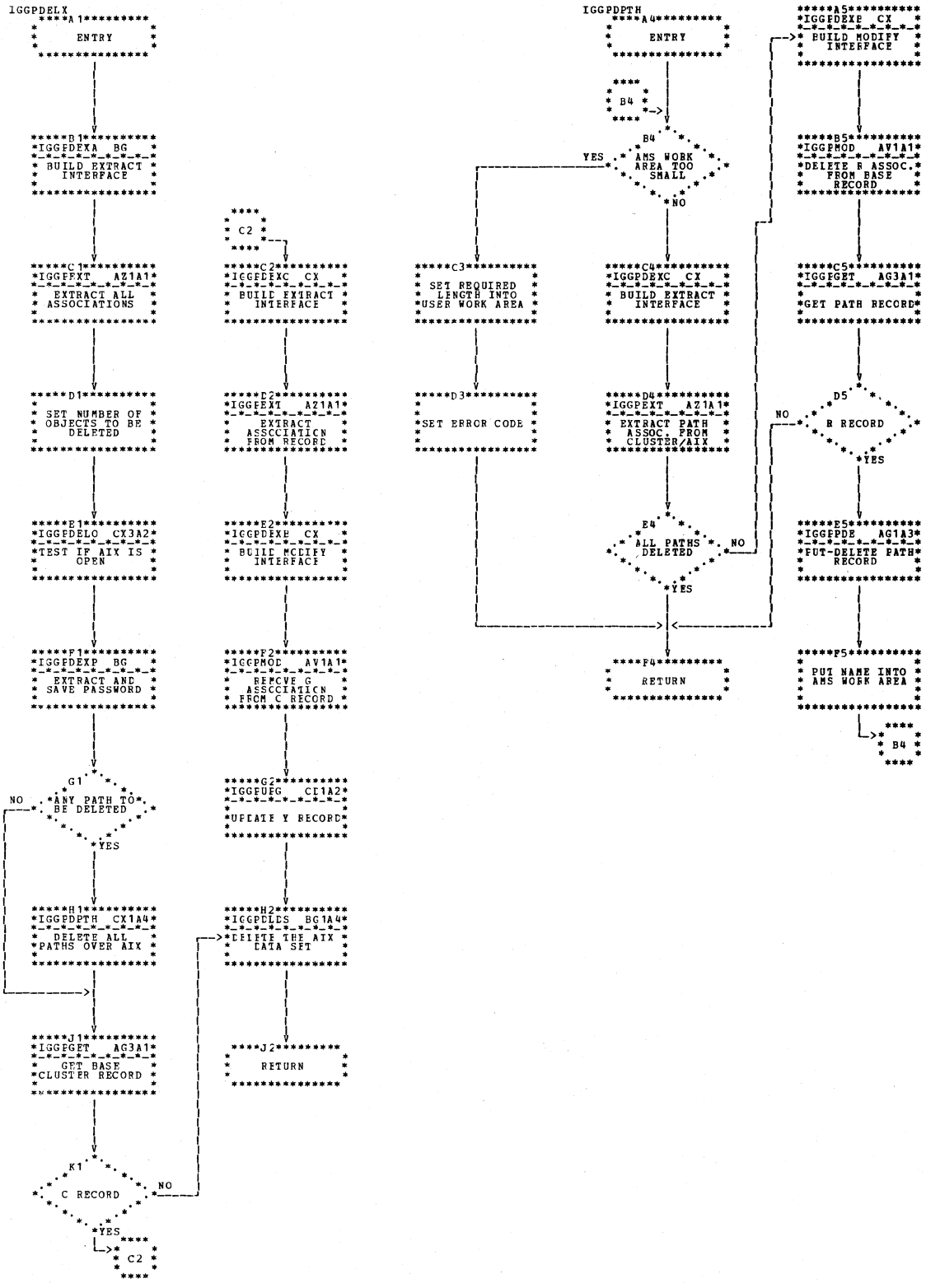


Chart CX1. CMS delete - third module (IGG0CLCX)

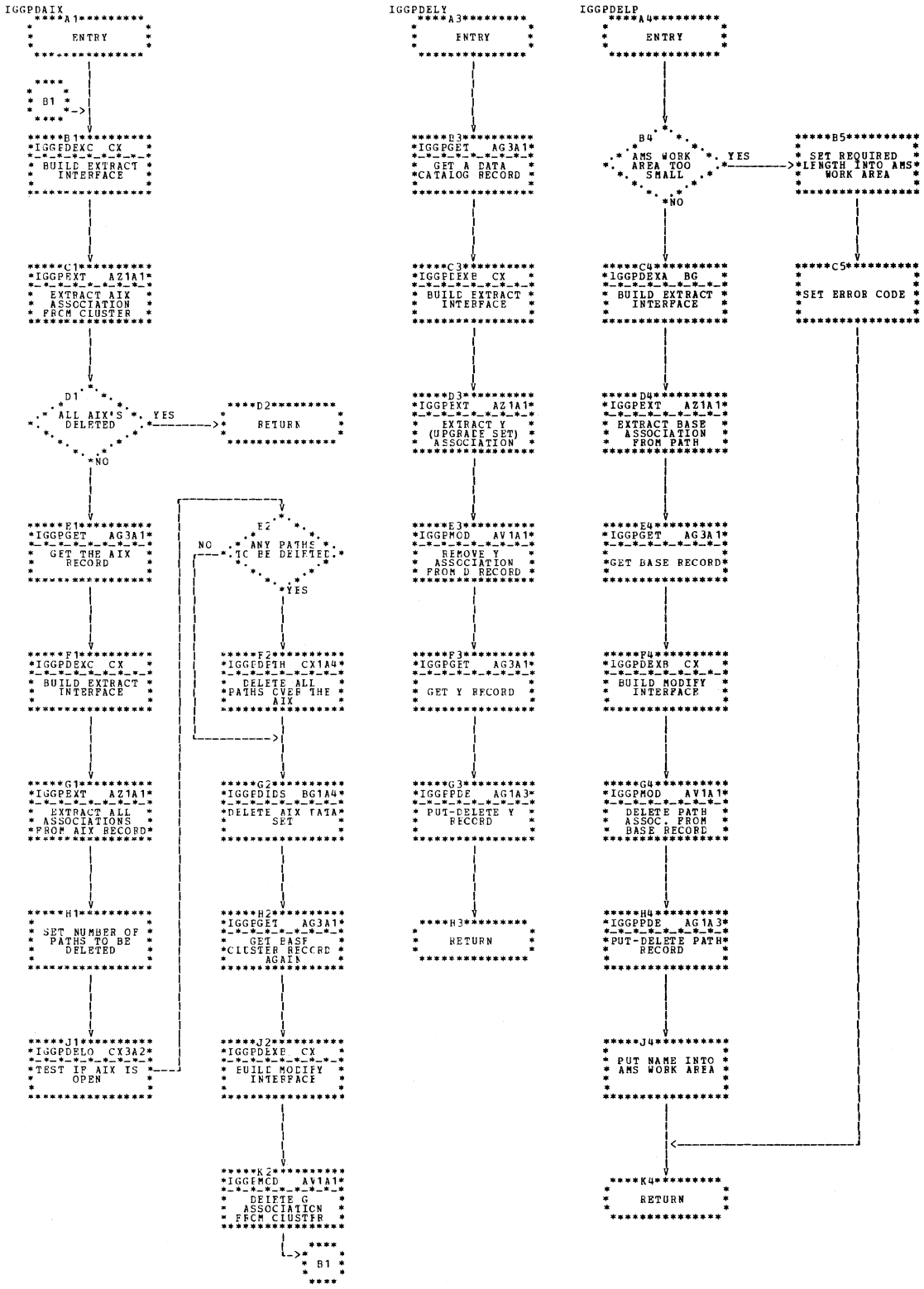


Chart CX2. CMS delete - third module (IGG0CLCX)

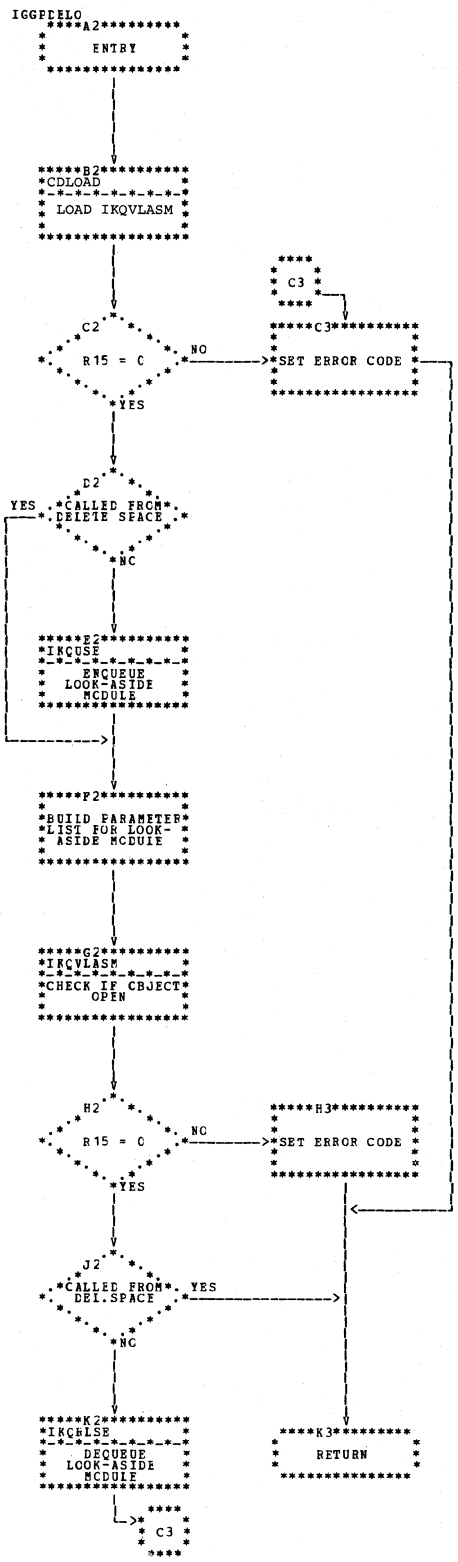


Chart CX3. CMS delete - third module (IGG0CLCX)

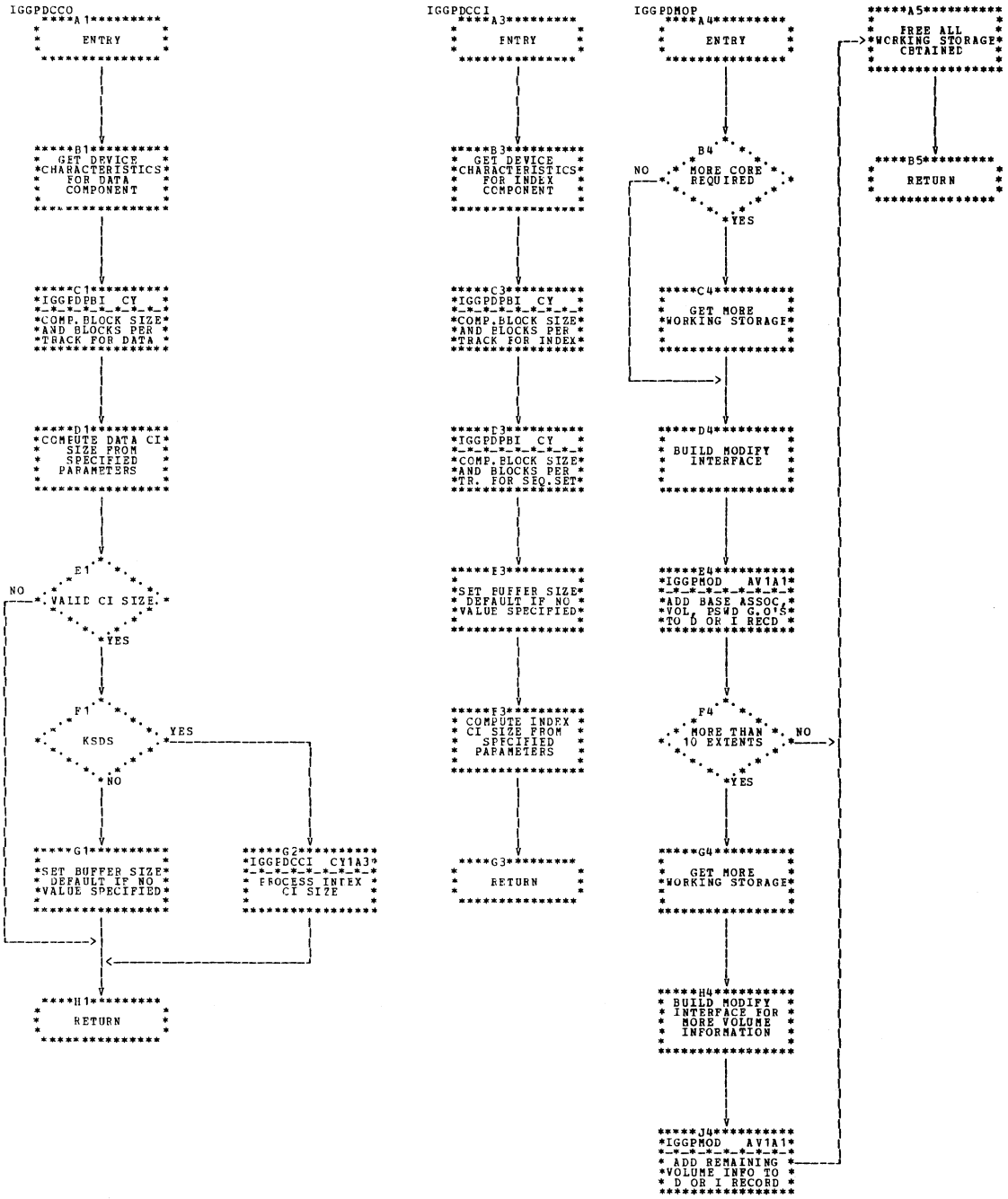


Chart CY1. CMS define - sixth module (IGG0CLCY)

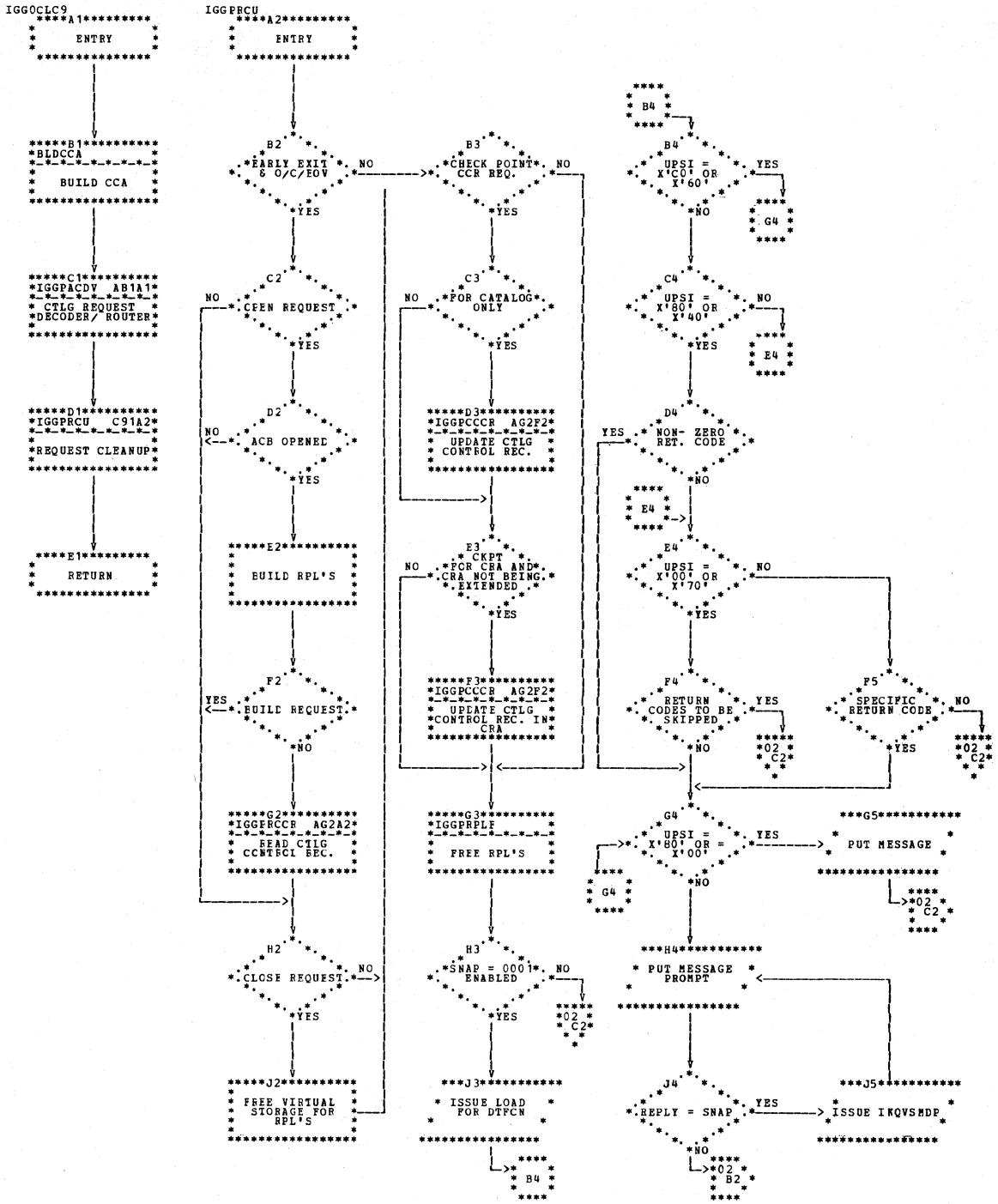


Chart C91. Catalog first module (IGG0CLC9)

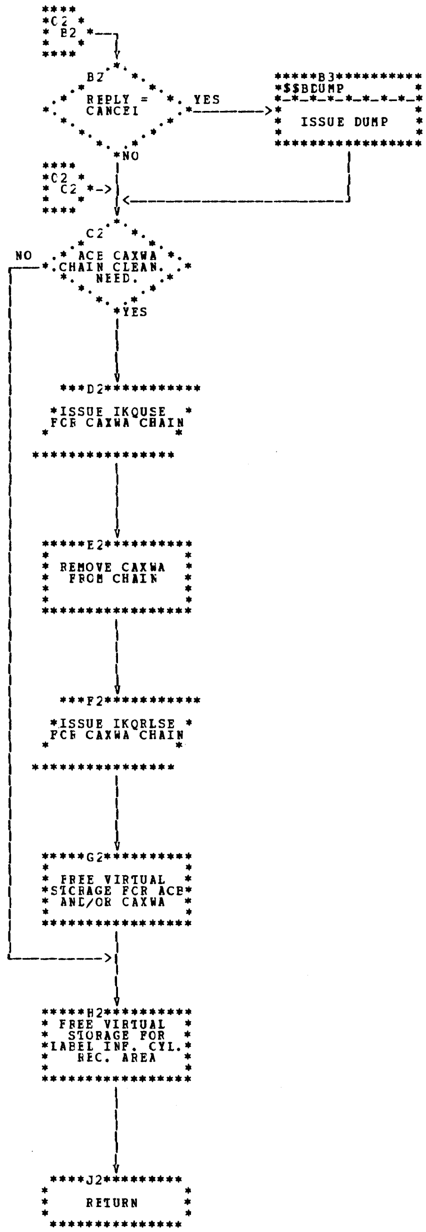


Chart C92. Catalog first module (IGG0CLC9)

Section 4. Directory

This section contains the following cross-reference material:

- VSAM Phase-to-Module Index
- IIP Phase-to-Module Index
- Component Index
- Module Directory
- Routine Directory
- Catalog External Entry Points
- Data Area Directory

VSAM Phase-to-Module Index

The core image library contains the DOS/VS VSAM phases. Their names are identifiable by IKQV or $$B. Packaged within the phases are the DOS/VS VSAM modules, identifiable by the leading characters IKQ, IGG0, or $$B. Two service aid phases, IKQVDU and IKQVEDA are not included in the link-edit of VSAM and must be placed in the core image library by executing a job described in *Service Aids*.

The following list includes the phase names and the names of the modules included within each phase.

Phase name	Module name(s)			
IKQVBRP	IKQBRP			
IKQVCAT	IGG0CLAB	IGG0CLAU	IGG0CLBH	IGG0CLCG
	IGG0CLAC	IGG0CLAV	IGG0CLBL	IGG0CLCL
	IGG0CLAD	IGG0CLAW	IGG0CLBM	IGG0CLCO
	IGG0CLAE	IGG0CLAX	IGG0CLBN	IGG0CLCP
	IGG0CLAF	IGG0CLAY	IGG0CLBQ	IGG0CLCR
	IGG0CLAG	IGG0CLAZ	IGG0CLBR	IGG0CLCS
	IGG0CLAH	IGG0CLA6	IGG0CLBS	IGG0CLCX
	IGG0CLAJ	IGG0CLA7	IGG0CLBT	IGG0CLCY
	IGG0CLAK	IGG0CLB8	IGG0CLBU	IGG0CLC9
	IGG0CLAL	IGG0CLBA	IGG0CLBW	IKQALL00
	IGG0CLAN	IGG0CLBB	IGG0CLBX	IKQCOV00
	IGG0CLAP	IGG0CLBC	IGG0CLBY	IKQPOP00
	IGG0CLAQ	IGG0CLBD	IGG0CLB8	IKQRDS00
	IGG0CLAR	IGG0CLBE	IGG0CLCA	IKQREN00
	IGG0CLAS	IGG0CLBF	IGG0CLCB	IKQSCR00
	IGG0CLAT	IGG0CLBG	IGG0CLCD	IKQVTC00
				IKQWSD00
IKQVCLC	IKQCLCAT			
IKQVCLOC	IKQCLOCL			
IKQCLOS	IKQCLO			
IKQVCLOV	IKQCLOVY			
IKQVDCN	IKQDCN			
IKQVDNT	IKQDNT			
IKQVDRP	IKQDRP			
IKQVDTPE	IKQVDTPE			
IKQVDU	IKQVDU			
IKQVDUMP	IKQDUMP			
	IKQDUMPC			
IKQVEDA	IKQVEDA			
IKQVEDX	IKQEDX			
IKQVEOV	IKQEOV			
IKQVGEN	IKQGEN			
IKQVJIBS	IKQJIBSM			
IKQVLAB	IKQLAB			
IKQVLASF	IKQLASFT			
IKQVLASM	IKQLASMD			
IKQVMSG	IKQOCMSG			
IKQVNEX	IKQNEX			
IKQVOPEN	IKQOPN	IKQOPNOV		
	IKQOPNAI	IKQOPNRD		
	IKQOPNCT	IKQOPNRP		
	IKQOPNDO	IKQOPNUC		
	IKQOPNHC	IKQOPNUS		
	IKQOPNNC			
IKQVPBF	IKQPBF00			
IKQVRBA	IKQRBA			

Figure 4.1 **VSAM phase-to-module index (part 1 of 2)**

Phase name	Module name(s)			
IKQVRM	IKQAIX	IKQIOA00	IKQMDY	IKQSPM00
	IKQBFA00	IKQIOB00	IKQNCA00	IKQSRG
	IKQBFB00	IKQINT	IKQPFO00	IKQSRT
	IKQBFC00	IKQIXE00	IKQRCL00	IKQSRU
	IKQBLD	IKQIXF00	IKQRQA	IKQUPD
	IKQCAS00	IKQRQB	IKQUPG	IKQVFY
	IKQCIR	IKQIXS00	IKQRQC	IKQVSM
	IKQCIS00	IKQJRN	IKQRRP	
	IKQERH	IKQKRD	IKQRTV	
	IKQERX	IKQLCD	IKQSCN	
	IKQGNX00	IKQLCN	IKQSFT	
	IKQGPT	IKQLCP		
IKQVRT	IKQVRT			
IKQVSCAT	IKQSCAT			
IKQVSTM	IKQSTM			
IKQVTMS	IKQTMSD IKQTMSF			
$$BACLOS	$$BACLOS			
$$BCLCRA	$$BCLCRA			
$$BCVSAM	$$BCVSAM			
$$BCVS02	$$BCVS02			
$$BCVS03	$$BCVS03			
$$BCVS04	$$BCVS04			
$$BJIBFF	$$BJIBFF			
$$BJIB00	$$BJIB00			
$$BODADE	$$BODADE			
$$BODADS	$$BODADS			
$$BOVSAM	$$BOVSAM			
$$BOVS01	$$BOVS01			
$$BOVS03	$$BOVS03			
$$BTCLOS	$$BTCLOS			

Figure 4.1 **VSAM phase-to-module index (part 2 of 2)**

IIP Phase-to-Module Index

The core image library contains the ISAM Interface Program phases, identifiable by the first three characters IIP or $$B. Packaged Within the phases are the IIP modules. The following list includes the phase names and the names of the modules included within each phase.

Phase name	Module name(s)
$$BOCISC	IIPBMR00
IIPCLOSE	IIPCLS00
IIPOPEN	IIPOPN00
IIPPROC	IIPPRCPR IIPPRCMR
IIPAMDTF	IIPAMT00

Figure 4.2 **IIP phase-to-module index**

Component Index

VSAM is logically grouped into components, each of which consists of several modules. This index (Figure 4.3) lists these components in the following order: catalog, control block manipulation, open close EOV, DADSM, ISAM interface, record management, and service aids.

Component	Module name	Module function
Catalog	IGG0CLAB	Act as switching station for various catalog routines
	IGG0CLAC[1]	Check whether VSAM master catalog is open
	IGG0CLAD[1]	Open VSAM master catalog
	IGG0CLAE[2]	Open and create VSAM catalog and write self-describing catalog records
	IGG0CLAF	Delete VSAM catalog
	IGG0CLAG	Perform VSAM catalog I/O subfunctions
	IGG0CLAH	Search VSAM catalog for required entry
	IGG0CLAJ[3]	Build data and index entries and allocate space
	IGG0CLAK[3]	Build data and index entries and construct fields in records
	IGG0CLAL[4]	Perform general Define processing
	IGG0CLAN[4]	Perform Define processing and construct cluster entry
	IGG0CLAP[4]	Perform Define processing and check AMDSBs, volume lists, and space parameters
	IGG0CLAQ[5]	Define VSAM data space
	IGG0CLAR[6]	Initialize for VSAM space suballocation
	IGG0CLAS[2]	Define catalog, allocate physical space, and initialize preliminary records
	IGG0CLAT	Act as Access Method Services request dispatcher to catalog functions
	IGG0CLAU[6]	Suballocate VSAM space
	IGG0CLAV[7]	Modify VSAM catalog fields
	IGG0CLAW[7]	Add new VSAM catalog fields
	IGG0CLAX[7]	Alter VSAM catalog fields
	IGG0CLAY[7-8]	Initialize and scan catalog parameter list
	IGG0CLAZ[8]	Extract VSAM catalog fields
	IGG0CLA6[5]	Define VSAM space
	IGG0CLA7[9]	Delete an entry from catalog and, if a unique data set on more than one volume, mount other volume and delete an entry from catalog for that volume

1 IGG0CLAC and IGG0CLAD are related master catalog open processing modules.
2 IGG0CLAE and IGG0CLAS are related Define (catalog build and open) modules.
3 IGG0CLAJ, IGG0CLAK, and IGG0CLA8 are related Define modules.
4 IGG0CLAL, IGG0CLAN, IGG0CLAP, IGG0CLBX, and IGG0CLBY are related modules commonly known as the Define routine.
5 IGG0CLAQ and IGG0CLA6 are related Define space modules.
6 IGG0CLAR and IGG0CLAU are related space suballocation modules.
7 IGG0CLAV, IGG0CLAY, IGG0CLBA, IGG0CLAW, IGG0CLAX, IGG0CLBW, IGG0CLBT and IGG0CLBS are related modules commonly known as the Modify routine.
8 IGG0CLAZ, IGG0CLAY, and IGG0CLBA are related modules commonly known as the Extract routine.
9 IGG0CLBG, IGG0CLA7 and IGG0CLX are related delete catalog entry modules.
10 IGG0CLBD, IGG0CLBE, IGG0CLBN, and IGG0CLCD are related Alter processing modules.

Figure 4.3 Component index (part 1 of 5)

Component	Module name	Module function
Catalog	IGG0CLA8[3]	Perform Define processing and free storage resources
	IGG0CLBA[7-8]	Test VSAM catalog fields
	IGG0CLBB[9]	Extend VSAM data sets
	IGG0CLBC[9]	Initialize for extending VSAM data sets
	IGG0CLBD[10]	Alter an entry in catalog except when processing volumes
	IGG0CLBE[10]	Alter a volume entry and add data set directory to volume entry
	IGG0CLBF	Release space to catalog
	IGG0CLBG[9]	Delete an entry from catalog
	IGG0CLBH	Define a non-VSAM entry in VSAM catalog
	IGG0CLBL	Delete a VSAM data space, mount volume, process F4 DSCBs, remove data from volume record, and scratch DASD space
	IGG0CLBM	Check authorization of catalog user, prompt terminal, and compare password
	IGG0CLBN[10]	Remove volumes for Alter processing and remove data set directories from volume entry
	IGG0CLBQ	List contents of catalog
	IGG0CLBR	Perform bit manipulation against VSAM space bit map
	IGG0CLBS[7]	Retrieve derived VSAM catalog fields
	IGG0CLBT[7]	Modify derived VSAM catalog fields
	IGG0CLBU	Read and/or write F4 DSCBs
	IGG0CLBW[7]	Modify VSAM catalog by deleting or inserting fields
	IGG0CLBX[4]	Define data set entries and calculate size
	IGG0CLBY[4]	Define data set entries and calculate space
	IGG0CLB8	Back-out Define processing and restore allocated space
	IGG0CLCA	Define alternate index
	IGG0CLCB	Release function
	IGG0CLCD	CMS alter (4th module)
	IGG0CLCG	I/O subroutine (2nd module)
	IGG0CLCL	CMS delete space (2nd module)
	IGG0CLCO	Open catalog recovery area
	IGG0CLCP	Define path
	IGG0CLCR	Define CRA (first module)
	IGG0CLCS	Define CRA (second module)
	IGG0CLCX	CMS delete (3rd module)
	IGG0CLCY	CMS define (6th module)
	IGG0CLC9	Act as general interface and build CCA
	IKQDCN	Define console file
	IKQDNT	Define device name and characteristics table
	IKQSCAT	Display catalog information
	IKQVDTPE	Catalog device type and label cylinder routine
CB Manip.	IKQBRP	Build VSAM resource pool
	IKQDRP	Delete VSAM resource pool
	IKQGEN	Build ACB, RPL, or EXLST
	IKQTMSD	Test, modify, or display ACB, RPL, or EXLST (with diagnosis of input)
	IKQTMSF	Test, modify, or display ACB, RPL, or EXLST (without diagnosis of input)
	IKQVRT	VSAM Shared Resource Table

Figure 4.3 Component index (part 2 of 5)

Component	Module name	Module function
O/C/EOV	IKQCLCAT	Update permanent data set information in the catalog
	IKQCLO	Disconnect a user's program from a VSAM data set
	IKQCLOCL	Alternate index clean up
	IKQCLOVY	Alternate index evaluation
	IKQEDX	Extend an EDB when the control blocks need to reflect additional space
O/C/EOV	IKQEOV	Mount a volume when the required volume is not mounted
	IKQJIBSM	Summarize extents in EDB chain so JIBs can be built
	IKQLAB	Read label information cylinder record (DLBL/EXTENT statements)
	IKQLASFT	Map a look-aside file table that contains all data sets that are open
	IKQLASMD	Ascertain if the data set is currently open
	IKQNEX	Get a new extent when space is needed
	IKQOCMSG	Open/Close message routine
	IKQOPN	Connect a user's program with a VSAM data set
	IKQOPNAI	Alternate index initialization
	IKQOPNCT	Open a catalog by means of special processing
	IKQOPNDO	Clean up after a failure to open a data set
	IKQOPNHC	Locate data set information in catalog
	IKQOPNNC	Next cluster
	IKQOPNOV	Build ARDB, EDB, LPMB and call IKQJIBSM for volume and extent processing
	IKQOPNRD	Reset reusable data set
	IKQOPNRP	Attach data set to resource pool
	IKQOPNUC	User catalog open
	IKQOPNUS	Alternate index upgrade set determination
	IKQRBA	Update the catalog
	IKQSTM	Storage management
	$$BACLOS	Automatic close
	$$BCLCRA	Mark deleted CRA
	$$BCVSAM	Provide an interface between DOS/VS and VSAM when a data set is closed
	$$BCVS02	Provide common exit processing for VSAM modules
	$$BOVSAM	Provide an interface between DOS/VS and VSAM when a data set is opened
	$$BOVS01	Provide an interface to the DOS/VS message writer to get a volume mounted for open or catalog/DADSM processing
	$$BOVS03	Delete or build JIBs for VSAM
	$$BTCLOS	Provide an interface between DOS/VS and VSAM when a data set is temporarily closed
DADSM	IKQALL00	Create a new F1 DSCB (and F3 DSCBs) from the system label cylinder record
	IKQCOV00	Check for overlap on existing files before allowing new DSCB to be written
	IKQPOP00	Build F1 and any needed F3 DSCBs from label cylinder record (subfunction or IKQALL00)
	IKQRDS00	Read VTOC records either by key or disk address
	IKQREN00	Rename a specified F1 DSCB to a new specified name

Figure 4.3 Component index (part 3 of 5)

Component	Module Name	Module Function
	IKQSCR00	Remove an F1 DSCB (and any associated F3 DSCBs) from the VTOC
	IKQVTC00	Order creation or deletion of JIBs in a DASD-file protected system
	IKQWDS00	Write VTOC records either by key or disk address
	$$BODADE	Interface to DADSM from the DOS/VS message writer
	$$BODADS	Interface to the DOS/VS message writer from DASDM
	$$BJIBFF	Delete a JIB in a DASD-file protected system
	$$BJIB00	Create a JIB in a DASD-file protected system
ISAM	IIPAMT00	Map the AMDTF table
interface	IIPBMR00	Issue an error message if a failure occurs when an ISAM program is trying to open or close a VSAM data set
	IIPCLS00	Close a VSAM data set for an ISAM program
	IIPIIP00	Link-edit phase and include statements
	IIPOPN00	Open a VSAM data set for an ISAM program
	IIPPRCMR	Issue error messages and cancel tasks in case an error occurred in IIP; issue a VSAM CLOSE for the data set if an error occurred during function other than Open or Close
	IIPPRCPR	Transform an ISAM request into an equivalent VSAM request
Rec. Mgmt	IKQAIX	Alternate index routine
	IKQBFA00	Control buffers and their contents
	IKQBFB00	LSR buffer management
	IKQBFC00	Track-hold control
	IKQBLD	Build RDFs for all changes to a control interval
	IKQBRP	Build VSAM resource pool
	IKQCAS00	Split a control area or high-level index record
	IKQCIR	CNV space reclamation routine
	IKQCIS00	Split a control interval or get a new control interval
	IKQDRP	Delete VSAM resource pool
	IKQERH	Handle errors for record management modules
	IKQERX	Process error exits for record management modules
	IKQGNX00	Get next buffer and read records into buffers in anticipation of further user request processing
	IKQGPT	Handle GET or POINT user requests
	IKQIOA00	Build channel programs for READs and WRITEs and process I/O
	IKQIOB00	Analyze hardware errors encountered in IKQIOA00
	IKQINT	PLH initialization for LSR processing
	IKQIXE00	Make index entries and create high-level indexes
	IKQIXF00	Balance section entries in index record
	IKQIXS00	Search the index for desired key
	IKQJRN	Journad Exit
	IKQKRD	Initialize for key-range requests
	IKQLCD	Locate a specific record by key or RBA
	IKQLCN	Locate next sequential (logical or physical) record
	IKQLCP	Get backwards function (locate previous)

Figure 4.3 Component index (part 4 of 5)

Component	Module Name	Module Function
	IKQMDY	Modify a control interval (insert, update, or delete)
	IKQNCA00	Construct a new control area
	IKQPBF00	Complete I/O processing already initiated preparatory to mounting another volume
	IKQPFO00	Format a data control area or index control interval and write SEOF
	IKQRCL00	Clean up record management requests before a VSAM CLOSE can be completed
	IKQRQA	Analyze record management requests
	IKQRQB	Complete analysis of record management requests
	IKQRQC	PLH assignment for AIX processing (LSR)
	IKQRRP	Relative record preformat
	IKQRTV	Retrieve a specific record for caller
	IKQSCN	Scan a control interval for a specific record
	IKQSFT	Shift data in a control interval
	IKQSPM00	Apportion data or index space within extents
	IKQSRG	Spanned record GET
	IKQSRT	Insert a record in a control interval
	IKQSRU	Spanned record update
	IKQUPD	Update a record in a control interval
	IKQUPG	Alternate index upgrade routine
	IKQVFY	Reestablish high-used and high-key RBAs (VERIFY function)
	IKQVSM	Perform initial processing for all record management requests and activate the modules that perform the operations
Service Aids	IKQCLEAN	DADSM utility
	IKQDUMP	Dump non-catalog control blocks
	IKQDUMPC	Dump catalog control blocks
	IKQVEDA	Enable and disable VSAM snap dump routine
	$$BCVS03	Load a phase
	$$BCVS04	I/O routine for IKQVEDA

Figure 4.3 Component index (part 5 of 5)

Module Directory

The module directory (Figure 4.4) is organized alphabetically by symbolic module name. It lists the descriptive name, the component to which that module belongs, the method of operation diagram and program structure figure numbers in which that module is referenced, and the external entry point(s).

Module name	Descriptive name	Component	Diag.#	Structure Figure 3.x	External entry points
IGG0CLAB	Catalog driver	Catalog	DB, DC	5	IGGPACDV IGGPRPLF IGGPRPLM
IGG0CLAC	Master catalog search	Catalog	DB	5	IGGPMCO
IGG0CLAD	Master catalog open	Catalog	DB, EE	5	IGGPMCO2
IGG0CLAE	Define catalog open and build	Catalog	EE, EH	-	IGGPDCOC IGGPMEBM IGGPDCBO
IGG0CLAF	Delete catalog	Catalog	EH, EO	5	IGGPDELC IGGPEMIO IGGPEMSG IGGPSDSP
IGG0CLAG	Catalog I/O subfunctions	Catalog	DB, DC, DG-DL, EC-EH, EK-EP, ER	5	IGGPGET IGGPISCI IGGPPUPC IGGPPAD IGGPPDE IGGPPDEC IGGPAOCI IGGPAXCI IGGPCCCR IGGPRCCR
IGG0CLAH	Search catalog	Catalog	DB-DC	5	IGGPSCAT IGGPSCA
IGG0CLAJ	Define and build data and index entries	Catalog	EC-ED,EI	-	IGGPDBDI IGGPDEXD
IGG0CLAK	Complete define of an entry	Catalog	ED	-	IGGPDCMB
IGG0CLAL	CMS define, 1st module	Catalog	EC, EO	5	IGGPDEF IGGPDTIM IGGPDCAV IGGPDDEP IGGPDCDE IGGPDWAI IGGPOSTY
IGG0CLAN	CMS define, 2nd module	Catalog	EC-EF,EI, EM,EP	-	IGGPDSCB IGGPDBSF IGGPDRDA IGGPDCCE IGGPDUND
IGG0CLAP	CMS define, 3rd module	Catalog	EC,EE	-	IGGPDCDA
IGG0CLAQ	Catalog define space	Catalog	ED-EE,EG	5	IGGPDEFS
IGG0CLAR	Suballocate	Catalog	DH, DJ, ED, EE, EH, EK	-	IGGPSALL
IGG0CLAS	VSAM catalog definition processing	Catalog	EC, EE, EH	-	IGGPDEFC IGGPDCRC
IGG0CLAT	CMS driver	Catalog	DB, EE	5	IGGPCDVR
IGG0CLAU	Suballocation	Catalog	DJ	-	IGGPSALS

Figure 4.4 **Module directory (part 1 of 7)**

Module name	Descriptive name	Component	Diag.#	Structure Figure 3.x	External entry points
IGG0CLAV	Modify catalog field	Catalog	DB, DG, DH, DJ, DL, ED-EH, EK, EM EN, EP, ER	3,5	IGGPMOD IGGPUPD IGGPSGOP IGGPDEL2
IGG0CLAW	Add group occurrence (modify)	Catalog	DG, DK, DL	-	IGGPADGO IGGPGNEX IGGPIGOP IGGPPREC
IGG0CLAX	Alter catalog field	Catalog	DG, DL	-	IGGPALT2 IGGPEXPD IGGPSHNK IGGPDGOP IGGPMGO IGGPDGO
IGG0CLAY	Scan CPL	Catalog	DB, DL	5	IGGPSCNC
IGG0CLAZ	Extract catalog field	Catalog	DB, DD DE, DH DJ, EK-EP	5 - -	IGGPEXT IGGPLOC
IGG0CLA6	CMS define space (part 2)	Catalog	- EG, EH		IGGPDFS2 IGGPF4PR IGGPVMTV
IGG0CLA7	CMS delete (part 2)	Catalog	EM, EP	-	IGGPVMSC IGGPDUSC IGGPDEMV IGGPDVMV
IGG0CLA8	Define clean up	Catalog	ED	-	IGGPDFRS
IGG0CLBA	Tests	Catalog	DB, DE, DG, DK DL	5	IGGPTSTS IGGPGVAL IGGPGREC
IGG0CLBB	Update extend	Catalog	DB, DH	-	IGGPUPDE
IGG0CLBC	Update extend initialization	Catalog	DH	-	IGGPINIT IGGPSVOL
IGG0CLBD	Catalog alter processing	Catalog	EK	5	IGGPALT
IGG0CLBE	Alter volume processing	Catalog	EK	-	IGGPALVL IGGPALEC
IGG0CLBF	Subscratch	Catalog	EM, EP	-	IGGPSSCR
IGG0CLBG	Delete	Catalog	EM, EP	5	IGGPDEL IGGPDLDS IGGPDEXA IGGPDEXP IGGPDLXT
IGG0CLBH	Define non-VSAM data set	Catalog	EF	5	IGGPDEFA
IGG0CLBL	Delete space	Catalog	EN	5	IGGPDELS
IGG0CLBM	Check authorization	Catalog	DB, DD, EL	5	IGGPCKAU
IGG0CLBN	Catalog alter, remove volume processing	Catalog	EK	-	IGGPALVR IGGPALSV
IGG0CLBQ	LISTCAT processing	Catalog	EL	5	IGGPLSTC
IGG0CLBR	Suballocate bit mask handler	Catalog	DJ	-	IGGPBMR
IGG0CLBS	Volume entry translation	Catalog	DK	-	IGGPXVAL IGGPXEXT

Figure 4.4 **Module directory (part 2 of 7)**

Module name	Descriptive name	Component	Diag.#	Structure Figure 3.x	External entry points
IGG0CLBT	Modify volume entry translation	Catalog	DL	-	IGGPXMOD IGGPXLT2 IGGPXEL2 IGGPXDGO
IGG0CLBU	Catalog read/write F4 DSCB	Catalog	EG, EH, EN, EO	-	IGGPF4RD IGGPF4WR
IGG0CLBW	Delete/insert (modify)	Catalog	DL	-	IGGPDEIN
IGG0CLBX	CMS define, 4th module	Catalog	EC, ED, EI	-	IGGPDSPC IGGPDALR IGGPDSPF
IGG0CLBY	CMS define, 5th module	Catalog	ED	-	IGGPDRSP
IGG0CLB8	Define, space recovery	Catalog	-	-	IGGPDFBO IGGPCNBO
IGG0CLCA	Define AIX	Catalog	EI, ER	5	IGGPAIX IGGPPRPW IGGPCKEN IGGPBOUT
IGG0CLCB	Release function	Catalog	ED	3	IGGPRELE IGGPPTBF IGGPCLBF IGGPGTBF
IGG0CLCD	CMS alter, 4th module	Catalog	EI, EP, ER	-	IGGPUPG IGGPALY
IGG0CLCG	VSAM catalog I/O subroutine (2nd load)	Catalog	EH	-	IGGPRBAP IGGPIORA IGGPCHAC IGGPTRPL IGGPXIO
IGG0CLCL	CMS delete space (2nd module)	Catalog	EN	-	IGGPDLSF IGGPDVSC
IGG0CLCO	Open CRA	Catalog	EH, EN	-	IGGPCRAO IGGPGLUB IGGPSCAX
IGG0CLCP	Define path	Catalog	EJ	5	IGGPPATH
IGG0CLCR	Define CRA (1st module)	Catalog	EH	-	IGGPCADR IGGPCRBO IGGPBCRA IGGPCRDI
IGG0CLCS	Define CRA (2nd module)	Catalog	EH	-	IGGPCRVL IGGPPRDS
IGG0CLCX	CMS delete (3rd module)	Catalog	EM, EN, EP	-	IGGPDELX IGGPDPTH IGGPDELP IGGPDAIX IGGPDELY IGGPDELO
IGG0CLCY	CMS define (6th module)	Catalog	ED	-	IGGPDCCO IGGPDPBI IGGPDMOP

Figure 4.4 **Module directory (part 3 of 7)**

Module name	Descriptive name	Component	Diag.#	Structure Figure 3.x	External entry points
IGG0CLC9	Catalog first load	Catalog	DB	3, 5, 10	IGG0CLC9
IIPAMT00	AMDTF (control block)	ISAM interface	-	-	IIPAMT00
IIPBMR00	$$B message routine	ISAM interface	CH	4	IIPBMR00
IIPCLS00	Close	ISAM interface	CG, CH	4	IIPCLS00
IIPIIP00	Phase and include statements	ISAM interface	-	-	
IIPOPN00	Open	ISAM interface	CB, CH	4	IIPOPN00
IIPPRCMR	Processor (messages)	ISAM interface	CH	4	IIPPRCMR
IIPPRCPR	Processor (processing)	ISAM interface	CC-CF, CH	4	IIPPRCPR
IKQAIX	AIX routine	Rec. Mgmt.	GC		IKQAIX
IKQALL00	Allocate data spaces	DADSM	EG, FB	6	IKQALL00
IKQBFA00	Buffer manager	Rec. Mgmt.	GQ, GU, HI-HN, IA	9, 14	IKQBFA00*
IKQBFB00	LSR buffer manager	Rec. Mgmt,	HV-HZ	9, 14	IKQBFB00*
IKQBFC00	Track hold control	Rec. Mgmt.	HI	14	IKQBFC00
IKQBLD	RDF-build, non-spanned records	Rec. Mgmt.	GP	9, 12	IKQBLD
IKQBRP	Build resource pool	CB Manip.	AH	-	IKQBRP
IKQCAS00	Control area split	Rec. Mgmt.	GT, GZ	10	IKQCAS00*
IKQCIR	CNV space reclamation routine	Rec. Mgmt.	GT, GU	10	IKQCIR
IKQCIS00	Control interval split	Rec. Mgmt.	GT, IA	9, 10	IKQCIS00
IKQCLCAT	Close catalog interface function	O/C/EOV	GX, IA	15	IKQCLCAT
IKQCLEAN	VTOC maintenance utility	Service aids		-	IKQCLEAN
IKQCLNLK	Phase and include statement	VSAM include			IKQCLEAN
IKQCLO	Close function	O/C/EOV	1A	15	IKQCLO00
IKQCLOCL	AIX clean up	O/C/EOV			IKQCLOCL
IKQCLOVY	AIX evaluation	O/C/EOV			IKQVCLOV
IKQCOV00	Check for overlapping extents	DADSM	FG, FH	6	IKQCOV00
IKQDCN	DTF console file	Catalog	DD		IKQDCN
IKQDNT	Device name table	Catalog	EF	-	IKQDNT
IKQDRP	Delete resource pool	CB Manip.	AI	-	IKQDRP
IKQDUMP	Block dump	Service aids		-	IKQDUMP IKQDUMPP
IKQDUMPC	Dump catalog control blocks	Service aids		-	IKQDUMPC
IKQEDX	EDB extend	O/C/EOV	HP, HS	11	IKQEDX00
IKQEOV	Mount volume	O/C/EOV	HP, HR	14	IKQEOV00
IKQERH	Error handler	Rec. Mgmt.	HF	7-9, 12, 13	IKQERH
IKQERX	VSAM error exit	Rec. Mgmt.	HG	7-9, 12, 13	IKQERX
IKQGEN	GENCB: Build a new control block	CB Gener.	AC	2	IKQGEN00

* For further entry points, see Figure 4.6

Figure 4.4 Module directory (part 4 of 7)

Module name	Descriptive name	Component	Diag.#	Structure Figure 3.x	External entry points
IKQGNX00	Get next buffer and read ahead	Rec. Mgmt.	GQ	7-9, 12	IKQGNX00
IKQGPT	Get/Point	Rec. Mgmt.	GE-GF	7, 8	IKQGPT
IKQINT	PLH initialization	Rec. Mgmt.		9	
IKQIOA00	I/O manager	Rec. Mgmt.	HO-HQ, HT	14	IKQIOA00*
IKQIOB00	I/O manager, I/O error analysis	Rec. Mgmt.	-	14	IKQIOB00 IKQIOB10
IKQIXE00	Index enter	Rec. Mgmt.	GY	10	IKQIXE00 IKQIXE20
IKQIXF00	Index format	Rec. Mgmt.	GU, GW	10	IKQIXF00
IKQIXS00	Index search	Rec. Mgmt.	HB	7-9, 13	IKQIXS00
IKQJRN	JRNAD exit	Rec. Mgmt.	GB, HU	8-12	IKQJRN
IKQJIBSM	Summarize JIBs for EDB chain	O/C/EOV	BA, IA	3, 14, 15	IKQJIBSM
IKQKRD	Key range determination routine (KRDR)	Rec. Mgmt.	GO	9	IKQKRD
IKQLAB	Look at label cylinder	O/C/EOV	BA, FB	3, 6	IKQLAB
IKQLASFT	Look-aside file table	O/C/EOV	BA	-	IKQLASFT
IKQLASMD	Look-aside module	O/C/EOV	BA, BD, EM EO, EP	3, 15	IKQLASMD
IKQLCD	Locate direct	Rec. Mgmt.	GF, GN	7-10	IKQLCD
IKQLCN	Locate next	Rec. Mgmt.	GL	7-9, 12	IKQLCN
IKQLCP	Locate previous	Rec. Mgmt.	GM, GR	8	IKQLCP
IKQMDY	Modify	Rec. Mgmt.	GI, GO	9, 12	IKQMDY
IKQNCA00	Get new control area	Rec. Mgmt.	GT, GX	10	IKQNCA00
IKQNEX	Get new extent	O/C/EOV	HA, HE	9	IKQNEX00
IKQOCMSG	Open/Close message routine	O/C/EOV	HF		IKQOCMSG
IKQOPN	Open function	O/C/EOV	BA	3	IKQOPN
IKQOPNAI	Alternate index initialization	O/C/EOV	BA	3	IKQOPNAI
IKQOPNCT	Open catalog	O/C/EOV	BA	3	IKQOPNCT
IKQOPNDO	Clean up after open failure	O/C/EOV	BA	3	IKQOPNDO
IKQOPNHC	Locate data set information in catalog	O/C/EOV	BA	3	IKQOPNHC
IKQOPNNC	Next cluster	O/C/EOV	BA, BC	3	IKQOPNNC
IKQOPNOV	Open volume extent	O/C/EOV	BA	3	IKQOPNOV
IKQOPNRD	Reset reusable data set	O/C/EOV	BA	3	IKQOPNRD
IKQOPNRP	Attach data set to resource pool	O/C/EOV	BB	3	IKQOPNRP
IKQOPNUC	User catalog	O/C/EOV	BA	3	IKQOPNUC
IKQOPNUS	Alternate index upgrade set determination	O/C/EOV	BA	3	IKQOPNUS
IKQPBF00	Purge buffer	Rec. Mgmt.	HR, HT	14	IKQPBF00
IKQPFO00	Format data CA or index CNV	Rec. Mgmt.	HA, HC, IA	9, 15	IKQPFO00
IKQPOP00	Build DSCBs	DADSM	FB, FE	6	IKQPOP00
IKQRBA	Update catalog for sharing	O/C/EOV	GU, HA, HD	10	IKQRBA00

* For further entry points, see Figure 4.6

Figure 4.4 Module directory (part 5 of 7)

Module name	Descriptive name	Component	Diag.#	Structure Figure 3.x	External entry points
IKQRCL00	Record management close	Rec. Mgmt.	HH, IA	15	IKQRCL00
IKQRDS00	Read DSCBs	DADSM	EG, EN, FB-FD, FF-FN	6	IKQRDS00
IKQREN00	Rename data set	DADSM	EK, FC	6	IKQREN00
IKQRQA	Request analyzer 1	Rec. Mgmt.	GB	7-13, 15	IKQRQA
IKQRQB	Request analyzer 2	Rec. Mgmt.	GB	7-13, 15	IKQRQB
IKQRQC	Request analyzer 3	Rec. Mgmt.	GB	7-13, 15	IKQRQC
IKQRRP	Relative record preformat	Rec. Mgmt.	GV, HH	10	IKQRRP IKQRRP20
IKQRTV	Retrieve	Rec. Mgmt.	GF	8	IKQRTV
IKQSCAT	SHOWCAT: Display catalog information	Catalog	ES	-	IKQSCAT
IKQSCN	Scan control interval	Rec. Mgmt.	GG, GN	7-9	IKQSCN
IKQSCR00	Scratch DSCBs	DADSM	EM-EP, FD, FE	6	IKQSCR00
IKQSFT	Shift	Rec. Mgmt.	GW, GZ	9, 10, 12, 15	IKQSFT
IKQSPM00	Manage space within extents	Rec. Mgmt.	HA	11	IKQSPM00
IKQSRG	Spanned record GET	Rec. Mgmt.	GF, GJ	8	IKQSRG
IKQSRU	Spanned record UPDATE	Rec. Mgmt.	GK	9	IKQSRU
IKQSRT	Insert	Rec. Mgmt.	GG	9	IKQSRT
IKQSTM	Storage manager	O/C/EOV	BA	3	IKQSTM
IKQTMSD	MODCB, SHOWCB, TESTCB: Modify, display, or test a control block	CB. Manip.	AD-AG	2	IKQTMSD
IKQTMSF	MODCB, SHOWCB, TESTCB: Modify, display, or test a control block	CB. Manip.	AD-AG	2	IKQTMSF
IKQUPD	Update	Rec. Mgmt.	GH, GI	9, 12	IKQUPD
IKQUPG	Alternate index upgrade routine	Rec. Mgmt.	GD	9	IKQUPG
IKQVDTPE	Device type routine	Catalog	ED, EE, EG, EO	3	IKQVDTPE
IKQVEDA	Enable and disable VSAM diagnostic aids	Service Aids			IKQVEDA
IKQVFY	Verify	Rec. Mgmt.	GS	13	IKQVFY
IKQVSMLK	Phase and include statements	VSAM	-	-	-
IKQVSM	VSAM request driver	Rec. Mgmt.	GB, IA	7-9, 12, 13, 15	IKQVSM

Figure 4.4 **Module directory (part 6 of 7)**

Module name	Descriptive name	Component	Diag.#	Structure Figure 3.x	External entry points
IKQVTC00	Open/Close VTOC	DADSM	EG, EN, FB-FE, FI	6	IKQVTC00
IKQWDS00	Write DSCBs	DADSM	EN, FB-FE, FG	6	IKQWDS00
$$BACLOS	Automatic close	O/C/EOV	IA		$$BACLOS
$$BCLCRA	Mark deleted CRA	Catalog	EN		$$BCLCRA
$$BCVSAM	Close interface	O/C/EOV	IA	15	$$BCVSAM
$$BCVS02	Common exit	O/C/EOV	BA, IA	3	$$BCVS02
$$BCVS03	LKMOD routine	Service aids		-	$$BCVS03
$$BCVS04	I/O routine for IKQVEDA	Service aids		-	$$BCVS04
$$BJIBFF	Delete JIB	DADSM	FI	6	$$BJIB00
$$BJIB00	Create JIB	DADSM	FI	6	$$BJIB00
$$BODADE	End of message interface	DADSM	-	3, 6, 14	$$BODADE
$$BODADS	Start of message interface	DADSM	FE	6	$$BODADS
$$BOVSAM	Open interface	O/C/EOV	BA	3, 14	$$BOVSAM
$$BOVS01	Catalog/DADSM interface to mount volume	O/C/EOV	BA, EG, EM-EP, FB-FE, HR	3, 6	$$BOVS01
$$BOVS03	Delete/build JIBs for VSAM	O/C/EOV	BA	3, 14, 15	$$BOVS03
$$BTCLOS	TCLOSE interface	O/C/EOV	IA	15	$$BTCLOS

Figure 4.4 **Module directory (part 7 of 7)**

Routine Directory

Some of the VSAM modules contain several routines which are listed alphabetically by the entry points along with the appropriate module. Figure 4.5 contains catalog management modules, Figure 4.6 record management modules.

Entry point	Module name	Procedure description
IGGPACDV	IGG0CLAB	Catalog management driver
IGGPADGO	IGG0CLAW	Add group occurrence
IGGPAIX	IGG0CLCA	Define AIX
IGGPALEC	IGG0CLBE	Check for index or data and sequence set with data
IGGPALSV	IGG0CLBN	Rename DSCBs for alter data set name function
IGGPALT	IGG0CLBD	ALTER processing
IGGPALT2	IGG0CLAX	ALTER catalog record's field value
IGGPALVL	IGG0CLBE	ALTER: Volume processing
IGGPALVR	IGG0CLBN	ALTER: Remove volume processing
IGGPALY	IGG0CLCD	Get and initialize work area for upgrading/no-upgrading routine
IGGPAOCI	IGG0CLAG	Assign contiguous control intervals
IGGPAXCI	IGG0CLAG	Assign one control interval
IGGPBMR	IGG0CLBR	Suballocate bit mask handler
IGGPBOUT	IGG0CLCA	Backout AIX or path association group occurrence
IGGPCADR	IGG0CLCR	Build CRA
IGGPCCCR	IGG0CLAG	Checkpoint the catalog control record (CCR)
IGGPCDVR	IGG0CLAT	Catalog management services common processing
IGGPCHAC	IGG0CLCG	Compute RBAs for next extent
IGGPCKAU	IGG0CLBM	Check the caller's authorization to access the catalog record
IGGPCKEN	IGG0CLCA	Check entry name and entry name of related object
IGGPCLBF	IGG0CLCB	Clear buffer
IGGPCNBO	IGG0CLB8	Remove candidate volume occurrences
IGGPCRAO	IGG0CLCO	Open catalog recovery area (CRA)
IGGPCRBO	IGG0CLCR	Back out volume record and reset TT-pointer
IGGPCRVL	IGG0CLCS	Suballocate CRA and write initial CRA records
IGGPDAIX	IGG0CLCX	Implicit delete AIX (for DEL cluster request)
IGGPDALR	IGG0CLBX	Average logical record size FPL structure processing
IGGPDBDI	IGG0CLAJ	DEFINE: Build the data set and index catalog records of a cluster
IGGPDBSF	IGG0CLAN	Buffer size FPL structure processing
IGGPDCAV	IGG0CLAL	Cross check and validity check
IGGPDCBO	IGG0CLAE	Define space backout
IGGPDCCE	IGG0CLAN	DEFINE: Build the cluster's catalog record
IGGPDCCO	IGG0CLCY	Determination of data and index characteristics
IGGPDCDA	IGG0CLAP	DEFINE Catalog processing (2 of 2)
IGGPDCDE	IGG0CLAL	Date and entry name processing
IGGPDCMB	IGG0CLAK	DEFINE: Completion (build the volume information group occurrence)
IGGPDCOC	IGG0CLAE	DEFINE Catalog: Catalog open, build, and close

Figure 4.5 **External entry points of catalog management modules (part 1 of 4)**

Entry point	Module name	Procedure description
IGGPDCRC	IGG0CLAS	Compute RBAs of data space
IGGPDDEP	IGG0CLAL	Date and entry name processing
IGGPDEF	IGG0CLAL	DEFINE common processing
IGGPDEFA	IGG0CLBH	DEFINE non-VSAM processing
IGGPDEFC	IGG0CLAS	DEFINE Catalog processing (1 of 2)
IGGPDEFS	IGG0CLAQ	DEFINE Space processing
IGGPDEIN	IGG0CLBW	Modify: Delete/insert processing
IGGPDEL	IGG0CLBG	DELETE Cluster/non-VSAM processing
IGGPDELC	IGG0CLAF	DELETE Catalog processing
IGGPDELO	IGG0CLCX	Check if cluster or AIX data set is open
IGGPDELP	IGG0CLCX	Delete path driver
IGGPDELS	IGG0CLBL	DELETE Space processing
IGGPDELX	IGG0CLCX	Delete AIX driver
IGGPDELY	IGG0CLCX	Delete and upgrade set (for DEL cluster request)
IGGPDEL2	IGG0CLAV	Delete a group occurrence
IGGPDEMV	IGG0CLA7	DELETE: Extract the volume information group occurrence
IGGPDEXA	IGG0CLBG	Build interface to extract cluster, AIX, or path associations
IGGPDEXD	IGG0CLAJ	Delete work area
IGGPDEXP	IGG0CLBG	Extract the password of a cluster or an AIX
IGGPDFBO	IGG0CLB8	DEFINE: Space recovery
IGGPDFRS	IGG0CLA8	Free unused and unneeded storage resources
IGGPDFS2	IGG0CLA6	DEFINE Space: Build the space header, space escriptor group, and data set directory entry group occurrences
IGGPDGO	IGG0CLAX	MODIFY: Delete group occurrence processing
IGGPDGOP	IGG0CLAX	MODIFY: Delete group occurrence pointer processing
IGGPDLDS	IGG0CLBG	Delete the space of the cluster or AIX
IGGPDLSF	IGG0CLCL	Forced delete space
IGGPDLXT	IGG0CLBG	Delete clean up routine (exit)
IGGPDMOP	IGG0CLCY	Complete entry construction process
IGGPDPBI	IGG0CLCY	Determine physical block size index value
IGGPDPTH	IGG0CLCX	Implicit delete path (for DEL cluster or DEL AIX request)
IGGPDRDA	IGG0CLAN	Regular define AMDSB processing
IGGPDRSP	IGG0CLBY	DEFINE Cluster processing (5th module)
IGGPDSCB	IGG0CLAN	DEFINE common processing (space calculations nd build the cluster catalog record)
IGGPDSPC	IGG0CLBX	DEFINE Cluster processing (4th module)
IGGPDSPF	IGG0CLBX	Space parameter FPL structure processing
IGGPDSTY	IGG0CLAL	Security FPL structure checking
IGGPDTIM	IGG0CLAL	DEFINE: Call the system timer
IGGPDUND	IGG0CLAN	DEFINE: Undo the previous processing
IGGPDUSC	IGG0CLA7	DELETE: Scratch the data space (format-1 - identifier - DSCB) from the volume's VTOC
IGGPDVMV	IGG0CLA7	DELETE: Mount and verify volumes
IGGPDVSC	IGG0CLCL	Clean VTOC from VSAM spaces
IGGPDWAI	IGG0CLAL	Work area initialization
IGGPEMIO	IGG0CLAF	I/O error message writer
IGGPEMSG	IGG0CLAF	Error message writer
IGGPEXPD	IGG0CLAX	Expand a catalog record's variable-length field
IGGPEXT	IGG0CLAZ	Extract processing
IGGPF4PR	IGG0CLA6	Read format-4 DSCB and either set or reset time stamps, CRA pointer and ownership

Figure 4.5 **External entry points of catalog management modules (part 2 of 4)**

Entry point	Module name	Procedure description
IGGPF4RD	IGG0CLBU	Read the format-4 DSCB
IGGPF4WR	IGG0CLBU	Write the format-4 DSCB
IGGPGET	IGG0CLAG	Get catalog record: Call record management to retrieve a catalog record
IGGPGLUB	IGG0CLCO	Get LUB index associated with CRA volume
IGGPGNEX	IGG0CLAW	Get an available RAB and format a new catalog extension record
IGGPGREC	IGG0CLBA	Retrieve a catalog record
IGGPGVAL	IGG0CLBA	Locate a catalog record field
IGGPIGOP	IGG0CLAW	Insert a group occurrence pointer
IGGPINIT	IGG0CLBC	Update-Extend: Initialization
IGGPIORA	IGG0CLCG	Error code analyzer
IGGPISCI	IGG0CLAG	Insure control-interval availability
IGGPLOC	IGG0CLAZ	Locate processing
IGGPLSTC	IGG0CLBQ	LISTCAT processing
IGGPMCO	IGG0CLAC	DEFINE Catalog: Master catalog build and open (1 of 2)
IGGPMCO2	IGG0CLAD	DEFINE Catalog: Master catalog build and open (2 of 2)
IGGPMEBM	IGG0CLAE	Handle multiple extents for catalog open and build
IGGPMGO	IGG0CLAX	Move group occurrence from one extension record into another
IGGPMOD	IGG0CLAV	Modify common processing
IGGPPAD	IGG0CLAG	PUT-Add: Call record management to write a new catalog record
IGGPPATH	IGG0CLCP	Define path
IGGPPDE	IGG0CLAG	ERASE: Call record management to erase a catalog record
IGGPPDEC	IGG0CLAG	Delete catalog record
IGGPPREC	IGG0CLAW	Call PUT-Add or PUT-Update to write a catalog record
IGGPPRPW	IGG0CLCA	Check password of related object
IGGPPTBF	IGG0CLCB	Put into buffer
IGGPPUPC	IGG0CLAG	PUT-Update: Call record management to rewrite a catalog record
IGGPRBAP	IGG0CLCG	Scan the ARDBs and AMDSBs for all RBAs
IGGPRCCR	IGG0CLAG	Read catalog control record
IGGPRELE	IGG0CLCB	Release function
IGGPRPLF	IGG0CLAB	Dequeue the catalog
IGGPRPLM	IGG0CLAB	Assign RPLs from the catalog RPL pool
IGGPSALL	IGG0CLAR	Suballocate: Candidate volume assignment
IGGPSALS	IGG0CLAU	Suballocate: Space assignment
IGGPSCA	IGG0CLAH	Set catalog ACB address/open user catalog
IGGPSCAT	IGG0CLAH	Search catalog processing
IGGPSCAX	IGG0CLCO	Scan CAXWA chain
IGGPSCNC	IGG0CLAY	Initial CTGPL processing
IGGPSDSP	IGG0CLAF	Remove space and close CRA

Figure 4.5 **External entry points of catalog management modules (part 3 of 4)**

Entry point	Module name	Procedure description
IGGPSGOP	IGG0CLAV	Retrieve the group occurrence pointer
IGGPSHNK	IGG0CLAX	Shrink a catalog record's variable-length field
IGGPSSCR	IGG0CLBF	Subscratch: Release a cluster's space within a VSAM data space
IGGPSVOL	IGG0CLBC	Search for the volume information group occurrence
IGGPTRPL	IGG0CLCG	Test RPL last used
IGGPTSTS	IGG0CLBA	CTGFL-for-tests processing
IGGPUPD	IGG0CLAV	Update catalog field
IGGPUPDE	IGG0CLBB	Update-Extend processing
IGGPUPG	IGG0CLCD	Add an AIX to the upgrade set (UPGRADE) or delete an AIX from the upgrade set (NOUPGRADE)
IGGPVMSC	IGG0CLA7	DELETE: Delete all space information in the volume catalog record
IGGPVMTV	IGG0CLA6	Volume mount
IGGPXDGO	IGG0CLBT	Add derived group occurrence
IGGPXEL2	IGG0CLBT	Delete derived group occurrence
IGGPXEXT	IGG0CLBS	Extract derived group occurrence
IGGPXIO	IGG0CLCG	I/O routine for catalog and CRA
IGGPXLT2	IGG0CLBT	Alter derived field value
IGGPXMOD	IGG0CLBT	Modify derived group occurrence
IGGPXVAL	IGG0CLBS	Get derived field value
IGG0CLC9	IGG0CLC9	Catalog management first load

Figure 4.5 **External entry points of catalog management modules (part 4 of 4)**

Entry point	Procedure description
IKQBFA10	Buffer manager, GETBUFF
IKQBFA20	Buffer manager, FREEBUFF
IKQBFA30	Buffer manager, get a BCB
IKQBFA40	Buffer manager, free a buffer
IKQBFA50	Buffer manager, do I/O
IKQBFA60	Buffer manager, steal a BCB
IKQBFA70	Buffer manager, check RBA for exclusive control
IKQBFA80	Buffer manager, return a BCB
IKQBFB10	Buffer manager, defer writing buffer
IKQBFB20	Buffer manager, write deferred buffers
IKQBFB30	Buffer manager, get scratch buffer from resource pool
IKQBFB40	Buffer manager, return scratch buffer to resource pool
IKQBFB50	Buffer manager, search resource pool for requested RBA
IKQCAS80	Control area split, count index entries
IKQCAS90	Control area split, decompress keys
IKQIOA00	I/O manager
IKQIOA20	I/O manager, RBA conversion
ILQIOA30	I/O manager, get and free blocks
IKQIOB00	I/O error analysis
IKQIOB10	I/O error analysis

Figure 4.6 **External entry points of record management modules**

Control Block Directory

The control block directory (Figure 4.7) contains a short entry for each of the most important VSAM control blocks, giving the length and the purpose of each block.

Data Area	Total size	Purpose
ACB	68 bytes	To describe a VSAM cluster
AMBL	104 bytes	To connect an ACB to the PLH and AMDSB(s)
AMCBS	28 bytes	To contain addresses of CAXWA chain, master and job catalog ACBs, and recovery information
AMDSB	200 bytes	To record data set status and statistics (not including buffer header and first EDB)
AMDTF	469 bytes	To contain save areas, lists, addresses for ISAM interface programs, and the error message build area
ARDB	48 + (2 x key length)	to record data about volumes and RBAs per key range
BCB	108 bytes	To point to a buffer
BHD	52 bytes	To contain information about buffers and buffer processing
BKPHD	64 bytes	To describe the storage allocation for the CCW build area
BSPH	72 bytes	To contain information about buffers in a subpool of the resource pool
CAXWA	138 bytes	To contain pointers to control blocks and work areas needed when a catalog is being processed
CCA	1308 bytes	To contain information about the catalog being processed and about the catalog record and its extensions
CIW	392 bytes + (5 x keylength)	To describe a control interval split workarea
CTGFL*(also known as FPL or FL)	24 bytes (variable)	To contain catalog field name, address, and length
CTGFV*(also known as FVT)	92 bytes	To contain addresses of user-supplied information fields and lists
CTGPL*(also known as CPL)	48 bytes (variable)	To contain a description of the call for catalog
* Rebuilt for each use, that is, not permanent		

Figure 4.7 **Control block directory (part 1 of 2)**

Data Area	Total size	Purpose
DASDM parameter list	170 bytes	To contain the input parameters for the DASD Space Management routines
DTFIS	294 bytes (variable)	To describe an ISAM file
EDB	40 bytes	To contain the extent descriptions
EXLST	30 bytes (variable)	To contain addresses for user exit routines
FCDB	64 bytes	Describes the channel program block.
LPMB	24 bytes	To describe the logical and physical nature of device
OAL	24 bytes (variable)	To contain all opened VSAM ACBs
OPNWA	2300 bytes	To contain information needed when a data set is being opened
PLH	ESDS 472 bytes KSDS } 540 + RRDS } keylength	To determine record or CNV position
RPHD	8 bytes	To contain information about the resource pool
RPL	52 bytes	To contain user request information and error feedback information
RSCB	16 bytes	To contain information needed for sharing resources
THB	100 bytes	To contain information needed to do track hold
USB	28 bytes (variable)	To maintain request information
VRPPL	20 bytes	To contain the input parameters needed for building the VSAM Resource Pool
VSRT	76 bytes	To contain information and pointers for the resource pool

Figure 4.7 **Control block directory (part 2 of 2)**

Section 5: Data Areas

This section deals with the internal data areas of VSAM, describing their formats, functions, and interrelationships. It is assumed that the reader is familiar with the basic structure of VSAM, such as the types of data sets, the structure of indexes, the concept of the catalog, etc., as these are described in the *DOS/VS Data Management Guide*, GC33-5372.

The section is divided into two parts:

- Descriptions of the VSAM data set, index, alternate index, and catalog.
- Description and format of the VSAM control blocks, together with figures showing their interrelationships.

VSAM Data Set

A VSAM data set is a collection of records grouped into control intervals. Control intervals are grouped into larger units called control areas. If the VSAM data set is key-sequenced, then the control interval(s) in which it resides are pointed to by entries in an associated index. The VSAM stored record, control interval, control area, and index are described in the topics that follow.

VSAM Record

Records are normally treated by VSAM as variable-length records. Records can be spanned across control intervals within a control area, and their maximum size is thus equal to the length of a control area. The only exception to this is a relative-record data set, whose records must have a fixed length.

Control Interval

A control interval is a continuous area of auxiliary storage that VSAM uses for storing records. The control interval is the unit of information that VSAM transfers between virtual and auxiliary storage.

The length of each control interval is an integral multiple of block size. The size of a control interval is determined by the system from the size of the records, user-specified minimum buffer size, device characteristics, and the user-specified percentage of free space. The user can specify the size of the control interval, but it must be within limits acceptable to VSAM.

Data records are put in the low-address portion of the control interval. Control information about each data record is put in the high-address portion of the control interval. The combination of a data record and its control information, though they are not physically adjacent, is called a stored record. The control information in a control interval consists of a Control Interval Definition Field and one or more Record Definition Fields. Figure 5.1 shows the format of a control interval.

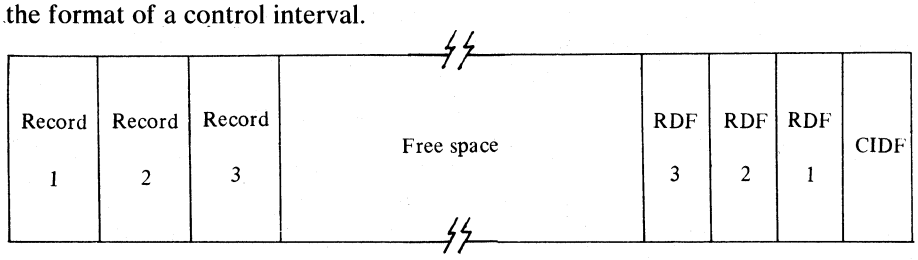


Figure 5.1 **Control interval format**

Control Interval Definition Field

The Control Interval Definition Field (CIDF) describes the control interval. Its format is shown in Figure 5.2.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
0	0	2	CIDFDD	Free space offset Displacement from the beginning of the control interval to the beginning of the free space[1]
2	2	2	CIDFLL	Free space length Length of the free space area within this control interval[1]

[1] If the CIDF contains only 0s, *end-of-data-set* or *end-of-key-range* is indicated; either the end of the data set was detected or the end of a key range in a key-sequenced data set was detected when the data set was to be divided between volumes. Information in the volume group occurrence (see VOLFLG) in the data set's catalog record helps to differentiate between the end-of-data-set and end-of-key-range conditions.

Figure 5.2 Control interval definition field format

Record Definition Field

The Record Definition Fields (RDFs) describe the records in the control interval. They are inserted into the control interval from right to left, which means that the rightmost RDF describes the leftmost data record.

There is normally one RDF for each record, except in two special cases. These are:

- When two or more consecutive records in the control interval have the same length. In this case, two RDFs are used to describe the whole group of records. The first (right-hand) RDF describes the characteristics of the records, and the second (left-hand) RDF contains a count of the number of records.

  Note that this is true only for key-sequenced and entry-sequenced data sets. The slots or records in a relative record data set have a fixed length, but specific information is required for each one. The records cannot, therefore, be grouped, and one RDF is required for each record.

- When the record is spanned. In this case, only one segment of one record can be located in the control interval. Nevertheless, two RDFs are used. The first (right-hand) RDF describes the record segment, and the second (left-hand) RDF contains a "level number", which is used for data integrity checking. This number is assigned and updated by VSAM whenever the spanned record is processed. The level number in all segments of a spanned record will always be the same, unless an error has occurred.

The format of an RDF is shown in Figure 5.3.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
0	0	1	RDFFLAG	Flag byte
			RDFEXT	RDF extension flag
		.0..		There is no RDF to the left of this RDF that contains additional information about record(s) described in this RDF.
		.1..		There is an RDF to the left of this RDF that contains additional information about the record(s) described in this RDF. Byte two and three of the RDF to the left contain the following information: • if there are more consecutive records than one of fixed length they contain the number of these records beginning with the record associated with the previous (to the right) RDF (see replication count flag) • in the case of spanned records they contain the level number
		..11	RDFSRM	The RDF to the left contains information about spanned records (middle segment)
		..10	RDFSRL	The same as above but last segment
		..01	RDFSRF	The same as above but first segment
			RDFREPL	Replication count flag
		.... 0...		The second and the third byte of this RDF contain the data records length
		.... 1...		This RDF contains additional information about the record(s) described in the RDF to the right.
			RDFRESL	Empty slot indicator (for relative record processing where one RDF is associated with one slot in the control interval - no extended RDFs)
		.... .1..		The record in the corresponding slot is invalid (it has been deleted or not yet inserted)
		.... .0..		The record in the corresponding slot is valid
				Depending on the kind of record(s) described, byte two and three of an RDF contain one of the following values:

Figure 5.3 Record definition field format (part 1 of 2)

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
1	1	2	RDFLL	Length field Byte two and three contain the data record's length. This type of RDF describes only a single record. It has no RDF to its left containing additional information. Byte 0, Bit 4 = 0, Bit 2, 3 = 0
1	1	2	RDFCOUNT	Count field Byte two and three contain the number of consecutive fixed-length records. It is a type of RDF that contains additional information about the records described in the RDF to the right. Byte 0, Bit 4 = 1, Bit 2, 3 = 0
1	1	2	RDFSRLVL	Level number Byte two and three contain the level number for spanned records. It is a type of RDF that contains additional information about the records described in the RDF to the right. Byte 0, Bit 4 = 1, Bit 2,3 = 11 or 10 or 01

Figure 5.3 Record definition field format (part 2 of 2)

Control Area

A control area consists of control intervals; the number of control intervals in a control area is determined by VSAM. The control area is the amount of space that VSAM preformats so that data integrity is ensured for records added to a data set.

Control areas are also used to simplify and localize the movement of records when records are inserted in a key-sequenced data set. If an insertion requires a free control interval and there isn't one, a control-area split results. VSAM establishes a new control area and moves the contents of approximately half of the full control area to free control intervals in the new control area. The new records, as their keys dictate, are then inserted into one of the two control areas. The control area has no specific control information.

Index

An index is created at the same time as a key-sequenced data set. The index structure exists in its own address space and consists of one or more levels. The lowest level or *sequence set* consists of one or more index records. There is an index record in the sequence set for each formatted control area. Within a sequence-set record there is either an index entry or a free data control interval pointer for each control interval in the control area. (Free data control interval pointers are discussed later in this section.) The key in each entry of a sequence set record is the same as the key of the last (highest) entry in the corresponding control interval. To save space, VSAM compresses the keys in the index.

The upper levels of the index are collectively called the index set, and contain index entries which point to the next lower level of the index. Figure 5.4 shows a simple index structure.

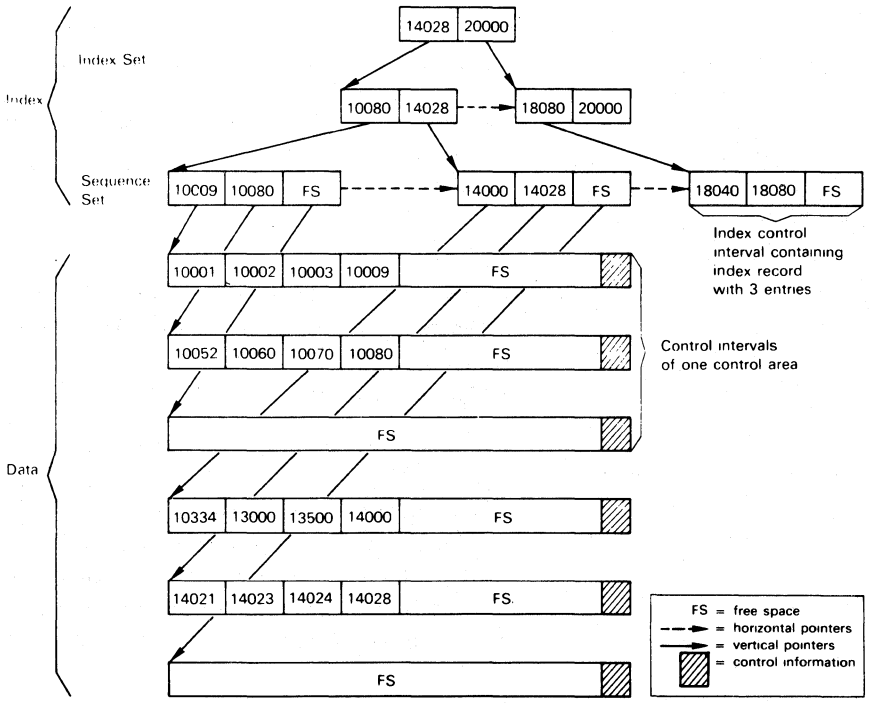


Figure 5.4 **Example of a simple VSAM index**

Index Record

The index records and control intervals are fully compatible with VSAM data records and control intervals, and are handled by record-management modules in the same way. The only differences between index records and data records are:

- There is only a single index record in an index control interval (and thus only one RDF).
- The internal format of an index record is fixed. This format is shown in the example of Figure 5.5, and its various parts are discussed below.

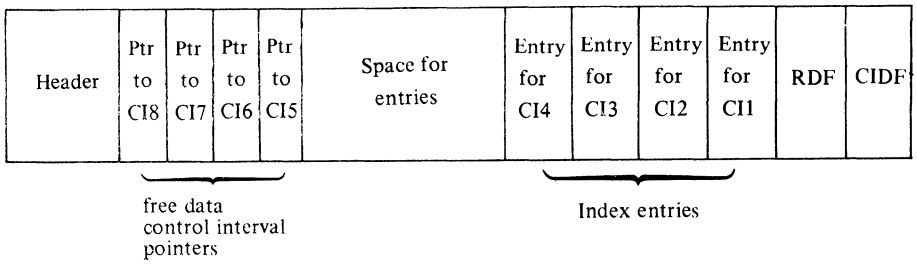


Figure 5.5 Example of an index control interval

Index Record Header

The index record header contains the information needed to insert index entries, to locate entries within the record, and to convert pointers into RBAs. The format of the index record header is shown in Figure 5.6.

Offset Dec	Offset Hex	Field Name	Field Size	Description
0	0	IXRL	2	Length in bytes, of the index record, including this field.
2	2	IXCINL	1	Length, in bytes, of the control information (the IXENTRYF, IXENTRYL, and IXENTRYP fields) in each index entry.
3	3	IXPMASK	1	Length of the pointers to free data control intervals in this index record[1]. This field is used as a mask for insert character (store character) under mask instructions that are used to access pointers. The value contained in this field specifies the length of these pointers, as follows: B'0001' 1-byte pointer B'0011' 2-byte pointer B'0111' 3-byte pointer

Figure 5.6 Index record header format (part 1 of 2)

Offset Dec	Offset Hex	Field Name	Field Size	Description
4	4	IXBASRBA	4	For a sequence-set index record, the RBA of a data control area that contains data to be referenced. This RBA and index-entry pointers are used together to calculate the 4-byte RBA of another index record or of a data control interval (0 for high-level indexes).
8	8	IXNXTIR	4	Pointer to the logically next index record in this index level. (Horizontal pointer)
12	C		4	Reserved (0).
16	10	IXLVLNO	1	Index level number. A sequence-set index is assigned a value of 1; the next higher-level index is assigned a value of 2; etc.
17	11		1	Reserved (0).
18	12	IXINSOS	2	Displacement from the beginning of this record to the space available for inserting index entries. For higher-level indexes, the entry space immediately follows the record header; for sequence-set indexes, the entry space follows the record header and free data-control-interval pointers.
20	14	IXLENTRY	2	Displacement from the beginning of this record to the last (high-key) entry in the index record.[2]
22	16	IXFSECTN	2	Displacement from the beginning of this record to the first (low key) section entry in the index record.[2]

[1] Pointers are allowed to vary in length to conserve index space. If, for example, the number of items to be referenced by an index record is less than 256, a one-byte pointer can be used; if the number is greater than 256 and less than 65,536, a two-byte pointer can be used; and if the number is greater than 65,536, a three-byte pointer can be used.

[2] This displacement is to the F (front-key compression count) byte of the entry, not to the beginning of the entry.

Figure 5.6 Index record header format (part 2 of 2)

Free Data-Control-Interval pointers

Free data-control-interval pointers, which exist only in sequence-set index records, are used to calculate the RBAs of available data control intervals. The length of a pointer is specified in the record header.

When the index is first built, and before records have been loaded into the data set, the index records of the sequence set contain one free data control interval pointer for each data control interval.

VSAM always uses the rightmost free data-control-interval pointer when a data control interval is needed. The value of the pointer is set to 0 when the control interval is used. As pointers are set to 0, the displacement to space that is available for index entries (contained in the record header) is adjusted by the length of the free data-control-interval pointer. In this way, space used by free data-control-interval pointers is made available for index entries when the pointers are no longer required.

The example in Figure 5.5 shows a sequence set record for a control area with eight control intervals. Of these eight, the first four are now occupied by data, and the last four are still free.

Index Entries

The index entries are the link between the index and the data set. They contain the key, the pointer to the data control interval containing the data record, and information about key compression. The format of an index entry is shown in Figure 5.7.

Field Size (in bytes)	Field Name	Description
Variable	IXKEY	Key characters that determine the sequence of records in a key-sequenced data set.
1	IXENTRYF (F byte)	Front-key compression count, that is, the number of characters by which the beginning of the key has been compressed.
1	IXENTRYL (L byte)	Length of the IXKEY field.
1-3	IXENTRYP (P field)	Pointer to an index or data control interval. This value is the number of the CI within the CA (for example '4' for the fifth CI). To calculate the RBA of the CI, this value must be multiplied by the CI size and added to the contents of IXBASRBA.

Figure 5.7 Index entry format

Index Entries for Spanned Records

Since spanned records extend across two or more data control intervals, their index entries, sometimes called "complex index entries", consist of a series of "normal" entries (one for each data control interval). These entries, in turn, are basically standard index entries, but they have some special features:

- The key is contained only in the entry for the last segment of the spanned records, whose F byte contains the actual key compression count.
- The entries for all other segments contain no key, and their F byte contains a compression count equal to the key length, thus indicating a key length (in the entry) of zero.
- Each entry contains a pointer to its associated segment (or data control interval).

Index Entry Sections

To save time when searching index records for a given key, the index entries are grouped into sections. This allows a rapid search, scanning only the highest key in each section, to locate the correct section, which is then searched for the correct key.

A section is defined by a two-byte field to the left of the high-key entry in the section. This field contains the displacement from the F byte of the high-key entry in this section to the F byte of the high-key entry in the next section (to the left). The index record header contains a pointer to the F byte in the high-key entry in the first section.

For technical reasons, this division of the index entries into sections is not carried out until a control interval split is necessary in a control area. There will thus be no section definition fields in the index of a freshly loaded data set, and only some of the sequence set records in an "older" data set will have such fields.

Alternate Index

The alternate index (AIX) provides an alternate means of access, using different keys, to the data records in the base cluster, which can be a key-sequenced or entry-sequenced data set, but not a relative-record data set. The alternate index itself is a key-sequenced data set. The index component of the AIX is identical in structure, format, and function to the index of any other key-sequenced data set. The basic structure of the data component of the AIX is also identical to that of a normal key-sequenced data set, as far as CIDFs and RDFs are concerned.

The only difference in format between the AIX and a normal key-sequenced data set concerns the records in the data component of the AIX, which have a fixed format, shown in Figure 5.8. These records form the logical connection between the AIX and the base cluster, and contain control information, the alternate key, and one or more pointers to the base cluster. If this base cluster is a key-sequenced data set, the pointers consist of the prime keys of the required data records, which are then located by means of the base cluster index. If the base cluster is an entry-sequenced data set, which has no index, the pointers are relative-byte addresses (RBAs) of the required records, which can then be located directly.

As it is possible to have more than one pointer in an AIX record, the length of such a record can vary. In extreme cases, it may be greater than the control interval length, and the record is treated as a spanned record.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
0	0	1	AIXFG	Flag byte
		xxxx xxx.		Reserved
		1	AIXPKP	Prime key pointers are used
		0		RBA pointers are used
1	1	1	AIXPL	Pointer length (in binary)
2	2	2	AIXPC	Number of pointers in this record (in binary)
4	4	1	AIXKL	Length of alternate key (in binary)
5	5	Note 1	AIXKY	Alternate key
Note 2		Note 3	AIXPT	First pointer to base cluster

Note 1: The length of this field is specified in AIXKL

Note 2: The displacement of this field is 5 + the length of AIXKY

Note 3: The length of this field is specified in AIXPL

Figure 5.8 Alternate index record format

Catalog

VSAM employs two types of catalogs - the master catalog and user catalogs. The internal structure and format of the two types is identical; the only difference is that the master catalog contains an entry for each user catalog.

Purpose

The VSAM catalog is built and processed by catalog-management modules. Catalog-management modules, via the catalog, enable a user to locate a data set, volume, index, or cluster by specifying a name or volume serial number. In addition, the VSAM catalog provides VSAM with the information required to allocate space for data sets, verify authorization to gain access to them, compile usage statistics on them, and relate RBAs to physical locations within data sets. The catalog indicates, therefore, much more than the simple location of data sets. The catalog maintains the relationship between a key-sequenced data set and its index, or between any data set and its alternate index(es), describes the location of VSAM data spaces and the data sets that reside in them, and describes the space that is available for new data sets.

Structure

The VSAM catalog is conceptually a key-sequenced VSAM data set divided into two key ranges. VSAM data set processing options, such as record replication and sequence set with data, are utilized in both key ranges of the catalog. The catalog record size is 505 bytes in the low key range and 47 bytes in the high key range; the catalog control interval size is 512 bytes. Figure 5.9 shows the VSAM catalog. The figure shows:

- The low key range of the catalog, shown on the left, contains records that describe objects, that is, data sets, indexes, volumes, and clusters.
- The high key range of the catalog, shown on the right, contains the true name (a data set name or volume serial number) of an object specified by the user.
- The index, shown in the middle, points to both the low and high key ranges of the catalog.

With the exception of catalog records that are built when the catalog is created and describe the catalog itself, catalog records are built as objects are cataloged. The order of the records depends upon which portion of the catalog the records belong to. If the catalog records reside in the low key range of the catalog, they are ordered according to control-interval number. As objects are cataloged, available control intervals are used. If the catalog records reside in the high key range of the catalog, they are ordered according to their true name (data-set name or volume serial number).

Catalog management relies on record management for all record retrieval and storage. When a user specifies a data-set name, record management uses the index to retrieve a catalog record that contains the data-set name (in the high key range of the catalog); that record, in turn, contains the control-interval number of the catalog record that describes the data set. Catalog management converts the control-interval number to an RBA in the low key range of the catalog.

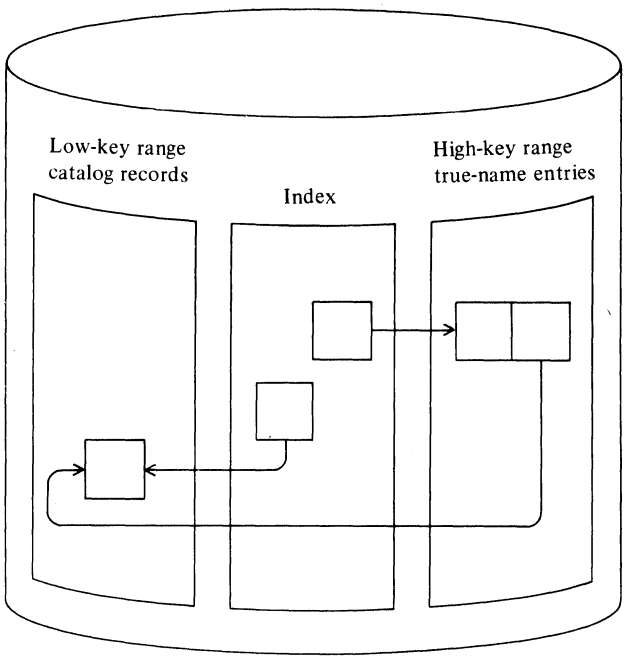


Figure 5.9 **Parts of the VSAM catalog**

Catalog Records which Describe the Catalog

Catalog records that describe the catalog as a data set are in fixed positions at the beginning of the catalog. Figure 5.10 shows the control-interval numbers of records that describe the catalog, the kind of catalog record each is, and the contents of each. The various types of catalog record are described later in this section.

Note that the self-describing records of the catalog do not contain CRA information (bytes 5-17). They do however use the release indicator (byte 4).

When the catalog is built, there are two True Name records. One contains the catalog's volume serial number and points to control-interval 9. The other contains the catalog's name and points to control-interval 2.

Types of Catalog Records

There are various types of catalog records. They are shown below, grouped according to the key range of the catalog in which they are located.

Control Interval Number	Record Type	Contents
0	Data	Description of the data portion of the catalog (low and high key ranges)
1	Index	Description of the index portion of the catalog
2	Cluster	Description of the catalog as a key-sequenced VSAM cluster. This catalog record contains the catalog's password information group occurrence.
3	Control	Catalog control record (CCR), which describes the catalog's free control intervals within the low key range.
4	Extension	Extension of the catalog-index record (control interval #1). This Extension record contains a description of the high-level index extents of the catalog.
5	Extension	Extension of the catalog-data record (control interval #0). This Extension record contains a description of the low key range data extents of the catalog.
6	Extension	Extension of the catalog-index record (control interval #1). This Extension record contains a description of the index sequence-set extents for the low key range of the catalog.
7	Extension	Extension of the catalog-data record (control interval #0). This Extension record contains a description of the extents of the True Name records in the high key range of the catalog.
8	Extension	Extension of the catalog index record (control interval #1). This Extension record contains a description of the index sequence-set extents for the high key range of the catalog.
9	Volume	Description of the track allocation and VSAM data spaces on this volume.
10,11...	Volume Extension	As many volume extension records as are necessary to describe the total space on the volume.

Figure 5.10 **Catalog records that describe the catalog**

High Key Range of the Catalog

The high key range of the catalog contains 47-byte True Name records. The True Name records associate user-specified name or volume serial numbers with the control-interval number of the catalog record that describes the specified object.

Low Key Range of the Catalog

Each catalog record in this part of the catalog occupies a full control interval and each contains the number of the control interval in which it resides. Each catalog record also contains the record type of the record. The low key range of the catalog is made up of the following types of records:

A: Non-VSAM record, which describes a data set organized differently from VSAM. There is one non-VSAM record for each non-VSAM data set cataloged. Sometimes called "Alien" record.

C: Cluster record, which describes a VSAM data-set cluster. This record contains the control-interval number of a Data record and, if the VSAM data set is a key-sequenced data set, the control-interval number of an Index record. There is one Cluster record for each VSAM cluster cataloged.

D: Data record, which describes the data component of a catalog, cluster, or AIX. There is one data record for each data set cataloged.

E: Extension record, which contains overflow information from another catalog record (except type 'V'). There are as many Extension records as are required to contain overflow information.

F: Free record, which marks the control interval in which it resides as available for use as another kind of catalog record. There is one Free record for each previously assigned control interval that is available for use.

G: Alternate Index record, which describes an alternate index. There is one such record for each alternate index cataloged.

I: Index record, which describes the index component of a catalog, cluster, or AIX. There is one index record for each index cataloged.

L: Control record, which describes the free control intervals in the low key range of the catalog. The Control record is the fourth record in the catalog.

R: Path record, which describes a VSAM path. There is one such record for each path cataloged.

U: User Catalog record, which describes a user catalog. One user catalog record is present in the master catalog for each user catalog which is cataloged.

V: Volume record, which describes each VSAM data space on a volume, the data sets that reside in the data space, and the space available within the data space. There is one Volume record for each volume controlled by this catalog.

W: Volume Extension record, which is used to extend volume records as required.

Y: Upgrade set record, which describes an upgrade set. There is one such record for each upgrade set cataloged.

Catalog Recovery Area

For a catalog defined with the recovery attribute, a Catalog Recovery Area (CRA) is reserved on each volume owned by the catalog. The CRA on each volume is conceptually an entry-sequenced data set which contains a self-describing part similar to that of the catalog and copies of catalog records describing the data sets on the volume. These copies are immediately updated whenever the original records in the catalog are changed.

Self-Describing Part of the CRA

This part of the CRA contains those records which are necessary to describe the CRA. They are basically the same records as those in the self-describing part of a catalog, except that the records describing an index are not needed, because the CRA is an entry-sequenced data set. The unused control intervals contain free records. Figure 5.11 shows the format of the self-describing part of the CRA.

Control Interval number	Record type	Contents
0	Data	Description of the data portion of the CRA The field CRAVOL in this record contains the volume serial number of the catalog which owns the volume.
1	Free record	
2	Cluster	Description of the CRA as an entry-sequenced VSAM cluster. This record contains the name of the catalog which owns the volume.
3	Control	Catalog control record which describes the free control intervals in the CRA
4	Free record	
5	Extension	Extension of the CRA data record (CI#0)
6	Free record	
7	Free record	
8	Free record	
9	Volume	Description of the track allocation for the CRA
10	Volume extension	
11	Volume extension	Extensions of the volume record (CI#9)
12	Volume extension	

Figure 5.11 Self-describing part of the CRA

Copies of catalog records

All catalog records which describe data sets or volumes are duplicated on specific volumes, as shown below:

A volume record is duplicated in the CRA of the volume which it describes.

All records concerning a key-sequenced data set or its alternate index (Cluster, Alternate Index, Data, Index, Path, and Upgrade Set records) are duplicated in the CRA of the first volume on which space was allocated for the index of the base cluster.

All records concerning an entry-sequenced data set or its alternate index, or a relative record data set, (Cluster, Alternate Index, Data and Index for the AIX, Path, and Upgrade Set records) are duplicated on the first volume on which space was allocated for the data component of the base cluster.

If a volume is imported from an OS/VS VSAM system, the CRA may contain other records, such as records which describe a non-VSAM data set. These records are not, however, used by DOS/VS VSAM.

True-name Catalog Record

The True Name record associates the volume serial number, data-set name, or cluster name specified by the user with the control-interval number of the catalog record that describes the object (volume, cluster, non-VSAM [alien] data set, data component, index component, path, or alternate index). True Name records are contained in the high key range part of the catalog and are pointed to by the catalog's index records. The True Name record is retrieved using key-sequenced processing. The catalog-management modules convert the control-interval number in the True Name record to an RBA which can be used to retrieve the associated record in the low key range.

True Name records are 47 bytes long; several might be contained in a catalog's control interval (512 bytes). The format of that record is shown in Figure 5.12.

Offset Hex	Offset Dec	Bytes	Description
0	0	44	Name of a data set or cluster, filled on the right with blanks, or a volume serial number, filled on the right with zeros, specified by the user.
44	2C	3	Control-interval number of the catalog record that describes the object.

Figure 5.12 True-name catalog record format

Non-VSAM Catalog Record

The non-VSAM catalog record describes a non-VSAM data set. Figure 5.13 shows the format of a non-VSAM catalog record.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
0	0	44		Key and ID area
0	0	1		Binary zeros
1	1	3	ENTIDNO	Control interval number of entry
4	4	1	RELIND	Release indicator; X'00' This record was created with a DOS/VS Release prior to Release 31 X'01' This record was created with DOS/VS Release 31 or later

Figure 5.13 Non-VSAM catalog record format (part 1 of 3)

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
5	5	6	CRAVOL	CRA volume serial number
11	B	3	CRAIDNO	CRA control interval number
14	E	4	CRADEVT	CRA device type
18	12	26		Binary zeros
44	2C	1	ENTYPE	Record type - 'A'
45	2D	2		Record length
47	2F	1		Number of variable-length fields that precede the pointer to an Extension record. Always zero.
48	30	1		Length of the fixed-length fields in this record, excluding any fixed-length fields that follow displacement 92 (5C). This value is always equal to the displacement from the beginning of the record to the pointer to an Extension record.
49	31	44	ENTNAME	Entry name (data set name)
The following six-byte entry contains control information for the group occurrence pointers that follow it. [1]				
93	5D	5		Pointer to Extension record. If this record is not continued in an Extension record, this field contains zeros.
98	62	1		The number of group occurrence pointers that follow.
			Group occurrence pointer (repetitive)	
Bit 0 and 1 of Byte 3 identify the group occurrence further:				
		00xx xxxx		Pointer to a group occurrence within the record.
			Byte	**Meaning**
			0	Reserved
			1-2	Displacement of the group occurrence from the beginning of all group occurrences in this record
			3	Bits 0 and 1 are set to zero. Bits 2 through 7 contain a code describing the group occurrence pointed to.
			4	Sequence number of the group occurrence pointed to.

Figure 5.13 **Non-VSAM catalog record format (part 2 of 3)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
		10xx xxxx		Pointer to a group occurrence contained in an Extension record. **Byte** **Meaning** 0-2 Control-interval number of the Extension record that contains this group occurrence. 3 Bits 0 and 1 are set to 1 and 0, respectively. Bits 2 through 7 contain a code describing the group occurrence pointed to. 4 Sequence number of the group occurrence pointed to.
		01xx xxxx		Pointer to a group occurrence which has been deleted. The code in bits 2 through 7 of byte 3 and the sequence number in byte 4 have been retained however.

[1] For further information see section *Group occurrences in catalog records*.

Figure 5.13 Non-VSAM catalog record format (part 3 of 3)

Cluster Catalog Record

The Cluster record describes a data set and its index. Figure 5.14 shows the format of a cluster catalog record.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
0	0	44		Key and ID area
0	0	1		Binary zeros
1	1	3	ENTIDNO	Control interval number of entry
4	4	1	RELIND	Release indicator; X'00' This record was created with a DOS/VS Release prior to Release 31 X'01' This record was created with DOS/VS Release 31 or later
5	5	6	CRAVOL	CRA volume serial number
11	B	3	CRAIDNO	CRA CI number
14	E	4	CRADEVT	CRA device type
18	12	26		Binary zeros
44	2C	1	ENTYPE	Record type - 'C' for a cluster record
45	2D	2		Record length
47	2F	1		Number of variable-length fields that precede the pointer to an Extension record. Always zero.

Figure 5.14 Cluster catalog record format (part 1 of 3)

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
48	30	1		Length of the fixed-length fields in this record, excluding any fixed-length fields that follow displacement 107 (6B). This value is always equal to the displacement from the beginning of the record to the pointer to an Extension record.
49	31	44	ENTNAME	Entry name (Normally true name of the cluster. If this record is in the self-describing part of the CRA, it contains the name of the catalog which owns this volume)
93	5D	8	OWNERID	Owner of data set
101	65	3	DSETCRDT	Data set creation data (ydd)
104	69	3	DSETEXDT	Data set expiration date (ydd)
107	6B	1	CATTR	Reserved (for OS)
The following six-byte entry contains control information for the group occurrence pointers that follow it. [1]				
108	6C	5		Pointer to Extension record. If this record is not continued on an Extension record, this field contains zeros.
113	71	1		The number of group occurrence pointers that follow.
			Group occurrence pointer (repetitive)	
Bit 0 and 1 of Byte 3 identify the group occurrence further:				
		00xx xxxx		Pointer to a group occurrence within the record.
				Byte **Meaning** 0 Reserved 1-2 Displacement of the group occurrence from the beginning of all group occurrences in this record. 3 Bits 0 and 1 are set to zero. Bits 2 through 7 contain a code describing the group occurrence pointed to. 4 Sequence number of the group occurrence pointed to by code.
		10xx xxxx		Pointer to a group occurrence contained in an Extension record.
				Byte **Meaning** 0-2 Control-interval number of the Extension record that contains this group occurrence. 3 Bits 0 and 1 are set to 1 and 0, respectively. Bits 2 through 7 contain a code that describes the group occurrence pointed to. 4 Sequence number of the group occurrence pointed to by code.

Figure 5.14 Cluster catalog record format (part 2 of 3)

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
		01xx xxxx		Pointer to a group occurrence which has been deleted. The code in bits 2 through 7 of byte 3 and the sequence number in byte 4 have been retained however.

[1] For further information see section *Group occurrences in catalog records*.

Figure 5.14 Cluster catalog record format (part 3 of 3)

Data and Index Catalog Record

Data and Index records describe data sets and their indexes. Figure 5.15 shows the format of the data and index records.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
0	0	44		Key and ID area
0	0	1		Binary zeros
1	1	3	ENTIDNO	Control interval number of entry
4	4	1	RELIND	Release indicator; X'00' This record was created with a DOS/VS Release prior to Release 31 X'01' This record was created with DOS/VS Release 31 or later
5	5	6	CRAVOL	CRA volume serial number
11	B	3	CRAIDNO	CRA control interval number
14	E	4	CRADEVT	CRA device type
18	12	26		Binary zeros
44	2C	1	ENTYPE	Record type - 'D' for a data record or 'I' for an index record
45	2D	2		Record length
47	2F	1		Number of variable-length fields that precede the pointer to an Extension record. Always zero.
48	30	1		Length of the fixed-length fields in this record, excluding any fixed-length fields following displacement 142 (8E). This value is always equal to the displacement from the beginning of the record to the pointer to an Extension record.
49	31	44	ENTNAME	For a Data or Index record, the data-set name
93	5D	8	OWNERID	Owner of the data set, specified when the data set was defined.

Figure 5.15 Data and index catalog record format (part 1 of 4)

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
101	65	3	DSETCRDT	Data-set creation data, in packed-decimal form YDD, specified when the data set was defined.
104	68	3	DSETEXDT	Data-set expiration date, in packed-decimal form YDD, specified when the data set was defined.
107	6B	1	ATTR1	Data-set attributes, which are defined in Access Method Services commands, as follows:
		1...		Speed – recovery features will be minimized or omitted in order to optimize operating speed during initial loading
		.1..		Unique component. Entire space occupied by a unique component.
		..1.		Reusable data set
		...1		Erase the component upon deletion
		.... 1...		Recoverable catalog
		.... .1..		Inhibit update
		.... ..1.		This component has been temporarily exported
		x		Reserved for MVM
108	6C	1	ATTR2	Data-set sharing attributes as follows: Cross-partition sharing:
		00..		The data set can be shared by READ users *or* it can be used by one UPDATE/OUTPUT user.
		01..		The data set can be shared by READ users *and* one UPDATE/OUTPUT user.
		10..		The data set can be fully shared
		11..		The data set can be fully shared; with assistance supplied by VSAM.
				Cross-system sharing (set by DOS but used only by OS):
		..00		The data set can be shared by READ users *or* it can be used by one UPDATE/OUTPUT user.
		..01		The data set can be shared by READ user *and* one UPDATE/OUTPUT user.
		..10		The data set can be fully shared
		..11		The data set can be fully shared; with assistance supplied by VSAM.
		1		DS not usable
		.... xxx.		Reserved
109	6D	1	OPENIND	Open indicator flag; if this byte contains X'80', the data set is open for output.
110	6E	4	BUFSIZE	Minimum buffer size

Figure 5.15 **Data and index catalog record format (part 2 of 4)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
114	72	3	PRIMSPAC	Primary space allocated for the data set or index, specified when the data set or index was defined.
117	75	3	SCONSPAC	Secondary space allocation for the data set or index, specified when the data set or index was defined.
120	78	1	SPACOPTN	Space options flags.
		10..		Track request, which indicates that space was allocated in terms of tracks.
		11..		Cylinder request, which indicates that space was allocated in terms of cylinders.
		..xx xxxx		Reserved
121	79	4	HURBADS	High used RBA of the data set or index.
125	7D	4	HARBADS	High allocated RBA of the data set or index.
129	81	4	LRECL	For a Data record, the logical record size of the data set described by this Data record. For an Index record, always X'FF's.
133	85	2	USERINFO	User information for the DOS/VS ISAM interface program (IIP).
135	87	8	EXCPEXIT	Exception exit
The following six-byte entry contains control information for the group occurrence pointers that follow it. [1]				
143	8F	5		Pointer to Extension record. If this record is not continued in an Extension record, this field contains zeros.
148	94	1		The number of group occurrence pointers that follow.[1]
			Group occurrence pointer (repetitive)	
Bit 0 and 1 of Byte 3 identify the group occurrence further:				
		00xx xxxx		Pointer to a group occurrence within the record
			Byte	**Meaning**
			0	Reserved
			1-2	Displacement of the group occurrence from the beginning of all group occurrences in this record.
			3	Bits 0 and 1 are set to zero. Bits 2 through 7 contain a code that describes the group occurrence pointed to.
			4	Sequence number of the group occurrence pointed to.

Figure 5.15 **Data and index catalog record format (part 3 of 4)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
		10xx xxxx		Pointer to a group occurrence contained in an Extension record. **Byte** **Meaning** 0-2 Control-interval number of the Extension record that contains this group occurrence. 3 Bits 0 and 1 are set to 1 and 0, respectively. Bits 2 through 7 contain a code that describes the group occurrence pointed to. 4 Sequence number of the group occurrence pointed to by code.
		01xx xxxx		Pointer to a group occurrence which has been deleted. The code in bits 2 through 7 of byte 3 and the sequence number in byte 4 have been retained however.

[1] For further information see section *Group occurrences in catalog records*.

Figure 5.15 Data and index catalog record format (part 4 of 4)

Extension Catalog Record

The Extension record contains overflow information from another catalog record. Figure 5.16 shows the format of an extension catalog record.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
0	0	44		Key and ID area
0	0	1		Binary zeros
1	1	3	ENTIDNO	Control interval number of entry
4	4	1	RELIND	Release indicator; X'00' This record was created with a DOS/VS Release prior to Release 31 X'01' This record was created with DOS/VS Release 31 or later
5	5	6	CRAVOL	CRA volume serial number
11	B	3	CRAIDNO	CRA control interval number
14	E	4	CRADEVT	CRA device type
18	12	26		Binary zeros
44	2C	1	ENTYPE	Record type - 'E', for an extension of any other record except a volume catalog record where record type is 'W'.
45	2D	2		Record length

Figure 5.16 Extension catalog record format (part 1 of 3)

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
47	2F	1		Number of variable-length fields that precede the pointer to an Extension record. Always zero.
48	30	1		Length of the fixed-length fields in the header fields, excluding any fixed length fields following displacement 48 (30). This value is always equal to the displacement from the beginning of the record to the extension record's pointer.
The following six-byte entry contains control information for the group occurrence pointers that follow it. [1]				
49	31	5		Pointer to Extension record. If this record is not continued on an Extension record, this field contains zeros.
54	36	1		The number of group occurrence pointers that follow.
			Group occurrence pointer (repetitive)	
Bit 0 and 1 of Byte 3 identify the group occurrence further:				
		00xx xxxx		Pointer to a group occurrence within the record.
			Byte	**Meaning**
			0	Reserved
			1-2	Displacement of the group occurrence from the beginning of all group occurrences in this record.
			3	Bits 0 and 1 are set to zero. Bits 2 through 7 contain a code describing the group occurrence pointed to.
			4 (4-5)[2]	Sequence number of the group occurrence pointed to be code.

Figure 5.16 **Extension catalog record format (part 2 of 3)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
		10xx xxxx		Pointer to a group occurrence contained in an Extension record. **Byte** **Meaning** 0-2 Control-interval number of the Extension record that contains this group occurrence. 3 Bits 0 and 1 are set to 1 and 0, respectively. Bits 2 through 7 contain a code that describes the group occurrence pointed to. 4 (4-5)[2] Sequence number of the group occurrence pointed to be code.
		01xx xxxx		Pointer to a group occurrence which has been deleted. The code in bits 2 through 7 of byte 3 and the sequence number in bytes 4 and 5 of "W" type have been retained however.

1 For further information see section *Group occurrences in catalog records*.

2 For 'W' type records, this sequence number occupies bytes 4 and 5, and the group occurrence pointer is 6 bytes long.

Figure 5.16 Extension catalog record format (part 3 of 3)

Free Catalog Record

The Free record indicates that the control interval in which it resides is free and points to the next control interval that is free because of deletion. Note that the free space in the catalog that has never been assigned is not represented by a Free record; a Free record is used only to mark a record that was used and deleted. Figure 5.17 shows the format of the free record.

Offset Dec	Offset Hex	Bytes	Description
0	0	44	Key. **Byte** **Meaning** 0 Zeros. 1-3 Control-interval number of this record 4-43 Zeros
44	2C	1	Record type - 'F'.
45	2D	3	Control-interval number of the next free control interval

Figure 5.17 Free catalog record format

Alternate Index Catalog Record

The alternate index catalog record describes the alternate index as a key-sequenced data set and relates it to the base cluster it belongs to. Figure 5.18 shows the format if the alternate index catalog record.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
0	0	44		Key and ID area
0	0	1		Binary zeros
1	1	3	ENTIDNO	Control interval number of entry
4	4	1	RELIND	Release indicator; X'00' This record was created with a DOS/VS Release prior to Release 31 X'01' This record was created with DOS/VS Release 31 or later
5	5	6	CRAVOL	CRA volume serial number
11	B	3	CRAIDNO	CRA control interval number
14	E	4	CRADEVT	CRA device type
18	12	26		Binary zeros
44	2C	1	ENTYPE	Record type - 'G'
45	2D	2		Record length
47	2F	1		Number of variable-length fields that precede the pointer to an Extension record. Always zero.
48	30	1		Length of the fixed-length fields in this record, excluding any fixed-length fields that follow displacement 107 (6B). This value is always equal to the displacement from the beginning of the record to the pointer to an Extension record.
49	31	44	ENTNAME	Entry name (data set name)
93	5D	8	OWNERID	Owner of data set
101	65	3	DSETCRDT	Data set creation date (ydd)
104	69	3	DSETEXDT	Data set expiration date (ydd)
107	6B	1	RGATTR	AIX attribute, X'80' = upgrade (AIX)
The following six-byte entry contains control information for the group occurrence pointers that follow it. [1]				
108	6C	5		Pointer to Extension record. If this record is not continued on an Extension record, this field contains zeros.
		1		The number of group occurrence pointers that follow.

Figure 5.18 **Alternate index catalog record format (part 1 of 2)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
				Group occurrence pointer (repetitive)
				Bit 0 and 1 of Byte 3 identify the group occurrence further:
		00xx xxxx		Pointer to a group occurrence within the record. **Byte** **Meaning** 0 Reserved 1-2 Displacement of the group occurrence from the beginning of all group occurrences in this record. 3 Bits 0 and 1 are set to zero. Bits 2 through 7 contain a code that describes the group occurrence pointed to. 4 Sequence number of the group occurrence pointed to.
		10xx xxxx		Pointer to a group occurrence contained in an Extension record. **Byte** **Meaning** 0-2 Control-interval number of the Extension record that contains this group occurrence. 3 Bits 0 and 1 are set to 1 and 0, respectively. Bits 2 through 7 contain a code that describes the group occurrence pointed to. 4 Sequence number of the group occurrence pointed to by code.
		01xx xxxx		Pointer to a group occurrence which has been deleted. The code in bits 2 through 7 of byte 3 and the sequence number in byte 4 have been retained however.
[1] For further information see section *Group occurrences in catalog records*.				

Figure 5.18 Alternate index catalog record format (part 2 of 2)

Index Catalog Record

For index catalog record format see data catalog record format. The record type (byte 44) is 'I'.

Catalog Control Record (CCR)

The catalog control record (CCR) is used by catalog management to control the allocation of control intervals in the low key range of the catalog, where catalog records, excluding the True Name records and the index, reside. The CCR also shows the catalog's high-used and high-allocated RBA values. The catalog control record is the fourth record (control interval) in the catalog.

For a request of one catalog record, catalog management tries to use a record that was freed because of deletion. This process is done before using unassigned space. If more than one catalog record is needed, catalog management tries to use contiguous unassigned space in the current extent; if sufficient unassigned space is not available, records that have been deleted are used. Figure 5.19 shows the CCR format.

Offset Dec	Offset Hex	Bytes	Description
0	0	44	Key **Byte** **Meaning** 0 Zeros. 1-3 Control-interval number of this record. 4-43 Zeros.
44	2C	1	Record type - 'L'.
45	2D	3	Number of the highest control interval within the current extent.[1]
48	30	3	Number of the next free control interval of those that have not been previously assigned.[1]
51	33	3	Count of deleted control intervals, that is, the count of control intervals that are free because of deletion.[2]
54	36	3	First deleted control interval in a chain of control intervals that are free because of deletion.[2]
The following fields are used to keep track of the RBA values that denote the current logical end of parts of the catalog:			
57	39	4	Data, low key range: high-key RBA
61	3D	4	Data, low key range: high-used RBA
65	41	4	Data, low key range: high-allocated RBA
69	45	4	Data, high key range: high-key RBA
73	49	4	Data, high key range: high-used RBA
77	4D	4	Data, high key range: high-allocated RBA
81	51	4	Index, high level: high-used RBA
85	55	4	Index, high level: high-allocated RBA
89	59	4	Index, low key range - sequence set: high-used RBA
93	5D	4	Index, low key range - sequence set: high-allocated RBA
97	61	4	Index, high key range - sequence set: high-used RBA
101	65	4	Index, high key range - sequence set: high-allocated RBA
1	This field is used to keep track of unassigned space within the current extent.		
2	This field is used to keep track of previously-used records that are now available for use as another catalog record.		

Figure 5.19 Catalog control record format

Path Catalog Record

The path catalog record describes a VSAM-path which is a logical connection between a base cluster and an alternate index. Figure 5.20 shows the format of the path catalog record.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
0	0	44		Key and ID area
0	0	1		Binary zeros
1	1	3	ENTIDNO	Control interval number of entry
4	4	1	RELIND	Release indicator; X'00' This record was created with a DOS/VS Release prior to Release 31 X'01' This record was created with DOS/VS Release 31 or later
5	5	6	CRAVOL	CRA volume serial number
11	B	3	CRAIDNO	CRA control interval number
14	E	4	CRADEVT	CRA device type
18	12	26		Binary zeros
44	2C	1	ENTYPE	Record type - 'R'
45	2D	2		Record length
47	2F	1		Number of variable-length fields that precede the pointer to an Extension record. Always zero.
48	30	1		Length of the fixed-length fields in this record, excluding any fixed-length fields following displacement 107 (6B). This value is always equal to the displacement from the beginning of the record to the pointer to an Extension record.
49	31	44	ENTNAME	Entry name (data set name)
93	5D	8	OWNERID	Owner of data set
101	65	3	DSETCRDT	Data set creation date (ydd)
104	69	3	DSETEXDT	Data set expiration date (ydd)
107	6B	1	RGATTR	Path attribute, X'80' = update (path)
The following six-byte entry contains control information for the group occurrence pointers that follow it. [1]				
108	6C	5		Pointer to Extension record. If this record is not continued on an Extension record, this field contains zeros.
		1		The number of group occurrence pointers that follow.

Figure 5.20 Path catalog record format (part 1 of 2)

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
				Group occurrence pointer (repetitive)
				Bit 0 and 1 of Byte 3 identify the group occurrence further:
		00xx xxxx		Pointer to a group occurrence within the record.
				Byte **Meaning**
				0 Reserved
				1-2 Displacement of the group occurrence from the beginning of all group occurrences in this record.
				3 Bit 0 and 1 are set to zero. Bits 2 through 7 contain a code that describes the group occurrence pointed to.
				4 Sequence number of the group occurrence pointed to.
		10xx xxxx		Pointer to a group occurrence contained in an Extension record
				Byte **Meaning**
				0-2 Control-interval number of the Extension record that contains this group occurrence.
				3 Bits 0 and 1 are set to 1 and 0, respectively. Bits 2 through 7 contain a code that describes the group occurrence pointed to.
				4 Sequence number of the group occurrence pointed to by code.
		01xx xxxx		Pointer to a group occurrence which has been deleted.
				The code in bits 2 through 7 of byte 3 and the sequence number in byte 4 have been retained however.
[1] For further information see section *Group occurrences in catalog records*.				

Figure 5.20 Path catalog record format (part 2 of 2)

User-Catalog Catalog Record

This record can only occur in the VSAM master catalog. It describes a user catalog. Figure 5.21 shows the format of the user-catalog catalog record.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
0	0	44		Key and ID area
0	0	1		Binary zeros
1	1	3	ENTIDNO	Control interval number of entry
4	4	1	RELIND	Release indicator; X'00' This record was created with a DOS/VS Release prior to Release 31 X'01' This record was created with DOS/VS Release 31 or later
5	5	6	CRAVOL	CRA volume serial number
11	B	3	CRAIDNO	CRA control interval number
14	E	4	CRADEVT	CRA device type
18	12	26		Binary zeros
44	2C	1	ENTYPE	Record type - 'U'
45	2D	2		Record length
47	2F	1		Number of variable-length fields that precede the pointer to an Extension record. Always zero.
48	30	1		Length of the fixed-length fields in this record, excluding any fixed-length fields following displacement 92 (5C). This value is always equal to the displacement from the beginning of the record to the pointer to an Extension record.
49	31	44	ENTNAME	Entry name (data set name)
The following six-byte entry contains control information for the group occurrence pointers that follow it. [1]				
93	5D	5		Pointer to Extension record. If this record is not continued on an Extension record, this field contains zeros.
98	62	1		The number of group occurrence pointers that follow.[1]
			Group occurrence pointer (repetitive)	
Bit 0 and 1 of Byte 3 identify the group occurrence further:				
		00xx xxxx		Pointer to a group occurrence within the record.
			Byte	**Meaning**
			0	Reserved
			1-2	Displacement of the group occurrence from the beginning of all group occurrences in this record.
			3	Bit 0 and 1 are set to zero. Bits 2 through 7 contain a code that describes the group occurrence pointed to.
			4	Sequence number of the group occurrence pointed to.

Figure 5.21 **User-catalog catalog record format (part 1 of 2)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
		10xx xxxx		Pointer to a group occurrence contained in an Extension record. **Byte** **Meaning** 0-2 Control-interval number of the Extension record that contains this group occurrence. 3 Bits 0 and 1 are set to 1 and 0, respectively. Bits 2 through 7 contain a code that describes the group occurrence pointed to. 4 Sequence number of the group occurrence pointed to by code.
		01xx xxxx		Pointer to a group occurrence which has been deleted. The code in bits 2 through 7 of byte 3 and the sequence number in byte 4 have been retained however.

[1] For further information see section *Group occurrences in catalog records*.

Figure 5.21 **User-catalog catalog record format (part 2 of 2)**

Volume Catalog Record

The Volume record describes VSAM data spaces, their extents, and the data sets that reside in VSAM data spaces. Figure 5.22 shows the format of the volume record.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
0	0	44		Key and ID area
0	0	1		Binary zeros
1	1	3	ENTIDNO	Control interval number of entry
4	4	1	RELIND	Release indicator; X'00' This record was created with a DOS/VS Release prior to Release 31 X'01' This record was created with DOS/VS Release 31 or later
5	5	6	CRAVOL	CRA volume serial number
11	B	3	CRAIDNO	CRA control interval number
14	E	4	CRADEVT	CRA device type
18	12	26		Binary zeros
44	2C	1	ENTYPE	Record type - 'V'
45	2D	2		Record length

Figure 5.22 **Volume catalog record format (part 1 of 3)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
47	2F	1		Number of variable-length fields that precede the pointer to an Extension record. Always zero.
48	30	1		Length of the fixed-length fields in this record, excluding any fixed-length fields following displacement 126 (7E). This value is always equal to the displacement from the beginning of the record to the pointer to an Extension record.
49	31	44	ENTNAME	Volume serial number, filled with binary zeros on the right, of the volume described by this record.
93	5D	8	VOLTSTMP	Volume time stamp, which indicates the time-of-day clock value when space was last added to, or deleted from, this volume.
101	65	20	VOLDVCHR	Device characteristics.
				Byte **Meaning**
				0-3 Volume device type.
				4-7 Maximum device blocksize.
				8-9 Number of cylinders on this volume.
				10-11 Number of tracks per cylinder on this volume.
				12-13 Number of bytes per track on this volume.
				14 Number of bytes required for gaps and check bits for each keyed block other than the last block on a track for this volume.[1]
				15 Number of bytes required for gaps and check bits for the last keyed block on a track for this volume.[1]
				16 Number of bytes to be subtracted for a block that is not keyed.[1]
				17 Flags:
		.... 1...		Block overhead for keyed records is a two-byte field (bytes 15 and 16).
		1		Use tolerance factor (bytes 18 and 19) on all blocks except the last block to calculate the effective length of a block.[1]
		xxxx .xx.		Reserved.
121	79	1	VOLRFLG	Volume record flags
122	7A	1	SYSEXTDS	Number of extents per suballocation-request allowed by the DOS/VS system.
123	7B	4		Space in the catalog record into which the following 4 bytes of derived information will be moved (in the buffer) if requested.

Figure 5.22 **Volume catalog record format (part 2 of 3)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
The following field names identify information that is not contained in the volume catalog record; the information is derived from the group occurrences in the volume catalog record and will be placed in the buffer if requested:				
123	7B	2	NODSPACE	Number of data spaces on the volume - a count of the Data Space Group occurrences.
125	7D	2	NODSET	Number of data sets on the volume - a count of the Data Set Directory Entry group occurrences.
The following six-byte entry contains control information for the group occurrence pointers that follow it. [1]				
127	7F	5		Pointer to Extension record. If this record is not continued on an Extension record, this field contains zeros.
132	84	1		Number of group occurrence pointers that follow.
				Group occurrence pointer (repetitive)
Bit 0 and 1 of Byte 3 identify the group occurrence further:				
		00xx xxxx		Pointer to a group occurrence within the record.
				Byte **Meaning** 0 Reserved. 1-2 Displacement of the group occurrence from the beginning of all group occurrences in this record. 3 Bits 0 and 1 are set to zero. Bits 2 through 7 contain a code describing the group occurrence pointed to. 4-5 Sequence number of the group occurrence pointed to.
		10xx xxxx		Pointer to a group occurrence contained in an Extension record.
				Byte **Meaning** 0-2 Control-interval number of the Extension record that contains this group occurrence. 3 Bits 0 and 1 are set to 1 and 0, respectively. Bits 2 through 7 contain a code that describes the group occurrence pointed to. 4-5 Sequence number of the group occurrence pointed to.

[1] For further information see section *Group occurrences in catalog records*.

Figure 5.22 **Volume catalog record format (part 3 of 3)**

Volume Extension Catalog Record

For this format see extension catalog record format. The record type (byte 44) is 'W'. Note that the Group occurrence pointers in a 'W' type record are 6 bytes long, similar to those in a 'V' type record.

Upgrade Set Catalog Record

An upgrade set catalog record describes an AIX in the upgrade set which is the group of alternate indexes belonging to a base cluster which are to be updated automatically. Figure 5.23 shows the format of the upgrade set catalog record.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
0	0	44		Key and ID area
0	0	1		Binary zeros
1	1	3	ENTIDNO	Control interval number of entry
4	4	1	RELIND	Release indicator; X'00' This record was created with a DOS/VS Release prior to Release 31 X'01' This record was created with DOS/VS Release 31 or later
5	5	6	CRAVOL	CRA volume serial number
11	B	3	CRAIDNO	CRA control interval number
14	E	4	CRADEVT	CRA device type
18	12	26		Binary zeros
44	2C	1	ENTYPE	Entry type - 'Y'
45	2D	2		Record length
47	2F	1		Number of variable-length fields that precede the pointer to an Extension record. Always zero.
48	30	1		Length of the fixed-length fields in this record, excluding any fixed-length fields following displacement 48 (30). This value is always equal to the displacement from the beginning of the record to the pointer to an Extension record.
The following six-byte entry contains control information for the group occurrence pointers that follow it. [1]				
49	31	5		Pointer to Extension record. If this record is not continued on an Extension record, this field contains zeros.
		1		The number of group occurrence pointers that follow.

Figure 5.23 Upgrade set catalog record format (part 1 of 2)

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
				Group occurrence pointer (repetitive)
				Bit 0 and 1 of Byte 3 identify the group occurrence further:
		00xx xxxx		Pointer to a group occurrence within the record.
				Byte **Meaning**
				0 Reserved.
				1-2 Displacement of the group occurrence from the beginning of all group occurrences in this record.
				3 Bits 0 and 1 are set to zero. Bits 2 through 7 contain a code that describes the group occurrence pointed to.
				4 Sequence number of the group occurrence pointed to.
		10xx xxxx		Pointer to a group occurrence contained in an Extension record.
				Byte **Meaning**
				0-2 Control-interval number of the Extension record that contains this group occurrence.
				3 Bits 0 and 1 are set to 1 and 0, respectively. Bits 2 through 7 contain a code that describes the group occurrence pointed to.
				4 Sequence number of the group occurrence pointed to by code.
		01xx xxxx		Pointer to a group occurrence which has been deleted.
				The code in bits 2 through 7 of byte 3 and the sequence number in byte 4 have been retained however.

[1] For further information see section *Group occurrences in catalog records*.

Figure 5.23 **Upgrade set catalog record format (part 2 of 2)**

Group Occurrences in Catalog Records

Group occurrences are fields of related information within a catalog record which are grouped together so that they can be treated as a unit. For example, all fields relating to a volume on which a data set resides are located in one group occurrence. There is one group occurrence for each volume concerned. Thus, if a data set resides on three volumes, there are three group occurrences, which are not necessarily contiguous.

The group occurrences are found by means of group occurrence pointers in the catalog record. These pointers also contain a code which identifies the type of group occurrence to which they point. The pointers are grouped in such a way that all pointers to a particular type of group occurrence are

together. In the example above, the pointers to the three volume information group occurrences would be contiguous, even if the group occurrences are not.

Group Occurrences in Extension Records

This description is designed to show the steps involved in building the group of records shown in Figure 5.24. The numbers in parentheses in the description refer to the record numbers in Figure 5.24.

Initially, when the base record (1) is created, the group occurrence pointers (GOPs) and group occurrences (GOs) are placed in the base record. As further GOs and GOPs are added, the available space becomes insufficient and the base record must be extended.

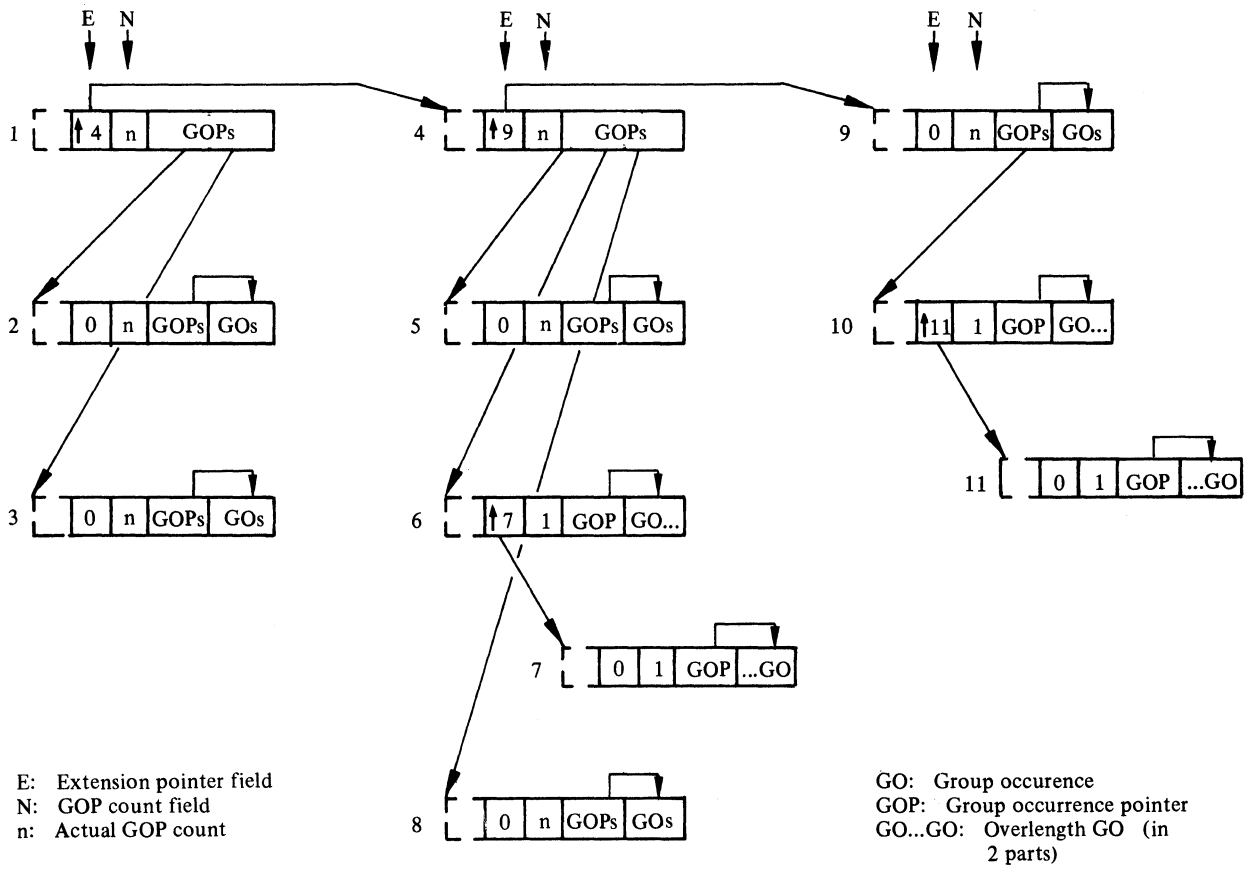
This is done by means of a **vertical chain** of extension records. GOs are moved out of the base record (1) into an extension record (2) to create space for GOPs. If necessary, further extension record(s) (3) can be added to this vertical chain. The GOs in the extension records are located by means of two GOPs: the first GOP, in the base record, points to the extension record, where a second GOP with the same group code and sequence number as the first, points to the associated GO within the extension record.

Later, the GOPs in the base record may require more space than is available, although all GOs have been moved to the vertical chain. In this case, the base record (1) is extended by a **horizontal chain**. The first record in the horizontal chain(4) is located by means of the extension pointer in the base record (1). This extension record (4) is now treated similarly to the original base record. It first contains both GOPs and GOs, then extended with records in a vertical chain (5,6,8), and finally extended horizontally (9).

If it becomes necessary to store a GO which will not fit into a single record, a **pseudo-horizontal chain** becomes necessary. If this is required within a vertical chain (6), the extension record (7) is simply chained with the aid of the extension pointer in the record containing the first part of the GO (6).

A pseudo-horizontal chain is however, not allowed within the horizontal chain (1,4,9). Thus, if an overlong GO is encountered at a time where the last record in the horizontal chain (9) still contains GOPs, GOs, and free space (which means that the next GO would normally be placed in this record) an artificial vertical chain (10) must first be built. This can the be extended in the form of a pseudo-horizontal chain (10,11) to hold the GO.

The first record in a pseudo-horizontal chain always contains an extension pointer to the second record and always has a GOP count of 1, because such a "spanned" GO may not share a record with any other GO.



Record 1: Base record, containing GOPs pointing to records 2 and 3 (vertical pointers) and an extension pointer to record 4 (horizontal pointer).

Record 2: Extension record, containing GOs from record 1 } Vertical
Record 3: Extension record, containing GOs from record 1 } chain

Record 4: Extension record in the horizontal chain, containing further GOPs pointing to records 5,6, and 8 (vertical pointers) and an extension pointer to record 9 (horizontal pointer).

Record 5: Extension record containing GOs from record 4
Record 6: Extension record containing 1st part of long GO } Pseudo chain } Vertical chain
Record 7: Extension record containing 2nd part of long GO } Pseudo chain } Vertical chain
Record 8: Extension record containing GOs from record 4

Record 9: Extension record containing further GOPs and GOs (last record in the horizontal chain)

Record 10: Extension record containing 1st part of long GO } Artifical vertical chain
Record 11: Extension record containing 2nd part of long GO } with pseudo-horizontal chain

Figure 5.24 Group occurrences in extension records.

Types of Group Occurrences

The following list shows the various types of group occurrences, and the types of catalog records in which they can occur. Note that it is possible for one catalog record to contain many group occurrences.

- AMDSB (Access Method Data Set Statistics Block), which appears in Data and Index records. Only one copy of an AMDSB appears in a record. A pointer to AMDSB information contains a code of 1.
- Association information, which occurs in Data, Index, Cluster, Alternate Index, Path, and Upgrade Set records. For further details, see *Association Group Occurrences* later in this section. A pointer to association information contains a code of 2.
- Volume information, which appears in Data, Index, User Catalog, and non-VSAM records. This group occurrence describes all of the direct-access device space allocated to the data set (or index, etc.) on a particular volume. A separate set of volume information fields is used to describe the total space on each volume. If the data set's space on a volume is divided into key ranges, each key range is described in a separate set of volume-information fields. As many sets of volume-information fields as are required to describe allocated space can appear. A pointer to volume information contains a code of 3.
- Password information, which can appear in Data, Index, Path, Cluster and Alternate index records. This group occurrence contains the security information for a data set (or index, etc.). Only one set of password-information fields can appear. A pointer to password information contains a code of 4.

The following group occurrences occur only in Volume catalog records.

- Space map group occurrence. This group occurrence describes each track on the volume as allocated to a VSAM object or unallocated. Each volume record contains as many group occurrences as are necessary to reflect the total space on the volume. A pointer to track allocation information contains a code of 5.
- Data space group occurrence. This group occurrence describes a VSAM data space on the volume. One group occurrence is required to describe each data space and its extents on the volume. A pointer to data-space information contains a code of 6.
- Data set directory entry group occurrence. This group occurrence describes a data set that resides in a VSAM data space. One group occurrence is required for each data set. A pointer to data-set information contains a code of 8.

Association Group Occurrences

These group occurrences identify the relationships between the various records which describe a data set and its associated components.

Each Cluster, Data, Index, Alternate index, Path, or Upgrade Set catalog record can contain one or more association group occurrences, depending on the overall configuration of the data set.

Figure 5.25 shows a possible configuration for a data set which uses all possible association group occurrences. It consists of a key-sequenced data set (base cluster), an alternate index which belongs to the upgrade set, and two paths. Path 1 is defined over the base cluster and serves as an "alias", while path 2 is defined over the AIX and is used to address the AIX/base cluster combination.

The arrows in the figure represent the "direction" of the pointers in the group occurrences. For example, the base cluster record and the path 1 record contain pointers to each other. The path 1 record, however, points only to the base index and data records; there is no "backward" pointer from these records to the path record.

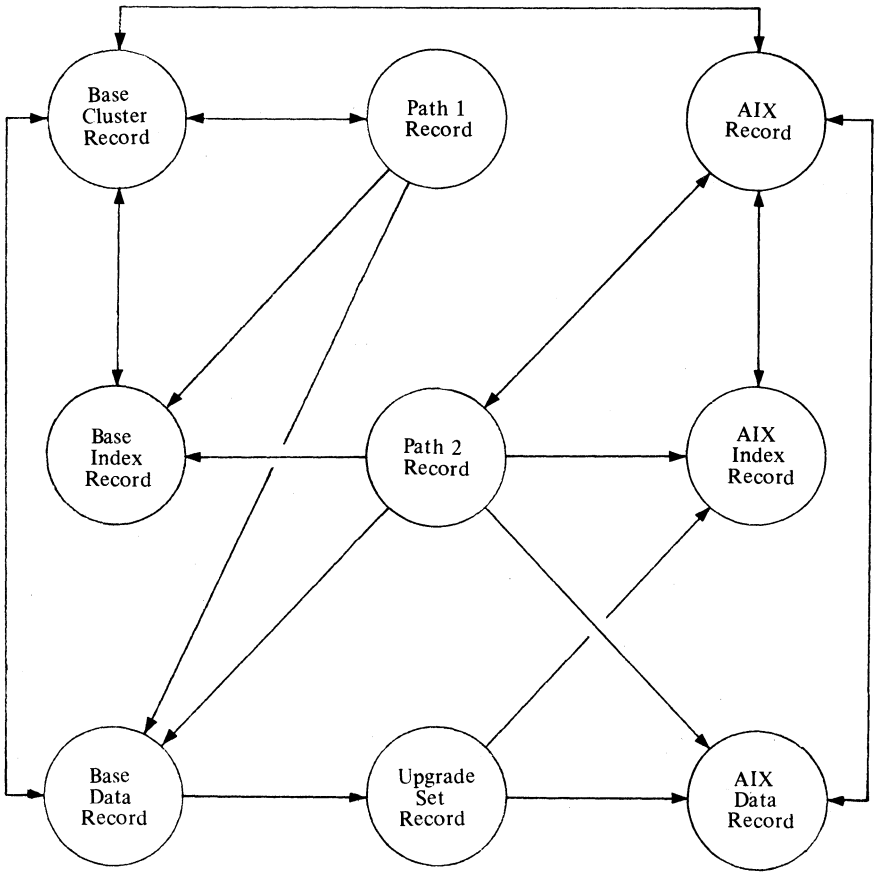


Figure 5.25 **Association group occurrences**

Figure 5.25 shows at least one of each type of association group occurrence. If, however, the data set configuration contains more alternate indexes, paths, etc. there will be correspondingly more association group occurrences which point to the additional components. The base cluster record, for example, will have an association group occurrence pointing to each alternate index record.

Group Occurrence Formats

The following section shows the format of each type of group occurrence.

AMDSB Group Occurrence

The AMDSB group occurrence contains a copy of the AMDSB control block, and is updated each time the data set is closed. This group occurrence is associated with a pointer that contains a group code of 1. Figure 5.26 shows the AMDSB format.

Field Size	Field Name	Description
2		Control information **Byte** **Meaning** 0 Count of the number of variable fields in this group occurrence 1 Hexadecimal displacement to the first variable-length field from the beginning of this group occurrence
96	AMDSBCAT	AMDSB. Only the first 96 bytes of the AMDSB are cataloged. The remainder are ignored. See description and format of AMDSB for detailed information, except for the following special case in the AMDSBs of an AIX. The field at offset 4 of the index AMDSB and of the group occurrence is referenced with the name AMDNEST and contains the number of entries in an index section. In the data AMDSB group occurrence, this field is called AMDAXRKP and contains the relative key position (in the base cluster record) for this AIX. When the AMDSB group occurrences are transferred to the AMDSBs at OPEN time, the contents of AMDAXRKP are moved to field AMDAIRKP (at offset 118 decimal) and the field at offset 4 (now again called AMDNEST) is loaded with the contents of AMDNEST from the index AMDSB. Note that this movement of data is carried out by OPEN for all AMDSBs, even through the fields may be empty, or already contain the correct information.

Figure 5.26 AMDSB group occurrence format

Association Group Occurrence

The control interval number of a related record is contained in an association group occurrence. This group occurrence is associated with a pointer which contains a group code of 2. Figure 5.27 shows the format.

Field Size	Field Name	Description	
2		Control information	
		Byte	**Meaning**
		0	Count of the number of variable fields in this group occurrence
		1	Hexadecimal displacement to the first variable-length field from the beginning of this group occurrence
1	TYPE	'D'	if this entry describes a Data record
		'I'	if this entry describes an Index record
		'C'	if this entry describes a Cluster record
		'G'	if this entry describes an Alternate Index record
		'R'	if this entry describes a Path record
		'Y'	if this entry describes an Upgrade Set record
3	NAME	Control interval number of the record specified above	
The following fields are present only in an association group occurrence located in an Upgrade Set Record [1]			
1	TYPE2	'I'	This entry can describe only an Index record
3	NAME2		Control interval number of the Index record

[1] Each association group occurrence in an Upgrade Set record describes the data component (TYPE, NAME) and the index component (TYPE2, NAME2) of an alternate index.

Figure 5.27 Association group occurrence format

Volume Information Group Occurrence

All extents allocated to the data set, index, or data set's key range on a volume are described by a volume information group occurrence. This group occurrence is associated with a pointer that contains a group code of 3. Figure 5.28 shows its format.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description	
0	0	2	ENTVOL	Control information	
				Byte	**Meaning**
				0	Count of the number of variable fields in this group occurrence
				1	Hexadecimal displacement to the first variable-length field from the beginning of this group occurrence.
2	2	4	DEVTYP	Device type	
6	6	6	VOLSER	Volume serial number	
12	C	2	FILESEQ	File sequence number. (This field is used for nonVSAM data sets that reside on tape volumes.)	

Figure 5.28 Volume information group occurrence format (part 1 of 2)

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
14	E	1	VOLFLG	Volume flags, as follows:
		1...		Prime, which indicates that this volume was allocated when the data set was defined or that a data set that is not divided into parts according to key has been extended to this volume.
		.1..		Candidate, which indicates that this volume is available for use by the data set described by this record.
		..1.		Overflow, which indicates that this volume is being used by a data set that is divided into parts according to key, but this volume was not allocated when the data set was defined.
		...x xxxx		Reserved
The following fields are used only in Data and Index records				
15	F	1	NOEXTNT	Number of extents allocated in this set of extents on this volume for this data set.
16	10	4	HKRBA	RBA of the data control interval with the high key.
20	14	4	HURBA	High used RBA on this volume
24	18	4	HARBA	High allocated RBA on this volume
28	1C	4	PHYBLKSZ	Block size
32	20	2	NOBLKTRK	Number of blocks per track
34	22	2	NOTRKAU	Number of tracks per control area
36	24	1	ITYPEXT	Flags:
		1...		In an index record: the sequence set is with the data.
		.1..	ARDPRFMT	The extents are not preformatted
		..xx xxxx		Reserved
37	25	2	DSDIRSN	Data-set directory sequence number in the volume control
39	27	Variable	LOKEYV	Low key on the volume. This field can be a maximum of 64 bytes long; the first two bytes indicate the length of the field. For non-KSDS files, this field contains two bytes of zeros.
		Variable	HIKEYV	High key on the volume. This field can be a maximum of 64 bytes long; the first two bytes indicate the length of the field. For non-KSDS files, this field contains two bytes of zeros.
		Variable	EXTENT	The field contains a 2-byte length field, followed by a 20-byte field for each extent. The 20-byte field describes the start and end of the extent, in the form SSCCHHCCHHTTDDDDDDDD, where SS is the data-space sequence number, CCHHCCHH is the low and high cylinder and head, TT is the number of tracks, and DDDDDDDD is the low and high RBA of the extent.

Figure 5.28 Volume information group occurrence format (part 2 of 2)

Password Group Occurrence

Password information, if any, is contained in the password group occurrence. This group occurrence is associated with a pointer that contains a group code of 4. Figure 5.29 shows its format.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
0	0	2		Control information **Byte** — **Meaning** 0 — Count of the number of variable fields in this group occurrence 1 — Hexadecimal displacement to the first variable-length field from the beginning of this group occurrence.
2	2	32	PASSWORD	Four eight-character passwords, in the following order: MASTER, CONTROL-INTERVAL, UPDATE, and READ.
34	22	8	PASSPRMT	Password prompting code name that allows the operator to provide the correct password without displaying the data-set name.
42	2A	2	PASSATMP	Maximum number of attempts allowed for the operator to provide correct password.
44	2C	8	USVRMDUL	Name of the user's security-verification module, if any.
52	34	Variable	USERAREC	User-authorization record. The first two bytes give the length of the user record (up to 256) which is contained in the following bytes.

Figure 5.29 Password group occurrence format

Space Map Group Occurrence

The tracks on a VSAM volume are allocated to a VSAM object, or are unallocated, as described by the Space Map group occurrence. Each bit position describes one track as allocated (bit =0) or unallocated (bit = 1). This group occurrence is associated with a pointer that contains a group code of 5. Figure 5.30 shows its format.

All tracks on the volume are mapped, starting at cylinder 0, track 0. Any tracks not owned by VSAM are marked 'allocated' in order to ensure that they are not used by VSAM data sets.

Field Size	Field Name	Description
2		Control information: **Byte** **Meaning** 0 Count of the number of variable-length fields in this group occurrence (X'01') 1 Hexadecimal displacement to the variable-length field from the beginning of the group occurrence (X'02').
2		Length of the field BITMAP
Variable	BITMAP	Portion of the volume bit map (1 to 440 bytes describing the allocated/unallocated status of 1 to 3520 tracks).

Figure 5.30 **Space map group occurrence format**

Data Space Group Occurrence

Each VSAM data space on the volume is described with a Data Space Group ocurrence. This group occurrence is associated with a pointer that contains a group code of 6. Figure 5.31 shows its format.

Bytes and Bit Pattern	Field Name	Description
2		Control information **Byte** **Meaning** 0 Count of the number of variable-length fields in this group occurrence (X'00'). 1 Hexadecimal displacement to the first variable-length field from the beginning of this group occurrence (X'55').
8	DSCBTS	Format-1 DSCB time stamp, which indicates when the DSCB was created. The time stamp is part of the name given to the format-1 DSCB.
5	DSCBPTR	CCHHR of the format-1 DSCB
1	SPHDFLG	Data-space flags
1...		Unique data space, that is, this data space contains all or part of only one VSAM object.
0...		Shared data space, that is, this data space contains all or part of two or more VSAM objects.
.1..		Automatically built data space, that is, this data space was built as a result of an end-of-volume request.
..1.		This data space contains a VSAM user catalog.
...1		This data space contains a VSAM master catalog.
.... 1...		This data space was built when the user issued an Access Method Services DEFINE catalog command.
.... .xxx		Reserved
1	NODSPEXT	Number of extents in this data space

Figure 5.31 **Data space group occurrence format (part 1 of 2)**

Bytes and Bit Pattern	Field Name	Description
1	DSPSOPT	Data-space allocation options
10..		Track request, which indicates that primary space was allocated in terms of tracks.
11..		Cylinder request, which indicates that primary space was allocated in terms of cylinders.
..xx xxxx		Reserved
3	DSPSSQ	Secondary space allocation quantity by which space is to be extended if required. This value is taken either from an Access Method Services DEFINE command or from the first data set on this volume that caused space to be used. This field is used only by OS/VS VSAM.
64	SPEXTENT	Sixteen 4-byte extent descriptors in the form TTNN: TT - starting track number of the extent (relative to the beginning of the volume). NN - number of tracks in the extent

Figure 5.31 Data space group occurrence format (part 2 of 2)

Derived Data Space Information

The following field names identify information that is expected, but not contained in the data space group occurrence. The information is derived from fields in the volume catalog record. Figure 5.32 shows its format.

Bytes and Bit Pattern	Field Name	Description
1	SPHDFLG	Data space flags:
..1.		A VSAM user catalog is in this data space
...1		A VSAM master catalog is in this data space
xx.. xxxx		Reserved
2	NODSDSP	Number of data sets in the data space - this information is derived by searching each data and index catalog record (pointed to by Data Set Directory Entry group occurrences and Cluster catalog records) for a volume information group occurrence that contains the volume's serial number. Each group occurrence so identified is searched to determine if the data set or index has been allocated space in one of the data space's extents.
The following field names refer to information about an extent of the data space:		
2	TRKSUSED	Number of allocated tracks in the extent - the Space Map group occurrence is scanned to determine the number of allocated tracks, based on the extent's starting track number and total number of tracks (contained in SPEXTENT).

Figure 5.32 Derived data space information format (part 1 of 2)

Bytes and Bit Pattern	Field Name	Description
4	EXTSTART	Cylinder and track on which the extent begins - the extent's TT value (contained in SPEXTENT) is converted to a CCHH value.
2	NOTRKEXT	Number of tracks in the extent - the extent's value (contained in SPEXTENT).
2	SNSPHD	Sequence number of the group occurrence that describes the extent's data space - the sequence number of the Data Space Group occurrence.
Variable	SPACEMAP	A variable-length space map that defines the allocated and unallocated space in the extent - the Space map group occurrence is converted to the format of this variable-length field based on the extent's starting tracks number and total length (contained in SPEXTENT).

Figure 5.32 Derived data space information format (part 2 of 2)

Data Set Directory Entry Group Occurrence

Each data set that resides in a VSAM data space on the volume is described with a Data Set Directory Entry group occurrence. This group occurrence is associated with a pointer that contains a group code of 8. Its format is shown in Figure 5.33.

Field Size	Field Name	Description
2		Control information **Byte** **Meaning** 0 Count of the number of variable fields in this group occurrence (X'00') 1 Hexadecimal displacement to the first variable-length field from the beginning of this group occurrence.
3	DSIDNO	Control-interval number of the Data or Index catalog record that describes this data set or index.

Figure 5.33 Data set directory entry group occurrence format

Derived Data Set Information

The following field names identify information that is expected, but not contained in, the Data Set Directory Entry group occurrence. The information is derived from fields in the volume catalog record. Figure 5.34 shows its format.

Bytes and Bit Pattern	Field Name	Description
1	DSDIRFLG	Flags:
1...		The Data or Index catalog record identifies the volume as a candidate volume -the Data or Index catalog record is searched; its volume information group occurrence VOLFLG field candidate volume indicator (B '.1.. ....' flag) is on.
Variable	DSSPSN	A variable-length collection of 3-byte fields that identify each data space within which the data set has extents allocated to it - this information is obtained by converting each volume information group occurrence's extent descriptor's (EXTENT) SS value (data space extent's sequence number) so that the resulting 3-byte field is: **Byte** **Meaning** 0-1 Sequence number of the Data Space Group occurrence 2 Number of extents (groups of contiguous tracks) assigned to the data set or index from the data space (limits: 1 - 123).

Figure 5.34 **Derived data set information**

Field Name Dictionary

The field name dictionary (defined in IGG0CLAY) is an internal data area that provides a map between field names and fields within catalog records, as well as derived information that is not contained in catalog records. The dictionary also allows the dictionary user to specify values (for example, the number of group occurrences to be processed) by associating them with a dictionary name. Using a field name specified in the CTGFL (field parameter list), the catalog management modules reference the field name dictionary for the location, length, and type of fields.

The field name dictionary is a series of 8-byte entries. The first four bytes contain a shortened field name consisting of the first, second, fifth, and sixth character of the field name in IKQCTGFL. Bytes 4-7 of the field name dictionary entry describe the field. When a caller makes a request in a CTGFL, dictionary information is moved from the dictionary to the CTGFL.

Figure 5.35 shows the field name dictionary entry format.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Description
0		4	Shortened field name: the first, second, fifth, and sixth characters of the eight-character field name.
4		1	Flags that describe the field.
			Value **Meaning**
		xxx.	000 The field is fixed-length and appears in the header portion of a record (before the Extension record pointer).
			001 A combination field name[1]
			010 The field is fixed-length and is part of a group occurrence that follows the Extension record pointer.
			100 The field is variable-length and appears in the header portion of a record (before Extension record pointer).
			110 The field is variable-length and is part of a group occurrence that follows the Extension record pointer.
			011 Special field[2]
			111 Special field[2]
		...x	0 Not a flag field, which means that a CLC (compare logical character) instruction can be used to test this field.
			1 Flag field, which means that a TM (test under mask) instruction can be used to test this field.
		.... x...	0 Not a fixed-length field within a variable-length field in a group occurrence
			1 A fixed-length field within a variable-length field in a group occurrence
		xxx	Reserved
5	5	1	Bytes that identify the location of the field:
			Type of field name: — **Contents of this byte:**
			Fixed-length: — Displacement in bytes from the
			In the header: — beginning of the record
			In a group occurrence: — beginning of the group occurrence
			In a group of fixed-length fields within a variable-length field: — Length of the group of fixed-length fields
			Variable-length: — Zero
			Combination: — Index value in the combination-name index

Figure 5.35 **Field name directory entry format (part 1 of 2)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Description
6	6	1	Bytes that identify the location of the field (continued):
			Type of field name: / **Contents of this byte:**
			Fixed-length: / Length of the field (in bytes)
			Variable-length: / Sequence number of the field
			Combination: / Number of fields identified by the combination name
			Value / **Meaning**
7	7	x...	0 / Not a non-repeating volume entry field
			1 / A non-repeating volume entry field (i.e., a header field)
		.xxx xxxx	Group code

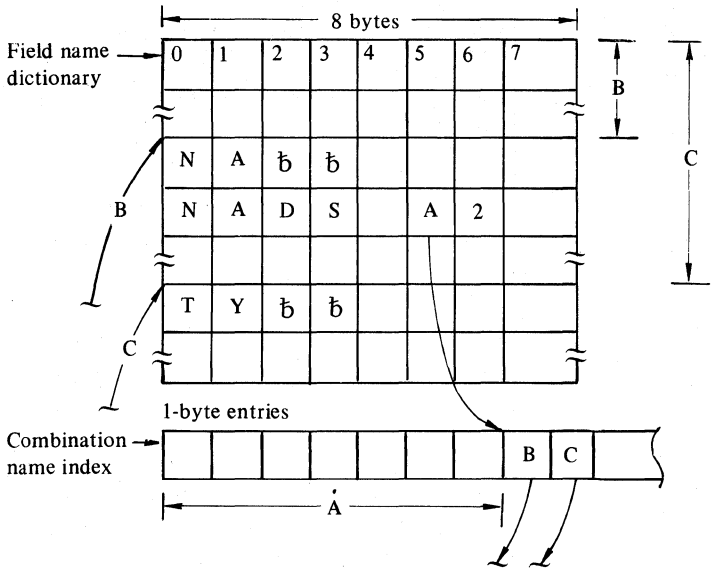
[1] *Combination field name* indicates that the name supplied is a name that allows catalog management to locate a group of related fields.

[2] The field-name dictionary permits catalog management to locate information that is not contained in catalog records.

Figure 5.35 Field name dictionary entry format (part 2 of 2)

Combination names are also included in the field name dictionary. Each combination name allows catalog management to locate more than one field at a time by means of a combination name index.

The following example shows the relationship of the field name dictionary to the combination name index. The three entries in this example are shortened field names for NAME, NAMEDS, and TYPE. Since NAME and TYPE are only four characters in length, the fifth and sixth characters are blanks. The shortened versions, NA␣␣ and TY␣␣, are combinations of the first, second, fifth and sixth characters of the original field names. NAMEDS is a combination field name that includes fields NAME and TYPE. Thus, byte 5 contains an index value (A) that points to the displacement in the combination name index (B) which contains, in turn, the displacement of the first field name (NAME) in the field name dictionary. Because byte 6 contains a 2 (meaning that this combination name includes two field names), the displacement of the second field name (TYPE) in the field name dictionary is given in the byte C following the first displacement value in the combination name index.



where

A, B, C	=	displacement values
NAbb	=	NAMEbb
NADS	=	NAMEDS
TYbb	=	TYPEbb

Figure 5.36 **Example of a combination name in field name dictionary**

Figure 5.37 identifies the catalog dictionary entries by field name and briefly describes each field.

Field Name	Record Type[1]	Group Code[2]	Description
AMDSBCAT	DI	1	Combination name that includes fields AMDS1, HILIRBA, SSRBA, AMDS2, TIMESTMP, CATSTAT
AMDS1	DI	1	Part of AMDSB
AMDS2	DI	1	Part of AMDSB
ATTR1	DI	-	Data set attributes
ATTR2	DI	-	Data set sharing attributes
BITMAP	V	5	Portion of volume bit map showing the allocated and unallocated tracks on a direct-access volume
BUFSIZE	DI	-	Minimum buffer size
CATACB[3]			Address of catalog ACB
CATSTAT	DI	1	Part of AMDSB
CATVOL	ADIU	3	Combination name[4] that includes fields RELREPNO, DEVTYP, VOLSER, FILESEQ, VOLFLG
CNTREPNO[3]			Maximum number of RELREPNOs to be processed
CRAIDNO	ACDGIRUVY	-	CRA control interval number
CRADEVT	ACDGIRUVY	-	CRA device type
CRAVOL	ACDGIRUVY	-	CRA volume serial number
DATASPAC	V	6	Combination name that includes fields RELREPNO, SPHDFLG, DSCBPTR, DSCBTS, NODSPEXT, DSPSOPT, DSPSSQ, SPEXTENT
DEVTYP	ADIU	3	Device type
DEXTENTS	V	6	Combination name that includes fields RELREPNO, SPHDFLG, NODSPEXT, SPEXTENT
DIRECTRY	V	8	Combination name that includes fields RELREPNO, DSIDNO
DSATRO	DI	-	Combination name that includes fields ATTR1, ATTR2, OPENIND
DSATTR	DI	-	Combination name that includes fields ATTR1, ATTR2
DSCBPTR	V	6	CCHHR of F1 DSCB
DSCBTS	V	6	F1 DSCB time stamp
DSDIRECT	V	8	Combination name that includes fields RELREPNO, NODSEXT, DSIDNO, DSDIRFLG, DSSPSN
DSDIRFLG[5]	V	8	Flags
DSDIRSN	DI	3	Data set directory sequence number in the extent descriptor of the volume information group occurrence

Figure 5.37 **Catalog dictionary entries (part 1 of 6)**

Field Name	Record Type[1]	Group Code[2]	Description
DSETCRDT	CDGIR	-	Data set creation date (YDD)
DSETEXDT	CDGIR	-	Data set expiration date (YDD)
DSIDNO	V	8	Control interval number of Data or Index catalog record that describes this data set or index
DSPDSCRP	V	7	Combination name that includes fields RELREPNO, TRKSUSED, EXTSTART, NOTRKREXT, SNSPHD, SPACEMAP
DSPSOPT	V	6	Options for data space creation
DSPSSQ	V	6	Secondary data space allocation quantity
DSSPSN[5]	V	8	Sequence number and number of extents
DSTYPNAM	ACDGIRUVY	-	Combination name that includes fields ENTYPE, ENTIDNO
ENTIDNO	ACDGIRUVY	-	Control interval number of this catalog record
ENTNAME	ACDGIRUVY	-	Data set name (Data, Index, or NonVSAM catalog record), cluster name (Cluster catalog record), volume serial number (Volume catalog record)
ENTVOL	DI	3	Combination name that includes fields RELREPNO, DEVTYP, VOLSER, FILESEQ, VOLFLG, NOEXTNT, HKRBA, HURBA, HARBA, PHYBLKSZ, NOBLKTRK, NOTRKAU, ITYPEXT, DSDIRSN, LOKEYV, HIKEYV, EXTENT
ENTYPE	ACDGIRUVY	-	Catalog record type
EXCPEXIT	DI	-	Exception exit
EXTENT	DI	3	Extent descriptors
EXTSTART[5]	V	7	Cylinder and track on which extent begins
EXTVOL	DI	3	Combination name that includes fields RELREPNO, DEVTYP, ITYPEXT, EXTENT
FILESEQ	ADIU	3	File sequence number for a nonVSAM data set that resides on tape volume(s)
GENDSP[3]			Generated data space name
HARBA	DI	3	High-allocated RBA on this volume for this data set
HARBADS	DI	-	High-allocated RBA of the data set or index (not always the same as HARBA if the data set resides on more than one volume)

Figure 5.37 Catalog dictionary entries (part 2 of 6)

Field Name	Record Type[1]	Group Code[2]	Description
HIKEYV	DI	3	Key-sequenced data set's high key value on a volume or, if the data set is divided into key ranges, the key range's high key value
HILIRBA	DI	1	AMDSB high-level index RBA
HKRBA	DI	3	RBA of the record containing the high key of a key-sequenced data set on a volume or, if the data set is divided into key ranges, the key range's high key value
HKURBA	DI	3	Combination name that includes fields HKRBA, HURBA
HURBA	DI	3	High-used RBA on this volume for this data set
HURBADS	DI	-	High-used RBA for the data set or index (not always the same as HURBA if the data set resides on more than one volume)
ITYPEXT	DI	3	Flag
LOKEYV	DI	3	Key-sequenced data set's low key value on a volume, or if the data set is divided into key ranges, the key range's low key value
LRECL	DI	-	Logical record size (data set) or X'FF' (index)
MAPSPACE	V	5	Combination name that includes fields RELREPNO, BITMAP
NAME	CDGIRY	2	Control interval number of associated catalog record
NAME2	Y	2	Control interval number of associated catalog record
NAMEDS	CDGIRY	2	Combination name that includes fields TYPE, NAME
NOBLKTRK	DI	3	Number of blocks per track
NODSDSP[5]	V	6	Number of data sets in this data space
NODSET[5]	V	-	Number of data sets on this volume
NODSEXT[5]	V	8	Number of data set directory extents
NODSPACE[5]	V	-	Number of data spaces on this volume
NODSPEXT	V	6	Number of extents in this data space
NOEXTNT	DI	3	Number of extents allocated in this set of extents on this volume for this data set
NOTRKAU	DI	3	Number of tracks per allocation unit
NOTRKEXT[5]	V	7	Number of tracks in extent
OPENIND	DI	-	Open indicator flag

Figure 5.37 **Catalog dictionary entries (Part 3 of 6)**

Field Name	Record Type[1]	Group Code[2]	Description
OPNCALL1	DI	-	Combination name that includes fields SPACOPTN, BUFSIZE, ENTNAME, HURBADS
OWNERID	CDGIR	-	Data set owner identification
PASSATMP	CDGIR	4	Maximum number of attempts allowed operator to provide correct password
PASSPRMT	CDGIR	4	Password prompting code name for security verification
PASSWALL	CDGIR	4	Combination name that includes fields PASSWORD, PASSPRMT, PASSATMP, USVRMDUL, USERAREC
PASSWORD	CDGIR	4	Four 8-character password
PHYBLKSZ	DI	3	Physical block size
PRIMSPAC	DI	-	Primary space allocation
RELCRA	ACDGIRUVY	-	Combination name that includes fields RELIND, CRAVOL, CRAIDNO, CRADEVT
RELIND	ACDGIRUVY	-	VSAM release indicator
RELREPNO[3]			Two-byte relative repetition number
REPNO[3]			Two-byte highest repetition number found
RGATTR	GR	-	Path/alternate index attributes
SCONSPAC	DI	-	Secondary space allocation requirement
SNSPHD[5]	V	7	Sequence number of the group occurrence that describes extent's data space
SPACEHDR	V	6	Combination name that includes fields RELREPNO, DSCBTS, DSCBPTR, SPHDFLG, NODSDSP, NODSPEXT, DSPSOPT, DSPSSQ
SPACEMAP[5]	V	7	Data space map that defines allocated and unallocated space in the extent
SPACOPTN	DI	-	Space options flags
SPACPARM	DI	-	Combination name that includes fields PRIMSPAC, SCONSPAC, SPACOPTN
SPEXTENT	V	6	Extent descriptors

Figure 5.37 Catalog dictionary entries (Part 4 of 6)

Field Name	Record Type[1]	Group Code[2]	Description
SPHDFLG[5]	V	6	Data space flags
SSRBA	DI	1	AMDSB sequence-set RBA
SYSEXTDS	V	-	Number of system-allowed extents per suballocation request
TIMESTMP	DI	1	AMDSB time stamp
TRBAEXT[3]	DI	3	This is a test RBA for EOV mount by RBA
TRKSUSED[5]	V	7	Number of allocated tracks per extent
TYPE	CDGIRY	2	Associated catalog record type
TYPE2	Y	2	Associated catalog record type
UPDVOL	DI	3	Combination name that includes fields RELREPNO, DEVTYP, VOLSER, FILESEQ, VOLFLG, NOEXTNT, HKRBA, HURBA, HARBA, PHYBLKSZ, NOBLKTRK, NOTRKAU, ITYPEXT, DSDIRSN, LOKEYV, HIKEYV
UPGRADE	Y	2	Combination name that includes fields TYPE, NAME, TYPE2, NAME2
USERAREC	CDIGR	4	User authorization record
USERINFO	DI	-	User information for DOS/VS ISAM Interface Program
USVRMDUL	CDIGR	4	User security verification module name
VOLDEV	DI	3	Combination name that includes fields DEVTYP, PHYBLKSZ, NOBLKTRK, NOTRKAU
VOLDVCHR	V	-	Device characteristics
VOLEXT	DI	3	Combination name that includes fields RELREPNO, VOLSER, DSDIRSN, EXTENT
VOLFLG	ADIU	3	Volume flags
VOLPHY	DI	3	Combination name that includes fields RELREPNO, VOLSER, VOLFLG, NOEXTNT, HKRBA, HURBA, HARBA, ITYPEXT, LOKEYV, HIKEYV, EXTENT
VOLRFLG	V	-	Volume record flags
VOLSER	ADIU	3	Volume serial number
VOLTSTMP	V	-	Volume time stamp

Figure 5.37 **Catalog dictionary entries (Part 5 of 6)**

[1] Record type indicates which catalog record contains the field name:

A - non-VSAM
C - Cluster
D - Data
G - Alternate index
I - Index
R - Path
U - User catalog
V - Volume
Y - Upgrade set

[2] Group code indicates which group occurrence contains the field:

- Field is in the header portion of the record, not associated with any group occurrence
1 AMDSB group occurrence
2 Association group occurrence
3 Volume information group occurrence
4 Password group occurrence
5 Space map group occurrence (contained only in a Volume catalog record)
6 Data space group occurrence (contained only in a Volume catalog record)
7 Space descriptor group occurrence (derived)
8 Data set directory entry group occurrence (contained only in a Volume catalog record)

[3] Special names not contained in the catalog; their values are derived from other sources, in most cases to locate repetitive data and control processing.

[4] Combination names allow catalog management to locate more than one field at a time

[5] These field names identify information that is expected, but not contained in, the data space or data set directory group occurrences. The information is derived from fields in the Volume catalog record.

Figure 5.37 Catalog dictionary entries (Part 6 of 6)

To clarify the use of the dictionary as a means of gaining access to catalog information, refer to the examples that follow.

Dictionary Example 1

The DSETCRDT (data set creation date) field appears in the dictionary, as follows:

C4E2C3D900650300

The first 00 is the fifth-byte value of the record; it indicates that DSETCRDT is (a) a fixed-length field, (b) not part of a group occurrence, and (c) not a flag field.

The 65 is the sixth-byte value of the record; it indicates, that DSETCRDT is at displacement X"65" from the beginning of the record in which it appears.

The 03 is the seventh-byte value of the record; it indicates that DSETCRDT is three bytes long.

The 00 is the eighth-byte value of the record; it is zero because DSETCRDT is not part of a group occurrence and, therefore, is not associated with a group occurrence code.

Although the dictionary entries are 8 bytes long, only the first, second, fifth, and sixth characters of the field name appear. For DSETCRDT, only DSCR (C4E2C3D9) appears.

Dictionary Example 2

The DSPSOPT (data-space-creation space options) field appears in the dictionary, as follows:

C4E2D6D750130106

The 50 is the fifth-byte value of the record; it indicates, when converted to binary, that DSPSOPT is (a) a fixed-length field that is part of a group occurrence, (b) a flag field, and (c) not a repeating field within a variable-length field.

The 13 is the sixth-byte value of the record; it indicates that DSPSOPT is at displacement X'13' from the beginning of the record in which it appears.

The 01 is the seventh-byte value of the record; it indicates that DSPSOPT is one byte long.

The 06 is the eighth-byte value of the record; it indicates that DSPSOPT is part of a group occurrence associated with a code of 6, which means that it is part of a group occurrence that contains VSAM data-space information.

Access Method Block List (AMBL)

The AMBL describes a VSAM cluster and points to the cluster's data set and index AMDSBs. When the cluster is opened, an AMBL is built to describe the cluster. AMDSBs are built to describe the cluster's data set and, if the cluster is key-sequenced, to describe the index. The AMBL is pointed to by the cluster's ACB (ACBAMBL). Figure 5.38 shows the AMBL format.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0		AMBLST		Beginning of AMBL
0	0	1	AMBLID	X'11'	AMBL identifier
1	1	1	AMBLACT		AMBL active byte (X'FF'=AMBL is active)
2	2	2	AMBLLEN		Length of control block
4	4	4	AMBLDTA		Pointer to data AMDSB
8	8	4	AMBLIX		Pointer to index AMDSB
12	C	4	AMBLPLHF		Pointer to first PLH*
16	10	4	AMBCHAIN		Reserved
20	14	4	AMBLACB		Pointer to ACB
24	18	2	AMBLPLHL		Length of PLH*
26	1A	1	AMBLPLHN		Total number of strings*
27	1B	1	AMBLFLAG		Flag byte
			AMBLPOST	X'80'	POST must be issued
28	1C	4	AMBAMBUF		Size of buffer space
32	20	2	AMBMACRF		Flags (copy of flags in ACBMACR1 and ACBMACR2)
32	20	1	AMBMACR1		*First byte*:
			AMBKEY	X'80'	Access data via index or relative record number
			AMBADD	X'40'	Access via RBA
			AMBADR	X'40'	Access via RBA
			AMBCNV	X'20'	Control interval processing
			AMBSEQ	X'10'	Sequential processing
			AMBDIR	X'08'	Direct processing
			AMBIN	X'04'	GET, READ processing
			AMBOUT	X'02'	PUT, WRITE processing
			AMBUBF	X'01'	User buffers

* When LSR is active, AMBLRPHD≠0, and AMBLSR in AMBMACR2 is set on. In this case, the fields indicated by the asterisk refer to the PLH pool.

Figure 5.38 Access Method Block List (AMBL) description and format (part 1 of 3)

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
33	21	1	AMBMACR2		*Second byte:*
			AMBLSR	X'80'	Local shared resources
			AMBDFR	X'40'	Defer writing of buffers
			AMBSKP	X'20'	Skip sequential accessing
			AMBRST	X'10'	Reusable Data Set
			AMBAIX	X'08'	AIX processing
			AMBJRACT	X'02'	JRNAD exit enabled
			AMBOPEN	X'01'	Open is in process
34	22	2	AMBLTLEN		Length of GETVIS for close work area
36	24	2	AMBDBUF		Number of data buffers
38	26	2	AMBIBUF		Number of index buffers
40	28	4	AMBLOPWA		Pointer to open work area
				Split Control	
44	2C	4	AMBSECB		Split/pseudo-split ECB
44	2C	1	AMBS0		Reserved
45	2D	1	AMBS1		Reserved
46	2E	1	AMBSCOM		ECB post byte
			AMBSWAIT	X'80'	Wait bit-share
47	2F	1	AMBSECBT		Share gate-test and set byte
48	30	4	AMBBECB		ECB for Buffer Manager
48	30	1	AMBB0		Reserved
49	31	1	AMBB1		Reserved
50	32	1	AMBBCOM		ECB Post byte
			AMBBWAIT	X'80'	Wait bit-buffer manager
51	33	1	AMBBECBT		Buffer gate-test and set byte
52	34	4	AMBLORBA		Low RBA of control area being split
56	38	4	AMBHIRBA		High RBA of control area being split
60	3C	1	AMBSTRID		ID of string which holds control area.
61	3D	1			Reserved.
62	3E	2			Reserved.
64	40	4	AMBPLH		Address of PLH in control

Figure 5-38 Access Method Block List (AMBL) description and format (part 2 of 3)

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
				Pointers	
68	44	4	AMBALIST		Address of RSCB list
72	48	4	AMBLRPLS		Address of RPL causing split
76	4C	4	AMBLCLWA		Pointer to close work area
80	50	4	AMBLCIWA		Pointer to CI split work area
84	54	4	AMBLBC		Pointer to base cluster PLH pool
88	58	4	AMBLUSB		Pointer to USB
92	5C	4	AMBBCACB		Pointer to base cluster ACB
96	60	4	AMBPEACB		Pointer to path entry ACB
100	64	4	AMBLRPHD		Pointer to resource pool header
104	68	0	AMBLEND		End of AMBL

Figure 5.38 Access Method Block List (AMBL) description and format (part 3 of 3)

Access Method Control Block (ACB)

The VSAM ACB describes a VSAM cluster. It is built by the user's program. The ACB points to the Exit List (EXLST). After the cluster is opened, the ACB is pointed to by the RPL (RPLDACB) that describes the user's record processing request. The ACB also describes the processing options, password, and I/O buffer space applicable to the user's program. The description and format of the ACB are shown in Figure 5.39.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	0	ACBST		
0	0	1	ACBID		ACB identifier = 'A0'
			ACBIDD	X'A0'	ACB equate
			ACBIDVAL	X'A0'	ACB equate
1	1	1	ACBSTYP		Release indicator
				X'00'	VSAM Release 1
				X'10'	VSAM Release 2
				X'20'	VTAM
2	2	2	ACBLEN		Length of ACB in bytes
2	2	2	ACBLENG		Length of ACB in bytes[1]
4	4	4	ACBAMBL		Address of the AMBL
8	8	4	ACBAM0		Pointer to VSAM code
12	C	1	ACBACT		ACB active byte (X'FF'=ACB is active).

[1] If specified length is too small for a VSAM Release 2 ACB, a Release 1 ACB is built (X'00' in byte 1).

Figure 5.39 Access Method Control Block (ACB) description and format (part 1 of 5)

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
13	D	1	ACBINFLG		Catalog recovery flags
			ACBSCRA	X'80'	ACB is for a system-initiated OPEN of the CRA
			ACBUCRA	X'40'	ACB is for a user-initiated OPEN of the CRA
				X'20'	Reserved for CRA
				X'10'	Reserved for CRA
			ACBSTSKP	X'08'	Skip updating of statistics
				X'04'	Reserved for CMS
				X'02'	Reserved for CMS
				X'01'	Reserved for CMS
14	E	2	ACBDBUF		Number of data buffers
14	E	2	ACBBUFND		Number of data buffers
16	10	2	ACBIBUF		Number of index buffers
16	10	2	ACBBUFNI		Number of index buffers
18	12	2	ACBMACRF		MACRF
18	12	1	ACBMACR1		MACRF first byte
			ACBKEY	X'80'	Access data via index or relative record number
			ACBADD	X'40'	Access via RBA
			ACBADR	X'40'	Access via RBA
			ACBCNV	X'20'	Control interval processing
			ACBSEQ	X'10'	Sequential processing
			ACBDIR	X'08'	Direct processing
			ACBIN	X'04'	GET
			ACBOUT	X'02'	PUT
			ACBUBF	X'01'	User buffers
19	13	1	ACBMACR2		MACRF second byte
			ACBLSR	X'80'	Local shared resources
			ACBDFR	X'40'	Defer writing of buffers
			ACBSKP	X'20'	Skip sequential access
			ACBRST	X'10'	Reusable data set
			ACBAIX	X'08'	AIX processing only
			ACBJRACT	X'02'	JRNAD exit active
20	14	1	ACBDOSID		DOS DTF identifier
			ACBDTFID	X'28'	DTF type for VSAM
21	15	1	ACBOFLGS		Open/close flags
			ACBVOLMT	X'80'	Verify volume mounted
			ACBVMSG	X'40'	Message requested bit
			ACBEOV	X'20'	EOV detects completed
			ACBOPEN	X'10'	ACB is open
			ACBCAT	X'08'	ACB for VSAM catalog
			ACBEXFG	X'04'	User exit flag
			ACBSUB	X'02'	ACB is suballocated (is located in a control block allocation unit)
			ACBKEYOK	X'01'	Key processing all right for this ACB
22	16	1	ACBNST		Number of strings
22	16	1	ACBSTRNO		Number of strings

Figure 5.39 Access Method Control Block (ACB) description and format (part 2 of 5)

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
23	17	1	ACBERFLG		Error flags
					Open error return codes:
			ACBOALR	X'04'	This ACB is already open
			ACBOLLUB	X'0E'	The symbolic unit in the DLBL statement is invalid.
			ACBONJIB	X'0F'	No job information blocks (JIBs) are available from the label information cylinder.
			ACBOLIGN	X'11'	The address in the ASSGN statement for the logical unit was IGN (assignment ignored).
			ACBOLUNA	X'12'	The address in the ASSGN statement for the logical unit was UA (logical unit unassigned).
			ACBOCEXT	X'22'	The volume serial numbers specified in the EXTENT statement do not match those specified in the catalog entry.
			ACBOCDLD	X'32'	Unable to load VSAM modules via a CDLOAD macro instruction.
			ACBONMNT	X'50'	Attempt to mount two volumes on the same drive when direct or keyed processing was specified. Or the operator failed to mount the volume.
			ACBONCRA	X'5C'	CRA volume not mounted
			ACBOIERR	X'60'	Unusable input data set
			ACBOUEMP	X'64'	Empty upgrade AIX
			ACBOTMST	X'68'	The time stamp of the volume on which a data set is stored doesn't match the system time stamp in the volume catalog entry.
			ACBOTIME	X'6C'	The system time stamps of a data set and its index do not match. This indicates that the data has been updated separately. This test is greater than or equal, i.e., no warning is given if the index time stamp is greater than the data time stamp.
			ACBOEMPT	X'6E'	Open empty data set for read only.
			ACBODSNC	X'74'	Data set was not closed the last time it was processed.
			ACBODEVT	X'75'	The symbolic unit specified in the EXTENT statements is not a valid VSAM device type.
			ACBONDLB	X'80'	The DLBL statement is missing or the filename in the DLBL doesn't match the ACB.

Figure 5.39 Access Method Control Block (ACB) description and format (part 3 of 5)

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
			ACBOIOER	X'84'	A permanent I/O error occurred while VSAM was reading label information from the label information cylinder.
			ACBONVRT	X'88'	Not enough virtual storage space is available in the partition for work areas, control blocks, or buffers.
			ACBOIOCA	X'90'	A permanent I/O error occurred while VSAM was reading or writing a catalog entry.
			ACBONCAT	X'94'	No entry was found in the catalog for this ACB.
			ACBOSECU	X'98'	Security verification failed; the password specified in the ACB for a specific level of access doesn't match the password in the catalog for that level of access.
			ACBOPARC	X'A0'	The operands specified in the ACB are inconsistent with each other or with the information in the catalog entry, for example, an open of an ESDS for keyed processing.
			ACBOKBUF	X'A1'	User-specified buffers with keyed access (user buffers can be specified only with CNV access).
			ACBOIOVL	X'A4'	A permanent I/O error occurred while VSAM was reading the volume label of the volume the data set is on.
			ACBONAVA	X'A8'	The data set is not available because it is being updated by (under the exclusive control of) another ACB or has been exported by Access Method Services.
			ACBONOCT	X'B4'	The VSAM catalog is not connected to the system on logical unit SYSCAT, or insufficient virtual storage available for OPEN.
			ACBOACT	X'BC'	ACB was active
			ACBOOERR	X'C0'	Unusable output data set
			ACBOPEMP	X'C4'	Access via empty path
			ACBODSCB	X'C8'	Format-4 label error
			ACBOCNVP	X'E0'	Invalid control interval processing
			ACBONRST	X'E8'	Non-reusable is not empty
			ACBOCTER	X'FF'	Unexpected return from catalog locate function.

Figure 5.39 Access Method Control Block (ACB) description and format (part 4 of 5)

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
					Close error return codes
				X'02'	Invalid control block, or ACB address not in OAL
			ACBCALR	X'04'	ACB already closed
			ACBCNVRT	X'88'	Insufficient space available in user's partition for work area.
			ACBCIOCA	X'90'	Permanent I/O error occurred while VSAM was reading or writing a catalog entry.
			ACBCNCAT	X'94'	No catalog entry found
			ACBCIOER	X'B8'	Permanent I/O error occurred while VSAM was completing outstanding I/O requests.
			ACBCBUSY	X'BC'	ACB busy.
24	18	4	ACBAMBUF		Length of buffer pool
28	1C	8	ACBDDNM		DDname
36	24	4	ACBPRTCT		Pointer to password
40	28	4	ACBUAPTR		Pointer to user work area, or to CAXWA if ACB is for a catalog
44	2C	4	ACBBFPL		Pointer to first data buffer in buffer pool
48	30	4	ACBEXLST		User exit list pointer
52	34	4	ACBNXT		Reserved
56	38	1			Reserved
57	39	1			Reserved
58	3A	2	ACBMSGLN		Message area length
60	3C	4	ACBMSGAR		Message area
64	40	4			Reserved
68	44	0	ACBEND		End of ACB

Figure 5.39 **Access Method Control Block (ACB) description and format (part 5 of 5)**

Access Method Control Block Structure (AMCBS)

The AMCBS is pointed to by the Anchor Table (ANCHT) and, in turn, points to the VSAM catalog's ACB and CAXWA. Figure 5.40 shows the description and format of the AMCBS.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	1	CBSID	X'00'	AMCBS identifier
1	1	1	CBSFLAGS		AMCBS flags
			CBSJCAT	X'80'	Job catalog not present
2	2	2	CBSSIZ		Length of the AMCBS
4	4	4	CBSCRACB		Pointer to CRA ACB
8	8	4	CBSACB		Pointer to ACB (Master)
12	C	4	CBSCRAPL		Pointer to AMS CRA parameter list.
16	10	4	CBSSYSUC		Pointer to job catalog ACB
20	14	4	CBSCAXCN		Pointer to CAXWA chain
24	18	4	CBSCBMM		Pointer to Control Block Manipulation Macro load module

Figure 5.40 **Access Method Control Block Structure (AMCBS) description and format**

The AMDSB contains statistical information about record processing in the data set. It also contains some of the data set's attributes and specifications. The AMDSB is built, using the data index or index catalog record's AMDSB group occurrence, when the cluster is opened. The AMBL (AMBLDTA/AMBLIX) points to the data and index AMDSBs. Figure 5.41 shows the description and format of the AMDSB.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
				General	
0	0		AMDCOMM		Common part
0	0	1	AMDSBID	X'60'	AMDSB identifier
1	1	1	AMDATTR		Attributes of the data set
			AMDATTR1		Attributes (first byte):
			AMDDST	X'80'	Key-sequenced data set
			AMDWCK	X'40'	Check each record when it is written
			AMDSDT	X'20'	Sequence set is stored with the data
			AMDREPL	X'10'	Replication
			AMDORDER	X'08'	Use the volumes in the same order as in the volume list
			AMDRANGE	X'04'	The data set is divided into key ranges
			AMDRRDS	X'02'	Relative record data set
			AMDSPAN	X'01'	Spanned records
2	2	2	AMDLEN		Length of AMDSB in the catalog
4	4	2	AMDNEST		Number of entries in an index section (in all cases except AMDSB group occurrence in data record of AIX)[1]
4	4	2	AMDAXRKP		Relative key position in base record (only for AMDSB group occurrence in data record of a AIX)[1]
6	6	2	AMDRKP		Relative key position
8	8	2	AMDKEYLN		Key length
10	A	1	AMDPCTCA		Percentage of free control intervals in the control area
11	B	1	AMDPCTCI		Percentage of free bytes in the control interval
12	C	2	AMDCIPCA		Number of control intervals in a control area
14	E	2	AMDFSCA		Number of free control intervals in a control area

Figure 5.41 **Access Method Data Statistics Block (AMDSB) description and format (part 1 of 4)**

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
16	10	4	AMDFSCI		Number of free bytes in a control interval
20	14	4	AMDCINV		Control interval size
24	18	4	AMDLRECL		Maximum record size
28	1C	4	AMDHLRBA		RBA of the high-level index record
28	1C	4	AMDNSLOT		Number of relative record slots
32	20	4	AMDSSRBA		RBA of first sequence set record
32	20	4	AMDMAXRR		Max. relative record number
36	24	4	AMDPARDB		Pointer to first ARDB
40	28	1	AMDATTR3		Attributes
			AMDUNQ	X'80' X'00'	Non-unique keys in AIX Unique keys in AIX
41	29	3			Reserved
44	2C	4			Reserved
				Statistics	
48	30		AMDSTAT		Statistics
48	30	8	AMDSTMST		System time stamp
48	30	8	AMDSTSP		System time stamp
56	38		AMDSTAT1		
56	38	2	AMDNIL		Number of index levels
58	3A	2	AMDNEDB		Number of EDBs
58	3A	2	AMDNEXT		Number of extents in the data set
60	3C	4	AMDNLR		Number of user-supplied (logical) records in the data set
64	40	4	AMDDELR		Number of deleted records
68	44	4	AMDIREC		Number of inserted records
72	48	4	AMDUPR		Number of updated records
76	4C	4	AMDRETR		Number of retrieved records
80	50	4	AMDASPA		Number of bytes of free space in the data set
84	54	4	AMDNCIS		Number of times a control interval was split
88	58	4	AMDNCAS		Number of times a control area was split
92	5C	4	AMDEXCP		Number of times EXCP was issued by VSAM I/O routines

Figure 5.41 **Access Method Data Statistics Block (AMDSB) description and format (part 2 of 4)**

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
					General Continue
96	60	1	AMDSHOPT		Share option byte
			AMDSHR1	X'80'	Share option 1
			AMDSHR2	X'40'	Share option 2
			AMDSHR3	X'20'	Share option 3
			AMDSHR4	X'10'	Share option 4
97	61	4	AMDCDSN		Pointer to catalog ACB
101	65	3	AMDDSN		Catalog control interval number for data (index)
104	68	4	AMDHWRBA		High-water RBA for the data set
108	6C	1	AMDATTR2		Attributes (second byte):
			AMDREL	X'80'	Release unused space
			AMDLOAD	X'40'	Load mode
			AMDSPEED	X'20'	Speed option
			AMDINDX	X'10'	Index option
			AMDSHR	X'08'	Sharing
			AMDKR	X'04'	Key-range processing, duplicate of AMDRANGE
			AMDCAT	X'01'	AMDSB for catalog
109	6D	1	AMDACT		AMDSB test and set byte
110	6E	2	AMDFILT		User area (ISAM compatibility)
112	70	4	AMDPVOL		Pointer to volume list
116	74	1	AMDAMS		AMS flag byte
			AMDAIX	X'80'	Alternate index
			AMDPATH	X'40'	Access via path
			AMDBASE	X'20'	Access via base
117	75	1			Reserved
118	76	2	AMDAIRKP		Relative key position in base record (only in data AMDSB of AIX)[1]
					Local Statistics
120	78		AMDLSTAT		Local statistics
120	78	2	AMDLNIL		Local number of index levels
122	7A	2	AMDLNEST		Local number of entries in the index section
124	7C	4	AMDLNLR		Local number of user-supplied (logical) records
128	80	4	AMDLDELR		Local number of deleted records
132	84	4	AMDLIREC		Local number of inserted records
136	88	4	AMDLUPR		Local number of updated records
140	8C	4	AMDLRETR		Local number of retrieved records

Figure 5.41 **Access Method Data Statistics Block (AMDSB) description and format (part 3 of 4)**

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
144	90	4	AMDLASPA		Local bytes of free space
148	94	4	AMDLNCIS		Local number of control interval splits
152	98	4	AMDLNCAS		Local number of control area splits
156	9C	4	AMDLEXCP		Local number of EXCPs issued by VSAM I/O routines
					Exception Exit
160	A0	8	AMDEXEXT		Exception exit
					Buffer Mangement Information
168	A8	2	AMDBCBNO		Number of buffers
170	AA	2	AMDBFREE		Number of unassigned buffers
172	AC	4	AMDFSBCB		Address of the first BCB (for LSR: address of the BSPH)
176	B0	4	AMDFFBCB		Address of the first free BCB
180	B4	4	AMDCCWA		Pointer to CCW build area
184	B8	8			Reserved
					EDB Header
192	C0	4	AMDFSEDB		Address of first EDB
196	C4	2			Reserved
198	C6	2	AMDLEDB		Length of EDB

[1]For more details of these fields, see the explanation of the AMDSB group occurrence.

Figure 5.41 **Access Method Data Statistics Block (AMDSB) description and format (part 4 of 4)**

Access Method Define The File (AMDTF) Table

The AMDTF table, used by the ISAM interface program, is an extension to each ISAM DTF table which a VSAM data set is associated. It contains all the information necessary to process the VSAM data set that is not contained in the DTF table. The AMDTF table is contained in preformatted form in the core image library. It is loaded by IIPOPEN the first time an OPEN is issued for a file and is completed by IIPOPEN at this time. Figure 5.42 shows the description and format of the AMDTF.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	72	SAVARPP		Used to store register contents of problem program
72	48	72	SAVARCI		ISAM interface program save area
144	90	4	ACBAD		Address of ACB
148	94	4			X'0A020000' (SVC2)
152	98	4	RPLAD		Address of RPL
156	9C		EREPL		Error exit parameter list (Valid only if ERREXT=YES is specified in DTFIS.)
156	9C	4	DTFISAD		Address of DTFIS
160	A0	4	EPLRECAD		Address of record in error (not supported by IIP)
164	A4	8	EPLDASDA		DASD address of record in error (not supported by IIP)
172	AC	1	EPLRECID		Record identification
			EPRLRECID	X'80'	Data record (VSAM data set)
			EPLXREC	X'40'	Index record (VSAM sequence set)
			EPLCXREC	X'20'	Cylinder index record (VSAM index set)
			EPLMXREC	X'10'	Master index record (VSAM index set)
			EPLREAD	X'02'	Read
			EPLWRITE	X'01'	Write
173	AD	1	EPLCMNDC	X'00'	Command code of failing CCW (not supported by IIP)
174	AE	2			Unused
176	B0		GENACB		GENCB information to generate the ACB
176	B0	4	GACBHAD		Address of header
180	B4	4	MACRFEAD		Address of MACRF element
184	B8	4	FILENEAD		Address of filename element
188	BC		GACBH		Header
188	BC	1	GACBBTC	X'A0'	Block-type code (ACB)
189	BD	1	GACBFTC	X'01'	Function-type code (GENCB)
190	BE	2	GACBNOC	X'0001'	Number of copies (1 copy)
192	C0	4	GACBWAAD		Address of work area set to 0; VSAM obtains space via GETVIS
196	C4	4	GACBWALN		Length of work area
200	C8		MACRFEL		MACRF element
200	C8	4	MACRFKTC	X'00120000'	Keyword-type code
204	CC	4	MACRFVAL		Value supplied by IIPOPEN

Figure 5.42 **Access Method Define The File (AMDTF) table description and format (part 1 of 4)**

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
208	D0		FNAMEEL		Filename (DDname) element
208	D0	4	FNAMEKTC	X'00090000'	Keyword-type code
212	D4	8	FNAMEACB		Filename (inserted by IIPO-PEN)
220	DC		GENRPL		GENCB information to generate the RPL
220	DC	4	GRPLHAD		Address of header
224	E0	4	ARLNEAD		Address of AREALEN element
228	E4	4	ACBEAD		Address of ACB element
232	E8	4	KEYLNEAD		Address of KEYLEN element
236	EC	4	RECLNEAD		Address of RECLEN element
240	F0		GRPLH		Header
240	F0	1	GRPLBTC	X'C0'	Block-type code (RPL)
241	F1	1	GRPLFTC	X'01'	Function-type code (GENCB)
242	F2	2	GRPLNOC	X'0001'	Number of copies (1 copy)
244	F4	4	GRPLWAAD		Address of work area set to 0; VSAM obtains space via GET-VIS
248	F8	4	GRPLWALN		Length of work area set to 0
252	FC		ARLNEL		AREALEN element
252	FC	4	ARLNKTC	X'002D0000'	Keyword-type code
256	100	4	ARLNVAL		Area length
260	104		ACBEL		ACB element
260	104	4	ACBKTC	X'002B0000'	Keyword-type code
264	108	4	ACBAD1		Address of ACB
268	10C		KEYLNEL		KEYLEN element
268	10C	4	KEYLNKTC	X'00300000'	Keyword-type code
272	110	4	KEYLNVAL		Key length
276	114		RECLNEL		RECORDLEN element
276	114	4	RECLNKTC	X'00350000'	Keyword-type code
280	118	4	RECLNVAL		Record length
284	11C		SHOWCB		Information to show ACB or RPL
284	11C	4	SHHAD		Address of header
288	120	4	SHEAD		Address of element
292	124		SHH		Header
292	124	1	SHBTC	X'00'	Block-type code
293	125	1	SHFTC	X'03'	Function-type code (SHOWCB)
294	126	2	SHOTC	X'0000'	Object-type code

Figure 5.42 **Access Method Define The File (AMDTF) table description and format (part 2 of 4)**

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
296	128	4	SHBAD		Address of block to be shown
300	12C	4	SHARAD		Address of area
304	130	4	SHARLN		Length of area
308	134	4	SHAR		Area where information is to be placed
312	138		SHEL		Element
312	138	4	SHKTC		Keyword-type code (set by IIP)
316	13C		MODRPL		MODCB information to modify the RPL
316	13C	4	MRPLHAD		Address of header
320	140	4	OPTCDEAD		Address of OPTCD element
324	144	4	AREAEAD		Address of AREA element
328	148	4	ARGEAD		Address of ARG element
332	14C		MRPLH		Header
332	14C	1	MRPLBTC	X'C0'	Block-type code (RPL)
333	14D	1	MRPLFTC	X'02'	Function-type code (MODCB)
334	14E	2			Unused
336	150	4	MRPLBAD		Address of block to be modified (supplied by IIPOPEN)
340	154		OPTCDEL		OPTCD element
340	154	4	OPTCDKTC	X'00340000'	Keyword-type code
344	158	4	OPTCDVAL		Bit pattern (supplied by IIP)
348	15C		AREAEL		AREA element
348	15C	4	AREAKTC	X'002C0000'	Keyword-type code
352	160	4	AREAAD		Address of area (supplied by IIP)
356	164		ARGEL		ARG element
356	164	4	ARGKTC	X'002E0000'	Keyword-type code
360	168	4	ARGAD		Address of ARG parameter (supplied by IIP)
364	16C		MSGOUT		Header
364	16C	16	MSCCB		CCB

Figure 5.42 **Access Method Define The File (AMDTF) table description and format (part 3 of 4)**

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
384	180	8	MSCCW		CCW
392	188	6	ERRCDE		Error code of message
398	18E	5	ISAMCM		'ISAM'
403	193	9	ISCM		ISAM command area
412	19C	5	VSAMCM		'VSAM'
417	1A1	9	VCCM		VSAM command area
426	1AA	4	CRCM		'RC= '
430	1AE	5	CRC1		Return code area
435	1B3	20	SHOWCBF		Area if SHOWCB failed
455	1C7	5	CRC2		Return code from SHOWCB
460	1CC	4	CRSCM		'EC= '
464	1D0	4	CRSC		Error code area
468	1D4	1	BRKT		Closing bracket

Figure 5.42 **Access Method Define The File (AMDTF) table description and format (part 4 of 4)**

Address Range Definition Block (ARDB)

The ARDB contains information about space allocated and space actually used by a data set. The block is built by the Open module from information in the data set's catalog record. The ARDB is updated by record management routines as additional space is used. The first ARDB in an ARDB chain is pointed to by the AMDSB (AMDPARDB). Figure 5.43 shows the ARDB description and format.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	1	ARDID	X'AD'	Control block identifier
1	1	1	ARDTYPE		Identifies the type of space defined by the ARDB:
			ARDKR	X'80'	One key range of a key-range data set
			ARDHLI	X'40'	The total index of a key-sequenced data set that does not have the sequence set with the data or The non-sequence set levels of a key-sequenced data set's index, when the sequence set is stored with the data
			ARDSS	X'20'	The sequence set of a key-sequenced data set, when the sequence set is stored with data

Figure 5.43 **Address Range Definition Block (ARDB) description and format (part 1 of 2)**

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
			ARDUOVFL	X'10'	Use overflow volumes for this key
			ARDEOD	X'08'	End of data ARDB
			ARDLGCC	X'04'	Device contains more than 256 cylinders
2	2	2	ARDLEN		Length of the ARDB
4	4	1	ARDPRF		Address range definition preformat byte (this byte is a literal copy of the catalog record byte called ITYPEXT)
			ARDPRFMT	X'80'	Sequence set with data
				X'40'	No preformat done
5	5	3			Reserved
8	8	4	ARDNPTR		Address of the next ARDB in the ARDB chain
12	C	4	ARDHRBA		The RBA of the next free-space control interval at the end of the data set (RBA of VSAM SEOF)
16	10	4	ARDEDBA		Pointer to the active EDB
20	14	4	ARDPREL		Pointer to related ARDB (index)
24	18	4	ARDERBA		The RBA of the highest control interval allocated to the key range
28	1C	4	ARDPKEYS		Pointer to ARDKEYS
32	20	4	ARDHKRBA		The RBA of the data set control interval containing the key range's high-key value
36	24	2	ARDVOLNM		Number of volumes in list
The following ten-byte entry, called an ARDB volume group, repeats for each volume in this ARDB.					
38	26	10	ARDVOLGP		Volume serial (VOLSER) list
38	26	6	ARDVOLSR		The serial number of the volume containing the highest RBA allocated to the key range
44	2C	2	ARDRELRP		Catalog relative replication number
46	2E	2	ARDSYMU		Symbolic unit
46	2E	1	ARDSUCLS		Symbolic unit class
47	2F	1	ARDSUNUM		Symbolic unit number
Variable (after last volume group)		Variable	ARDKEYS		Space reserved for the key range's low and high key values. The length of this field equals twice the key length

Figure 5.43 **Address Range Definition Block (ARDB) description and format (part 2 of 2)**

BLDVRP Parameter List (VRPPL)

The BLDVRP parameter list contains all parameters needed by module IKQBRP to build the VSAM resource pool. The address of the parameter list is held in register 1.

Figure 5.44 shows the format and description of the parameter list.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	4	VRPBFLST		Address of buffer list
4	4	1	VRPKEYLN		Maximum keylength in VSAM Resource Pool
5	5	1	VRPSTRNO		String number of VSAM Resource Pool
6	6	6			Reserved
		VRP Buffer List			
The following 8 bytes are repeated for each subpool specified.					
12	C	4	VRPBFSZE		Size of buffers in subpool
16	10	1	VRPBFIND		Indicator byte
			VRPBLEND	X'80'	End of buffer list
17	11	1			Reserved
18	12	2	VRPBFCNT		Number of buffers in subpool

Figure 5.44 BLDVRP parameter list description and format

Buffer Control Block (BCB)

The BCB consists of a buffer control entry that describes each buffer requested by the user and each buffer required for preformat processing. Each buffer control entry contains function codes, status indicators, and RBAs to describe the buffer. The buffer control entry also contains the address of the data buffer, the associated channel program built in the CCWAREA, and the next BCB in the chain. The buffer control entry is created by Open and released by Close. The BCB is the interface between the I/O Manager and the Buffer Manager modules. The BCB is pointed to by the PLH (PLHDBCB points to the data BCB and PHLXBCB points to the index BCB). Figure 5.45 shows the BCB description and format.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	4	BUFNBCB BUFCHAIN		Address of the next BCB entry Offset to chain pointer (equate value)
4	4	4	BUFCBAD		Buffer address
8	8	20	BUFRIODR		Read I/O driver block
8	8	2	BUFCURRU		Read symbolic unit number
10	A	2	BUFBKSTR		Number of physical blocks to read
12	C	8	BUFRSEEK		Computed DASD address for read
12	C	1	BUFRM		M (Contains X'80' for an RPS device)
			BUFRDSK		Offset to read MBBCCHHR (equate value)
13	D	2	BUFRBB		BB
15	F	2	BUFRCC		CC
17	11	2	BUFRHH		HH
19	13	1	BUFRR		R
20	14	4	BUFCRRBA		RBA for the read
24	18	4	BUFRLPMB		Address of the read LPMB
28	1C	20	BUFWIODR		Write I/O driver block
28	1C	2	BUFCURWU		Write symbolic unit number
30	1E	10	BUFCKIN		Write initialize area
30	1E	2	BUFBKSTW		Number of physical blocks to write
32	20	8	BUFWSEEK		Computed DASD address for write
32	20	1	BUFWM		M (Contains X'80' for an RPS device)
			BUFWTSK		Offset to write MBBCCHHR (equate value)
33	21	2	BUFWBB		BB
35	23	2	BUFWCC		CC
37	25	2	BUFWHH		HH
39	27	1	BUFWR		R

Figure 5.45 **Buffer Control Block (BCB) description and format (part 1 of 3)**

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
40	28	4	BUFCWRBA		RBA for the write
44	2C	4	BUFWLPMB		Address of the write LPMB
48	30	2	BUFFLAG		Flag bytes
48	30	1	BUFFLAG1		Flag byte 1
			BUFALLF1	X'FF'	BUFIOS + BUFRDAHD + BUFCVAL + BUFSSRCD + BUFRES1
			BUFIOS	X'E8'	BUFCMW + BUFCFMT + BUFCRRD + BUFPFMT
			BUFCMW	X'80'	Write indicator
			BUFCFMT	X'40'	Format writer indicator
			BUFCRRD	X'20'	Read indicator
			BUFRDAHD	X'10'	Read ahead request
			BUFPFMT	X'08'	Format remainder of control area
			BUFCVAL	X'04'	Buffer contents are valid
			BUFSSRCD	X'02'	Buffer is a sequence set record
			BUFRES1	X'01'	Reserved
49	31	1	BUFFLAG2		Flag byte 2
			BUFALL2	X'FF'	BUFPURG1 + BUFPURG2 + BUFRIXRD + BUFWRINV + BUFFREP + BUFRES2
			BUFPURG1	X'80'	Purge - must write or read
			BUFPURG2	X'40'	Purge - format
			BUFRIXRD	X'20'	Replicated index read
			BUFWRINV	X'10'	Control interval was written - another string
			BUFFREP	X'08'	Return buffer by REPBUF
			BUFRES2	X'07'	Reserved
50	32	10	BUFBKTWI		Write check initialize area
50	32	2	BUFBKTCK		Number of physical blocks to check
52	34	8	BUFWCKSK		Computed DASD address for check
52	34	1	BUFCM		M (Contains X'80' for an RPS device)
53	35	2	BUFCBB		BB
55	37	2	BUFCCC		CC
57	39	2	BUFCHH		HH
59	3B	1	BUFCR		R

Figure 5.45 Buffer Control Block (BCB) description and format (part 2 of 3)

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
60	3C	4	BUFVCCHH		CCHH for replicated index read
60	3C	4	BUFVCCB		CCB address
64	40	1	BUFERFLG		I/O error indicator
			BUFERAL1	X'FC'	BUFEIOER + BUFESRCH + BUFESEEK + BUFEREAD + BUFEWRT + BUFERBCK
			BUFEIOER	X'80'	I/O error on this buffer
			BUFESRCH	X'40'	I/O error on search ID
			BUFESEEK	X'20'	I/O error on seek
			BUFEREAD	X'10'	I/O error on read
			BUFEWRT	X'08'	I/O error on write
			BUFERBCK	X'04'	I/O error on readback check
			BUFENTCM	X'02'	Buffer operation not complete
			BUFBDSK	X'01'	2314 seek incorrect
65	41	1	BUFSTRID		String ID of this set of buffers
66	42	2	BUFCNOI		No. of blocks in control interval to process
68	44	4	BUFNABCB		Next BCB in AMDSB chain
			BCB Extension for Local Shared Resources		
72	48	4	BUFMDBTS		Modification mask (one bit per transaction - refer to BSPH)
76	4C	4	BUFUCHUP		Address of previous BCB in chain
80	50	4	BUFUCHDN		Address of next BCB in chain
84	54	4	BUFBSPH		Address of BSPH
88	58	2			Reserved
90	5A	1	BUFFLAG3		Reserved
91	5B	1	BUFUSE		Buffer use byte (X'FF'=in use)
92	5C	7	BUFHDSID		Catalog ACB address (4 bytes) and CI number (3 bytes) of the catalog record for this index or data component
99	63	1			Reserved
100	64	4	BUFAMDSB		Pointer to AMDSB
104	68	4	BUFACB		Pointer to the ACB

Figure 5.45 **Buffer Control (BCB) description and format (part 3 of 3**

Block Pool Header (BKPHD)

The BKPHD describes the storage allocation for the CCW build area. It points to the FCDB and is pointed to by the AMDSB. It is built by IKQOPN and can be extended by IKQIOA. Figure 5.46 shows the BKPHD format.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	2	BKPLENG		Length of the pool of blocks
2	2	2			Available
4	4		BKPHDECB		Control allocation of blocks
4	4	2			Not used
6	6	1	BKPHDCOM		Communications byte
			BKPHWAIT	X'80'	Wait flag
7	7	1	BKPHDTS		Allocation test and set byte
8	8	32	BKPHRSAV		Space for saving registers during steal BCB
8	8	4	BKPHRS13		Save register 13, (original PLH)
12	C	4	BKPHRS14		Save register 14
16	10	4			Save register 15
20	14	4	BKPHRS00		Save register 0
24	18	4	BKPHRS01		Save register 1
28	1C	4			Save register 2
32	20	4			Save register 3
36	24	4			Save register 4
40	28	4	BKPHDBHD		Save data buffer header during steal BCB
44	2C	4	BKPHIBHD		Save index buffer header during steal BCB
48	30	4	BKPSPCHN		Address of next area of blocks
52	34	4	BKPERCCB		Address of error CCB (first error CCB in VSAM error queue)
56	38	4	BKPFSTBK		Address of first available block
60	3C	4	BKPSTECB		ECB - steal BCB, other string
60	3C	2			Available
62	3E	1	BKPSTCOM		Communications byte
			BKPSWAIT	X'80'	Wait flag
63	3F	1	BKPSTTS		Test and set byte
			Equate values		
			BKPSIZE	X'800'	Size of space to extend pool (2048)
			BKPBLKSZ	X'40'	Size of a block (64)
			BKPNOBKS	X'20'	Number of blocks in new space (32)

Figure 5.46 **Block Pool Header (BKPHD) description and format**

The BHD contains information about buffers and buffer processing. The PLH points to the data and index BHDs. The BHD is created by Open and released by Close. Figure 5.47 shows the description and format of the BHD.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	2	BHDNO		Number of buffers
2	2	2	BHDLEN		Length of control block
4	4	2	BHDRMAX		Maximum number of buffers available
6	6	2	BHDRMIN		Minimum number of buffers available
8	8	2	BHDBRC		Read-ahead count
10	A	1	BHDHFLAG		Buffer header flag 1
			BHDRAHOK	X'80'	Read-ahead OK flag
			BHDIXREP	X'40'	Replicated index read indicator
			BHDNSKD	X'08'	I/O with wait for no-schedule queue (BCBNSKDQ)
			BHDSKD	X'04'	I/O with wait for schedule queue (BCBSKDQ)
			BHDMVBCB	X'02'	'Free buffer' is really a move
11	B	1	BHDFLAG		Buffer header flag 2 (reserved)
12	C	4			Reserved
16	10	4	BHD1STF		Address of chain of free buffers
20	14	4	BHDSKDQ		Address of BCB chain with I/O scheduled
24	18	4	BHDNSKDQ		Address of BCB chain with pending I/O
18	1C	4	BHD1STW		Address of first BCB requiring I/O
32	20	1	BHDID	X'77'	BHD idendification
33	21	1			Reserved
34	22	2	BHDIOCNT		I/O count of no-schedule queue (BCBNSKDQ)
36	24	2	BHDWMIN		Write threshold
38	26	2	BHDTRACT		Temporary read-ahead count
40	28	2	BHDQNO		Number of BCBs on queues
42	2A	2			Reserved
44	2C	4	BHDCCHH		CCHH of last held control area
48	30	4	BHDCCBCH		CCB chain pointer

Figure 5.47 Buffer Header (BHD) description and format

Buffer Subpool Header (BSPH)

The Buffer Sub-Pool Header contains information concerning a buffer subpool. The BSPHs are chained together in a sequence of ascending buffer sizes. AMDFSBCB points to the BSPH associated with a particular data set component. The address of the first BSPH is stored in the VSRT.

Figure 5.48 shows the format and description.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	1	BSPHID	X'72'	Control block identifier
1	1	1			Reserved
2	2	2	BSPHLEN		Length of BSPH
4	4	4	BSPHNM		Name 'BSPH'
8	8	4	BSPHNBSP		Pointer to next BSPH in pool
12	C	2	BSPHBFNO		Number of buffers in this subpool
14	E	2			Reserved
16	10	4	BSPHMDBN		Number of modified buffers in this subpool
20	14	4	BSPHFRBN		Number of free buffers in this subpool
24	18	4	BSPHBCB		Address of first BCB in the subpool
28	1C	4	BSPHMDBT		32-bit modification mask. Each bit corresponds to a transaction which has modified the buffer
32	20	4	BSPHBSZ		Length of each buffer in this subpool
36	24	4	BSPHCPLH		Address of the PLH currently in control of the BSPH
40	28	16			Reserved*
56	38	4	BSPHUTOP		Pointer to the top of the use chain
60	3C	4	BSPHUBTM		Pointer to the bottom of the use chain*
64	40	4	BSPH1ST		Address of the first BSPH in the buffer pool
68	44	2	BSPHECB		Control bytes for changing use chain
70	46	1	BSPHCOM BSPHWAIT	 X'80'	Communications byte Wait flag
71	47	1	BSPHTS		Mask byte for test and set
*	The use chain is a chain of all BCBs in the subpool. The least recently used BCB is at the bottom of the chain and the most recently used BCB is at the top.				

Figure 5.48 Buffer Subpool Header description and format

The CAXWA is built when the VSAM catalog is opened or is being created. The CAXWA is used to contain the addresses of control blocks and work areas needed when a catalog is being processed. The CAXWA also contains flags that indicate the type of processing being performed on the catalog and the DOS/VS component that invoked the processing. The CAXWA is pointed to by the ACB (ACBUAPTR). The AMCBS (CBSCAXCN) contains the address of the CAXWA chain. Figure 5.49 shows the CAXWA description and format.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
0	0	1	CAXID	Control Block identifier X"CA"
1	1	3		Reserved
4	4	4	CAXCHN	Address of the next CAXWA in the chain
8	8	1	CAXFLGS	Flags:
		1...	CAXBLD	Build request
		.1..	CAXOPN	The catalog is being opened
		..1.	CAXCLS	The catalog is being closed
		...1	CAXEOV	An end-of-volume rountine is in control
		.... 1...	CAXCMP	Open/Close/EOV processing is complete
		.... .x..	CAXMCT	1=Master catalog 0=User catalog
		.... ..1.	CAXCMR	Catalog management has been called by a catalog management routine
		x	CAXSCR	Reserved for OS
9	9	1	CAXFLG2	Flags:
		1...	CAXF2DT	The catalog has been deleted
		.1..	CAXF2NDD	No DLBL filename found
		..x.	CAXF2CCR	0 = CCR needs to be read 1 = CCR has been read
		...1	CAXF2CRA	CAXWA for CRA
		.... 1...	CAXF2REC	Recoverable catalog
		.... .1..	CAXF2EOV	End of volume flag
		.... ..x.		Reserved for OS
		1	CAXF2CA	Free CAXWA if error
10	A	1		Reserved
11	B	1	CAXACT	Catalog activity count
12	C	4	CAXATIOT	Reserved for OS
16	10	4	CAXSCHWA	Reserved for OS
20	14	4	CAXDRWP	Address of the catalog's DSCB read work area

Figure 5.49 Catalog Auxiliary Work Area (CAXWA) description and format (part 1 of 2)

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
24	18	4	CAXACB	Address of the catalog's ACB
24	18	4	CAXCRACB	Address of CRA's ACB
28	1C	4	CAXUCB	Address of the COMREG
32	20	12	CAXCCR	Catalog control record information
32	20	3	CAXHACI	Catalog interval number of the highest allocated control interval in the catalog
35	23	3	CAXNFCI	Control interval number of the next free control interval in the catalog
38	26	3	CAXCDCI	Number of deleted control intervals
41	29	3	CAXFDCI	Control interval number of the first deleted control interval in the catalog
44	2C	2		Reserved
46	2E	2	CAXRPLCT	Number of RPLs associated with the CAXWA
48	30	4	CAXRPL	Address of the first RPL in the CAXWA's RPL chain
52	34	44	CAXCNAM	Catalog name
96	60	4	CAXOPLST	Open/Close parameter list:
96	60	1 1... .... .xxx xxxx	COPTS CENLST	Option flags: End-of-list indicator Reserved
97	61	3	COPACB	Address of the catalog's ACB
100	64	4	CAXOPEWA	Address of Open/Close/EOV work area
104	68	4	CAXCCA	Address of the CCA
108	6C	4	CAXJDE	Reserved for OS
112	70	4	CAXCAT	Address of the catalog's ACB associated with CRA
115	74	6	CAXVOLCR	Volume serial of CRA volume
112	7A	2	CAXSYSCR	SYS-number of CRA volume
124	7C	6	CAXVOLRM	Volume serial of volume containing CRA (at present not mounted)
130	82	2	CAXSYSRM	SYS-number of volume containing CRA (at present not mounted)
132	84	6	CAXOCPAR	O/C parameter list
132	84	4	CAXOCACB	ACB address
136	88	2	CAXOCEOL	End of list indicator (x'0A02')

Figure 5.49 **Catalog Auxiliary Work Area (CAXWA) description and format (part 2 of 2)**

The CCA is built each time a DOS/VS component issues the CATLG macro instruction to process a VSAM catalog record. The CCA contains information about the catalog being processed and about the catalog record and its extensions contained in each of the six buffers available to process the user's request. The CCA is used to pass information between catalog management procedures. Register 11 contains the address of the CCA. Figure 5.50 shows the CCA description and format.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
0	0	2	CCAID	Identifier - set to X'ACCA'
2	2	2	CCASZ	Size
4	4	1	CCACD1	Return code 1
5	5	1	CCACD2	Return code 2
6	6	1	CCAFLG1	Flag byte 1:
		1...	CCAF1LPS	Stop the loop
		.1..	CCAF1ARA	Assign RPL to auxiliary record area
		..1.	CCAF1LRD	Catalog control record read into virtual storage
		...1	CCAF1KEY	Retrieve the catalog record based on a DSNAME value (GET)
		...0		Retrieve by CI number
		.... 1...	CCAF1KGE	Retrieve the next catalog record (next GET)
		.... .1..	CCAF1CR	A checkpoint of the CCR required
		.... ..1.	CCAF1UP	GET macro instruction issued for update
		1	CCAF1DK	When the caller is renaming a data set, this flag indicates that the data set's true-name record is to be deleted, but the data set's catalog record is not to be deleted.
7	7	1	CCAFLG2	Flag byte 2:
		1...	CCAF2SYS	Reserved for OS
		.1..	CCAF2NVC	No validity check on the caller's CTGFL or work area is required
		..1.	CCAF2CCT	Single catalog search
		..0.		Not used by DOS/VS
		...1	CCAF2XEQ	Exclusive enqueue
		...0		Shared enqueue
		.... 1...	CCAF2RHS	When a catalog management routine calles the VSAM Open routines to open a newly created catalog, and the Open routines call VSAM Catalog Management routines to obtain information about the catalog to be opened, the situation is called a "recursive call". The catalog cannot be dequeued when the Catalog Management routines return to the caller (VSAM Open routines).
		.... .xx.	CCAF2COB	Combination of catalog open and build:
		.... .1..	CCAF2CO	Catalog is being opened
		.... ..1.	CCAF2CB	Catalog open during build
		1	CCAF2SMO	Reserved for OS

Figure 5.50 Catalog Communication Area (CCA) description and format (part 1 of 12)

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
8	8	1	CCAFLG3	Flag byte 3:
		1...	CCAEXGR1	Exit indicator
		.1..	CCAGC4	The catalog record contains a password group occurrence (identified by Group Code 4) (detected during IGGPSCNC processing)
		..1.	CCAGDSP	GENDSP
		...1	CCAEXGR2	Exit indicator
		.... 1...	CCANF	The group occurrence cannot be found
		.... .1..	CCAELC2	Exit indicator
		.... ..1.	CCALFT	First time
		1	CCAEGREC	Exit indicator
9	9	1	CCAFLG4	Flag byte 4:
		1...	CCAF4DRQ	The catalog must be dequeued after the request completes
		.1..	CCAF4BYS	Bypass the security verification
		..1.	CCAGVNC	The required variable-length field is not completely contained in the record currently in the buffer
		...1	CCAGVNF	The group occurrence identified by the caller-specified sequence number cannot be found
		.... 1...	CCAGVNBS	There is no buffer space available to contain an extension record
		.... .1..	CCAGVEX	Exit indicator
		.... ..1.	CCAGVNE	The field does not exist in the located group occurrence
		1	CCATCOMP	Test complete: all group occurrence pointers have been examined and all designated fields have been tested
10	A	1	CCAFLG5	Flag byte 5:
		1...	CCAMEX2	Exit indicator
		.1..	CCAMEX	Exit indicator
		..1.	CCAMEX1	Exit indicator
		...1	CCAMODPA	The catalog record's base record must be written (using IGGPPAD) into the catalog
		.... 1...	CCATHIT	Successful test: a group occurrence has been found that satisfies the test conditions
		.... .1..	CCATEX	Exit indicator
		.... ..1.	CCATEX1	Exit indicator
		1	CCATEX2	Exit indicator

Figure 5.50 **Catalog Communication Area (CCA) description and format (part 2 of 12)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
11	B	1	CCAFLG6	Flag byte 6:
		1...	CCAMCODR	The catalog must be dequeued when the request completes
		.1..	CCADELP	A deleted group occurrence pointer was found
		..1.	CCAMNOSP	The catalog record's free space isn't large enough to contain all the new catalog information during the group occurrence move operation
		...1	CCAINIT	Insert switch for variable-length field being retrieved
		.... 1...	CCASUPFD	Suppress password field information during field retrieval
		.... .1..	CCAREUSE	The contents of the caller's record areas (buffers) can be used by IGGPEXT and IGGPMOD
		.... ..1.	CCAEXT	Set when a catalog management routine calls the Extract routine (IGGPEXT)
		1	CCAMOD	Set when a catalog management routine calls the Modify routine (IGGPMOD)
12	C	4	CCALAB	Address of the label cylinder area
12	C	1	CCALBLEN	Count field in units of 128 bytes
13	D	3	CCALBCYL	Address field
16	10	4	CCADPL	Address of the DADSM parameter list
			CCARB	Reserved for OS
20	14	4	CCACPL	Address of the caller's CTGPL
24	18	4	CCAACB	Address of catalog's ACB
28	1C	4	CCANPCCB	Address of saved CAXWA
32	20	4	CCAURAB	Address of the record area block (RAB) currently in use
36	24	44	CCASRCH	Search argument (DSNAME of cluster, data, index, catalog, or nonVSAM data set, or a volume serial number)
36	24	3	CCASRID	Control interval number
36	24	3	CCASRCIN	Control interval number
39	27	41		Reserved (or continuation of CCASRCH)
80	50	20	CCARAB0	Record Area Block 0: Each record area block describes the catalog record contained in one of the six catalog management buffers available for the request. RABs 1 through 5 are identical in format to RAB 0.
				Note: "x" in each field name is replaced by "0" through "5" to indicate a particular RAB's field.

Figure 5.50 **Catalog Communications Area (CCA) description and format (part 3 of 12)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
80	50	1	CCARxFLG	Flags: The first three flags are used by IGGPEXT and IGGPMOD:
		1...	CCARxUR	The RAB is in use. It cannot be used by IGGPEXT or IGGPMOD
		.1..	CCARxU1	The RAB is temporarily in use by IGGPEXT or IGGPMOD. It cannot be overlaid.
		..1.	CCARxU2	(Same as CCARxU1)
		...1	CCARxWR	The buffer must be written before another catalog record can be read into it.
		.... 1...	CCARxPA	The buffer contains a new catalog record; PUT ADD is required to add the record to the catalog
		.... .xx.		Reserved
		1	CCARxUPD	Update buffer not reused
81	51	1	CCARxRPL	Last assign, RPL index
82	52	2		Reserved
84	54	4	CCARxREC	Address of the record in the buffer
84	54	4	CCACPE1x	Address of the record in the buffer
88	58	12	CCARxSEG	Addresses of the segments
88	58	4	CCACPE2x	Address of the first byte after the fixed-length header fields
92	5C	4	CCACPE3x	Address of the first group occurrence
96	60	4	CCACPE4x	Address of the first free-space byte in the record
100	64	20	CCARAB1	Record Area Block 1 (See RAB 0 description)
120	78	20	CCARAB2	Record Area Block 2 (See RAB 0 description)
140	8C	20	CCARAB3	Record Area Block 3 (See RAB 0 description)
160	A0	20	CCARAB4	Record Area Block 4 (See RAB 0 description)
180	B4	20	CCARAB5	Record Area Block 5 (See RAB 0 description)
200	C8	1	CCARPLK	Assigned RPL count
201	C9	1	CCARPLF	Index to RPL found

Figure 5.50 Catalog Communications Area (CCA) description and format (part 4 of 12)

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
202	CA	1	LPINDX	Loop indexing control (counter)
202	CA	1	CCARPLX	Work byte for catalog RPL (mult.-use)
202	CA	1	XIOOPT	I/O options
		1...	XIOGET	1=GET, 0=PUT
		.1..	XIOERS	ERASE
		..1.	XIOARA	1=auxiliary record area required 0=user record area required
		...1	XIOKEY	1=keyed required 0=address required
		.... 1...	XIONUP	No update required
		.... .1..	XIONCK	No error check required
		.... ..1.	XIOTNE	1=true name entry 0=normal entry
		1	XIOKGE	GET NEXT (GET)
		1	XIOSEQ	PUTSEQ (PUT)
203	CB	1	CCARPLT	Work byte for catalog RPL (mult.-use)
204	CC	6	CCARPLAA	Indices to assigned RPLs
210	D2	2		Reserved
212	D4	4	CCARPL1	Address of the RPL in use

Figure 5.50 **Catalog Communications Area (CCA) description and format (part 5 of 12)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
216	D8	132	CCADESA	Save area for the extent information returned by VSAM, DADSM and Catalog Management: Suballocate
216	D8	1	CCANDEXT	Number of extents
217	D9	1	CCAIXEXT	Extent index value
218	DA	2	CCASSVOL	Sequence number of the data set directory entry in the volume catalog record
220	DC	128	CCAEXTDE	Sixteen 8-byte extent descriptors:
			First extent descriptor	
220	DC	2	CCAEXTSS	Sequence number of the Data Space group occurrence that this extent's space is a part of
222	DE	4	CCAEXTAD	The extent's starting physical address:
222	DE	2	CCAEXTCC	Cylinder number CC
224	E0	2	CCAEXTHH	Head number HH
226	E2	2	CCAEXTTH	Number of tracks in the extent
228	E4	120		Space for remaining 15 extent descriptors
348	15C	1	CCAASCIK	Number of control intervals required to satisfy the caller's request
349	15D	1	CCACRRP	X'80' Build "caller" chain for message 4223I
			EWFLGS	Equate
			EWFCCHN	Equate
350	15E	1	CCAASCIX	Used by the ASSGN functions - points to the element in CCAASCI currently being processed
351	15F	1	CCASRPLX	Saved RPL flags
352	160	9	CCAASCI	Number of each assigned control interval
361	169	3	CCAUPGD	Control interval for UPG modification
364	16C	16	CCAEQDQ	Enqueue/Dequeue parameter list
364	16C	1	CCAEDXFF	End of parameter list, indicator byte =X'FF' (if list is empty)
365	16D	1	CCAEDRLN	Length of minor name
366	16E	1	CCAEDOPT	Enqueue/Dequeue Options
		x...	CCAEDSHR	1=Shared, 0=Exclusive
		.1..	CCARLSEB	Release control bit
		..xx xxxx		Other options (set by macro)
367	16F	1	CCAEDRCD	Enqueue/Dequeue return code
368	170	4	CCARTSAV	Save area for CCAMLRET
372	174	4	CCACOMRG	COMRG pointer
376	178	4	CCAEDUCB	Work area
380	17C	4	CCAMLRET	Address of the caller's save area used by IGG0CLAG

Figure 5.50 **Catalog Communications Area (CCA) description and format (part 6 of 12)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
384	180	12	CCAMSSPL	GETVIS/FREEVIS parameter list area
384	180	4	CCAMNLEN	Number of bytes to process
388	184	4	CCAMNPTR	Address of the return address
392	188	1		Reserved for OS
393	189	1	CCAMNSPL	Reserved for OS
394	18A	2		Reserved for OS
396	18C	4	CCARPRM	Return parameters
400	190	8	CCACMS	Catalog Management Services work area
400	190	4	CCACMSWA	Address of the CMS calling routine's work area
404	194	4	CCAEXCMS	Address of a secondary CMS work area
The following fields are set and used by IGGPLOC, IGGPEXT, and IGGPTSTS, and catalog management subfunctions which these procedure call:				
408	198	0	CCALUME	
408	198	4	CCACPE5	Address of a selected group occurrence pointer
412	19C	4	CCACPE51	(Same as CCACPE5)
416	1A0	4	CCACPE52	(Same as CCACPE5)
420	1A4	4	CCACPE53	(Same as CCACPE5)
424	1A8	4	CCACPE6	Address of a selected group occurrence
428	1AC	4	CCACPE61	(Same as CCAPE6)
			CCARABSE	Save extract caller URAB
432	1B0	4	CCACPE7	Address of field value
			CCAIDPT	Insert data address
436	1B4	4	CCACPE71	Alternate address to field value
440	1B8	2	CCAGOPLN	Length of the group occurrence pointer
442	1BA	2	CCASL	Length of sequence number field (RELREPNO) in the group occurrence
444	1BC	4	CCAILNG	Length of the selected retrieved field
448	1C0	4	CCAFLPT CCATFLPT	Address of the requested-field CTGFL Address of the CTGFL-for-tests
452	1C4	4	CCARABPT	Address of the record area block
456	1C8	4	CCADICT	Dictionary information to describe the field, based on its field name
460	1CC	4	CCAXCPL CCAMCPL	Address of the CTGPL built when IGGPEXT and IGGPMOD are called, so that information in the caller's CTGPL is not altered
464	1D0	4	CCARABB	Address of the RAB that identifies the base catalog record
468	1D4	4	CCARABF	Address of the RAB that identifies the first record area (buffer) that can be used by IGGPEXT or IGGPMOD

Figure 5.50 Catalog Communications Area (CCA) description and format (part 7 of 12)

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
472	1D8	4	CCARABL	Address of the RAB that identifies the last record area (buffer) that can be used by IGGPEXT or IGGPMOD
476	1DC	3	CCACBASE	The control interval number of the base catalog record
479	1DF	1	CCAGC	Group code of the requested group occurrence
480	1E0	2	CCALREL CCALREL1	Relative repetition number of a selected group occurrence
482	1E2	2	CCASN CCASN1	Sequence number of a selected group occurrence
484	1E4	1	CCAFLG8	CRA flags
		1...	CCARPUT	Inhibit CRA PUT/UPDATE
		.1..	CCALSTC	Listcat request
		..1.	CCAEXTCR	Extend CRA in process
		...1	CCABLDCR	Open request for CRA build
		.... 1...	CCASPUCO	Special UCAT
		.... .x..	CCASCAX	1=CRA CAXWA search, 0=UCAT CAXWA search
		.... ..x.	CCAUPG	1=upgrade, 0=no upgrade
		1	CCABUF	Output buffering flag
485	1E5	1	CCAFLGA	More flags
		1...	CCAUPGRR	RAB1 to be restored by upgrade module
		.1..	CCARGET	Get record for compare before update CRA
		..1.	CCALBFVT	Multiple file parameter search at define
		...1	CCACCARD	Indicate CCR for CRA has been reached
		.... 1...	CCAF1LSV	Save indicator flag CCAF1LRD
		.... .1..	CCACANIN	Cancel INHIBIT
		xx		Reserved
486	1E6	2	CCAIXFPL	Index to the current CTGFL being processed
488	1E8	2	CCAIXREL	Index for CCATREL
490	1EA	2	CCATNREL	The sequence number of the next group occurrence to perform tests against if CCATREL is full or if there are no buffers available to contain the catalog record's next extension
492	1EC	2	CCATNUM	Number of successful relative repetition numbers (cannot exceed 16)
494	1EE	32	CCATREL	Successful relative repetition numbers
526	20E	2	CCATNO	Total number of successful relative repetition numbers (might exceed 16)
528	210	4	CCATEST	Address of the test CTGFL

Figure 5.50 **Catalog Communications Area (CCA) description and format (part 8 of 12)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
532	214	20	CCARBA	Work area for extent descriptors
532	214	2	CCASS	Sequence number of the Data Space group occurrence that contains the extent
534	216	4	CCACCHH1	Physical address - CCHH - of the extent's first track
538	21A	4	CCACCHH2	Physical address - CCHH - of the extent's last track
542	21E	2	CCATT	Number of tracks in the extent
544	220	4	CCARBA1	Low relative byte address (RBA)
548	224	4	CCARBA2	High relative byte address (RBA)
552	228	2	CCATLNG CCATLEN	Total length of the extent information that has been processed (CCATLNG); total length of the scanned field so far (CCATLEN)
554	22A	2	CCARBAL	RBA extent balance
556	22C	2	CCACNIX	Combination name index
558	22E	2	CCAREASN	Reason code
560	230	4	CCAIDPT2	Address of the available space in the caller's work area or of the caller-supplied update information
564	234	4	CCAIDPT3 CCARABSM	Address of the length-field of a variable length field in the user's return area
568	238	2	CCAGVCT	Number of group occurrence pointers processed so far
570	23A	2	CCANEVV	If the requested variable-length field is non-existent, this field is set to binary zero
572	23C	3	CCAGVEXT	Control interval number of the record's next extension record (not yet in a buffer)
575	23F	1	CCANEFV	If the requested fixed-length field is non-existent, this byte is set to X"FF"
576	240	1		Reserved
577	241	1	CCAGRGC	Group code of the requested group occurrence
578	242	2	CCAGRHI CCAGRHI1	High relative repetition number of the requested group occurrence
580	244	2	CCAIXTPL	Index to test FPL
582	246	2	CCADLEN	Number of bytes to be deleted from the catalog record
584	248	2	CCADIFF	The difference between the insert length and the delete length (can be a negative number)

Figure 5.50 **Catalog Communications Area (CCA) description and format (part 9 of 12)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
586	24A	2	CCAREPCT	Number of relative repetition numbers processed so far
588	24C	2	CCADISP	Displacement into variable-length field to the delete/insert location
590	24E	3	CCASVCI	Save area for the control interval number of the base catalog record
593	251	3	CCASVCI1	Save area for the control interval number
596	254	4	CCADTA	Address of the dictionary
600	258	4	CCACDTA	Address of the index combination table
604	25C	2	CCADTCT	Number of dictionary entries
606	25E	2	CCACDTCT	Number of index combination entries
608	260	4	CCACWAP	Controller work area
612	264	4	CCAMNADR	Address of the virtual storage obtained by a GETVIS request
616	268	4	CCAILNG3	Save area for the insertion length
620	26C	4	CCAILNG2	Length of the user-supplied insert data
624	270	4	CCAALPTR	Address of the space management work area
628	274	4		Reserved
632	278	4	CCALCPL	Reserved for OS
636	27C	1	CCAFLG7	Flags:
		x...	CCALSP	Reserved for OS
		.1..	CCANRLSE	Release Control Bit
		..1.	CCACKDEL	Delete switch
		...1	CCASMFBR	Do GET for base record
		.... 1...	CCAONCE	Move only one occurrence
		.... .1..	CCAROREQ	Read only request
		.... ..1.	CCAFEOV	Force EOV
		1	CCAEQOPN	Enqueued on SYSOPEN
637	27D	3	CCARCI	CRA Record control interval number
640	280	4	CCALABSV	Saved address of IKQLAB area
644	284	4	CCARABSV	Saved address of RAB
648	288	2	CCAMODUL	Last two bytes of module name (IGG0CLxx)
650	28A	3	CCACHAIN	Control interval number save area
653	28D	3	CCACI1	Control interval number save area
656	290	3	CCACI2	(Same as CCACI1)
659	293	3	CCACI3	(Same as CCACI1)
662	296	2	CCAVARLN	Number of bytes to be inserted into the record
664	298	4	CCARRAB	Address of the RAB containing the group occurrence pointers where delete/insert processing is to begin
668	29C	4	CCARBASE	(Same as CCARRAB)

Figure 5.50 Catalog Communications Area (CCA) description and format (part 10 of 12)

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
672	2A0	4	CCAVARPT	Address of the information to be inserted into the record
676	2A4	2	CCADELN	Number of bytes to be deleted from the record
678	2A6	20	CCAVAR	Insert information save area
698	2BA	20	CCAVAR1	(Same as CCAVAR)
718	2CE	3	CCADEL1	The control interval number of the first record in a series of records to be deleted
721	2D1	3	CCADEL2	The control interval number of the last record in a series of records to be deleted
724	2D4	40	CCAXLATE	Translation work area
764	2FC	4	CCAR14S	Register 14 save area
768	300	0	CCABMINP	Input parameters to IGG0CLBR
768	300	2	CCABMTRK	Starting track
770	302	2	CCABMLIM	Check limit, nn for set
772	304	2	CCABMMIN	Conditional check minimum
774	306	1	CCABMFLG	State and function code
		x...	CCABMST	This bit can be 0 or 1, and is the state for which an extent is to be checked (if bit 1 is on) or the state to which a map bit is to be set (if bit 2 is on)
		.1..	CCABMCHK	ON - Perform check
		..1.	CCABMSET	ON - Perform set
		...1	CCABMCCK	ON - Perform condition check
		.... 1...	CCABMLST	ON - Last set request (write)
		xxx		Reserved
775	307	1		Reserved
776	308	0	CCABMOUT	Output parameters from IGG0CLBR
776	308	2	CCABMONN	Track number
778	30A	2	CCABMOTR	Starting track
780	30C	1	CCABMOFG	Output flags
		1...	CCABMOST	State of bits
		.xxx xxxx		Reserved
781	30D	6	CCAVOLCR	CRA volume identification
787	313	1	CCABMPAD	Padding character
788	314	4	CCABMGOP	Current bit mask GOP
792	318	4	CCABMPTR	Address of current bit mask byte
796	31C	4	CCABMEND	End of current bit mask
800	320	2	CCABMBT1	Bit count, first byte
802	322	2	CCABMBTL	Bit count, last byte
804	324	2	CCABMBYT	Number of full bytes
806	326	2	CCABMSTR	Current bit mask, start track
808	328	4	CCABMWK1	Work field
812	32C	4	CCABMWK2	Work field

Figure 5.50 **Catalog Communications Area (CCA) description and format (part 11 of 12)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
816	330	4	CCABMWK3	Work field
820	334	4	CCABMWK4	Work field
824	338	4	CCABMRB1	Address of first bit map RAB
828	33C	4	CCABMRB2	Address of second bit map RAB
832	340	4	CCACARWA	Address of CRA definition work area
836	344	4	CCACRABF	Address of CRA buffer
840	348	4	CCASACB	Address of saved CCAACB field
844	34C	4	CCAEXC	Save area for CCAACB
848	350	4	CCASRPL	Address of saved CCA, RPL field
852	354	4	CCAADBUF	Address of cluster record buffer (cluster record saved until CRA volume known)
856	358	4	CCASCAXS	Address of search argument for CAXWA chain search
860	35C	4	CCASCAXA	Address of found CAXWA
864	360	4	CCADEVT	CRA volume device type
868	364	8	CCANMF1	Name field of variable open resource
876	36C	8	CCANMF2	Name field of variable open resource
884	374	8	CCANMF3	Name field of variable open resource
The following two fields are used by the no-upgrade/upgrade function, called by ALTER, DEFINE or DELETE				
892	37C	3	CCAXDCI	AIX data control interval number
895	37F	3	CCAXICI	AIX index control interval number
898	382	1	CCACATIN	CLAH indicator
899	383	1		Reserved
900	384	4	CCACOPTR	CLCO work area
904	388	4	CCADEVA	Address of device attribute return area
908	38C	4	CCAFARE	Address of file identification
912	390	4	CCAAREA	Pointer to address of label record area
916	394	2	CCAMDSAV	Save area for CCA module
918	396	2	CCARSSAV	Save area for CCA
920	398	40	CCATEMPS	Temporary area for PLS
960	3C0	348	CCAREGS	Save area for registers
960	3C0	4		Address of user save area
964	3C4	8	CCAMODNM	Load module name
1308		0	CCAEND	End CCA

Figure 5.50 Catalog Communications Area (CCA) description and format (part 12 of 12)

Record management uses its own CCB macro to map a FDCB into a CCB. Figure 5.51 shows the CCB format.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0		CCBST		
0	0		CCBD		
0	0	16	CCBLEN		Map of DOS/VS CCB
0	0	2	CCBCNT		Residual count
2	2	4	CCBERMAP		Error codes
2	2	1	CCBCOM1		Communication byte 1
			CCBWAIT	X'80'	Traffic switch (set at channel end)
			CCBEOF	X'40'	End of File
			CCBIOERR	X'20'	Unrecoverable I/O error
			CCBERROK	X'10'	Accept unrecoverable I/O error
			CCBRDC	X'08'	Return data checks
			CCBPDE	X'04'	Post at device end
			CCBDCV	X'02'	Return data check read/check
			CCBUERR	X'01'	User error routine
3	3	1	CCBCOM2		Communication byte 2
			CCBDCCNT	X'80'	Data check in count field
			CCBTRKOV	X'40'	Track overrun
			CCBEOC	X'20'	End of cylinder
			CCBDC	X'10'	Data check
			CCBNOREC	X'08'	No record found
			CCBRETRY	X'04'	Retry no record found
			CCBVER	X'02'	Verify error
			CCBCC	X'01'	Command chain (retry)
4	4	1	CCBCSW1		CSW status byte 1
			CCBATTN	X'80'	Attention
			CCBSTMOD	X'40'	Status modifier
			CCBCUE	X'20'	Control unit end
			CCBBUSY	X'10'	Busy
			CCBCE	X'08'	Channel end
			CCBDE	X'04'	Device end
			CCBUC	X'02'	Unit check
			CCBUE	X'01'	Unit exception

Figure 5.51 Command Control Block (CCB) description and format (part 1 of 2)

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
5	5	1	CCBCSW2		CSW status byte 2
			CCBPCI	X'80'	Program-controlled interrupt
			CCBILEN	X'40'	Incorrect length
			CCBPROGM	X'20'	Program check
			CCBPROT	X'10'	Protection check
			CCBCHAND	X'08'	Channel data check
			CCBCHANC	X'04'	Channel control check
			CCBICTRL	X'02'	Interface control check
			CCBCHAIN	X'01'	Chaining check
6	6	2	CCBSYMU		Symbolic unit
6	6	1	CCBSUCLS		U - LUB class
7	7	1	CCBSUNUM		N - LUB number within class
8	8	1	CCBLIOCS		Reserved for LIOCS
9	9	3	CCBCCW		Address of channel program
12	C	1	CCBCOM3		Communication byte 3
			CCBAPEND	X'40'	Appendage end at interrupt
13	D	3	CCBCSW		Address of CSW in appendage routine
16	10	4	CCBDATB		Address of last data block
20	14	4	CCBLCCWB		Address of last CCW block
24	18	4			Reserved
28	1C	1	CCBUFLGS		I/O manager CCB flags
			CCBUEAIC	X'80'	Error analysis in control
			CCBUEAC	X'40'	Error analysis complete
			CCBURDCW	X'20'	Read CCW active
			CCBRPS	X'10'	RPS channel program candidate
29	1D	3	CCBFSCCW		Save area for first CCW address
32	20	4	CCBRDCCW		Save area for first read CCW
36	24	4	CCBWTCCW		Save area for first write CCW
40	28	4	CCBLWCCW		Save area for last write CCW
44	2C	12			Reserved
56	38	4	CCBNCCB		Address of next CCB block
60	3C	4			Reserved

Figure 5.51 Command Control Block (CCB) description and format (part 2 of 2)

Record management uses the CCW macro to map a CCW slot within an FCDB for building CCW strings. Figure 5.52 shows the CCW description and format.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	8	CCWAREA		Length of one CCW
0	0	1	CCWOP		Operation code
1	1	3	CCWARG		Argument address
4	4	1	CCWFLAG		CCW flags
			CCWDCH	X'80'	Data chaining
			CCWCCH	X'40'	Command chaining
			CCWSLI	X'20'	Suppress incorrect length
			CCWSKIP	X'10'	Skip data transfer
5	5	1			Reserved
6	6	2	CCWCNT		Count field
8	8	1	CCWOP1		Next CCW operation code
			CCWSRHE	X'31'	Search ID equal
			CCWSSEC	X'23'	Set sector
			CCWWTCKD	X'1D'	Write count key data
			CCWSKHD	X'1B'	Seek head
			CCWRDC	X'12'	Read count
			CCWTIC	X'08'	TIC
			CCWSEEK	X'07'	Full seek
			CCWRD	X'06'	Read data
			CCWWT	X'05'	Write data
			CCWNOP	X'03'	NOP

Figure 5.52 **Channel Command Word (CCW) description and format**

CCW Skeleton DSECT (CWS)

The I/O manager (IKQIOA) uses the CWS DSECT to map the CCW skeletons used for building CCW strings. Figure 5.53 shows the CWS description and format.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	1	CWSOFSET		Offset of CCWs together
1	1	1	CWSFLAG		CCW string flag byte 1
			CWSPLHAD	X'80'	PLH address function
			CWSIVLR	X'40'	Invalidate R function
			CWSBFADC	X'20'	Buffer address and count function
			CWSARGAD	X'10'	Argument address function
			CWSASTER	X'08'	CCW address function
			CWSINCR	X'04'	Increment R function
			CWSDECR	X'02'	Decrement R function
			CWSNOOPT	X'01'	No optimization function
2	2	1	CWSFLAGC		CCW string flag byte 2
			CCWSRPS	X'80'	RPS function
3	3	8	CWSCCW		CCW to build
		Equate Value			
			CWSLENTH	X'0B'	One CCW string argument length

Figure 5.53 CCW Skeleton DSECT (CWS) description and format

These are the I/O Manager (IKQIOA) CCW skeletons that CWS maps. They are used for building CCW strings. Figure 5.54 shows the CCW Skeletons description and format.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
		Head Seek CCW			
0	0	1	SCCWSK	X'08'	Length of contiguous CCWs (FCBCINC)
1	1	1		X'51'	CCW build flags (CWSIVLR + CWSARGAD + CWSNOOPT)
2	2	1		X'00'	CCW build flags
3	3	1		X'1B'	CCW opcode (CCWSKHD)
4	4	3		X'01'	Offset into ARG address
7	7	1		X'40'	Command chain flag (CCWCCH)
8	8	1		X'00'	Reserved
9	9	2		X'0006'	Count field
11	B	1		X'00'	End of chain indicator
		Search CCW			
12	C	1	SCCWCRH	X'10'	Length of continuous CCWs (2 x FCBCINC)
13	D	1		X'10'	CCW build flags (CWSARGAD)
14	E	1		X'00'	CCW build flags
15	F	1		X'31'	CCW op code (CCWSRHE)
16	10	3		X'000003'	Offset into ARG address
19	13	1		X'40'	CC flag
20	14	1		X'00'	Reserved
21	15	2		X'0005'	Count field
23	17	1		X'08'	Length of contiguous CCWs (FCBCINC)
24	18	1		X'08'	CCW build flags (CWSASTER)
25	19	1		X'00'	CCW build flags
26	1A	1		X'07'	CCW op code minus carry (CCWTIC - 1)

Figure 5.54 CCW Skeletons description and format (part 1 of 5)

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
27	1B	3		X'FFFFF8'	Offset in CCW chain from here (minus 8)
30	1E	4		4X'00'	Reserved
34	22	1		X'00'	End of chain
		Format Write CCW and Write CCW			
35	23	1	SCCWFMTW	X'08'	Length of contiguous CCWs (FCBCINC)
36	24	1		X'13'	CCW build flags (CWSARGAD + CWSDECR + CWSNOOPT)
37	25	1		X'00'	CCW build flags
38	26	1		X'1D'	CCW op code (CCWWTCKD)
39	27	3		X'000003'	Offset into ARG address
42	2A	1		X'80'	Data chain flag (CCWDCH)
43	2B	1		X'00'	Reserved
44	2C	2		X'0008'	Count field
46	2E	1		X'08'	Length of continuous CCWs (FCBCINC)
47	2F	1		X'24'	CCW build flags (CWSBFADC + CWSINCR)
48	30	1		X'00'	CCW build flags
49	31	1		X'05'	CCW op code (CCWWT)
50	32	3		X'000000'	Offset into buffer address
53	35	1		X'40'	Command chain flag (CCWCCH)
54	36	1		X'00'	Reserved
55	37	2		X'0000'	Count field
57	39	1		X'00'	End of chain

Figure 5.54 **CCW Skeletons description and format (part 2 of 5)**

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
		Write CCW			
58	3A	1	SCCWWT	X'08'	Length of contiguous CCWs (FCBCINC)
59	3B	1		X'64'	CCW build flags (CWSBFADC + CWSINCR + CWSIVLR)
60	3C	1		X'00'	CCW build flags
61	3D	1		X'05'	CCW op code (CCWWT)
62	3E	3		X'000000'	Offset into buffer address
65	41	1		X'40'	Command chain flag (CCWCCH)
66	42	1		X'00'	Reserved
67	43	2		X'0000'	Count field
69	45	1		X'00'	End of chain
		Index Read CCW and Read CCW			
70	46	1	SCCWIXRD	X'08'	Length of contiguous CCWs (FCBCINC)
71	47	1		X'80'	CCW build flags (CWSPLHAD)
72	48	1		X'00'	CCW build flags
73	49	1		X'12'	CCW op code (CCWRDC)
74	4A	3		X'000000'	Offset into PLH address
77	4D	1		X'60'	SLI and command chain flag (CCWCCH + CCWSLI)
78	4E	1		X'00'	Reserved
79	4F	2		X'0004'	Count field
81	51	1	SCCWRD	X'08'	Length of contiguous CCWs (FCBCINC)
82	52	1		X'24'	CCW build flags (CWSBFADC + CWSINCR)
83	53	1		X'00'	CCW build flags
84	54	1		X'06'	CCW op code (CCWRD)

Figure 5.54 CCW Skeletons description and format (part 3 of 5)

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
85	55	3		X'000000'	Offset into buffer address
88	58	1		X'40'	Command chain flag (CCWCCH)
89	59	1		X'00'	Reserved
90	5A	2		X'0000'	Count field
92	5C	1		X'00'	End of chain
		Read Back Check CCW			
93	5D	1	SCCWRBCK	X'08'	Length of contiguous CCWs (FCBCINC)
94	5E	1		X'24'	CCW build flags (CWSBFADC + CWSINCR)
95	5F	1		X'00'	CCW build flags
96	60	1		X'06'	CCW op code (CCWRD)
97	61	3		X'000000'	Offset into buffer address
100	64	1		X'50'	Skip and command chain flags (CCWCCH + CCWSKIP)
101	65	1		X'00'	Reserved
102	66	2		X'0000'	Count field
104	68	1		X'00'	End of chain
		RPS Seek Head and Set Sector CCW			
105	69	1	SCCWRPS	X'08'	Length of contiguous CCWs (FCBCINC)
106	6A	1		X'51'	CCW build flags (CWSIVLR + CWSARGAD + CWSNOOPT)
107	6B	1		X'00'	CCW build flags
108	6C	1		X'1B'	CCW op code (CCWSKHD)
109	6D	3		X'000001'	Offset into ARG address
112	70	1		X'40'	Command chain flag (CCWCCH)
113	71	1		X'00'	Reserved

Figure 5.54 **CCW Skeletons description and format (part 4 of 5)**

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
114	72	2		X'0006'	Count field
116	74	1		X'08'	Length of contiguous CCWs (FCBCINC)
117	75	1		X'00'	CCW build flag
118	76	1		X'80'	CCW build flag (CCWSRPS)
119	77	1		X'23'	CCW op code (CCWSSEC)
120	78	3		X'00000B'	Offset into ARG address
123	7B	1		X'40'	Command chain flag (CCWCCH)
124	7C	1		X'00'	Reserved
125	7D	2		X'0001'	Count field
127	7F	1		X'00'	End of chain

Figure 5.54 CCW Skeletons description and format (part 5 of 5)

The Close Work Area is built when a VSAM data set is opened. It contains work area data for alternate index processing and save areas for close registers and catalog registers. It is pointed to by CLWAAD, displacement 112 (X'70'), in the Open Work Area.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
0	0		WACOMMON	Common Open/Close work area
0	0	1	WAFLAG	Flag byte:
		1...	TCLOSE	Work area for TCLOSE
		.1..	CLOSE	Work area for Close
		..1.	OPEN	Work area for Open
		...1	OPAMDINX	Index AMDSB is being processed
		.... 1...	VOLFOUND	Volume serial number is in label cylinder record
		.... .1..	SSFLAG	Sequence set with data
		.... ..1.	RETRY	Catalog should be reupdated by Close
		1	FILEPROT	DOS Supervisor DASD file protect
1	1	1	WAERCODE	Error condition code
2	2	2	WALEN	Length of GETVIS area
4	4	4	WAPIBSV	Address of USERSAVE field
8	8	4	WALISTP	Address of user ACB/DTF list
12	C	2	WACOMR	Address of DOS communication region
14	E	1	EDBCODE	One GETVIS obtains enough space for 3 EDBs; this field is used to count EDBs
15	F	1		Reserved
16	10	4	CATEXTPT	Pointer to extent information in order to build EDBs
20	14	2	CATEXTLN	Length of total extents
22	16	2	EXTNUMB	Number of extents
24	18	80	USERSAVE	Room to save user jobname, PSW, and registers (from partition save area)
104	68	0	WACOMEND	End of common work area

Figure 5.55 Close Work Area description and format (part 1 of 3)

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
104	68	0	WORKAREA	Close work area
104	68	8	TIME	Time used to update catalog
112	70	4	RETREG1	Return address of savearea 1
116	74	4	RETREG2	Return address of savearea 2
120	78	0	PARM	Parameter list to be passed to IKQLASMD
120	78	1	CALLERID	Caller ID
121	79	0	DSID	Data set ID
121	79	3	DSCI	CI number
124	7C	4	CTACBPTR	Address of catalog ACB
128	80	1	SHAREOPT	Share option from catalog
129	81	1		Reserved
130	82	2	OUTCNT	Number of output users returned from IKQLASMD
132	84	1	BITBANK	Close flags
		1...	ENQACT	USE macro was issued
		.1..	ENQOPN	Enqueue on SYSOPEN
		..1.	UPDOPN	Need to update OPEN indicator
		...x xxxx		Reserved
133	85	1	SET0FLG	Byte of zero for resetting OPEN indicator
134	86	2		Reserved
136	88	0	CLAIXWA	AIX work area
136	88	4	ACBREQ	Address of ACB for which CLOSE/TCLOSE is requested
140	8C	4	ACBBASE	Address of base cluster ACB
144	90	4	USBADDR	Address of USB

Figure 5.55 **Close Work Area description and format (part 2 of 3)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
148	94	4	ACBCURR	Address of ACB currently being processed
152	98	4	RPLCURR	Address of header RPL
156	9C	1	AIXATTR	AIX attribute of current ACB
		1...	AIXUPGRD	Member of upgrade set but not path entry
		.1..	AIXBASE	Base ACB
		..1.	AIXENTRY	Path entry ACB
		...x xxxx		Reserved
157	9D	1	AIXPROC	Processing status
		1...	UPGD2	First member of upgrade set has been processed
		.xxx xxxx		Reserved
158	9E	2	CLMSGFLG	IKQOCMSG flags
160	A0	8	FTAB	Resource name field for protection of file tab
168	A8	8	CAT	Resource name field for protection of file tab
176	B0	4	USBCURR	Address of current USB ACB
180	B4	72	REGSAVE	Close register savearea
252	FC	72	CATSAVE	Catalog register savearea
324	144	0	CATDATA	Catalog data
324	144	4	CATLSTP	Address of catalog list
328	148	2	CATLSTSZ	Catalog list size
330	14A	2	CATWASIZ	Catalog work area size
332	14C	4	CATWAPTR	Address of catalog work area
336	150	0	CATWA	Catalog work area
336	150	52	DUMMYRPL	Dummy RPL
388	184	16	DUMMY234	Save area for registers 2 - 5
404	194	540	DUMMYPLH	Dummy PLH for LSR
944	3B0	64	DUMLSRA	LSR savearea
1008	3F0	52	DUMDBHD	Dummy data buffer header
1060	424	52	DUMIBHD	Dummy index buffer header

Figure 5.55 **Close Work Area description and format (part 3 of 3)**

Control Interval Work Area (CIW)

The CIW describes a control interval split workarea. It contains workareas for all routines that are activated during a control interval pseudo split, control interval split or a control area split. It is created by IKQCIS00 whenever a split occurs. It points to the data buffer needed in case of a split and is pointed to by the AMBL (AMBLCIWA). The space is acquired as needed by GETVIS. At completion of CI-split processing, it is freed via FREEVIS, and AMBLCIWA is set to zeros. Figure 5.56 shows the description and format of the CIW.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
		Register Save Area for IKQCIS		
0	0	48	CIWAVE	Register save area (12 Reg.)
0	0	4	CIWAVR14	Register 14
4	4	4	CIWAVR15	Register 15
8	8	4	CIWAVR0	Register 0
12	C	4	CIWAVR1	Register 1, RDF shift count on entry
16	10	4	CIWAVR2	Register 2, RDF modification offset
20	14	4	CIWAVR3	Register 3, RDF data work area
24	18	24		Reserved
48	30	4	CIWLNGTH	Length of work area
		Space Manager Save Area		
52	34	4	CIWSPA14	Register 14
56	38	4	CIWSPA15	Register 15
60	3C	4	CIWSPA03	Register 3
		IKQPFO Work Area		
64	40	4	CIWPFO14	Register 14
68	44	4	CIWPFO00	Register 0
72	48	4	CIWPFO01	Register 1
76	4C	4	CIWPFO02	Register 2
80	50	4	CIWPFO03	Register 3
84	54	4	CIWPFO04	Register 4
88	58	4	CIWACB	ACB pointer for TCLOSE call
92	5C	2	CIWSVC	SVC2 in TCLOSE call list
94	5E	2		Unused
		IKQRRP Work Area		
The work area for IKQRRP overlays the work area for IKQPFO				
64	40	4	CIWRRP14	Register 14
68	44	4	CIWRRP00	Register 0
72	48	4	CIWRRP01	Register 1
76	4C	4	CIWRRP02	Register 2
80	50	4	CIWRRP03	Register 3
84	54	4	CIWRRBA	Beginning of RBA in extent
88	58	4	CIWRRPLN	Preformat length
92	5C	2	CIWRSEOF	SEOF indicator
94	5E	2		Unused

Figure 5.56 Control Interval Work Area (CIW) description and format (part 1 of 5)

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
		IKQNCA Work Area		
96	60	4	CIWNEW14	Register 14
100	64	4	CIWNEW01	Register 1
104	68	4	CIWNEW03	Register 3
108	6C	4	CIWCARBA	Low RBA of data control area (new control area)
112	70	4	CIWCIRBA	Index RBA of old sequence set record
116	74	4	CIWNXRBA	Index RBA of new sequence set record
120	78	4	CIWDARDB	Data ARDB
		IKQCAS Work Area		
124	7C	4	CIWCAS14	Register 14
128	80	4	CIWCAS03	Register 3
132	84	4	CIWHINEW	High section of new control area
136	88	4	CIWSPTR	Pointer save section
140	8C	4	CIWHIOLD	High section of old control area
144	90	4	CIWEPTR	Entry pointer
148	94	4	CIWAKEY	Address of key save area
152	98	2	CIWEINC	Entry increment bytes
154	9A	2	CIWSRR	Offset of last section from the high section of the new control area
156	9C	4	CIWXBUFA	Address of new index buffer
		IKQCIR Work Area **Control Interval Space Reclamation Work Area**		
The work area for IKQCIR overlays the work areas for IKQNCA and IKQCAS				
96	60	4	CIWCIR14	Register 14
100	64	4	CIWCIR09	Register 9
104	68	4	CIWCIR03	Register 3
108	6C	4	CIWSAVP	Free data of pointer save for control interval
112	70	1	CIWCIRSW	Switch byte
			CIWNEXT	X'80' Position to next entry index
			CIWSPAN	X'40' Spanned entry index
			CIWRECL	X'20' Space reclamation index
			CIWNOSPL	X'10' No control area split indicator
			CIWXWRT	X'08' Write index indicator
113	71	3		Reserved
116	74	0	CIWLASMD	IKQLASMD parameter list
116	74	1	CIWLID	Request type
			CIWLTST	X'04' Test request
117	75	0	CIWLDSID	Data set identification
117	75	3	CIWLDSCI	Control interval number
120	78	4	CIWLACB	Pointer to catalog ACB

Figure 5.56 **Control Interval Work Area (CIW) description and format (part 2 of 5)**

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
124	7C	1	CIWLSOPT		Share option
125	7D	1	CIWLFLG		Flag byte
			CIWLIN	X'80'	Input indicator
126	7E	2	CIWLOUT		Output count
128	80	8	CIWRES		Resource name field
136	88	24			Unused
		IKQCIS Work Area			
160	A0	32	CIWCIWA		Copy of PLH work area
192	C0	4	CIWRCDCT		Record count save for move
196	C4	4	CIWMODPT		Pointer to modification point
200	C8	4	CIWFPTR		Pointer to next record to be moved
204	CC	4	CIWFRDF		Pointer to RDF of the next record
208	D0	4	CIWTCIL		Total data length of control interval
212	D4	4	CIWCLNUP		RBA of control interval requires an update
216	D8	4	CIWDCRDB		Save of current ARDB pointer
220	DC	4	CIWNIRBA		RBA of new sequence set
225	E0	2	CIWOLDCT		Save of RDF count
226	E2	1	CIWFLAGS		Flags
			CIWNTWO	X'80'	Two control intervals are needed for this split
			CIWNCAS	X'40'	Control area split needed to continue
			CIWCASDN	X'20'	Control area split has been executed
			CIWUHKR	X'10'	ARDHKRBA requires update
			CIWCLN	X'08'	Control intervals written require clearing
			CIWCIR	X'04'	Space reclamation executed
227	E3	1			Unused
		IKQIXE Entry Stack			
228	E4	0	CIWENTRY		Index entry data stack
228	E4	0	CIWENT1		First stack position
228	E4	4	CIWRBA1		RBA to be put in entry
232	E8	4	CIWKADD1		Address of key
236	EC	2	CIWKL1		Length of key
238	EE	1	CIWFLG1		Flag byte
			CIWENTOK	X'81'	These two bits are used to indicate that this entry is valid
			CIWINC	X'40'	Index record in core
			CIWSPLIT	X'20'	Split entry to be done
			CIWNOIO	X'10'	No execution of input/output yet (I/O is required)

Figure 5.56 **Control Interval Work Area (CIW) description and format (part 3 of 5)**

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
239	EF	1	CIWIXLV1		Index level
240	F0	0	CIWENT2		Second stack position (Used to hold contents of stack 1 when stack 1 is needed for further processing.)
240	F0	4	CIWRBA2		RBA
244	F4	4	CIWKADD2		Key pointer
248	F8	2	CIWKL2		Key length
250	FA	1	CIWFLG2		Same as CIWFLG1
251	FB	1	CIWIXLV2		Index level
252	FC	0	CIWSTKND		End of stack
252	FC	4	CIWEKEYA		Address of index enter key
		Scratch Buffer Parameter List			
256	100	0	CIWDCNV		Scratch CI descriptor
256	100	4	CIWDRBA		Scratch control interval RBA
260	104	0	CIWDBUF		Buffer parameter list
260	104	4	CIWDBCB		Address of control block
264	108	4	CIWDBAD		Address of buffers
268	10C	0	CIWDCIDF		CIDF descriptor
268	10C	2	CIWDFSO		Free space offset
270	10E	2	CIWDFSL		Free space length
272	110	1	CIWDSW		Switch byte
273	111	1			Reserved
274	112	2	CIWDCSZ		Length of buffer - 10
		IKQIXE Work Area			
276	114	4	CIWIXEBA		Caller base save
280	118	4	CIWIXERT		Return register save
284	11C	4	CIWIXER0		Save GETVIS length
288	120	4	CIWIXER1		Save GETVIS address
		Work Area for Linkage from IKQCIS to IKQCAS			
292	124	4	CIWCILST		CI list for multi-string CA split
296	128	4	CIWCISR8		Register save for linkage return
		AMDSB Save Area for Updates to AMDSB Control Fields			
300	12C	4	CSXHLRBA		AMDHLRBA index
304	130	2	CSXNIL		AMDNIL index
306	132	2			Unused
		IXFORMAT Work Space			
308	134	4	CIWIXFBA		Save callers base
312	138	4	CIWIXFRT		Save return register
316	13C	4	CIWLSEP		Entry pointer for last section
320	140	4	CIWANLSE		Entry address for last section
324	144	4	CIWANLE		Last entry address

Figure 5.56 **Control Interval Work Area (CIW) description and format (part 4 of 5)**

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
328	148	2	CIWKEYL		Length of current key
330	14A	2	CIWNLSEL		Length of last section key
332	14C	2	CIWNLEL		Length of last entry key
334	14E	2			Unused
336	150	4	CIWXNSA		Address of next section
340	154	4	CIWXSOP		Offset pointer of last section
344	158	2	CIWFCNT		Format count
346	15A	2	CIWCINL		Control entry length
348	15C	44	CIWAREA		Work area for RDF build
392	188	*	CIWKEY		Index key work area

* Length = 5 x keylength

Figure 5.56 Control Interval Work Area (CIW) description and format (part 5 of 5)

Catalog Parameter List (CTGPL)

The CTGPL is built before a DOS/VS component issues the CATLG macro instruction to process a VSAM catalog record. The CTGPL defines the catalog management request and its options, the catalog record to be processed, and the VSAM catalog that contains the record. The CTGPL is pointed to by register 1. When the catalog management routines build a CCA to support the request, the address of the CTGPL is put into the CCA (CCACPL). Figure 5.57 shows the CTGPL description and format.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
0	0	1	CTGOPTN1	First option indicator:
		1...	CTGBYPSS	Bypass the catalog management security verification processing
		.1..	CTGMAST	Check the master password
		..1.	CTGCI	Check the control interval password
		...1	CTGUPD	Check the update password
		.... 1...	CTGREAD	Check the read password
		.... .1..	CTGNAME	The CTGENT field contains the address of a 44-byte DSNAME, or a 6-byte volume serial number (padded with binary zeros)
		.... .0..		The CTGENT field contains the address of a 3-byte control interval number
		....1.	CTGCNAME	The CTGCAT field contains the address of a 44-byte catalog DSNAME
		....0.		The CTGCAT field contains the address of a VSAM catalog's ACB
		x		Reserved

Figure 5.57 Catalog Parameter List (CTGPL) description and format (part 1 of 3)

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
1	1	1	CTGOPTN2	Second option indicator:
		1...	CTGEXT	Extend option (with UPDATE)
			CTGERASE	Erase option (with DELETE)
		.1..	CTGREL	Release (with UPDATE) Release secondary extents when data set is opened as a reusable data set
		..1. ...	CTGPURG	Purge option (with DELETE)
			CTGVMNT	Volume mount caller
		...1	CTGGTNXT	Get-next option (with LISTCAT)
		.... 1...	CTGDISC	Disconnect option (with DELETE)
		.... .1..	CTGOVRID	Erase override option (with DELETE)
		.... ..1.	CTGSCR	Scratch space option (with DELETE)
		x		Reserved
2	2	1	CTGOPTN3	Third option indicator:
		xxx.	CTGFUNC	Specifies the caller-requested function:
		001.	CTGLOC	LOCATE
		010.	CTGLSP	Reserved for OS
		011.	CTGUPDAT	UPDATE
		100.	CTGCMS	A Catalog Management Services function (see CTGOPTNS)
		...1	CTGMCE	Master catalog exists
		.... x...		Reserved
		.... .x..	CTGSRH	Reserved
		.... ..x.	CTGNUM	Reserved
		1	CTGAM0	VSAM request
		0		Non-VSAM request
3	3	1	CTGOPTN4	Reserved for OS
4	4	4	CTGENT	User entry address (address of volume in the case of OS)
			CTGFVT	Address of callers CTGFV (DEFINE, ALTER)
8	8	4	CTGCAT	Address of the catalogs DSNAME or ACB, as specified in CTGOPTN1
12	C	4	CTGWKA	Address of the callers work area
16	10	1	CTGOPTNS	CMS options:
		0000 1...	CTGDEFIN	DEFINE
		0001 0...	CTGALTER	ALTER
		0001 1...	CTGDELET	DELETE
		0010 0...	CTGLTCAT	LISTCAT
		xxx		Reserved
17	11	1	CTGCRFLG	CRA open flags
		1...	CTGLBCYL	Label cylinder information is passed for CRA
		.1..	CTGCTRBL	Control blocks are passed for CRA
		..xx xxxx		Reserved

Figure 5.57 **Catalog Parameter List (CTGPL) description and format (part 2 of 3)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
18	12	1	CTGTYPE	Type of catalog record
		C'D'	CTGTDATA	Data
		C'I'	CTGTINDX	Index
		C'A'	CTGTALIN	Non-VSAM
		C'U'	CTGTUCAT	User catalog
		C'V'	CTGTVOL	Volume
		C'C'	CTGTCL	Cluster
		C'M'	CTGTMCAT	Master catalog
		C'G'	CTGTAIX	Alternate index
		C'R'	CTGTPTH	Path
		C'Y'	CTGTUPG	Upgrade set
		C'F'	CTGTFREE	Free record
19	13	1	CTGNOFLD	Number of entries contained in CTGFIELD
20	14	4	CTGDDNM	Address of DD-name
			CTGNEWNM	Address of the new DSNAME, if the request is ALTER and the object's name is being changed
20	14	2	CTGFDBK	Module name feedback (last two characters of module name)
22	16	2	CTGFBFLG	DOS reason code feedback
24	18	4	CTGJSCB	Reserved for OS
			CTGPSWD	Address of the caller supplied password
28	1C	4	CTGDDUC	Address of UCAT file name
32	20	4	CTGDDCR	Address of CRA file name
36	24	4	CTGFIELD	First field pointer
40	28	4	CTGFLD2	Second field pointer
44	2C	4	CTGFLD3	Third field pointer

Figure 5.57 Catalog Parameter List (CTGPL) description and format (part 3 of 3)

DADSM Parameter List

The DADSM parameter list contains the information required by the DASD Space Management modules. Its address is held in register 1.

Figure 5.58 shows its format and description.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	6	DADVOLID		Volume identifier
6	6	2	DADSYSLN		Number of system LUB
8	8	2	DADSFLAG		Processing flags
					First byte:
			DADRF1	X'80'	Read format 1 label
			DADFR4	X'40'	Read format 4 label
			DADRADDR	X'20'	Address for read
			DADBYPS	X'10'	Bypass volume 1 checking
			DADNGV	X'08'	No GETVIS required
			DADLKLBK	X'04'	IKQLAB has been called
			DADSPROT	X'02'	Scratch/rename protected files
			DADSVTAD	X'01'	VTOC address is valid
					Second byte:
			DADSVTOP	X'80'	VTOC open indicator
			DADSMSG	X'40'	Message flag
			DADSKPF4	X'20'	DADSM caller is DADSM
			DADBYEXT	X'10'	Extents bypassed
			DADSCNCL	X'08'	Operator reply was cancel
10	A	1	DADSRC		DADSM return code
11	B	1	DADSARC		Return code save area
12	C	4	DADSLADD		Address of label record
16	10	4	DADSAREA		Address of I/O area
20	14	44	DADEXIST		Old data set name
20	14	44	DADSRDSN		Returned data set name
64	40	44	DADCREAT		New data set name
108	6C	4	DADSEXIT		DADSM exit address
112	70	24	DADSWORK		Work area
112	70	4	DADSWA		Parameter 1
116	74	4	DADSAVE		Work area
116	74	4	DADPARM2		Parameter 2
120	78	5	DADSPTR		Work area
125	7D	1	DADCODEA		Work area
126	7E	2			Reserved
128	80	4	DADBLD		Work area
132	84	4	DADEXT		Work area
136	88	8	DADATE		Date
144	90	8	DADSVTOC		VTOC address
144	90	2			Reserved
146	92	4	DADSVST		CCHH of VTOC
150	96	2			Reserved
152	98	4	DADSVEND		CCHH of end of VTOC
156	9C	8	DADFLPTR		File name
164	A4	2	DADHH		Number of tracks on device
166	A6	4	DADSTN1		Save area for IKQRDS00

Figure 5.58 DADSM parameter list description and format

The DTFIS table is provided by the user program and contains all the information needed to process a specific ISAM file. Part of it is used by IIP when a VSAM data set is to be processed by an ISAM program. If this is the case, the DTFIS table is reformatted at OPEN time by IIPOPEN. Figure 5.59 shows the DTFIS table description and format.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	16	DTFCCB		
2	2	1	DTFCCBB2		
2	2		ERREXT	X'10'	Accept physical I/O error
16	10	1	FLAGBYTE		
			AM0DTF	X'80'	VSAM bit (set to 1 if DTF belongs to a VSAM data set)
				X'20'	Assign "ignore" bit
17	11	3	LOGMODAD		Address of logic module; if AM0DTF is set to 1, then address of branch vector
20	14	1	FILETYPE		File type
			LOAD	X'24'	LOAD-type DTF
			ADD	X'25'	ADD-type DTF
			RETRVE	X'26'	RETRIEVE-type DTF
			ADDRTR	X'27'	ADD-RETRIEVE-type DTF
21	15	1	OPTIONS1		Options byte 1 (ISAM options)
			BLKDRECS	X'08'	Blocked records
22	16	7	FNAMEDTF		File name (DDname)
29	1D	1	OPTIONS2		Options byte 2 (not used by IIP)
30	1E	1	FNAMEC		Status byte
			LOAD files:		
			UNCIOERR	X'80'	Uncorrectable DASD I/O error
			WRGLEN	X'40'	Wrong length record (not used by IIP)
			PDARFULL	X'20'	No more VSAM data space available
			CYLXFULL	X'10'	No more VSAM data space available
			MASXFULL	X'08'	No more VSAM data space available
			DUPREC	X'04'	Duplicate record
			SEQCHECK	X'02'	Sequence check
			PDAROVFL	X'01'	Prime data area overflow (not used by IIP)

Figure 5.59 **Define The File Indexed Sequential (DTFIS) table description and format (part 1 of 3)**

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
			Non-LOAD files:		
			UNCIOERR	X'80'	Uncorrectable DASD I/O error
			WRGLEN	X'40'	Wrong length record (not used by IIP)
			EOF	X'20'	End of file
			NORECFND	X'10'	No record found
			ILLEGID	X'08'	Illegal identifier specified (not supported by IIP)
			DUPREC	X'04'	Duplicate record
			OFARFULL	X'02'	No more VSAM data space available
			OVFLREC	X'01'	Overflow record (RETRVE) (not used by IIP)
31	1F	12			Not used by VSAM
43	2B	1	RTRBYTE		RETRVE byte
			WORKR	X'80'	WORKR set to 1 if WORKR specified
			WORKS	X'40'	WORKS set to 1 if WORKS specified
44	2C	4	AMDTFAD		Address of AMDTF
48	30	4	CIPROCAD		Address of IIP processor
52	34	4	SAVERG		Save area for one register
56	38	4	PPRETAD		Return address to problem program if called from a $$B phase
60	3C	4	RECLOC		Address of record for LOAD IOREG
64	40	1	CISWITCH		IIP switches
			WNKA	X'80'	Write-new-key-add bit
			RKWK	X'40'	Read-key-write-key bit
			RK	X'20'	Read-key bit
			FIWRITE	X'08'	First write after SETFL
			FIWOK	X'04'	First write is all right
			LD	X'02'	LOAD
65	41	9			Not used by VSAM
74	4A	2	LRECLEN		Logical record length
76	4C	2	KEYLEN		Key length
78	4E	16			Not used by VSAM
94	5E	2	KEYLOC		Key location (not used by IIP)
96	60	4	KARGAD1		Address of KEYARG, moved from part 2 by IIPOPEN if RTR SEQ with KEY (POINT) or RTR RAN is specified
100	64	2	DSPLPRT2		Displacement of part 2 (ADD, RTR)
102	66	2	DSPLPRT3		Displacement of part 3 (ADD, RTR)

Figure 5.59 **Define The File Indexed Sequential (DTFIS) table description and format (part 2 of 3)**

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
104	68	4	LDIOREGS		For RTR SEQ: if WORKS=1, then NOP; if WORKS=0, then L IOREG, RECLOC
108	6C	4	LDIOREGR		For RTR RAN: If WORKR=1; then NOP; if WORKR=o, then L IOREG, RECLOC
112	70	4	WORKAD1		Address of WORKR moved from part 2
116	74	4	IOASAD1		Address of IOAREAS moved from part 2
120	78	64	SAVAR1		For LOAD-type DTF, save area for IIPOPEN
184	B8	4	IORLAD		Address of IOAREAL for LOAD
188	BC	4	DATIWLAD		Address of data in WORKL for LOAD
192	C0	4	KEYIWLAD		Address of key in WORKL for LOAD
196	C4	4			Not used by VSAM
200	C8	1	MIXEXTI		Extend indicator for LOAD
			CROREXT	X'10' X'00'	Extending file Creating file
201	C9	3			Not used by VSAM
204	CC	4	WORKLAD		Address of WORKL for ADD
208	D0	16			Not used by VSAM
224	E0	2	KLM1		KEYLEN-1 for LOAD
			Part 2 of DTF		
0	0	8			Not used by VSAM
8	8	4	IOASAD2		Address of IOAREAS
12	C	4	IOARAD		Address of IOAREAR
16	10	4	KARGAD2		Address of KEYARG
20	14	4	WORKRAD2		Address of WORKR
24	18	4	CURIOAAD		Address of current sequential I/O area
28	1C	4	LIOREGS		L IOREG, *-4 or NOP (RTR SEQ)
32	20	36			Not used by VSAM
68	44	2	NTAGRECS		Number of records tagged for deletion
70	46	2	LIOREGR		LR IOREG,) or NOP(RTR RAN)
			Part 3 of DTF		
0	0	64	SAVAR2		Save area for IIPOPEN, not LOAD type

Figure 5.59 **Define The File Indexed Sequential (DTFIS) table description and format (part 3 of 3)**

Extent Definition Block (EDB)

The EDB describes all extents of the space allocated to the cluster's data set. The EDB is built by the Open module from information in the data set's catalog record. The AMDSB contains the length of the EDB (AMDLEDB), the number of EDB entries (AMDNEDB) that follow the header, and the address of the first EDB (AMDFSEDB). Each EDB entry describes an extent, and contains the address of the associated LPMB. Figure 5.60 shows the EDB description and format.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	4	EDBNEDB		Address of next EDB
4	4	2	EDBSYMU		Symbolic unit (for CCB)
4	4	1	EDBSUCLS		Symbolic unit class
5	5	1	EDBSUNUM		Symbolic unit number
6	6	2	EDBNUMTR		Number of tracks of extent
8	8	1	EDBFLGS		Flags
			EDBDWSS	X'80'	Data RBA with sequence set
			EDBSSWD	X'40'	Sequence set RBA with data
			EDBIXREP	X'20'	Index replication
			EDBMNT	X'10'	Volume mount flag
			EDBLGCC	X'08'	Device contains more than 256 cylinders
			EDBRPS	X'04'	Indicator for RPS device
9	9	3	EDBMBB		Extent (M) and bin (BB) number
9	9	1	EDBM		Extent (M) number
10	A	2	EDBBB		Bin (BB) number
12	C	8	EDBXTNT		Combined name for low and high CCHH
12	C	4	EDBLCCHH		Low cylinder and head numbers
12	C	2	EDBLCC		Lowest cylinder
14	E	2	EDBLHH		Lowest head
16	10	4	EDBHCCHH		High cylinder and head numbers
16	10	2	EDBHCC		Highest cylinder
18	12	2	EDBHHH		Highest head
20	14	4	EDBLPMBA		Address of associated LPMB
24	18	4	EDBPARDB		Address of ARDB
28	1C	2	EDBVLSQ		Index to the VOLSER list
30	1E	2	EDBSTTRK		Relative track address of the start of the extent
32	20	8	EDBRBAS		Combined name for low and high RBAs
32	20	4	EDBLORBA		Low RBA limit
36	24	4	EDBHIRBA		High RBA limit

Figure 5.60 **Extent Definition Block (EDB) description and format**

EXP is a mapping macro that maps the following parameter list when an EXCPAD exit is taken. Figure 5.61 shows the EXCPAD Parameter List description and format.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	4	EXPST		Start of EXCPAD list
0	0	256	EXPAREA		Length of parameter list
0	0	56	EXPSAVE		Register 2-15 save area
0	0	4	EXPSAV02		Register 2 save area
4	4	4	EXPSAV03		Register 3 save area
8	8	4	EXPSAV04		Register 4 save area
12	C	4	EXPSAV05		Register 5 save area
16	10	4	EXPSAV06		Register 6 save area
20	14	4	EXPSAV07		Register 7 save area
24	18	4	EXPSAV08		Register 8 save area
28	1C	4	EXPSAV09		Register 9 save area
32	20	4	EXPSAV10		Register 10 save area
36	24	4	EXPSAV11		Register 11 save area
40	28	4	EXPSAV12		Register 12 save area
44	2C	4	EXPSAV13		Register 13 save area
48	30	4	EXPSAV14		Register 14 save area
52	34	4	EXPSAV15		Register 15 save area
56	38	3			Reserved
59	3B	1	EXPPECBT		Test and set byte
60	3C	68			Reserved
128	80	128	EXPARML		User's parameter list
128	80	4	EXPRPLC		Address of calling RPL
132	84	4	EXPECB		Address of ECB
136	88	4	EXPRPLS		Address of splitting RPL (zero indicates no split)
149	8C	116			Rest of user's parameter list (available to user)

Figure 5.61 EXCPAD Parameter List description and format

Exit List (EXLST)

The EXLST contains addresses for the user-exit processing routines EODAD, SYNAD, LERAD, EXCPAD, and JRNAD. The address of the EXLST is in the ACB (ACBEXLST). The description and format of the EXLST are shown in Figure 5.57.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	1	EXLID		Control block identifier = X'81'
0	0	0	EXLIDD	X'81'	EXLST identifier equate
1	1	1	EXLSTYP		Release indicator
				X'00'	VSAM Release 1
				X'10'	VSAM Release 2
				X'20'	VTAM
2	2	2	EXLLEN		Length of EXLST
4	4	1	EXACT		Active byte
5	5	0	EXLEOD		EODAD entry
5	5	1	EXLEODF		Entry description bits
6	6	4	EXLEODP		Address of the EODAD exit routine
10	A	0	EXLSYN		SYNAD entry
10	A	1	EXLSYNF		Entry description bits
11	B	4	EXLSYNP		Entry of the SYNAD exit routine
15	F	0	EXLLER		LERAD entry
15	F	1	EXLLERF		Entry description bits
16	10	4	EXLLERP		Address of the LERAD exit routine
20	14	0	EXLIOEX		EXCPAD entry
20	14	1	EXLIOEXF		Entry description bits
21	15	4	EXLIOEXP		Address of the EXCPAD exit routine
25	19	0	EXLJRN		JRNAD entry
25	19	1	EXLJRNF		Entry description bits
26	1A	4	EXLJRNP		JRNAD pointer
Bits used in individual exit flags in bytes shown as entry description:					
0	0	1	EXENFL		Flag byte
			EXENEXB	X'80'	Entry present bit
			EXENACTB	X'40'	Entry active bit
			EXENLEB	X'20'	Load bit
1	1	4	EXENADDR		Exit address

Figure 5.62 Exit List (EXLST) description and format (part 1 of 2)

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
Minimum length EXLST for specified entry:					
				Dec. Digit	
			EXLEODL	10	Minimum length if EODAD
			EXLSYNL	15	Minimum length if SYNAD
			EXLLERL	20	Minimum length if LERAD
			EXLIOEXL	25	Minimum length if EXCPAD
			EXLJRNL	30	Minimum length if JRNAD
Minimum and maximum size of EXLST:					
			EXLMINL	10	Minimum length of EXLST
			EXLMAXL	30	Maximum length of EXLST
32	20	0	EXLSTEND		End of EXLST

Figure 5.62 Exit List (EXLST) description and format (part 2 of 2)

Field Control and Data Block (FCDB)

The FCDB describes one of the blocks within the CCW build area. The first FCDB is pointed to by the BKPHD and is used by IKQIOA to construct the channel program. Figure 5.63 shows the FCDB format.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	64	FCB		Maps the module FCB
0	0	56			Space for use in the block
56	38	1	FCBTIC		Reserved for a TIC operation code
57	39	3	FCBCHAIN		Pointer to next block
60	3C	1	FCBCFL		Reserved for chaining flag
61	3D	1	FCBALI		Allocation indicator
			FCBPRVA	X'04'	Previous request allocated
			FCBPRVSV	X'08'	Previous request save
62	3E	2	FCBOFSET		Offset pointer in block
		Equate values			
			FCBCMAX	X'38'	Maximum CCW offset when full
			FCBDMAX	X'30'	Maximum data offset when full
			FCBDINC	X'0C'	Increment for data in space
			FCBCINC	X'08'	Increment for CCW(s) in space
			FCBCP2	X'03'	CCW increment in powers of 2

Figure 5.63 Field Control and Data Block (FCDB) description and format

Field Parameter List (CTGFL)

The CTGFL is built before a DOS/VS component issues the CATLG macro instruction to process a VSAM catalog record. The CTGFL defines one of the catalog record's fields or a group of logically related fields (a combination). The CTGFL is used in three situations:

- It identifies a catalog record field to retrieve or update. The CTGPL contains the address of each CTGFL used in this way.
- It identifies a catalog record field to compare against caller-supplied data. This is a "test" CTGFL and is addressed by another CTGFL.
- For update-extend processing, one or three FPLs identify the volume information group occurrence(s) to be extended. The catalog record fields identified by the CTGFL(s) are not explicitly retrieved or updated for the caller.

When a catalog management routine is processing a CTGFL, the CTGFL's address is in the CCA (CCAFLPT or CCATEST). Figure 5.64 shows the CTGFL description and format.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
0	0	1	CTGFLDNO	Number of entries in CTGFLDAT
1	1	1	CTGFLDCD	Test condition:
		X'00'		The FPL describes a field to be updated or retrieved. The FPL is pointed to by the caller's CTGPL (CTGFIELD entry).
		not X'00'		This FPL describes a test condition, and is pointed to by a request FPL. The codes for the test conditions are:
		X'80'		Equal
		X'60'		Not equal
		X'20'		Greater than
		X'40'		Less than
		X'A0'		Greater than or equal
		X'C0'		Less than or equal
		X'80'		Test under mask for zeros
		X'10'		Test under mask for ones
		X'40'		Test under mask for mixed
2	2	1	CTGFLDGC	Group code number
3	3	1	CTGFLDRE	Test results:
		xxxx xxx.		Reserved
		0		Successful test
		1		Test failed

Figure 5.64 **Field Parameter List (CTGFL) description and format (part 1 of 2)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
4	4	4	CTGFLDWA	Work area: contains information about the catalog record's field name from the dictionary
8	8	4	CTGFLDNM	Address of the field name
12	C	4	CTGFLCHN	Address of next field macro or zero
16	10	0	CTGFLDAT	Combined name for data length and address
16	10	4	CTGFLNG	Data length of:
20	14	4	CTGFLPT	Address (in caller's work area) of: • The field that was retrieved, if the request was LOCATE or CMS LISTCAT, or • New data to replace or add to data in the catalog record, if the request was UPDATE, CMS DEFINE or CMS ALTER, or • Data used to compare to catalog record fields, if the FPL is a FPL-for-tests.

Figure 5.64 Field Parameter List (CTGFL) description and format (part 2 of 2)

Field Vector Table (CTGFV)

The CTGFV is built by the Access Method Services utility programs and contains addresses of user-supplied information fields and lists. The CTGFV is built when the user issues a DEFINE or ALTER command. If the user is creating a cluster, a CTGFV is built for each catalog record that will be built to describe the cluster, i.e., Access Method Services builds a cluster CTGFV, a data CTGFV, and, if the cluster is key-sequenced, an index CTGFV. The CTGFV is pointed to by the CTGPL (CTGFVT). If Access Method Services builds more than one CTGFV, the cluster CTGFVs are pointed to by the CTGPL (CTGFVT) and the data and index CTGFVs are pointed to by the cluster CTGFV. Figure 5.65 shows the CTGFV description and format.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
0	0	1	CTGFVTYP	The CTGFV contains information used by the CMS Define routines to build a catalog record of the type:
		C'D'	CTGFVDTA	Data
		C'C'	CTGFVCL	Cluster
		C'I'	CTGFVIDX	Index
		C'V'	CTGFVVOL	Volume (Space)
		C'A'	CTGFVALN	Non-VSAM
		C'G'	CTGFVAIX	Alternate Index
		C'R'	CTGFVPTH	Path
1	1	1	CTGFVPRO	CMS processing option flags:
		1...	CTGFVAVL	ALTER: Add volumes
		.1..	CTGFVRVL	ALTER: Remove volumes
		..1.	CTGFVNDC	Device type converted switch
		...x xxxx		Reserved
2	2	1	CTGFVELM	Element number of CMSPCATR
3	3	1		Reserved
4	4	4	CTGFVDCH	Address of the cluster's data FVT
8	8	4	CTGFVICH	Address of the cluster's index FVT
12	C	4	CTGFVVCH	Address of the space vector table
16	10	4	CTGFVIND	Address of the associated DLBL statement
20	14	4	CTGFVENT	Address of the 44-byte entry name
24	18	4	CTGFVSTY	Address of the security information FPL (passwords, codewords, and number-of-tries)
28	1C	4	CTGFVOWN	Address of the owner identification FPL
32	20	4	CTGFVEXP	Address of the expiration data FPL
36	24	4	CTGFVCRE	Address of the creation date FPL
40	28	4	CTGFVVLT	Address of the volume serial number list
44	2C	4	CTGFVRNG	Address of the key range list
48	30	4	CTGFVDVT	Address of the device type FPL (for nonVSAM DEFINE only)
52	34	4	CTGFVSPC	Address of the space allocation information FPL
56	38	4	CTGFVAMD	Address of the AMDSB FPL (if VSAM DEFINE)
			CTGFVFSN	Address of the file sequence number (if NonVSAM DEFINE)
60	3C	4	CTGFVATR	Address of the data set attributes FPL
64	40	4	CTGFVBUF	Address of the buffer size FPL
68	44	4	CTGFVLRS	Address of the average record size FPL
72	48	4	CTGFVEXT	Address of exception exit
76	4C	4	CTGFVNAM	Address of related object
80	50	4	CTGFVUPG	Address of FPL for 'RGATTR'
84	54	4	CTGFVWKA	Address of CRA volume identification
88	58	4	CTGFVPWD	Relationship password

Figure 5.65 **Field Vector Table (CTGFV) description and format**

I/O Arguments (IOARG)

IKQIOARG maps the I/O data areas in the FCDB. Figure 5.66 shows the IOARG description and format.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	8	IOASEEK		Computed DASD address
0	0	1	IOAFLAG		Flag byte
			IOARPS	X'80'	RPS device
1	1	2	IOABB		BB
3	3	4	IOACCHH		CCHH
3	3	2	IOACC		CC
5	5	2	IOAHH		HH
7	7	1	IOAR		R
8	8	1	IOAKEY		Key size
9	9	1	IOADATA		Data size
11	B	1	IOASEC		RPS sector size

Figure 5.66 I/O Arguments (IOARG) description and format

I/O Drive Block (IODRB)

IKQIODRB maps the BUFRIODR and BUFWIODR fields of the BCB. Figure 5.67 shows the IODRB format and description.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	20	IODLEN		
0	0	2	IODCURU		Symbolic unit number
2	2	2	IODBKSTI		Number of blocks to I/O
4	4	8	IODSEEK		Compiled DASD address
4	4	1	IODFLAG		Flag byte
			IODRPS	X'80'	RPS device indicator
5	5	2	IODBB		BB
7	7	4	IODCCHH		CCHH
7	7	2	IODCC		CC
9	9	2	IODHH		HH
11	B	1	IODR		R
12	C	4	IODRBA		RBA for I/O
16	10	4	IODLPMB		Address of associated LPMB

Figure 5.67 I/O Driver Block (IODRB) description and format

IKQIOWKA maps the PLH Workarea (PLHWAREA, displacement X'C8'). Figure 5.68 shows the IOWKA description and format.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
200	C8	44	WKAREA		Beginning of IOWKA
200	C8	8	WKASEEK		Work area DASD address
200	C8	1	WKAM		M
201	C9	2	WKABB		BB
203	CB	4	WKACCHH		CCHH
203	CB	2	WKACC		CC
205	CD	2	WKAHH		HH
207	CF	1	WKAR		R
208	D0	4	WKABDC14		BLDCP000 save area
212	D4	12	WKATEMP		Temporary area
224	E0	12	WKAGETVS		ALLBK save area
224	E0	2	WKARTNCD		Return code
226	E2	2	WKABLKS		Number of blocks on active CCB
228	E4	1	WKAERRSW		Hold error indicator in IKQIOB
229	E5	1	WKAIOMSW		I/O manager flags
			IOMPROCR	X'80'	Process reads
			IOMPROCW	X'40'	Process writes
			IOMPROCK	X'20'	Process write checks
230	E6	2			Reserved
232	E8	4	WKADBHD		Address of working BHD
236	EC	4	WKAREGSV		ALLOC register save area
240	F0	4	WKASVCCB		Save address of previous CCB

Figure 5.68 I/O Work Area (IOWKA) description and format

Logical-to-Physical Mapping Block (LPMB)

The LPMB contains information about the direct-access device that contains the user's data set. The LPMB is built by the Open module, using information in the data set's catalog record. The EDB (EDBLPMBA) contains the address of the LPMB. Figure 5.69 shows the LPMB description and format.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	1	LPMID	X'FF'	Control block identifier
1	1	1	LPMBDTF		DTF device type indicator
2	2	2	LPMLEN		Length of the LPMB
4	4	4	LPMBPTRK		Number of bytes per track
8	8	4	LPMCASZ		Number of bytes per control area
12	C	4	LPMBLKSZ		Physical block size
16	10	2	LPMTRKCA		Number of tracks per control area
18	12	2	LPMTPC		Number of tracks per cylinder
20	14	2	LPMBNQBK		Number of physical records per track
22	16	2	LPMBPBCI		Physical blocks per control interval

Figure 5.69 **Logical-to-Physical Mapping Block (LPMB) description and format**

Open ACB List (OAL)

The OAL is a list which contains all VSAM ACB's that have been opened. It is built by IKQOPN. The addresses of the ACB's are also entered by IKQOPN. OAL entries are deleted by IKQCLO00 and $$BACLOS. Figure 5.62 shows the description and format of the OAL. The field at displacement X'0C' of the Anchor Table contains the address of the OAL.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	1	OALID OALIDD	 X'F0'	OAL identifier OAL identifier equate
1	1	1			Reserved
2	2	2	OALLEN		Length of this block (128)
4	4	4	OALNOAL		Pointer to next OAL
8	8	2	OALNOPN		No. of open data sets in the partition
10	A	2	OALNENT		No. of OAL entries (14)
12	C	4	OALSAVE		Save area for interrupt information during automatic close. Data consists of interrupt length (2 bytes) and interrupt code (2 bytes).
16	10	0	OALENTRY		Entry description (repetitive)
16	10	4	OALACB		Address of opened ACB
20	14	1	OALSVC		Delimiter (X'0A')
21	15	1	OALFLG OALACT	 X'80'	Flag byte ACB is open
22	16	2	OALCICHK		Value to check validity of control interval number of data set in catalog. Value is formed at open time by adding the third byte of AMDDSM to the first two bytes of AMDDSM.

Figure 5.70 **Open ACB List (OAL) description and format**

Open Work Area (OPNWA)

The Open Work Area is built when a VSAM data set is opened or is being created. It contains the addresses of control blocks and work areas needed when a data set is being opened. The Open Work Area also contains flags that indicate the type of processing being performed on the data set and the address of the SVC that invoked the processing. The Open Work Area is pointed to by the AMBL and register 4 (RWKA). Figure 5.71 shows the Open Work Area description and format.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
0	0		WACOMMON	Common Open/Close work area
0	0	1	WAFLAG	Flag byte:
		1...	TCLOSE	Work Area for TCLOSE
		.1..	CLOSE	Work Area for Close
		..1.	OPEN	Work area for Open
		...1	OPAMDINX	Index AMDSB is being processed
		.... 1...	VOLFOUND	Volume serial number is in label cylinder record
		.... .1..	SSFLAG	Sequence set with data
		.... ..1.	RETRY	Catalog should be reupdated by Close
		1	FILEPROT	DOS Supervisor DASD file protect
1	1	1	WAERCODE	Error condition code
2	2	2	WALEN	Length of GETVIS area
4	4	4	WAPIBSV	Address of USERSAVE field
8	8	4	WALISTP	Address of user ACB/DTF list
12	C	2	WACOMR	Address of DOS communication region
14	E	1	EDBCODE	One GETVIS obtains enough space for 3 EDBs; this field is used to count EDBs
15	F	1		Reserved
16	10	4	CATEXTPT	Pointer to extent information in order to build EDBs
20	14	2	CATEXTLN	Length of total extents
22	16	2	EXTNUMB	Number of extents
24	18	80	USERSAVE	Room to save user jobname, PSW, and registers (from partition save area)
104	68	0	WACOMEND	End of common work area
104	68	0	OWA	Partial map of work area obtained by GETVIS issued by $$BOVSAM
104	68	4	WAVSLOD	Address of location where VSAM has been placed by CDLOAD (set by $$BOVSAM)
108	6C	4	WAIKQLAB	Address of location where IKQLAB has been placed by CDLOAD (set by $$BOVSAM)
112	70	4	WARACB	Pointer to ACB being opened
112	70	4	CLWAAD	Close work area address saved
116	74	1	LBLRCLEN	Length of work area pointed to by LABICPTR in multiple of 128
117	75	3	LABICPTR	Pointer to work area reserved for label record
120	78	4	SVCATACB	Pointer to catalog ACB
124	7C	4	CTGPLPTR	Pointer to catalog parameter list (CPL)
128	80	4	CATWKPTR	Pointer to catalog work area (CTGWA) (contents moved to CPL)

Figure 5.71 Open Work Area (OPNWA) description and format (part 1 of 7)

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
132	84	4	OLDEDB	Address of EDB
136	88	4	NXTEDB	Address of next EDB
		Catalog Field List for AMDSB		
140	8C	0	FLAMDSB	Catalog field list work area (CTGFLDWA) for AMDSB
140	8C	0	SAVERET1	Pointer to contents of return register (R 14) if not catalog call
140	8C	4	RETREG1	Return address to save area 1
144	90	0	SAVERET2	Return address to save area 2
144	90	4	RETREG2	Return address to save area 2
148	94	0	FLAMDSBN	Pointer to catalog field name AMDSBCAT
148	94	4	RETREG3	Return address to save area 3
152	98	4	*	Next CTGFLCHN number
156	9C	4	FLAMDSBL	Length of AMDSBCAT
160	A0	4	FLAMDSBA	Address of AMDSBCAT
		Catalog Field List for Volume Entry(ies)		
164	A4	8	FLENTVOL	
164	A4	2	KRNKEYS	No. of key ranges equals number of ARDBs
166	A6	2	KRNVOLS	Number of volumes for this key range
172	AC	4	FLVOLNTN	Volume entry name ENTVOL
176	B0	4	*	
176	B0	4	SVLENG	Length of ENTVOL
180	B4	4	VOLENTLN	Length of volume entry
184	B8	4	VOLGPPTR	Address of ENTVOL data
		Catalog Field List for Data Set Attributes		
188	BC	20	*	DSATTR field list
208	D0	4	FLDSATRA	Base of DSATTR
		Catalog Field List for Open Indicator		
212	D4	8	FLOPNIND	Locate OPENIND and test for UPD
220	DC	4	FLOPNINN	Open indicator field list
224	E0	4	*	Chain
228	E4	4	FLOPNINL	Length of OPENIND
232	E8	4	FLOPNINA	OPENIND address
		Catalog Field List for Minimum Buffer Size		
236	EC	20	*	Flags, etc., for BUFSIZE
256	100	4	FLBUFSZA	Base of BUFSIZE
* Multi-use field				

Figure 5.71 **Open Work Area (OPNWA) description and format (part 2 of 7)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
		Catalog Field List for High-Used RBA per Data Set		
260	104	20	*	Miscellany for HURBADS
260	104	20	NVOLLIST	No. of volumes per key range (Space for 10 two-byte entries)
280	118	4	FLHURDSA	Base for HURBADS
284	11C	20	*	CATFILE field list
304	130	4	FLFILTA	Base for CATFILE
		Catalog Field List for Names of Related Data Sets		
308	134	8	FLNAMEDS	Flags for NAMEDS
308	134	8	PARMLIST	IKQVLAB parameter list
308	134	4	PARM1	ACB DD name
312	138	1	PARMLEN	Length of work area for IKQLAB
313	139	3	PARM2	Pointer to 'LABICPTR'
316	13C	4	FLNAMDSN	Pointer to 'NAMEDS'
320	140	4	*	
324	144	4	FLNAMDSL	Length of associated names
328	148	4	FLNAMDSA	Address of NAMEDS groups
		Catalog Field List for Entry Type and Control Interval No.		
332	14C	8	*	CTGFLDWA for this field list
340	154	4	FLMISCLN	Pointer to 'DSTYPNAM'
344	158	4	*	Chain
348	15C	4	FLMISCLL	Length of DSTYPNAM
352	160	4	FLMISCLA	Address of DSTYPNAM
		Catalog Field List to Find Catalog ACB Address		
356	164	20	FLCATACB	Field list #10 for catalog ACB
376	178	4	FLCTACBA	Pointer to catalog ACB pointer
		Catalog Field List to Test for Write of Open Indicator		
380	17C	8	FLWOPNND	Update OPENIND field list
380	17C	4	TSTENTVL	Address of test ENTVOL (scan)
384	180	4	TSTENTLN	Address of end scan ENTVOL
388	184	4	FLWOPNNN	Pointer to 'OPENIND'
392	188	4	*	Chain
396	18C	4	FLWOPNNL	Length of data
400	190	4	FLWOPNNA	Pointer to data
		Catalog Field List for Volume Time Stamp		
404	194	24	FLTMSTVF	VOLTSTMP field list
404	194	20	*	Greater part of field list
424	1A8	4	FLTMSTVA	Pointer to 'VOLTSTMP'

* Multi-use field

Figure 5.71 **Open Work Area (OPNWA) description and format (part 3 of 7)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
		End of Catalog Field List for Volume Time Stamp		
428	1AC	1	WARNFLG	Used to save warning error code
429	1AD	1		Reserved
430	1AE	2	I	Index for DO loops
432	1B0	2	LIMIT	Count of the ENTVOLS pointed to by VOL20PT
434	1B2	2	RELGP	Relative group number in the catalog
436	1B4	2	TEMP	Local calculations (on same listing page)
438	1B6	2	IARDB	Index for ARDB list
440	1B8	4	SAVDEV	Used to save device type
444	1BC	4	SAVDEV2	Used to save sequence set device type
448	1C0	2	SAVTRKAU	Used to save number of tracks per allocation unit (control area) to help identify type of LPMB
450	1C2	2	SAVTRKA2	Same as SAVTRKAU but used only if sequence set with data
452	1C4	4	RLPMB2	Pointer to sequence set (index) LPMB
456	1C8	1	OWAFLAGS	Open flags and switches
		1...	OWFLAGZB	User did not specify buffer size in ACB
		.1..	OWFLAGBF	BCB building in process
		..1.	OWFLAGIB	Got buffer with AMBL for index
		...1	WARSOPEN	USE macro has been issued for SYSOPEN (RELEASE macro must subsequently be issued)
		.... 1...	DTACNT	Open count in look-aside table is bumped for data
		.... .1..	IDXCNT	Open count in look-aside table is bumped for index
		.... ..1.	WARSCTLG	USE macro has been issued for SYSCTLG (RELEASE macro must subsequently be issued)
		x		Reserved
457	1C9	3	INDEXSAV	Used to save index file name
460	1CC	1	SAVTYPE	Used to save entry type when entry is not a cluster
461	1CD	2		Reserved
463	1CF	4	TESTSV1	Save a word for testing
467	1D3	1	SVOPNIN	Updated OPENIND for catalog
		1...	SVOPNINO	Flag open for output
		.xxx xxxx		Reserved
468	1D4	2	SVNEXTNT	Save number of EXTENT statements
470	1D6	2	SETNBUF	Count of buffers (used by SETADDR)
472	1D8	4	VOLSTPTR	Address of IDAVLST
476	1DC	4	VOLENTND	End of all ENTVOLs
480	1E0	2	VOLENTCT	Count of volume entries
482	1E2	2	IVOLS	Working index of ENTVOLs

* Multi-use field

Figure 5.71 **Open Work Area (OPNWA) description and format (part 4 of 7)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
484	1E4	4	VOL20PT	Pointer to volume entries to sort (address of VOLENT20 if less than 20)
488	1E8	80	VOLENT20	Volume entries to sort
568	238	4	VMTPTR	Pointer for right VOLSER
572	23C	4	REQBUFSP	Minimum buffer space required
576	240	4	CURBUFSP	Currently specified buffer space
580	244	4	CURBFSPD	Current buffer space specified for data
584	248	32	ADDAREA	Room to add without current specifications for index
584	248	4	CURBFSPI	Current buffer space specified for index
588	24C	1	SVLUBPUB	Save index of PUB
589	24D	1	NEXTJIB	Next JIB saved
590	24E	10	SVPUB	LUBs for mounted volumes
600	258	2	IPUB	Index for SVPUB
602	25A	4	WRKCINV	Control interval size used in pointing BCBs to buffers
606	25E	8	OWAPRTCT	Room to build password
614	266	2		Unused
616	268	12	PARM	Parameter list for IKQLASMD
616	268	1	CALLERID	Caller identification
617	269	7	DSID	Data set identification
617	269	3	DSCI	Control interval number
620	26C	4	CTACBPTR	Pointer to catalog ACB
624	270	1	SHAREOPT	Share option from catalog
625	271	1		Reserved
626	272	2	OUTCNT	Number of output users, returned from IKQLASMD
628	274	72	OWPLSAVE	Save area
700	2BC	72	OWPLSAV2	Save area 2
772	304	80	DUMCATPL	Room for catalog parameter list
852	354	512	OWACTWKA	Normal catalog work area
1364	554		CCWX	CCW definition
1364	554	1	CCWCODE	Write-to-console op code
1365	555	3	CCWDTA	Pointer to message buffer
1368	558	2	*	
1370	55A	2	CCWCNT	Length of message buffer
1372	55C	24	CCBX	CCB definition
1372	55C	9	*	
1381	565	3	CCWPT	Pointer to channel program (CCWX)
1384	568	12		Unused

* Multi-use field

Figure 5.71 Open Work Area (OPNWA) description and format (part 5 of 7)

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
1396	574	0	VMSG	Volume name is built and used as part of calling parameter when catalog is called to get the time stamp
1396	574	0	MSG	Volume time stamp built
1396	574	11	MSGID	Message identification
1407	57F	8	MSGDSN	Data set name
1415	587	46	MSGTXT	Message text
1461	5B5	3	*	
1464	5B8	4	OWSTRTGV	Start of GETVIS
1468	5BC	4	OWAOAL	Address of OAL section
1472	5C0	4	UACBAD	User's ACB address
1476	5C4	4	AIXACBAD	AIX cluster ACB address
1480	5C8	4	BCACBAD	Base cluster ACB address
1484	5CC	4	UPACBAD	ACB of upgrade member
1488	5D0	4	USBAD	Pointer to USB
1492	5D4	4	OWAUCPT	Pointer for IKQLAB
1496	5D8	24	FLRGATTR	Copy of CTGFL for "RGATTR"
1496	5D8	16	*	
1512	5E8	4	*	Length of "RGATTR" data
1516	5EC	4	FLRGATRA	Pointer to "RGATTR"
1520	5F0	24	FLEXCPEX	Copy of exception exit CTGFL
1520	5F0	16	*	
1536	600	4	FLEXCEPL	Length of exception exit data
1540	604	4	FLEXCEPA	Address of EXCPEXIT
1544	608	72	OWPLSAV3	Third level save
1616	650	80	INTCPL	Internal CPL
1696	6A0	4	RPLPAD	RPL pool just handled
1700	6A4	4	PLHADDR	Address of first PLH
1704	6A8	4	AIXBUFAD	Upgrade buffer pool
1708	6AC	4	AIXBUFLN	Length of upgrade buffer pool
1712	6B0	24	MSGPARMS	Parameter list for IKQOCMSG
1736	6C8	2	MSGFLGBT	Message flag byte
1738	6CA	2	NRPL	Number of user strings
1740	6CC	2	AIXBCLEN	GETVIS length for ACB/RPL
1742	6CE	2	UPGRM	Number of members in upgrade set
1744	6D0	2	UPGRCT	Upgrade set loop counter
1746	6D2	2	AIXUPLEN	Length of upgrade set (RPL+PLH)

Figure 5.71 **Open Work Area (OPNWA) description and format (part 6 of 7)**

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
1748	6D4	1	AIXFLG	Alternate index flags
		1...	AIXUPGR	Upgrade set available
		.1..	AIXBASE	Base cluster handled
		..1.	AIXPE	Path entry handled
		...1	AIXPEU	Path entry of upgrade set
		.... 1...	AIXPATH	Path structure open
		.... .1..	AIXMUS	Member of upgrade set handled
		.... ..1.	AIXEUO	AIX as end-use object
		1	AIXUSERR	Upgrade set error
1749	6D5	1	AIXFLG2	Alternate index flags 2
		1...	AIXTHB	THB is for upgrade set
		.1..	AIXPEUBF	One additional index buffer already present
		..xx xxxx		Reserved
1750	6D6	1	PATHFLG	Path flags, first byte
		1...	PFLUPD	Update option
		.xxx xxxx		Reserved
1751	6D7	1	*	Path flags, second byte
		1...	RESETSW	Switch for reset of data set at open time
		.1..	ESDSERR	ESDS error flag
		..1.	OALEFND	OAL entry found
		...1	JRNACT	JRNAD active
		.... 1...	CATOPEN	Catalog open in progress
		xxx		Reserved
1752	6D8	1	SAVAIX	Save area for AIXFLG
1753	6D9	2	AIXUSAV	Save area for ACB option
1755	6DB	3	AIXYENTR	Internal address of Y entry
1758	6DE	3	AIXDNAM	AIX data name
1761	6E1	3	AIXINAM	AIX index name
1764	6E4	3	BCDNAM	Base cluster data name
1767	6E7	3	BCINAM	Base cluster index name
1770	6EA	3	CLUNAME	Cluster name save area
1773	6EE	8	NAMEFLD	USE/RELSE parameter list
1781	6F5	5	INTWA	Internal catalog work area
1786	6FA	512	OWA2	Work area
1786	6FA	512	OWAUCAT	IKQCAT work area for UCAT
1786	6FA	512	USCTGWA	Catalog work area IKQOPNUS
1786	6FA	512	OWAMSGAR	Message work area
2298	8FA	4	OWAEND	Reserved
2302	8FE	2	LSRCNT	Number of data sets using LSR
2304	900	4	AUROOT	Address of first control block in allocation unit
2308	904	4	CPAADR	Address of channel program area
2312	908	4	WAIKQSTM	Address of storage manager module

Figure 5.71 **Open Work Area (OPNWA) description and format (part 7 of 7)**

Placeholder (PLH)

The PLH contains current information about a request. This information includes positioning information, request options, and buffer location and status. The PLH is built by the Open module and is pointed to by the AMBL (AMBLPLH). When a record management module is processing a PLH, the PLH's address is in register 13. Figure 5.72 shows the PLH description and format.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
			Standard Save Area		
0	0	0	PLHSAREA		Register save area
0	0	4			Reserved
4	4	4	PLHSADDR		Address of user's save area
8	8	4			Reserved
			Buffer Manager Save Area		
12	C	60	PLHSAVE		Save area for 15 registers
			I/O Manager Save Area and JRNAD Extended Save Area		
72	48	44	PLHBSAVE		I/O manager save area
116	74	48	PLHIXSSV		Index search and get next save area
164	A4	16	PLHJRNSV		JRNAD save area
			Return Register Stacks		
180	B4	0	PLHSTCK		Fixed return register stack
180	B4	4	PLHSTCK1		Return register from level 1
184	B8	4	PLHSTCK2		Return register from level 2
			RPL Pointers		
188	BC	4	PLHHRPL		Pointer to header RPL
192	C0	4	PLHCRPL		Pointer to current RPL
			PLH ECB		
196	C4	4	PLHECB		Event control block
196	C4	1			Reserved
197	C5	1	PLHAUSE		Request active on PLH
198	C6	1	PLHECOM		Communications byte
			PLHEWAIT	X'80'	Wait flag on ECB
199	C7	1	PLHECBT		Test and set byte for ECB
			PLH Work Area		
200	C8	44	PLHWAREA		PLH work area
			PLH Identification Byte		
244	F4	1	PLHID	X'55'	PLH identification byte

Figure 5.72 Placeholder (PLH) description and format (part 1 of 8)

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
			PLH Use Gate		
245	F5	1	PLHUSE		PLH use gate If ON (X'FF'), this PLH is available only to an RPL whose string identifier (RPLSTRID) is equal to the string identifier (PLHSTRID) of this PLH.
			PLH Condition Flags		
246	F6	1	PLHFLAG		PLH condition flags
			PLHST	X'80'	PLH status flag (bit 0) 1 - PLH set 0 - PLH invalid
			PLHPOS	X'40'	PLH position flag (bit 1) 1 - Next record 0 - previous record
			PLHEOD	X'20'	PLH end-of-data-condition flag (bit 2) 1 - EOD reached 0 - Not EOD
			PLHWAIT	X'10'	PLH wait flag (bit 3) 1 - I/O pending 0 - No I/O pending
			PLHSKIP	X'08'	PLH skip flag (bit 4) 1 - Skip control interval 0 - Don't skip control interval
			PLHRST	X'04'	PLH restart flag (bit 5) 1 - Restart 0 - No restart
			PLHFST	X'02'	PLH first-time flag (bit 6) 1 - First time 0 - Not first time
			PLHRREAD	X'01'	PLH exclusive control reread flag (bit 7) 1 - Need reread 0 - Reread not needed
247	F7	1	PLHFLG		PLH spare condition flag
			PLH Communication Switches		
248	F8	1	PLHSWTCH		PLH communication switches
			PLHLOAD	X'80'	PLH load or resume load indicator
			PLHKRCH	X'40'	PLH key range change indicator
			PLHMSRT	X'20'	Mass insert indicator
			PLHFSR	X'10'	First request for data set indicator
			PLHSTBCB	X'04'	Demand a BCB from STEAL000 (IKQBFA00)
			PLHEC	X'02'	Exclusive control needed
			Previous Request Characteristics		
249	F9	3	PLHPREQ		Previous request information
249	F9	1	PLHRTC		Previous request-type code
250	FA	2	PLHOPT		Previous request option bytes
250	FA	1	PLHOPT1		First option byte

Figure 5.72 Placeholder (PLH) description and format (part 2 of 8)

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
251	FB	1	PLHOPT2		Second option byte
252	FC	4	PLHPKEYA		Pointer to save area for keyed access
			Internal request characteristics for LSR		
256	100	4	PLHILTM		Test mask for WRTBFR
260	104	4	PLHILRM		Reset mask for WRTBFR
264	108	1	PLHPERC		Percentage value for number of least recently used buffers
265	109	1	PLHIOPT1		Option byte for WRTBFR
			PLHIDSC	X'80'	Data set processing for forced close
			PLHIDS	X'40'	Write buffers for a data set
			PLHILRU	X'20'	Write least recently used buffers
			PLHIALL	X'10'	Write all buffers
			PLHITRN	X'08'	Write buffers for specified transaction identifier
			PLHIBCB	X'04'	Subpool contains at least one modified buffer
			PLHIFIO	X'02'	Force I/O for buffer
266	10A	2			Reserved
268	10C	4	PLHDBSPH		Address of data buffer subpool
272	110	4	PLHIBSPH		Address of index buffer subpool
276	114	4	PLHACB		Pointer to data set's ACB
280	118	4	PLHEACB		Address of ACB of buffer with error
284	11C	4	PLHLSRA		Address of LSR save area
			Multiple String Support		
288	120	1	PLHSTRID		PLH string ID (1-255)
289	121	1	PLHENDRQ		ENDREQ request gate byte
290	122	1	PLHINDS		Indicator byte
			PLHCLOSE	X'80'	Close-type ENDREQ request
291	123	1			Reserved
			EXCPAD Parameter List Pointer		
292	124	4	PLHPARML		EXCPAD parameter list pointer
			JRNAD Parameter List Pointer		
296	128	4	PLHAJRN		JRNAD parameter list pointer
			I/O Manager Entry Point		
300	12C	4	PLHIOMGR		I/O Manager (IKQIOA00) entry point

Figure 5.72 Placeholder (PLH) description and format (part 3 of 8)

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
			Key Range Support Fields		
304	130	4	PLHDCRDB		Address current ARDB
308	134	4	PLHDTRDB		Address target ARDB
			Pointers to Buffer Headers (BHDs)		
312	138	4	PLHDBHD		Address of data BHD
316	13C	4	PLHIBHD		Address of index BHD
320	140	4	PLHBRPL		Save header RPL
324	144	4	PLHTHB		Address of THB (share option 4)
328	148	4			Reserved
			Data PLH		
332	14C	36	PLHDATA		Data PLH
332	14C	20	PLHDCNV		Data CNV information
332	14C	4	PLHDRBA		Data CNV RBA
336	150	8	PLHDBUF		Data buffer description
336	150	4	PLHDBCB		Address of data BCB
340	154	4	PLHDBAD		Address of data buffer
344	158	4	PLHDCIDF		Data CNV CIDF
344	158	2	PLHDFSO		Data CNV free space offset
346	15A	2	PLHDFSL		Data CNV free space length
348	15C	1	PLHDSW		Data CNV switches
			PLHHOLD	X'80'	Track hold indication
			PLHHELD	X'40'	Track free indication
			PLHNORD	X'10'	No read indication
			PLHLOG	X'08'	Logical GETBUFF request
			PLHRAHD	X'04'	Read-ahead request
349	15D	1	PLHDSW1		Buffer request control switch
			PLHEHOLD	X'80'	Exclusive control desired
			PLHEHELD	X'40'	Exclusive control held
			PLHEACTV	X'20'	Exclusive control active
			PLHCATH	X'10'	CA split track hold
			PLHCATF	X'08'	CA split track free
			PLHDIRQ	X'04'	Indication to the buffer manager that this direct write request is to be deferred (set and reset by IKQMDY)
350	15E	2	PLHDCSZ		Data CNV size 10 (rightmost RDF)
			Data Record Description		
352	160	16	PLHDRCD		Data record description
352	160	2	PLHDRO		Data record offset
354	162	2	PLHDRDF		Data record RDF-offset
356	164	2	PLHDRIX		Data record RDF-index
358	166	2			Spare
360	168	4	PLHDRRBA		Data record RBA
364	16C	4	PLHDRL		Data record length

Figure 5.72 Placeholder (PLH) description and format (part 4 of 8)

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
			Read-Ahead Data PLH		
368	170	24	PLHBDATA		Data read-ahead PLH
368	170	4	PLHBRBA		RBA of next CNV to read ahead
			Read-Ahead Data CNV Description		
372	174	20	PLHBDCNV		Read-ahead data CNV information
372	174	4	PLHBDRBA		Data CNV RBA
376	178	8	PLHBDBUF		Data buffer description
376	178	4	PLHBDBCB		Address of data BCB
380	17C	4	PLHBDBAD		Address of data buffer
384	180	4	PLHBDCDF		Data CNV CIDF
384	180	2	PLHBDFSO		Data CNV free space offset
386	182	2	PLHBDFSL		Data CNV free space length
388	184	1	PLHBDSW		Data CNV switches (same as PLHDSW)
389	185	1	PLHBDSW1		Buffer request control switch (same as PLHDSW1)
390	186	2	PLHBDCSZ		Data CNV size-10
			Alternate Index Record Information		
392	188	16	PLHAIX		AIX record information
392	188	4	PLHAIXPT		Address of base cluster pointer
396	18C	4	PLHBCPLH		Address of base cluster's PLH
400	190	4	PLHAIXWL		Reserved
404	194	2	PLHAIXPN		Number of base cluster pointers still to be processed in this AIX record
406	196	2	PLHAIXOP		RPL Option bytes
408	198	12	PLHUPG		Upgrade set information
408	198	4	PLHUPGP1		Current USB entry address
412	19C	4	PLHUPGP2		Last USB entry address
416	1A0	4	PLHUPGAD		Address of prime key (KSDS) or RBA (ESDS) of base cluster record
420	1A4	24	PLHAIXSV		AIX save area
			Spanned Record Flag Byte		
444	1BC	1	PLHSWT2		Spanned record switch byte
			PLHSPAN	X'80'	Spanned record indicator
			PLHSRU	X'40'	Called from IKQSRU
			PLHSRUF	X'20'	First call from IKQSRU
			PLHSRUL	X'10'	Last call from IKQSRU
			PLHSRCAS	X'08'	CA-split necessary
				X'04'	Reserved
			PLHSREC	X'02'	Exclusive control indicator
				X'01'	Reserved

Figure 5.72 **Placeholder (PLH) description and format (part 5 of 8)**

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
			JRNAD Flag Byte		
445	1BD	1	PLHJRN		JRNAD flag byte
			PLHJRVSM	X'40'	JRNAD called from IKQVSM
			PLHJRMDY	X'20'	JRNAD called from IKQMDY
			PLHJRCIS	X'10'	IRNAD called from IKQCIS
			PLHJRCA1	X'08'	JRNAD first call from IKQCAS
			PLHJRCA2	X'04'	JRNAD second call from IKQCAS
			PLHJRSRG	X'02'	JRNAD called from IKQSRG
			PLHJRSRU	X'01'	JRNAD called from IKQSRU
			Spanned Record Information		
446	1BE	2	PLHSRCNT		Number of segments
448	1C0	22	PLHSPREC		Spanned record information
448	1C0	8	PLHRCD		Spanned record description
448	1C0	4	PLHAREA		Pointer to user area
452	1C4	4	PLHRLEN		Length of spanned record
456	1C8	4	PLHSRRBA		RBA of record
460	1CC	2	PLHX1EO		Index entry offset of first part
462	1CE	2	PLHXPTR		Pointer number
464	1D0	6	PLHSRRDF		Double RDF for spanned record
464	1D0	1	PLHSRR2		R byte of 2nd (leftmost) RDF
465	1D1	2	PLHSRLVL		Level number
467	1D3	1	PLHSSR1		R byte of 1st (rightmost) RDF
468	1D4	2	PLHSRLL		Length of segment
			Additional PLH Switches		
470	1D6	1	PLHSWT1		PLH communication switch control
			PLHRSI	X'10'	RPL area switch indicator (indicates that the user RPL areas of an AIX and the base cluster have been switched)
			PLHUPRES	X'08'	AIX upgrade reset switch
			PLHPCI	X'04'	Previous index control interval required
			PLHBWD	X'02'	Backward processing
			PLHLRD	X'01'	Last record processing
471	1D7	1	PLHFLG1		Flag byte continuation
			PLHDUKEY	X'08'	Duplicate key in AIX record
			PLHAIXRP	X'04'	AIX repositioning flag (Previous AIX record must be read)
			Index PLH		
472	1D8	40	PLHINDEX		Index PLH
			PLHESDS		Length of PLH for ESDS (equate value)

Figure 5.72 **Placeholder (PLH) description and format (part 6 of 8)**

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
			Index CNV Description		
472	1D8	20	PLHXCNV		Index CNV information
472	1D8	4	PLHXRBA		Index CNV RBA
476	1DC	8	PLHXBUF		Index buffer description
476	1DC	4	PLHXBCB		Address of index BCB
480	1E0	4	PLHXBAD		Address of index buffer
484	1E4	4	PLHXCIDF		Index CNV CIDF
484	1E4	2	PLHXFSO		Index CNV free space offset
486	1E6	2	PLHXFSL		Index CNV free space length
488	1E8	1	PLHXSW		Index CNV switches (same as PLHDSW)
489	1E9	1	PLHXSW1		Buffer request control (same as PLHDSW1)
490	1EA	2	PLHXCSZ		Index CNV size-10
			Index Entry Description		
492	1EC	20	PLHXETRY		Index entry description
492	1EC	2	PLHXEO		Index entry offset
494	1EE	2	PLHXSEO		Next section entry offset
496	1F0	4	PLHXSOP		Last section entry offset pointer
500	1F4	2	PLHXLVL		Present index level in process
502	1F6	2	PLHXLEVP		Previous level index
504	1F8	4	PLHXPTRP		Previous entry's P field
504	1F8	2	PLHXEOP		Previous entry offset
506	1FA	2	PLHXSEOP		Previous section entry offset
508	1FC	4	PLHXRBAP		Previous index record RBA
			Read-Ahead Index PLH		
512	200	28	PLHBINDX		Read-ahead index PLH
			Read-Ahead Index CNV Description		
512	200	20	PLHBXCNV		Read-ahead index CNV information
512	200	4	PLHBXRBA		Index CNV RBA
516	204	8	PLHBXBUF		Index buffer description
516	204	4	PLHBXBCB		Address of index BCB
520	208	4	PLHBXBAD		Address of index buffer
524	20C	4	PLHBXCDF		Index CNV CIDF
524	20C	2	PLHBXFSO		Index CNV free space offset
526	20E	2	PLHBXFSL		Index CNV free space length
528	210	1	PLHBXSW		Index CNV switches (same as PLHDSW)
529	211	1	PLHBXSW1		Buffer request control switch (same as PLHDSW1)
530	212	2	PLHBXCSZ		Index CNV size-10

Figure 5.72 Placeholder (PLH) description and format (part 7 of 8)

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
			Read-ahead Index Entry Description		
532	214	2	PLHBXEO		Index entry offset
534	216	2	PLHBXSEO		Next section entry offset
536	218	4	PLHBXSOP		Last section entry offset pointer
			Previous Record Key Information		
540	21C	*	PLHPKEY		Key of previous record
540	21C	0	PLHEND		End of PLH
			PLHLKSDS		Length of PLH for KSDS (equate value)

* Variable, equal to key length.

Figure 5.72 Placeholder (PLH) description and format (part 8 of 8)

Request Parameter List (RPL)

The RPL contains user-request information and error feedback information. It also maintains information required by GET and PUT. The RPL is created by the user with the RPL macro instruction. Figure 5.73 shows the RPL description and format.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	1	RPLID		Control block identifier = X'00'
0	0	1	RPLIDD	X'00'	RPL equate
1	1	1	RPLSTYP		Release indicator
				X'00'	VSAM Release 1
				X'10'	VSAM Release 2
				X'20'	VTAM
2	2	2	RPLLEN		Length of RPL
4	4	4	RPLRBA		RBA of last record processed
4	4	4	RPLDDDD		DD field
8	8	4	RPLARG		Pointer to search argument
12	C	8	RPLRCD		Record description
12	C	4	RPLAREA		Address of the caller's work area
16	10	4	RPLRLEN		Length of record
20	14	4	RPLBUFL		User buffer size
24	18	4	RPLACB		Address of the caller's ACB
24	18	4	RPLDACB		Catalog compatibility
28	1C	1	RPLSTRID		RPL string identifier
29	1D	1	RPLREQ		Request type*
			RPLPOINT	X'00'	POINT request
			RPLGET	X'04'	GET request
			RPLERASE	X'08'	ERASE request
			RPLPUT	X'0C'	PUT request
			RPLUPDTE	X'0C'	Update request
			RPLINSRT	X'10'	Insert request
			RPLCHECK	X'14'	Check request
			RPLRCLSE	X'18'	RCLOSE request
			RPLENDRQ	X'1C'	ENDREQ request
			RPLFRCIO	X'1C'	FORCIO request
			RPLVERFY	X'20'	VERIFY request
			RPLPUTL	X'24'	PUT locate request
			RPLWRBFR	X'2C'	Write buffer request
30	1E	2	RPLKEYL		Key length
32	20	2	RPLOPTCD		Option codes
*	This value may be altered internally by VSAM, for example, X'24' from application program is changed to X'0C' by IKQRQA				

Figure 5.73 Request Parameter List (RPL) description and format (part 1 of 4)

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
32	20	1	RPLOPT1		First byte of options
			RPLKEY	X'80'	Keyed access
			RPLADR	X'40'	Addressed access
			RPLSEQ	X'20'	Sequential
			RPLDIR	X'10'	Direct processing
			RPLASY	X'08'	Asynchronous
			RPLSKP	X'04'	Skip sequential access
			RPLCNV	X'02'	CNV access (RBA)
			RPLUPD	X'01'	Update
33	21	1	RPLOPT2		Second byte of options
			RPLKGE	X'80'	Search key greater than or equal
			RPLGEN	X'40'	Generic key request
			RPLNSP	X'20'	Note string position
			RPLNUP	X'10'	No update
			RPLLOC	X'08'	Locate mode
			RPLUBF	X'04'	User buffers
			RPLBWD	X'02'	Backward processing
			RPLLRD	X'01'	Last record processing
34	22	1	RPLHLD2	X'FF'	Second test and set byte (RPL not available)
				X'00'	RPL available
35	23	1	RPLHLD	X'FF'	Test and set byte (RPL held - request not completed)
				X'00'	Request completed
36	24	1	RPLFLAG		Flag byte
			RPLECBPR	X'80'	CMS ECB indicator
37	25	3	RPLFDBK		Error feedback area
37	25	1	RPLFDB1		Error class (return) code
37	25	1	RPLRTNCD		Error class code
Error class codes (stored from Register 15)					
			RPLNOERR	X'00'	No error detected
			RPLNORPL	X'04'	RPL held by another request
			RPLLOGER	X'08'	Logical error
			RPLPHYER	X'0C'	Physical error
			RPLVABND	X'3C'	TP I/O prohibited
38	26	1	RPLFDB2		Function type code
			RPLFUPG	X'04'	Upgrade processing
			RPLFAIX	X'02'	AIX processing
			RPLFINC	X'01'	Upgrade set is incorrect
39	27	1	RPLFDB3		Error type code
39	27	1	RPLERRCD		Error type code
39	27	1	RPLFDBKC		Error type code
The following equates are for the various feedback returns that may be set for offset 39 (27). They fall into the three categories shown.					
Returns that are not errors (Register 15 = X'00')					
			RPLEOV	X'04'	EOV called during request
			RPLDPKEY	X'08'	Duplicate key (in AIX record)
			RPLNEWCA	X'10'	Index full - CA split required.

Figure 5.73 Request Parameter List (RPL) description and format (part 2 of 4)

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
Logical errors (register 15 = X'08')					
			RPLEOFDS	X'04'	End of data set encountered
			RPLEODER	X'04'	End of data set encountered
			RPLDUPRC	X'08'	Duplicate record
			RPLDUP	X'08'	Duplicate record
			RPLSEQCK	X'0C'	Sequence error
			RPLNRFND	X'10'	No record found
			RPLNOREC	X'10'	No record found
			RPLEXCTL	X'14'	Data already in exclusive control
			RPLNVOLM	X'18'	Volume or extent unavailable
			RPLNRSPA	X'1C'	No DASD space available
			RPLNOEXT	X'1C'	No DASD space available
			RPLSPACE	X'1C'	No DASD space available
			RPLINRBA	X'20'	Invalid RBA specified
			RPLNKEYR	X'24'	No key range for new record
			RPLNOVIR	X'28'	Insufficient virtual storage
			RPLWRKAS	X'2C'	User's work area not large enough
			RPLCDLOD	X'30'	CDLOAD failure
			RPLVLERR	X'34'	Internal VSAM logic error
			RPLNOPLH	X'40'	PLH in use (no string available)
			RPLNOPEN	X'44'	Access type not requested at Open
			RPLKEYES	X'48'	Keyed request for ESDS
			RPLADRKS	X'4C'	ADR or CNV insert for KSDS
			RPLINERS	X'50'	Illegal ERASE request
			RPLINLOC	X'54'	Illegal locate mode specification
			RPLNOPOS	X'58'	Positioning error
			RPLNGUPD	X'5C'	No valid GET UPD issued
			RPLUPDKC	X'60'	Key change during update
			RPLLENCN	X'64'	Length change for addressed update
			RPLCONOP	X'68'	Improper or conflicting RPL options
			RPLIMRCL	X'6C'	Improper RECLEN specified
			RPLIMGKL	X'70'	Improper generic key length specified
			RPLINLD	X'74'	Illegal request during data set load
			RPLCATLG	X'80'	Internal catalog call failure
			RPLSRLOC	X'84'	Illegal locate mode
			RPLSRADR	X'88'	Illegal request for spanned record
			RPLINCSR	X'8C'	Inconsistent spanned record
			RPLNOBAS	X'90'	No base record
			RPLMAXPT	X'94'	Maximum of pointers exceeded
			RPLNOBUF	X'98'	No buffers available (LSR only)
			RPLINVRR	X'C0'	Invalid relative record number
			RPLRRADR	X'C4'	Illegal address requested (RRDS)
			RPLIPATH	X'C8'	Illegal path access
			RPLINBWD	X'CC'	Illegal backward mode requested

Figure 5.73 **Request Parameter List (RPL) description and format (part 3 of 4)**

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
Physical errors (register 15 = X'0C')					
			RPLRDERD	X'04'	Data read error
			RPLRDERI	X'08'	Index read error
			RPLRDERS	X'0C'	Sequence set read error
			RPLWTERD	X'10'	Data write error
			RPLWTERI	X'14'	Index write error
			RPLWTERS	X'18'	Sequence set write error
40	28	4	RPLCHAIN		Pointer to next RPL
44	2C	1	RPLAIXID		AIX information byte
			RPLAXPKP	X'01'	Prime key pointers are used (base cluster is a KSDS)
45	2D	1			Reserved
46	2E	2	RPLAIXPC		Number of base cluster pointers in the AIX record
48	30	1	RPLXID		Transaction ID
49	31	3			Reserved
52	34	0	RPLEND		End of RPL

Figure 5.73 Request Parameter List (RPL) description and format (part 4 of 4)

Resource Pool Header (RPHD)

The VSAM Resource Pool Header contains general information concerning the VSAM Resource Pool. Its address is stored in the AMBL.

Figure 5.74 shows the description and format.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	2	RPHDASTR		Number of active strings
2	2	2	RPHDMSTR		Maximum number of strings ever simultaneously active
4	4	2			Reserved
6	6	2	RPHDSTNO		Total number of PLHs in the resource pool

Figure 5.74 Resource Pool Header (RPHD) description and format

Resource Sharing Control Block (RSCB)

The Resource Sharing Control Block provides exclusive control facilities for the VSAM shared resources. The VSRT points to the RSCB.

Figure 5.75 shows the description and format.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	4	RSCBRBA		RBA to be held in exclusive control
4	4	7	RSCBDSID		Data set identifier
4	4	4	RSCBCAT		Catalog ACB pointer
8	8	3	RSCBDSCI		CI number of data set component
11	b	1	RSCBCSID		ID of string using field RSCBGATE (for a control area split)
12	C	4	RSCBECB		ECB used by IKQBFA and IKQCAS
12	C	1			Reserved
13	D	1	RSCBSTID		ID of string (set by IKQBFA)
14	E	1	RSCBCOM		Communication byte
			RSCBWAIT	X'80'	Wait flag
15	F	1	RSCBGATE		Exclusive control byte: X'00' = ECB is free X'FF' = ECB is in use

Figure 5.75 **Resource Sharing Control Block (RSCB) description and format**

Track Hold Block (THB)

The THB contains information necessary to do track hold. The PLH points to the THB. The THB is created by Open and released by Close. Figure 5.76 shows the description and format of the THB.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	1	THBID	X'88'	Control block identification
1	1	1	THBFLAG		Flag byte
			THBACTV	X'80'	This THB is active
			THBPSUDO	X'40'	Track hold not issued
			THBREAL	X'20'	Track hold issued
2	2	2	THBLEN		Length of THB
4	4	2			Unused
6	6	2	THBTID		Task ID
8	8	16	THBCCB		CCB area
24	18	8	THBCCW		CCW area
24	18	1	THBCCWOP		CCW operation code

Figure 5.76 **Track Hold Block (THB) description and format (part 1 of 2)**

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
25	19	3	THBCCWAD		CCW argument address
28	1C	1			CCW flags
29	1D	1			Unused
30	1E	2	THBCCWCT		CCW byte count
32	20	20	THBIODRB		IODRB area
32	20	4			Unused
36	24	8	THBARG		MBBCCHHR
44	2C	4			RBA to be held
48	30	4			Unused
52	34	48	THBSAVE		Save area for 12 registers
100	64	4	THBNTHB		Address of the next THB

Figure 5.76 Track Hold Block (THB) description and format (part 2 of 2)

Upgrade Set Block (USB)

The USB declaration maintains information required by PUT and ERASE requests to the base cluster. The USB is created by OPEN (IKQOPNUS). Figure 5.78 shows the description and format.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	1	USBID		USB identifier
			USBIDD	X'E0'	USB equate
1	1	1	USBACT		Active byte, test and set
2	2	2	USBLEN		Length of this block
4	4	2	USBMAXDB		Max. data buffer in upgrade set
6	6	2	USBMAXIB		Max. index buffer in upgrade set
8	8	4	USBWAPTR		Pointer to work area pool
12	C	2	USBMIN		Min. required record length
14	E	2	USBWALEN		Work area length
16	10	4	USBPLH		Pointer to common USB PLH
20	14	4	USBRPL		Pointer to RPL
24	18	4	USBDBHD		Pointer to data buffer header
28	1C	4	USBIBHD		Pointer to index buffer header

Figure 5.77 Upgrade Set Block (USB) description and format (part 1 of 2)

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
			Begin of First/only Index Entry		
32	20		USBAIX		
		4	USBACB		Pointer to ACB
			USBLAST	X'80'	Last entry indicator
36	24	2	USBRKP		Relative key position
38	26	2	USBKL		Key length
			Further Alternate Index entries		
40	28	variable			

Figure 5.77 **Upgrade Set Block (USB) description and format (part 2 of 2)**

VSAM Shared Resource Table (VSRT)

The VSAM Shared Resource Table contains the addresses of the resource part of the resource pool (PLH pool, buffer pool, the RSCB, and CPA pool), together with the addresses of various control blocks built during processing of the BLDVRP macro.

The VSRT is contained in the phase IKQVRT and is loaded by means of CDLOAD when required.

Figure 5.78 shows the format and description.

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
0	0	1	VSRTBKID	X'15'	Control block identifier
1	1	1			Reserved
2	2	2	VSRTLEN		Length of VSRT
4	4	4	VSRTID		Control block name 'VSRT'
8	8	4			Reserved
12	C	1	VSRTFLG1		Flag byte 1
			LSR	X'40'	LSR pool indicator
13	D	1	VSRTFLG2		Flag byte 2
14	E	2	VSRTUNCT		Number of data sets opened with the LSR option
16	10	4	VSRTECB		ECB for the VSRT
18	12	1	VSRTCOM		Communication byte
			VSRTWAIT	X'80'	Wait flag
19	13	1	VSRTGATE		Exclusive control gate: X'00' = ECB is free X'FF' = ECB is in use

Figure 5.78 **VSAM Shared Resource Table (VSRT) description and format (part 1 of 2)**

Offset Dec	Offset Hex	Bytes	Field Name	Hex. Digit	Description
20	14	4	VSRTRP		Address of the control block part of the resource section of the VRP
24	18	4	VSRTRLEN		Length of the control block part
28	1C	4	VSRTPLHA		Address of the PLH pool in the resource section
32	20	1	VSRTSTRN		Total number of PLHs required for all data sets sharing the resource pool
33	21	1	VSRTKL		Maximum key length of the data sets sharing the resource pool
34	22	2	VSRTPLHL		Length of each PLH in the PLH pool
36	24	4	VSTRSCB		Address of the RSCB
40	28	4	VSRTBUFH		Address of buffer pool
44	2C	4	VSRT1STB		Address of first buffer
48	30	4	VSRTMBSZ		Maximum buffer size in buffer pool
52	34	2	VSRTSPNO		Number of subpools in buffer pool
54	36	2	VSRTBFNO		Number of buffers in buffer pool
56	38	4	VSRTBFLN		Length of the buffer part of the resource section
60	3C	4	VSRTCPAH		Address of the channel program area pool in the resource section
64	40	4	VSRTCPAL		Length of the channel program area pool
68	44	4	VSRTTHBA		Address of the THB pool
72	48	4	VSRTTHBL		Length of the THB pool

Figure 5.78 VSAM Shared Resource Table (VSRT) description and format (part 2 of 2)

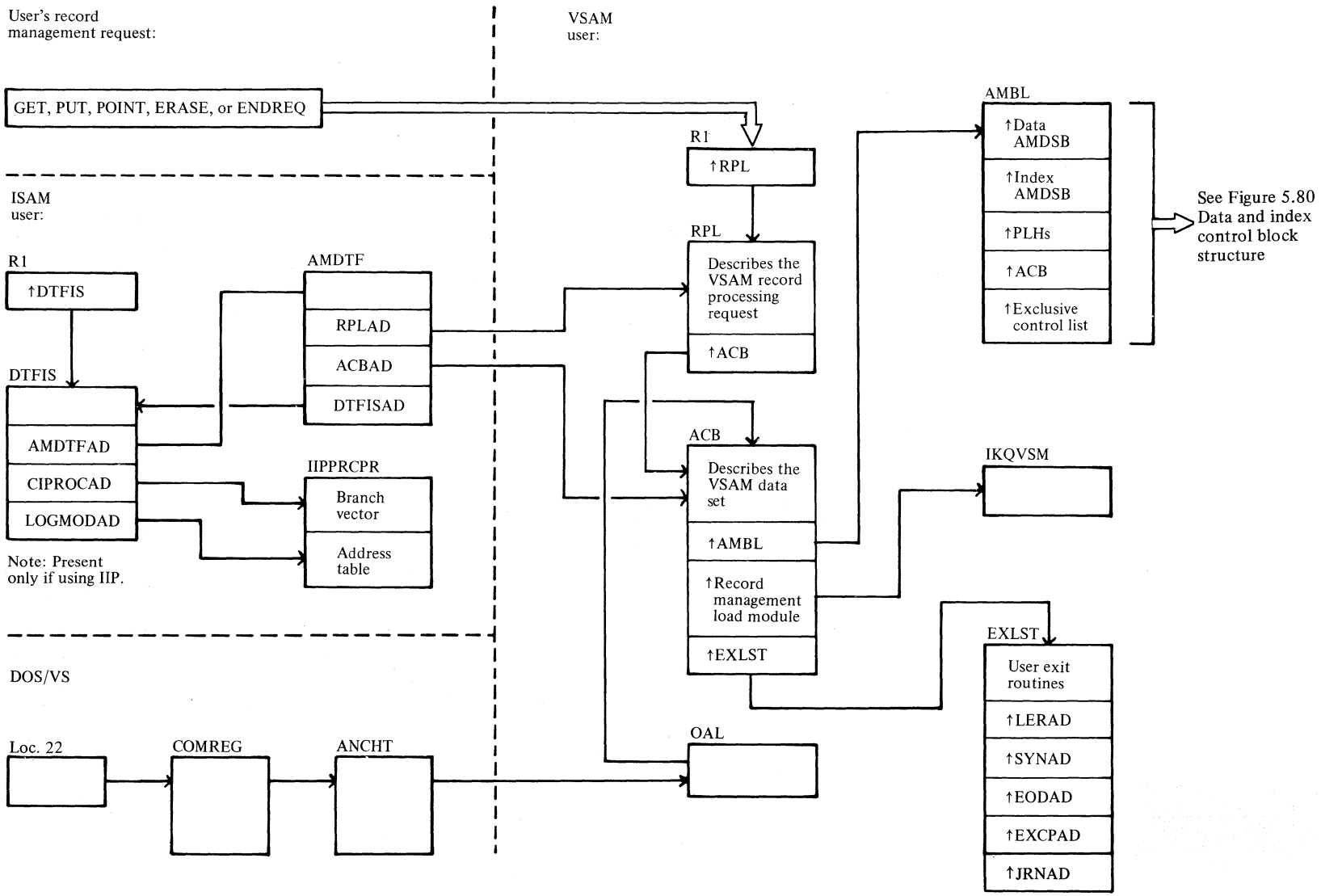


Figure 5.79 **VSAM control block structure for a key-sequenced data set**

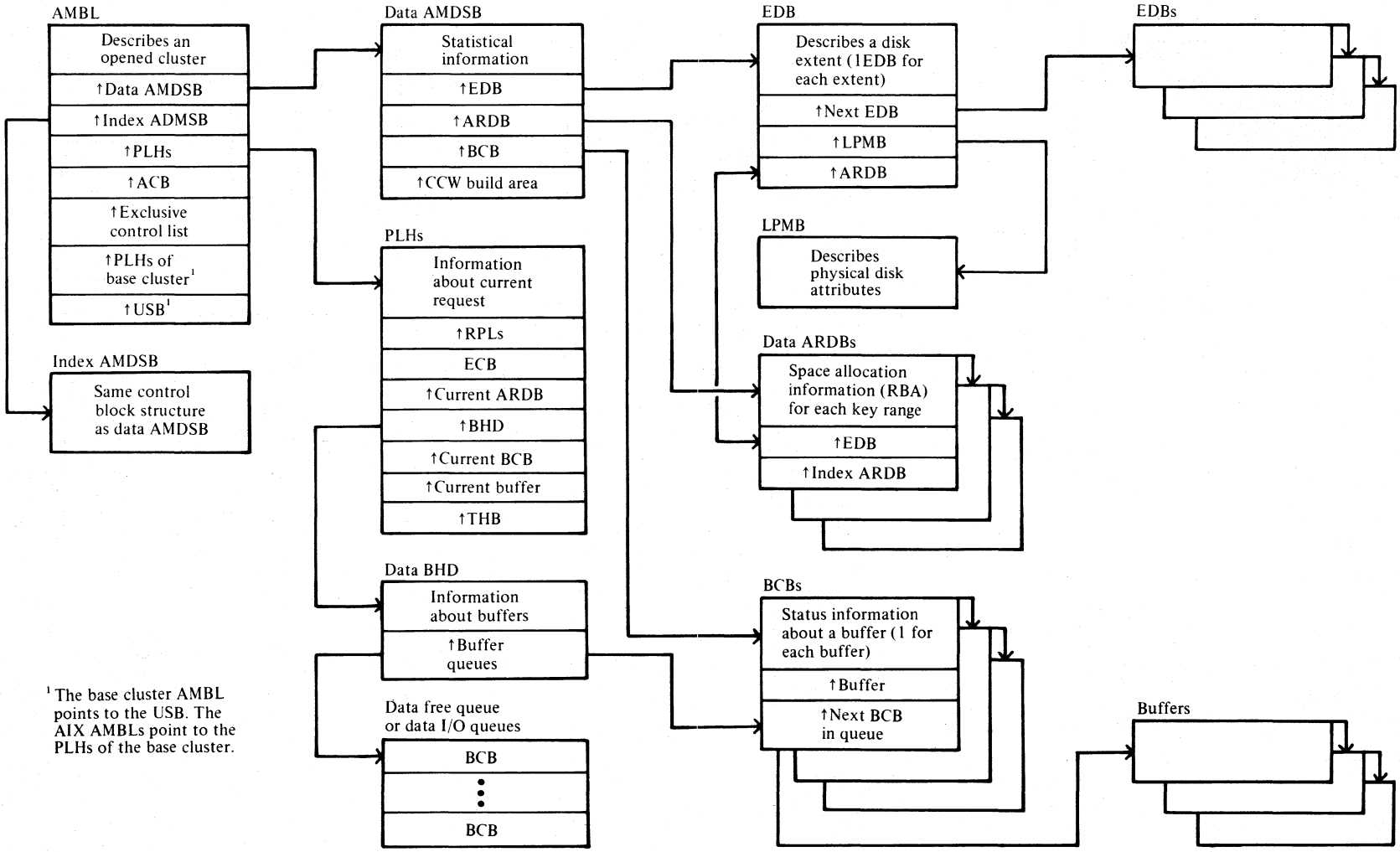


Figure 5.80 **Data and index control block structure**

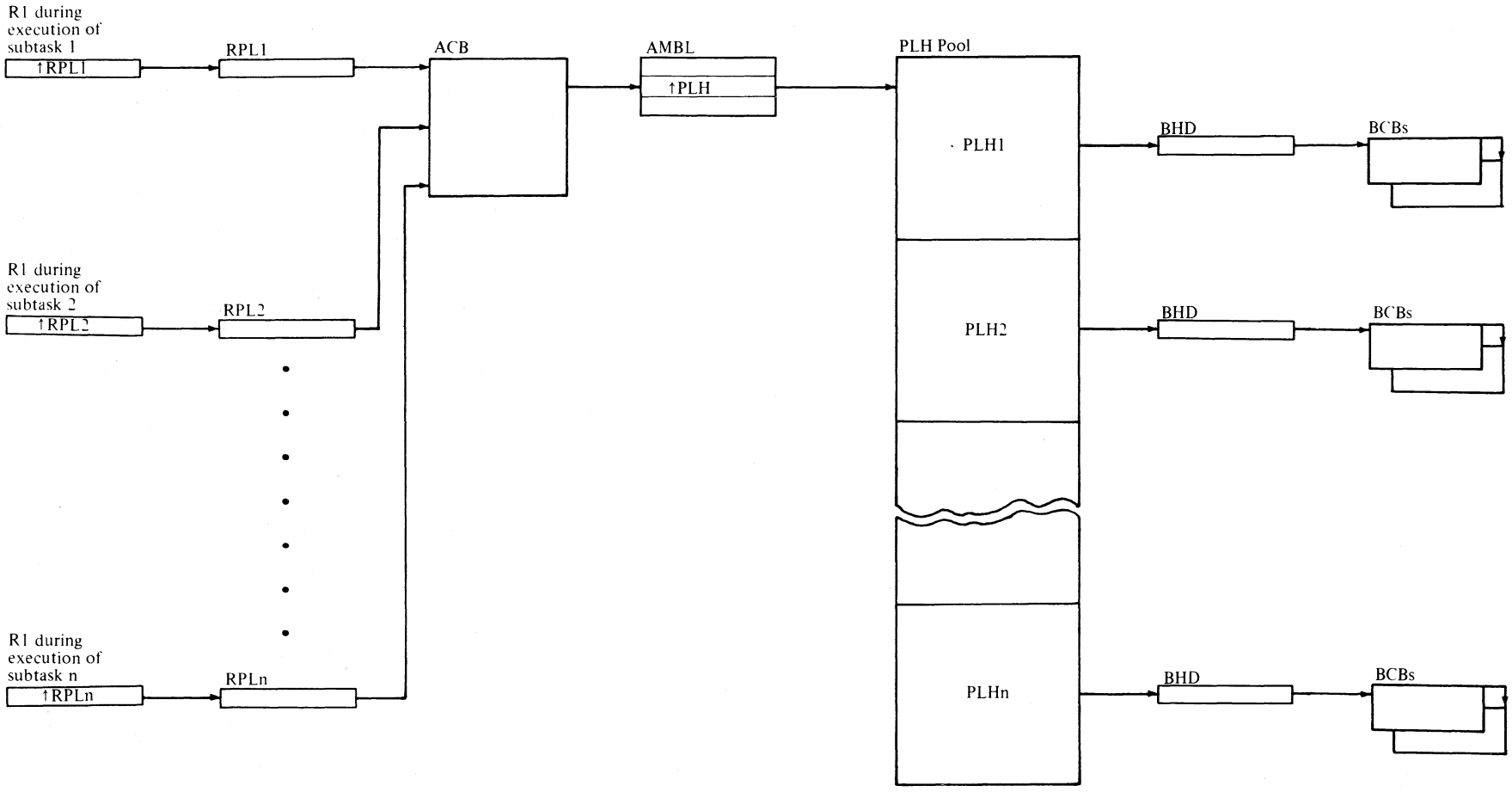


Figure 5.81 **Multiple string control block structure**

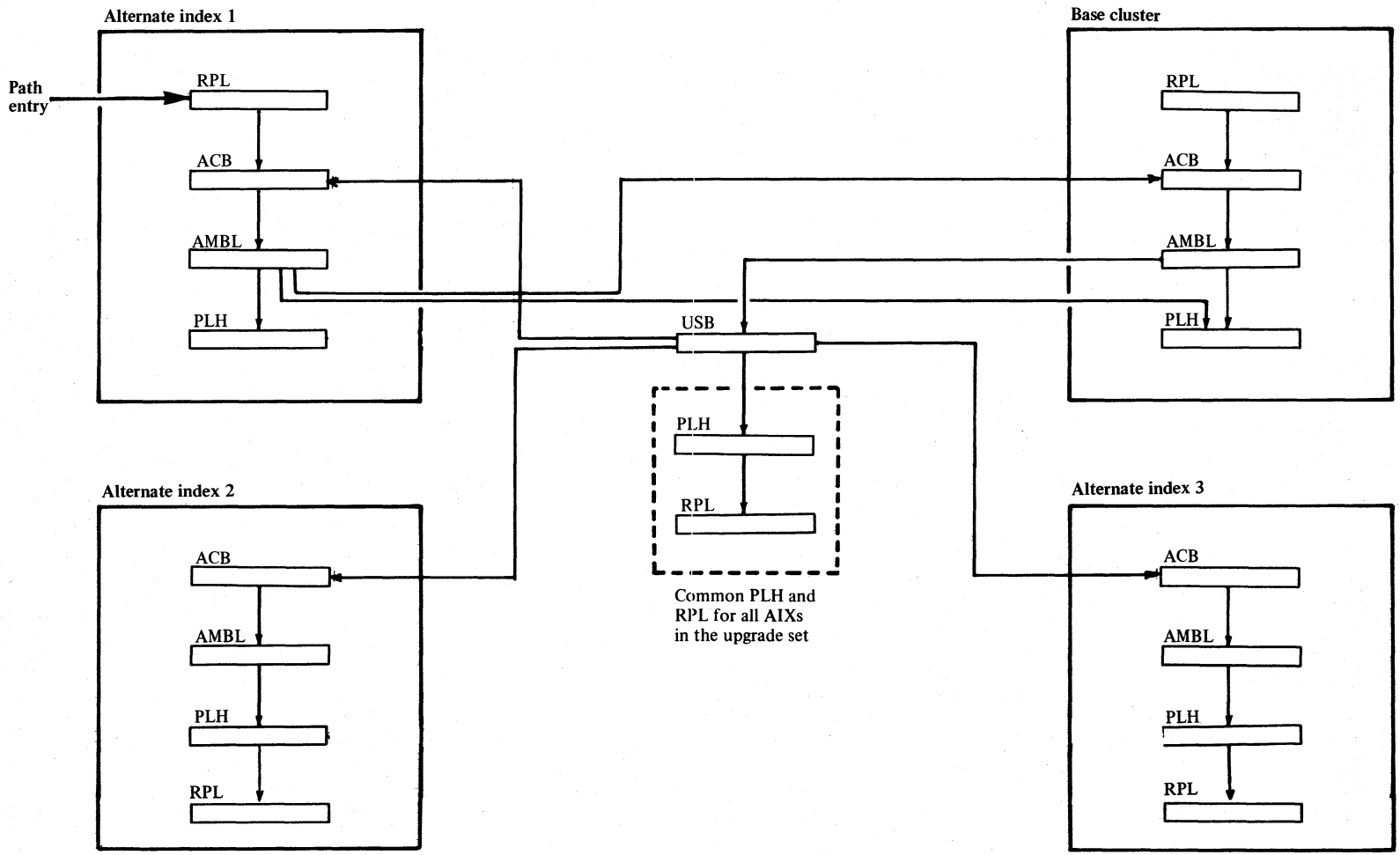


Figure 5.82 **Base cluster to alternate index control block structure**

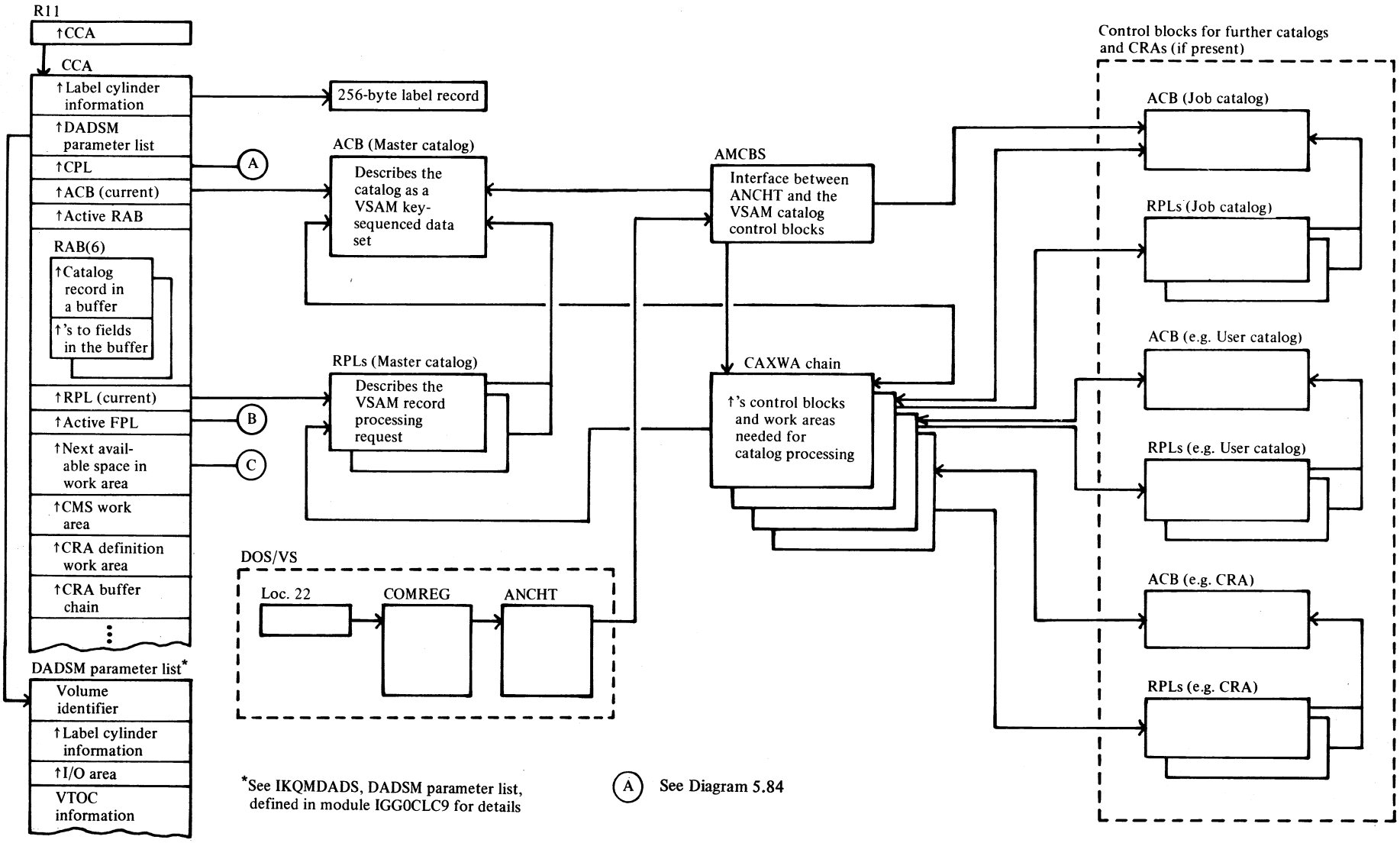


Figure 5.83 **Catalog management control blocks**

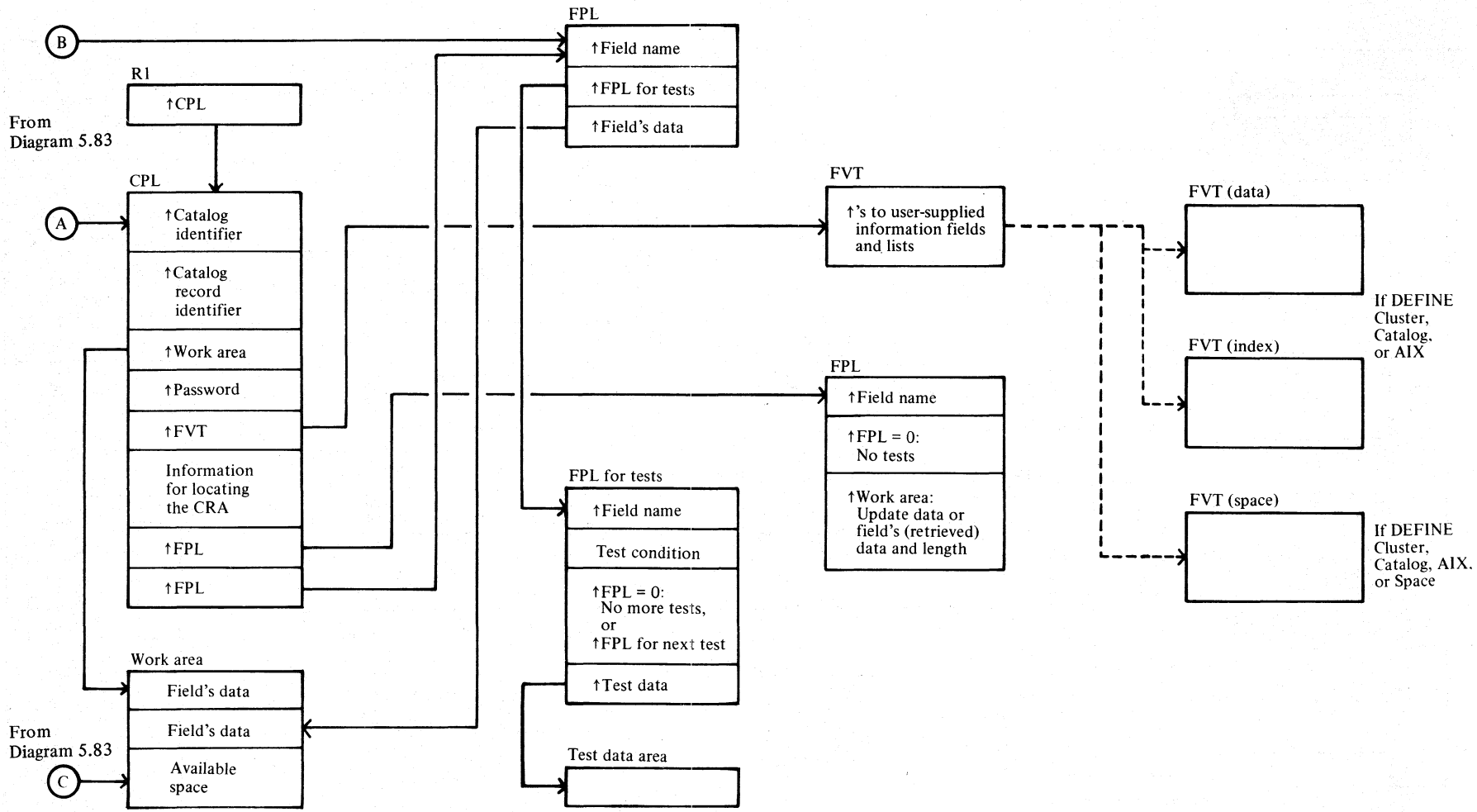


Figure 5.84 **Caller-supplied control blocks for catalog management**

Section 6. Diagnostic Aids

This chapter provides several aids that can be useful when trying to diagnose difficulties with VSAM modules. These aids include:

- A list of macro instructions (Figure 6.1) issued by VSAM users, modules or other macros and their use.
- Cross reference tables (Figures 6.2 and 6.3) showing the VSAM modules and the macros they issue.
- A description of the Catalog Communication Area, Register Save Area and a list of error codes (Figure 6.4), set in the CCA by catalog mangement modules, together with the reason codes belonging to each error code.
- A list of return codes (Figure 6.5) set in register 15 which indicate DADSM conditions when processing is completed.
- A list of error codes (Figure 6.6) set in the RPL which indicate record management errors. The list shows also the relationship between internal and external error codes.
- A list of error codes (Figure 6.7) showing record management modules and the error code(s) they might issue.
- A list of error codes (Figure 6.8) showing record management modules and the error code(s) they might issue when manipulating control blocks.
- A list of error codes (Figure 6.9) which may be issued by OPEN modules.
- A list of error codes (Figure 6.10) which may be issued by CLOSE and TCLOSE modules.
- A list of error codes (Figure 6.11) which may be issued by the SHOWCAT module.
- A list of error codes (Figure 6.12) which may be issued by the BLDVRP and DELVRP modules.
- A description of service aid phases and how to use them.

Additional Aids

Further aids can be found in other parts of the book and in the program listings. These include:

- Register contents on entry to a module, which are under *Input* in the module prologues.
- Use of registers and equated names for registers, which can be found under *Notes* in the module prologues.
- Error codes, which are under *Exit-Error* in the module prologues.
- A list, which is in the *Directory*, of modules, their component, their entry points, and their associated method of operation and program structure diagrams.
- A cross-reference list, which is in the *Directory*, of catalog external entry points and their associated modules.

Macro-to-Module Relationships

The following list in Figure 6.1 contains the macro instructions issued by VSAM users, modules, or other macros. Their types are identified as follows:

G – generating macro
SA – DOS/VS action macro
M – mapping macro
I – internal (called by another macro)
A – VSAM action macro
S – copy source book macro

Macro	Type	SVC	Use
ACB	G		Generate an ACB
ASYSCOM	SA		Get address of systems communications region
CANCEL	SA		Cancel a task
CATLG	A		Load address of catalog parameter list (CTGPL) into R1 and invoke catalog management
CCB	G		Build a CCB
CDLOAD	SA	65	Load module(s)
CLOSE	SA	2	Disconnect a user's program from a VSAM data set
COMRG	SA	33	Get DOS/VS communication region address
DEQB	SA		Free B-transient
DTFCN	SA		SYSLOG DTF
ENDREQ	SA		Free a PLH and terminate processing on associated string
ENQB	SA		Hold B-transient
EOJ	SA		End of job
ERASE	SA		Delete a record
EXCP	SA	0	Execute channel program
EXLST	G		Generate EXLST
FREEVIS	SA	62	Free virtual storage
GENCB	A		Generate a control block
GET	SA		Retrieve a record
GETVIS	SA	61	Get virtual storage
IGGCAXWA	M		Map catalog auxiliary work area (CAXWA)
IGGCCA	M		Map catalog communications area (CCA)
IGGMCDCL	M		Issue the following macros to define the commonly used declarations for VSAM catalog management modules: IGGCAXWA, IGGCCA, IGGMCTRC, IKQACB, IKQAMCBS, IKQCOMRG, IKQCTGFL, IKQCTGFV, IKQCTGPL, IKQVRGN
IGGMCMDM	M		Map the VSAM catalog management commonly used record structures
IGGMCMWA	M		Map the VSAM catalog management services work area
IGGMCTRC	M		Map catalog return codes
IGGMDLWA	M		Delete work area layout
IGGMDRWA	M		Map the VSAM catalog DSCB read-in work area
IGGMEND	G		Generate exit code at the end of catalog management modules
IGGMGVO	M		Map the volume information group occurrence
IGGMNAME	I		Generate catalog module name for error and reason codes
IGGMODUL	G		Generate header code for catalog modules

Figure 6.1 **Macro types and uses (part 1 of 4)**

Macro	Type	SVC	Use
IGGMPROC	G		Generate header code for catalog internal procedures
IGGMSAWA	M		Map the VSAM catalog management suballocate work area
IGGMUPDE	M		Issue IGGMVEDC, IGGMCDCL, IGGMCMDM, IKQAMDSB, and IGGMSAWA to define the commonly used declarations for VSAM catalog management Update-Extend modules
IGGMVEDC	M		Map the volume catalog record
IIPAMDTF	G/M		Generate/map AMDTF table
IIPDTF	G/M		Generate/map DTF table
IIPPRAT	G/M		Generate/map address list
IKQACB	M		Map ACB
IKQACBG	I		Generate ACB (called by IKQACB)
IKQAIR	M		Map alternate index record
IKQAMBL	M		Map AMBL
IKQAMCBS	M		Map AMCBS
IKQAMDSB	M		Map AMDSB
IKQARDB	M		Map ARDB
IKQAREX	M		Map EXLST argument entry
IKQARGH	M		Map argument header
IKQBHD	M		Map buffer header
IKQBKPHD	M		Map header for CCW area
IKQBLARD	G		Build an ARDB
IKQBUFE	M		Map BCB
IKQCBMTB	G		Define table of constants for control block manipulation modules
IKQCB1	I		Transform operands for control block manipulation macro instructions GENCB, TESTCB, MODCB, SHOWCB, IKQCB2, and IKQERMAC
IKQCB2	I		Scan keywords and generate code for control block manipulation macro instructions GENCB, TESTCB, MODCB, SHOWCB, IKQCB1, and IKQERMAC
IKQCCB	M		Map CCB
IKQCCBCW	M		Map CCB
IKQCCW	M		Map CCW
IKQCGETC	S		Obtain storage in which to copy old ARDB
IKQCIW	M		Map control interval split work area
IKQCLCOR	S		Get address of space in which to build EDB(s)
IKQCLNUP	S		Disconnect ACB and AMBL
IKQCLRLS	S		Free storage obtained by Open and/or EOV
IKQCLWA	M		Close work area

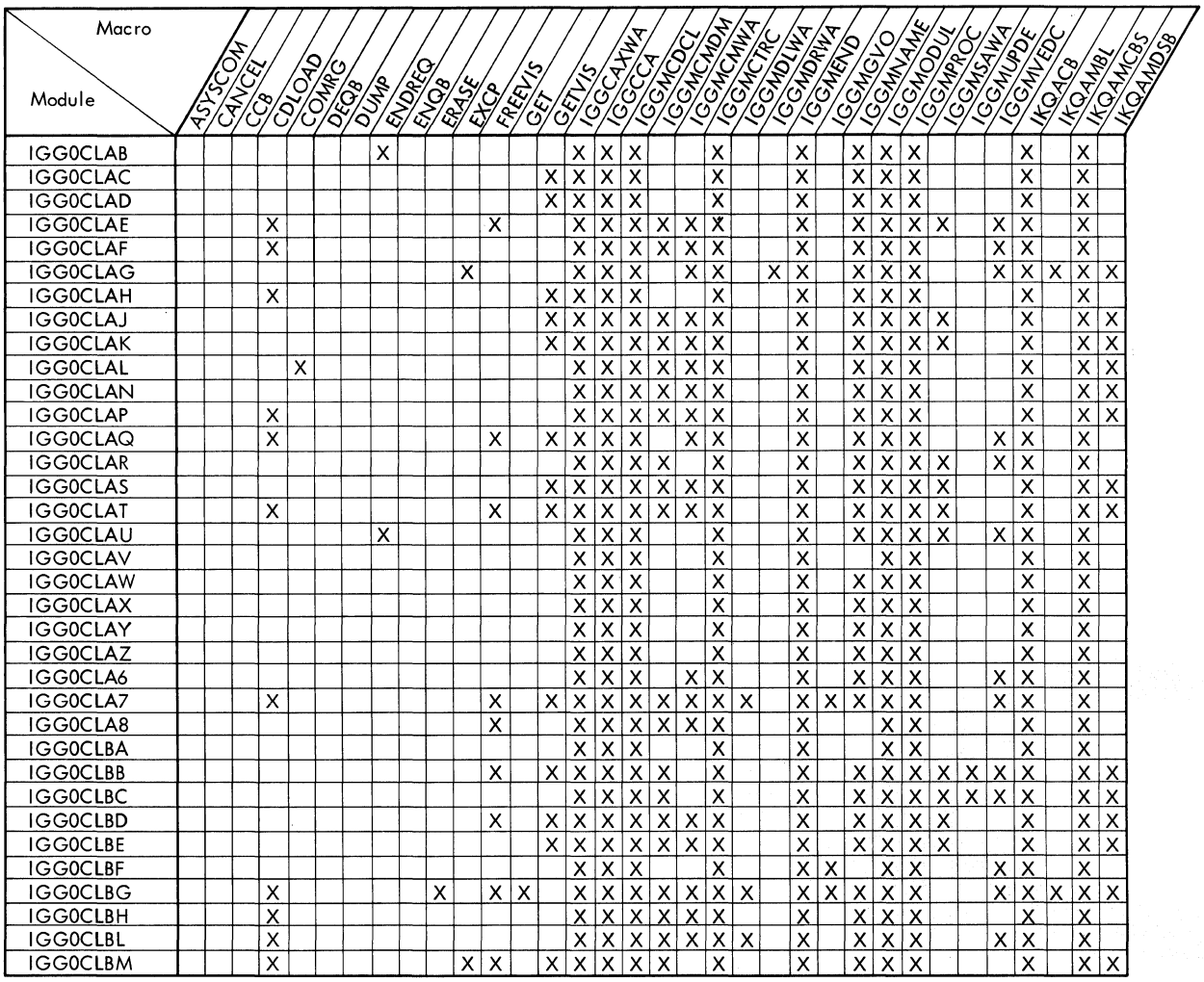
Figure 6.1 Macro types and uses (part 2 of 4)

Macro	Type	SVC	Use
IKQCOMB	G		Generate a combination name entry for the VSAM catalog dictionary
IKQCOMRG	M		Map DOS/VS communication region
IKQCTGFL	M		Map field parameter list (CTGFL)
IKQCTGFV	M		Map catalog field vector table (CTGFV)
IKQCTGPL	M		Map catalog parameter list (CTGPL)
IKQDEVT	A	65	Read label cylinder and/or determine the device type for the file-ID (IKQDEVT uses CDLOAD)
IKQEDB	M		Map EDB
IKQEDBLD	G		Build EDB
IKQEQU	M		Map register equates
IKQERC	M		Internal error codes equate
IKQERMAC	I		Issue M-notes (assembler macro error messages) for control block manipulation macro instructions GENCB, TESTCB, MODCB, SHOWCB, IKQCB1, and IKQCB2
IKQEXLG	I		Generate EXL (called by IKQEXL)
IKQEXLST	M		Map EXLST
IKQEXP	M		Description of EXPAD parameter list
IKQFCDB	M		Map CCW blocks in CCW pool
IKQFMT1	M		Map format-1 DSCB
IKQFMT3	M		Map format-3 DSCB
IKQFMT4	M		Map format-4 DSCB
IKQFNDLB	S		Find LUB (logical unit block) for symbolic unit
IKQIOARG	M		Map DASD address
IKQIODRB	M		Map I/O driver block
IKQIORQU	M		Map register equates
IKQIOWKA	M		Map I/O work area in PLH
IKQIXHDR	M		Map index record header
IKQJIB	M		Map JIB (job information block)
IKQJRNDS	M		Parameter list for journalling
IKQLBRC	M		Map label cylinder record
IKQLPMB	M		Map LPMB
IKQLUB	M		Map locial unit block
IKQMDADS	M		Map DADSM parameter list (interface block to DADSM)
IKQMSGPL	M		OPEN/CLOSE message primary list
IKQOAL	M		Open ACB list
IKQOPCLR	M		Register equates
IKQOPCLW	M		Map of common section of work area
IKQOPLCT	M		Map fields located by catalog
IKQOPNWA	M		Map Open work area

Figure 6.1 Macro types and uses (part 3 of 4)

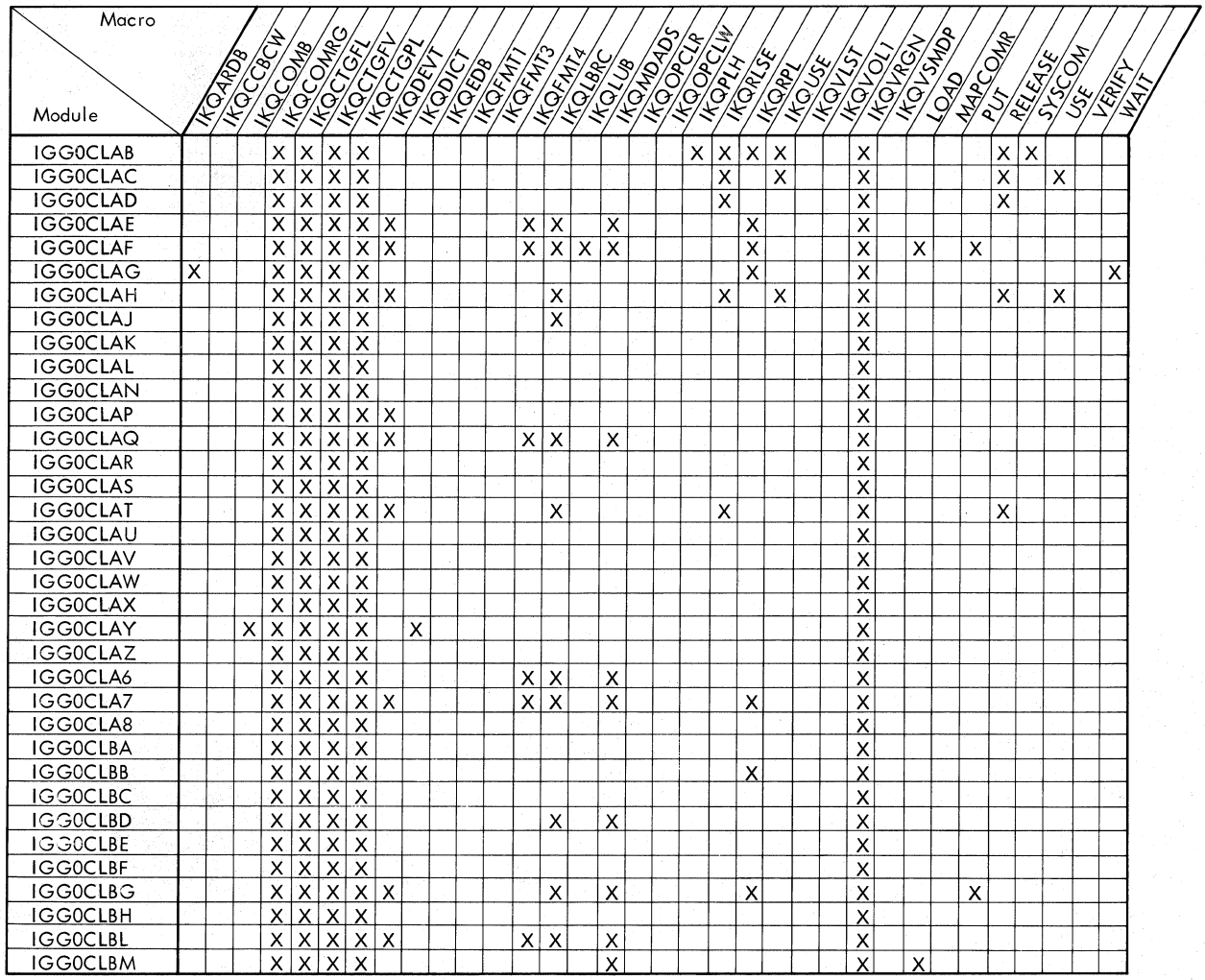
Macro	Type	SVC	Use
IKQPLH	M		Map PLH
IKQPUB	M		Map physical unit block
IKQRDF	M		Map RDF and CIDF fields
IKQRLSE	M	64	Dequeue a system resource by means of a RELEASE macro
IKQRPL	M		Map RPL
IKQRPLG	I		Generate RPL (called by IKQRPL)
IKQRQM	G		Generate modules IKQRQA and IKQRQB
IKQTHB	M		Map THB
IKQUSB	M		Upgrade set block
IKQUSE	A	63	Enqueue a system resource by means of a USE macro
IKQVLST	M		Map list of volume unit, symbolic unit, and volume time stamp
IKQVOL1	M		Map volume-1 label
IKQVRGN	M		Map anchor table
IKQVSMDP	A		Map VSAM dump
LOAD	A		Load a phase
MAPCOMR	M		Map partition COMREG layout
MAPPIB	M		Map program information block
MODCB	A		Modify a control block
OPEN	SA	2	Connect a user's program to a VSAM data set
POINT	SA		Position VSAM at a record
POST	SA		Post an ECB
PUT	SA		Store a new or updated record
RELEASE	A	64	Dequeue a system resource
RPL	G		Generate an RPL
SHOWCB	A		Display a control block
SYSCOM	M		Map system communication region layout
TCLOSE	SA		Purge buffer and update catalog (no disconnect)
TESTCB	A		Test a control block
USE	A	63	Enqueue a system resource
VERIFY	A		Build calling sequence for VSAM function VERIFY
WAIT	SA	7	Wait on a CCB for I/O to complete

Figure 6.1 **Macro types and uses (part 4 of 4)**



Module \ Macro	ASYSCOM	CANCEL	CCB	CDLOAD	COMRG	DEQB	DUMP	ENDREQ	ENQB	ERASE	EXCP	FREEVIS	GET	GETVIS	IGGCAXWA	IGGCCA	IGGMCDCL	IGGMCMDM	IGGMCMWA	IGGMCTRC	IGGMDLWA	IGGMDRWA	IGGMEND	IGGMGVO	IGGMNAME	IGGMODUL	IGGMPROC	IGGMSAWA	IGGMUPDE	IGGMVEDC	IKQACB	IKQAMBL	IKQAMCBS	IKQAMDSB
IGG0CLAB								X						X	X	X			X			X		X	X	X				X		X	
IGG0CLAC													X	X	X	X			X			X		X	X	X				X		X	
IGG0CLAD													X	X	X	X			X			X		X	X	X				X		X	
IGG0CLAE			X								X			X	X	X	X	X	X			X		X	X	X	X		X	X		X	
IGG0CLAF			X											X	X	X	X	X	X			X		X	X	X				X		X	
IGG0CLAG										X				X	X	X		X	X		X	X		X	X	X			X	X	X	X	X	X
IGG0CLAH			X										X	X	X	X			X			X		X	X	X				X		X	
IGG0CLAJ													X	X	X	X	X	X	X			X		X	X	X	X			X		X	X
IGG0CLAK													X	X	X	X	X	X	X			X		X	X	X	X			X		X	X
IGG0CLAL				X										X	X	X	X	X	X			X		X	X	X				X		X	X
IGG0CLAN														X	X	X	X	X	X			X		X	X	X				X		X	X
IGG0CLAP			X											X	X	X	X	X	X			X		X	X	X				X		X	X
IGG0CLAQ			X								X		X	X	X	X		X	X			X		X	X	X			X	X		X	
IGG0CLAR														X	X	X	X		X			X		X	X	X	X		X	X		X	
IGG0CLAS													X	X	X	X	X	X	X			X		X	X	X	X			X		X	X
IGG0CLAT			X								X		X	X	X	X	X	X	X			X		X	X	X	X			X		X	X
IGG0CLAU							X							X	X	X			X			X		X	X	X	X		X	X		X	
IGG0CLAV														X	X	X			X			X			X	X				X		X	
IGG0CLAW														X	X	X			X			X		X	X	X				X		X	
IGG0CLAX														X	X	X			X			X		X	X	X				X		X	
IGG0CLAY														X	X	X			X			X		X	X	X				X		X	
IGG0CLAZ														X	X	X			X			X		X	X	X				X		X	
IGG0CLA6														X	X	X		X	X			X		X	X	X			X	X		X	
IGG0CLA7			X								X		X	X	X	X	X	X	X	X		X	X	X	X	X			X	X		X	
IGG0CLA8											X			X	X	X	X	X	X			X			X	X				X		X	
IGG0CLBA														X	X	X			X			X			X	X				X		X	
IGG0CLBB											X		X	X	X	X	X		X			X		X	X	X	X	X	X	X		X	X
IGG0CLBC														X	X	X	X		X			X		X	X	X	X	X	X	X		X	X
IGG0CLBD											X		X	X	X	X	X	X	X			X		X	X	X	X			X		X	X
IGG0CLBE													X	X	X	X	X	X	X			X		X	X	X	X			X		X	X
IGG0CLBF														X	X	X			X			X	X		X	X			X	X		X	
IGG0CLBG			X						X		X	X		X	X	X	X	X	X	X		X	X	X	X	X			X	X	X	X	X	X
IGG0CLBH			X											X	X	X	X	X	X			X		X	X	X				X		X		
IGG0CLBL			X											X	X	X	X	X	X	X		X		X	X	X			X	X		X		
IGG0CLBM			X							X	X		X	X	X	X	X		X			X		X	X	X				X		X	X

Figure 6.2 **Macro-to-module relationships for catalog and DADSM components (part 1 of 4)**



Module \ Macro	IKQARDB	IKQCCBCW	IKQCOMB	IKQCOMRG	IKQCTGFL	IKQCTGFV	IKQCTGPL	IKQDEVT	IKQDICT	IKQEDB	IKQFMT1	IKQFMT3	IKQFMT4	IKQLBRC	IKQLUB	IKQMDADS	IKQOPCLR	IKQOPCLW	IKQPLH	IKQRLSE	IKQRPL	IKQUSE	IKQV1ST	IKQVOL1	IKQVRGN	IKQVSMDP	LOAD	MAPCOMR	PUT	RELEASE	SYSCOM	USE	VERIFY	WAIT
IGG0CLAB				X	X	X	X											X	X	X	X			X					X	X				
IGG0CLAC				X	X	X	X												X		X			X					X		X			
IGG0CLAD				X	X	X	X												X					X					X					
IGG0CLAE				X	X	X	X	X				X	X		X					X				X										
IGG0CLAF				X	X	X	X	X				X	X	X	X					X				X		X		X						
IGG0CLAG	X			X	X	X	X													X				X										X
IGG0CLAH				X	X	X	X	X					X						X		X			X					X		X			
IGG0CLAJ				X	X	X	X						X											X										
IGG0CLAK				X	X	X	X																	X										
IGG0CLAL				X	X	X	X																	X										
IGG0CLAN				X	X	X	X																	X										
IGG0CLAP				X	X	X	X	X																X										
IGG0CLAQ				X	X	X	X	X				X	X		X									X										
IGG0CLAR				X	X	X	X																	X										
IGG0CLAS				X	X	X	X																	X										
IGG0CLAT				X	X	X	X	X					X						X					X					X					
IGG0CLAU				X	X	X	X																	X										
IGG0CLAV				X	X	X	X																	X										
IGG0CLAW				X	X	X	X																	X										
IGG0CLAX				X	X	X	X																	X										
IGG0CLAY			X	X	X	X	X		X															X										
IGG0CLAZ				X	X	X	X																	X										
IGG0CLA6				X	X	X	X					X	X		X									X										
IGG0CLA7				X	X	X	X	X				X	X		X					X				X										
IGG0CLA8				X	X	X	X																	X										
IGG0CLBA				X	X	X	X																	X										
IGG0CLBB				X	X	X	X													X				X										
IGG0CLBC				X	X	X	X																	X										
IGG0CLBD				X	X	X	X						X		X									X										
IGG0CLBE				X	X	X	X																	X										
IGG0CLBF				X	X	X	X																	X										
IGG0CLBG				X	X	X	X	X					X		X					X				X			X							
IGG0CLBH				X	X	X	X																	X										
IGG0CLBL				X	X	X	X	X				X	X		X									X										
IGG0CLBM				X	X	X	X								X									X		X								

Figure 6.2 **Macro-to-module relationships for catalog and DADSM components (part 2 of 4)**

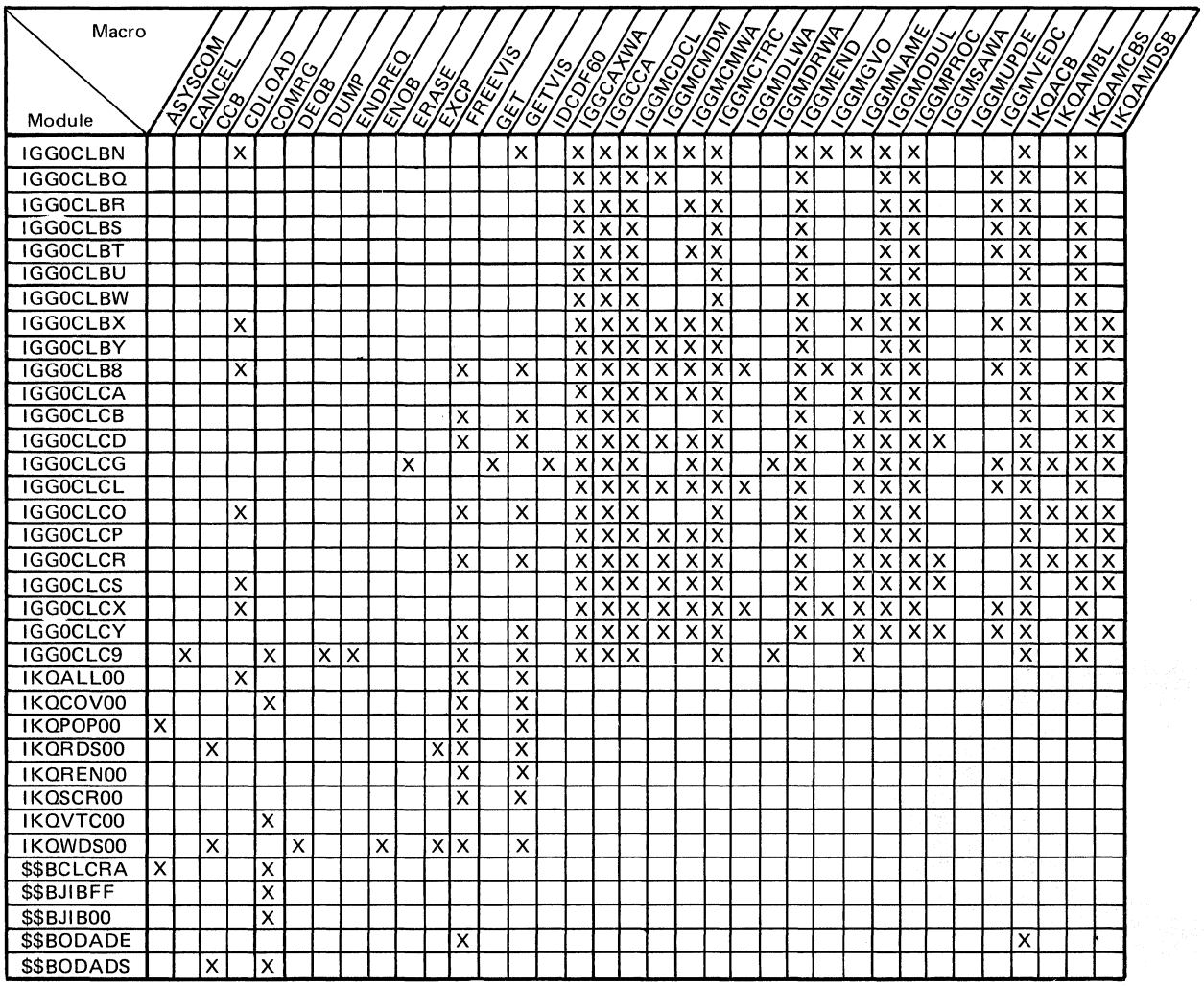


Figure 6.2 **Macro-to-module relationships for catalog and DADSM components (part 3 of 4)**

Module \ Macro	IKQARDB	IKQCCBCW	IKQCOMB	IKQCOMRG	IKQCTGFL	IKQCTGFV	IKQCTGPL	IKQDEVT	IKQDICT	IKQEDB	IKQFMT1	IKQFMT3	IKQFMT4	IKQLBRC	IKQLUB	IKQMDADS	IKQOPCLR	IKQOPCLW	IKQPLH	IKQRLSE	IKQRPL	IKQUSE	IKQVLST	IKQVOL1	IKQVRGN	IKQVSMDP	LOAD	MAPCOMR	PUT	RELEASE	SYSCOM	USE	VERIFY	WAIT
IGG0CLBN			X	X	X	X	X					X		X									X										
IGG0CLBQ			X	X	X	X																	X										
IGG0CLBR			X	X	X	X																	X										
IGG0CLBS			X	X	X	X																	X										
IGG0CLBT			X	X	X	X																	X										
IGG0CLBU			X	X	X	X								X									X										
IGG0CLBW			X	X	X	X																	X										
IGG0CLBX			X	X	X	X	X					X											X										
IGG0CLBY			X	X	X	X																	X										
IGG0CLB8			X	X	X	X	X					X		X					X				X										
IGG0CLCA			X	X	X	X																	X										
IGG0CLCB			X	X	X	X																	X										
IGG0CLCD			X	X	X	X																	X										
IGG0CLCG	X		X	X	X	X								X					X				X	X			X						
IGG0CLCL			X	X	X	X				X		X	X		X			X		X			X					X		X			
IGG0CLCO	X		X	X	X	X	X		X				X	X	X			X		X	X		X					X		X			
IGG0CLCP			X	X	X	X																	X										
IGG0CLCR	X		X	X	X	X						X			X				X				X										
IGG0CLCS			X	X	X	X	X					X	X		X				X				X										
IGG0CLCX			X	X	X	X												X	X	X			X					X		X			
IGG0CLCY			X	X	X	X																	X										
IGG0CLC9			X	X	X	X									X			X	X	X			X	X	X		X	X		X	X		
IKQALL00			X									X	X		X																		
IKQCOV00			X							X	X	X			X																		
IKQPOP00										X	X		X		X															X			
IKQRDS00		X										X			X							X											X
IKQREN00										X		X			X																		
IKQSCR00										X		X			X																		
IKQVTC00			X												X																		
IKQWDS00		X										X			X																		X
$$BCLCRA																		X		X					X		X	X	X				
$$BJIBFF																																	
$$BJIB00															X																		
$$BODADE																X																	
$$BODADS																																	

Figure 6.2 **Macro-to-module relationships for catalog and DADSM components (part 4 of 4)**

Module \ Macro	ACB	ASYSCOM	CANCEL	CATLG	CCB	CDLOAD	COMRG	DEQB	DTFCN	ENQB	EOJ	EXCP	FREEVIS	GENCB	GET	GETVIS	IGGCAXWA	IGGCCA	IGGMCMWA	IGGMDRWA	IIPAMDTF	IIPDTF	IIPPRAT	IKQACB	IKQAIR	IKQAMBL	IKQAMCBS	IKQAMDSB	IKQARDB	IKQAREX	IKQARGH	IKQAUHDR	IKQBHD	IKQBKPHD	IKQBLARD	IKQBSPH	IKQBUFE
IIPAMT00					X																X															
IIPBMR00			X		X							X																								
IIPCLS00					X	X															X	X	X	X		X		X								
IIPOPN00					X	X								X		X					X	X	X	X		X		X								
IIPPRCMR			X		X			X		X		X									X															
IIPPRCPR					X	X									X						X	X	X													
IKQAIX																								X	X	X		X								
IKQBFA							X																	X		X		X					X	X		X	X
IKQBFB																								X		X		X					X	X		X	X
IKQBLD																																					
IKQBRP						X							X			X																	X	X		X	X
IKQCAS						X																				X		X	X							X
IKQCIR						X																				X		X	X							X
IKQCIS						X							X			X										X		X	X				X			X
IKQCLCAT				X		X							X			X								X		X		X	X				X			
IKQCLEAN					X		X				X	X	X			X																				
IKQCLO00						X							X											X		X		X	X					X		X
IKQCLOCL													X											X		X										
IKQCLOVY																								X		X		X					X			X
IKQDCN									X																											
IKQDRP						X							X																					X		
IKQDUMP					X	X	X					X												X		X		X	X				X	X		X
IKQDUMPC							X										X	X	X	X				X			X									
IKQEDX				X		X	X						X			X										X		X	X						X	
IKQEOV			X			X	X						X			X										X		X	X							
IKQERH						X																		X		X		X								
IKQERX						X																		X		X										
IKQGEN													X			X								X		X		X		X	X	X				
IKQGNX00																										X		X	X				X			X
IKQGPT																										X		X	X							X
IKQIOA00						X										X								X		X		X					X	X		X
IKQIOB00																								X				X					X	X		X
IKQIXE00				X		X							X			X								X		X		X								X
IKQIXF00																										X		X								
IKQIXS00																												X								
IKQJIBSM													X			X										X		X	X							
IKQJRN						X										X								X		X		X								
IKQKRD																												X	X							X
IKQLAB							X					X	X			X								X												
IKQLASMD						X	X						X			X	X							X												
IKQLCD																										X		X	X							X
IKQLCN																												X								
IKQLCP																										X		X	X				X			X
IKQMDY																										X		X								X
IKQNCA00						X							X			X								X		X		X	X				X			X

Figure 6.3 **Macro-to-module relationships for all VSAM modules except catalog and DADSM (part 1 of 6)**

Module \ Macro	IKQCBMTB	IKQCB1	IKQCB2	IKQCCB	IKQCCBCW	IKQCCW	IKQCGETC	IKQCIW	IKQCLCOR	IKQCLRLS	IKQCLWA	IKQCOMRG	IKQCTGFL	IKQCTGFV	IKQCTGPL	IKQEDB	IKQEDBLD	IKQEQU	IKQERC	IKQEXLST	IKQFCDB	IKQFMT1	IKQFMT3	IKQFMT4	IKQFNDLB	IKQIOARG	IKQIODRB	IKQIOEQU	IKQIOWKA	IKQIXHDR	IKQJIB	IKQJRNDS	IKQKWTB	IKQLBRC	IKQLPMB	IKQLUB	IKQMDADS	IKQMSGPL
IIPAMT00																																						
IIPBMR00																																						
IIPCLS00		X	X																																		
IIPOPN00		X	X																																		
IIPPRCMR																																					
IIPPRCPR		X	X																																		
IKQAIX																			X																		
IKQBFA				X												X			X						X	X	X										
IKQBFB																			X								X										
IKQBLD																																					
IKQBRP																																					
IKQCAS								X											X										X								
IKQCIR								X																					X								
IKQCIS								X								X			X										X								
IKQCLCAT											X	X	X		X	X																					
IKQCLEAN					X							X										X		X												X	
IKQCLO00										X	X	X				X																		X			X
IKQCLOCL											X	X																									
IKQCLOVY											X	X																									
IKQDCN																																					
IKQDRP																																					
IKQDUMP								X								X											X							X			
IKQDUMPC												X	X	X	X																				X		
IKQEDX							X		X			X	X		X	X	X							X										X	X		
IKQEOV												X				X								X											X		
IKQERH																			X																		
IKQERX																				X																	
IKQGEN	X																			X																	
IKQGNX00																X			X								X		X								
IKQGPT																			X																		
IKQIOA00				X		X										X			X	X	X				X	X	X	X						X			
IKQIOB00				X		X													X		X						X	X									
IKQIXE00								X					X		X				X										X								
IKQIXF00								X								X													X								
IKQIXS00																													X								
IKQJIBSM																X																					
IKQJRN								X											X	X											X						
IKQKRD																X			X																		
IKQLAB					X							X												X									X		X		
IKQLASMD												X																									
IKQLCD																													X								
IKQLCN																			X																		
IKQLCP																X			X								X		X								
IKQMDY																			X																		
IKQNCA00								X								X			X										X								

Figure 6.3 Macro-to-module relationships for all VSAM modules except catalog and DADSM (part 2 of 6)

Module \ Macro	IKQOAL	IKQOPCLR	IKQOPCLW	IKQOPLCT	IKQOPNWA	IKQPLH	IKQPUB	IKQRDF	IKQRLSE	IKQRPHD	IKQRPL	IKQRQM	IKQRSCB	IKQTHB	IKQUSB	IKQUSE	IKQVLST	IKQVOL1	IKQVRGN	IKQRPPL	IKQVSMDP	IKQVSRT	LOAD	MAPCOMR	MAPPIB	MODCB	POINT	POST	PUT	RELEASE	SHOWCB	SYSCOM	USE	WAIT
IIPAMT00																																		
IIPBMR00																																		X
IIPCLS00																														X				
IIPOPN00																						X								X				
IIPPRCMR																																		X
IIPPRCPR																									X	X		X		X				
IKQAIX						X					X																							
IKQBFA						X					X		X	X							X			X				X						X
IKQBFB						X				X	X				X													X						X
IKQBLD						X					X																							
IKQBRP						X				X			X							X		X												
IKQCAS						X		X			X																							
IKQCIR						X					X																							
IKQCIS						X		X			X																	X						X
IKQCLCAT		X	X			X			X		X					X														X			X	
IKQCLEAN																		X																X
IKQCLO00	X	X	X			X			X		X			X		X			X		X	X								X			X	
IKQCLOCL		X	X			X					X				X																			
IKQCLOVY		X	X			X					X				X																			
IKQDCN																																		
IKQDRP																						X												
IKQDUMP			X			X					X			X	X		X							X				X						X
IKQDUMPC											X								X															
IKQEDX		X	X														X																	
IKQEOV		X	X														X																	
IKQERH						X					X										X													
IKQERX						X					X																							
IKQGEN											X											X												
IKQGNX00						X		X			X																							
IKQGPT						X					X																							
IKQIOA00						X					X										X							X						X
IKQIOB00						X					X										X													
IKQIXE00						X		X			X																							
IKQIXF00					X			X																										
IKQIXS00						X					X																							
IKQJIBSM		X																																
IKQJRN						X		X			X																							
IKQKRD						X					X																							
IKQLAB							X																											X
IKQLASMD																																		
IKQLCD						X					X																							
IKQLCN						X		X			X																							
IKQLCP						X		X			X																							
IKQMDY						X					X																			X		X		
IKQNCA00						X		X			X																							

Figure 6.3 Macro-to-module relationships for all VSAM modules except catalog and DADSM (part 3 of 6)

Module \ Macro	ACB	ASYSCOM	CANCEL	CATLG	CCB	CDLOAD	COMRG	DEQB	DTFCN	ENQB	EOJ	EXCP	FREEVIS	GENCB	GET	GETVIS	IGGCAXWA	IGGCCA	IGGMCMWA	IGGMDRWA	IIPAMDTF	IIPDTF	IIPRAT	IKQACB	IKQAIR	IKQAMBL	IKQAMCBS	IKQAMDSB	IKQARDB	IKQAREX	IKQARGH	IKQAUHDR	IKQBHD	IKQBKPHD	IKQBLARD	IKQBSPH	IKQBUFE
IKQNEX				X		X	X						X			X								X		X		X	X					X		
IKQOCMSG				X	X	X						X												X												
IKQOPN				X		X							X			X	X			X				X		X	X	X					X	X		X
IKQOPNAI				X		X																		X		X										
IKQOPNCT						X	X					X	X			X	X			X				X		X		X								
IKQOPNDO				X		X							X											X		X		X					X			X
IKQOPNHC				X		X							X			X	X							X				X								
IKQOPNNC																X								X		X		X					X	X			X
IKQOPNOV			X	X		X							X			X	X							X		X		X	X								X
IKQOPNRD				X		X							X			X								X				X									
IKQOPNRP																								X		X		X							X		
IKQOPNUC																								X		X											
IKQOPNUS				X		X							X			X								X		X											
IKQPBF00																										X		X				X					X
IKQPFO00						X							X			X										X		X	X								X
IKQRBA				X		X							X			X										X		X	X								
IKQRCL00													X			X										X		X	X								X
IKQRQA																								X		X		X	X								
IKQRQB																								X		X		X	X								
IKQRRP						X																				X		X	X								X
IKQRTV																										X		X									
IKQSCN																												X									
IKQSFT																																					
IKQSPM00						X																				X		X	X								
IKQSRG																										X		X									
IKQSRT																										X		X	X								
IKQSRU																										X		X	X								X
IKQSTM													X			X								X							X						
IKQTMSD																								X		X		X									X
IKQTMSF																								X		X		X									
IKQUPD																										X		X									X
IKQUPG													X			X								X	X	X		X	X								X
IKQVDTPE						X	X						X			X																					
IKQVEDA		X			X		X				X	X																									
IKQVFY																								X		X		X	X								
IKQVSM													X			X								X		X		X									
$$BACLOS		X			X		X					X												X		X											
$$BCVSAM						X	X																	X		X											
$$BCVS02													X																									
$$BCVS03							X						X			X																					
$$BCVS04												X																									
$$BOVSAM					X	X	X					X	X			X								X													
$$BOVS01					X		X									X								X													
$$BOVS03							X						X											X		X		X	X								
$$BTCLOS						X	X																	X		X											

Figure 6.3 **Macro-to-module relationships for all VSAM modules except catalog and DADSM (part 4 of 6)**

Module \ Macro	IKQCBMTB	IKQCB1	IKQCB2	IKQCCB	IKQCCBCW	IKQCCW	IKQCGETC	IKQCIW	IKQCLCOR	IKQCLRLS	IKQCLWA	IKQCOMRG	IKQCTGFL	IKQCTGFV	IKQCTGPL	IKQEDB	IKQEDBLD	IKQEQU	IKQERC	IKQEXLST	IKQFCDB	IKQFMT1	IKQFMT3	IKQFMT4	IKQFNDLB	IKQIOARG	IKQIODRB	IKQIOEQU	IKQIOWKA	IKQIXHDR	IKQJIB	IKQJRNDS	IKQKWTB	IKQLBRC	IKQLPMB	IKQLUB	IKQMDADS	IKQMSGPL
IKQNEX							X		X			X	X		X	X	X																		X	X		
IKQOCMSG												X	X		X																							X
IKQOPN				X								X			X	X				X	X													X	X	X		X
IKQOPNAI												X	X		X																			X				
IKQOPNCT				X								X				X						X		X	X									X		X		
IKQOPNDO										X		X	X		X	X																		X		X		
IKQOPNHC												X	X		X																			X				X
IKQOPNNC				X								X									X													X				X
IKQOPNOV												X	X		X	X	X								X									X	X	X		
IKQOPNRD												X	X		X																			X				
IKQOPNRP				X								X																										
IKQOPNUC												X			X																			X				
IKQOPNUS												X	X		X																							
IKQPBF00																X		X																				
IKQPFO00								X								X			X																			
IKQRBA													X		X	X																						
IKQRCL00								X								X			X												X							
IKQRQA																			X											X								
IKQRQB																			X											X								
IKQRRP								X								X			X																			
IKQRTV																			X																			
IKQSCN																																						
IKQSFT																																						
IKQSPM00								X								X			X																			
IKQSRG																			X											X								
IKQSRT																			X											X								
IKQSRU																														X								
IKQSTM																																						
IKQTMSD																				X													X					
IKQTMSF																				X													X					
IKQUPD																			X																			
IKQUPG																			X																X			
IKQVDTPE												X																					X					
IKQVEDA																																						
IKQVFY																														X								
IKQVSM				X															X																			
$$BACLOS																																						
$$BCVSAM												X																										
$$BCVS02																																						
$$BCVS03																																						
$$BCVS04																																						
$$BOVSAM																																						
$$BOVS01																																						
$$BOVS03												X				X									X						X							
$$BTCLOS												X																										

Figure 6.3 **Macro-to-module relationships for all VSAM modules except catalog and DADSM (part 5 of 6)**

Module	IKQOAL	IKQOPCLR	IKQOPCLW	IKQOPLCT	IKQOPNWA	IKQPLH	IKQPUB	IKQRDF	IKQRLSE	IKQRPHD	IKQRPL	IKQRQM	IKQRSCB	IKQTHB	IKQUSB	IKQUSE	IKQVLST	IKQVOL1	IKQVRGN	IKQRPPL	IKQVSMDP	IKQVSRT	LOAD	MAPCOMR	MAPPIB	MODCB	POINT	POST	PUT	RELEASE	SHOWCB	SYSCOM	USE	WAIT
IKQNEX		X	X														X																	
IKQOCMSG		X	X																															
IKQOPN	X	X	X	X	X	X			X					X	X	X	X		X	X	X									X			X	
IKQOPNAI		X	X	X	X																													
IKQOPNCT		X					X									X		X															X	X
IKQOPNDO		X	X	X	X	X			X						X	X					X	X							X			X	
IKQOPNHC		X	X	X	X				X							X													X			X	
IKQOPNNC		X	X		X	X					X			X	X																		
IKQOPNOV		X	X	X	X											X																		
IKQOPNRD		X	X	X	X																													
IKQOPNRP		X	X		X	X								X	X							X					X						X
IKQOPNUC		X	X	X	X																													
IKQOPNUS		X	X	X	X										X																			
IKQPBF00						X																												
IKQPFO00						X		X			X																							
IKQRBA		X																																
IKQRCL00						X					X																							
IKQRQA						X					X	X			X																			
IKQRQB						X					X	X			X																			
IKQRRP						X		X			X																							
IKQRTV						X					X																							
IKQSCN						X					X																							
IKQSFT						X																												
IKQSPM00						X					X																							
IKQSRG						X					X																X							X
IKQSRT						X					X																							
IKQSRU						X		X			X																							
IKQSTM																											X							X
IKQTMSD										X	X																							
IKQTMSF										X	X																							
IKQUPD						X					X																							
IKQUPG						X		X	X		X				X	X														X			X	
IKQVDTPE																																		
IKQVEDA																								X								X		X
IKQVFY						X					X																							
IKQVSM						X					X																X							X
$$BACLOS	X					X																		X						X		X		X
$$BCVSAM			X																															
$$BCVS02			X																					X	X									
$$BCVS03																																		
$$BCVS04																																		X
$$BOVSAM			X																					X							X		X
$$BOVS01			X																					X									X
$$BOVS03		X																																
$$BTCLOS			X																															

Figure 6.3 Macro-to-module relationships for all VSAM modules except catalog and DADSM (part 6 of 6)

Catalog Communication Area Register Save Area

A catalog communication area (CCA) is built for every call to catalog management. The CCA contains a register save area (CCAREGS) that allows the PSR (programming systems representative) to follow the flow of control from one catalog management procedure to another, through each procedure called to process the request.

The contents of the first three words in the CCA field named CCAREGS are as follows:

- the first word contains the contents of register 13, which is a pointer to the user's save area, and
- the second and third words are unused.

If a catalog management procedure is entered from another catalog management procedure, the fourth and subsequent words are used as three-word catalog save areas defined as follows:

- the first word contains the contents of register 12, which is the calling procedure's base address,
- the second word contains the contents of register 13, which is a pointer to the previous 12-byte entry in the register save area (CCAREGS), and
- the third word contains the contents of register 14, which is the return address (in the calling procedure).

Immediately after registers 12, 13, and 14 are saved (at register 13 + 12), register 12 is updated to contain the *called* procedure's base address. Register 13's value is increased by 12, so that it points to the latest entry in CCAREGS. While a catalog management procedure is processing, register 11 contains a pointer to the beginning of the CCA.

Note that backward movement is not recorded in the trace table. For example, if procedure B returns to procedure A, the return is not shown in the register save area.

Error Code-to-Module Relationship (Catalog Management)

The error codes issued by VSAM catalog modules are set in the CCACD1 field of the CCA and transferred to register 15 on exit from catalog managment. The reason codes are set in the CCAREASN field of the CCA. They are passed back to the user via AMS or Record Management.

Figure 6.4 contains the definitions of the error codes. To each error code belong one or several reason codes. Along with the reason code the module which detected the error is also shown. It is identified by the last two letters of its name.

The PL/S names are those names to which the catalog management error codes are equated. To find the assembler instructions which set the error code, first reference the PL/S statement, then relate the PL/S statement to the appropriate assembler language statement.

PL/S Name for Code	Error Code Dec	Error Code Hex	Reason Codes	Module Name	Description
RCCAT	4	4	2	AC AD AE AH	Error when opening a catalog.
			4	AE CS	Error when closing a catalog.
			8	AH	Internal error – an ACB was supplied to catalog management but its ID was not X'A0'. Can also be caused by a problem program overlaying core it does not own.
			10	AH	The user catalog entry in the master catalog cannot be found. The user has either made a spelling error, has exported the user catalog, or has more than one master catalog and has executed IPL using the wrong one.
RCENT	8	8	2	AF	During catalog DELETE, the cluster record for the catalog cannot be found at its normal location (3rd self-describing record).
			4	AG	The catalog I/O routine read a record from the catalog and found it to be a free record. Since this never occurs during normal processing, an internal error is returned.
			6	CG,AN,BG, BN	A request to read a record resulted in a no record found error from VSAM.
			8	AL,AT,BD, CA,CG	A request to place a record by key into a catalog resulted in a duplicate key error from VSAM.
			12	CB	An internal error has occurred indicating that a record thought to be on the buffer chain is not present.
RCNOTCYL	16	10	0	AT	The AMS cylinder parameter was used but the extents found on the corresponding DLBL and EXTENT statement do not start or end on a cylinder boundary. This error can only occur during DEFINE CATALOG, DEFINE SPACE or DEFINE UNIQUE.

Figure 6.4 Catalog Management error code-to-module relationship (part 1 of 9)

PL/S Name for Code	Error Code Dec	Hex	Reason Codes	Module Name	Description
RCINSP	20	14	–	CG	Insufficient direct access space available to extend the catalog or CRA. (Reason codes are from Record Management – see Figure 6.6)
					To correct this situation a DEFINE SPACE is needed to provide more space for suballocation.
RCIOL	24	18	2	AG,AZ,CG	I/O error during a LOCATE.
			4	C9	I/O error during a CATALOG VERIFY operation.
RCIONL	28	1C	2	AG	I/O error but not during LOCATE processing.
			4	AG,CG	I/O error during EXCP during catalog open (non-build case).
RCINCPL	32	20	2	BC	Internal error indicating that catalog management was unable to return the requested data in the catalog parameter list (CPL) for update extend.
RCDSNF	36	24	2	BD	Incorrect record type read.
			4	BN	Data set not found in the VTOC.
RCVLSZ	40	28	0	AL,BG,AZ CX	Internal workarea provided was too small.
RCVLSM	44	2C	2	AF	It has been detected during DELETE CATALOG that the CTGWKA work area is too small. The user (AMS) has to provide a larger area.
RCINFUNC	48	30	2	AB	An invalid CPL has been passed to the catalog management driver.
			4	AT	During DEFINE, an incorrect master catalog ACB was found.
			6	BD	Alter of non-VSAM data sets is not allowed.
			8	BD	Alter of catalog name is not allowed.
			20	CL	Forced delete space is not allowed on catalog volume.
RCIOU	52	34	–	A7,BD,BN A6,AF	I/O error during VTOC processing (Reason codes from DADSM – see Figure 6.5)
RCSEC	56	38	2	BM	All attempts to provide a password via the system operator are used without a successful compare.
			6	BM	No prompt allowed and password not provided via the catalog parameter list.
			8	BM	USVR requested stop.
			12	A7	Security violation from DADSM scratch.

Figure 6.4 **Catalog Management error code-to-module relationship (part 2 of 9)**

PL/S Name for Code	Error Code Dec	Hex	Reason Codes	Module Name	Description
RCINENT	60	36	2	BC	Invalid entry type for EXTRACT.
			4	BG,BE	Invalid entry type 'C' or non-VSAM.
			6	BD	Invalid entry type 'C' for alter of attributes.
			8	BD	Invalid 'C' or 'I' entry type for alter buffer size.
			10	BD	'C' record invalid for alter of AMDSB.
			12	BE	No alter volume allowed on 'C'-type records.
			14	BE	Test CPL error during ADD volume.
			16	CA,CD	AIX G record association is not 'D'|'I'|'C' (AIX is not a KSDS).
			18	CA,CD	Upgrade set Y record association not D|I.
			20	CA,CD	'Y' association in base cluster data record does not point to a 'Y' record.
			22	CA,CD	'D' association in 'C' record does not point to a 'D' record.
			24	BD	Upgrade or update for a record which is not a 'G' or 'R'.
			26	BD	Alter of exception exit but the record is not 'D' or 'I'.
			28	BD	Alter of average record size but the record is not 'D' or 'I'.
			30	BD	Alter of expire date but the record is not 'C' or 'G' or 'R'.
			32	AH	DEFINE, DELETE, ALTER of non-VSAM entry prohibited in recoverable catalog.
RCNAME	64	40	2	BL	Test field name not present in group space header.
			4	BE	Association names do not exist.
RCNOSP	68	44	2	BB	Cannot extend a unique file.
			4	BB	No secondary extent value specified.
			6	BB	No space for suballocation.
			12	BB	More than 16 extents/volume for reusable file.

Figure 6.4 **Catalog Management error code-to-module relationship (part 3 of 9)**

PL/S Name for Code	Error Code Dec	Hex	Reason Codes	Module Name	Description
RCNMNTD	72	48	–	A7	Volume not mounted.
				A6 AQ CG, BL BN,	The reason codes come from $$BOVS01 and are: ● Register 15 contains X'08': invalid or unassigned symbolic unit ● Register 0 contains X'04': catalog time stamp greater than time stamp in the volume's format 4 DSCB.
RCRELOP	80	50	0	CA	Related object is reusable.
			2	CA	Related object is a RRDS.
			4	CA	Related object does not exist.
			6	CA,CP	AIX/PATH not allowed to be built over a catalog.
			8	CA	Name of AIX/PATH and related object are identical.
			10	CA,CP	No pointer to a related object of an AIX/PATH.
			12	CA,CP	AIX not being built over a base cluster, or related path object not a cluster or an AIX.
RCDATE	84	54	0	AF,BG	Expiration date not reached.
RCCRAOP	88	58	0	CO	Open of CRA failed.
			2	–	I/O error in CRA.
			4	CO	Internal call to CO for CRA open has conflicting parameters.
			6	A6	Space of less than one cylinder for CRA is not allowed.
RCDSEXT	92	5C	0	BB	Attempt was made to exceed the maximum number of extents.
RCSPANCK	96	60	0	BX,CA	Prime key as specified for spanned record is not completely contained in the first control interval of the record
			4	CY	Maximum logical record size for spanned records exceeds CA size.
			6	CA	AIX key for spanned records is not completely contained within the first control interval of the base cluster record.
			8	CA	One of the following: The AIX key is not completely contained within the base cluster record, or the maximum record size for defining an alternate index is too small.
RCRECVOL	100	64	0	A6	Attempt was made to define a unique file on a volume owned by a recoverable catalog. You must define space before any unique files can be defined.

Figure 6.4 **Catalog Management error code-to-module relationship (part 4 of 9)**

PL/S Name for Code	Error Code Dec	Hex	Reason Codes	Module Name	Description
RCCATEX	104	68	0	AL	The master catalog is already open but a request has been made for master catalog DEFINE.
RCINFNAM	108	6C	0	AY	Invalid field name.
RCINFPL	112	70	2	AY	Invalid group code in FPL.
			4	AY	Invalid update request.
RCCATBAL	116	74	–	–	Catalog out of balance (not used by DOS/VS).
RCSYSFLD	120	78	0	AX	Non-existent field being modified.
RCINCI	124	7C	2	AG,CG	Invalid RBA return code from record management.
			4	AG	Catalog build open processing but specified CI was greater than 9.
			6	AG	CCR record ('L') read by error.
RCBLKVCK	128	80	–	–	Validity check on user block (not used by DOS/VS).
RCINPTR	132	84	2	AL,BH	No pointer to VOLSER list.
			4	AL	No FPL to AMDSB of data.
			6	AL	No FVT from cluster level.
			8	AL	No pointer in FPL to data set attribute.
			10	AL	No FPL for volume space parameters.
			12	AL	No pointer to expiration date value.
			14	AL	No pointer to creation date in FPL.
			16	BH	No pointer to device type in FPL.
			18	BH	No FPL in FVT.
			20	AL	No pointer to the workarea.
			22	AL	No pointer to password data of related object.
			24	AL	No pointer to owner ID in FPL.
			26	BX	No pointer to cluster space parameter in FPL.
			28	BX	No pointer to data space parameter in FPL.
			30	BX	No pointer to index space parameter in FPL.
			32	AN	No buffersize FPL in data FVT.
			34	AN	No buffersize FPL in cluster FVT.
			36	AN	No buffersize FPL in index FVT.
			38	BX	No logical record size FPL in cluster or data FVT.
			40	BH	No pointer to data set file sequence number in volume list FPL.
RCMISPAR	136	88	2	AL	No length for volume serial list area.
			4	AL	Missing DNAME parameter with DEFINE UNIQUE FILE.
			6	AL	Missing cluster entry name.
			8	AL	Missing space parameter in space FVT.

Figure 6.4 **Catalog Management error code-to-module relationship (part 5 of 9)**

PL/S Name for Code	Error Code Dec	Hex	Reason Codes	Module Name	Description
RCMISPAR	136	88	10	AL	Missing VOLSER list pointer in space FVT.
			12	AL	Missing DNAME pointer in space FVT.
			14	AL	No length in volume list from cluster FVT.
			16	BX	No space parameter on 'C' or 'D' FVT.
			18	BX	Average logical record size missing.
			20	AN	No key specified.
			22	BD	Unique file needs DLBL statement for rename; or
				BL	DLBL-statement missing.
			24	BH	No entries in volume list.
			26	BH	No entries in device type list.
			28	CA	AIX name missing.
			30	CP	Path entry name missing.
RCINCNPM	140	8C	2	AL	Index FVT found for RRDS or ESDS.
			4	AL	Keyrange is invalid.
			6	AL	Keyranges found on both data and cluster FVT.
			8	AL	Workarea too small.
			10	BX	Space parameters found on both cluster and data FVT.
			12	AN	Buffersize specified more than once.
			14	BX	Average logical record size specified on index FVT.
			16	BX	Average logical record not valid for CATALOG DEFINE.
			18	BX	Average logical record size specified on cluster and data FVT.
			20	AN	Inconsistent keylength specified in 'D' and 'I' FVT.
			22	AP	Inconsistent VOLSER lists with different name in each list; or
				AQ	DLBL and volume list do not match.
			24	AP,AT	Primary allocation for data space less than required for the catalog.
			26	AL	Invalid space request type for catalog DEFINE.
			28	BH	Unequal number of VOLSER and file sequence numbers in list entries.
			30	BH	More device type entries than VOLSER's.
			32	AN	Invalid keyposition specified on 'D' or 'I'.
			34	BX	Invalid space request type on DEFINE.
			36	AT	The number of keyranges and the number of VOLSER's are not equal.

Figure 6.4 **Catalog Management error code-to-module relationship (part 6 of 9)**

PL/S Name for Code	Error Code Dec	Hex	Reason Codes	Module Name	Description
RCINCNPM	140	8C	38	AL	Unique attribute is not allowed for catalog DEFINES.
			40	AL,CP,BH BG,BD,CA	FILE was not specified for a recoverable catalog.
			42	AL	RRDS has spanned attribute.
			44	AL	RRDS has a maximum record length which is not equal to the average record length.
			46	BD	An attempt was made to specify an exception exit for a data set which was originally defined with a DOS/VS Release earlier than Release 31.
RCINENTN	144	90	2	AL,CA,CP AL	AIX/PATH name is invalid, first character must be alphabetic.
			4	AL	Unique name is invalid because it uses Z999999, which is restricted.
			6	AL	Data and index names not allowed for a catalog.
RCVOLOWN	148	94	0	A6	Volume already owned by another catalog.
RCDNECAT	152	98	0	AF	Non-empty catalog cannot be deleted.
RCNOSPSA	156	9C	0	AU	No space available within the catalog for suballocation.
RCVNDSPD	160	A0	0	BL,BN	Volume record not deleted from the catalog because some file still owns space on the volume. All empty data spaces on this volume have been deleted.
RCINSSWA	164	A4	2	AQ,AT,AC AD,AH,A7 AS,BB,BD BE,BM,B8 C9,BG,BN CB,CD,CO CY	Catalog management GETVIS error: A continuous area of virtual storage of the requested size is not available.
			6	AQ,A6,BD BU	DADSM GETVIS error: As above, but the call came from DADSM.
			8	CG	Record management GETVIS error: As above, but the call came from record management.
RCINVDTY	168	A8	2	BX,AP	Device type not supported.
			4	BH	Invalid device name.
RCDUPNVL	172	AC	4	BD,BN,AQ	Duplicate name in VTOC.
RCNSPVTC	176	B0	0	AQ	No space available in the VTOC.
RCDSO	184	B8	2	CX	The catalog is in use by some partition. Until the number of users is zero, the catalog cannot be deleted.
			4	CX	The data or index component (or both) is (are) in use and the file cannot be deleted.

Figure 6.4 **Catalog Management error code-to-module relationship (part 7 of 9)**

PL/S Name for Code	Error Code Dec	Hex	Reason Codes	Module Name	Description
RCCATUNA	188	BC	2	AB,CG	No RPL available for processing, request ignored.
			4	AJ	Suballocate failure.
RCMLRSZ	192	C0	0	BX	Maximum logical record size exceeds allowable maximum.
RCMCISZD	196	C4	0	BX	CI size too large for allowable maximum for data component.
RCMCISZI	200	C8	0	BX	CI size too large for index device type.
RCKEYINC	204	CC	0	BX	Key extends beyond the end of the maximum logical record.
RCBUFSIZ	208	D0	0	BX	Bufferspace too small for minimum number of CIs.
RCSIZCAL	212	D4	2	BX	Not enough parameters specified in DEFINE.
			4	CY	Specified and default values result in only one CI per CA for a KSDS or zero CI per CA for an ESDS.
			6	CY	For a non-unique KSDS, the index CI size is too small, and an attempt to reduce the number of data CIs failed.
			8	CY	For a unique KSDS, the index CI size is too small. The number of data CIs cannot be reduced as their size is fixed at one cylinder.
			10	CY	Buffer space too small for a non-unique data set.
			12	CY	Buffer space too small for a unique data set.
RCEXTOVL	216	D8	2	AQ	Overlap on: a. VTOC b. Expired secure file c. Expired file d. Unexpired secure file e. Unexpired file f. New extents
RCINEXT	220	DC	–	–	Invalid EXTENT card.
RCMXGRP	224	E0	0	AW	Maximum number of group occurrence pointers have been processed.
			2	AW	More than 125 AIXs in the upgrade set.
RCLOCKER	228	E4	0	A7,AL,BL	Time of day clock hardware error.
RCRCDLDF	232	E8	–	AQ,BM,BH CX,CG	CDLOAD error, reason is: a. from CDLOAD b. 0 if no reason is available
RCINDER	240	F0	4	AQ,AF,AH AP,AE,A7 BX,BG,B8 BL,BN CO CS	IKQVLAB error (see Figure 6.9).
			6	AQ,A7 BL,B8,AT	No extents found.

Figure 6.4 **Catalog Management error code-to-module relationship (part 8 of 9)**

PL/S Name for Code	Error Code Dec	Hex	Reason Codes	Module Name	Description
RCINDER	240 (Cont.)	F0	8	AQ,AT,A8 BL	Invalid device type, or IKQVLAB error during device type processing.
			10	AT,AQ	Too many extents, or Duplicate volume serial numbers.
			12	AT	Number of tracks required by the Access Method Services command exceeds DLBL total per volume.
			14	BG	No DLBL found for unique file.
			16	BG	Unable to scratch because DLBL is missing.
			18	BG	Unable to open for ERASE because DLBL is missing.
			20	BN	System LUB not found.
			22	AH	Catalog name in DLBL and CPL do not match.
			24	AH	Master catalog DLBL not found.
			26	AH	User or job catalog DLBL not found, or job catalog required but not found.
			28	AH	DLBL catalog name is missing.
			30	AH	Volume serial number in extent statement for job catalog does not match volume serial in the entry for this job catalog in master catalog.
			32	CO	DLBL missing so volume serial number cannot be found and consequently CRA cannot be opened.
RCEFRMPH	242	F2	–	BG	Physical I/O error occurred while trying to erase the data set being deleted. (Reason codes are from VSAM record management – see Figure 6.6.)
RCEF	244	F4	–	BG	This ACB could not be opened (OPEN failed while trying to erase). The address of the catalog ACB or the CI number may be wrong. (Reason codes from OPEN – see Figure 6.9.)
RCENQ	246	F6	16	AB	Catalog management was unable to gain exclusive control of a resource.
RCVOLENT	248	F8	0	AR,AT,BX BN	Volume entry does not exist in this catalog.
RCEFRM	250	FA	–	BG	VSAM record management has found an error during ERASE. (Reason codes from VSAM record management – see Figure 6.6.)
RCEE	252	FC	0	AH	Early exit. (Internal indicator; if found in CCACD1, it does *not* indicate an error, but that the last O/C/EOV request for catalog open has completed.)

Figure 6.4 **Catalog Management error code-to-module relationship (part 9 of 9)**

Module	Routine	Return Code (Reg.15)	Dec.	Meaning
IKQWDS00*	–	X'00'	00	Write completed successfully
		X'0C'	12	VTOC I/O error
		X'10'	16	Duplicate name on volume
		X'14'	20	VTOC full
		X'40'	64	GETVIS failed
IKQRDS00	–	X'00'	00	Successful read
		X'04'	04	I/O error reading VOL1 label
		X'08'	08	Volume not mounted
		X'0C'	12	VTOC I/O error
		X'2C'	44	DSCB not found
		X'30'	48	Invalid request
		X'34'	52	DADRADDR not on but DADBYPS is
		X'40'	64	GETVIS failed
IKQCOV00*	–	X'00'	00	No overlap found
		X'18'	24	Overlap on expired file
		X'1C'	28	Overlap on unexpired file
		X'20'	32	Overlap on protected unexpired file
		X'24'	36	Overlap on VTOC
		X'38'	56	Overlap on expired protected file
		X'40'	64	GETVIS failed
IKQREN00*	–	X'00'	00	Rename successful
		X'08'	08	Volume not mounted
		X'40'	64	GETVIS failed
		X'44'	68	Security violation
IKQSCR00*	–	X'00'	00	Scratch successful
		X'08'	08	Volume not mounted
		X'40'	64	GETVIS failed
		X'44'	68	Security violation
IKQALL00*	–	X'00'	00	Successful allocate
		X'08'	08	Volume not mounted
		X'28'	40	No extents given in EXTENT card
		X'40'	64	GETVIS failed
		X'48'	72	CDLOAD failed
IKQVTC00	–	X'00'	00	Return codes have no meaning; 00 is always returned
IKQPOP00*	–	X'00'	00	Successful DSCB build
		X'08'	08	Volume not mounted
		X'40'	64	GETVIS failed
*Note: These modules also return codes issued by other internally called modules.				

Figure 6.5 DADSM error code-to-module relationships

Error Code-to-Module Relationship (Record Management)

There are internal and external error codes. Internal error codes are set in register 15 by record management modules and passed to IKQERH for handling. Three classes of internal error codes exist:

- Specification errors
  (with a value from X'01' to X'1F')
- Processing errors
  (with a value from X'20' to X'3F')
- I/O errors
  (with a value from X'40' to X'5F')

External error codes are set in the RPL (see section *Data Areas*) and register 15 by IKQERH, according to the internal error codes, and passed back to the user. Figure 6.6 shows the Record Management internal - external error code relationship.

Internal error codes (IKQERC macro)		**External error codes (IKQRPL macro)**			**Meaning**
Symbolic			**Symbolic**		
code	**R15**	**R15**	**name**	**RPL**	
E01	X'01'	X'04'	–	–	RPL held by another request
E02	X'02'	X'02'	–	–	Reserved
E03	X'03'	X'08'	RPLNOPLH	X'40'	No PLH available
E04	X'04'	X'08'	RPLNOPEN	X'44'	CNV processing not requested
E05	X'05'	X'08'	RPLNOPEN	X'44'	Keyed processing not requested
E06	X'06'	X'08'	RPLNOPEN	X'44'	Output not requested
E07	X'07'	X'08'	RPLRRADR	X'C4'	Invalid address requested
E08	X'08'	X'08'	RPLKEYES	X'48'	Keyed access requested for ESDS
E09	X'09'	X'08'	RPLADRKS	X'4C'	ADR or CNV insert for KSDS
E10	X'0A'	X'08'	RPLINERS	X'50'	Illegal ERASE request
E11	X'0B'	X'08'	RPLINLOC	X'54'	Illegal Locate mode specification
E12	X'0C'	X'08'	RPLINLD	X'74'	Illegal request during data set load
E13	X'0D'	X'08'	RPLNOPOS	X'58'	No keyed positioning done
E14	X'0E'	X'08'	RPLNOPOS	X'58'	No sequential positioning done
E15	X'0F'	X'08'	RPLNGUPD	X'5C'	No valid GET UPD issued
E16	X'10'	X'08'	RPLUPDKC	X'60'	Key change during update
E17	X'11'	X'08'	RPLLENCN	X'64'	Length change for addressed update
E18	X'12'				Reserved
E19	X'13'	X'08'	RPLCONOP	X'68'	Improper RPL-option (BWD)
E20	X'14'	X'08'	RPLCONOP	X'68'	Improper or conflicting RPL options, invalid transaction ID or LRU percentage value, or WRTBFR without LSR/DFR
E21	X'15'	X'08'	RPLIMGKL	X'70'	Improper generic key length
E22	X'16'	X'08'	RPLIMRCL	X'6C'	Improper RECLEN
E23	X'17'	X'08'	RPLINERS	X'50'	Invalid ERASE request (AIX)

Figure 6.6 **Record Management internal/external error code relationship (part 1 of 2)**

Internal error codes (IKQERC macro)		External error codes (IKQRPL macro)			Meaning
Symbolic			Symbolic		
code	R15	R15	name	RPL	
E24	X'18'				Reserved
E25	X'19'	X'08'	RPLNOPOS	X'58'	Invalid switching FWD-BWD
E26	X'1A'				Reserved
E27	X'1B'	X'08'	RPLINVRR	X'C0'	Invalid RR number (RRDS)
E28	X'1C'	X'08'	RPLIPATH	X'C8'	Invalid path access (AIX)
E29	X'1D'	X'08'	RPLINBWD	X'CC'	Illegal PUT in BWD-mode
E30	X'1E'				Reserved
E32	X'20'	X'08'	RPLINRBA	X'20'	Invalid RBA
E33	X'21'	X'08'	RPLNRFND RPLNOREC	X'10'	No record found
E34	X'22'	X'08'	RPLEOFDS RPLEODER	X'04'	End of data set encountered
E35	X'23'	X'08'	RPLWRKAS	X'2C'	User's work area not large enough
E36	X'24'	X'08'	RPLSEQCK	X'0C'	Sequence error
E37	X'25'	X'08'	RPLDUPRC RPLDUP	X'08'	Duplicate record
E38	X'26'	X'08'	RPLNKEYR	X'24'	No key range for new record
E39	X'27'	X'08'	RPLNOVIR	X'28'	Insufficient virtual storage
E40	X'28'	X'08'	RPLNRSPA RPLNOEXT RPLSPACE	X'1C'	No DASD space available
E41	X'29'	X'08'	RPLNVOLM	X'18'	Volume or extent unavailable
E42	X'2A'	X'08'	RPLCDLOD	X'30'	CDLOAD failure
E43	X'2B'	X'08'	RPLVLERR	X'34'	No BCB available or invalid attempt to insert RRDS after preformat
E44	X'2C'	X'08'	RPLEXCTL	X'14'	Exclusive control failure
E45	X'2D'	X'08'	RPLCATLG	X'80'	Internal catalog call failure
E46	X'2E'	X'08'	RPLSRLOC	X'84'	Illegal GET in LOC-mode
E47	X'2F'	X'08'	RPLINCSR	X'8C'	Inconsistent spanned record
E48	X'30'	X'08'	RPLSRADR	X'88'	Illegal addr. retrieval for spanned records KSDS
E49	X'31'	X'08'	RPLNOBAS	X'90'	No base record for associated AIX pointer
E50	X'32'				Reserved
E51	X'33'	X'08'	RPLMAXPT	X'94'	Max. no of AIX pointers exceeded
E52	X'34'	X'08'	RPLNOBUF	X'98'	No buffers available (LSR)
E64	X'40'	X'0C'	RPLRDERD	X'04'	Data read error
E65	X'41'	X'0C'	RPLRDERI	X'08'	Index read error
E66	X'42'	X'0C'	RPLRDERS	X'0C'	Sequence set read error
E72	X'48'	X'0C'	RPLWTERD	X'10'	Data write error
E73	X'49'	X'0C'	RPLWTERI	X'14'	Index write error
E74	X'4A'	X'0C'	RPLWTERS	X'18'	Sequence set write error

Figure 6.6 Record Management internal/external error code relationship (part 2 of 2)

Module	Internal error code(s)
IKQAIX	E49
IKQBFA00	E43, E44
IKQCAS00	E42, E39
IKQCIR	E42
IKQCIS00	E39, E42, -1
IKQEDX	E39, E41, E42, E45
IKQEOV	E39
IKQERH	E01, E33, E34
IKQGNX00	E64
IKQGPT	E32, E33, E34, E36, E48
IKQIOA00	E32, E41, E42, E65, E66
IKQIOB00	E64, E65, E66, E72, E73, E74
IKQIXE00	E39, E44, E45, -5
IKQIXF00	-5
IKQJRN	E39, E42
IKQKRD	E38
IKQMDY	E43
IKQNCA00	E39, E42, E45
IKQNEX	E39, E40, E42, E45
IKQPFO00	E39, E42, E45
IKQRBA	E39, E45
IKQRCL00	E39
IKQRQB	E03, E04, E05, E06, E07, E08, E09, E10, E11, E12, E13, E14, 15, E16, E17, E19, E20, E21, E22, E23, E25, E27, E28, E29, E32, E35, E39, E42, E52
IKQRRP	E42, E45
IKQRTV	E35
IKQSPM00	E42
IKQSRG	E35, E44, E46, E47
IKQSRT	E36, E37
IKQUPD	E34
IKQUPG	E22, E39, E44, E51
IKQVSM	E01, E03, E39
Note: A value of minus is no error code but an internal VSAM interface return code.	

Figure 6.7 **Record Management internal error code-to-module relationship**

Module	Return Code (Reg. 15)	Meaning	Error Code (Reg. 0)	Meaning
IKQGEN	X'00'	Successful completion	-	-
	X'04'	An error has been detected	X'01'	Invalid function type code
			X'02'	Invalid control block type-code
			X'03'	Invalid keyword type-code
			X'08'	Not enough virtual storage available
			X'09'	User area too small
			X'0A'	Exit address is not specified in the input
			X'0E'	Inconsistent parameters
			X'0F'	The user area is not on a fullword boundary
	X'08'	Invalid use of the execute form of this macro instruction.	-	Since the return code is set by the macro expansion and not by IKQGEN, R0 contents do not indicate an error code.
	X'0C'	CDLOAD failure	-	The return code is set by the macro expansion, not by IKQGEN, and R0 contains the return code from CDLOAD.
IKQTMSD or IKQTMSF	X'00'	Successful completion	-	-
	X'04'	An error has been detected	X'01'	Invalid function type-code.
			X'02'	Invalid control block type code.
			X'03'	Invalid keyword type-code
			X'04'	Control block not of type specified
			X'05'	The ACB is closed; it must be open
			X'06'	The cluster is not key-sequenced (does not include an index)
			X'07'	The EXLST entry is not present
			X'09'	User area is too small (SHOWCB)
			X'0A'	Exit address is not specified in the input
			X'0B'	The RPL is active
			X'0C'	The ACB is open; it must be closed
			X'0D'	The exit address is not in the control block
			X'0E'	Inconsistent parameters
			X'0F'	The user area is not on a fullword boundary
			X'20'	The parameter 'STRMAX' was specified, but LSR is not active.
	X'08'	Same as for IKQGEN	-	Same as for IKQGEN
	X'0C'	Same as for IKQGEN	-	Same as for IKQGEN

Figure 6.8 **Control block manipulation error code-to-module relationships (Record Management)**

Error Code (ACBERFLG)	Return Code[1] (Reg.15)	Issuing Module	Meaning
X'04'	X'08'	IKQOPN	This ACB is already open
X'0E'	X'08'	IKQOPN IKQOPNCT	The symbolic unit in the DLBL statement is invalid
X'0F'	X'08'	$$BOVS03	No job information blocks (JIBs) are available from the label information cylinder
X'11'	X'08'	$$BOVS03 IKQLAB IKQOPN IKQOPNOV	The address in the ASSIGN statement for the logical unit was IGN (assignment ignored)
X'12'[2]	X'08'	$$BOVS03 IKQOPN IKQOPNOV	The address in the ASSIGN statement for the logical unit was UA (logical unit unassigned)
X'22'[2]	X'08'	IKQLAB IKQOPNOV	The volume serial numbers in the EXTENT statement do not match those in the catalog entry
X'32'	X'08'	$$BOVSAM IKQOPNHC IKQOPNOV IKQOPNRD	One or more VSAM processing modules cannot be loaded because VSAM=YES was not specified in the FOPT macro at system generation or the user's partition is too small
X'50'	X'08'	IKQOPNOV	An attempt was made to assign more than one volume of a multivolume data set to one unit when direct or keyed processing was specified in the ACB
X'5C'	X'04'	IKQOPN	LSR was specified in the ACB macro, but no message area was specified
X'60'	X'04'	IKQOPNHC	A data set which is unusable (due to a failure in recovery) was opened for input
X'64'	X'04'	IKQOPN	Open encountered an empty alternate index in the upgrade set
X'68'	X'04'	IKQOPNOV	The time stamp of the data set's volume does not match the system time stamp in the volume catalog entry
X'6C'	X'04'	IKQOPN	The system time stamps in the catalog entries for the data and index components of a data set do not match. This indicates that the data has been updated separately. **Note:** No warning is given if the index time stamp is greater than the data time stamp.
X'6E'	X'08'	IKQOPN	An attempt was made to open an empty data set for input only; or to open a data set which was not closed properly after initial loading.
X'74'	X'04'	IKQOPNHC	The data set was not closed the last time it was processed.

Figure 6.9 OPEN error code-to-module relationships (part 1 of 3)

ERROR Code (ACBERFLG)	Return Code[1] (Reg.15)	Issuing Module	Meaning
X'75'	X'08'	IKQLAB IKQOPNOV IKQOPNRD	The symbolic unit specified in the EXTENT statement is not a valid device type
X'80'	X'08'	IKQLAB	The DLBL statement is missing or the filename in the DLBL does not match the ACB
X'84'	X'08'	IKQLAB	A permanent I/O error occurred while VSAM was reading the label information cylinder
X'88'	X'08'	$$BOVSAM IKQOPN IKQOPNAI IKQOPNHC IKQOPNNC IKQOPNOV IKQOPNRD IKQOPNRP IKQOPNUS IKQLAB IKQSTM	VSAM could not obtain a continuous area of virtual storage of the size required for work areas, control blocks, or buffers
X'90'	X'08'	IKQOPNHC IKQOPNAI IKQOPNUS	A permanent I/O error occurred while VSAM was reading or writing a catalog entry
X'94'	X'08'	IKQOPN IKQOPNHC IKQOPNOV IKQOPNAI	No valid entry was found in the catalog for this ACB or for the alternate index structure related to this ACB
X'98'	X'08'	IKQOPNHC	Security verification failed: the password specified in the ACB, or supplied by the operator, for a specific level of access does not match the corresponding password in the catalog entry
X'A0'	X'08'	IKQOPNAI IKQOPN	The operands specified in the ACB are inconsistent with each other or with information in the catalog entry, such as keyed processing specified for an ESDS or DFR specified when LSR is inactive.
X'A1'	X'08'	IKQOPN	User buffers were specified with keyed access (they may be specified only with CNV access) or with LSR
X'A4'	X'08'	IKQOPNRD	A permanent I/O error occurred while VSAM was reading the volume label of the volume containing the data set
X'A8'	X'08'	IKQOPN IKQOPNHC IKQOPNRD	The data set is not available because it is being updated, loaded, or reset by (and under exclusive control of) another ACB or because it has been flagged *read only* by Access Method Services
X'B4'	X'08'	IKQOPN IKQOPNCT IKQOPNHC IKQOPNAI	An error occurred while a catalog was being opened

Figure 6.9 OPEN error code-to-module relationships (part 2 of 3)

ERROR Code (ACBERFLG)	Return Code[1] (Reg.15)	Issuing Module	Meaning
X'C0'	X'08'	IKQOPNHC	A data set which is unusable (because of a failure in recovery) was opened for output
X'C4'	X'08'	IKQOPN	Access to data was requested via an empty alternate index
X'D4'	X'08'	IKQOPN	LSR is specified, but the data set being opened is empty (which implies that it is to be loaded).
X'D8'	X'08'	IKQOPNRP	LSR is specified, but the keylength of the data set being opened is greater than the maximum keylength specified for the resource pool.
X'DC'	X'08'	IKQOPNRP	LSR is specified, but the CI size of the data set being opened is greater than the largest buffer size specified for the resource pool.
X'E4'	X'08'	IKQOPNRP	LSR is specified, but there is no resource pool defined; may also be caused by problems while loading the Resource Table.
X'E8'	X'08'	IKQOPNRD	Reset was specified for a non-reusable data set and the data set is not empty

[1] The contents of register 15 have the following meaning:

X'00'	Open was successfully completed
X'04'	All ACBs were opened successfully, but one or more ACBs had a warning message
X'08'	One or more ACBs were not successfully opened

Note that register 15 contains the worst return code encountered while opening a list of ACBs. This means that some of the ACBs in the list may have been opened successfully, even though register 15 contains X'04' or X'08'.

[2] This code is ignored by OPEN, but is meaningful to catalog and DADSM routines.

Figure 6.9 OPEN error code-to-module relationships (part 3 of 3)

Error Code (ACBERFLG)	Return Code (Reg.15)	Issuing Module	Meaning
X'02'	X'02'	IKQCLO00	Invalid control block ID or ACB address not in OAL
X'04'	X'04'	IKQCLO00 $$BOVSAM	ACB is already closed or invalid control block structure
X'88'	X'08'	IKQCLO00 IKQCLCAT	VSAM could not obtain a continous area of virtual storage large enough for the work area
X'90'	X'08'	IKQCLCAT	A permanent I/O error occurred while VSAM was reading or writing a catalog entry.
X'B8'	X'04'	IKQCLO00	An internal error occurred while VSAM was completing outstanding I/O requests
X'BC'	X'D4'	IKQCLO00	The ACB is busy; it is being used, for example, by a SHOWCB or TESTCB macro
X'E4'	X'08'	IKQCLO00	Resource pool is invalid

Figure 6.10 CLOSE and TCLOSE error code-to-module relationships

Return Code (Reg. 15)	Issuing Module	Meaning
X'00'	IKQSCAT	Request successfully completed
X'04'	IKQSCAT	The specified work area is less than the minimum size (64 bytes) or too small to display all associated objects (in this case, as many objects as possible are displayed).
X'08'	IKQSCAT	Either the ACB address is invalid, or the VSAM master catalog does not exist or could not be opened.
X'0C'	IJQSCAT	CDLOAD failure while VSAM routines were being loaded
X'14'	IKQSCAT	The specified object or control interval does not exist.
X'18'	IKQSCAT	An I/O error occurred while accessing the catalog.
X'1C'	IKQSCAT	The specified CI number is invalid.
X'20'	IKQSCAT	The specified object is not a cluster, data set, index, alternate index, path, or upgrade set.
X'28'	IKQSCAT	The SHOWCAT module received an unexpected error code from catalog management.
X'2C'	IKQSCAT	Error while searching the label cylinder for the data set name

Note: In the case of return codes 0C, 14, 18, 1C, or 28, the SHOWCAT work area contains the return code and reason code issued by catalog management and the ID of the module in which the error was detected. In the case of return code X'2C', the work area contains the return code from IKQLAB, as shown in Figure 6.9. The format of the work area is then:

Offset	Length	Description
0	2	Length of work area
2	2	Catalog management or IKQLAB return code
4	2	Catalog management reason code
6	2	Catalog management module ID

Figure 6.11 SHOWCAT error code-to-module relationships

Macro	Return Code (Reg. 15)	Issuing Module	Description
BLDVRP	X'00'	IKQBRP	Request was successfully completed
	X'04'	IKQBRP	A resource pool already exists for the partition
	X'0C'	IKQBRP	CDLOAD failure
	X'14'	IKQBRP	STRNO is specified as less than one or more than 255
	X'18'	IKQBRP	The specified BUFFERS parameter is invalid
DELVRP	X'00'	IKQDRP	Request was successfully completed
	X'04'	IKQDRP	There is no resource pool to be deleted
	X'08'	IKQDRP	There is at least one open data set still using the resource pool
	X'0C'	IKQDRP	CDLOAD failure

Figure 6.12 Error codes for BLDVRP and DELVRP macros

Service aid phases are available for:

- Enabling and disabling snap dumps within the VSAM component.
- Obtaining snap dumps of control blocks.
- Using UPSI to obtain diagnostic information for the VSAM catalog.
- Maintaining DSCBs in the VTOC and VOL1 labels on DASD.
- Loading a VSAM phase or a program you have written.

The service aid phases IKQVDUMP and $$BCVS03 are included in the link-edit of VSAM. The other three phases, IKQVEDA, IKQVDU, and $$BCVS04 can be placed in the core image library by executing the following job.

```
// JOB          JOBNAME
// OPTION       CATAL
   INCLUDE      IKQCLNLK
/*
// EXEC         LNKEDT,REAL
/&
```

Enabling and Disabling Snap Dumps

The following snap points are available in VSAM. Each snap ID, if enabled with IKQVEDA, will produce the result indicated. If VSAM is running in the SVA, it must be reloaded from the core image library after the snap dump has been enabled in order to activate the dump.

Snap number	Result of Enabling this Snap
0001	This snap allows Catalog Management diagnostic information to be obtained. (See section *Using UPSI to obtain Diagnostic Information for the VSAM Catalog* for details.) As snap 0001 uses the UPSI byte, it cannot be run when the user program in the partition also uses the UPSI byte.
0002	This snap enables the Buffer Manager trace, which provides the current usage of VSAM buffering.
0003	This snap enables the CLOSE control block dump at the beginning of CLOSE processing.
0004	This snap enables the VSAM I/O trace facility.
0005	This snap enables the I/O error trace.
0006	This snap enables the OPEN control block dump facility when open processing is complete.
0007	This snap enables the OPEN error trace. Control blocks are printed if an error occurs during open processing.

0008	This snap enables the Catalog Management I/O trace. All I/O operations done by catalog management are printed on SYSLST.
0009	This snap enables the VSAM Record Management error handler trace, allowing display of control blocks for any error detected by VSAM record management.
0010	This snap disables automatic close

IKQVEDA is called by:

```
// EXEC IKQVEDA
```

The routine will print on SYSLOG:

```
ENTER FUNCTION ENABLE|DISABLE|END
```

You must enter either:

```
ENABLE SNAP = xxxx
              (where xxxx is one of the snap numbers)
```

or

```
DISABLE SNAP = xxxx
```

or

END (to terminate processing).

The program will look for a private core image library and print:

```
NO PRIVATE CORE IMAGE LIBRARY ASSIGNED
```

if it cannot be found and will then look in the core image library for the VSAM phase needed.

If the phase needed cannot be found in a library the program will inform you with the following message:

```
phase NOT FOUND IN THE SYSTEM PRIVATE
CORE IMAGE LIBRARY
```

(where phase is the actual phase name)

Any error in input will result in the INVALID REPLY message and the ENTER FUNCTION message is reissued.

The program can only be ended by the END reply as noted earlier.

The following examples illustrate the use of IKQVEDA to enable and disable SNAP 0001:

```
// EXEC IKQVEDA
   ENTER FUNCTION ENABLE|DISABLE|END
   ENABLE SNAP = 0001
   NO PRIVATE CORE IMAGE LIBRARY ASSIGNED
   SNAP 0001 ENABLED
   ENTER FUNCTION ENABLE|DISABLE|END
   DISABLE SNAP = 0001
   NO PRIVATE CORE IMAGE LIBRARY ASSIGNED
   SNAP 0001 DISABLED
   ENTER FUNCTION ENABLE|DISABLE|END
   END
```

IKQVDUMP enables you to print out snap dumps of record management and catalog control blocks. Code is provided at certain points in VSAM modules which is nonoperational so far as normal execution of the modules is concerned. Refer to *Enabling and Disabling Snap Dumps,* above.

IKQVDUMP is called by the following sequence of instructions (see also *Loading a VSAM phase or a Program You Have Written*):

```
          LA    1,PARMLIST
          SVC   2
          .
          .
          .
PARMLIST  DC    CL8'$$BCVS03'     B transient
          DC    CL8'IKQVDUMP'     phase that provides dump
                                  of control blocks
```

When the program has completed processing, $$BCVS03 returns the program to the instruction immediately following the SVC instruction.

Note that IKQVDUMP requires SYSLST to be assigned to a printer; assigment to disk or tape will result in an error.

Figure 6.11 shows the description and format of the parameter list that follows the two phase names in the above calling sequence.

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
0	0	1	PARMSW1	First byte of parameter list
		1...	PARMAMBL	Dump the AMBL
		.1..	PARMACB	Dump the ACB
		..1.	PARMAMDS	Dump the AMDSB
		...1	PARMARDB	Dump the ARDB
		.... 1...	PARMBCB	Dump the BCB
		.... .1..	PARMBUFE	Dump the buffer
		.... ..1.	PARMEDB	Dump the EDB
		1	PARMLPMB	Dump the LPMB
1	1	1	PARMSW2	Second byte of parameter list
		1...	PARMCCW	Dump the CCW
		.1..	PARMPLH	Dump the PLH
		..1.	PARMBHD	Dump the BHD
		...1	PARMRPL	Dump the RPL
		.... 1...	PARMEXCP	Dump the EXCPAD work area
		.... .1..	PARMCAT	Dump the catalog blocks
		.... ..1.	PARMDATA	Dump the non-catalog blocks
		1	PARMTHB	Dump the THB
2	2	1	PARMSW3	Third byte of parameter list
		1...	PARMOPEN	Dump the open work area
		.1..	PARMCLOS	Dump the close work area
		..1.	PARMCIW	Dump the control interval split area
		...1	PARMVLST	Dump the volume list
		.... 1...	PARMREGS	Dump the registers
		.... .1..	PARMCECL	Dump the control interval exclusive control list
		.... ..1.	PARMODLB	Dump the open DLBL
		1	PARMREQR	Dump the requester's registers

Figure 6.13 IKQVDUMP parameter list description and format (part 1 of 3)

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
3	3	1	PARMSW4	Fourth byte of parameter list
		1...	PARMPAMB	1=Pointer to start dump is in parameter list (PARMAMBA) 0=Pointer to start dump is in register 11
		..1.	PARMCCAA	1=Pointer to CCA 0=Pointer to AMBL
		...1	PARMRTNA	Call the test routine
		.... 1...	PARMHDID	Dump the header ID
		.x.. .xxx		Available
4	4	4	PARMAMBA	Pointer to start dump
8	8	4	PARMID	Pointer to header
8	8	1	PARMIDLN	Length of the header
9	9	3	PARMIDAD	Address of the ID
12	C	1	PARMSW5	Fifth byte of parameter list
		1...	PARMCCA	Dump the CCA
		.1..	PARMCADL	Dump the CCA DLBL
		..1.	PARMCADP	Dump the CCA DADSM parameter list
		...1	PARMCARA	Dump the CCA record areas
		.... 1...	PARMCPL	Dump the catalog parameter list (CTGPL)
		.... .1..	PARMPLDN	Dump the CTGPL data set name
		.... ..1.	PARMPLNN	Dump the CTGPL new name
		1	PARMPLPW	Dump the CTGPL password
13	D	1	PARMSW6	Sixth byte of parameter list
		1...	PARMPLCN	Dump the CTGPL catalog name
		.1..	PARMPLCI	Dump the CTGPL control interval number
		..1.	PARMPLDL	Dump the CTGPL file CTGDDNM field
		...1	PARMPLWA	Dump the CTGPL work area
		.... 1...	PARMCFL	Dump the catalog field parameter list (CTGFL)
		.... .1..	PARMFLFD	Dump the CTGFL fields
		.... ..1.	PARMFLFN	Dump the CTGFL field name
		x		Available
14	D	1	PARMSW7	Seventh byte of the parameter list
		1...	PARMCFV	Dump the catalog field vector table (CTGFV)
		.1..	PARMFVDL	Dump the CTGFV file name
		..1.	PARMFVEN	Dump the CTGFV entry name
		...1	PARMFVKR	Dump the CTGFV key range list
		.... 1...	PARMFVVL	Dump the CTGFV volume serial list
		.... .1..	PARMDPDL	Dump the DADSM parameter list DLBL
		.... ..1.	PARMDPIO	Dump the DADSM parameter list I/O area
		1	PARMDPWA	Dump the DADSM parameter list work area

Figure 6.13 IKQVDUMP parameter list description and format (part 2 of 3)

Offset Dec	Offset Hex	Bytes and Bit Pattern	Field Name	Description
15	F	1	PARMSW8	Eighth byte of parameter list
		1...	PARMDPSV	Dump the DADSM parameter list save I/O area
		.1..	PARMCBS	Dump the AMCBS
		..1.	PARMCAXW	Dump the CAXWA
		...1	PARMCXRL	Dump the CAXWA RPL
		.... 1...	PARMCXDR	Dump the CAXWA DSCB read-in work area (DRWA)
		.... .1..	PARMCMSW	Dump the CMS work area
		xx		Available
16	10	8	PARMRTNN	Name of test routine

Figure 6.13 IKQVDUMP parameter list description and format (part 3 of 3)

Testing if a Dump is Required

IKQVDUMP allows a phase to be called before a dump is taken to see if a dump is desired. (The name of the test routine must be inserted into the parameter list at field name PARMRTNN.) The phase can use any logic to determine whether a dump is needed, and this logic will override a call for a dump if it is not needed. If a 0 is returned in register 15, the dump will be taken; if register 15 holds a nonzero return, the dump will not be taken.

The registers on entry to the test routine have the following contents:

R2 = Pointer to the parameter list
R11 = Caller's register 11
R13 = Pointer to 18-word save area
R14 = Return address of calling phase
R15 = Address of entry point

Using UPSI to Obtain Diagnostic Information for the VSAM Catalog

Manipulation of the UPSI job control statement enables you to screen catalog return codes and obtain a snap dump, cancel a job (which causes a full dump to be taken), or simply continue processing. You must first use IKQVEDA to enable Snap = 0001. Otherwise the UPSI statement will be inoperative. As snap 0001 uses the UPSI byte, it cannot be run when the user program in the partition also uses the UPSI byte.

The purpose of this service aid is to diagnose catalog errors that occur while running any program that causes the VSAM catalog to execute. Typically this would be an Access Method Services module or a record management program you have written.

The // UPSI nnnnnnnn job control statement must precede the // EXEC [progname] statement. If no UPSI statement is included, the default is // UPSI 00000000 (see type 3 request below).

On exit from catalog management after processing, a message will be printed out depending on the type of UPSI bit setting you have selected. Some messages require a reply from the operator. The return codes in the message are obtained from register 15. The format is:

```
** NNN,MN,RRR,FFFF,CCCCCCCCCCCCCCCC
```

where

NNN is the return code in decimal

MN are the last two characters of the module name which issued the error. This is blank in case of error code 0.

RRR is the reason code in decimal

FFFF is one of the following catalog management functions that had been processed:

DEFC	(define catalog)
DEFA	(define non-VSAM data set)
DEFS	(define space)
DEF	(define VSAM data set)
ALT	(alter)
DELC	(delete catalog)
DELS	(delete space)
DEL	(delete VSAM or non-VSAM data set)
LSTC	(list catalog)
UPD	(update or update-extend)
LOC	(locate)

C...C is either the control interval number in decimal or the first 16 characters of the data set name or volume serial number in EBCDIC.

If a reply is required from the system operator for certain types of requests, the operator must enter one of the following replies from the system console:

- Type in SNAP to get a snap dump by means of IKQVDUMP (see *IKQVEDA for enabling snap dumps*). The message will then be repeated and the operator should press the END key to continue processing.
- Type in CANCEL to cancel the job and obtain a full dump.
- Press the END key to resume processing.

The following paragraphs describe the four types of UPSI settings you can use to elicit a message and/or to determine the degree of return code screening done:

Type 1 UPSI Setting. If you want to obtain an operator message for all VSAM catalog return codes (including 0), you must include one of the following statements:

`// UPSI 11000000` No reply is required from the operator

`// UPSI 01100000` A reply is required from the operator

Type 2 UPSI Setting. An operator message is issued only if the return code is not 0 for the following statements:

`// UPSI 10000000` No reply is required from the operator

`// UPSI 01000000` A reply is required from the operator

Type 3 UPSI Setting. An operator message is not issued if one of the following conditions exists:

1. the Access Method Services command being processed was a LISTCAT and the return code is 8, or

2. the return code is 0, 40, 68, or 160

(these codes occur during normal processing and are, therefore, excluded).

If neither of these conditions exists, an operator message is issued for the following statements:

`// UPSI 00000000` No reply is required from the operator

`// UPSI 01110000` No reply is required from the operator

Type 4 UPSI Setting. If you want an operator message on a specific return code, you must include the following statements:

`// UPSI 00nnnnnn` nnnnnn is set to the value, in binary, of the code divided by 4. A reply is required from the operator

Maintaining DSCBs in the VTOC and VOL1 Labels on DASD

A VSAM DADSM service aid has been provided to assist the programmer and operator in maintaining the VTOC and VOL1 labels on DASD devices.

The following procedures should be followed to use IKQVDU at the system console for such maintenance. The key difference in the three procedures is the presence, or absence, of a // UPSI job control statement. Steps of the procedure in lower case letters are typed in at the console; steps in upper case letters are printed out.

Procedure 1	Explanation
`// assgn sys000,x'cuu'` (press END key)	*cuu* points at the volume you want to use.
`// upsi 1` (press END key)	This job control statement is optional. If it is included, the following events take place on the volume that was assigned to SYS000: • The VSAM volume ownership bit and CRA TT pointer in the F4 DSCB are reset. • The entire VTOC is scratched, that is, empty DSCBs are written over existing F1, F2, and F3 DSCBs, with the exception of DSCBs that have names starting with the characters 'DOS.' or 'PAGE'. • An operator authorization prompt is issued if the DSCB to be scratched is security protected.
`// exec ikqvdu,size=auto` (press END key)	Start execution of the IKQVDU phase

Procedure 2	Explanation
`// assgn sys000,x'cuu'` (press END key)	*cuu* points at the volume you want to use.
`// upsi 11` (press END key)	This job control statement is optional. If it is included, the following events take place on the volume that was assigned to SYS000: • The VSAM volume ownership bit and CRA TT pointer in the F4 DSCB are reset. • The entire VTOC is scratched, that is, F0 DSCBs are written over existing F1, F2, and F3 DSCBs, with the exception of DSCBs that have names starting with the characters 'DOS.' or 'PAGE'.
`// exec ikqvdu,size=auto` (press END key)	Start execution of the IKQVDU phase.

Procedure 3	Explanation
`// assgn sys000,x'cuu'` (press END key)	*cuu* points at the volume you want to use.

`// exec ikqvdu,size=30k` **(press END key)**	Start execution of the IKQVDU phase.

`SPECIFY FUNCTION OR REPLY '?' FOR OPTIONS READY`
`?`
(press END key)
	The character ? causes a list of the various functions that IKQVDU performs to be printed out at the system console.

```
TO SET THE VOLUME OWNERSHIP FLAG REPLY 'SET OWNERSHIP'
TO SET THE CRA POINTER REPLY 'SET OWNERSHIP'
TO RESET THE VOLUME OWNERSHIP FLAG AND CRA POINTER REPLY
    'RESET OWNERSHIP' OR 'RESET CRA'
TO SET THE SECURITY FLAG IN A F1 DSCB REPLY 'SET
    SECURITY'
TO RESET THE SECURITY FLAG IN A F1 DSCB REPLY 'RESET
    SECURITY'
TO REMOVE A DSCB FROM THE VTOC REPLY 'SCRATCH'
TO RENAME A DSCB REPLY 'RENAME'
TO ALLOCATE A DSCB REPLY 'ALLOCATE'
TO REINITIATE PROCESSING REPLY 'RESTART'
TO ALTER OR DISPLAY A DASD VOL1 LABEL
REPLY 'CLIP LABEL=SER=N..N' OR 'CLIP LABEL=DISPLAY'
TO TERMINATE PROCESSING REPLY 'END'
READY
```

You can avoid printing out this list of functions simply by specifying the function you wish as follows:

Procedure	**Explanation**
`set ownership` **(press END key)**	Causes the VSAM ownership bit to be set in the F4 DSCB and optionally allows the user to set the CRA TT pointer.
`reset CRA or` `reset ownership`	Causes the VSAM ownership bit and CRA TT pointer to be reset in the F4 DSCB.
`set security` **(press END key)**	Causes the security bit to be set in the F1 DSCB. When the console responds with ENTER DSN, reply with the data set name of the DSCB to be modified.
`reset security` **(press END key)**	Causes the security bit in the F1 DSCB to be reset. When the console responds with ENTER DSN, reply with the data set name of the DSCB to be modified.
`scratch dsn=dsname` **(press END key)**	Causes the DSCB with the specified data set name to be scratched.

`scratch vtoc` (press END key)	Causes the entire VTOC to be scratched with the exception of data set names starting with the characters 'DOS.' and 'PAGE'. In addition, an operator-authorization prompt will be issued if the DSCBis security-protected or describes a catalog.
`rename` (press END key)	Causes the DSNAME portion of the F1 DSCB to be changed. When the console responds with ENTER OLD DSN, reply with the data set name of the DSCB to be changed. When the console responds with ENTER NEW DSN, reply with the new data set name.
`allocate` (press END key)	Causes a new DSCB to be created and written in the VTOC. In order to utilize this function, a DLBL/EXTENT job control statement must be provided (see *DOS/VS System Control Statements,* GC33-5376). When the console responds with ENTER FILENAME, reply with the same filename as that in the DLBL statement referred to above. When the console responds with ENTER NEW DSN, reply with the data set name of the data set to be created. When the console responds with DO YOU WISH TO SECURITY PROTECT THIS DATA SET? reply YES or NO. A reply of YES causes the data security bit to be set in the F1 DSCB. A reply of NO causes the data security bit to be reset in the F1 DSCB.
`restart` (press END key)	Causes processing to be reinitiated with a READY prompt. This keyword can be used as a response to any operator prompt.
`clip label=display` (press END key)	Causes the volume serial number to be displayed on the system console.
`clip label=ser=n..n` (press END key)	Causes the existing volume serial number to be changed to the one specified as n..n.

`end` (press END key)	Causes processing to terminate.

If an error occurs during execution of IKQVDU,

```
**ERROR** DADSM RETURN CODE IS nnn message
```

prints out on the system console. The following shows the message code (nnn), the associated message, and the action required to correct the condition.

Example:

```
** ERROR** DADSM RETURN CODE IS 020 VTOC FULL
```

004 I/O ERROR WHILE READING VOLUME LABEL
Action: If the problem was not caused by a hardware error, restore the volume.

008 VOLUME NOT MOUNTED
Action: Mount the correct volume.

012 I/O ERROR ON VTOC
Action: If the problem was not caused by a hardware error, restore the volume.

016 DUPLICATE NAME ON VOLUME
Action: Choose another filename or scratch the original file from the volume. If duplication is due to key ranges, ensure each UNIQUE key range is on a separate volume.

020 VTOC FULL
Action: Delete any non-VSAM files or VSAM data spaces no longer needed from the volume to make additional Format 1 labels available, or reinitialize the volume with a larger VTOC.

024 EXTENT OVERLAPS EXPIRED FILE
Action: Examine the VTOC listing to determine where the overlap occurred. Correct the EXTENT statement causing the error. To delete the expired file, open a DTF using the same file-ID as that of the expired file, and instruct the operator to reply DELETE to message 4n33A when it is issued.

028 EXTENT OVERLAPS UNEXPIRED FILE
Action: Compare the high and low extent limits on the EXTENT statement or LSERV output with the file or data space limits on the VTOC display. If the extents overlap, correct the EXTENT statement in error.

032 EXTENT OVERLAPS PROTECTED UNEXPIRED FILE
Action: Examine the VTOC to determine where the overlap occurred. Correct the EXTENT statement causing the error. If necessary, use another volume.

036 EXTENT OVERLAPS VTOC
Action: Execute LVTOC. The Format 4 label (the first label in the VTOC display) contains the extent limits of the VTOC. If the program being executed uses a temporary label set and overlaps the VTOC, correct the

EXTENT statements that overlap. If the job uses standard or partition standard labels, use the LSERV output to correct the extents of the overlapping file, VSAM data space, or UNIQUE VSAM file. Then rebuild the appropriate label tracks.

040 REQUIRED EXTENTS MISSING
Action: If temporary labels were used, match the extents on the incoming EXTENT card with the extents in the LVTOC output. If standard (permanent) labels were used, match the extents in the LSERV output with those in the LVTOC output.

044 DSCB NOT FOUND
Action: Use the LVTOC output to check for all file labels used in OPEN macros. If the file has been destroyed, it was probably due to deletion of overlapping extents on an unexpired file, and the file must be rebuilt.

056 EXTENT OVERLAPS PROTECTED EXPIRED FILE
Action: Examine the VTOC listing to determine where the overlap occurred. Correct the EXTENT statement causing the error. If it is not necessary to save the expired file, open a DTF using the same file-ID as that of the expired file, and instruct the operator to reply DELETE to message 4n33A when it is issued.

064 GETVIS FAILURE ENCOUNTERED
Action: Allocate GETVIS area. If VSAM is running in the SVA, re-IPL and specify a new value for SET SVA. If VSAM is running in a partition, rerun the job in a larger partition.

072 CDLOAD FAILURE ENCOUNTERED
Action: Either the CDLOAD directory or the GETVIS area is full. Allocate more space.

080 OVERLAP AMONG NEW EXTENTS
Action: If DLBL and EXTENT statements are included in the program, determine the conflicting extents and correct them. If a standard label set is being used, use the LSERV output to locate and correct the conflicting file extents, and rebuild the standard label tracks.

Loading a VSAM Phase or a Program You Have Written

If you want to load and transfer control to and from a selected VSAM phase or a program you have written, you can use B-transient $$BCVS03 without destroying any registers in the following calling sequence:

```
          LA    1,PARMLIST
          SVC   2
          .
          .
          .
PARMLIST  DC    CL8'$$BCVS03'     B transient
RTNNAME   DC    CL8'XXXXXXXX'     Name of phase or program
                                  you have written
USERLIST  DC                      Parameter list for phase
          .                       'XXXXXXXX'
          .
          .
```

When control is received by 'XXXXXXXX', the registers have the following contents:

R0	=	Address of a work area (the size of the work area is specified by a halfword at offset 4 of 'XXXXXXXX' phase)
R1	=	Pointer to user's parameter list (USERLIST)
R2-13	=	Remain the same as they were when SVC 2 was issued
R14	=	Return address of calling module
R15	=	Address of entry point in 'XXXXXXXX'

Control is returned from 'XXXXXXXX' by a BR 14 instruction.

Glossary

Definitions of Terms Used In This Book

Access Method Services: A multifunction service program that defines VSAM data sets and allocates space for them, converts indexed sequential data sets to key-sequenced data sets with indexes, modified data-set attributes in the catalog, recognizes data sets, facilitates data portability between operating systems, creates backup copies of data sets and indexes, helps make inaccessible data sets accessible, and lists data-set records and catalog entries.

address direct access: The retrieval or storage of a data record identified by its RBA (relative byte address) independent of the record's location relative to the previously retrieved or stored record. (See also keyed direct access, addressed sequential access, and keyed sequential access.)

addressed sequential access: The retrieval or storage of a data record in its entry (RBA) sequence relative to the previously retrieved or stored record. (See also keyed sequential access, addressed direct access, and keyed direct access.)

allocation chain (AC): All allocation units containing control blocks for the same ACB.

allocation unit (AU): One or more pages of virtual storage containing control blocks referenced by record management.

alternate index: A collection of index entries, related to a give base cluster and organized by a key other than the prime key of the associated base data records. Its function is to provide an alternate means of locating records in the data portion of the base cluster.

alternate index upgrade: The process of reflecting changes made to a base cluster in its associated alternate index(es). (See also upgrade set.)

alternate key: A key, other than the prime key, used to form an alternate index.

application: As used in this publication, the use to which an access method is put or the end result that it serves; contrasted to the internal operation of the access method.

backward processing: A variation of sequential processing, whereby the previous, rather than the next, record in the entry, key, or relative-record sequence is retrieved.

base catalog record: The first catalog record (control interval) that describes the VSAM object. This record contains the object's data set name, cluster name, or volume serial number in the ENTNAME field. This record also contains the header fields required for the object. The base catalog record can contain group occurrence pointers that point to group occurrences in the base catalog record, or that point to group occurrences in extension records (vertical extension). The base catalog record's extension pointer can point to a control interval that continues the information (group occurrence pointers) contained in the base catalog record (horizontal extension).

base cluster: A key-sequenced or entry-sequenced data set over which one or more alternate indexes are built.

candidate volume: A direct-access storage volume that has been defined in a VSAM catalog as a VSAM volume; VSAM can automatically allocate space on this volume, as needed. (See also overflow volume.)

catalog: (See VSAM catalog.)

cluster: A combination of related VSAM data sets, identified by one name in the VSAM catalog, that is, a key-sequenced data set and its index or an entry-sequenced data set alone.

collating sequence: An ordering assigned to a set of items, such that any two sets in that assigned order can be collated. As used in this publication, the order defined by the System/370 8-bit code for alphabetic, numeric, and special characters.

component: As used in this manual, a group of modules that perform a function, such as Open.

compression: (See key compression.)

control area: A group of control intervals used as a unit for formatting a data set before adding records to it. Also, in a key-sequenced data set, the set of control intervals pointed to by a sequence-set index record; used by VSAM for distributing free space and for placing a sequence-set index record adjacent to its data.

control-area split: The movement of the contents of some of the control intervals in a control area to a newly created control area, to facilitate the insertion or lengthening of a data record when there are no remaining free control intervals in the original control area.

control interval: A fixed-length area of direct-access storage in which VSAM stores records and distributes free space. It is the unit of information transmitted to or from direct-access storage by VSAM, independent of blocksize.

control interval access: The retrieval or storage of the contents of a control interval.

control-interval split: The movement of some of the stored records in a control interval to a free control interval, to facilitate the insertion or lengthening of a record that won't fit in the original control interval.

CRA: Catalog recovery area. An entry-sequenced data set which exists on each volume owned by a recoverable catalog, including the catalog volume itself. The CRA contains records which describe the volume and the data sets on the volume.

data integrity: Preservation of data or programs for their intended purpose. As used in this publication, the safety of data from inadvertent destruction or alteration.

data record: A collection of items of information from the standpoint of its use in an application and not from the standpoint of the manner in which it is stored (see also stored record).

data security: Prevention of access to or use of data or programs without authorization. As used in this publication, the safety of data from unauthorized use, theft, or purposeful destruction.

data set: The major unit of data storage and retrieval in the operating system, consisting of data in a prescribed arrangement and described by control information to which the system has access. As used in this publication, a collection of fixed- or variable-length records in direct-access storage, arranged by VSAM in key sequence or in entry sequence. (See also key-sequenced data set and entry-sequenced data set.)

data space: A storage area defined in the volume table of contents of a direct-access volume for the exclusive use of VSAM to store data sets, indexes, and catalogs.

direct access: The retrieval or storage of data by a reference to its location in a data set rather than relative to the previously retrieved or stored data. (See also addressed direct access and keyed direct access.)

distributed free space: Space reserved within the control intervals of a key-sequenced data set for inserting new records into the data set in key sequence; also, whole control intervals reserved in a control area for the same purpose.

entry-sequence: The order in which data records are physically arranged in direct-access storage, without respect to their contents. (Contrast to key sequence.)

entry-sequenced data set: A data set whose records are loaded without respect to their contents, and whose relative byte addresses cannot change. Records are retrieved and stored by addressed access, and new records are added at the end of the data set.

exclusive control: (See hold.)

extension record: The continuation of a catalog record that contains group occurrence pointers and their group occurrences. Group occurrence pointers in an extension record always point to group occurrences within the extension record. The extension record's extension pointer can point to a control interval that contains part of a group occurrence too large to fit in the extension record (horizontal extension).

extent: A continuous space allocated on a direct-access storage volume, reserved for a particular data space or data set.

external procedure: A procedure that can be called by any other VSAM procedure; a procedure whose name is in the module's (assembler listing) "external symbol dictionary."

field: In a record or a control block, a specified area used for a particular category of data or control information.

file: (See data set.)

free space: (See distributed free space.)

free space percentage: (See packing factor.)

generic key: A high-order portion of a key, containing characters that identify those records that are significant for a certain application. For example, it might be desirable to retrieve all records whose keys begin with the generic key AB, regardless of the full key values.

group code: A code that identifies the type of group occurrence. (See Field Name Dictionary for a list of group codes.)

group occurrence: Related fields of information in catalog records. See *Group Occurrences in Catalog Records* in the *Data Areas* section for further details.

group occurrence pointer: A field used to identify and locate a group occurrence by its displacement from the beginning of the record's group occurrences (the group occurrence is in the same control interval as the group occurrence pointer) or by a control interval number (the group occurrence point is in the base catalog record or its extension and the group occurrence is in an extension record). Group occurrence pointers are grouped by type code and are in ascending sequence by sequence number.

high-used RBA: The next byte past the end of the last control interval containing significant data, for ESDA; otherwise, the RBA at which the last SEOF is written.

high-water RBA: The high-used RBA of a data set.

hold: Exclusive control exercised over data or index during an update, erase, or insert operation to prevent another request from making interim changes between initiation and completion of the original request.

horizontal extension: An extension record pointed to by a catalog record's extension field. (See also vertical extension.)

horizontal pointer: A pointer in a sequence set index record that gives the location of the next index record in the same sequence set; used for keyed sequential access.

index: As used in this publication, an ordered collection of pairs, each consisting of a key and a pointer, used by VSAM to sequence and locate the records of a key-sequenced data set; organized in levels of index records. (See also index level, index set and sequence set.)

index entry: A key and a pointer paired together, where the key is the highest key (in compressed form) entered in an index record in the next lower level or contained in a data record in a control interval, and the pointer gives the location of that index record or control interval.

index level: A set of index records that order and give the location of records in the next lower level or of control intervals in the data set that it controls.

index record: A collection of index entries that are retrieved and stored as a group. (Contrast to data record.)

index replication: The use of an entire track of direct-access storage to contain as many copies of a single index record as possible; reduces rotational delay.

index set: The set of index levels above the sequence set. The index set and the sequence set together comprise the index.

integrity: (See data integrity.)

internal procedure: A procedure that can be called only by another procedure within the module. (See also external procedure.)

I/O threshold: The maximum number of buffers that can be filled with data before I/O will be started.

ISAM interface: A set of routines that allow a processing program coded to use ISAM (Indexed Sequential Access Method) to gain access to a VSAM key-sequenced data set.

job catalog: A catalog made available for a job by means of the filename IJSYSUC in the corresponding DLBL job statement.

key: One or more characters within an item of data that are used to identify it or control its use. As used in this publication, one or more consecutive characters taken from a data record, used to identify the record and establish its order with respect to other records. (See also key field and generic key.)

key compression: The elimination of characters from the front and the back of a key that VSAM does not need to distinguish the key from the preceding or following key in an index record; reduces storage space for an index.

key-field: A field located in the same position in each record of a data set, whose contents are used for the key of the record.

key-sequence: The collating sequence of data records, determined by the value of the key field in each of the data records. May be the same as, or different from, the entry sequence of the records.

key-sequenced data set: A data set whose records are loaded in key sequence and controlled by an index. Records are retrieved and stored by keyed access or by addressed access, and new records are inserted in the data set in key sequence by means of distributed free space. Relative byte addresses of records can change.

keyed direct access: The retrieval or storage of a data record by use of either an index that related the record's key to its relative location in the data set, or a relative-record number, independent of the record's location relative to the previously retrieved or stored record. (See also addressed direct access, keyed sequential access, and addressed sequential access.)

keyed sequential access: The retrieval or storage of a data record in its key or relative-record sequence relative to the previously retrieved or stored record, as defined by the sequence set of an index. (See also addressed sequential access, keyed direct access, and addressed direct access.)

mass sequential insertion: A technique VSAM uses for keyed sequential insertion of two or more records in sequence into a collating position in a data set; more efficient than inserting each record directly.

master catalog: The main VSAM catalog, that contains extensive data set and volume information required by VSAM to be able to locate data sets to allocate and deallocate storage space, to verify the authorization of a program or operator to gain access to a data set, and to accumulate usage statistics. (See also job catalog, user catalog.)

module: As used in this manual, a program unit that is identifiable by means of a symbolic name starting with IGG0 or IKQ.

nonunique: Space for a nonunique data set or index must be a suballocation from existing data spaces.

object: As used in this manual, a cluster, a data set, an index, a catalog, or a data space.

overflow volume: When space on a candidate volume is allocated by VSAM, that volume is then termed an overflow volume. (See also candidate volume.)

overlapped operation: An operation in which processing continues without waiting for completion of input or output which had been initiated.

packing factor: Percentage of the data object's space allocation to be reserved during its initial loading and during subsequent reorganization. (See also distributed free space.)

password: A unique string of characters stored in a catalog that a program, a computer operator, or a terminal user must supply to meet security requirements before a program gains access to a data set.

physical record: The smallest readable or writable unit of data that is stored on a direct-access storage device. Records are separated from each other by interrecord gaps.

pointer: An address or other indication of location. For example, an RBA is a pointer that gives the relative location of a data record or a control interval in the data set to which it belongs. (See also horizontal pointer and vertical pointer.)

portability: The ability to use VSAM data sets with different operating systems. Volumes whose data sets are cataloged in a user catalog can be demounted from storage devices of one system, moved to another system, and mounted on storage devices of that system. Individual data sets can be transported between operating systems using Access Method Services.

prime index: The index of a key-sequenced data set which is a base cluster, and thus has one or more alternate indexes. (See also index, alternate index.)

prime key: The key which is used to form the prime index. (See also key, alternate key.)

procedure: A functional unit of VSAM code that is entered only at one entry point and exits at the end of the procedure (the last line of the procedure's code). The procedure can call (transfer control, with a return to the procedure expected) other procedures within the module (internal calls) and can call other procedures in other VSAM modules (external calls). (See also internal procedure and external procedure.)

RBA: Relative byte address. The displacement of a data rcord or a control interval from the beginning of the data set to which it belongs; independent of the manner in which the data set is stored.

record: (See index record, data record, physical record, stored record.)

recoverable catalog: A catalog defined with the recoverable

attribute. Duplicate catalog entries are stored in CRAs, and can be used to recover data in the event of a catalog failure. (See also CRA.)

recovery mode: A user option that causes the data object's initial allocation of space to be written throughout with special records, the last of which is set to 0 and is termed the SEOF (software end of file) record. This must be done if VERIFY is to be used. (See also speed mode.)

relative byte address: (See RBA.)

relative-record data set: A data set whose records are loaded into fixed-length slots and numbered by the relative numbers of the slots they occupy.

relative-record number: A number that identifies not only the slot, or data space, in a relative-record data set but also the record occupying the slot. Used as a key for keyed access to a relative-record data set.

relative repetition number: An integer representing the position of a particular field in a group of repeating fields. For example, in EOV, the relative repetition number (RELREPNO) of a particular volume in the catalog data record of a particular cluster is that number of its occurrence in the volume repeating group. EOV uses the RELREPNO to obtain information about a particular volume in order to update the ARDB and EDB.

replication: (See index replication.)

reusable data set: A VSAM data set which can be reused as a work data set, regardless of its old contents.

routine: As used in this manual, an ordered set of instructions that may have frequent use, generally internal usage within a module.

scratch (adj.): Used to describe the contents of a buffer that are no longer valid.

scratch (v.): In buffer management, used to indicate that a buffer contains nothing of significance; in DADSM, to remove a DSCB.

section: A subdivision of an index record used to expedite location of the place in an index record where an entry-by-entry key search can begin.

security: (See data security.)

SEOF: (See software end of file.)

sequence set: The lowest level of the index of a key-sequenced data set; it gives the locations of the control intervals in the data set and orders them by the key sequence of the data records they contain. The sequence set and the index set together comprise the index.

sequential access: The retrieval or storage of a data record in either its entry sequence, its key sequence, or its relative-record number sequence relative to the previously retrieved or stored record. (See also addressed sequential access and keyed sequential access.)

skip sequential access: Keyed sequential retrieval or storage of records in ascending, non-consecutive sequence (with skips); VSAM scans the sequence set to find a record or a collating position.

slot: A fixed-length, numbered space in a relative-record data set which accepts one data record. (See also relative-record data set, relative-record number.)

software end of file: A control interval with a CIDF of 0 that marks the end of preformatted records in a data object's initial allocation of space when the user specifies recovery mode of processing. (See also recovery mode.)

spanned record: A logical record whose length exceeds the control interval size and thus crosses, or spans, one or more control interval boundaries within a single control area.

stored record: A data record, together with its control information, as stored in a direct-access storage device.

string: A string is a single record or a sequentially ordered set of records in a data set. The maximum number of strings (STRNO) to be processed concurrently in a data set is established when a data set is opened. The number of active RPLs determines the number of concurrent strings being processed at any point in time.

unique: (1) A unique data space is occuplied by only one VSAM data set, and cannot be shared with other data sets. (2) A unique alternate key is one which occurs in only one data record in the base cluster. The alternate index record containing this key thus has only one pointer to the base cluster.

upgrade set: All the alternate indexes that VSAM has been instructed to update whenever there is a change to the data of the related base cluster.

user catalog: An optional catalog used in the same way as the master catalog and pointed to by the master catalog. It lessens the contention for the master catalog and facilitates volume portability.

vertical extension: An extension record pointed to by a group occurrence pointer in the object's base catalog record or its horizontal extension. (See also base catalog record and horizontal extension.)

vertical pointer: A pointer in an index record of a given level that gives the location of an index record in the next lower level or the location of a control interval in the data set controlled by the index.

VSAM catalog: A key-sequenced data set containing extensive data-set and volume information that VSAM requires to locate data sets, to allocate and deallocate storage space, to verify authorization of a program or operator to gain access to a data set, and to accumulate usage statistics for data sets. (See also master catalog, job catalog, user catalog.)

Index

This index lists the modules by their descriptive names, followed by their symbolic names. The symbolic names, together with further information about the modules, can be found in the module directory in the *Directory* chapter.

* For further information, see the module directory in the *Directory* chapter.

* For further information, see the module directory in the *Directory* chapter.

* For further information, see the module directory in the *Directory* chapter.

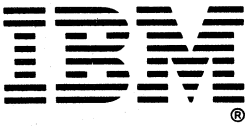
* For further information, see the module directory in the *Directory* chapter.

* For further information, see the module directory in the *Directory* chapter.

* For further information, see the module directory in the *Directory* chapter.

* For further information, see the module directory in the *Directory* chapter.

DOS/VS LIOCS Volume 4 VSAM Logic (File No. S370-30) Printed in U.S.A. SY33-8562-5



International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N. Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N. Y., U. S. A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N. Y., U. S. A. 10601

DOS/VS LIOCS Volume 4
VSAM Logic
SY33-8562-5

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM shall have the nonexclusive right, in its discretion, to use and distribute all submitted information, in any form, for any and all purposes, without obligation of any kind to the submitter. Your interest is appreciated.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. 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.*

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

• What is your occupation? _____

• How do you use this publication:

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

Your comments:

If you would like a reply, please supply your name and address on the reverse side of this form.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

Reader's Comment Form

Cut or Fold Along Line

Fold and tape | Please Do Not Staple | Fold and Tape

First Class
Permit 10
Endicott
New York

Business Reply Mail
No postage stamp necessary if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
Department G60
P. O. Box 6
Endicott, New York 13760

Fold | Fold

If you would like a reply, *please print:*

Your Name _____

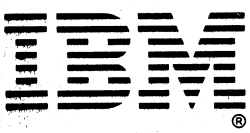
Company Name _____ *Department* _____

Street Address _____

City _____

State _____ *Zip Code* _____

IBM Branch Office serving you _____



International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N. Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N. Y., U. S. A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N. Y., U. S. A. 10601

DOS/VS LIOCS Volume 4 VSAM Logic (File No. S370-30) Printed in U.S.A. SY33-8562-5

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM shall have the nonexclusive right, in its discretion, to use and distribute all submitted information, in any form, for any and all purposes, without obligation of any kind to the submitter. Your interest is appreciated.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. 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.*

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

• What is your occupation? _____

• How do you use this publication:

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

Your comments:

If you would like a reply, please supply your name and address on the reverse side of this form.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

Reader's Comment Form

Cut or Fold Along Line

Fold and tape | Please Do Not Staple | Fold and Tape

First Class
Permit 10
Endicott
New York

Business Reply Mail
No postage stamp necessary if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
Department G60
P. O. Box 6
Endicott, New York 13760

Fold | Fold

If you would like a reply, *please print:*

Your Name _____

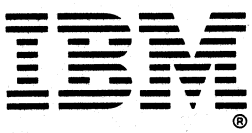
Company Name _____ *Department* _____

Street Address _____

City _____

State _____ *Zip Code* _____

IBM Branch Office serving you _____



International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N. Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N. Y., U. S. A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N. Y., U. S. A. 10601

DOS/VS LIOCS Volume 4 VSAM Logic (File No. S370-30) Printed in U.S.A. SY33-8562-5