



MVS/370  
BDAM Logic

Program  
Product





Contains Restricted Materials of IBM  
Licensed Materials – Property of IBM



**MVS/370**  
**BDAM Logic**

Data Facility Product 5665-295  
Release 1.1

LY26-3913-1

**Third Edition (December 1985)**

This is a major revision of, and makes obsolete, LY26-3913-0.

This edition applies to Release 1.1 of MVS/370 Data Facility Product, Program Product 5665-295, and to any subsequent releases until otherwise indicated in new editions or technical newsletters.

The changes for this edition are summarized under "Summary of Amendments" following the preface. Specific changes are indicated by a vertical bar to the left of the change. These bars will be deleted at any subsequent republication of the page affected. Editorial changes that have no technical significance are not noted.

Changes are made periodically to this publication; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/370 and 4300 Processors Bibliography, GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM program product in this publication is not intended to state or imply that only IBM's program product may be used. Any functionally equivalent program may be used instead.

Publications are not stocked at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California, U.S.A. 95150. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

This is a licensed document that contains restricted materials of International Business Machines Corporation. © Copyright International Business Machines Corporation 1974, 1976, 1983, 1985. All rights reserved.

## PREFACE

This publication describes the internal logic of the basic direct access method (BDAM) for MVS/370 DFP. It is intended as a reference book for IBM programming support representatives and system programmers who maintain or alter BDAM routines.

## ORGANIZATION

This publication contains the following:

- "Introduction" on page 1 explains what BDAM does, the macros, records, and options it uses, and how it relates to the rest of the operating system.
- "Method of Operation" on page 12 describes the major operations BDAM performs, and the input to and output from these operations.
- "Module Directory" on page 58 lists each BDAM module, and indicates the method of operation diagram or module flow diagram that describes it.
- "Data Areas" on page 60 shows the major data areas used by BDAM.
- "Diagnostic Aids" on page 87 lists the messages and codes issued by BDAM, and diagnostic information written to the SYS1.LOGREC and GTF data sets.
- Appendix A, "Periods of an Extent" on page 93, explains the concept of a "period" within a DASD extent.
- Appendix B, "Calculations Done in Module IGG019KE to Get a Relative Track Address" on page 95, goes through the calculations performed in module IGG019KE to obtain relative track addresses.
- Appendix C, "Calculations Done in Module IGG019KF to Get Relative Track Address" on page 97, goes through the calculations performed in module IGG019KF to obtain relative track addresses.
- Appendix D, "Channel Programs" on page 99, describes the channel programs for BDAM requests.
- "Glossary of Terms and Abbreviations" on page 132 defines the terms used in this book.

An index is also included.

## PREREQUISITE KNOWLEDGE

To use this book efficiently, you must have a general understanding of data management.

**REQUIRED PUBLICATIONS**

You should be familiar with the background information on data management presented in:  
MVS/370 Data Administration Guide, GC26-4058

**RELATED PUBLICATIONS**

Within the text, references are made to the various publications listed in the table below.

Short Title	Publication Title	Order Number
DADSM Diagnosis Reference	<u>MVS/370 DADSM Diagnosis Reference</u>	LY26-3919
Data Administration: Macro Instruction Reference	<u>MVS/370 Data Administration: Macro Instruction Reference</u>	GC26-4057
Data Administration Guide	<u>MVS/370 Data Administration Guide</u>	GC26-4058
Open/Close/EOV Logic	<u>MVS/370 Open/Close/EOV Logic</u>	LY26-3924
SAM Logic	<u>MVS/370 SAM Logic</u>	LY26-3925
Service Aids	<u>OS/VS2 MVS System Programming Library: Service Aids</u>	GC28-0674
SYS1.LOGREC Error Recording	<u>OS/VS2 System Programming Library: SYS1.LOGREC Error Recording for MVS</u>	GC28-0677
System Codes	<u>OS/VS Message Library: VS2 System Codes</u>	GC38-1008
System Messages	<u>MVS/370 Message Library: System Messages, Volumes 1 and 2</u>	GC28-1374 and GC28-1375

SUMMARY OF AMENDMENTS

RELEASE 1.1 LIBRARY UPDATE, DECEMBER 1985

**SERVICE CHANGES**

Rather than reflecting system capability, the structure of this publication reflects the customer's approach to a specific task.

All other MVS/370 titles referred to in this publication have been changed to their corresponding MVS/XA titles. Order numbers of the MVS/370 books remain the same.

Information has been added to reflect service changes.

Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM

**CONTENTS**

Introduction	1
Creating a BDAM Data Set	1
BDAM Macros, Records, and Options	1
Macros	1
Records	1
Options	3
Terminology Associated with BDAM Requests	4
Relationship of BDAM to the Rest of the Operating System	4
Overview of BDAM Processing	4
Opening a Data Set	8
Processing an I/O Request	8
Controlling the Processing	8
Converting Addresses	9
Building Channel Programs	9
Allocating Buffers Dynamically	9
Switching Extents	10
Scheduling Further Processing	10
Getting and Releasing Exclusive Control	10
Checking for Request Completion	11
Closing a Data Set	11
Cleaning Up Resources after Abnormal Termination	11
Method of Operation	12
Diagram 1. Contents of Method of Operation Diagrams	13
Diagram 2. Open Data Set	14
Diagram 3. Read or Write a Record—Overview	17
Diagram 4. Prepare to Execute Channel Program	18
Diagram 5. Convert Address	20
Diagram 6. Build Channel Program	22
Diagram 7. Assign Buffer: Handle I/O Interrupts (Appendage Processing)	24
Diagram 8. Complete Processing Format F, U, or V Request or Schedule Further Processing	
Diagram 9. Determine Whether Format U or V Record Will Fit on Track	32
Diagram 10. Complete Processing Format VS Request or Schedule Further Processing	
Diagram 11. Determine Whether Format VS Record Will Fit on Track	
Diagram 12. Get or Release Exclusive Control	
Diagram 13. Free Buffer	48
Diagram 14. Check if I/O Operation Completed	50
Diagram 15. Close Data Set	52
Diagram 16. Free System Resources after Abnormal Termination	54
Module Directory	58
Data Areas	60
BCB (Buffer Control Block)	60
BCB for Spanned Records—No Dynamic Buffering (DSECT Name: BCBDEFS)	60
BCB for Dynamic Buffering—Real Storage Specified (DSECT Name: BCBDEFR)	61
BCB for Dynamic Buffering—Virtual Storage Specified (DSECT Name: BCBDEFV)	62
Buffer Pools	62
Buffer for Dynamic Buffering of Nonspanned Records	63
Buffer for Dynamic Buffering of Spanned Records	63
Buffer for Simple Buffering of Spanned Records	64
DCB (Data Control Block)	65
DEB (Data Extent Block)	70
DECBC (Data Event Control Block)	72
Exclusive Control List (DSECT Name: RDXLIST)	74
IOB (Input/Output Block)	75
Unposted Queue	84
Unscheduled List (DSECT Name: USL)	84
Format of the Unscheduled List	85



Format of an IOB Address Field in the Unscheduled List	86
Diagnostic Aids	87
Messages and Codes Issued by BDAM Modules	87
Problem Determination	87
Error Information Recorded in the SDWA on SYS1.LOGREC	87
Variable Recording Area Written by IGCT005C and IGCT105C (Exclusive Control Recovery Routine)	88
Variable Recording Area Written by IGCT005G (FREEDBUF Recovery Routine)	90
Error Information Recorded in the GTF Data Set	91
Appendix A. Periods of an Extent	93
Appendix B. Calculations Done in Module IGG019KE to Get a Relative Track Address	95
Appendix C. Calculations Done in Module IGG019KF to Get Relative Track Address	97
Appendix D. Channel Programs	99
Glossary of Terms and Abbreviations	132
Index	135

FIGURES

1.	BDAM Record Formats . . . . .	2
2.	Relationship Among Processing Program, BDAM Routines, and Other Parts of the Operating System . . . . .	5
3.	Graphic Symbols Used in Method of Operation Diagrams . . . . .	12
4.	Channel Program Generating Modules . . . . .	23
5.	Message/Module Cross-Reference . . . . .	87
6.	Control Blocks Written in the GTF Data Set . . . . .	91
7.	Format of the GTRACE Work Area . . . . .	92
8.	Track Overflow . . . . .	93
9.	DEB Information Needed to Calculate Relative Track Addresses in IGG019KE . . . . .	95
10.	DEB Information Needed to Calculate Relative Track Addresses in IGG019KF . . . . .	98

**Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM**

## INTRODUCTION

The basic direct access method (BDAM) is a group of routines that retrieves data from and stores data into data sets that are directly organized and reside on direct-access devices.

BDAM is a part of the MVS/370 Data Facility Product (DFP) program product. BDAM routines are grouped into modules that are placed in the link pack area library (SYS1.LPALIB) at system generation. When the system is initialized, these modules are loaded into the pageable link pack area.

## CREATING A BDAM DATA SET

A BDAM data set can be created on any supported direct-access volume. It is created using BSAM routines, which are documented in Sam Logic.

## BDAM MACROS, RECORDS, AND OPTIONS

Which BDAM modules are executed in a given operation depends on what macro was issued by the processing program, the type of record being processed, and the options that are in effect. Information on these topics is provided in the sections that follow.

### MACROS

The following macros are used to access and manipulate BDAM data sets: OPEN, READ, WRITE, FREEDBUF, RELEX, CHECK, and CLOSE. These macros, the only ones issued by a program processing a BDAM data set that will cause BDAM code to be entered, are documented in Data Management Macro Instruction Reference.

### RECORDS

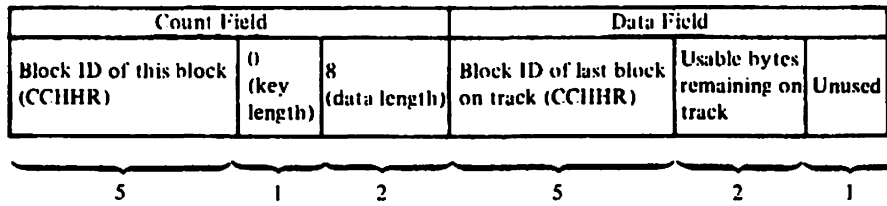
A track in a BDAM data set contains a capacity record, data records, and, in some cases, dummy records.

- Capacity record (R0). This is always the first record on each track. It contains the ID of the last block on the track, and the number of usable bytes remaining on the track.
- Data records. BDAM data records can be format F, U, V, or VS and can contain count, key, and data fields or just count and data fields.
- Dummy records. The user can request that BSAM put dummy records on the track when data sets containing format F records are created. The purpose of dummy records is to set aside unused space on the track so new records can be added after the data set is created. Dummy records contain X'FF' in the first byte of the key field, and the position of the dummy record on the track (the R in MBBCCHHR) in the first byte of the data field.

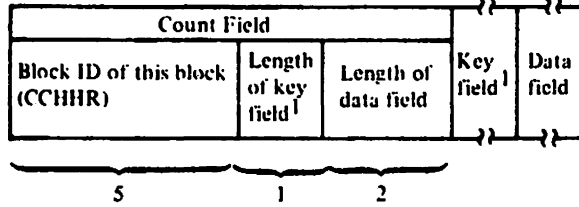
Figure 1 on page 2 shows the format of the different types of records in a BDAM data set. Further information about BDAM records is contained in Data Administration Guide.



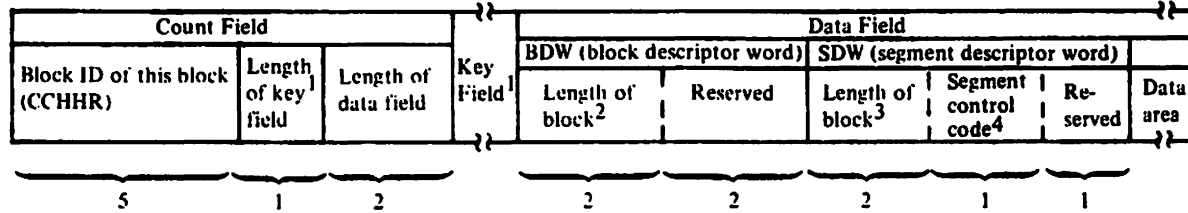
**Track Capacity Record (R0)**



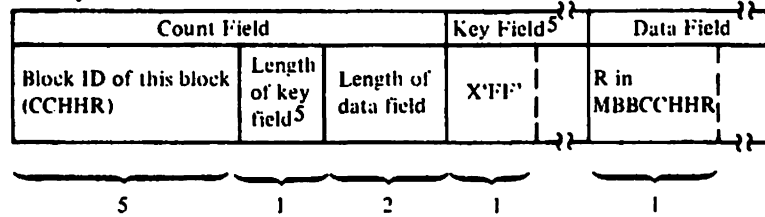
**Data Record (Format F, V, or U)**



**Data Record (Format VS)**



**Dummy Record**



- <sup>1</sup> If keys are not being used, the sixth byte in the count field (key length field) is zeros and the key field does not exist.
- <sup>2</sup> The length in bytes 1 and 2 of the BDW is the length from the beginning of the BDW to the end of the data field.
- <sup>3</sup> The length in bytes 1 and 2 of the SDW is the length from the beginning of the SDW to the end of the data field.
- <sup>4</sup> 00 = format VS record that fits on a single track  
01 = first segment of a format VS record  
02 = last segment of a format VS record  
05 = intermediate segment of a format VS record
- <sup>5</sup> Keys are required if dummy records are written.

Figure 1. BDAM Record Formats

## OPTIONS

The following options are available to the user of BDAM:

- **Dynamic buffering**—requests that the system handle acquisition, assignment, and release of buffers.
- **Exclusive control**—when specified by the user, requests that the system prevent the data block about to be read from being modified by other requests. Exclusive control of a block is obtained using a READ macro and released using a WRITE or RELEX macro.

Exclusive control can also be acquired automatically by the system in certain cases (see the Glossary entry for exclusive control for further details).

- **Extended search**—requests that the system search for the specified block or a place in which to add a new block, starting with the first block on the track containing the block address specified in the READ or WRITE macro, and continuing either for as many tracks or blocks (rounded up to a complete track) as are specified in the request macro or until the search ends successfully.

Extended search is only applicable if relative addressing is being used (relative block addressing is only applicable for format F records).

- **Feedback**—there are two types of feedback:
  - **Block position feedback** requests that the system put the address of the block just read or written into the area specified in the block address operand of the READ or WRITE macro. The format of the address will be MBBCCHHR if feedback was not specified in the DCB macro; otherwise, the format will be the same as the addressing scheme in the DCB macro.
  - **Next-address feedback** requests that the system put the relative track address of the next data or capacity record (RO) into the area specified in the next address operand of the READ macro. (If the type operand of the READ macro terminated with an R, the address of the next data record is returned; if it terminated with an RU, the address of the next data or capacity record, whichever occurs first, is returned.)

Next-address feedback is only applicable for operations involving format VS records.

- **Relative addressing**—occurs when the processing program specifies the position of a block in a data set relative to the first block of a data set. The relative address can be either a relative track or relative block number. (See the Glossary entry for more information about these two types of relative addressing.)
- **Write-validity check**—requests that the system verify the accuracy of any information written by the channel program.
- **Track overflow**—requests that the system allow a format F record, whose space requirements exceed the space remaining on the track, to be partially written on that track and completed on the following track(s).

## TERMINOLOGY ASSOCIATED WITH BDAM REQUESTS

The following terms are used throughout this book to describe types of requests:

- **READ-exclusive request**—is a READ request specifying that exclusive control should be acquired for the record about to be read.
- **WRITE-add request**—is a request to write a new block to the data set.
- **WRITE-update**—is a request to write an already existing block to the data set.
- **WRITE-release request**—is a WRITE-update request that specifies exclusive control should be released for the record about to be written. (It releases the exclusive control that was acquired for the corresponding READ request.)

## RELATIONSHIP OF BDAM TO THE REST OF THE OPERATING SYSTEM

When a data set to be processed using BDAM is opened, the Open routine of O/C/EOV gets control, makes sure that the volume(s) containing the data set is mounted, and then calls the BDAM module that will continue opening the data set. When BDAM open processing is completed, this BDAM module returns control to the Open routine.

During BDAM processing, conversion of a block or track address may be required if relative addressing is specified in the processing program. Parts of this conversion are done by the system address conversion routine IECPCNVT or IECPLTV.

If abnormal termination occurs during processing related to the FREEDBUF or RELEX SVCs, the recovery/termination manager (R/TM) gets control to initiate error analysis and release of resources, and to provide diagnostic information.

When an input/output operation terminates, the EXCP processor gives control to a BDAM appendage, which will schedule the remaining processing required for the I/O request.

When a data set is closed, the Close routine of O/C/EOV gets control and calls the BDAM module that continues close processing by freeing resources associated with the data set. When BDAM close processing is completed, this BDAM module returns control to the Close routine of O/C/EOV.

## OVERVIEW OF BDAM PROCESSING

For purposes of discussion, BDAM can be divided into the following major operations:

- Opening a data set
- Processing an I/O request
- Closing a data set
- Cleaning up resources after abnormal termination

The following discussion summarizes these operations. Figure 2 on page 5 shows at what point in BDAM processing each operation is performed and which modules are involved in the operation.

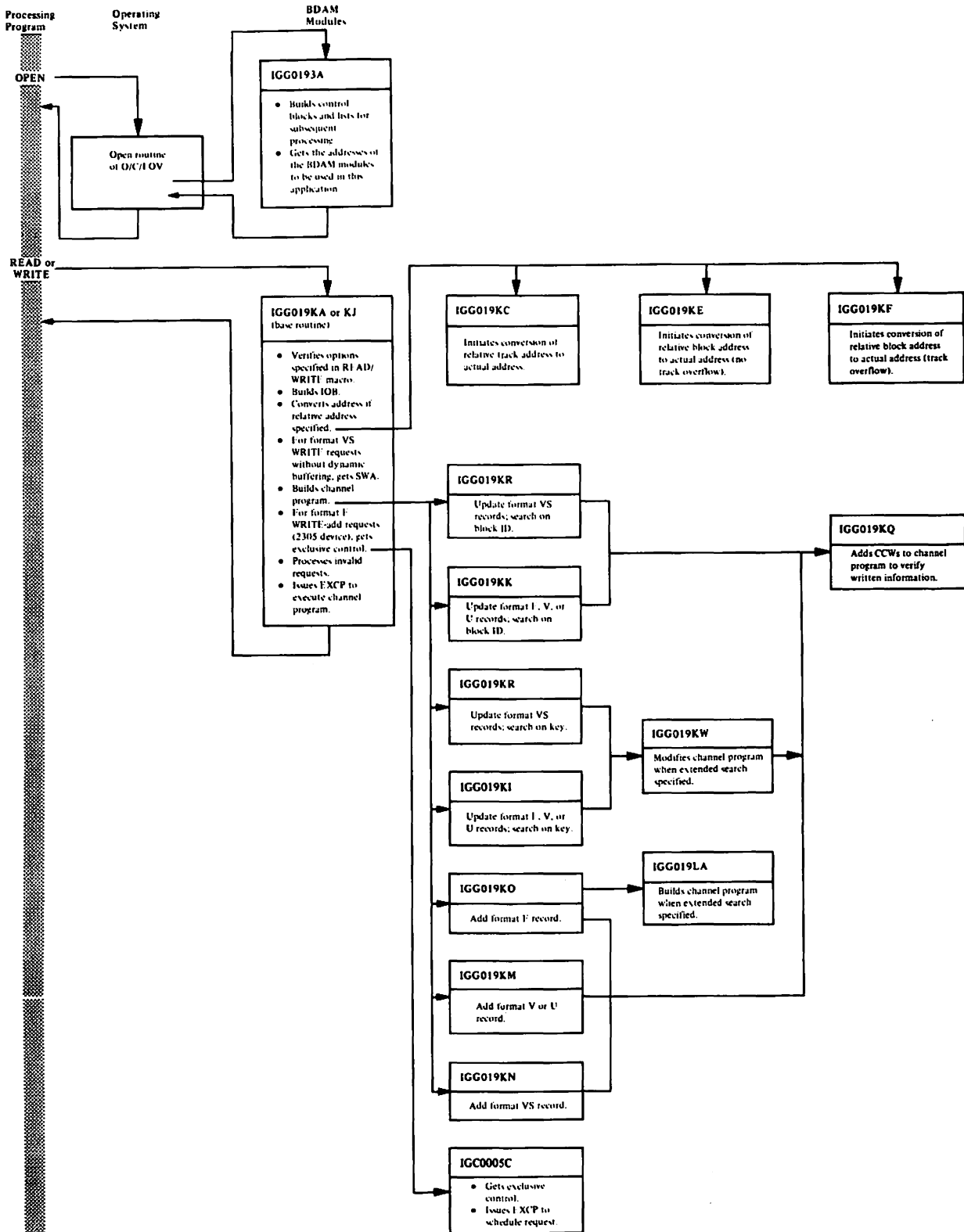


Figure 2 (Part 1 of 3). Relationship Among Processing Program, BDAM Routines, and Other Parts of the Operating System



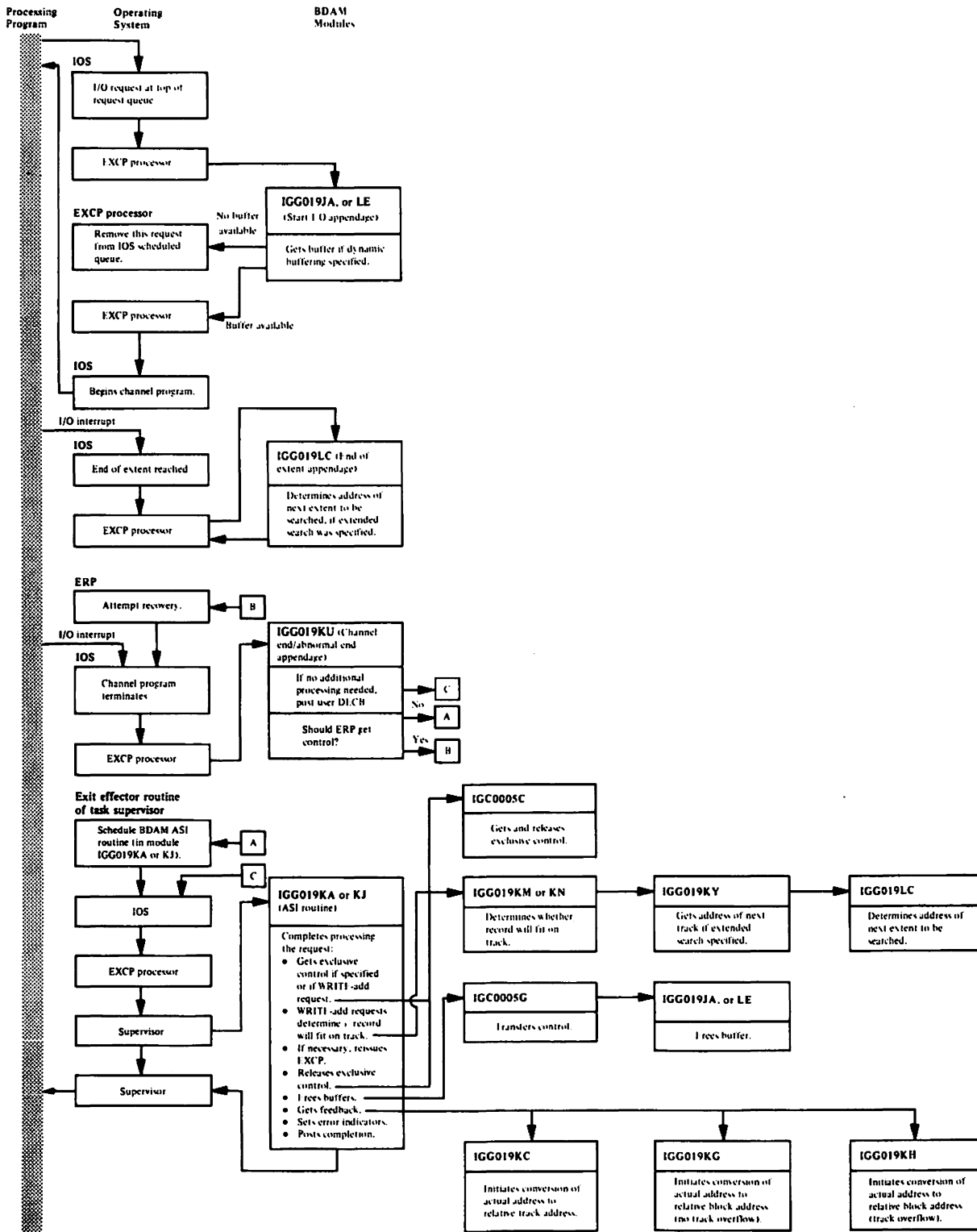


Figure 2 (Part 2 of 3). Relationship Among Processing Program, BDAM Routines, and Other Parts of the Operating System

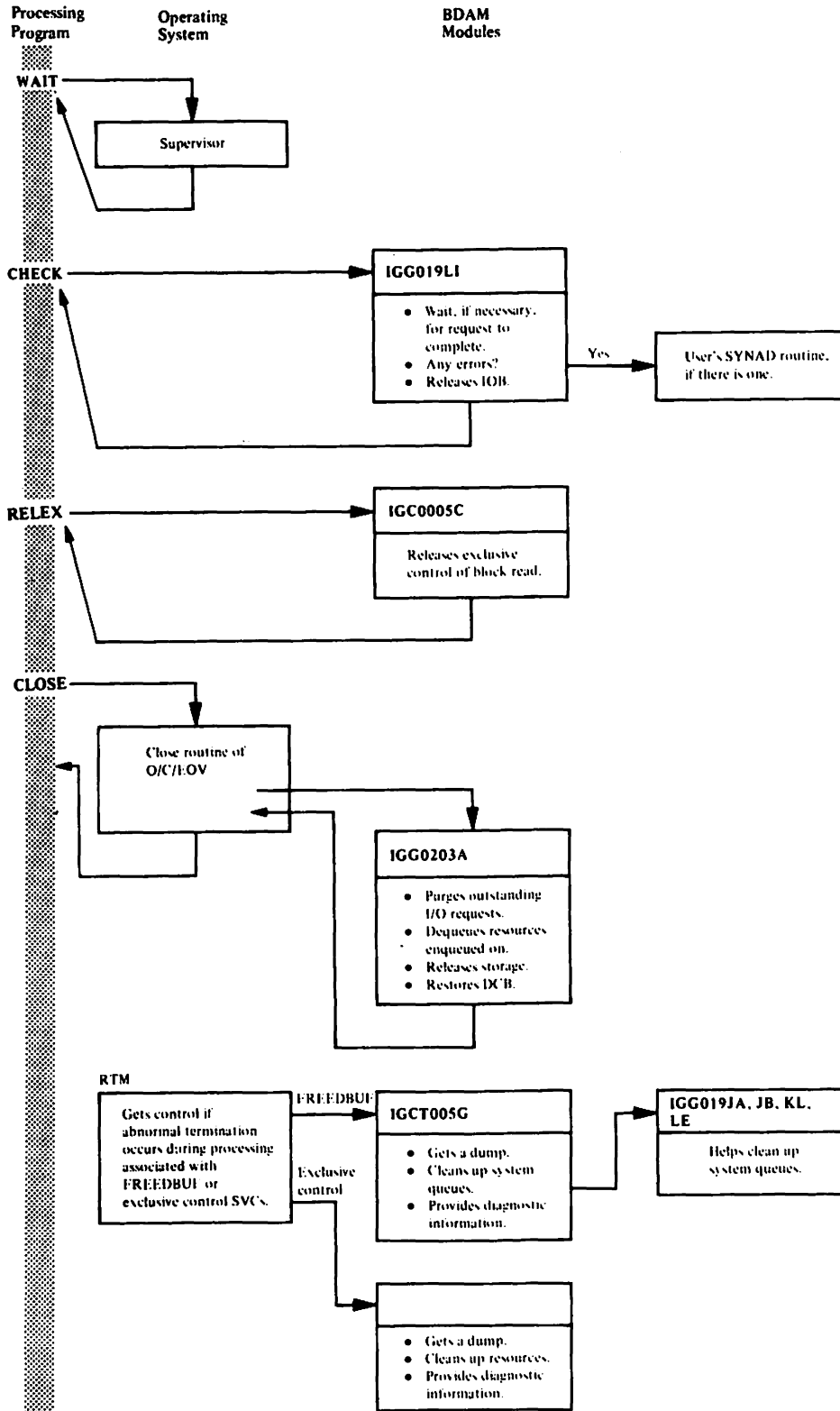


Figure 2 (Part 3 of 3). Relationship Among Processing Program, BDAM Routines, and Other Parts of the Operating System

## OPENING A DATA SET

When a processing program issues the OPEN macro to open a BDAM data set, the Open routine of O/C/EOV gets control. After partially opening the specified data set (which consists of performing the operations that are not dependent on data set organization), the Open routine gives control to a BDAM module called the open executor.

The open executor module continues the open process by:

- Determining which BDAM modules will be used in this application and getting their addresses, and by
- Building control blocks and lists for subsequent use by BDAM.

(See "Diagram 2. Open Data Set" on page 14 for the details of these operations.)

After performing these operations, the open executor module returns control to the Open routine of O/C/EOV, which finishes opening the data set.

## PROCESSING AN I/O REQUEST

The following operations are performed when an I/O request is processed:

- Controlling the processing
- Converting addresses
- Building channel programs
- Allocating buffers dynamically
- Switching extents
- Scheduling further processing
- Getting and releasing exclusive control
- Checking for request completion

### Controlling the Processing

When a processing program issues a READ or WRITE macro, a module called the foundation module gets control. The foundation module is composed of two routines: a base routine and an asynchronous interrupt (ASI) routine. The base routine completes the preparations necessary before an I/O request can be processed. The ASI routine completes processing of a request after an I/O operation has terminated. Both routines route control to other BDAM modules to complete their tasks. For the details of these operations, see:

- "Diagram 4. Prepare to Execute Channel Program" on page 18
- "Diagram 8 (Part 1 of 2). Complete Processing Format F, U, or V Request or Schedule Further Processing" on page 28
- "Diagram 10 (Part 1 of 2). Complete Processing Format VS Request or Schedule Further Processing" on page 34

## **Converting Addresses**

The modules involved in converting addresses are called the address conversion modules. BDAM converts block addresses whenever relative addressing is specified in the processing program.

The address of a block may be converted:

- From a relative address to an actual address (necessary because the channel program, when it searches for a block, does so using the actual address of that block), or
- From an actual address to a relative address (done when the processing program requests feedback).

See "Diagram 5. Convert Address" on page 20 for the details of address conversion.

## **Building Channel Programs**

The modules that build channel programs are called channel program generating modules. A channel program is built dynamically for every valid BDAM READ or WRITE request.

The channel program generating modules build a basic channel program. If extended search was specified in the processing program, these modules add CCWs to the basic channel program for extended search. If the write-verify option was specified in the processing program, one of the channel program generating modules generates CCWs to verify data that has just been written.

See "Diagram 6. Build Channel Program" on page 22 for the details of these operations.

## **Allocating Buffers Dynamically**

If dynamic buffering has been specified in the processing program, modules called dynamic buffering modules get control. The major operations performed by these modules are assignment and release of buffers.

**ASSIGNING A BUFFER:** The dynamic buffering modules (which are also the start I/O appendages) get control from the EXCP processor when a request to read data is ready to be executed. The dynamic buffering modules obtain a buffer, assign it to the request, and then return to the EXCP processor, which starts the channel program.

If a buffer is not available for assignment, the IOB representing the request is put on a queue called the unscheduled list (virtual storage specified for the task) if there is room on it, or the IOB buffer queue (real storage specified for the task). The request waits on the queue until a buffer can be assigned to it. In the case of the unscheduled list, if there's no room to put the IOB on the list, the list will be extended when the ASI routine in the foundation module subsequently gets control and issues a FREEDBUF macro.

See "Diagram 7. Assign Buffer: Handle I/O Interrupts (Appendage Processing)" on page 24 for the details of buffer assignment.

**FREEING A BUFFER:** When a FREEDBUF macro is issued by either the processing program or the ASI routine in the foundation module, one of the dynamic buffering modules again gets control.

If the processing program issues the FREEDBUF macro or a WRITE macro specifying 'S' for the area address, the dynamic buffering module gets control (from the processing program or the foundation module respectively) to free the buffer assigned to the request. When the buffer is freed, any request waiting on



the unscheduled list or IOB buffer queue will be assigned the buffer and then its channel program will be scheduled for execution. If no requests are waiting, the freed buffer is added to the queue of available buffers.

See "Diagram 13. Free Buffer" on page 48 for the details of these operations.

### Switching Extents

When the extended search option is specified in the processing program and the channel program must switch from one extent to another while searching for a block, the EXCP processor gives control to a module called the end-of-extent appendage module.

This module determines the address of the next extent (if one is available) to be searched, and whether the search limit is reached in that extent.

See "Diagram 7. Assign Buffer: Handle I/O Interrupts (Appendage Processing)" on page 24 for the details of crossing extents.

### Scheduling Further Processing

When a channel program ends, the EXCP processor gives control to a module called the channel end/abnormal end appendage module. This module analyzes the status of the I/O request and, based on that analysis, posts the request complete or schedules further processing for the request. Further processing usually involves either restarting the channel program or scheduling the ASI routine in the foundation module for execution.

See "Diagram 7. Assign Buffer: Handle I/O Interrupts (Appendage Processing)" on page 24 for the details of this processing.

### Getting and Releasing Exclusive Control

If the exclusive control option is specified in the processing program, or a request to add a new record to a data set is being processed, a module called the exclusive control module is used in processing an I/O request. This module prevents concurrent access of a block, a track, or, in certain cases, an entire data set. It ensures that, if more than one request is made to use one of these resources, only the first request can use it while exclusive control is in effect.

The exclusive control module maintains an exclusive control list, which contains the addresses of all resources currently being used that require exclusive control. Associated with each entry in the exclusive control list is a nonposted queue. This queue contains the addresses of any IOBs waiting to get exclusive control of the resource in use (as represented by the entry in the exclusive control list).

When the processing program issues (1) a WRITE macro that specifies exclusive control, (2) a RELEX macro, or (3) a request to add a new record to a data set is completed, the exclusive control module again gets control. The exclusive control module gives exclusive control to any I/O requests waiting to use the resource under exclusive control; that is, to any requests represented by an IOB on the unposted queue. If no requests are waiting to get exclusive control, it removes the address of the resource from both the exclusive control list and the system queue.

See "Diagram 12 (Part 1 of 2). Get or Release Exclusive Control" on page 42 for the details of getting and releasing exclusive control.

### **Checking for Request Completion**

A module called the check module is used in processing an I/O request if the processing program uses the CHECK macro instruction to ensure that a given I/O request has been satisfied. This module determines if, and waits if necessary until, a request is posted as complete. If errors have been detected during the I/O operation, the check module gives control to a user's SYNAD routine. If no SYNAD routine exists, the check module gives control to the EOVS routine (via SVC 55), which abnormally terminates the task.

See "Diagram 14. Check if I/O Operation Completed" on page 50 for the details of checking for request completion.

### **CLOSING A DATA SET**

When the processing program issues a CLOSE macro, the Close routine of O/C/EOV gets control and, after partially closing the specified data set, gives control to a BDAM module called the close executor.

The close executor module continues the close process by:

- Releasing to the system all resources still under exclusive control, and all storage obtained by BDAM
- Restoring the fields of the DCB that have been changed by BDAM routines

See "Diagram 15. Close Data Set" on page 52 for the details of these operations.

### **CLEANING UP RESOURCES AFTER ABNORMAL TERMINATION**

If abnormal termination occurs during any of the processing associated with the FREEDBUF or exclusive control SVCs, the recovery/termination manager (R/TM) receives control. It, in turn, will give control to either the FREEDBUF or exclusive control recovery modules.

These recovery modules attempt to clean up system resources and provide diagnostic information about the problem that caused abnormal termination.

See "Diagram 16. Free System Resources after Abnormal Termination" on page 54 for the details of these operations.

METHOD OF OPERATION

This section consists of method of operation diagrams. These diagrams describe the major operations performed by BDAM, and the data that is input to and output from those operations.

Some operations described in each diagram are further amplified in the adjoining "Notes." The "Notes" tie the operations to a specific module and, where possible, to a label in that module. In the notes, the module name is in boldface type (for example, IGG0193A) and appears before the text describing the operations performed in that module. Labels are enclosed in brackets (for example, [IGSTAR]) and appear within or at the end of the text describing the operations performed at that label.

"Diagram 1. Contents of Method of Operation Diagrams" on page 13 is a table of contents for the method of operation diagrams. It shows, at the highest level, each operation performed by BDAM and directs you to the specific diagram documenting the operation. "Diagram 3. Read or Write a Record—Overview" on page 17 is an overview diagram of the operations involved in reading or writing a record. It directs you to other, more detailed, diagrams explaining these operations. Figure 3 explains the graphic symbols used in the diagrams.

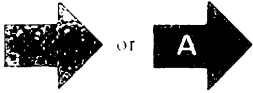
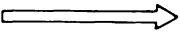




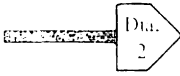
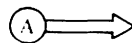
Control flow	Data flow
 <p>Entry point to diagram. The second type of arrow is used when a diagram has more than one entry point.</p>	 <p>Input to or output from a processing step</p>
 <p>Flow of control</p>	 <p>Modification of data</p>
 <p>On-page connector; number represents number of processing step receiving control</p>	 <p>Testing of, or reference to, data</p>
 <p>Off-page connector; number is number of diagram where processing resumes</p>	 <p>On-page connector; used to indicate input to or output from a processing step</p>

Figure 3. Graphic Symbols Used in Method of Operation Diagrams

DIAGRAM 1. CONTENTS OF METHOD OF OPERATION DIAGRAMS

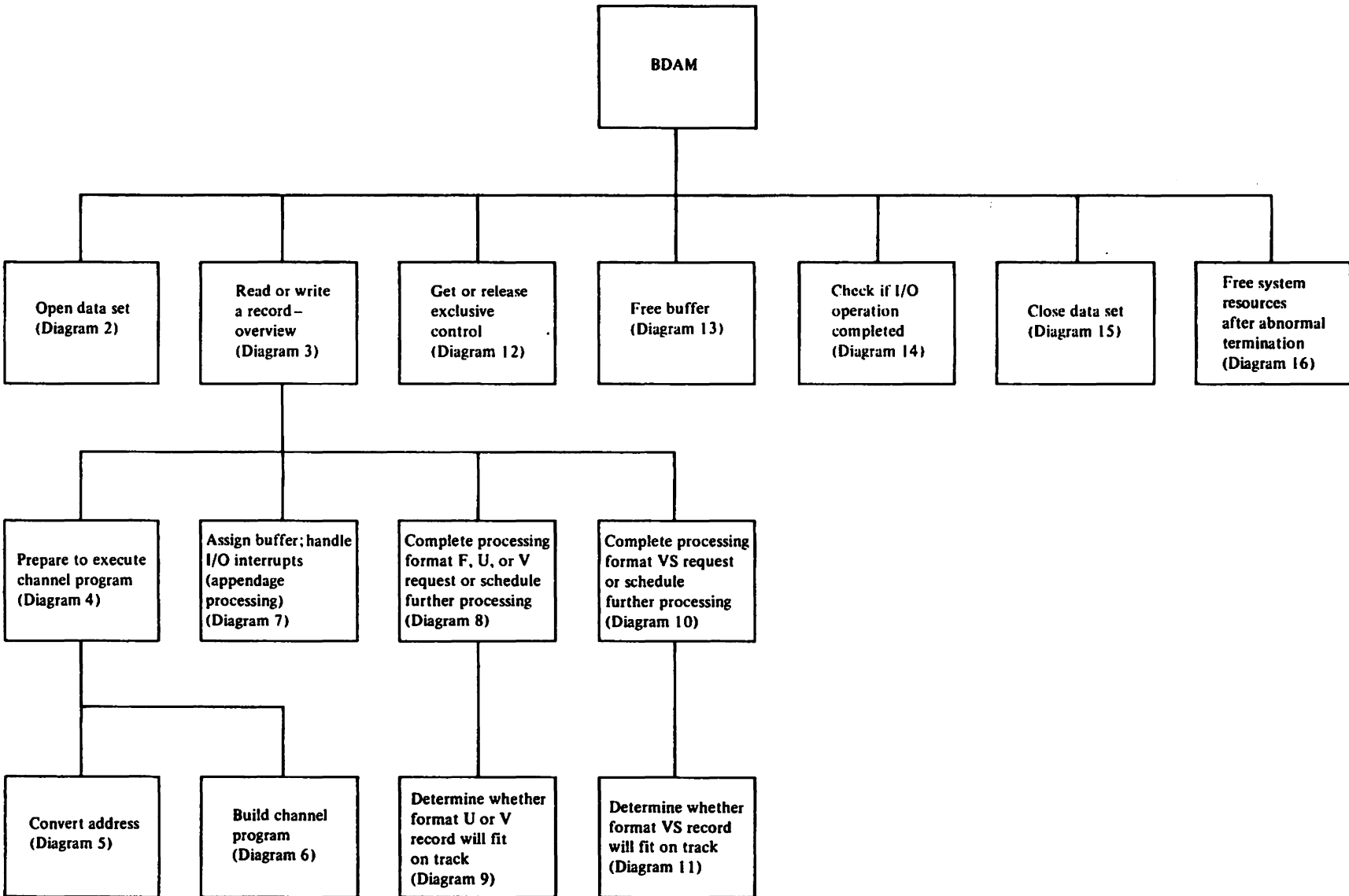
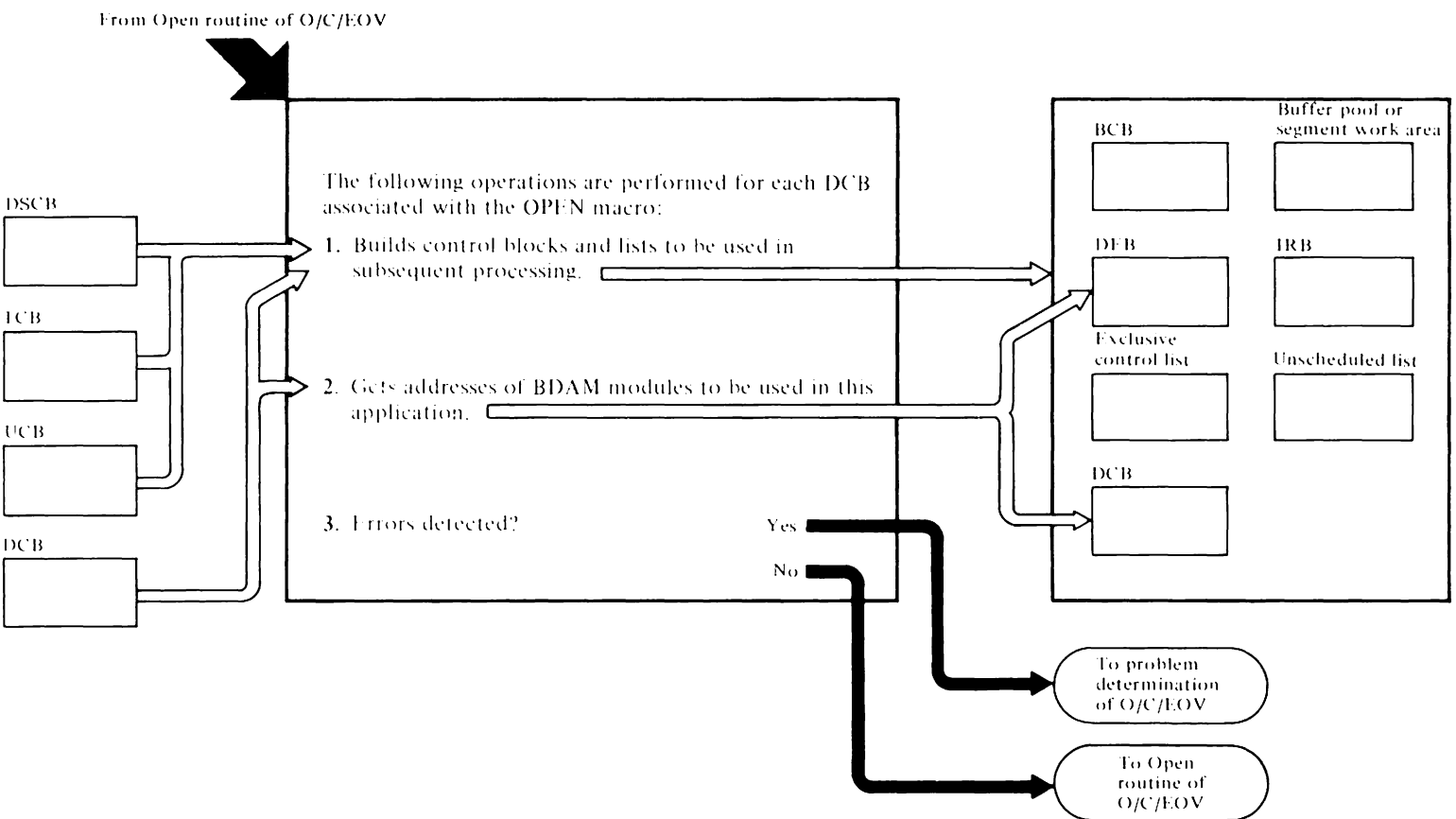




DIAGRAM 2. OPEN DATA SET



Notes for Diagram 2

IGG0193A

This module sets bits, called audit trail bits, in the O/C/EOV work area whenever it completes processing that changes the copy of the DCB. If abnormal termination occurs during open processing, a module called the force close executor (IGG020T1) frees resources obtained by the BDAM open executor, unless all BDAM open processing was completed (indicated by a bit set in the user's DCB) before abnormal termination occurred. If all processing was completed by the BDAM open executor, IGG020T1 gives control to the BDAM close executor to free resources. (See SAM Logic for further details about IGG020T1.)

- 1 (This module is entered from the Open routine of O/C/EOV to do the open processing dependent on data set organization.)

Determines the amount of storage needed to build the DEB (including the AVT, prefix, and actual and relative extents), and gets that storage from subpool 230 (key 5). It then builds the DEB and chains it to the TCB. (A user label track, if allocated, is not included in the DEB extents, since the user label track will already have been verified or updated during common open processing.) [FUNCT1A]

After all extents are built, tests the UCB of each extent to see if it resides on a Mass Storage System (MSS) virtual volume. For each extent that resides on a virtual volume, builds a parameter list and issues the ICBACREL macro to allocate and stage space on a DASD staging device. [OBDAMOUT]

Initializes some of the fields of the DCB using the DCB copy. [FUNCT3]

If exclusive control was specified, gets storage from subpool 230 (key 5) for an exclusive control list (subsequently used to contain the addresses of blocks under exclusive control and IOBs for requests waiting to get exclusive control). [READX]

Issues the CIRB macro, which gives control to a supervisor routine to build an IRB. [SETUPIRB]

If relative block addressing was specified for format F records, builds the relative extents in the DEB. (The relative extents are

subsequently used to calculate the position of a block within the data set.) One relative extent is built for each actual extent in the DEB. If track overflow is being used, builds modified relative extents in the DEB. It then builds the overflow section of the DEB, which contains information about the size of a period (Appendix A explains what a period is). [FUNCT4]

If dynamic buffering was specified for format F, V, or U records, builds buffers and a BCB. [FUNCT5]

If the record format is VS, either builds a BCB and a pool of segment work areas, or builds a BCB and buffers (if dynamic buffering was specified). [FUNCT6]

If virtual storage and dynamic buffering were specified for the task, builds an unscheduled list (subsequently used to store the addresses of IOBs for I/O requests that, because a buffer is not available, cannot be scheduled).

- 2 Finds the address of each BDAM module to be used in this application and puts it either in the DCB or in the AVT in the DEB. Puts the module identification of each BDAM module to be used in the DEB. [FUNCT2]

- 3 If any of the following errors was detected during processing, gives control to a problem determination module (via the DMABCOND macro) in the EOVS routine of O/C/EOV:

- MACRF field was not specified [RECFMOK]
- BUFL field was specified as 0 and dynamic buffering was requested [NOBUFNO]
- Primary extents were specified as 0 [NOOVFLOW]

Problem determination issues a WTP message identifying the data set in error and the cause of the error.

If processing was error free, issues the IECRES macro to update the user's DCB from the copy of the DCB used during open processing. When control is returned, scans the WTG table to determine whether any other DCBs associated with the OPEN macro require the use of this module. If so, the module is reentered and does the required processing. If not, returns control to the Open routine or to the next open executor (for another access method) indicated in the WTG table. [FUNCT7]

This page intentionally left blank.

**DIAGRAM 3. READ OR WRITE A RECORD—OVERVIEW**

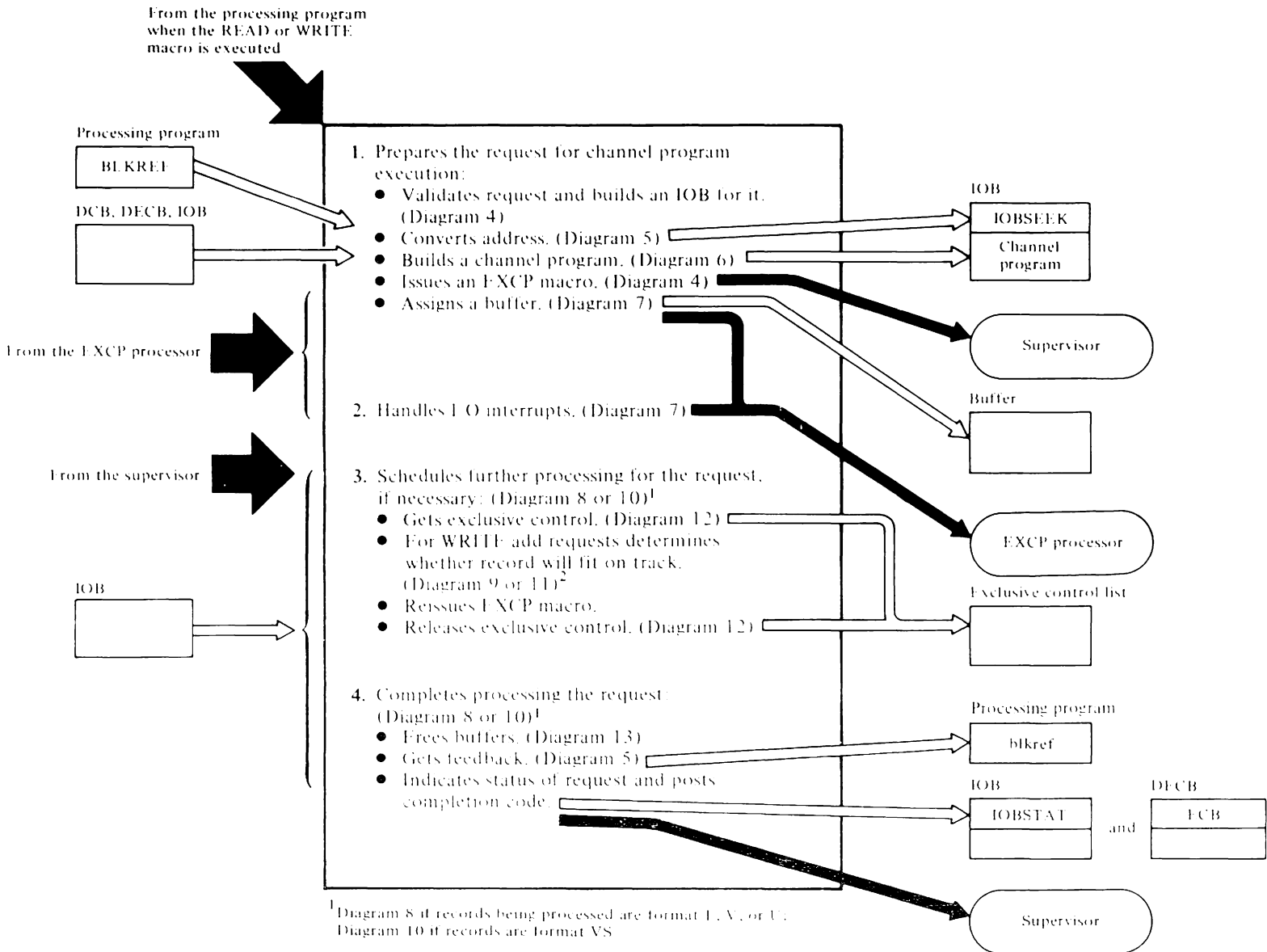
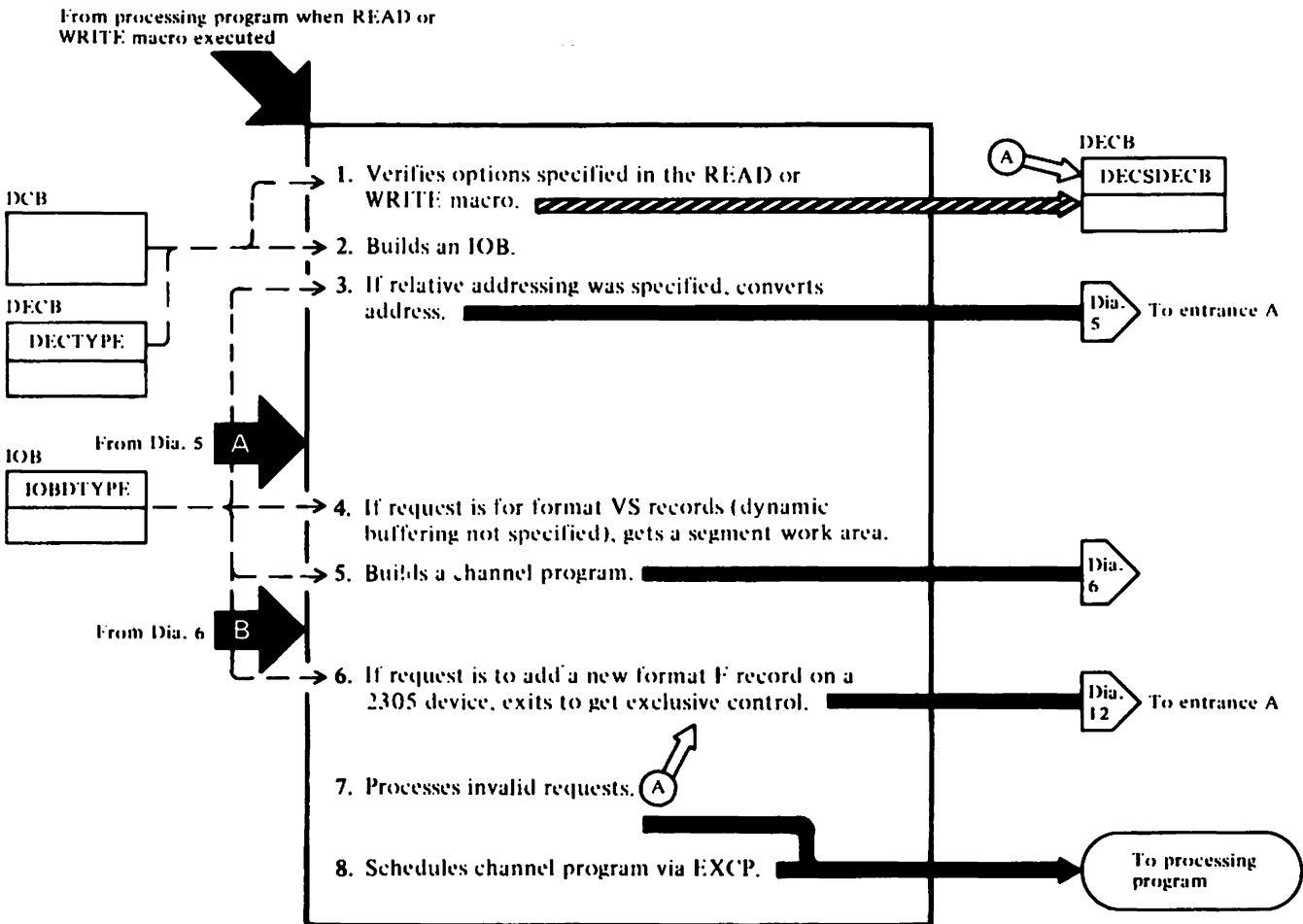


DIAGRAM 4. PREPARE TO EXECUTE CHANNEL PROGRAM



**Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM**

**Notes for Diagram 4**

**IGG019KA** (if format F, U, or V records),  
**IGG019KJ** (if format VS records)

- 1 (This module is entered when a READ or WRITE macro associated with a BDAM record is executed.)  
  
Verifies the options the user specified in the READ or WRITE macro. Combines options specified in the type field of the READ or WRITE macro with options specified in the OPTCD parameter of the DCB macro. The result of this combination of options is put in the DECB and subsequently transferred to the IOB. [BEGIN]
- 2 Determines, by checking the DCB and DECB, the amount of storage needed to build the IOB. (The size of the channel program, which is a part of the IOB, must be calculated, because its size varies.) [SZRTN]  
  
Gets an available IOB from the pool of IOBs or, if the pool doesn't contain one that's large enough, issues a GETMAIN macro to get the storage from subpool 0. [CHKPOOL]  
  
Builds the IOB. [CLRIOB]
- 3 If relative addressing is specified in the DCB, branches to one of the address conversion modules to convert the relative address to an actual address. [ADDRDEC]

**IGG019KJ only**

- 4 After getting a segment work area (SWA) from the pool of SWAs, assigns it to the IOB for the request. If the request was a READ request and the READ macro does not specify

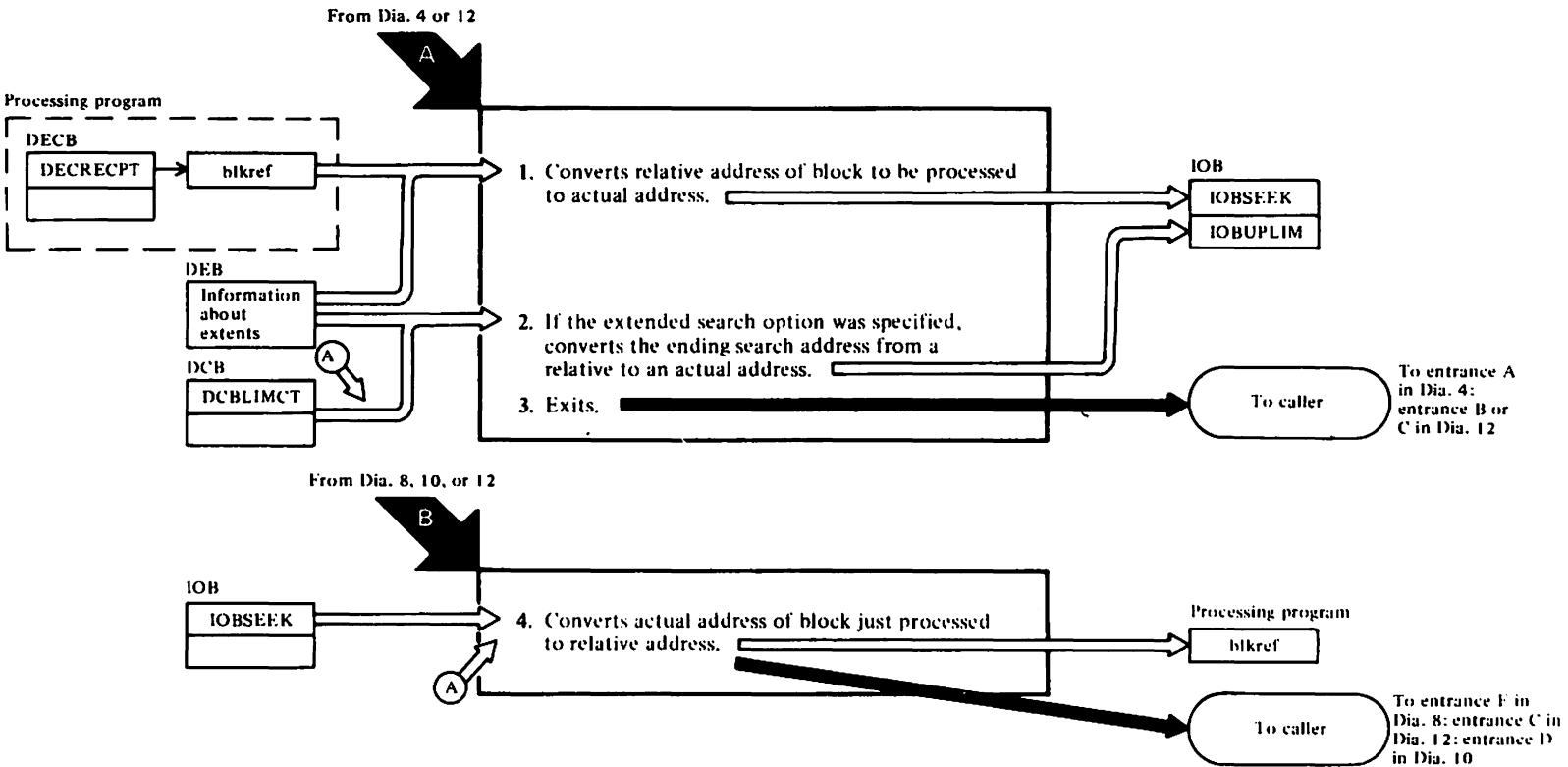
dynamic buffering but the DCB does, gets storage for a SWA. [GETSWA]

**IGG019KA,KJ**

- 5 Determines which module should get control to build the channel program and then branches to it. (The Notes in "Diagram 6. Build Channel Program" on page 22 show which module will get control.) [TYPEDEC]

**IGG019KA,KJ**

- 6 If the request is to add a new format F record to a data set on an IBM 2305 Fixed Head Storage device, issues SVC 53 to get exclusive control of either the track requested or, if extended search is specified, the entire data set. (Exclusive control is necessary to prevent multiple WRITE-add requests from updating the same dummy record.) The request is subsequently scheduled (via the EXCP macro) by IGC0005C, which ultimately gets control when SVC 53 is issued. [CHK2305]
- 7 Invalid requests could have been detected in the processing done in this routine, or in modules that received control from this routine. [LATERR]  
  
If the availability bit in the IOB is off (which means the IOB for this request was gotten via a GETMAIN macro rather than from the IOB pool), frees the IOB to the IOB pool and then posts a completion code in the DECB.
- 8 Issues an EXCP macro to schedule the channel program. On return from EXCP processing, returns control to the processing program. [EXCP]



### Notes for Diagram 5

One of the following address conversion modules gets control when relative addressing is specified in the processing program:

- IGG019KC (relative track addressing used)
- IGG019KE (relative block addressing used; no track overflow)
- IGG019KF (relative block addressing used; track overflow)

Relative block addressing can only be specified for format F records.

#### IGG019KE, KF

- 1 If relative block addressing is being used, converts the relative block address to TTR. (See Appendix B (IGG019KE) or C (IGG019KF) for the details of this conversion.)  
[RRSTART]

#### IGG019KC, KE, KF

Branches to the system convert routine IECPCNVT to convert the TTR for the specified record to MBBCCHHR. IECPCNVT puts the MBBCCHHR in the IOBSEEK field and returns control. [D3LINK in IGG019KC; CONVERT in IGG019KE, KF]

#### IGG019KE, KF

- 2 If the extended search option was specified and relative block addressing is being used, converts the relative block address for the ending search address to TTR.  
[RREXTS1]

#### IGG019KC, KE, KF

Branches to IECPCNVT to convert the TTR for the ending search address to

MBBCCHHR. IECPCNVT puts the MBBCCHHR in the IOBUPLIM field and returns control. [D3LOOP in IGG019KC; CONVERT in IGG019KE, KF]

- 4 One of the following feedback address conversion modules gets control when feedback and relative addressing are specified in the processing program:

- IGG019KC (relative track addressing used)
- IGG019KG (relative block addressing used; no track overflow)
- IGG019KH (relative block addressing used; track overflow)

#### IGG019KC, KG, KH

Branches to the system convert routine IECPLTV to convert the MBBCCHHR of the block just processed to TTR. [D3FDBF in IGG019KC; FDBSTAR in IGG019KG, KH]

#### IGG019KG, KH

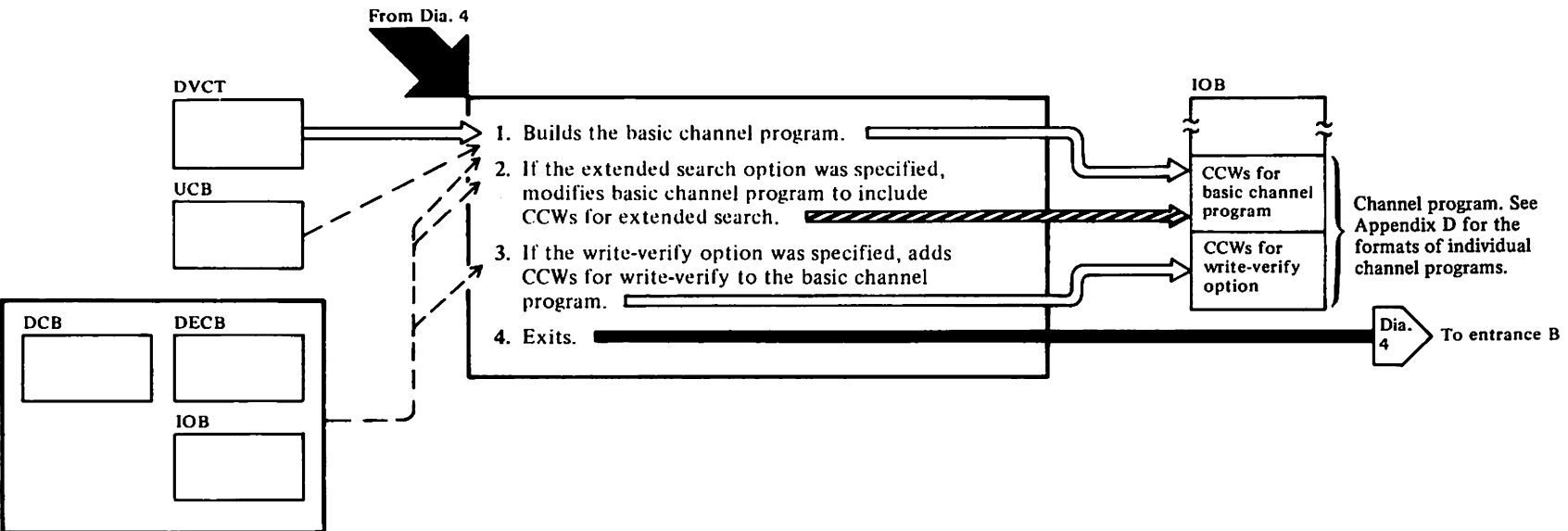
If relative block addressing is specified in the processing program, converts the TTR of the block just processed to a relative block address. The method by which these modules convert a TTR to a relative block address is basically a reversal of the technique used to convert a relative block address to a TTR (the latter conversion is shown in Appendixes B and C).  
[RETURN]

#### IGG019KC, KG, KH

Puts the relative block or relative track address in the user's blkref field. [CNVRET in IGG019KC; NOMORE in IGG019KG, KH]



**DIAGRAM 6. BUILD CHANNEL PROGRAM**



Notes for Diagram 6

Appendix D contains information on the channel programs built by BDAM modules. Figure 4 is cross-referenced to those channel programs under the column Channel Program Number.

- 1 A channel program is built for every BDAM READ or WRITE request. Which module builds the basic channel program depends on the type of request; Figure 4 contains this information.
- 2 See Figure 4 for the modules that handle extended search. If an extended search module is entered at this point in the processing, it will modify the basic channel program so that, when the channel program is executed, it will search additional tracks or blocks beyond

those specified in the READ or WRITE macro.

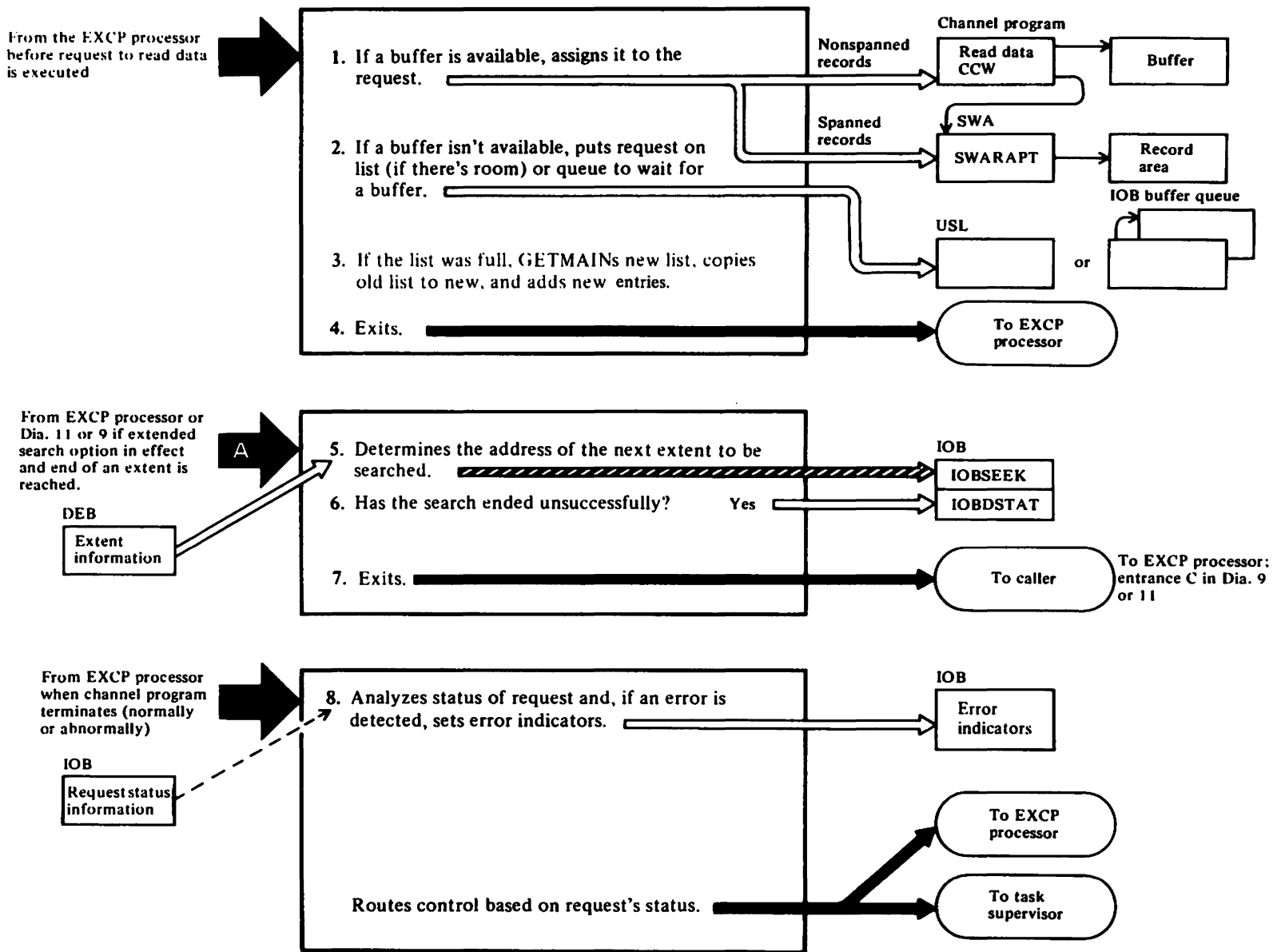
Extended search is implemented in a different way and at a later point in the processing for requests to add a new format V, U, or VS record (see "Diagram 10 (Part 1 of 2). Complete Processing Format VS Request or Schedule Further Processing" on page 34 and "Diagram 11 (Part 1 of 2). Determine Whether Format VS Record Will Fit on Track" on page 38. Extended search is not applicable for requests that involve a search by block identification.

- 3 If the write-verify option was specified, adds CCWs to the basic channel program so that, when it is executed, the channel program will verify the accuracy of the information it writes. There is only one write-verify module: IGG019KQ.

Type of Request	Module That Builds Basic Channel Program	Extended Search Module	Write Verify Module	Channel Program Number	
				No Extended Search	Extended Search
<b>Add a new record</b>					
Format F	IGG019K0	IGG019LA		3	6
Format U or V	IGG019KM	IGG019KY		4	Not applicable
Format VS	IGG019KN	IGG019KY		7	Not applicable
<b>Update an old record</b>					
Format F, V, or U (search on key)	IGG019KI	IGG019KW		2	5
Format F, V, or U (search on block identification)	IGG019KK	Not applicable	IGG019KQ	1	Not applicable
Format VS (search on key)	IGG019KR	IGG019KW		12(read) 13(write)	8(read) 9(write)
Format VS (search on block identification)	IGG019KR	Not applicable		10(read) 11(write)	Not applicable

Figure 4. Channel Program Generating Modules

DIAGRAM 7. ASSIGN BUFFER: HANDLE I/O INTERRUPTS (APPENDAGE PROCESSING)



### Notes for Diagram 7

One of the following dynamic buffering modules gets control from the EXCP processor before a request to read or write data is executed:

- IGG019LE—real storage is specified for the task; supports all record formats
- IGG019JA—virtual storage is specified for the task; supports all record formats

Note that the dynamic buffering modules also function as start I/O appendages in this processing.

The SIO appendages determine if EXCP was issued from the ASI routine, RELEX, or if the request is from the CEA/ABE appendage (to retry a request). If the request is from any of these routines, a branch entry is made to SMF with a negative one in register 1.

#### IGG019LE, JA

- 1 Determines whether the request should have a buffer assigned to it (it must be a read request, specify dynamic buffering, and not already have a buffer assigned to it). If the request should not be assigned a buffer, returns to the EXCP processor to initiate execution of the channel program. [SIORTNE]

For requests that need a buffer, gets a buffer from the buffer pool (if a buffer is available), puts the address of the assigned buffer in the channel program (thereby completing the channel program), and returns to the EXCP processor to initiate execution of the channel program. [ASSGNBUF]

#### IGG019LE

- 2 If real storage was specified for the task and a buffer is not available, puts the address of the IOB representing the request on the IOB buffer queue to wait until a buffer is available. Then returns to the EXCP processor (execution of the channel program will not be initiated). [PTONBUFQ]

#### IGG019JA

If virtual storage was specified for the task and a buffer is not available, puts the address of the IOB representing the request on the unscheduled list (if there's room on the list) to wait until a buffer is available. Then returns to the EXCP processor (execution of the channel program will not be initiated). [PTONBUFQ]

- 3 If no room was available on the unscheduled list, a larger unscheduled list will be obtained and the IOB will then be put on the newly acquired unscheduled list. When a buffer is freed by subsequent processing, it will be assigned to this request and the channel program will be scheduled. (See "Diagram 10 (Part 1 of 2). Complete Processing Format VS Request or Schedule Further Processing" on page 34.) [EXTEND]

#### IGG019LC

**Note:** This module, an end-of-extent appendage, can be entered only if extended search is specified. The EXCP processor enters it when a channel program reaches the end of an extent while searching for a block to be read or written, or a dummy record in which to write a block. IGG019KY enters it when the address of the next track to be searched is not in the current extent.

- 5 Determines the beginning address of the next extent to be searched. [TSTEXTNO]

If the upper limit of the search (the address of the track after the last track to be searched) is in this next extent, or if the upper limit of the search in the next extent does not equal the search address, exits as follows (these exits indicate the search will continue): [TSTUPLIM]

- If the module got control from IGG019KY, returns control to it. [EXITA]
- If the module got control from the EXCP processor and the new extent is on the same volume as the last extent, returns control to the EXCP processor, which will restart the channel program. [EXITB]
- If the module got control from the EXCP processor and the new extent is on a different volume from the last extent, returns control to the EXCP processor, which gives control to the IGG019KU (see IGG019KU (see step 8 below). [CLRENTR1]

- 6 If the upper limit of the search is in this extent and the search address and upper limit of the search are equal (which means the search has ended unsuccessfully), exits as follows: [MOVEZERO]
- If it was a request to add a record to the data set, sets the 'no space found' indicator in

the IOBDSTAT field and returns to: [NOSPACE]

- IGG019KY, if it was entered from that module [EXITE]
- The EXCP processor, which gives control to the abnormal end routine of IGG019KU (see step 8 below). [CLRENT1]

- If it was not a WRITE-add request, sets the 'no record found' indicator in the IOBDSTAT field and returns to the EXCP processor, which gives control to the abnormal end routine of IGG019KU (see step 8 below).

#### IGG019KU

- 8 Channel end routine—entered by the EXCP processor when a channel program terminates with a normal channel end or end-of-data, or an incorrect length condition is indicated.

If the channel program terminated normally, posts the user DECB, marks the IOB available, and returns to the EXCP processor requesting no post. This occurs unless one of the following conditions exists:

- Spanned records
- WRITE-add
- Exclusive control on this request
- Dynamic buffering on this request
- Relative feedback
- A newly obtained IOB is being used

Otherwise, if the channel program terminated normally (except for format F WRITE-add requests where the record was not written out and the upper limit of the search was not reached), branches to the exit effector routine of the task supervisor to schedule the ASI routine in module IGG019KA (format F, U, or V records) or IGG019KJ (format VS record). The exit effector routine returns to the EXCP processor. [NORMEND]

If the channel program terminated normally for a format F WRITE-add request, the record has not been written yet, and the upper limit of the search has not been reached, one of the following actions is taken:

- If the abnormal end routine of this appendage was previously

entered during the search for a dummy record, it turned off the command chain bit in the CCW that read in the data byte identifying the dummy record number. Then it restarted the channel program. In this situation, the channel end routine rechains the CCW, and then changes the starting address (IOBSTART field) of the channel program so it points to CCWs that will verify that the dummy record already found is still a dummy record. Then returns control to the EXCP processor to restart the channel program. [IECDCEA]

- If the dummy record, previously found, is no longer a dummy record when the channel program to verify the dummy record is executed (see preceding paragraph), changes the IOBSEEK and IOBSTART fields so a new dummy record will be searched for, beginning at the point where the previous dummy record was found. Then returns control to the EXCP processor to restart the channel program. [CHECKNOP]

If an end-of-data condition was detected during execution of the channel program, and the extended search option is in effect or the request is to add a new record to the data set, changes the channel program so the search will continue at the beginning of the data set. Then returns to the EXCP processor to restart the channel program. [SETEODS]

For any other end-of-data conditions, sets error codes in the IOB and branches to the exit effector routine of the task supervisor to schedule the ASI routine in module IGG019KA (format F, U, or V records) or IGG019KJ (format VS record). The exit effector routine returns to the EXCP processor. [SETEODS1]

If an incorrect length condition (block length specified by the user in the request macro or the DCB is not equal to the number of bytes read from or written to the device) was detected during execution of the channel program, does one of the following:

- WRITE request—leaves the incorrect length indicator in the IOBCSW field on and returns control to the EXCP processor. The EXCP processor will flag the IOB as a permanent error and then give control to the abnormal end routine in this module. [CEEXIT1]

**Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM**

- READ request (format F)—leaves the incorrect length indicator in the IOBCSW field on and returns control to the EXCP processor. [TSTV]
- READ request (format U)—turns the incorrect length indicator in the IOBCSW field off, and branches to the exit effector routine of the task supervisor to schedule the ASI routine in IGG019KA. The exit effector routine returns control to the EXCP processor. [RESET]
- READ request (format V or VS)—compares the length (in the block descriptor word) of the block being read to the number of bytes read by the channel program. If they are equal, turns the incorrect length indicator in the IOBCSW field off, and branches to the exit effector routine of the task supervisor to schedule the ASI routine in IGG019KA (format V records) or IGG019KJ (format VS records). The exit effector routine returns control to the EXCP processor. If the two lengths are not equal, leaves the incorrect length indicator in the IOBCSW field on and returns control to the EXCP processor. The EXCP processor will flag the IOB as a permanent error, and then give control to the abnormal end routine in this module. [CEEXIT2]

**Abnormal end routine**—entered by the EXCP processor when a channel program terminates with a permanent or device error condition.

If a permanent error (one from which the system cannot recover) was detected during execution of the channel program, sets the abnormal completion code in the IOBSTAT1 field and branches to the exit effector routine of the task supervisor to schedule the ASI routine in IGG019KA (format F, V or U records) or IGG019KJ (format VS records). The exit effector routine returns control to the EXCP processor. [IECDCEB]

If the request was a format F WRITE-add request, one of the following actions will be taken:

- If the error occurred before the dummy record was found, turns off the command chain bit in the CCW that read in the data byte identifying the dummy record number, and updates the IOBSEEK address with the last track searched (IOBDNRFC field). Then turns off the I/O error flag

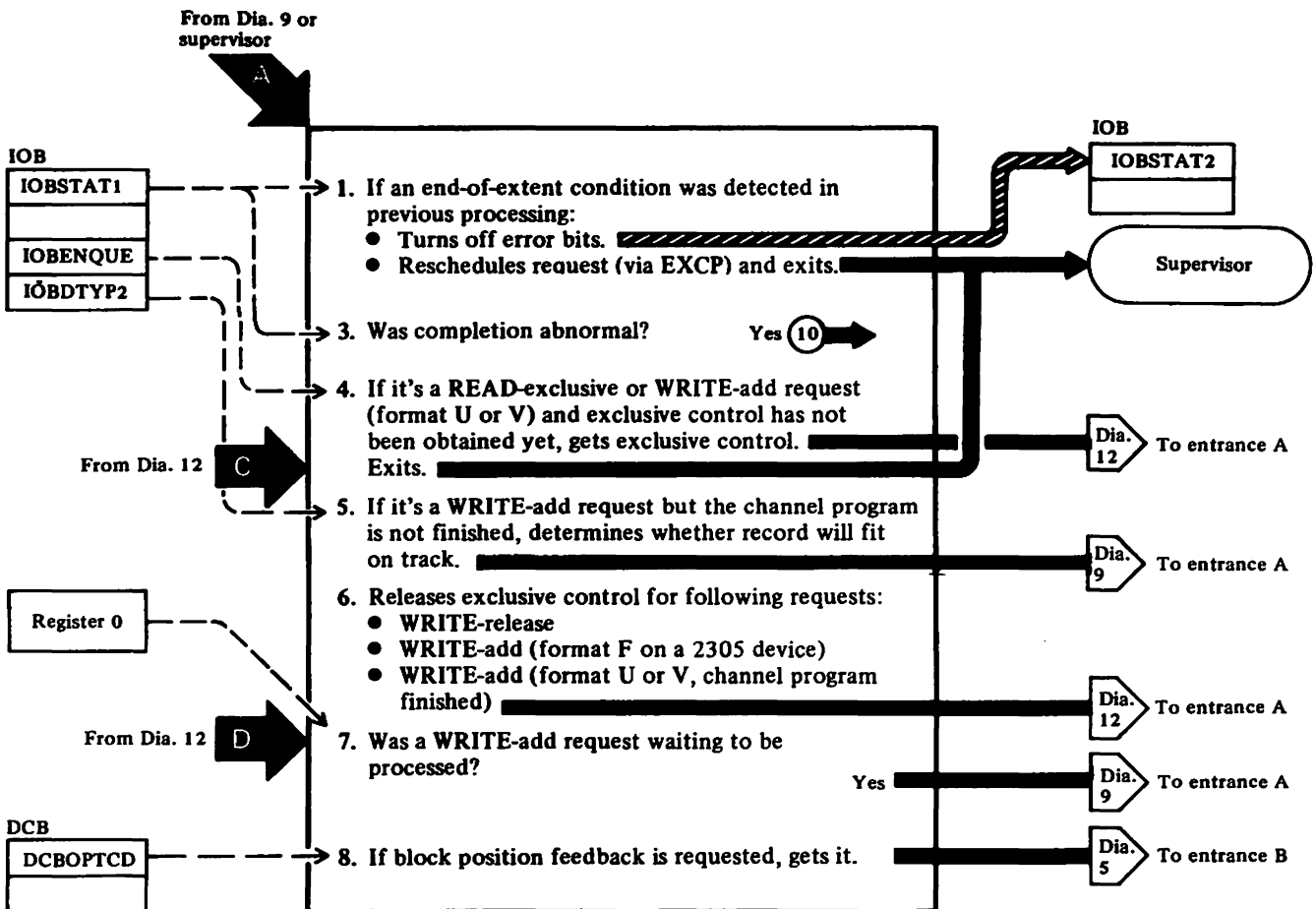
(IOBFLAG1) and returns control to the EXCP processor to restart the channel program. (If this routine was previously entered for this condition and the command chain bit was turned off, control is immediately returned to the EXCP processor. ERP routines will then get control and retry the channel program. If a permanent error does not occur, the channel end routine will be entered next.) [IECDCEB]

- If an interrupt occurred after the dummy record was found but before it was updated, changes the starting address (IOBSTART) of the channel program so it points to CCWs that will verify that the dummy record already found is still a dummy record. Turns off the I/O error flag (IOBFLAG1) so that the abnormal end routine can be reentered (bypassing processing in the ERP routines). Returns control to the EXCP processor to restart the channel program. (If this section of the abnormal end routine is entered ten times, turns on the permanent error flag and branches to the exit effector routine of the task supervisor to schedule the ASI routine in IGG019KA.
- If the error occurred after or during updating of the dummy record, changes the starting address (IOBSTART) of the channel program to execute only those CCWs that update the dummy record. Then returns control to the EXCP processor. ERP routines will ultimately get control and retry the channel program. (If a permanent error does not occur, the channel end routine is entered when I/O activity is completed.) [RESETIT]

If a device error occurred during execution of the channel program, returns control to the EXCP processor to attempt recovery using an ERP. (If the error condition remains after the ERP is used, the error is classified as permanent.)

**Note:** If the EXCP call being processed is other than the initial one for this request, the ECB address in the IOB (IOBECBPT) was changed to point to the IOBCSW. This preserves the contents of the user's ECB, whose address is saved in the IOBDQPTR. This address is restored to the IOBECBPT on entry to this module (for either channel end or abnormal end processing).

DIAGRAM 8 (PART 1 OF 2). COMPLETE PROCESSING FORMAT F, U, OR V REQUEST OR SCHEDULE  
 FURTHER PROCESSING



Notes for Diagram 8 (Part 1 of 2)

IGG019KA (ASI routine)

The ASI routine gets control from the supervisor if the channel end/abnormal end appendage caused the ASI routine to be scheduled (see "Diagram 7. Assign Buffer: Handle I/O Interrupts (Appendage Processing)" on page 24). The other entry points to this module are the result of control being returned from modules to which the ASI routine gives control.

- 1 Before beginning processing, adds the IOB representing this request to the IOB pool if the IOB was gotten via a GETMAIN macro in the base routine (see "Diagram 4. Prepare to Execute Channel Program" on page 18) of this module. [POOL]

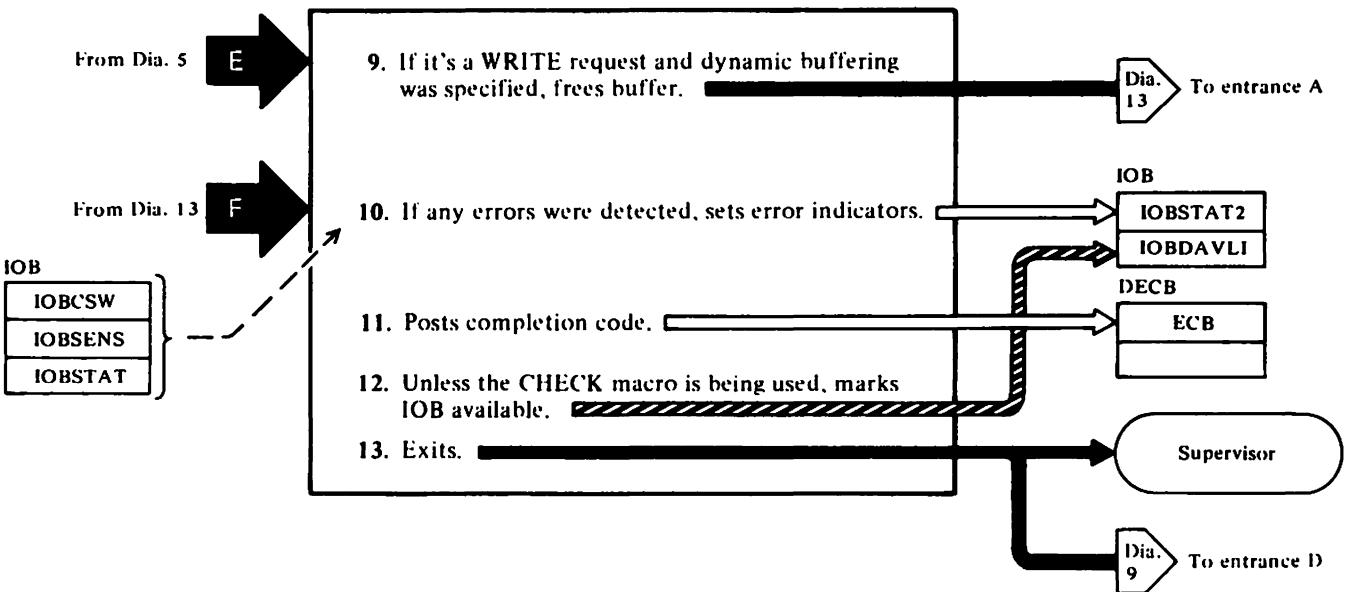
If an end-of-extent condition was detected in previous processing and the new extent was on another volume, turns off the error bits in the IOB and then reschedules execution of the channel program for this request by issuing the EXCP macro. [INPOOL]

- 4 If this request is a WRITE-add request (and the two-part channel program has not completed—only R0 has been read in) or a READ-exclusive request and exclusive control has not been obtained yet (IOBENQUE bit off), issues SVC 53, which gives control to module IGC0005C to get exclusive control. [ENQUEUE]

- 5 The channel program for a WRITE-add request has two parts. The first part reads in R0. The second part, subsequently executed, writes the record on the track. If R0 has been read at this point in the processing and exclusive control has been obtained (IOBENQUE bit on), branches to module IGG019KM to see if the record to be added will fit on the track. [ASI2NDEN]
- 6 Issues SVC 53, which gives control to module IGC0005C to release exclusive control for these types of requests. In the case of the WRITE-release request, exclusive control is being released for the corresponding READ-exclusive request. [DEQUEUE]
- 7 On return from IGC0005C, if the request releasing exclusive control was a WRITE-add (format U or V) request, the address of an IOB for a request waiting to get exclusive control of the track being released may be returned. IGC0005C will have given this waiting request exclusive control of the track. The waiting request is now ready to be processed by module IGG019KM, which will determine whether the record to be added will fit on the track. No further processing is necessary for the current WRITE-add request—the one that released exclusive control—as module IGC0005C completed processing that request before returning to IGG019KA.
- 8 If block position feedback is requested or the request was a READ-exclusive, branches to module IGG019KC, KG, or KH to get feedback. [DASIB]



**DIAGRAM 8 (PART 2 OF 2). COMPLETE PROCESSING FORMAT F, U, OR V REQUEST OR SCHEDULE  
 FURTHER PROCESSING**

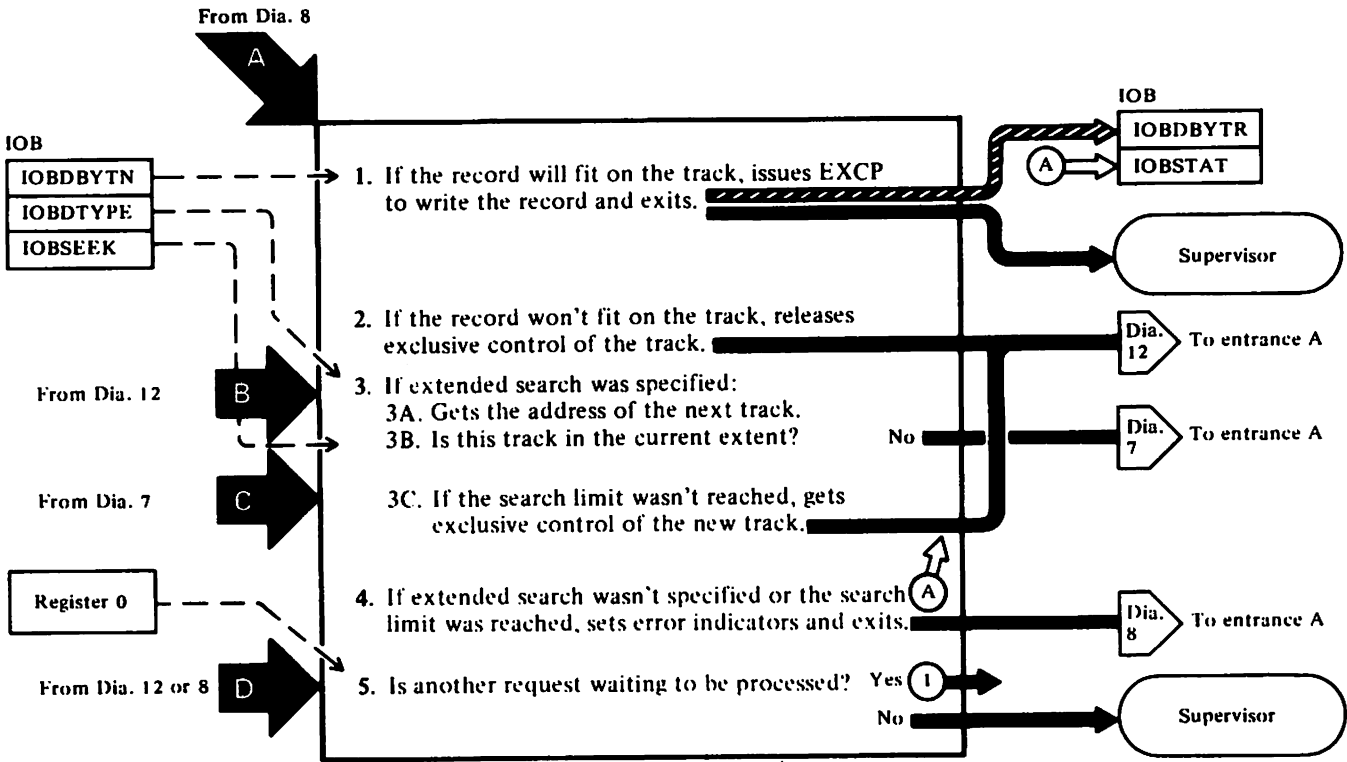


**Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM**

**Notes for Diagram 8 (Part 2 of 2)**

- 9 Issues the FREEDBUF macro, which gives control to module IGC0005G to free the buffer. [DASIC]
- 10 Analyzes the cause of the error by examining the CSW and sense bytes (put in the IOB by the EXCP processor) and the IOBSTAT field (contains error indicators set by other BDAM modules).
- 11 Issues the POST SVC to post the completion code. [MAKEAVL]
- 12 If the CHECK macro is being used by the processing program to wait for the completion of READ or WRITE requests (rather than the WAIT macro), the check module IGG019LI will mark the IOB available. [MAKEAVL]
- 13 If this module was branched to by IGG019KM (rather than entered from the supervisor), returns to IGG019KM, which will determine if a request is waiting to be processed. [MAKEAVL]

**DIAGRAM 9. DETERMINE WHETHER FORMAT U OR V RECORD WILL FIT ON TRACK**



Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM

Notes for Diagram 9

IGG019KM

- 1 Determines whether R0 is valid. If not, continues processing at label NOFIT (step 2) below. [IECDSF2]

If it is the first time this module has been entered for this request, calculates the number of bytes in the record to be added to the track. [MAKECALC]

Determines whether the record will fit on the track by comparing the size of the record with the number of available bytes left on the track. [NOCALC]

If the record will fit on the track, updates the track capacity field (IOBDBYTR), turns on the 'record fits' bit (IOBSTAT field), and issues an EXCP macro to write the record on the track. [STORE]

- 2 If the record will not fit on the track, issues SVC 53, which gives control to module IGC0005C to release exclusive control of the track. On return from IGC0005C, saves the contents of register 0. (This is done because, if an IOB representing a request is waiting to get exclusive control of the track just released, its address is in register 0. IGC0005C will have given exclusive control of the track to that waiting request. The waiting request is now ready to be processed by this module (IGG019KM) when processing on the current request is complete.)

- 3A If extended search is specified, branches to module IGG019KY. [XSCHLNK]

IGG019KY

Gets the address of the next track in the data set. [BEGIN]

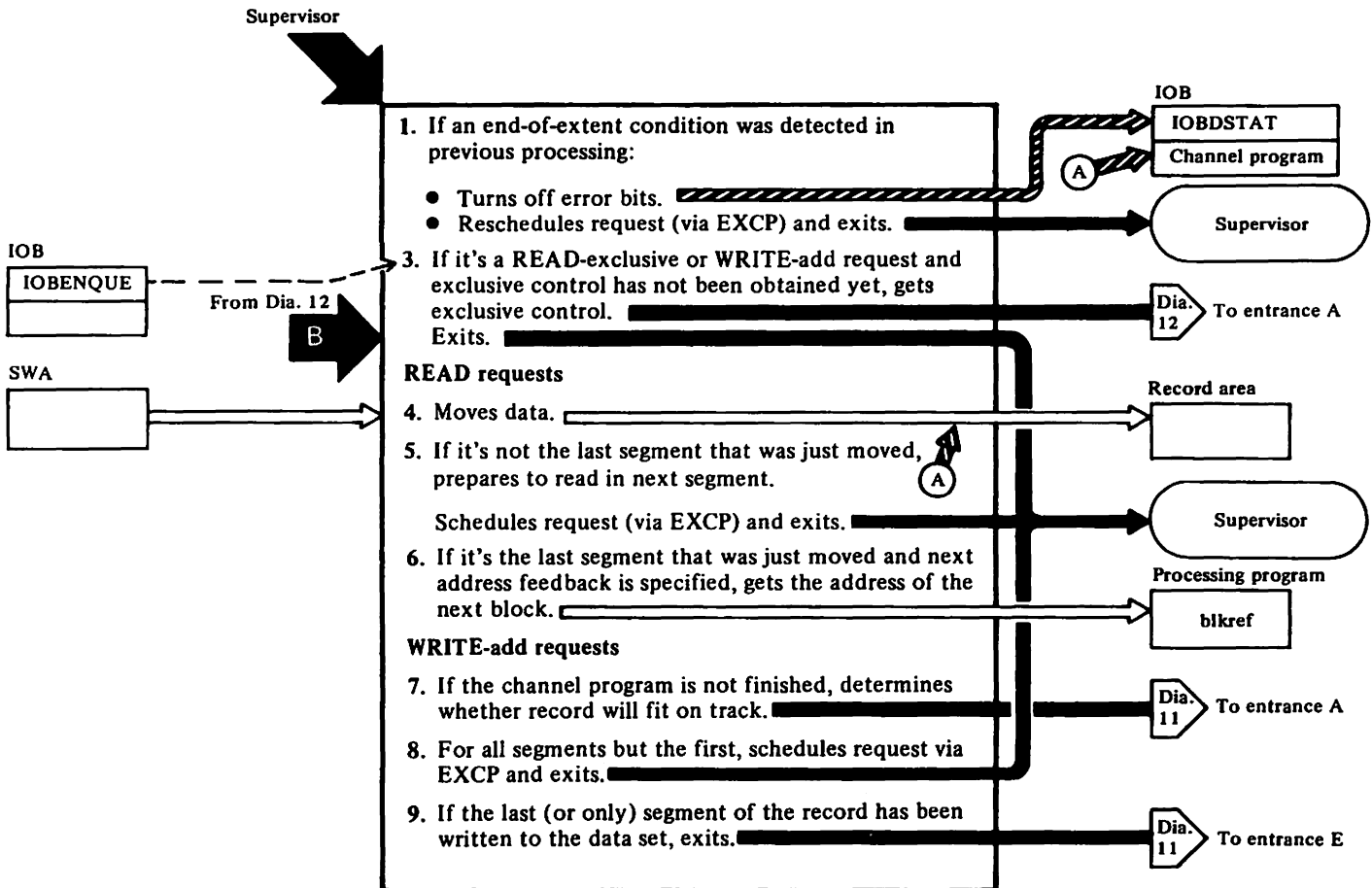
- 3B If this track is in a different extent, branches to module IGG019LC to get the address of the next extent. On return from IGG019LC, returns control to IGG019KM. [EXIT1]

IGG019KM

- 3C If the search limit was not reached when the new track was obtained, issues SVC 53, which gives control to module IGC0005C to get exclusive control of the new track and then returns. (At this point, all processing on this current request is ended in this module. IGC0005C will have issued an EXCP macro for the current request, or added the IOB for the request to the unposted queue to wait until the track is available. IGG019KA will ultimately get control and enter this module at step 1 to see if the record will fit on the track enqueued by IGC0005C.) [KMENQ]

- 5 If the address of a waiting IOB was not returned in register 0 earlier in the processing (see step 2, above), exits to the supervisor. If the address of a waiting IOB was returned, begins processing this waiting IOB (which now becomes the current IOB) at label MAKECALC in step 1.

**DIAGRAM 10 (PART 1 OF 2). COMPLETE PROCESSING FORMAT VS REQUEST OR SCHEDULE FURTHER PROCESSING**



Notes for Diagram 10 (Part 1 of 2)

IGG019KJ (ASI routine)

- 1 Before beginning processing, adds the IOB representing this request to the IOB pool if the IOB was gotten via a GETMAIN macro in the base routine (see "Diagram 4. Prepare to Execute Channel Program" on page 18) of this module. [POOLIOB]

If an end-of-extent condition was detected in previous processing and the new extent was on another volume, turns off the error bits in the IOB and then reschedules execution of the channel program for this request by issuing the EXCP macro. [RESTART]

- 3 If this request is a WRITE-add request (and the channel program has not completed—only R0 has been read in) or a READ-exclusive request and exclusive control has not been obtained yet (IOBENQUE bit off), issues SVC 53, which gives control to module IGC0005C to get exclusive control. (Note that the channel program for a WRITE-add request has two parts. The first part reads in R0 which, at this point in the processing, has been done. The second part, subsequently executed, writes the record on the track.) [ENQUEUE]

- 4 If this request is a READ request, moves the data that has been read from the device to the SWA into the record area. (This data may be just a segment of the record or, if the whole record will fit in the SWA, the entire record.) [ASIREAD2]

Increases the record length in the BDW in the record area by the segment length in the SDW. (Figure 1 contains a description of the fields in a format VS record.) [NOTTM]

- 5 If the segment control code in the SDW indicates that the segment just read from the device was not the last (or only) segment, modifies the channel program to read the next segment of the record. [VERMID]

Calculates the address of the next segment to be read [ADD1RTN], and issues an EXCP macro to schedule execution of the channel program to read the segment. [EXCPAGN]

- 6 If it is the last or only segment that was just moved and next address feedback was specified, gets the actual address of the next data block or R0 (if RU was coded in the type operand of the READ macro and R0 is the next record in the data set). [ISITNA]

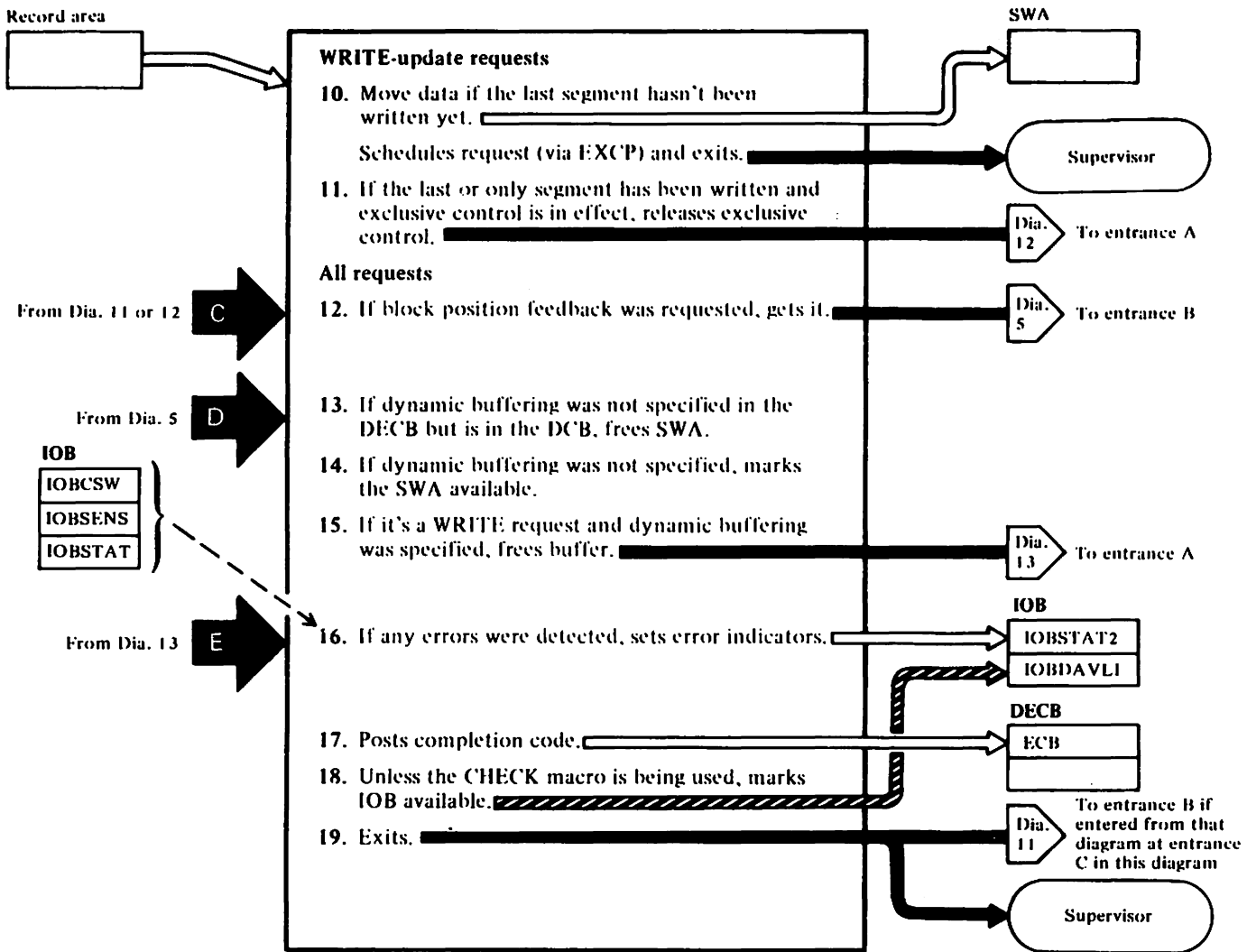
Branches to the system convert routine IECPLTV to convert the MBLCCHHR of the block to TTR. Then puts the TTR in the user's blkref field. [ACTTST]

- 7 If the request is a WRITE-add request and the channel program is not finished (only R0 has been read) but exclusive control has been obtained, branches to module IGG019KN to see if the record to be added will fit on the track. (If the record will fit, IGG019KN will issue an EXCP macro to write it on the track; if it will not fit, IGG019KN will try to accumulate enough space to write out the entire record and will then issue an EXCP macro to write the first segment.) [KNP8EXIT]

- 8 Issues an EXCP macro to write any segments of the record after the first segment. [ENDFW]

- 9 If the last segment has been written, branches to module IGG019KN to dequeue all tracks that have been enqueued for the request. [WRTDONE]

**DIAGRAM 10 (PART 2 OF 2). COMPLETE PROCESSING FORMAT VS REQUEST OR SCHEDULE FURTHER PROCESSING**



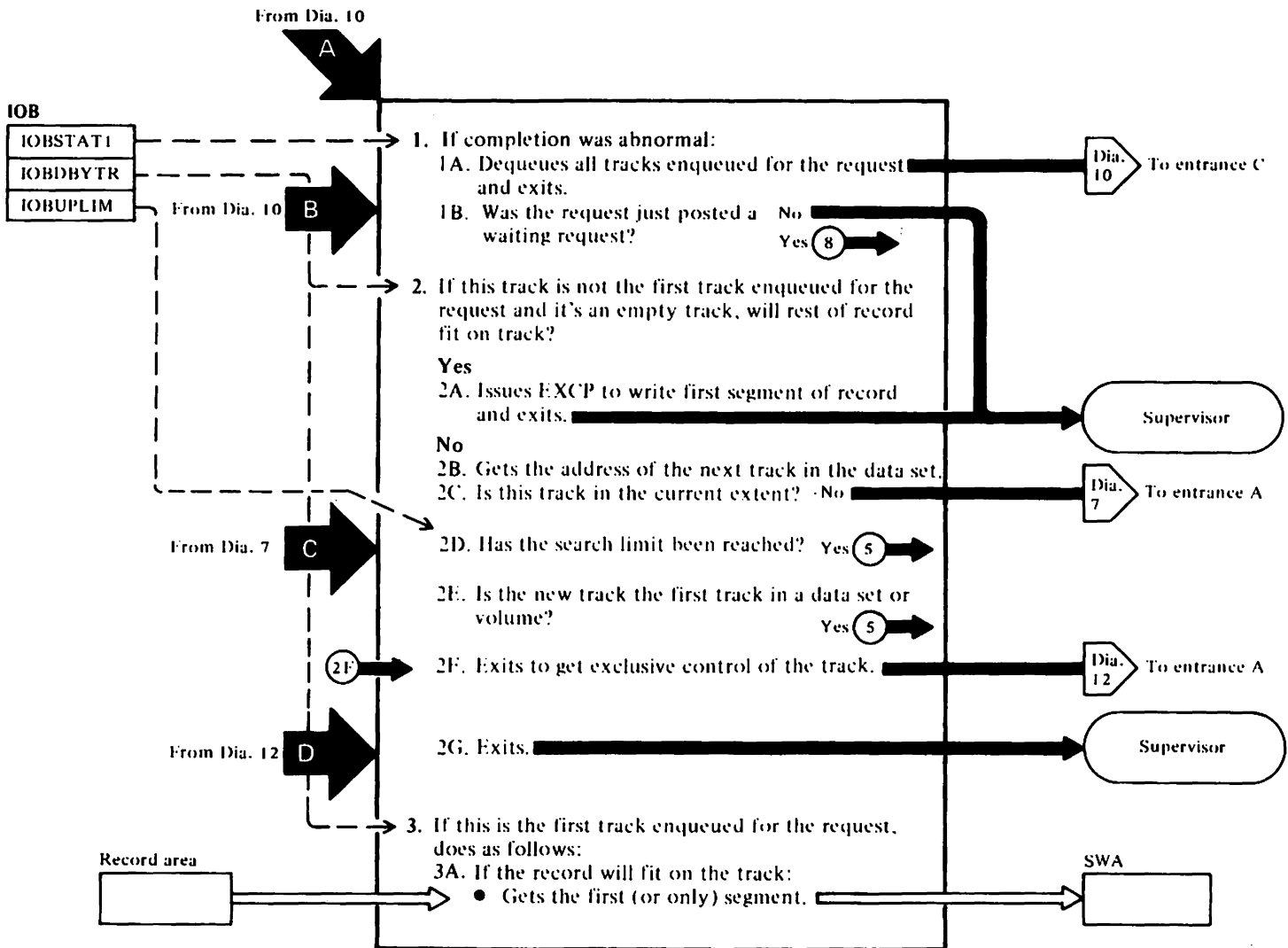
**Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM**

**Notes for Diagram 10 (Part 2 of 2)**

- 10** If the request is a WRITE-update request, moves data from the record area into the SWA. This data may be just a segment of the record or, if the entire record will fit into the SWA, the whole record. The SWAFINC field in the SWA is updated in this process. The SWAFINC field contains the offset, in bytes, in the record area from which the next segment of information should be moved to the SWA. [AFTRFLG]
- Issues an EXCP macro to write the segment in the SWA out to the device. [EXCPAGN]
- 11** If the last segment of the record has been written and the WRITE macro specified exclusive control (which is a request to release exclusive control for the READ request that corresponded to this WRITE request), issues SVC 53 which gives control to module IGC0005C to release exclusive control. [RDSEXIT]
- 12** If block position feedback was requested in the WRITE macro and the DCB, the search was by key, and relative addressing is in effect, branches to a conversion module to convert the MBBCCHHR of the specified block to TTR. [RTEXTIT]
- 13** If dynamic buffering was not specified in the DECB but is specified in the DCB, frees the SWA. [DASIC]
- 14** If dynamic buffering was not specified in the DCB or the DECB, zeros out the availability bit in the SWA. [FREESWA]
- 15** Issues the FREEDBUF macro to free the buffer, which gives control to module IGC0005G. [ASID]
- 16** Analyzes the cause of the error by examining the CSW and sense bytes (put in the IOB by the EXCP processor) and the IOBSTAT field (contains error indicators set by other BDAM modules). [ASIERR]
- 17** Issues the POST SVC to post the completion code. [SKPAVL]
- 18** If the CHECK macro is being used by the processing program to wait for the completion of READ or WRITE requests (rather than the WAIT macro), the check module IGG019LI will mark the IOB available. [MAKEAVL]



DIAGRAM 11 (PART 1 OF 2). DETERMINE WHETHER FORMAT VS RECORD WILL FIT ON TRACK



Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM

Notes for Diagram 11 (Part 1 of 2)

IGG019KN

- 1A If completion was abnormal while writing a record to the track, dequeues all tracks that were enqueued for the request [ACA1] and then turns on the 'record fits' bit (IOBSTAT1 = X'02') to indicate to module IGG019KJ that this request is complete and should be posted. [ACB4] Branches to IGG019KJ to post the request. [GOTOPOST]
- 1B On return from module IGG019KJ, determines whether the request just posted was a waiting request (one that got exclusive control of a track in module IGC0005C while that module was dequeuing tracks for a previous request). If not, exits. [POSTNOQ] If so, continues processing the current request (the one for which tracks were being dequeued) at step 8 below. [GOTOPOST]
- 2 If this is not the first track enqueued for the request (IOBDBYTR is greater than 0) and the track is empty, determines whether the rest of the record will fit on the track. [NOTFIRST]
- 2A If so, turns on 'record fits' bit (IOBSTAT1 = X'02'). [FINSETUP] Issues an EXCP macro to write the first segment of the record on the track. (Module IGG019KJ—see "Diagram 10 (Part 1 of 2). Complete Processing Format VS Request or Schedule Further Processing" on page 34—will get control after this segment is written and issue the EXCP macro to write all other segments.) [EXCP]

IGG019KY

- 2B If the rest of the record will not fit on the track, gets the MBBCCHHR of the next logical track in the

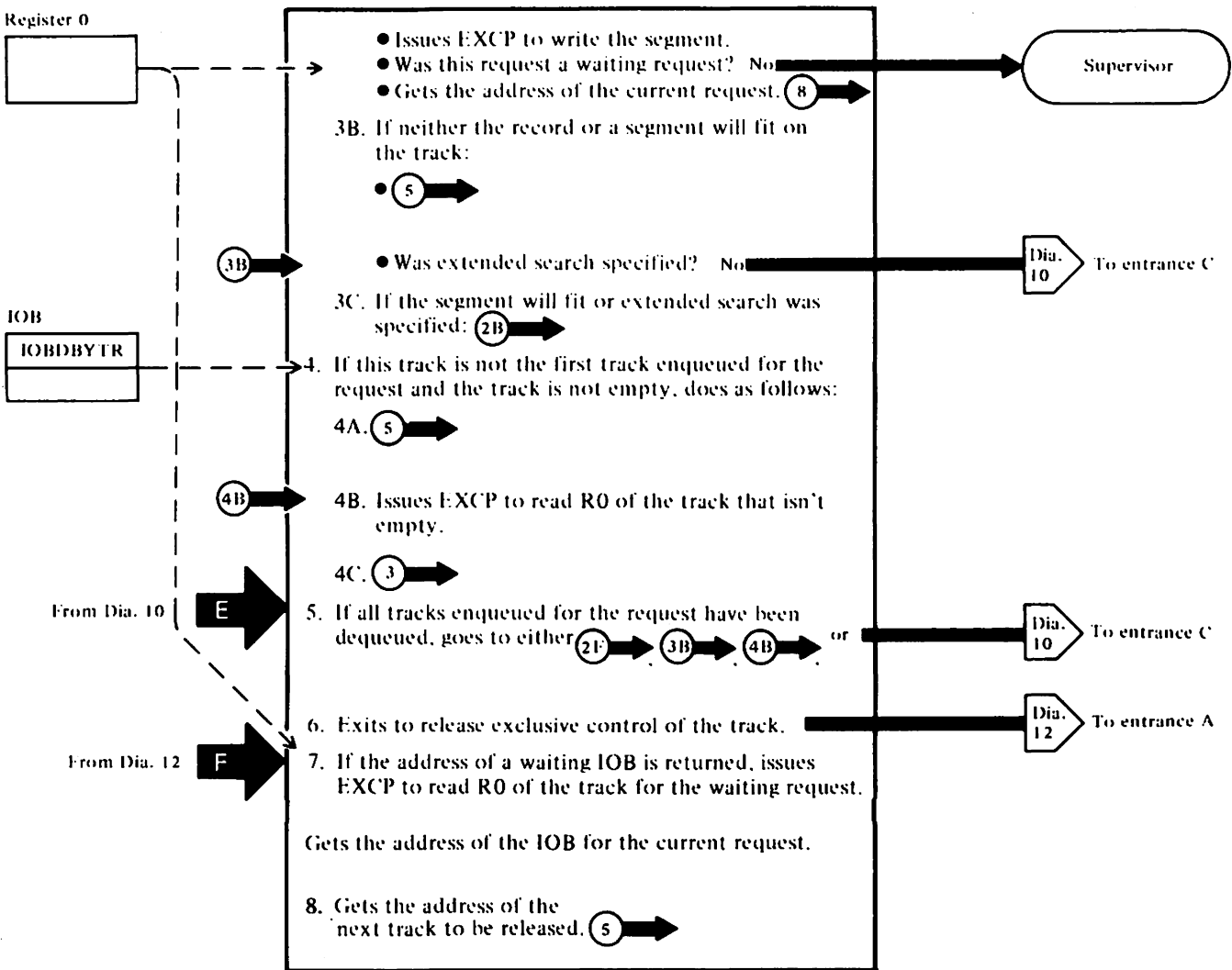
data set if it is in the same extent as the last track. [BEGIN]

- 2C If the next track is not in the same extent as the last track, branches to module IGG019LC to get the address of the next extent. [TSTEXT]

IGG019KN

- 2D If the search limit has been reached, sets the 'no space found' bit in the IOB and then continues processing at step 5 below. (The tracks enqueued for the request will be dequeued, and then control will be given to module IGG019KJ to post the request.)
- 2E If the new track is the first track in a data set, extent, or volume, continues processing at step 5 below. (All the tracks previously enqueued in another extent or volume must be dequeued.) After all tracks have been dequeued, step 5 will return control at (2F) below, and the new track will be the first track enqueued on for this request. [REINIT]
- 2F Issues SVC 53, which gives control to module IGC0005C to get exclusive control. [TRKOK]
- 2G On return from module IGC0005C, exits. [TSTIFDON]
- 3 If this is the first track enqueued for the request (IOBDBYTR=0) and it is the first pass through this code (IOBSTAT1 is not equal to X'10'), calculates the size of the record. [KEYED]
- 3A If the entire record will fit on the track, turns on the 'record fits' bit (IOBSTAT1=X'02') and moves from the record area to the segment work area as many bytes of data as will be written out in this segment (which may be the whole record). [AFTERLL]

DIAGRAM 11 (PART 2 OF 2). DETERMINE WHETHER FORMAT VS RECORD WILL FIT ON TRACK



Notes for Diagram 11 (Part 2 of 2)

Issues an EXCP macro to write the first segment of the record on the track. (Module IGG019KJ—see "Diagram 10 (Part 1 of 2). Complete Processing Format VS Request or Schedule Further Processing" on page 34—will get control after this segment is written and issue the EXCP macro to write all other segments.) [EXCP]

If this request was not a waiting request (one that got exclusive control of a track in module IGC0005C while that module was dequeuing tracks for a previous request), exits. [TSTIFDON]

If it was a waiting request, gets the address of the current request (the one for which tracks were being dequeued) and continues dequeuing tracks for it at step 8 below.

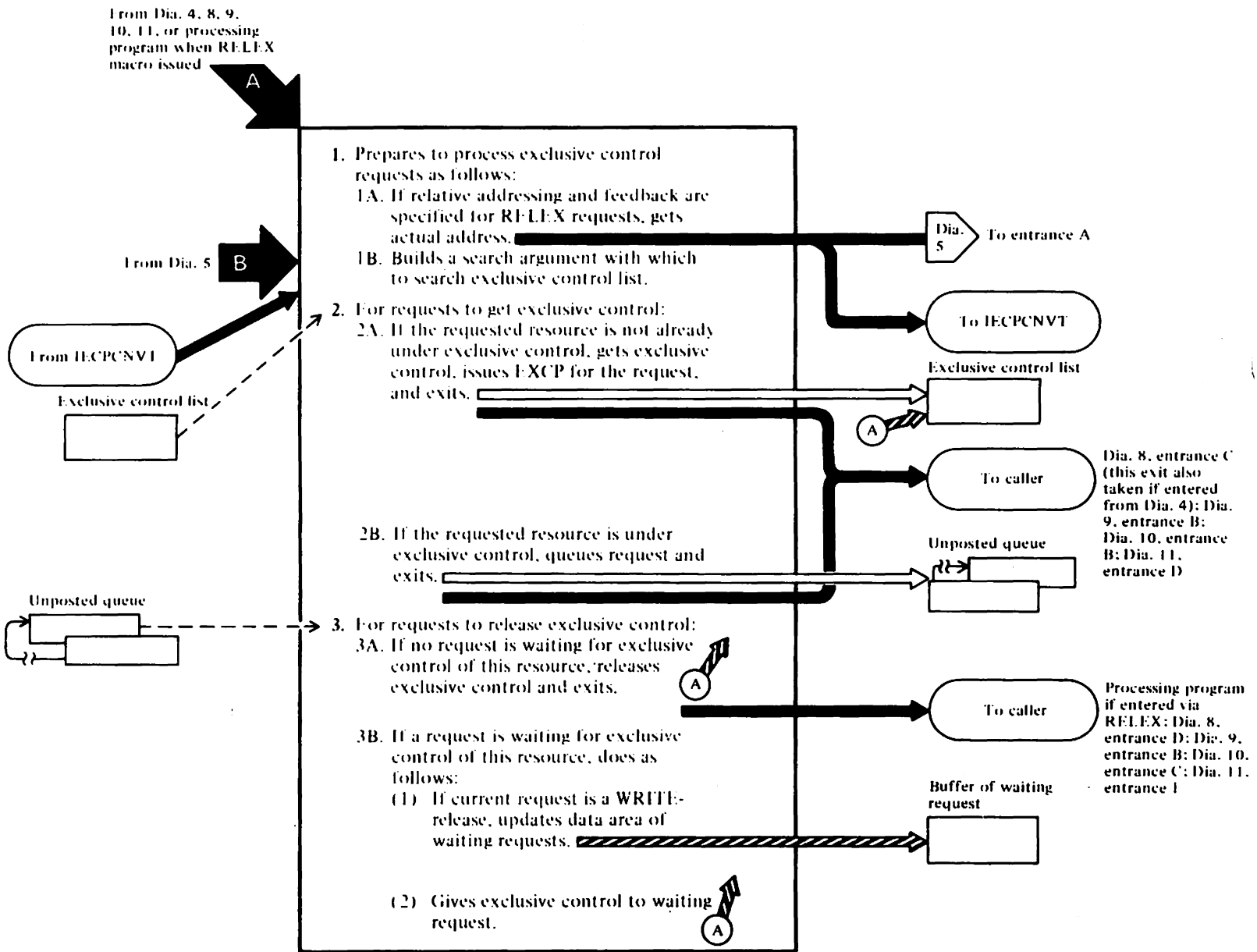
- 3B If neither the whole record or the first segment of it will fit on this track, continues processing at step 5 below. (The track will be dequeued.) [TRKFULL] On return from the dequeuing routine, determines whether extended search was specified. If not, sets the 'no space found' bit in the IOB, turns on the 'record fits' bit (IOBSTAT1 = X'02') to indicate to module IGG019KJ that this request should be posted, and then returns to IGG019KJ.
- 3C If the first segment will fit on the track, or extended search was specified, continues processing at 2C above to get the address of the next track.
- 4A If this is not the first track enqueued for the request and the track is not empty, continues processing at step 5 below. (The track will be dequeued.) [NOTEMPTY]

- 4B On return from the dequeuing routine, modifies the channel program to read in R0 for the new track (which is the track that was not empty). [OTHRFRST]
- 4C Continues processing at step 3.
- 5 When all tracks for the request have been dequeued, routes control. Control may go to that section of the code in this module that gave step 5 control (2F, 3B, or 4B as indicated in the diagram) or, if no return to the step that gave control is indicated in the diagram, control will go to module IGG019KJ where the request will be posted as complete. [ACAI]
- 6 Puts the address of the track to be dequeued in the IOBSEEK field, turns off the 'record fits' bit in the IOBSTAT1 field (so the request won't be posted by module IGC0005C), turns on the IOBENQUE bit (so module IGC0005C will know it's supposed to dequeue, not enqueue), and issues SVC 53, which gives control to IGC0005C to dequeue. [GOTOLG]
- 7 If the address of an IOB for a waiting request (one that has just received exclusive control of the track being dequeued) is returned from module IGC0005C in register 0, processing changes from releasing tracks for the current request to processing the waiting request. Issues the EXCP macro to again read in R0 of the track the waiting request has exclusive control of. [EXCP] Then gets the address of the IOB for the current request and continues processing at step 8 below. [TSTIFDON]

IGG019KY

- 8 Gets the address of the next track to be released and then continues processing at step 5.

DIAGRAM 12 (PART 1 OF 2). GET OR RELEASE EXCLUSIVE CONTROL.



Notes for Diagram 12 (Part 1 of 2)

IGC0005C

Bits, called audit trail bits, are set in the SVRB extended save area after each step in the processing in this module has been completed. If the exclusive control recovery modules subsequently get control from the recovery/termination manager (R/TM) after abnormal termination (see "Diagram 16. Free System Resources after Abnormal Termination" on page 54), they examine these bits to determine what resources need to be freed.

This module is entered to get or release exclusive control of a resource. The following list shows which requests cause this module to be entered, and which resource each request is getting or releasing exclusive control of:

Type of Request	Resource
RELEX	Block
WRITE-release	Block
READ (exclusive control specified)	Block
WRITE-add	R0 of first track in data set, or R0 of track being examined for space in which to add a block

1 Before beginning processing, issues an ESTAE macro to establish the entry point in this module for R/TM, which enters it if an abnormal termination condition is encountered during processing related to this exclusive control SVC. ("Diagram 16. Free System Resources after Abnormal Termination" on page 54 documents this entry point.) [BASEVAL]

1A For RELEX requests, gets a work area and builds dummy control blocks (IOB and DECB) in it. [RELEX] If feedback and relative addressing are specified in the DCB, gives control (via the SYNCH macro) to one of the following to convert the relative address of the specified block to an actual address:

- IGG019KE (relative block addressing specified—no track overflow) [RELBLK]
- IGG019KF (relative block addressing and track overflow specified)
- IECPCNVT (relative track addressing specified) [TTR]

When control is returned, the actual address of the specified block is in the request's IOB.

1B Before getting or releasing exclusive control of a resource, does the following processing.

- Verifies the DEB. [DEBCHKRT]
- Builds a search argument to use when searching the exclusive control list for an entry. The 8-byte search argument consists of the UCB address and CCHHR of the resource for which exclusive control is requested. [READEXCL]
- Except for WRITE-add requests for format F records with extended search specified, verifies that the track is in the data set described by the DEB. [LOADEXTN]
- Determines whether the request is to get or release exclusive control of a resource: [TYPETST]

Type of Request	Action Requested
READ (exclusive control specified)	Get exclusive control
WRITE-add	Get exclusive control if IOBENQUE bit off; release exclusive control if IOBENQUE bit on
WRITE-release	Release exclusive control
RELEX	Release exclusive control

2A If the requested resource is not already under exclusive control, searches the exclusive control list for an available entry in which to put the UCB address and CCHHR of the requested resource. If an entry is not found, gets a new segment of the exclusive control list before continuing. [SCANZERO]

Puts the UCB address and CCHHR for the requested resource on the exclusive control list and issues the ENQ macro to put the resource on the system queue. Then turns on the IOBENQUE bit (which indicates that this request has exclusive control), and issues an EXCP macro to:

- Get the latest copy of the record (READ request),

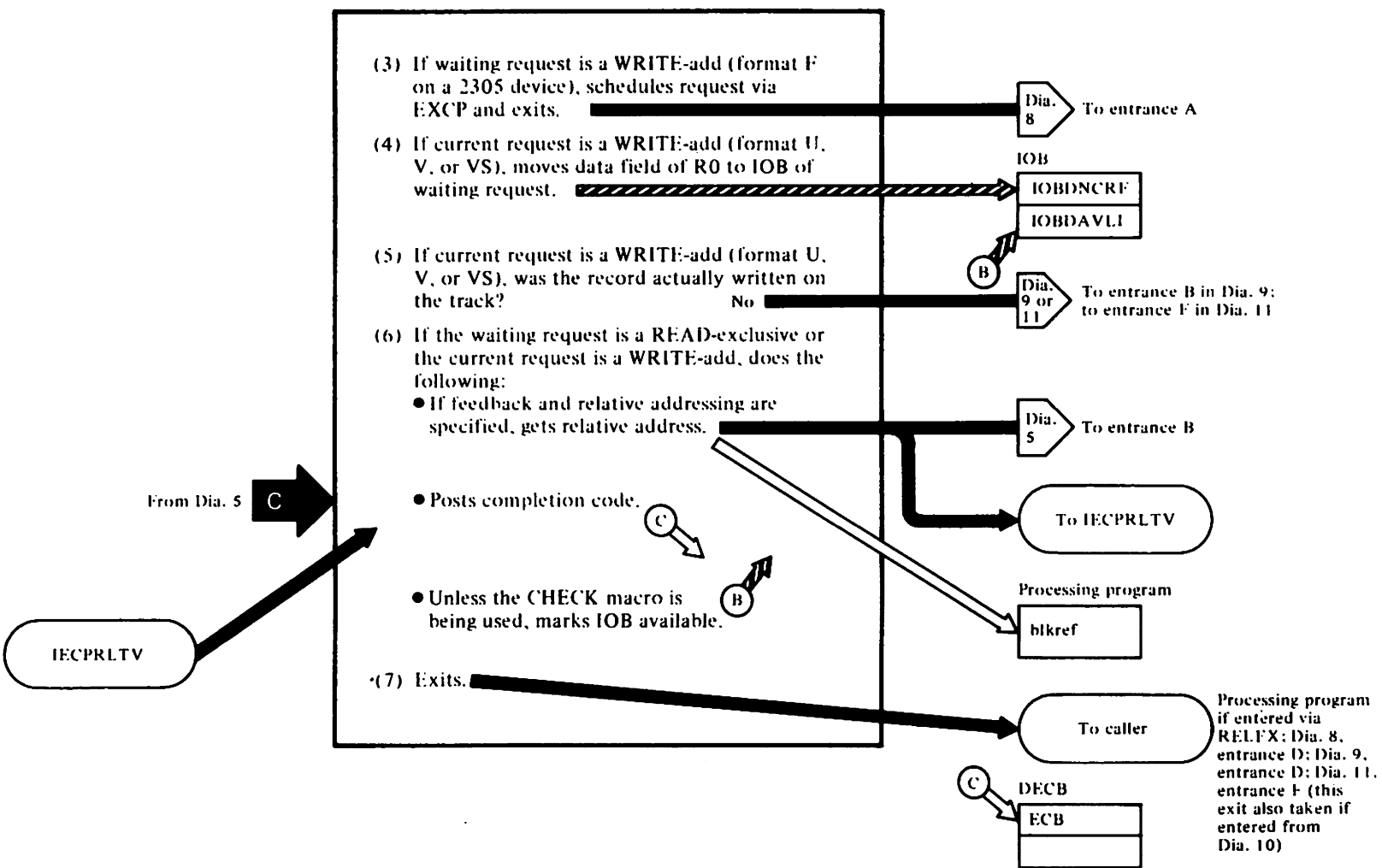
- Get the latest copy of R0 (WRITE-add, format V, U, or VS), or
  - Begin execution of channel program (WRITE-add, format F record on a 2305 device; this module was entered from the base routine of IGG019KA for this type of request before the EXCP macro was issued to begin the channel program). [POSTQ]
- 2B** If the requested resource is already under exclusive control (an entry for it is on the exclusive control list), puts the IOB representing the request on the unposted queue so the request can be completed later, when exclusive control of the resource is released. [PUTONQ]
- 3** Searches the unposted queue for IOBs waiting to get exclusive control of the resource for which exclusive control is about to be released.
- 3A** If there are no IOBs waiting on the queue, issues a DEQ macro to remove the resource from the system queue, and then clears the associated exclusive control list entry. [WRTOTYPE]
- 3B** If an IOB is waiting on the unposted queue and
1. The request for which exclusive control is being released is a WRITE-release request, moves the data portion of the record from the buffer of the IOB about to release exclusive control to the buffers of all waiting IOBs. (These waiting IOBs will be for READ-exclusive requests.) This is done so the waiting requests will have a current copy of the record to work with. [DATAMOVE]
  2. Takes the first IOB waiting for exclusive control off the unposted queue and gives it exclusive control; turns on IOBENQUE bit.

**Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM**

**This page intentionally left blank.**



DIAGRAM 12 (PART 2 OF 2). GET OR RELEASE EXCLUSIVE CONTROL



Notes for Diagram 12 (Part 2 of 2)

3. If this IOB was for a WRITE-add request format F on a 2305-2 device, issues an EXCP macro to begin execution of the channel program; then returns to IGG019KA (ASI routine) to finish processing the request that just released exclusive control. [DEQEXCP]
4. If this IOB was for a WRITE-add request (format U, V, VS), updates R0 in its IOB with the contents of R0 from the releasing requests IOB. [WTADVU]
5. For WRITE-add requests (format U, V, VS) that just released exclusive control, determines whether the record to be added was actually written on the track. If not, returns to IGG019KM (nonspanned records) or IGG019KN (spanned records).
6. If either the IOB just removed from the unposted queue (which will be a READ-exclusive request) or the IOB that just released exclusive control (for WRITE-add, format U, V, or VS, requests) specified feedback and

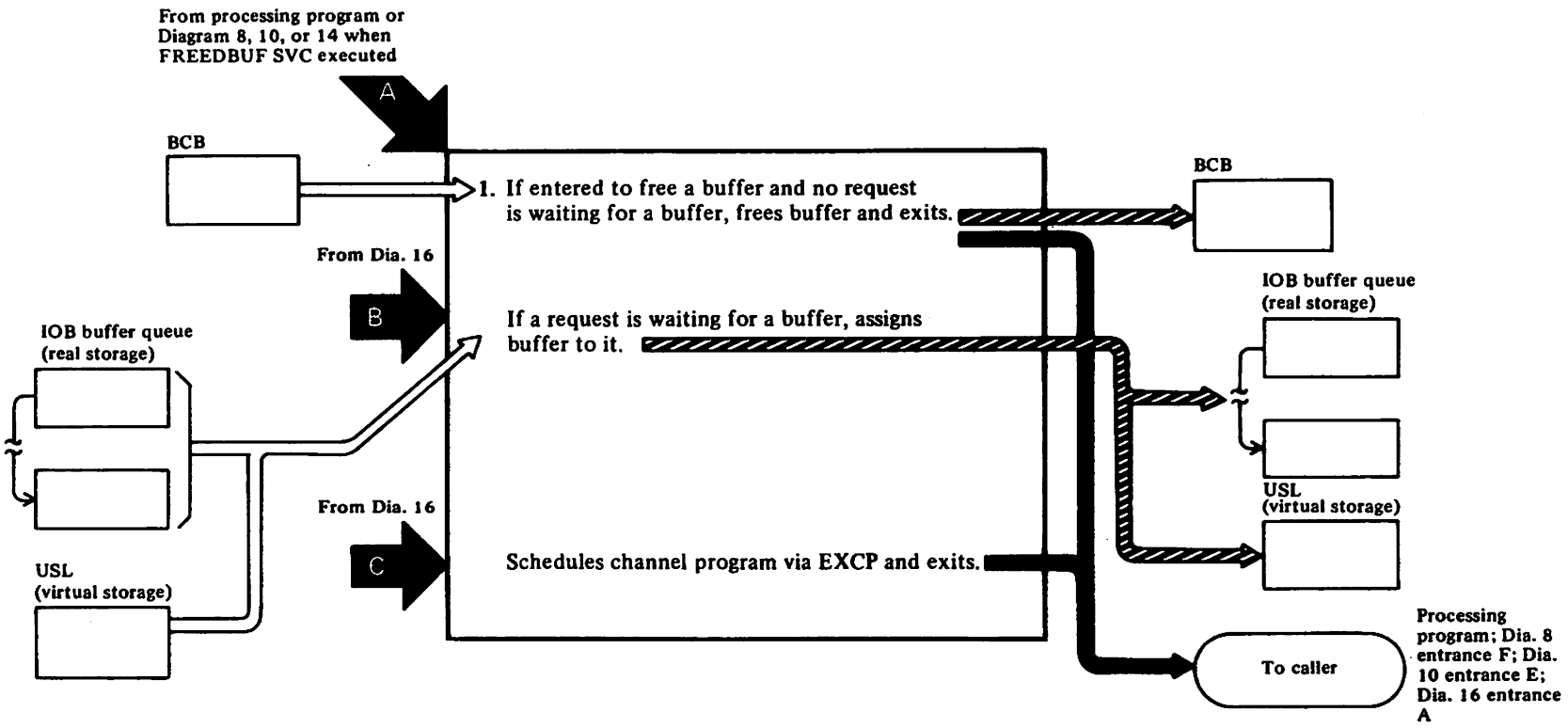
relative addressing in the DCB, branches (via the SYNCH macro) to one of the following routines or modules to convert the actual address of the block to a relative address:

- IGG019KG (relative block addressing specified; no track overflow) [TESTREL]
- IGG019KH (relative block addressing specified; track overflow)
- IECPRLTV (relative track addressing specified) [RELTRK]

On return from the address conversion routines, posts a completion code in the ECB for this request [POSTIOB] and marks this request's IOB available if the CHECK macro is not being used. [MAKEAVL]

7. Returns to the caller. If the request that just got exclusive control was a WRITE-add request, puts the address of its IOB in register 0. When the ASI routine or module IGG019KM gets control, it finishes processing this request.

**DIAGRAM 13. FREE BUFFER**



Notes for Diagram 13

IGC0005G

- 1 Before processing begins, issues an ESTAE macro to establish the entry point in this module for the recovery/termination manager (R/TM), which enters it if an abnormal termination condition is encountered during processing associated with the FREEDBUF SVC. [MOVELIST] Validates DEB. [CHECKDEB]

If the request to free a buffer is invalid, returns to the processing program. [RETURN]

IGG019LE, JA

**Note 1:** IGG019LE (all record formats) gets control if real storage is specified for the task; IGG019JA (all record formats) gets control if virtual storage is specified for the task.

**Note 2:** Bits called audit trail bits are set in the extended save area after each step in the processing in these modules has been completed. If the FREEDBUF recovery module, IGC0005G, subsequently gets control from R/TM as the result of an abnormal termination condition (see "Diagram 16. Free System Resources after Abnormal Termination" on page 54), it examines these bits to determine which resources need to be freed, and then gives control to the dynamic buffering modules at the appropriate entry point to free these resources.

If no IOBs are waiting on the unscheduled list (virtual storage)

or IOB buffer queue (real storage), frees the buffer for this request by returning it to the chain of available buffers. [FBUNOREQ in IGG019KL,LE;MAKEVAL in IGG019JA,JB]

IGC0005G

Returns control to the caller. [RETURN]

IGG019LE, JA

If an IOB is waiting on the unscheduled list (virtual storage) or IOB buffer queue (real storage) for a buffer, clears the IOB address from the list or queue, assigns the newly freed buffer to the request represented by this IOB, initializes the channel program, and then issues an EXCP macro to schedule the I/O request. [NXTSLT in IGG019LE;CHAIN in IGG019JA]

IGC0005G

Returns control to the caller. [RETURN]

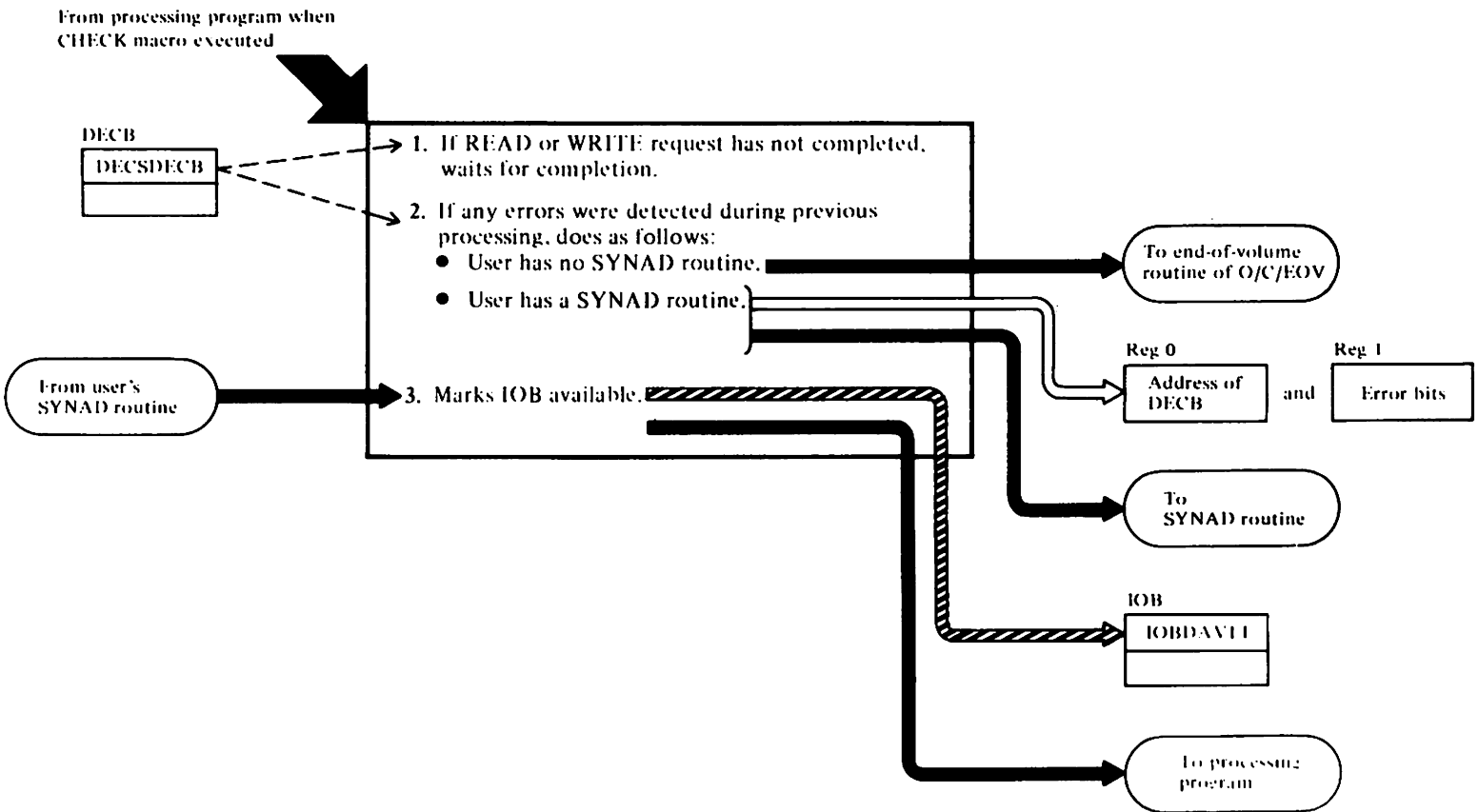
IGG019JA

- 2 Gets an unscheduled list twice the size of the current unscheduled list and initializes it by moving all entries from the current unscheduled list to the new unscheduled list. Then clears the unused slots in the new unscheduled list (which now becomes the current list). [EXTEND]

IGC0005G

Returns control to the caller. [RETURN]

**DIAGRAM 14. CHECK IF I/O OPERATION COMPLETED**



**Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM**

**Notes for Diagram 14**

**IGG019LI**

- 1 If the READ or WRITE operation for which the CHECK macro was issued has not yet completed, establishes a wait condition until the operation is completed. [WAIT]
- 2 Determines whether any errors have been posted in the DECB. If so, determines the type of error and then passes this information and control to the user's SYNAD routine, if there is one. (If the user returns control from the SYNAD

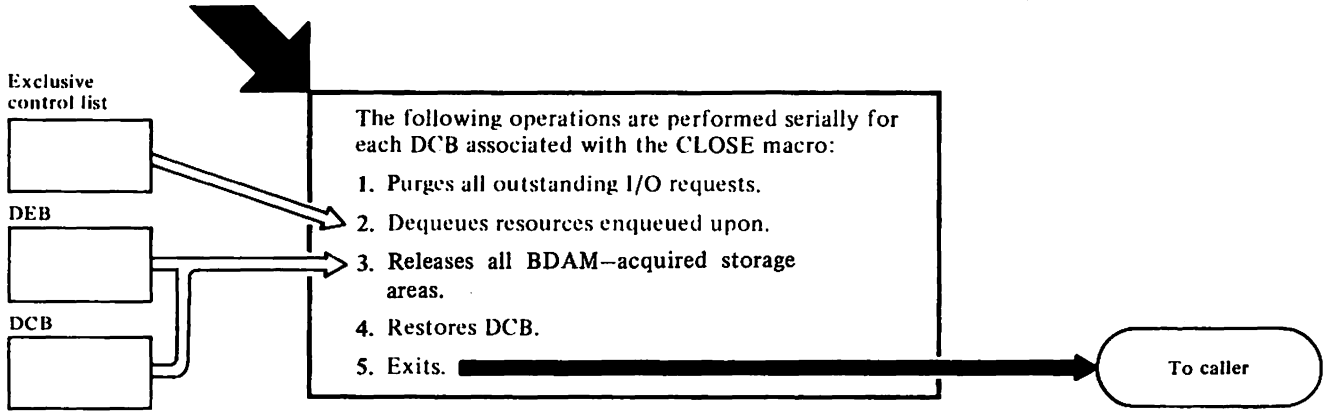
routine, processing continues at step 3 as though no error had been detected.) [CHECKERR]

If the user has not supplied a SYNAD routine, issues SVC 55, which gives control to the end-of-volume routine of O/C/EOV with a request for abnormal termination (ABEND 001). [SYNAD]

- 3 If no errors were posted in the DECB, or upon return from the user's SYNAD routine, marks the IOB available and returns control to the processing program. [MKIOBAVL]

DIAGRAM 15. CLOSE DATA SET

From Close routine of O/C/EOV or  
force close executor



Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM

Notes for Diagram 15

IGG0203A

This module will get control from the force close executor, module IGG020T1, if IGG020T1 determines that the user's DCB was updated from the DCB copy (which concludes the processing done by the BDAM open executor—see "Diagram 2. Open Data Set" on page 14 for further details). IGG020T1 assumes that, if open executor processing was completed, this close executor module can release, without error, all resources obtained during open executor processing. (See SAM Logic for further details about IGG020T1.)

- 1 Builds a purge list and then issues SVC 16, which removes from the I/O supervisor's queue of scheduled requests any requests that have been scheduled but whose channel programs have not yet completed. [START]
- 2 If an exclusive control list exists, issues a DEQ macro to release system control of each resource (which could be a block, a track, or a data set) for which there's an entry in the exclusive control list. [DEQRDX]
- 3 (The DEB and the DCB contain the addresses of either the control blocks being released or the control blocks that point to IOBs that should be released.)

Frees all available IOBs in the pool of IOBs. [IOBPOOL]

Frees the IOBs whose addresses are on the unposted queue associated with each exclusive control list entry. [REEQIOB]

If virtual storage was specified for the task, frees all IOBs whose addresses are in the unscheduled list(s). Then frees the unscheduled list(s). [GETIOB]

If real storage was specified for the task, frees all IOBs whose addresses are on the IOB buffer queue. [BCBQ]

Frees the BCB and all buffers. [FREEBCB]

If the record format was VS, frees the pool of segment work areas. [VRETST]

Frees the exclusive control list. [GOTONEXT]

- 4 Restores to their original status all DCB fields that BDAM has changed. [CLEANUP]

- 5 Scans the WTG table to determine whether any other BDAM DCBs associated with the CLOSE macro require the use of this module. If so, the module is reentered and does the required processing. If not, returns control to the Close routine.



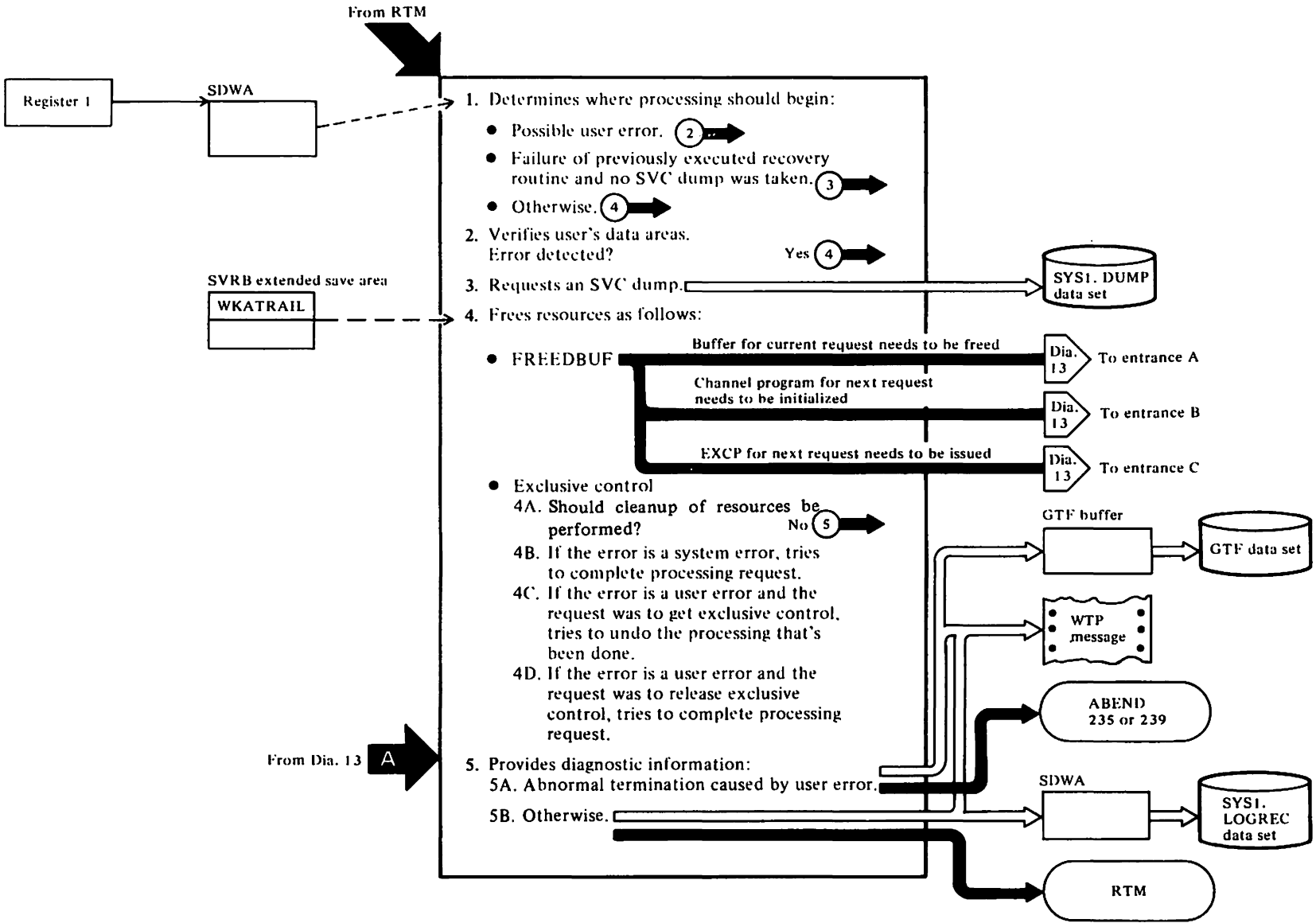


DIAGRAM 16. FREE SYSTEM RESOURCES AFTER ABNORMAL TERMINATION

## Notes for Diagram 16

The recovery/termination manager (R/TM) gets control only if abnormal termination occurs during processing associated with the FREEDBUF or exclusive control SVCs. Because the FREEDBUF and exclusive control recovery routines perform similar operations, they are treated together, where possible, in the discussion that follows.

Note that to understand the processing done by the cleanup routine in the recovery modules, you must be familiar with the operations performed by the dynamic buffering modules (see "Diagram 13. Free Buffer" on page 48) and the exclusive control module (see "Diagram 12 (Part 1 of 2). Get or Release Exclusive Control" on page 42).

### IGC0005G (exclusive control) or IGC0005G (FREEDBUF)

- 1 Before processing begins, determines whether R/TM passed a STAE diagnostic work area (SDWA) to this module. If not (R/TM may have been unable to get the storage for the SDWA), returns control to R/TM. If so, gives control to the contents supervisor routine IEAVVMSR to get the address of the FREEDBUF or exclusive control recovery routine. If IEAVVMSR finds the address of the recovery routine, gives control to it. Else returns control to R/TM. [ESTAERTN in IGC0005C; EXITRTN in IGC0005G. (These addresses were established when an ESTAE macro was issued earlier in the processing in these modules—see "Diagram 12 (Part 1 of 2). Get or Release Exclusive Control" on page 42 and "Diagram 13. Free Buffer" on page 48.)]

### IGCT005C (exclusive control) or IGCT005G (FREEDBUF)

Determines where processing should begin by analyzing error bits set in the SDWA. These error bits contain information about the cause and stage of abnormal termination. In most cases in which the system suspects a user error, a program check has occurred. [ANALYZE]

- 2 Branches to the system validity check routine IEAOVL01 or IEAOVL00 to validate ownership of storage and, where applicable, boundary alignment for the following control blocks and data areas:
  - FREEDBUF recovery routine—DCB, DECB, IOB, USL, and BCB.
  - Exclusive control recovery routine—DCB, DECB, IOB, DEB, blkref field, and next waiting IOB. [USERRTN]

If no error is detected during validity check processing, assumes the error that caused abnormal termination was a system, rather than a user, error and continues processing at step 3 below. [SYSERRTN] If an error is detected, continues processing at step 4 below. [ABEND]

- 3 Issues an SDUMP macro, which requests that an SVC dump be written to the SYS1.DUMP data set. [DUMPRTN or SYSERRTN]
- 4 Before freeing resources, issues an ESTAE macro so that, if an abnormal termination condition is encountered in this cleanup routine, R/TM will again give control to this recovery module (at label BACKUP5C in the exclusive control recovery module; at label BACKUP5G in the FREEDBUF recovery module). (The routines at these labels determine whether abnormal termination was caused by a program check. If so, they give control to the next sequential instruction after the branch to this cleanup routine, and processing continues. If abnormal termination was not the result of a program check, these routines request an SVC dump, issue a WTP message identifying the error, put diagnostic information in the SDWA, and return control to R/TM. R/TM will write the SDWA to SYS1.LOGREC.

### FREEDBUF recovery routine

Examines audit trail bits set in the WKATRAIL field in the SVRB extended save area to determine how much FREEDBUF processing had been completed before abnormal termination occurred. (These bits were set by one of the dynamic buffering modules—see "Diagram 13. Free Buffer" on page 48—as discrete operations associated with the FREEDBUF SVC were completed.) Exits to one of the dynamic buffering modules to try to finish processing associated with the FREEDBUF SVC. This is done so buffer queues will be cleared of entries associated with the request that caused the abnormal termination condition. [CLEANUP]

### IGCT105C

#### Exclusive control recovery routine

- 4A Determines whether cleanup operations should be performed. In the following situations, cleanup is not attempted and control returns to IGCT005C (step 5 below):
  - Abnormal termination occurred before the exclusive control list was modified. [CLEANUP]

- Either the DEB or the audit trail bit is invalid. [CLEANUP]
  - No processing was done for the request.
  - All processing on the request is completed, and the request is not a WRITE-add request to release exclusive control of a resource for which another request is waiting to get exclusive control of. [CLEANUP]
  - For requests to get exclusive control:
    - The IOB for the request was put on the unposted queue, and the error is not a user error.
    - An ENQ macro was issued to put an entry for the request on the system list, and the system has determined the error is a user error. [CHKENQ]
  - For requests to release exclusive control:
    - An IOB was waiting on the unposted queue for the released resource. The error is a user error. [CHKENQ]
    - The IOB for the waiting request was dequeued. The error is a user error. [CHKENQ]
    - The DEQ macro was issued to remove from the system queue the entry for the current request. The error is a user error. [CHKENQ]
    - The entry in the exclusive control list was cleared successfully. [CHKENQ]
    - An EXCP macro was issued for a waiting IOB. This IOB was added to the unposted queue while the DEQ SVC was being executed.
    - A POST macro has been issued for the waiting request or, if the current request was a WRITE-add, the current request.
- 4C** If abnormal termination was caused by a user error and the request was to get exclusive control, tries to undo whatever processing has been done.
- 4D** If abnormal termination was caused by a user error and the request was to release exclusive control, completes processing for the request in the following cases:
- An IOB, added to the unposted queue during processing of the DEQ SVC, has been removed from the queue. In this case, an ENQ and an EXCP macro are issued for this request (or just an EXCP if the ENQ macro has already been issued).
  - Feedback has been gotten for the request. In this case, the request is posted.
- IGCT005C (exclusive control) or IGCT005G (FREEDBUF)**
- 5A** If abnormal termination was caused by a user error (an invalid control block was found), builds a GTRACE work area and moves to the work area all verified control blocks chained to each other up to the control block containing the address of the block in error. Then issues the GTRACE macro, which gives control to GTF to write this information to the GTF data set. (The format and contents of the GTRACE work area are shown under "Diagnostic Aids.") [GTRACE]
- Issues a WTP message identifying the error as a user error. The WTP message contains a return code identifying the block in error, and an abend code. [WTP]
- Removes the address of this recovery routine from the queue maintained by R/TM so control can no longer return to this recovery routine. Sets return codes and issues an ABEND. [RETRYRTN]
- 5B** If, in step 1, it was determined that a previous recovery routine failed, issues a WTP message describing the error as a system error. [SYSERRTN]
- Puts the following information in the SDWA: [RECORD]
- Audit trail bits.
  - Return codes indicating the status of the attempts to clean up resources and write information to the GTF data set.
- 4B** If abnormal termination was caused by a system error, attempts to complete whatever processing was requested and, in the case of requests to release exclusive control, to start processing any requests found waiting on the unposted queue.

**Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM**

- Registers 0 and 1 at the time the FREEDBUF or exclusive control SVCs were issued.
- The SVC old PSW, which contains the address +2 from which the SVC was issued.

Returns control to R/TM, which will write the information in the SDWA to the SYS1.LOGREC data set and continue ABEND processing. (The format and contents of the SDWA are shown under "Diagnostic Aids.")

MODULE DIRECTORY

Microfiche Name of Module	Function/Description	Record Formats Processed by This Module	Method of Operation Diagram	Program Organization Diagram
IGCT005C	Exclusive control recovery	all	16	L
IGCT005G	FREEDBUF recovery	all	16	K
IGCT105C	Exclusive control recovery	all	14	L
IGC0005C	Exclusive control SVC	all	12	J
IGC0005G	FREEDBUF SVC	all	13	G
IGG019JA	Dynamic buffering; start I/O appendage (virtual storage)	F,U,V,VS	7,13	E
IGG019KA	Foundation	F,U,V	4,8	A (base routine), F (ASI routine)
IGG019KC	Relative track address conversion and feedback	all	5	B
IGG019KE	Relative block address conversion—no track overflow	F	5	B
IGG019KF	Relative block address conversion—track overflow	F	5	B
IGG019KG	Relative block feedback—no track overflow	F	5	G
IGG019KH	Relative block feedback—track overflow	F	5	G
IGG019KI	Update channel program—search on key	F,U,V	6	C
IGG019KJ	Foundation	VS	4,10	A (base routine), H (ASI routine)
IGG019KK	Update channel program—search on block identification	F,U,V,VS	6	C
IGG019KM	WRITE-add channel program	U or V	9	F
IGG019KN	WRITE-add channel program	VS	11	I
IGG019KO	WRITE-add channel program	F	6	C

Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM

Microfiche Name of Module	Function/Description	Record Formats Processed by This Module	Method of Operation Diagram	Program Organization Diagram
IGG019KQ	Write verify	all	6	C
IGG019KR	Update channel program—search on key or block identification	VS	6	C
IGG019KU	Channel end/abnormal end appendage	all	7	D
IGG019KW	Extended search for update	all	6	C
IGG019KY	Extended search for WRITE-add	U,V,VS	9,11	G
IGG019LA	Extended search for WRITE-add	F	6	C
IGG019LC	End-of-extent appendage	all	7	D
IGG019LE	Dynamic buffering; start I/O appendage (real storage)	F,V,U,VS	7,13	E
IGG019LI	Check	all	14	M
IGG0193A	Open executor	all	2	A
IGG0203A	Close executor	all	15	M

DATA AREAS

BCB (BUFFER CONTROL BLOCK)

A BCB is built if the dynamic buffering option has been specified, or if spanned records are being processed. Module IGG0193A obtains a continuous area of virtual storage for both the BCB and the required number of buffers. The BCB is initialized by module IGG0193A, but subsequent entries are placed in it by the dynamic buffering module. The storage area for both the BCB and the buffers is released by the BDAM close executor module.

**BCB FOR SPANNED RECORDS—NO DYNAMIC BUFFERING (DSECT NAME: BCBDEFS)**

0(0)	BCBNXBUF	
4(4)	BCBBFNUM	6(6) BCBBFLG

Offset	Field	Field Size (in Bytes)	Field Contents and Comments
0(0)	BCBNXBUF	4	The first byte is a flag field, set as follows:  X'FF' SWA is being used X'00' SWA is available  The last three bytes contain the address of the next available SWA.
4(4)	BCBBFNUM	2	Number of SWAs.
4( )	BCBBFLG	2	SWA length.

BCB FOR DYNAMIC BUFFERING—REAL STORAGE SPECIFIED (DSECT NAME: BCBDEFB)

0(0)	BCBFRQT
4(4)	BCBFRQB
8(8)	BCBNABFR
12(C)	BCBTBRS

Offset	Field	Field Size (in Bytes)	Field Contents and Comments
0(0)	BCBFRQT	4	Address of the first IOB waiting to be assigned a buffer from the buffer queue. The dynamic buffering module inserts this address.
4(4)	BCBFRQB	4	Address of the last IOB waiting to be assigned a buffer from the buffer queue. The dynamic buffering module inserts this address.
8(8)	BCBNABFR	4	Address of the next buffer available for assignment to an IOB. Initially, module IGG0193A inserts this address. Subsequent addresses are inserted by the dynamic buffering module.
12(C)	BCBTBRS	4	The total size, in bytes, of the buffer pool and the BCB. Module IGG0193A inserts this value.



BCB FOR DYNAMIC BUFFERING—VIRTUAL STORAGE SPECIFIED (DSECT NAME: BCBDEFV)

0(0)		BCBBUF1
4(4)		BCBBUFL
8(8)	BCBBFAVL	9(9) BCBRAOFS
12(C)		BCBTBRS1
16(10)	BCBSUFFIX	

Offset	Field	Field Size (in Bytes)	Field Contents and Comments
0(0)	BCBBUF1	4	Address of the first buffer in the buffer pool.
4(4)	BCBBUFL	4	Length, in bytes, of each buffer.
8(8)	BCBBFAVL	1	Buffer number of the first available buffer.
9(9)	BCBRAOFS	3	For variable-length spanned blocks, the offset to the record area in the buffer.
12(C)	BCBTBRS1	4	Size, in bytes, of the buffer pool, the buffer control block (including the BCB suffix), and the first copy of the unscheduled list.
16(10)	BCBSUFFIX	1	Buffer number of the next available buffer in the buffer pool. This field is built once for every buffer in the pool, thereby creating a chain of available buffers. The BCBSUFFIX field for the last available buffer contains 0.

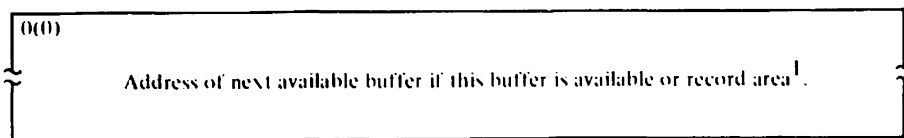
**BUFFER POOLS**

A buffer pool is a continuous area of virtual storage divided into buffers. BDAM uses three kinds of buffer pools:

- A pool of segment work areas for processing spanned records without dynamic buffering
- A pool of segment work areas and record areas for processing spanned records with dynamic buffering
- A pool of record areas for dynamic buffering of nonspanned records

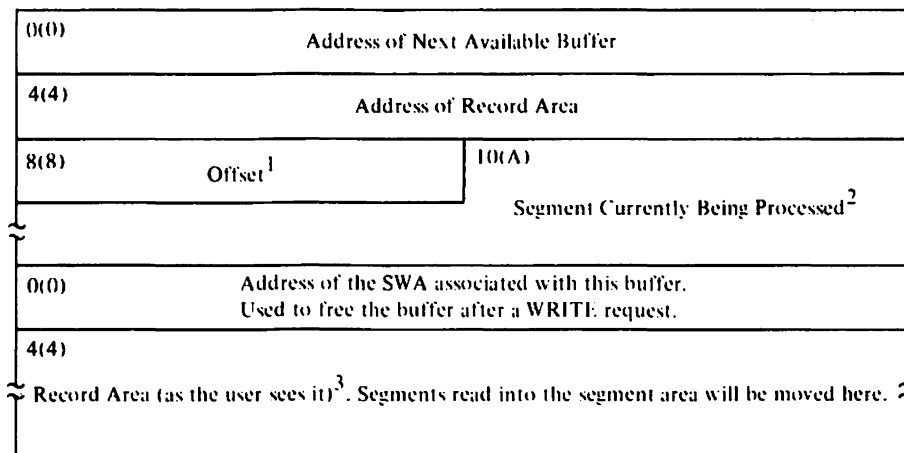
A buffer pool follows its buffer control block in virtual storage. The dynamic buffering pools use the BCB with the DSECT name BCBDEFR (real storage specified) or BCBDEFV (virtual storage specified). The segment work area pool uses the BCB with the DSECT name BCBDEFS.

**BUFFER FOR DYNAMIC BUFFERING OF NONSPANNED RECORDS**



- <sup>1</sup> If it contains the address of the next available buffer, this field is 4 bytes long. If it is the record area, its length is DCBBLKSI + DCBKEYLE, with the length, if necessary, rounded to the next multiple of 8.

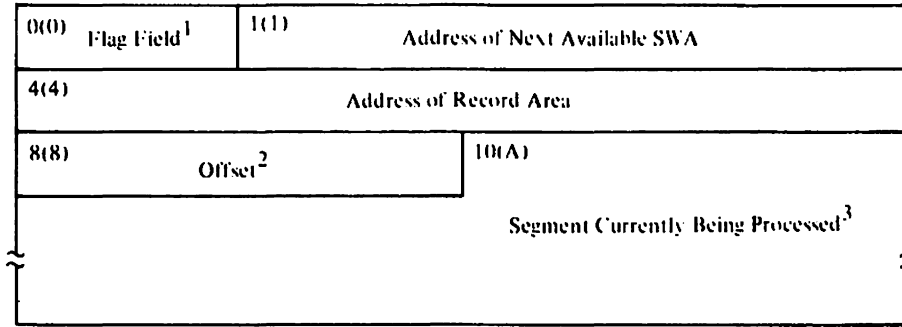
**BUFFER FOR DYNAMIC BUFFERING OF SPANNED RECORDS**



- <sup>1</sup> When this offset is added to the address of the record area (above), the sum is the address of the next segment to be processed.
- <sup>2</sup> Length of segment = smaller of: track size and DCBBLKSI + DCBKEYLE. The end of this field is padded with up to 7 bytes, if necessary, for alignment of the next field.
- <sup>3</sup> Length of record area = DCBBLKSI + DCBKEYLE. The end of this field is padded, if necessary, to round the buffer length to a multiple of 8.

**Note:** The first three fields (bytes 0 to 10 in decimal) compose the segment work area.

**BUFFER FOR SIMPLE BUFFERING OF SPANNED RECORDS**



<sup>1</sup> X'FF' = buffer in use  
X'00' = buffer available

<sup>2</sup> When this offset is added to the address of the record area (above), the sum is the address of the next segment to be processed.

<sup>3</sup> Length of segment = smaller of: track size and DCBBLKSI + DCBKEYLE

**Note:** The first four fields of the buffer (bytes 0 to 10 in decimal) compose the segment work area.

**Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM**

**DCB (DATA CONTROL BLOCK)**

The DCB contains information about the current use of a data set. The DCB that follows shows the fields especially important in BDAM applications. A more complete description of the DCB is in OS/VS2 Data Areas.

16(10)	DCBKEYLE	17(11)	DCBRFL	} Device interface
20(14)	DCBBUFNO	21(15)	DCBBUFCB	
24(18)	DCBBUFL		26(1A) DCBDSORG	} Common interface
28(1C)	DCBIOBAD			
32(20)	DCBFODAD		DCBFODA	} Foundation extension
	DCBBFTEK	33(21)		
36(24)	DCBEXLST		DCBEXLSA	} Foundation after open
	DCBRECTM	37(25)		
40(28)	DCBTIOT		42(2A) DCBMACRF	} Foundation after open
44(2C)	DCBDFBAD		DCBDFBA	
	DCBDFLGS	45(2D)		
48(30)	DCBOFLGS	49(31)	DCBREAD or DCBWRITE	} Foundation after open
52(34)	DCBCHK		DCBCKCA	
	DCBOPTCD	53(35)		
56(38)	DCBSYNAD			} Foundation after open
60(3C)	Reserved		62(3E) DCBBLKSI	
64(40)	DCBIOBSQ			} BDAM interface
68(44)	DCBSQND			
72(48)	DCBIOBUQ			
76(4C)	DCBUQND			
80(50)	Reserved	81(51)	DCBLIMCT	
84(54)	DCBXARG		DCBXARGA	
	DCBXCNT	85(55)		
88(58)	DCBDRDX		DCBDRDXA	
	DCBMVXNO	89(59)		
92(5C)	DCBDFOR			
96(60)	DCBDFBK			
100(64)	DCBDYNB			

Offset	Field	Field Size (in Bytes)	Field Contents and Comments
16(10)	DCBKEYLE	1	Length of key field for each block in the data set.
17(11)	DCBREL	3	Number of relative tracks or blocks in the data set. This number is placed in the DCBREL field by BDAM open executor IGG0193A, and it can be used by the processing program in converting a relative address.
10(14)	DCBBUFNO	1	Number of buffers required for this data set. May range from 0 to 255. If format VS records are used, the number of segment work areas required for this data set.
21(15)	DCBBUFCB	3	Address of buffer pool control block or of dynamic buffer pool control block.
24(18)	DCBBUFL	2	Length of buffer. May range from 0 to 32,767.
26(1A)	DCBDSORG	2	Data set organization being used.
			<b>Byte 1      Code</b>
			..1. .... DA    Direct organization.
			.... ..1 U    Unmovable (the data contains location dependent information).
			<b>Byte 2              Reserved.</b>
28(1C)	DCBIOBAD	4	Address of the standard fields of the first IOB in the pool of IOBS.
	DCBBFTEK	1	Buffering technique.
			<b>Code</b>
			..1. .... R    Format VS records are being used. The open executor forms a segment work area pool unless the user has elected to do so. The number of segment work areas is determined by DCBBUFNO (offset 20). The open executor stores the address of the segment work area control block in DCBDYNB (offset 100) if dynamic buffering is not used, or in the dynamic buffer pool control block (see DCBBUFCB, offset 21) if dynamic buffering is used. WRITE requests use a segment work area to write a record as one or more segments. READ requests use a segment work area to read a record that was written as one or more segments.

Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM

Offset	Field	Field Size (in Bytes)	Field Contents and Comments
33(21)	DCBEODA	3	Address of a user-provided routine to handle end-of-data conditions.
36(24)	DCBEXLST	4	Address of a user-provided exit list.
36(24)	DCBRECFM	1	Record format.
<b>Code</b>			
	10.. ....	F	Fixed record length.
	01.. ....	V	Variable record length.
	11.. ....	U	Undefined record length.
	..1. ....	T	Track overflow.
	...1 ....	B	Blocked (allowed only with VS).
	.... 1...	S	Spanned (allowed only with VS).
	.... .00.		Always zeros.
	.... ...1		Key length (KEYLEN) was specified in the DCB macro instruction. This bit is inspected by the Open routine of O/C/EOV to prevent overriding a specification of KEYLEN=0 by a nonzero specification in the JFCB or data set label.
37(25)	DCBEXLSA	3	Address of a user-provided exit list.
40(28)	DCBTIOT	2	Offset from the TIOT origin to the TIOELNGH field in the TIOT entry for the DD statement associated with this DCB.
42(2A)	DCBMACRF	2	Major macro instructions and various options associated with them that will be used.
<b>Byte 1 Code</b>			
	00.. ....		Always zero for BDAM.
	..1. ....	R	READ
	...1 ....	K	Key segment with READ.
	...1 ....	I	ID argument with READ.
	.... .1..	S	System provides area for READ (dynamic buffering).
	.... ..1.	X	Read exclusive.
	.... ...1	C	CHECK macro instruction.
<b>Byte 2 Code</b>			
	00.. ....		Always zero for BDAM.
	..1. ....	W	WRITE
	...1 ....	K	Key segment with WRITE.
	.... 1...	I	ID argument with WRITE.
	.... .x..		Reserved.
	.... ..1.	A	Add type of WRITE.
	.... ...1		Unblocked spanned records, with BFTEK=R specified and no dynamic buffering. The user's program has provided a segment work area pool and stored the address of the segment work area control block in DCBDYNB (offset 100).

**Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM**

Offset	Field	Field Size (in Bytes)	Field Contents and Comments
44(2C)	DCBDEBAD	4	Address of the associated DEB.
44(2C)	DCBIFLGS	1	Used by I/O supervisor in communicating error conditions and in determining corrective procedures.  00.. .... Not-in-error procedure. 01.. .... Error correction in process. 11.. .... Permanent error condition. ..00 .... Always zeros. .... 00.. Always use I/O supervisor error routine. .... 11.. Never use I/O supervisor error routine. .... 10.. Never use I/O supervisor error routine. .... 01.. Never use I/O supervisor error routine. .... ..XX (Reserved bits)
45(2D)	DCBDEBA	3	Address of the associated DEB.
48(30)	DCBOFLGS	1	Flags used by the Open routine at O/C/EOV.  00.. .... Not-in-error procedure. 1... .... The data set is being opened for INPUT or OUTPUT. .x.. .x.. (Reserved bits) ..0. 0... Always set to 0. ...1 .... The OPEN request has been successfully completed. Set to 0 by an O/C/EOV function when that function takes a user exit. It is set to 0 to inhibit other O/C/EOV functions from processing this particular DCB. .... ..1. Set to 1 on return from user exit to the O/C/EOV function that took the exit. .... ...1 Set to 1 by an O/C/EOV function if the DCB is to be processed by that function.
49(31)	DCBREAD or DCBWRITE	3	Address of the BDAM foundation module IGG019KA or IGG019KJ.
52(34)	DCBCHECK	4	Address of the check module.

Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM

Offset	Field	Field Size (in Bytes)	Field Contents and Comments																								
52(34)	DCBOPTCD	1	Indication of options specified for the data set. Bits of the DCBOPTCD field and their interpretations for BDAM are as follows (when the bit is set to 1, the interpretation is in effect; when set to 0, the interpretation is not in effect):  <div style="text-align: center;">Code</div> <table style="margin-left: 40px;"> <tr> <td>1... ..</td> <td>W</td> <td>Write-validity-check option has been specified.</td> </tr> <tr> <td>.1.. ..</td> <td></td> <td>Track overflow has been specified. (This bit is set by BDAM.)</td> </tr> <tr> <td>..1. ....</td> <td>E</td> <td>Extended search has been specified.</td> </tr> <tr> <td>...1 ....</td> <td>F</td> <td>Feedback has been specified.</td> </tr> <tr> <td>.... 1...</td> <td>A</td> <td>Actual addressing has been specified.</td> </tr> <tr> <td>.... .1..</td> <td></td> <td>Dynamic buffering has been specified. This bit is set by BDAM.)</td> </tr> <tr> <td>.... ..1.</td> <td></td> <td>Exclusive control has been specified. (This bit is set by BDAM.)</td> </tr> <tr> <td>.... ...1</td> <td>R</td> <td>Relative block addressing has been specified.</td> </tr> </table> <p><b>Note:</b> If neither actual addressing nor relative block addressing has been specified (that is, if bits 4 and 7 are both 0), relative track addressing is assumed.</p>	1... ..	W	Write-validity-check option has been specified.	.1.. ..		Track overflow has been specified. (This bit is set by BDAM.)	..1. ....	E	Extended search has been specified.	...1 ....	F	Feedback has been specified.	.... 1...	A	Actual addressing has been specified.	.... .1..		Dynamic buffering has been specified. This bit is set by BDAM.)	.... ..1.		Exclusive control has been specified. (This bit is set by BDAM.)	.... ...1	R	Relative block addressing has been specified.
1... ..	W	Write-validity-check option has been specified.																									
.1.. ..		Track overflow has been specified. (This bit is set by BDAM.)																									
..1. ....	E	Extended search has been specified.																									
...1 ....	F	Feedback has been specified.																									
.... 1...	A	Actual addressing has been specified.																									
.... .1..		Dynamic buffering has been specified. This bit is set by BDAM.)																									
.... ..1.		Exclusive control has been specified. (This bit is set by BDAM.)																									
.... ...1	R	Relative block addressing has been specified.																									
53(35)	DCBCHCKA	3	Address of the check module, IGG019LI.																								
56(38)	DCBSYNAD	4	Address of user's SYNAD routine.																								
60(3C)		2	Reserved.																								
62(3E)	DCBBLKSI	2	Maximum size of a record block in the data set.																								
64(40)	DCBIOBSQ	4	Address of first IOB on unscheduled queue for either a WRITE-add request when another WRITE-add is in progress, or a READ-exclusive request when the READ-exclusive list is full.																								
68(44)	DCBSQND	4	Address of the last IOB on the unscheduled queue.																								
72(48)	DCBIOBUQ	4	Address of the first IOB on the unposted queue.																								
76(4C)	DCBUQND	4	Address of the last IOB on the unposted queue that is maintained by the READ-exclusive module.																								
80(50)		1	Reserved.																								
82(51)	DCBLIMCT	3	Number of tracks (for relative track addressing) or number of blocks (for relative block addressing) to be searched when extended search option is specified.																								
84(54)	DCBXARG	4	Address of READ-exclusive list.																								
84(54)	DCBXCNT	1	Number of entries in READ-exclusive list.																								



Offset	Field	Field Size (in Bytes)	Field Contents and Comments
85(55)	DCBXARGA	3	Address of READ-exclusive list.
88(58)	DCBDRDX	4	Address of READ-exclusive module.
88(58)	DCBMVXNO	1	Total number of extents in a multivolume data set.
89(59)	DCBDRDXA	3	Address of READ-exclusive module.
92(5C)	DCBDFOR	4	Address of the format channel program generating module (IGG019K0, IGG019KM, or IGG019KN) required for the block format indicated in the DCB macro instruction. This address is placed in DCBDFOR by the BDAM open executor IGG0193A.
96(60)	DCBDFBK	4	Address of the feedback module, IGG019KG or IGG019KH (if relative block feedback was specified).
100(64)	DCBDYNB	4	If dynamic buffering was specified, and virtual storage is specified for the task, address of the current copy of the unscheduled list; otherwise, unused.

The initial value of each DCB field containing an address is 00 00 00 01. When the data set is opened, the value of each of these fields that corresponds to a required BDAM module is changed to the storage address of the module; the values of address fields corresponding to modules that are not required remain at 00 00 00 01.

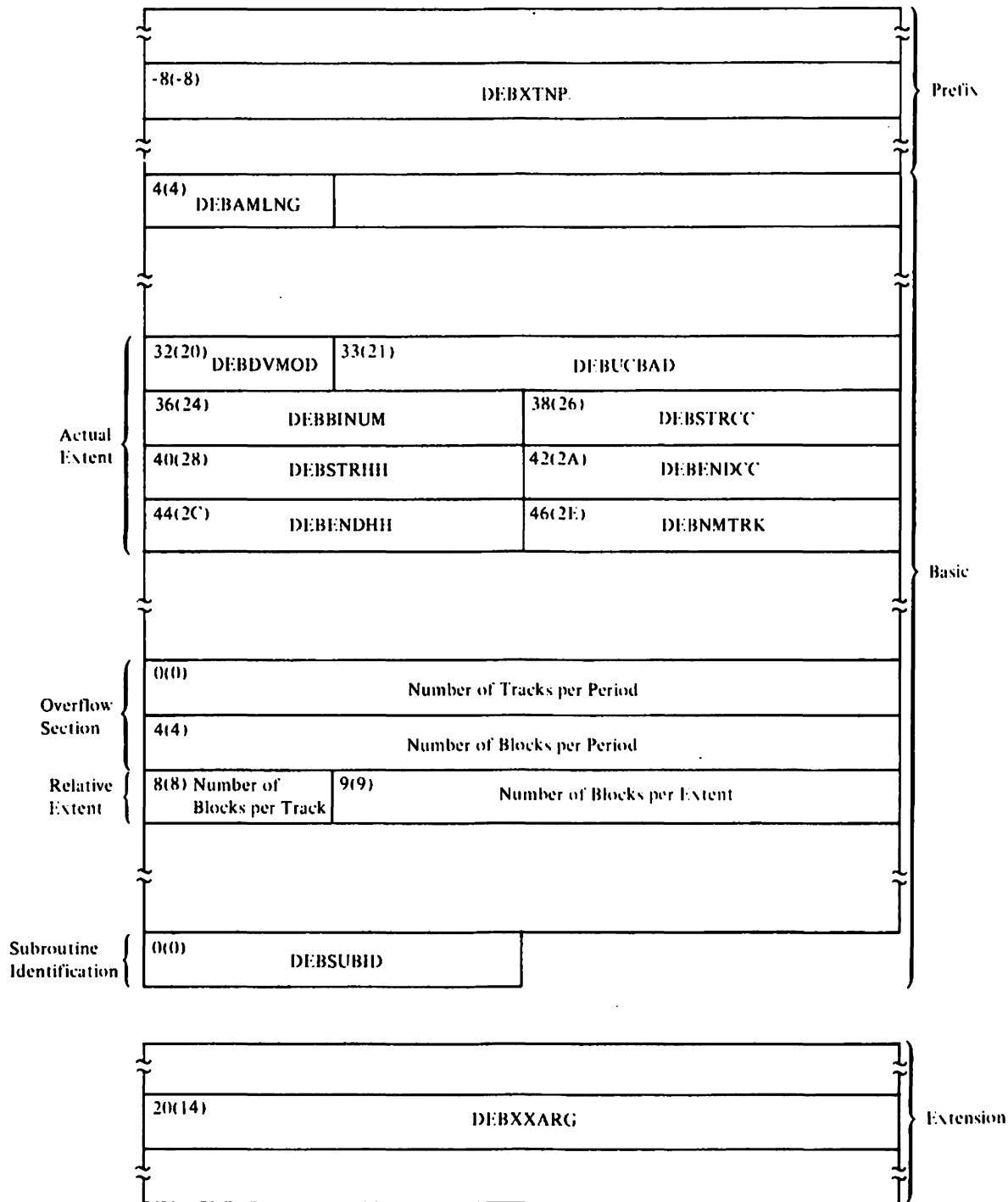
#### DEB (DATA EXTENT BLOCK)

The DEB, which is built by module IGG0193A, contains information about the physical characteristics of a data set. The DEB that follows shows the fields especially important in BDAM applications. A more complete description of the DEB is in Data Areas.

The relative extents in the DEB are built only when relative block addressing has been specified. There is one relative extent for each actual extent in the DEB.

If track overflow has not been specified, each relative extent consists of a 1-byte field that contains the number of blocks on a track (for the device used) and a 3-byte field that contains the number of blocks in the extent. The latter value is obtained by multiplying the number of tracks in the extent (given as the value in the last 2 bytes of the associated actual extent) by the number of blocks on a track.

If track overflow has been specified, each relative extent consists of only a 3-byte blocks-per-extent field. The byte preceding each blocks-per-extent field is unused. In addition, two 1-word fields constituting an overflow section are inserted between the last actual extent and the first relative extent. The values in these fields are based on the size of the period that is calculated by module IGG0193A.



Offset	Field	Field Size (in Bytes)	Field Contents and Comments
8(8)	DEBXTNP	4	Address of the DEB extension.

Offset	Field	Field Size (in Bytes)	Field Contents and Comments
4(4)	DEBAMNG	1	Number of words of virtual storage used to contain the relative extents.
32(20)	DEBDVMOD	1	Device modifier: file mask.
33(21)	DEBUCBAD	3	Address of UCB associated with this extent.
36(24)	DEBBINUMnn	2	Bin number.
38(26)	DEBSTRCC	2	Cylinder address for the start of an extent.
40(28)	DEBSTRHH	2	Read/write track address for the start of an extent.
42(2A)	DEBENDCC	2	Cylinder address for the end of an extent.
44(2C)	DEBENDHH	2	Read/write track address for the end of an extent.
46(2E)	DEBNMTRK	2	Number of tracks in the actual extent.
0(0)	DEBSUBID repeated for each routine name	2	Each access method module has a unique 8-byte name. The low-order 2 bytes of each module name are put in this field if the module is loaded into virtual storage by the open executor module.
20(14)	DEBXXARG	4	Address of the exclusive control list.

DECB (DATA EVENT CONTROL BLOCK)

The DECB, which results from the expansion of either a READ or a WRITE macro instruction, contains information about the I/O operation requested by the macro instruction. The following DECB contains the fields used by BDAM.

0(0)	DECSDECB	
4(4)	DECTYPE	6(6) DECLNGTH
8(8)	DECDCBAD	
12(C)	DECAREA	
16(10)	DECIOBPT	
20(14)	DECKYADR	
24(18)	DECRCPT	
28(1C)	DECNA	

Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM

Offset	Field	Field Size (in Bytes)	Field Contents and Comments														
0(0)	DECSDECB	4	Standard event control block (ECB). (Refer to the IOBDSTAT field of the IOB.)														
4(4)	DECTYPE	2	Type of request operation. The contents of this field are described in the discussion of the IOBDTYPE field.														
6(6)	DECLNGTH	2	Length of data portion of the block being processed.														
8(8)	DECDCBAD	4	Address of the DCB to which a request is related.														
12(C)	DECAREA	4	Address of area into which the data portion of a block is to be written, or from which it is to be read.														
16(10)	DECIOBPT	4	Address of the IOB associated with this DECB.														
20(14)	DECKYADR	4	The contents of this field vary depending on the type of request the DECB refers to. If there is no key, or if key is not to be read, written, or used as a search argument, this field is 0.														
			<table border="1"> <thead> <tr> <th>Type of Request</th> <th>DECKYADR Contents</th> </tr> </thead> <tbody> <tr> <td>Write by ID</td> <td>Address of the key to be written.</td> </tr> <tr> <td>Write-Add</td> <td>Address of the key to be written.</td> </tr> <tr> <td>Read by ID</td> <td>Address of the area into which the key is to be read.</td> </tr> <tr> <td>Read by Key</td> <td>Address of the key to be used as a search argument.</td> </tr> <tr> <td>Write by Key</td> <td>Address of the key to be used as a search argument.</td> </tr> <tr> <td>Write-Add</td> <td>Key to be written to replace the dummy (Format-F) key. (Searching is done on the hexadecimal 'FF' that is in the first byte of the key field of the dummy record.)</td> </tr> </tbody> </table>	Type of Request	DECKYADR Contents	Write by ID	Address of the key to be written.	Write-Add	Address of the key to be written.	Read by ID	Address of the area into which the key is to be read.	Read by Key	Address of the key to be used as a search argument.	Write by Key	Address of the key to be used as a search argument.	Write-Add	Key to be written to replace the dummy (Format-F) key. (Searching is done on the hexadecimal 'FF' that is in the first byte of the key field of the dummy record.)
Type of Request	DECKYADR Contents																
Write by ID	Address of the key to be written.																
Write-Add	Address of the key to be written.																
Read by ID	Address of the area into which the key is to be read.																
Read by Key	Address of the key to be used as a search argument.																
Write by Key	Address of the key to be used as a search argument.																
Write-Add	Key to be written to replace the dummy (Format-F) key. (Searching is done on the hexadecimal 'FF' that is in the first byte of the key field of the dummy record.)																
24(18)	DECRCPT	4	Address of the blkref field.														

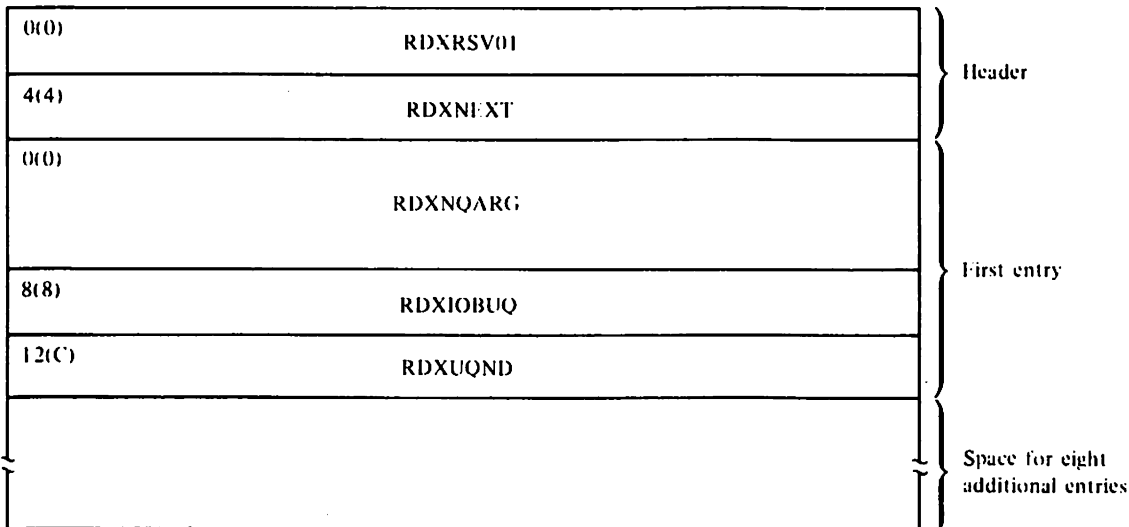
Offset	Field	Field Size (in Bytes)	Field Contents and Comments
28(1C)	DECNA	4	Address of the next address feedback field (applicable for format VS records only). This field is present only if "R" or "RU" is coded in the READ macro.

**EXCLUSIVE CONTROL LIST (DSECT NAME: RDXLIST)**

The exclusive control list is an area of storage containing the addresses of resources under exclusive control, and the addresses of requests that are waiting to get exclusive control of a resource. The list is composed of one or more 152-byte segments. Each segment has space for nine 16-byte entries, each entry identifying a resource under exclusive control and, if any requests are waiting to get exclusive control of that resource, the addresses of the first and last IOBs representing those requests.

IGG0193A obtains storage for the first 152-byte segment; if additional segments are required, the exclusive control module, IGC0005C, obtains storage for them. The close executor module releases the storage obtained for the exclusive control list segments.

The address of the exclusive control list is in the DEBXXARG field in the DEB extension at offset X'14'. The DEBXTNP field in the DEB prefix (offset X'8') contains the address of the DEB extension.



Offset	Field	Field Size (in Bytes)	Field Contents and Comments
0(0)	RDXRSV01	4	Unused.
4(4)	RDNEXT	4	Address of the next segment of the exclusive control list, if there's more than one segment.

Offset	Field	Field Size (in Bytes)	Field Contents and Comments
0(0)	RDXNQARG	8	The first three bytes of this field contain the UCB address of the resource under control. The last five bytes contain the CCHHR address of that resource.
8(8)	RDXIOBUQ	4	The address of the first IOB for a request waiting to get exclusive control of the resource whose address is in the RDXNQARG field.
12(C)	RDXUQND	4	The address of the last IOB for a request waiting to get exclusive control of the resource whose address is in the RDXNQARG field. (Intermediate IOBs are chained together via the IOBDQPTR field in the IOB.)

### IOB (INPUT/OUTPUT BLOCK)

The IOB contains information required by the I/O supervisor to perform an input/output operation. In BDAM, the IOB is built by the base routine in the foundation module (IGG019KA or IGG019KJ).

The fields of the IOB are constructed as processing takes place. The storage area used for the IOB is obtained either from a pool of available IOBs for which storage has been previously obtained, or by use of the GETMAIN routine. If the area is taken from a pool of IOBs, that area is made unavailable to the pool until the request associated with the IOB is completed.

When a request is completed, the IOB is either replaced in the pool or assigned to the pool for the first time, depending on how it was obtained. If the IOB was obtained from the pool, the availability byte in the IOB is set to 0, indicating the IOB has been returned to its former position in the pool. If the IOB was obtained by the GETMAIN routine, it is placed in the pool (placement is by size), the next IOB pointers are updated as necessary, and the availability byte is set to 0, unless use of the CHECK macro is indicated in the DCBMACRF field. If it is, the check module (IGG019LI) sets the availability byte to 0.

All storage areas assigned to IOBs are released to the system by the FREEMAIN routine when the data set is closed. If the first use of an IOB storage area occurs with an invalid request, the area is returned to the system by the FREEMAIN routine rather than being placed in the pool when the request is completed.

There are four main sections to the IOB as used by BDAM. The first part of the IOB, an 8-byte prefix, contains information applicable only to spanned records. It is present only if BFTEK=R and RECFM=VS.

The second part is a standard 40-byte section and is described in the Data Areas microfiche. BDAM refers to this part, for example, to determine the status of a completed channel program and to locate addresses of storage areas and control blocks.

The third part of the IOB is a 40-byte section that contains information needed by BDAM to process a request. The 11 fields in this part are described below.

The fourth part contains the channel program constructed for the input or output request. The channel command words are placed in this part of the IOB as they are formed.

**Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM**

Various BDAM routines use fields in the IOB as temporary work areas until such time as these fields are filled in with the information shown in the IOB that follows.

-8(-8)	IOBDEQIN	-7(-7)	IOBDQADB	IOB Prefix
-4(-4)	IOBSWAP			
				Standard IOB
40(28)	IOBDYTR	42(2A)	IOBDIOBS	
44(2C)	IOBDAVLI	45(2D)	IOBDPLAD	BDAM Extension to IOB
48(30)	IOBDTYPE	50(32)	IOBDSTAT	
52(34)	IOBDCPND			
56(38)	IOBDYTN	58(3A)	Reserved	
60(3C)	IOBDQPTR			
64(40)	IOBUPLIM			
72(48)	IOBDNRCE			
80(50)	Channel Program			

Offset	Field	Field Size (in Bytes)	Field Contents and Comments
-8(8)	IOBDEQIN	1	Bit 0: Indicates that a track containing spanned records is being dequeued.  Bit 1: Indicates that an RPS device is being used.  Bits 2-7: Reserved for future use.
-7(7)	IOBDQADB	3	Address of the IOB waiting to dequeue the tracks occupied by a spanned record.
-4(4)	IOBSWAP	4	Address of the segment work area.

Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM

Offset	Field	Field Size (in Bytes)	Field Contents and Comments
40(28)	IOBDBYTR	2	Number of unused bytes remaining on a track on which a new variable- or undefined-length block is to be written. This value is initially placed in IOBDBYTR when the channel program reads the capacity record. Subsequent updating of the IOBDBYTR field is done by the self-format module, IGG019KM or IGG019KN. The channel program later updates the capacity record before the I/O operation is completed.
42(2A)	IOBDIOBS	2	Overall size of the IOB, in bytes. The base routine of the foundation module places this value in IOBDIOBS after the virtual-storage area for an IOB has been obtained.
44(2C)	IOBDAVLI	1	Indication of the availability of the IOB. When the IOB is taken from the pool of available IOBs, the value of IOBDAVLI is set to hexadecimal 'FF' to indicate that the IOB is being used (is unavailable). This is done by the base routine of the foundation module. When an I/O operation is posted as complete, and an IOB is either returned to or placed in the pool of available IOBs, the value of IOBDAVLI is set to 0. Depending on the cause of the completion of the I/O operation, the 0 value is set by the asynchronous interrupt routine, the base routine of the foundation module, or the check module.
45(2D)	IOBDPLAD	3	Address of the next IOB area in the pool of IOBs attached to the current DCB. If there are no more IOBs in the pool, the value of this field is 0. Each IOB in the pool became a member of the pool after the first use of the IOB. When a new IOB is added to the pool, the IOBDPLAD field of the preceding IOB is updated.



Offset	Field	Field Size (in Bytes)	Field Contents and Comments
48(30)	IOBDTYPE	2	<p>Indication of the type of request and the options specified in the DECB related to the request. The contents of the DECTYPE field (see DECB) are placed in the IOBDTYPE field when the IOB is initialized. Significant bits of the IOBDTYPE field and their interpretations for BDAM are as follows (when the bit is set to 1, the interpretation is in effect; when set to 0, the interpretation is not in effect):</p> <p><b>First Byte:</b></p> <p>Bit 0: Verification of written block has been specified.</p> <p>Bit 1: Track overflow (that is, overflow blocks are being used). (Refer to &amp;saml..)</p> <p>Bit 2: Extended search has been specified.</p> <p>Bit 3: Feedback of block address has been specified.</p> <p>Bit 4: Actual block addressing is being used.</p> <p>Bit 5: Dynamic buffering is being used.</p> <p>Bit 6: Exclusive control is being used.</p> <p>Bit 7: Relative block addressing is being used.</p> <p><b>Note:</b> If bits 4 and 7 are both 0, relative track addressing is being used.</p>

Offset	Field	Field Size (in Bytes)	Field Contents and Comments
			<b>Second Byte:</b>
			<b>Bit 0:</b> S has been specified in the key address operand of the READ or WRITE macro instruction. For dynamic buffering, a buffer is to be allocated for a READ request, and the key part of a buffer is to be freed after a WRITE request.
			<b>Bit 1:</b> S has been specified in the length operand of the READ or WRITE macro instruction. When variable-length blocks are being written, the setting of this bit is ignored (that is, not tested), because the block length is given in the first 2 bytes of the data field.
			<b>Bit 2:</b> Next address feedback can be the address of either a data record or a capacity record, whichever occurred first. This bit can be set to 1 only when bit 3 (below) is set to 1.
			<b>Bit 3:</b> R is suffixed to the type of the READ or WRITE macro instruction, meaning that next address feedback has been requested.
			<b>Bit 4:</b> A READ request. (0 indicates a WRITE request.)
			<b>Bit 5:</b> The search argument is the block key. (0 indicates the search argument is the block ID.)
			<b>Bit 6:</b> Indicates a WRITE request to add a new block.
			<b>Bit 7:</b> A RELEX macro instruction has been issued.

Offset	Field	Field Size (in Bytes)	Field Contents and Comments
50(32)	IOBDSTAT	2	<p>Indication of status of the request. Significant bits of the IOBDSTAT field and their interpretations for BDAM are as follows (when the bit is set to 1, the interpretation is in effect; when set to 0, the interpretation is not in effect):</p> <p><b>First Byte:</b></p> <p><b>Bit 0:</b> The request has completed abnormally. See Second Byte for details.</p> <p><b>Bit 1:</b> On extended search, the ASI routine in the foundation module is to issue the EXCP macro instruction after the end-of-extent appendage module has determined that the next extent is on a new volume. The end-of-extent appendage module cannot issue an EXCP macro instruction in this case.</p> <p><b>Bit 2:</b> SYNCH bit set by exclusive control module IGC0005C before it gives control (via the SYNCH macro) to address conversion and feedback modules. The address conversion and feedback modules test this field to determine which register to return control on.</p>

Offset	Field	Field Size (in Bytes)	Field Contents and Comments
			<p>Bit 3: On extended search, indicates to the relative block conversion routine that the second pass of a two-pass conversion routine has been completed.</p> <p><b>Note:</b> The first pass of the routine converts the starting address for a track search, and the second pass converts the address for the search limit. The bit is set to 1 when the second pass begins. This bit is also set by the self-format module after the module has calculated the number of bytes required to write a block on a track. Then, if additional tracks must be examined for space, the calculation of bytes required is bypassed.</p>
			<p>Bit 4: The READ-exclusive request related to this IOB has been placed on an intertask queue by the exclusive control module.</p>
			<p>Bit 5: A buffer has been assigned to this IOB.</p>
			<p>Bit 6: A given block (to be written) can fit on the track associated with the capacity record that has just been read into storage. Module IGG019KM or IGG019KN sets this indicator.</p>
			<p>Bit 7: Unused.</p>

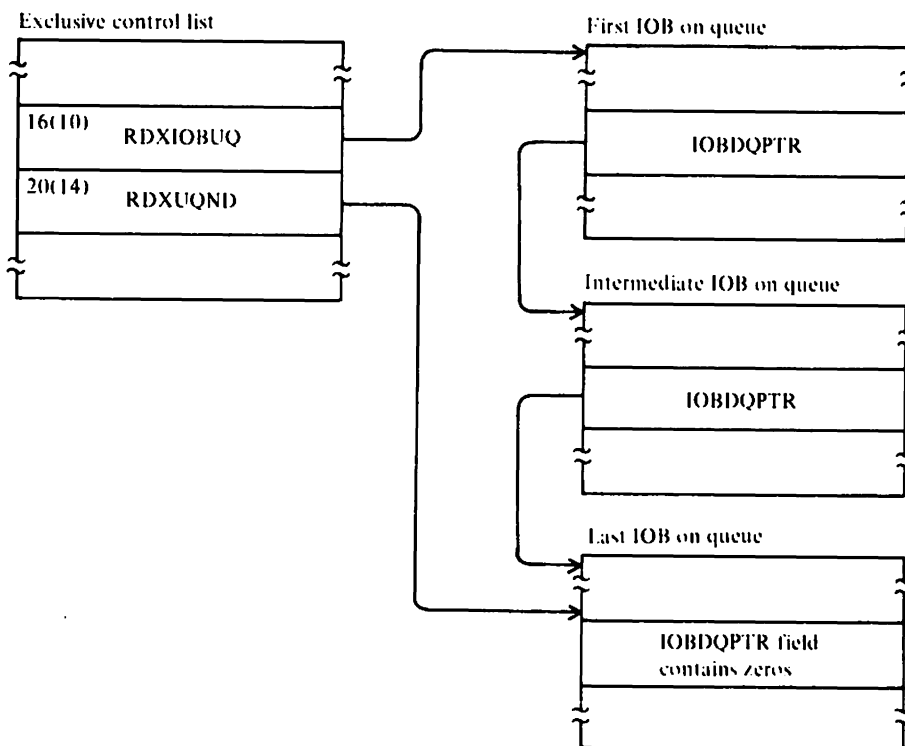
Offset	Field	Field Size (in Bytes)	Field Contents and Comments
			<b>Second Byte:</b>
			The bits in this byte, when on, indicate abnormal completion of a request. When the request is posted as complete, these indications of abnormal completion are placed in byte 1 (the second byte) of the DECSDECB field of the DECB.
			<b>Bit 0:</b> The requested block was not found on the indicated track.
			<b>Bit 1:</b> The length of the block was incorrect. Refer to "IGG019KU (Channel End/Abnormal End Appendage Module)" for more information about incorrect length.
			<b>Bit 2:</b> No space in which to write a new block was found.
			<b>Bit 3:</b> Request is invalid.
			<b>Bit 4:</b> An I/O error occurred.
			<b>Bit 5:</b> The request has been completed but the block the user has requested to be read or written is an end-of-data set record (indicated as having a data field length of 0). Refer to "IGG019KU (Channel End/Abnormal End Appendage Module)" for more information about the end-of-data set conditions.
			<b>Bit 6:</b> An error has occurred that cannot be attributed to any of the other causes indicated by bits in this byte.
			<b>Bit 7:</b> No match has been found on the exclusive control list during the processing of a WRITE-release request.

Offset	Field	Field Size (in Bytes)	Field Contents and Comments
52(34)	IOBDCPND	4	<p>The virtual-storage address of the expected end of the channel program if the program goes to a normal completion. This address is placed in the IOBDCPND field by the base routine of the foundation module.</p> <p>At the completion of a request, the I/O supervisor places an address in the channel status word. This address is equal to the address of the last channel command word executed, plus 8 bytes. A normal completion is indicated if the two addresses are equal and there have been no error indications.</p>
56(38)	IOBDBYTN	2	<p>The required number of bytes to contain a new block. This value is calculated by the self-format module after control has been given to it by the ASI routine in the foundation module.</p>
58(3A)	IOBRSV34	2	Reserved.
60(3C)	IOBDQPTR	4	<p>Either the address of the next IOB in the unposted queue of IOBs (address determined by the exclusive control module, IGC0005C) or the address of the next IOB in the IOB buffer queue (address determined by dynamic buffering module). During EXCP processing, if other than the first EXCP issued for this request, this field will contain the user's ECB address.</p>
64(40)	IOBUPLIM	8	<p>On extended search, the address of the first track following the last track to be searched.</p>
72(48)	IOBDNRCF	8	<p>The count field developed by the self-format module when a new block of variable-length or undefined-length records is to be added to a track.</p>

UNPOSTED QUEUE

IOBs waiting to get exclusive control of a resource are said to be on the unposted queue, which is a part of the exclusive control list.

The unposted queue is chained together as follows:



The address of the exclusive control list is in the DEBXXARG field in the DEB extension at offset X'14'. The DEBXTNP field in the DEB prefix (offset X'8') contains the address of the DEB extension.

UNSCHEDULED LIST (DSECT NAME: USL)

An unscheduled list can be built by either the open executor module or the dynamic buffering modules. Open executor module IGG0193A builds an unscheduled list when dynamic buffering is in effect and virtual storage is specified for the task. The list consists of a 16-byte header area and twelve 4-byte fields, each of which can contain one IOB address.

Dynamic buffering module IGG019JA gets storage for a larger unscheduled list if, when a buffer is not available to be assigned to a request, there is no room to put the address of the IOB associated with the request on the unscheduled list. Each time the dynamic buffering modules obtain a new unscheduled list, it is twice the size of the previous one. Because the open executor module obtains a 64-byte list, the unscheduled list first obtained by the dynamic buffering modules is 128 bytes long, the next is 256 bytes long, and so on.

The address of the unscheduled list in use at any given time is in the DCBDYNB field of the DCB.

FORMAT OF THE UNSCHEDULED LIST

0(0)	USLSIZE	Unused	} Header  } First IOB address field in unscheduled list
4(4)	USLFLAGS	5(5) USLBFRQT	
8(8)	USLBFRQB		
12(C)	USLCHAIN		
16(10)	USLSLOT1		

Offset	Field	Field Size (in Bytes)	Field Contents and Comments
0(0)	USLSIZE	4	Size, in bytes, of the unscheduled list currently being used. The last 2 bytes in this field are unused.
4(4)	USLFLAGS	1	Flag field. If bit 0 in this field is on, this is the list currently being used.
5(5)	USLBFRQT	3	Pointer to the field that contains the IOB address at the top (logically, not physically) of the unscheduled list. The request associated with this IOB is assigned the first available buffer.
8(8)	USLBFRQB	4	Pointer to the field that contains the IOB address at the bottom (logically, not physically) of the unscheduled list.
12(C)	USLCHAIN	4	Address of the unscheduled list used before this one was built. This field chains each new copy of the unscheduled list to the previously used unscheduled list so that, when the data set is closed, the virtual storage obtained for each list can be located and then released.
16(10)	USLSLOT1	4	First IOB address field in the unscheduled list. The IOB address fields are used in a wraparound fashion; when the last IOB address field available in the unscheduled list is used, the next IOB address is put in the USLSLOT1 field if it is free.



FORMAT OF AN IOB ADDRESS FIELD IN THE UNSCHEDULED LIST



USLENDL bit—if on, this IOB address field is the last one in the unscheduled list.

USLINUSE bit—if on, this IOB address field is in use.

When the IOB address field is in use, contains the address of an IOB associated with a request that is waiting to be assigned a buffer.

DIAGNOSTIC AIDS

MESSAGES AND CODES ISSUED BY BDAM MODULES

Figure 5 directs you to the BDAM module that detected the error that caused a particular completion code and/or message to be produced. (For the meaning of a specific code, refer to &scod.; for the meaning of a specific message, refer to System Messages.)

---

System Completion Code	Message Number	BDAM Module Detecting Error
001	IEC020I	IGG019LI
020	IEC138I	IGG0193A
013	IEC141I	IGG0193A
014	IEC208I	IGG0203A
020	IEC138I	IGG0193A
135	IEC903I	IGCT005C,IGCT105C
139	IEC905I	IGCT005G
235	IEC902I	IGCT005C
239	IEC904I	IGCT005G
335		IGC0005C
339		IGC0005G
435		IGC0005C
535		IGC0005C

Figure 5. Message/Module Cross-Reference

---

PROBLEM DETERMINATION

Problem determination (of O/C/EOV) can be used to help debug BDAM problems that occur during BDAM open executor processing. How to use problem determination is documented in Open/Close/EOV Logic.

ERROR INFORMATION RECORDED IN THE SDWA ON SYS1.LOGREC

The SDWA (STAE diagnostic work area) is the communications area used by the exclusive control or FREEDBUF recovery routines and R/TM (recovery/termination manager) when errors occur in the processing associated with the exclusive control or FREEDBUF SVCs. The following types of information are recorded in the SDWA:

- Which operations associated with the SVC were performed before abnormal termination occurred, and
- Which operations the recovery routine has performed in response to abnormal termination.

The recovery routines write the SDWA in the SYS1.LOGREC data set under the circumstances described in "Diagram 16. Free System Resources after Abnormal Termination" on page 54. You can retrieve the SDWA by running the service aid IFCEREPO, which is described in System Programming Library: SYS1.LOGRC Error Recording.

The SDWA has two parts: a fixed area, whose size and contents remain the same, and a variable area, whose size and contents change for each recovery routine. The format of the variable area in the SDWA is as follows:

**Control Area**

Offset (Hex)	Size in Bytes	Field Name	Field Contents
0	2	SDWAVRAL	Length of the variable recording area.
2	1	SDWADPVA	Form in which the data was dumped from SYS1.LOGREC by IFCEREPO.
		1... ....	Data dumped, in hexadecimal (the default if neither bit 1 or 2 is on).
		.1.. ....	Data dumped, in decimal.
		..xx xxxx	Reserved.
3	1	SDWAURAL	Length of the information contained in the variable recording area.

**Variable Recording Area Written by IGCT005C and IGCT105C (Exclusive Control Recovery Routine)**

Offset (Hex)	Size in Bytes	Field Name	Field Contents
0	1	RECAUDIT	Audit trail byte set by exclusive control SVC as processing is performed. Indicates how much processing was performed before abnormal termination occurred.
		.... ...1	The block ID (the 3-byte UCB address plus CCHHR) was saved in the SVRB.
		.... ..1.	The DEQ macro was issued.
		.... ..11	The exclusive control list entry was cleared.
		.... ..1..	The address of the waiting IOB was saved in the SVRB.
		.... ..1.1	The IOB's address was removed from the unposted queue.
		.... ..11.	Feedback was gotten for the waiting IOB.
		.... ..111	The waiting IOB was posted. (If it is a WRITE-add request, the current IOB was posted.)
		.... 1...	An ENQ macro was issued and the processing associated with it was completed.
		.... 1..1	An EXCP macro associated with a request to release exclusive control was issued.
		.... 1.1.	An ENQ macro should have been but was not issued.
		..1. ....	An entry was added to the exclusive control list.
		..11 ....	An ENQ macro was issued associated with a request to get exclusive control.
		..1.. ....	An EXCP macro was issued associated with a request to get exclusive control.
		..1.1 ....	The address of the next or last IOB on the unposted queue was saved.
		..11. ....	The current IOB was put on the unposted queue.

Contains Restricted Materials of IBM  
 Licensed Materials — Property of IBM

Offset (Hex)	Size in Bytes	Field Name	Field Contents
1	1	RECRCDP	Return code byte. It indicates the status of the attempt to get an SVC dump. .... .1.. The SVC dump was taken successfully. .... 1... Only a partial SVC dump was taken. .... 11.. The attempt to get an SVC dump was unsuccessful.
2	1	RECRCLN	Return code byte. It indicates the status of the attempt to clean up system resources. .... ...1 Cleanup was not attempted. The request for which the SVC was entered has not been satisfied. .... ..1. An attempt was made to finish getting or releasing exclusive control for this request. If the error was a user error, cleanup of queues was performed for the current request. If the error was not a user error, processing of the current request was completed. In all cases, if a waiting IOB was removed from the unposted queue, an attempt was made to complete processing the waiting request (represented by the waiting IOB). The DCB should be reusable with exclusive control requests. .... ..11 The resource for which exclusive control was requested was not enqueued; an attempt was made to restore all control blocks to their original state. The DCB should be reusable with exclusive control requests. .... .1.. Cleanup was not attempted. None was required, as all processing associated with the SVC was completed. The DCB should be reusable. However, because the error may be associated with a completed request, results are unpredictable. .... .1.1 Cleanup was not done. The task that terminated abnormally is enqueued on a resource that is inaccessible to other users. Reuse of this DCB with exclusive control requests is unpredictable. .... .11. Cleanup was not attempted. The resource for which exclusive control was being released has been removed from the system queue but remains on the exclusive control list. The resource is inaccessible to other users. .... .111 Cleanup was not attempted. Either the DEB or the audit trail bit was invalid, or the condition causing abnormal termination is unknown. .... 1... Abnormal termination occurred during cleanup processing. The cause of termination is unknown. Reuse of this DCB with exclusive control requests is unpredictable.
3	1		Unused.
4	4	RECREG0	Contents of register 0 on entry to the SVC. If a RELEX macro was issued, register 0 should contain the address of the block. Register 0 is not used if a BDAM module issued the SVC.
8	4	RECREG1	Contents of register 1 on entry to the SVC. If a RELEX macro was issued, register 1 should contain the DCB address. Register 1 should contain the IOB address complemented if a BDAM module issued the SVC.
C	8	RECOPSW	The SVC old PSW, which contains the address (+4) from which the SVC was issued and protect key.
14	3	RECPERC	Abend code issued before the 135 abend code.

Variable Recording Area Written by IGCT005G (FREEDBUF Recovery Routine)

Offset (Hex)	Size in Bytes	Field Name	Field Contents
0	1	RECAUDIT	Audit trail byte set by FREEDBUF SVC as processing is performed. Indicates how much processing was performed before abnormal termination occurred.
		...1 ....	The buffer was freed and put back on the queue of available buffers.
		..1. ....	The buffer was freed and was then given to a waiting request. An EXCP macro was issued to schedule execution of the channel program for the waiting request.
		.1.. ....	The buffer was freed and then given to a waiting request. The channel program for that request was initialized but the EXCP macro was not issued.
		1... ....	The buffer was freed and then given to a waiting request. The channel program for that request was not initialized.
1	1	RECRCDP	Return code byte. It indicates the status of the attempt to get an SVC dump.
		.... .1..	The SVC dump was taken successfully.
		.... 1...	Only a partial SVC dump was taken.
		.... 11..	The attempt to get an SVC dump was unsuccessful.
2	1	RECRCLN	Return code byte. It indicates the status of the attempt to clean up system resources.
		.... ...1	Cleanup was unsuccessful. During cleanup, a program check occurred. Reuse of this DCB, if dynamic buffering is in effect, is unpredictable.
		.... ...1.	Cleanup was successful. This DCB should be reusable with dynamic buffering requests.
		.... ...11	Cleanup was not attempted. It was determined to be unnecessary, because all required processing was completed. This DCB should be reusable.
		.... .1..	Cleanup was not attempted. Either the DEB or the audit trail bit was invalid. Reuse of this DCB with dynamic buffering requests is unpredictable.
3	1		Unused.
4	4	RECREG0	Contents of register 0 on entry to the SVC. Register 0 contains the address of the DECB, unless the SVC was entered to extend the unscheduled list. In that case, the DECB address will be complemented.
8	4	RECREG1	Contents of register 1 on entry to the SVC. Register 1 contains the address of the DCB.
C	8	RECOPSW	The SVC old PSW, which contains the address (+4) from which the SVC was issued and the protect key.
14	3	RECPERC	Abend code issued before the 139 abend code.

ERROR INFORMATION RECORDED IN THE GTF DATA SET

When a user's control block is found to be invalid by recovery module IGCT005C or IGCT105C (exclusive control) or IGCT005G (FREEDBUF), the GTRACE macro is issued by the recovery module. This macro results in diagnostic information being moved into the GTRACE work area and subsequently written to the GTF data set.

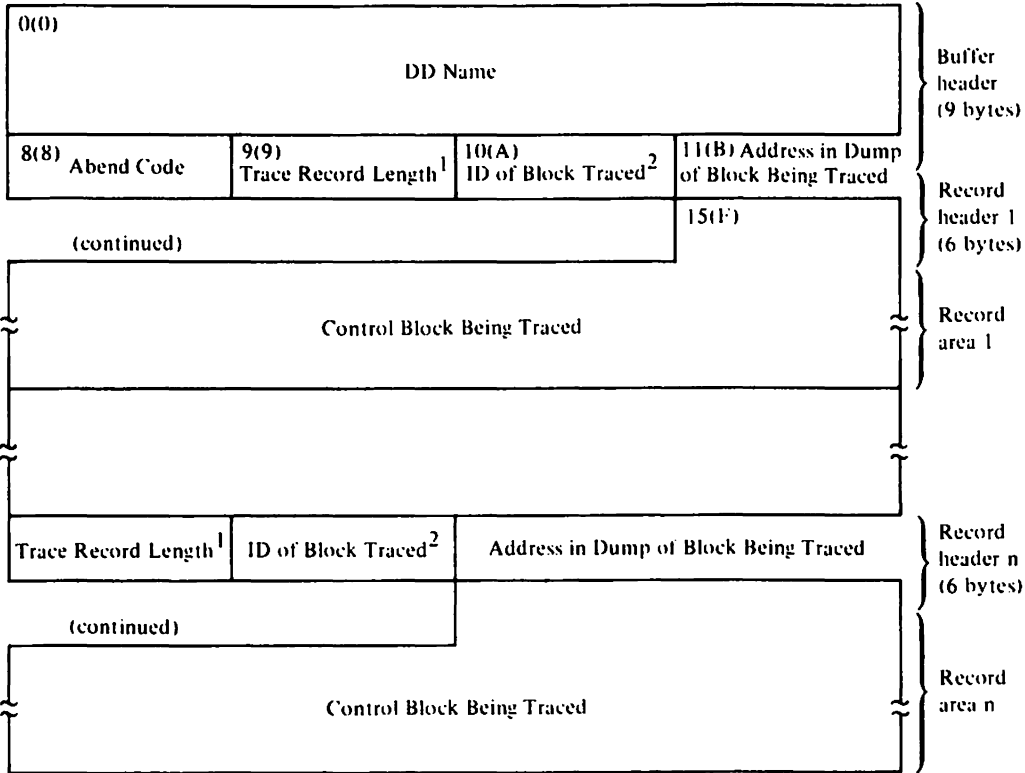
You can retrieve this information from the GTF data set by running the AMDPRDMP service aid, which is described in System Programming Library: Service Aids.

Figure 6 shows which control blocks will be written in the GTF data set. The abend and return code are printed in the WTP message generated when a user's control block is found to be invalid by one of the recovery modules.

Abend Code	Return Code	Control Blocks Written in the GTF Data Set
235	84	First 72 bytes of the IOB (excluding the prefix if the application involved format VS records).
	85	DCB (104 bytes).
	86	Exclusive control list entry (16 bytes).
239	82	DCB (104 bytes).
	83	DCB (104 bytes) and DECB (24 bytes).
	84	DCB (104 bytes), DECB (24 bytes), and BCB (16 bytes).
	85	DCB (104 bytes), DECB (24 bytes), and BCB (16 bytes).

Figure 6. Control Blocks Written in the GTF Data Set

Figure 7 on page 92 shows the format of the GTRACE work area. The maximum size of this work area is 256 bytes.



<sup>1</sup> The length from the beginning of this field to the beginning of the next trace record length field.

<sup>2</sup> Block ID Control Block Being Traced

129	BCB
130	DCB
131	DECB
133	IOB
140	Exclusive control list entry

Figure 7. Format of the GTRACE Work Area

APPENDIX A. PERIODS OF AN EXTENT

BDAM data sets are created by the basic sequential access method (BSAM). When BSAM places blocks on a direct-access device, it is possible that a block may start on one track and finish on a following track within the same extent. Such a block is called an overflow block. At least 1 byte of the data portion of a block must fit on a track in order for the block to overflow to the next track. For purposes of calculating the size of a period, the track on which a block begins is considered to contain the block. When a track is reached on which not enough space is left to contain at least 1 byte of the data portion of a new block to be written, the end of a period has been reached.

Thus, a period is a group of tracks containing a group of blocks such that the first track does not begin with an overflow block, and the last track does not contain a block that overflows to another track. The example illustrates the concept of a period.

**EXAMPLE:** In this example, assume the following:

- A given data set is on a device that permits 3625 bytes per track to be allocated to data blocks, excluding a track capacity record (R0).
- The block length for the data set is 844 bytes, divided as follows:
  - 14 bytes for address marker plus count area
  - 100 bytes for the block key portion
  - 730 bytes for the block data portion
- There are no interrecord gaps on the tracks. (This assumption simplifies the calculations in the example.)
- Part of the given data set occupies an extent consisting of consecutive tracks beginning with track 47 on this device.

From the second assumption, it is determined that at least the first 115 bytes (count + key + 1 data byte) of a block must fit on a track for track overflow to occur. Figure 8 illustrates the manner in which the data blocks would appear on the tracks of the extent.

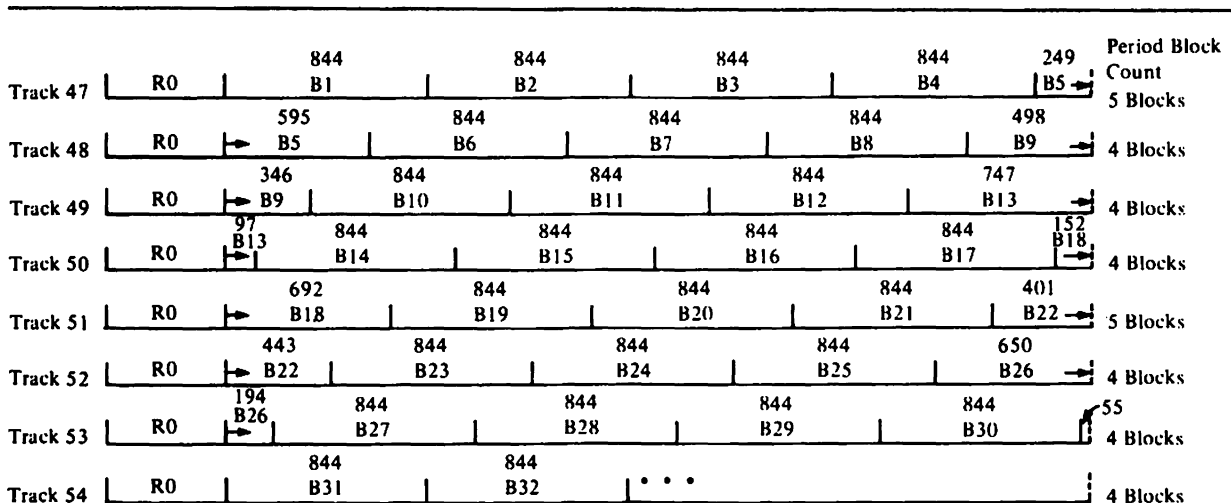


Figure 8. Track Overflow



**Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM**

The identifications B1, B2, ..., B32 represent the first 32 blocks placed in this extent. The numbers above the blocks represent the number of bytes of the block appearing on a track. Arrows at the end of a track and at the beginning of a track indicate where track overflow occurs.

As shown in Figure 8 on page 93, track overflow occurs from track 47 to track 48 (block B5), from track 48 to track 49 (block B9), and so on until the end of track 53. Because track overflow (in this example) requires at least the first 115 bytes of a block to appear on a track, and only 55 bytes remain on track 53 after block B30 has been placed there, block B31 cannot overflow from track 53 to track 54. Therefore, tracks 47 through 53 constitute a period, and a new period begins with block B31 on track 54. For purposes of calculating relative block addresses (see Examples 2 and 3 in Appendixes B and C), the number of blocks on each track is given in Figure 8 as the "Period Block Count."

Module IGG0193A computes, based on block characteristics (key length and data length) and device characteristics, the size of the period for the data set. It then computes both the number of blocks in a period and the number of tracks in a period, and places the computed values in the two fields of the overflow section of the DEB. These fields are each 1 word long, and they occur only once for a given data set. The values placed in these fields are constant for a given data set.

Because the allocation of actual extents to members of a data set is performed by space management routines (refer to &dadsoref.), while the period is a concept used by BDAM, the boundaries of extents and periods may differ. However, the end of an extent terminates the last period in the extent. In this case, the last period may be complete, or it may be only partially complete. In either case, the start of a new extent coincides with the start of a new period.

APPENDIX B. CALCULATIONS DONE IN MODULE IGG019KE TO GET A RELATIVE TRACK ADDRESS

To convert a relative block address to a relative track address (track overflow not in effect), module IGG019KE does the following calculations: Starting with the first extent in the data set, IGG019KE subtracts the number of blocks in that and each successive extent from the BLKREF field. (The BLKREF field, whose address is in the DECRCPT field of the DECB, contains the relative block number that was specified in the block address field of the READ or WRITE macro instruction.) As the number of blocks in each extent is subtracted, the number of tracks in each subtracted extent is accumulated. Thus, the value in the BLKREF field decreases while the number of accumulated tracks increases. This process continues until an extent is reached in which the number of blocks, if subtracted from the number that remains in the blkref field, would result in a negative value. This extent is called the terminal extent.

When the terminal extent is reached, the number of remaining blocks in the blkref field is divided by the blocks per track field in the DEB. The quotient in this division is added to the cumulative total of tracks. This cumulative total represents the track number, relative to the beginning of the data set, on which the block whose address is to be converted resides. The remainder in this division is the number of blocks on that particular track that must be counted to get to the block whose address is to be converted.

The relative track address, in the form TTR, is the result of these calculations. TT is the number of the track on which the block whose address is to be converted resides, and R is the number of that block.

The example illustrates conversion of a relative block number to a relative track address.

**EXAMPLE:** Assume a data set is contained in four extents identified as I, II, III, and IV. Let extent I contain 10 tracks with 80 data blocks; extent II contain 14 tracks with 112 data blocks; extent III contain 8 tracks with 64 data blocks; and extent IV contain 12 tracks with 96 data blocks. (This assumes that the data set is on a device permitting 8 data blocks to be placed on one track.) The information needed from the DEB for this data set is summarized in Figure 9.

---

Without Track Overflow

DEB Field	Extent I	Extent II	Extent III	Extent IV
Blocks per Track	8	8	8	8
Tracks per Extent	10	14	8	12
Blocks per Extent	B1=80	B2=112	B3=64	B=96

Figure 9. DEB Information Needed to Calculate Relative Track Addresses in IGG019KE

---

If the blkref field contains 284, the calculations to find the relative track address are as follows:

$$\text{blkref value} - B1 = R1 \text{ (remainder)}$$

$$284 - 80 = 204 \text{ The 80 blocks from extent I are on 10 tracks.}$$

$$R1 - B2 = R2$$

$$204 - 112 = 92 \text{ The 112 blocks from extent II are on 14 tracks.}$$

$$R2 - B3 = R3$$

$$92 - 64 = 28 \text{ The 64 blocks from extent III are on 8 tracks.}$$

$$R3 - B = R$$

$$28 - 96 < 0$$

Because R is less than 0, the last extent (IV) cannot be subtracted. Extent IV becomes the terminal extent. The previous remaining value ( $R3 = 28$ ) is divided by the blocks per track value (8) to give a quotient of 3 and a remainder of 4. The 3 represents the number of tracks of the terminal extent that must be added to the sum of the underlined numbers of tracks from extents I, II, and III. The 4 represents the number of data blocks that must be counted from the beginning of the terminal track. Thus, the relative track address (TTR) of the block in this example is 35 tracks (the TT value) and 4 blocks (the R value) from the beginning of the data set.

APPENDIX C. CALCULATIONS DONE IN MODULE IGG019KF TO GET RELATIVE TRACK ADDRESS

To convert a relative block address to a relative track address when track overflow is in effect, module IGG019KF does the following calculations:

Starting with the first extent in the data set, IGG019KF subtracts the number of blocks in that and each successive extent from the blkref field. (The blkref field, whose address is in the DECRCPT field of the DECB, contains the relative block number that was specified in the block address field of the READ or WRITE macro instruction.) As the number of blocks in each extent is subtracted, the number of tracks in each subtracted extent is accumulated. This process continues until an extent is reached in which the number of blocks, if subtracted from the number that remains in the blkref field, would result in a negative value. This extent is called the terminal extent.

The number of blocks in each period of the terminal extent is then subtracted from the blkref field. As the number of blocks in each period are subtracted, the number of tracks on which these blocks reside is added to the cumulative total of tracks. This process continues until a period is reached in which the number of blocks, if subtracted from the number that remains in the blkref field, would result in a negative value. This period is called the terminal period.

Beginning with the terminal period, the number of bytes required for each block (plus overhead and count and key field lengths) is calculated and then subtracted from the track capacity. If the result is not negative, 1 is subtracted from the blkref field. Then the position of the next block on the track is calculated. This process continues until or unless only a portion of a block fits on a track (an overflow block is reached). At this point, 1 is added to the track count and the segment (plus overhead) of the overflow block on the next track is subtracted from the track capacity.

When the value in the blkref field is finally reached, the accumulated tracks and the block number become the relative track address, in the form TTR. TT is the track on which the block whose address is to be converted resides, and R is the actual number of the block on that track.

The example illustrates conversion of a relative block number, when track overflow has been specified, to a relative track address.

**EXAMPLE:** Assume a data set is contained in three extents identified as I, II, and III. Let extent I contain 20 tracks with 114 data blocks; extent II contain 10 tracks with 57 data blocks; and extent III contain 27 tracks with 153 data blocks. Further, assume that open executor module IGG0193A has established that each period contains 3 tracks with a total of 17 blocks, and that the blocks are placed on the tracks so that the first 2 tracks contain 6 blocks each and the third track contains 5.

The information needed from the DEB for this data set is summarized in Figure 10 on page 98.

---

With Track Overflow

DEB Field	Extent I	Extent II	Extent III
Tracks per Extent	20	10	27
Tracks per Period	3	3	3
Blocks per Period	17	17	17
Blocks per Extent	B1=114	B2=57	B3=153

Figure 10. DEB Information Needed to Calculate Relative Track Addresses in IGG019KF

---

If the blkref field contains 217, the calculations to find the relative track address are as follows:

blkref value - B1 = R1 (remainder)

217 - 114 = 103. The 114 blocks (from extent I) are on 20 tracks.

R1 - B2 = R2

103 - 57 = 46. The 57 blocks (from extent II) are on 10 tracks.

R2 - B3 = R3

46 - 153 < 0

Because R3 is less than 0, the last extent (III) cannot be subtracted. Extent III becomes the terminal extent.

Now the periods in the terminal extent are considered.

Let the periods of extent III be designated as IIIa, IIIb, IIIc, etc. The calculations proceed as follows:

R2 - IIIa = R3

46 - 17 = 29. The 17 blocks (from period IIIa) are on 3 tracks.

R3 - IIIb = R

29 - 17 = 12. The 17 blocks (from period IIIb) are on 3 tracks.

R - IIIc = R

12 - 17 < 0

Because R is less than 0, all the blocks in the period (IIIc) cannot be subtracted. Period IIIc becomes the terminal period. Now the block positions on the tracks in the terminal period are calculated. The 12 remaining blocks will take up 1 track (of 6 blocks), plus 6 blocks (on the terminal track).

The total number of tracks plus additional blocks thus is equal to the sum of the tracks in the two full extents (I and II), and the two full periods (IIIa and IIIb) in extent III, plus the 2 tracks from period IIIc. This value is 38 (which is track 37, because the track value is relative to track 0) tracks and 6 blocks, giving a TTR value that can be used by the BPAM convert-to-actual routine to obtain an actual address for the block.

APPENDIX D. CHANNEL PROGRAMS

The channel program for each request using BDAM is constructed by the appropriate channel program generating module and placed in the IOB for that request. "Diagram 4. Prepare to Execute Channel Program" on page 18 is a cross-reference from the module generating a channel program to the actual channel program shown in this section.

A channel program consists of a group of channel command words (CCWs), each word having the following format:

Command Code (1 byte)	Address (3 bytes)	Flags (5 bits)	OOO (3 byte)	(ignored) (1 byte)	Count (2 bytes)
-----------------------	-------------------	----------------	--------------	--------------------	-----------------

**Note:** The last 4 bytes are ignored by a transfer-in-channel (TIC) command word.

The entry in the address field is one of the following:

- The virtual-storage address at which data is to be placed or found; this is for a read or write command word.
- The location of the search argument; this is for a search command word.
- The address of the CCW to which a transfer is made; this is for a transfer-in-channel command word.

The entry (or entries) in the flags field have the following meanings:

- CC Command chaining
- DC Data chaining
- SKIP Skip the transferring of data
- SILI Suppress incorrect length indication

The entry in the count field represents either the number of bytes of data to be transferred, or the number of bytes of data on which a search is to be made for comparison.

In the channel programs that follow, the purpose of each command word is given in the comment following the count field. The channel command words are identified by the number to the left of the command code.

If track overflow has been specified, the applicable form of the channel program will end with a CCW having NOP as the command code and ignoring the other fields. The preceding CCW will also have the command chaining (CC) flag bit set on.

1. CHANNEL PROGRAM FOR READING OR WRITING BY BLOCK ID (TYPE DI)

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
1 <sup>1,2</sup>	23	Set sector	Sector address 1	40	CC	1	Wait for record—sector value is precalculated for format-F record
2	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for specified block
3	08	TIC	CCW2	00			Repeat search if block not found
4 <sup>3</sup>	0E 0D	Read Write Key and data	Contents of DECKYADR	80	DC	Key Length	Read or write key portion of block
5	06 05	Read Write Data	Contents of DECAREA	00 40 <sup>4</sup>	(CC)	Data length	Read or write data portion of block
6 <sup>2,4,5</sup>	22	Read sector	Sector address 2	40	CC	1	Get sector value for validity-check routine address 2
7 <sup>4,6</sup>	1B	Seek cylinder head (CCHH)	IOBDNRCF	40	CC	6	Seek back to track where block begins
8 <sup>2,4</sup>	23	Set sector	Sector address 2	40	CC	1	Wait for record to come into position again address 2
9 <sup>4</sup>	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for block just updated equal
10 <sup>4</sup>	08	TIC	CCW9	00			Repeat search if block not found
11 <sup>4</sup>	0E	Read key and data		30	SILI, SKIP	256	Read block just written to check for errors—write-validity-check

**Notes:**

1. CCW1 is present only for fixed-length records.
2. This CCW is present only for devices with the rotational position sensing feature.
3. CCW4 is omitted if either the DCBKEYLE or DECKYADR is zero.

**Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM**

4. CCWs 6 through 11 are present and the command-chain bit in CCW4 is on only when the DCBOPTCD field of the DCB specifies the write-validity-check option.
5. CCW6 is omitted for fixed-length records.
6. CCW7 is present only if track overflow is specified.



**2. CHANNEL PROGRAM FOR READING OR WRITING BY BLOCK KEY (TYPE DK)**

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
1	12	Read count	IOBDNRCF+2	60	CC, SILI	5	Read full address (CCHHR) for feedback
2	29	Search key equal	Contents of DECKYADR	60	CC, SILI	Key length	Search for block with specified key length
3	08	TIC	CCW1	00			Repeat search if block not found
4	06 05	Read data Write	Contents of DECAREA	00 40 <sup>2</sup>	(CC)	Data length	Read or update the desired block
5 <sup>1,2</sup>	22	Read sector	Sector address 2	40	CC	1	Get sector value for validity-check routine
6 <sup>2,3</sup>	1B	Seek head (CCHH)	IOBDNRCF	40	CC	6	Seek back to track where block begins
7 <sup>1,2</sup>	23	Set sector	Sector address 2	40	CC	1	Set to sector of last write
8 <sup>2</sup>	29	Search key equal	Contents of DECKYADR	60	CC, SILI	Key length	Search for block just updated
9 <sup>2</sup>	08	TIC	CCW6	00			Repeat search if block not found
10 <sup>2</sup>	06	Read data		30	SILI, SKIP	256	Read block just written to check for errors—write-validity-check

**Notes:**

1. These CCWs are present only for devices with the rotational position sensing feature.
2. CCWs 5 through 10 are present and the command-chain bit in CCW4 is on only when the DCBOPTCD field of the DCB specifies the write-validity-check option.
3. CCW6 is present only if track overflow is specified.

**3. CHANNEL PROGRAM FOR ADDING A NEW BLOCK OF FIXED-LENGTH RECORDS (TYPE DA)**

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
1	12	Read count	IOBDNRCF+2	60	CC, SILI	5	Read full address (CCHHR) for feedback
2	29	Search key equal	Dummy key	60	CC, SILI	1	Search for a dummy record
3	08	TIC	CCW1	00			Repeat search if record not found
4 <sup>5</sup>	06	Read data	IOBDNRCF+6	60	CC, SILI	1	Read dummy record's position—"R" of CCHHR
5 <sup>1,5</sup>	22	Read sector	Sector address 2	40	CC	1	Note sector value of dummy record
6 <sup>2</sup>	0B	Seek cylinder (CCHH)	IOBDNRCF	40	CC	6	Seek back to track where block begins
7 <sup>1</sup>	23	Set sector	Sector address 2	40	CC	1	Wait for dummy record to come into position again
8	31	Search ID equal	IOBDNRCF+2	40	CC	5	Search for dummy record
9	08	TIC	CCW8	00			Repeat search if record not found
10	0D	Write key and data	Contents of DECKYADR	80	DC	Key length	Write the new key
11	05	Write data	Contents of DECAREA	00 40 <sup>3</sup>	(CC)	Data length	Write the new data
12 <sup>3</sup>	0B	Seek cylinder (CCHH)	IOBDNRCF	40	CC	6	Seek back to track where block begins
13 <sup>1</sup>	23	Set sector	Sector address 2	40	CC	1	Wait for block just written to come into position again
14	31	Search ID equal	IOBDNRCF+2	40	CC	5	Search for block just written
15	08	TIC	CCW14	00			Repeat search if block not found

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
16	0E	Read key and data		30	SILI, SKIP	256	Check block just written for errors—write-validity-check
17 <sup>4</sup>	23	Set sector	Sector address 2	40	CC	1	Wait for dummy record to come into position again
18	31	Search ID equal	IOBDNRCF+2	40	CC	5	Search for dummy record
19	08	TIC	CCW18	00			
20	29	Search key equal	Dummy key address	60	CC, SILI	1	Verify record is still dummy
21	03	NOP		20	SILI	1	End channel program if record not dummy (Key is not X'FF')
22	1A	Read home address		70	CC, SILI, SKIP	1	Read home address on this track
23	08	TIC	CCW 7/8	00			

**Notes:**

1. These CCWs are present only for devices with the rotational position sensing feature.
2. CCWs 6 and 12 are present only if track overflow is specified.
3. CCWs 12 through 16 are present and the command-chain bit in CCW11 is on only when the DCBOPTCD field of the DCB specifies the write-validity-check option.
4. CCWs 17 through 23 verify that the dummy record is still a dummy record. They are executed from the channel-end or abnormal-end appendage (IGG019KU), but only if the abnormal end appendage is entered before the write key and data CCW is executed.
5. If the abnormal-end appendage (IGG019KU) is entered before or when this CCW is executed, the command chain bit is turned off.

**4. CHANNEL PROGRAM FOR WRITING A NEW BLOCK OF VARIABLE-LENGTH OR UNDEFINED-LENGTH RECORDS (TYPE DA)**

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
1 <sup>1</sup>	23	Set sector	Sector=0	40	CC	1	Set to beginning of track
2	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for track capacity record
3	08	TIC	CCW2	00			Repeat search if record not found
4	06	Read data	IOBDNRCF	20	SILI	7	Read capacity record into IOB
5 <sup>1,2</sup>	23	Set sector	Sector=0	40	CC	1	Wait for beginning of track to come into position again
6	31	Search ID equal	IOBUPLIM	40	CC	5	Search for track capacity record
7	08	TIC	CCW6	00			Repeat search if record not found
8	05	Write data	IOBSEEK+3	60	CC, SILI	7	Update capacity record
9	31	Search ID equal	CCW2	40	CC	5	Search for current last block on track
10	08	TIC	CCW9	00			Repeat search if block not found
11	1D	Write count	IOBDNRCF	80	DC	8	Write new record
12 <sup>3</sup>	ID	Write key	Contents of DECKYADR	80	DC	Key length	Write new record
13	1D	Write data	Contents of DECAREA	00 40 <sup>4</sup>	DC	Data length	Write new record
14 <sup>1,4</sup>	22	Read sector	Sector address 2	40	CC	1	Note sector value of new record
15 <sup>1,4</sup>	23	Set sector	Sector address 2	40	CC	1	Wait for new record to come into position again
16 <sup>4</sup>	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for new record

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
17 <sup>4</sup>	08	TIC	CCW16	00			Repeat search if record not found
18 <sup>4</sup>	0E	Read key and data		70	CC, SILI, SKIP	256	Check new record for errors—write-validity
19 <sup>1,4</sup>	23	Set sector	Sector=0	40	CC	1	Wait for beginning of track to come into position again
20 <sup>4</sup>	16	Read R0		00		16	Check capacity record for errors—write-validity-check

**Notes:**

1. These CCWs are present only for devices with the rotational position sensing feature.
2. This is actually two channel programs. The first program (CCWs 1 through 4) reads the capacity record. After the BDAM routines find a capacity record that indicates enough space is available for the add request, CCWs 1 through 4 are overlaid with data that includes the address of the current last block. The channel program that actually writes the new block consists of CCWs 5 through 20.
3. CCW12 is omitted if either the DCBKEYLE OR DECKYADR field is 0.
4. CCWs 14 through 20 are present and the command-chain bit in CCW13 is on only when the DCBOPTCD field of the DCB specifies the write-validity-check option.)

**5. CHANNEL PROGRAM FOR READING OR WRITING BY BLOCK KEY USING EXTENDED SEARCH (TYPE DK)**

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
1 <sup>1</sup>	23	Set sector	Sector=0	40	CC	1	Set to beginning of track
1 <sup>1</sup>	03	NOP		60	CC, SILI	1	
2	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for capacity record
3	08	TIC	CCW2	00			Repeat search if record not found
4 <sup>5</sup>	B1	Search ID equal (multi-track)	IOBUPLIM+3	40	CC	5	Check for search limit
5	08	TIC	CCW7	00			Go to key search if still within search limit
6	03	NOP		20	SILI	1	End channel program if search limit reached
7	29	Search key equal	Contents of DECKYADR	60	CC, SILI	Key length	Search for given key
8	08	TIC	CCW4	00			Check limit again before continuing search
9 <sup>1</sup>	22	Read sector	Sector address 2	40	CC	1	Note sector value to be used for read or write
9 <sup>1</sup>	03	NOP		60	CC, SILI	1	
10 <sup>1,2</sup>	23	Set sector	Sector=0	40	CC	1	Wait for beginning of track to begin feedback loop
10 <sup>1,2</sup>	03	NOP		60	CC, SILI	1	
11	1A	Read home address		70	CC, SILI, SKIP	1	Note beginning of track

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
12 <sup>1</sup>	23	Set sector	Sector address 2	40	CC	1	Wait for desired record to come into position
12 <sup>1</sup>	03	NOP		60	CC, SILI	1	
13	12	Read count	IOBDNRCF+2	60	CC, SILI	5	Read CCHHR for feedback
14	29	Search key equal	Contents of DECKYADR	60	CC, SILI	Key length	Search for user-specified record
15	08	TIC	CCW13	00			Update CCHHR and repeat search if wrong record
16	06 05	Read data Write	Contents of DECAREA	00 40 <sup>3</sup>	(CC)	Data length	Read or update the block
17 <sup>3,4</sup>	1B	Seek head (CCHH)	IOBDNRCF	40	CC	6	Seek back to track where block begins
18 <sup>1</sup>	23	Set sector	Sector address 2	40	CC	1	Wait for desired record to come into position
18 <sup>1</sup>	03	NOP		60	CC, SILI	1	
19	29	Search key equal	Contents of DECKYADR	60	CC, SILI	Key length	Search for block just updated
20	08	TIC	CCW19	00			Repeat search if block not found
21	06	Read data		30	SILI, SKIP	256	Check block for errors—write-validity-check

**Notes:**

1. These CCWs are set to NOP for devices without the rotational position sensing feature.
2. CCWs 10 through 15 are present only if feedback is requested.
3. CCWs 17 through 21 are present and the command-chain bit in CCW16 is on only when the DCBOPTCD field of the DCB specifies the write-validity-check option.
4. CCW17 is present only if track overflow is specified.
5. This command code is changed to X'31' if the user specified actual addressing.

6. CHANNEL PROGRAM FOR ADDING A NEW FIXED-LENGTH BLOCK USING EXTENDED SEARCH (TYPE DA)

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
1 <sup>1</sup>	23	Set sector	Sector=0	40	CC	1	Wait for beginning of track to come into position
2	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for capacity record
3	08	TIC	CCW2	00			Repeat search if record not found
4	06	Read data	IOBDNRCF+2	60	CC, SILI	5	Read full address (CCHHR) for feedback
5	31	Search ID equal	IOBDNRCF+2	40	CC	5	Search for last data record on track
6	08	TIC	CCW13	00			If not last data block, check for dummy record
7	29	Search key equal	Dummy key address	60	CC, SILI	1	Search for dummy record
8	08	TIC	CCW10	00			If not dummy record, go to extended search routine
9	08	TIC	CCW15	00			If dummy record found, go to add routine
10	B1	Search ID equal (multi-track)	IOBUPLIM+3	40	CC	5	Check for search limit
11	08	TIC	CCW4	00			Read capacity record from next track if still within search limit
12	03	NOP		20	SILI	1	End channel program if search limit reached
13	29	Search key equal	Dummy key address	60	CC, SILI	1	Search for dummy record
14	08	TIC	CCW5	00			Check next record if dummy record not found



CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
15 <sup>4</sup>	06	Read data	IOBDNRCF+6	60	CC, SILI	1	Read dummy record's position—"R" of CCHHR
16 <sup>1,4</sup>	22	Read sector	Sector address 2	40	CC	1	Get sector value of dummy record
17 <sup>2</sup>	1B	Seek head (CCHH)	IOBDNRCF	40	CC	6	Seek back to track where dummy record began
18 <sup>1</sup>	23	Set sector	Sector address 2	40	CC	1	Wait for dummy record to come into position again
19	31	Search ID equal	IOBDNRCF+2	40	CC	5	Search for dummy record
20	08	TIC	CCW19	00			Repeat search if record not found
21	0D	Write key and data	Contents of DECKYADR	80	DC	Key length	Write new key
22	05	Write data	Contents of DECAREA	00 40 <sup>3</sup>	(CC)	Data length	Write data portion of record
23 <sup>3,2</sup>	1B	Seek head (CCHH)	IOBDNRCF	40	CC	6	Seek back to track that contains beginning of track just written
24 <sup>1</sup>	23	Set sector	Sector address 2	40	CC	1	Wait for record just written to come into position again
25	31	Search ID equal	IOBDNRCF+2	40	CC	5	Search for record just written
26	08	TIC	CCW25	00			Repeat search if record not found
27	0E	Read key and data		30	SILI, SKIP	256	Check block just written for error—write-validity-check
28 <sup>5</sup>	23	Set sector	Sector address 2	40	CC	1	Wait for dummy record to come into position again

Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
29	31	Search ID equal	IOBDNRCF+2	40	CC	5	Search for dummy record
30	08	TIC	CCW29	00			
31	29	Search key equal	Dummy key address	60	CC, SILI	1	Verify record is still dummy
32	03	NOP		20	SILI	1	End channel program if record not dummy (Key is not X'FF')
33	1A	Read home address		70	CC, SILI, SKIP	1	Read home address on this track
34	08	TIC	CCW 18/19	00			

**Notes:**

1. This CCW is present only for devices with the rotational position sensing feature.
2. This CCW is present only if track overflow is specified.
3. CCWs 23 through 27 are present and the command-chain bit in CCW22 is on only when the DCBOPTCD field of the DCB specifies the write-validity-check option.
4. If the abnormal-end appendage (IGG019KU) is entered before this CCW is executed, the command chain bit is turned off.
5. CCWs 28 through 34 verify that the dummy record is still a dummy record. They are executed from the channel-end or abnormal-end appendage (IGG019KU), but only if the abnormal end appendage is entered before the write key and data CCW is executed.

7. CHANNEL PROGRAM FOR ADDING A NEW VARIABLE-LENGTH SPANNED (FORMAT-VS) RECORD (TYPE DA)

A. ORIGINAL CHANNEL PROGRAM FOR READING CAPACITY RECORD AND WRITING FIRST SEGMENT

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
1 <sup>1</sup>	23	Set sector	Sector=0	40	CC	1	Wait for beginning of track to come into position
1 <sup>1</sup>	03	NOP		60	CC, SILI	1	
2	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for capacity record
3	08	TIC	CCW2	00			Repeat search if record not found
4	06	Read data	IOBDNRCF	20	SILI	7	Read capacity record into IOB
57 <sup>1</sup>	23	Set sector	Sector=0	40	CC	1	Wait for beginning of track to come into position again
5 <sup>1</sup>	03	NOP		60	CC, SILI	1	
6	31	Search ID equal	CCW2 <sup>2</sup>	40	CC	5	Search for capacity record
7	08	TIC	CCW6	00			Repeat search if record not found
8	05	Write data	IOBSEEK+3	60	CC, SILI	7	Update capacity record
9	31	Search ID equal	CCW1 <sup>2</sup>	40	CC	5	Search for last block on track
10	08	TIC	CCW9	00			Repeat search if block not found
11	1D	Write count	IOBDNRCF	80	DC	8	Write count field of new record
12 <sup>3</sup>	1D	Write key	Contents of DECKYADR	80	DC	Key length	Write key field
13	1D	Write data	Segment work area	00 40 <sup>4</sup>	(CC)	Segment length	Write data field
14 <sup>4,5</sup>	22	Read sector	Sector address 2	40	CC	1	Note sector value of record just written

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
15 <sup>4,5</sup>	23	Set sector	Sector address 2	40	CC	1	Wait for new record to come into position again
16 <sup>4</sup>	31	Search ID equal	I0BSEEK+3	40	CC	5	Search for record just written
17 <sup>4</sup>	08	TIC	CCW16	00			Repeat search if record not found
18 <sup>4</sup>	0E	Read key and data		70	CC, SILI, SKIP	256	Check the new block for error—write-validity-check
19 <sup>4</sup>	16	Read R0		10	SKIP	16	Check the capacity record

**Notes:**

1. This CCW is set to NOP for devices without the rotational position sensing feature.
2. This is really two channel programs. The first, CCWs 1 through 4, reads the capacity record. The second, CCWs 5 through 19, updates the capacity record and writes the first segment of the new record.
3. CCW12 is omitted if the DCBKEYLE or DECKYADR field is 0.
4. CCWs 14 through 19 are present and the command-chain bit in CCW13 is on only when the DCBOPTCD field of the DCB specifies the write-validity-check option.
5. This CCW is present only for devices with the rotational position sensing feature.

**B. CHANNEL PROGRAM FOR WRITING ALL SEGMENTS BUT THE FIRST (SUBSEQUENT SEGMENTS HAVE NO KEYS)**

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
1 <sup>1</sup>	23	Set sector	Sector=0	40	CC	1	Wait for beginning of track
1 <sup>1</sup>	03	NOP		60	CC, SILI	1	
2	31	Search ID equal	CCW2 of Part A	40	CC	5	Search for capacity record
3	08	TIC	CCW2	00			Repeat search if record not found
4	05	Write data	IOBSEEK+3	60	CC, SILI	7	Update capacity record
5	1D	Write count	IOBDNRCF	80	DC	8	Write count field of new segment (key length is specified as 0 in IOBDNRCF)
6	1D	Write data	Segment work area	00	(CC)		Segment length of write data field
7 <sup>1,2</sup>	23	Set sector	Sector=0	40	CC	1	Wait for beginning of track to come into position again
7 <sup>1,2</sup>	03	NOP		60	CC, SILI	1	
8 <sup>2</sup>	16	Read R0		50	CC, SKIP	16	Check capacity record for errors
9 <sup>2</sup>	0E	Read key and data		30	SILI, SKIP	256	Check new record for errors—write-validity-check

**Notes:**

1. This CCW is set to NOP for devices without the rotational position sensing feature.
2. CCWs 7 through 9 are present and the command-chain bit in CCW6 is on only when the DCBOPTCD field of the DCB specifies the write-validity-check option.

**8. CHANNEL PROGRAM FOR READING VARIABLE-LENGTH SPANNED RECORDS BY BLOCK KEY USING EXTENDED SEARCH**

**A. ORIGINAL CHANNEL PROGRAM AS DEVELOPED BY IGG019KR AND IGG019KW**

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
1 <sup>1</sup>	23	Set sector	Sector=0	40	CC	1	Wait for beginning of track
1 <sup>1</sup>	03	NOP		60	CC, SILI	1	
2	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for capacity record
3	08	TIC	CCW2	00			Repeat search if record not found
4	B1	Search ID equal (multi-track)	IOBUPLIM+3	40	CC	5	Check for search limit
5	08	TIC	CCW7	00			Check record's key if not beyond search limit
6	03	NOP		20	SILI	1	End channel program if search limit reached
7	29	Search key equal	Contents of DECKYADR	60	CC, SILI	Key length	Search for specified key
8	08	TIC	CCW4	00			Repeat search if key not found
9 <sup>1</sup>	22	Read sector	Sector address 2	40	CC	1	Note sector value of desired record
9 <sup>1</sup>	03	NOP		60	CC, SILI	1	
10 <sup>1</sup>	23	Set sector	Sector=0	40	CC	1	Go back to beginning of track for feedback loop
10 <sup>1</sup>	03	NOP		60	CC, SILI	1	
11	1A	Read home address		70	CC, SILI, SKIP	1	Note beginning of track

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
12 <sup>1</sup>	23	Set sector	Sector address 2	40	CC	1	Wait for desired record to come into position
12 <sup>1</sup>	03	NOP		60	CC, SILI	1	
13	12	Read count	IOBDNRCF+2	60	CC, SILI	5	Read CCHHR for feedback
14	29	Search key equal	Contents of DECKYADR	60	CC, SILI	Key length	Search again for desired record
15	08	TIC	CCW13	00			Repeat search if record not found
16	06	Read data	Segment work area	00		Data length	Read the record

**Note:**

1. This CCW is set to NOP for devices without the rotational position sensing feature.

**B. CHANNEL PROGRAM AFTER READING FIRST SEGMENT**

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
1 <sup>1</sup>	23	Set sector	Sector=0	40	CC	1	Wait for beginning of track
1 <sup>1</sup>	03	NOP		60	CC, SILI	1	
2	31	Search ID equal	I0BSEEK+3	40	CC	5	Search for second block on next track
3	08	TIC	CCW2	00			Repeat search if second block not found
4	06	Read data	Segment work area	00		Data length	Read the segment
5 <sup>2</sup>	03	NOP		20	SILI	1	

**Notes:**

1. This CCW is set to NOP for devices without the rotational position sensing feature.
2. This CCW is present only when "next address" feedback is requested.



9. CHANNEL PROGRAM FOR WRITING VARIABLE-LENGTH SPANNED RECORDS BY BLOCK KEY USING EXTENDED SEARCH

A. ORIGINAL CHANNEL PROGRAM AS DEVELOPED BY IGG019KR AND IGG019KW

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
1 <sup>1</sup>	23	Set sector	Sector=0	40	CC	1	Wait for beginning of track
1 <sup>1</sup>	03	NOP		60	CC, SILI	1	
2	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for capacity record
3	08	TIC	CCW2	00			Repeat search if record not found
4	B1	Search ID equal (multi-track)	IOBUPLIM+3	40	CC	5	Check for search limit
5	08	TIC	CCW7	00			Check record's key if not beyond search limit
6	03	NOP		20	SILI	1	End channel program if search limit reached
7	29	Search key equal	Contents of DECKYADR	60	CC, SILI	Key length	Search for specified key
8	08	TIC	CCW4	00			Check limit again if key not found
9 <sup>1</sup>	22	Read sector	Sector address 2	40	CC	1	Note sector value of desired record
9 <sup>1</sup>	03	NOP		60	CC, SILI	1	
10 <sup>1</sup>	23	Set sector	Sector=0	40	CC	1	Wait for beginning of track to begin feedback loop
10 <sup>1</sup>	03	NOP		60	CC, SILI	1	
11	1A	Read home address		70	CC, SILI, SKIP	1	Note beginning of track

Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
12 <sup>1</sup>	23	Set sector	Sector address 2	40	CC	1	Wait for desired record to come into position
12 <sup>1</sup>	03	NOP		60	CC, SILI	1	
13	12	Read count	IOBDNRCF+2	60	CC, SILI	5	Read CCHHR for feedback
14	29	Search key equal	Contents of DECKYADR	60	CC, SILI	Key length	Search for desired segment
15	08	TIC	CCW13	00			Repeat search if segment not found
16	06	Read data	Segment work area	20	SILI	8	Read block and segment descriptor words
17 <sup>1,2</sup>	23	Set sector	Sector address 1	40	CC	1	Wait for record to come into position again
17 <sup>1,2</sup>	03	NOP		60	CC, SILI	1	
18 <sup>2</sup>	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for record just written
19 <sup>2</sup>	08	TIC	CCW18	00			Repeat search if record not found
20 <sup>2</sup>	0E	Read key and data		30	SILI, SKIP	256	Check data just written

**Notes:**

1. This CCW is set to NOP for devices without the rotational position sensing feature.
2. CCWs 17 through 20 are not used in the first pass through this channel program.

**B. CHANNEL PROGRAM AFTER LOCATING AND READING THE COUNT FIELD OF THE FIRST SEGMENT TO DETERMINE ITS LENGTH**

The channel program now consists entirely of CCWs 1 through 4 and CCWs 16 through 20, which will be used to write all the segments. The channel command words were modified by the ASI routine in IGG019KJ.

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
1 <sup>1</sup>	23	Set sector	Sector address 1	40	CC	1	Wait for desired record
1 <sup>1</sup>	03	NOP		60	CC, SILI	1	
2	31	Search ID equal	I0BSEEK+3	40	CC	5	Search for record found during last pass
3	08	TIC	CCW2	00			Repeat search if record not found
4	B1	TIC	CCW16	00			If record is found, go to write it
5	08	TIC	CCW7	00			
6	03	NOP		20	SILI	1	
7	29	Search key equal	Contents of DECKYADR	60	CC, SILI	Key length	
8	08	TIC	CCW4	00			
9 <sup>1</sup>	22	Read sector	Sector address	40	CC	1	
9 <sup>1</sup>	03	NOP		60	CC, SILI	1	
10 <sup>1</sup>	23	Set sector	Sector=0	40	CC	1	
10 <sup>1</sup>	03	NOP		60	CC, SILI	1	
11	1A	Read home address		70	CC, SILI, SKIP	1	
12 <sup>1</sup>	23	Set sector	Sector address	40	CC	1	
12 <sup>1</sup>	03	NOP		60	CC, SILI	1	
13	12	Read count	Segment work area	60	CC, SILI	8	
14	29	Search key equal	Contents of DECKYADR	40	CC	Key length	

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
15	08	TIC	CCW13	00			
16	05	Write data	Segment work area	00 40 <sup>2</sup>	(CC)	Segment length	Write the updated record
17 <sup>1,2</sup>	23	Set sector	Sector address 1	40	CC	1	Wait for the record to come into position again
17 <sup>1,2</sup>	03	NOP		60	CC, SILI	1	
18 <sup>2</sup>	31	Search ID equal	I0BSEEK+3	40	CC	5	Search for an equal CCHHR
19 <sup>2</sup>	08	TIC	CCW18	00			Repeat search if not equal
20 <sup>2</sup>	0E	Read key and data		30	SILI, SKIP	256	Check record for errors

**Notes:**

1. This CCW is set to NOP for devices without the rotational position sensing feature.
2. CCWs 17 through 20 are present and the command-chain bit in CCW16 is on only when the DCBOPTCD field of the DCB specifies the write-validity-check option.

10. CHANNEL PROGRAM FOR READING VARIABLE-LENGTH SPANNED RECORDS (TYPE DI)

A. ORIGINAL CHANNEL PROGRAM AS DEVELOPED BY IGG019KR

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
1 <sup>1,2</sup>	23	Set sector	Sector=0	40	CC	1	Wait for beginning of track
1 <sup>1,2</sup>	03	NOP		60	CC, SILI	1	
2 <sup>2</sup>	31	Search ID equal	IOBUPLIM+3	40	CC	5	Search for capacity record
3 <sup>2</sup>	08	TIC	CCW2	00			Repeat search if record not found
4 <sup>2</sup>	06	Read data	IOBDNRCF+2	06	CC, SILI	5	Read data portion of R0 to find next address
5	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for specified record
6	08	TIC	CCW5	00			Repeat search if record not found
7 <sup>3</sup>	0E	Read key and data	Contents of DECKYADR	80	DC	Key length	Read the key
8	06	Read data	Segment work area	00		Data length	Read the data

**Notes:**

1. This CCW is set to NOP for devices without the rotational position sensing feature.
2. CCWs 1 through 4 are present only when "next address" feedback is specified.
3. CCW7 is omitted if either the DCBKEYLE or DECKYADR field is zero.

**B. CHANNEL PROGRAM AFTER READING THE FIRST SEGMENT**

CCW8 has replaced CCW7 because there are no keys for subsequent segments. The channel program now effectively ends with CCW7. The change was made by the ASI routine in IGG019KJ.

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
1 <sup>1,2</sup>	23	Set sector	Sector=0	40	CC	1	Wait for beginning of track
1 <sup>1,2</sup>	03	NOP		60	CC, SILI	1	
2 <sup>1</sup>	31	Search ID equal	IOBUPLIM+3	40	CC	5	Search for capacity record
3 <sup>1</sup>	08	TIC	CCW2	00			Repeat search if record not found
4 <sup>1</sup>	06	Read data	IOBDNRCF+2	60	CC, SILI	5	Read capacity record for "next address" feedback
5	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for next segment
6	08	TIC	CCW5	00			Repeat search if segment not found
7	06	Read data	Segment work area	00		Data length	Read the data
8	06	Read data	Segment work area	00		Data length	(This CCW not used)

**Notes:**

1. CCWs 1 through 4 are included only if "next address" feedback is specified.
2. This CCW is set to NOP for devices without the rotational position sensing feature.

11. CHANNEL PROGRAM FOR WRITING VARIABLE-LENGTH SPANNED RECORDS (TYPE DI)

A. ORIGINAL CHANNEL PROGRAM AS DEVELOPED BY IGG019KR

This channel program reads the segment descriptor word to determine the segment's length.

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
1	03	NOP		60	CC, SILI	1	(Will be changed in later pass)
2	31	Search ID equal	I0BSEEK+3	40	CC	5	Search for specified ID
3	08	TIC	CCW2	00			Repeat search if ID not found
4	06	Read data	Segment work area	20	SILI	8	Read 8 bytes from the data field (BDW/SDW)
5	05	Write data	Segment work area	00 40	(CC)	Data length	This CCW not used in this pass
6	22	Read sector	Sector address 2	40	CC	1	This CCW not used in this pass
6	03	NOP		60	CC, SILI	1	
7	23	Set sector	Sector address 2	40	CC	1	This CCW not used in this pass
7	03	NOP		60	CC, SILI	1	
8	31	Search ID equal	I0BSEEK+3	40	CC	5	This CCW not used in this pass
9	08	TIC	CCW8	00			This CCW not used in this pass
10	0E	Read key and data		60	SILI, SKIP	256	This CCW not used in this pass

**B. CHANNEL PROGRAM AFTER READING THE BLOCK DESCRIPTOR WORD OF THE FIRST SEGMENT TO DETERMINE ITS LENGTH**

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
1	03	NOP		60	CC, SILI	1	(Will be changed in later pass)
2	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for first segment
3	08	TIC	CCW2	00			Repeat search if segment not found
4 <sup>1</sup>	0D	Write key and data	Contents of DECKYADR	80	DC	Key length	Update the key
5	05	Write data	Segment work area	00 40 <sup>2</sup>	(CC)	Data length	Update the data
6 <sup>2,3</sup>	22	Read sector	Sector address 2	40	CC	1	Note position of this segment
6 <sup>2,3</sup>	03	NOP		60	CC, SILI	1	
7 <sup>2,3</sup>	23	Set sector	Sector address 2	40	CC	1	Wait for segment to come into position again
7 <sup>2,3</sup>	03	NOP		60	CC, SILI	1	
8 <sup>2</sup>	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for segment
9 <sup>2</sup>	08	TIC	CCW8	00			Repeat search if segment not found
10 <sup>2</sup>	0E	Read key and data		30	SILI, SKIP	256	Check the updated record for errors—write-validity-check

**Notes:**

1. CCW4 is omitted if either the DCBKEYLE or DECKYADR field is 0.
2. CCWs 6 through 10 are present and the command-chain bit in CCW5 is on only when the DCBOPTCD field of the DCB specifies the write-validity-check option.
3. This CCW is set to NOP for devices without the rotational position sensing feature.



C. CHANNEL PROGRAM AFTER WRITING THE FIRST SEGMENT

CCW4 (when present) has changed to TIC to CCW5. This was the CCW to write the key, but keys are not necessary for subsequent segments. This change was made by the ASI routine in IGG019KJ.

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
1 <sup>1</sup>	23	Set sector	Sector=0	40	CC	1	Wait for beginning of track
1 <sup>1</sup>	03	NOP		60	CC, SILI	1	
2	31	Search ID equal	I0BSEEK+3	40	CC	5	Search for segment ID
3	08	TIC	CCW2	00			Repeat search if ID not found
4	08	TIC	CCW5	00			When found, go to update segment
5	05	Write data	Segment work area	00 40 <sup>2</sup>	(CC)	Data length	Update the segment
6 <sup>1,2</sup>	22	Read sector	Sector address 2	40	CC	1	Note its sector value
6 <sup>1,2</sup>	03	NOP		60	CC, SILI	1	
7 <sup>1,2</sup>	23	Set sector	Sector address 2	40	CC	1	Wait for segment to come into position again
7 <sup>1,2</sup>	03	NOP		60	CC, SILI	1	
8 <sup>2</sup>	31	Search ID equal	I0BSEEK+3	40	CC	5	Search for updated segment
9 <sup>2</sup>	08	TIC	CCW8	00			Repeat search if segment not found
10 <sup>2</sup>	0E	Read key and data		30	SILI, SKIP	256	Check the record for errors—write-validity-check

Notes:

1. This CCW is set to NOP for devices without the rotational position sensing feature.
2. CCWs 6 through 10 are present and the command-chain bit in CCW5 is on only when the DCBOPTCD field of the DCB specifies the write-validity-check option.

12. CHANNEL PROGRAM FOR READING VARIABLE-LENGTH SPANNED RECORDS BY BLOCK KEY (TYPE DK)

A. ORIGINAL CHANNEL PROGRAM AS DEVELOPED BY IGG019KR

CCWs 5 through 9 are not used in this pass.

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
1	12	Read count	IOBDNRCF+2	60	CC, SILI	5	Read CCHHR for feedback
2	29	Search key equal	Contents of DECKYADR	60	CC, SILI	Key length	Search for given key
3	08	TIC	CCW1	00			Update feedback and search again if key not found
4	06	Read data	Segment work area	00		Data length	Read the segment
5 <sup>1,2</sup>	23	Set sector	Sector=0	40	CC	1	Wait for beginning of track
5 <sup>1,2</sup>	03	NOP		60	CC, SILI	1	
6 <sup>1</sup>	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for capacity record
7 <sup>1</sup>	08	TIC	CCW6	00			Repeat search if record not found
8 <sup>1</sup>	06	Read data	Segment work area	60	CC, SILI	5	Read capacity record for "address" feedback
9 <sup>1</sup>	03	NOP		20	SILI	1	

**Notes:**

1. CCWs 5 through 9 are present only when "next address" feedback is specified.
2. CCW 5 is set to NOP for devices without the rotational position sensing feature.

**B. CHANNEL PROGRAM AFTER READING THE FIRST SEGMENT**

CCWs 5 through 9 are not used in this pass.

CCWs 1 through 3 of Part A have been changed to search on the block ID for a subsequent segment. The change was made by the ASI routine in IGG019KJ.

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
1 <sup>1</sup>	23	Set sector	Sector=0	40	CC	1	Wait for beginning of track
1 <sup>1</sup>	03	NOP		60	CC,SILI	1	Wait for beginning of track
2	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for next segment
3	08	TIC	CCW2	00			Repeat search if segment not found
4	06	Read data	Segment work area	00		Data length	Read the segment
5 <sup>1,2</sup>	23	Set sector	Sector=0	40	CC	1	Wait for beginning of track to come into position again
5 <sup>1,2</sup>	03	NOP		60	CC, SILI	1	
6 <sup>2</sup>	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for track capacity record
7 <sup>2</sup>	08	TIC	CCW6	00			Repeat search if record not found
8 <sup>2</sup>	06	Read data	Segment work area	60	CC, SILI	5	Read the capacity record
9 <sup>2</sup>	03	NOP		20	SILI	1	

**Notes:**

1. This CCW is set to NOP for devices without the rotational position sensing feature.
2. CCWs 5 through 9 are present only when "next address" feedback is specified.

**C. CHANNEL PROGRAM AFTER READING THE LAST SEGMENT**

This channel program is contained in the channel programs shown in Parts A and B, but it was not executed with them. It is executed independently, after the last segment has been read, in order to provide the disk address of the next record when "next address" feedback has been requested in the READ macro instruction.

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
5 <sup>1</sup>	23	Set sector	Sector=0	40	CC	1	Wait for beginning of track
5	03	NOP		60	CC, SILI	1	
6	31	Search ID equal	I0BSEEK+3	40	CC	5	Search for capacity record
7	08	TIC	CCW2	00			Repeat search if record not found
8	06	Read data	Segment work area	60	CC, SILI	5	Read the next address
9	03	NOP		20	SILI	1	(This command code serves as switch for ASI routine)

**Note:**

1. This CCW is set to NOP for devices without the rotational position sensing feature.

13. CHANNEL PROGRAM FOR WRITING VARIABLE-LENGTH SPANNED RECORDS (TYPE DK)

A. ORIGINAL CHANNEL PROGRAM AS DEVELOPED BY IGG019KR

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
1	12	Read count	I0BDNRCF+2	60	CC, SILI	5	Read CCHHR for internal feedback
2	29	Search key equal	Contents of DECKYADR	60	CC, SILI	Key length	Search for specified key
3	08	TIC	CCW1	00			Update feedback and search again if key not found
4	06	Read data	Segment work area	60	CC, SILI	8	Read segment descriptor words (BDW/SDW)
5 <sup>1</sup>	22	Read sector	Sector address 2	40	CC	1	Note segment's sector value
5 <sup>1</sup>	03	NOP		20	SILI	1	
6	31	Search ID equal	I0BSEEK+3	40	CC	5	This CCW not used in this pass
7	08	TIC	CCW6	00			This CCW not used in this pass
8	0E	Read key and data		30	SILI, SKIP	256	This CCW not used in this pass

**Note:**

1. This CCW is set to NOP for devices without the rotational position sensing feature.

**B. CHANNEL PROGRAM AFTER READING THE COUNT FIELD OF THE FIRST SEGMENT TO DETERMINE ITS LENGTH**

CCWs 1 through 4 of Part A have been changed to write the updated segment and validity check it when necessary. All segments will now be written by searching on a known ID. The changes to the channel program were made by the ASI routine in IGG019KJ.

CCW No.	Cmd. Code Hex	Cmd. Code Descr.	Address	Flags Hex	Descr.	Count	Comments
1 <sup>1</sup>	23	Set sector	Sector address 1	40	CC	1	Wait for positioning found on last pass
1 <sup>1</sup>	03	NOP		60	CC, SILI	1	
2	31	Search ID equal	I0BSEEK+3	40	CC	5	Search for segment
3	08	TIC	CCW2	00			Repeat search if segment not found
4	05	Write data	Segment work area	00 40 <sup>2</sup>	(CC)	Segment length	Update segment when found
5 <sup>1,2</sup>	23	Set sector	Sector address 1	40	CC	1	Wait for updated segment to come into position again
5 <sup>1,2</sup>	03	NOP		60	CC, SILI	1	
6 <sup>2</sup>	31	Search ID equal	I0BSEEK+3	40	CC	5	Search for updated segment
7 <sup>2</sup>	08	TIC	CCW6	00			Repeat search if segment not found
8 <sup>2</sup>	0E	Read key and data		30	SILI, SKIP	256	Check the record for errors

**Notes:**

1. This CCW is set to NOP for devices without the rotational position sensing feature.
2. CCWs 5 through 8 are present and the command-chain bit in CCW4 is on only when the DCBOPTCD field of the DCB specifies the write-validity-check option.

## GLOSSARY OF TERMS AND ABBREVIATIONS

The following terms are defined as they are used in this book or in BDAM source code. If you do not find the term you are looking for, refer to the index or to the IBM Vocabulary for Data Processing, Telecommunications, and Office Systems, GC20-1699.

**actual address.** A pattern of characters that, without further modification, identifies a unique storage location. The format is MBBCCHHR.

**actual extent.** An area in the DEB containing data that describes the space occupied by an extent of a data set. BDAM module IGG0193A builds one actual extent for each extent in the data set.

**ASI.** Asynchronous interrupt.

**AVT.** Appendage vector table.

**BCB.** Buffer control block.

**BDAM.** Basic direct access method.

**BDW.** Block descriptor word.

**blkref field.** A field the user specifies in a program and that contains either the relative or the actual address of the record the user wants access to. If it is the relative address, the BDAM address conversion routines convert it to an actual address (MBBCCHHR). The actual address is then placed in the IOBSEEK field of the IOB so that the channel program can use the address to find a block. The address of the blkref field is in the block address operand of the READ or WRITE macro.

**block position feedback.** A user-specified option that causes the system to put the actual or relative address of the block just read or written into the area specified in the block address operand of the READ or WRITE macro. The format of the address will be MBBCCHHR if feedback was not specified in the DCB macro; otherwise, the format will be the same as the addressing scheme in the DCB macro.

**BPAM.** Basic partitioned access method.

**BSAM.** Basic sequential access method.

**buffer pool.** A continuous area of virtual storage divided into buffers.

**capacity record.** The first block (block 0) on each track of a data set. It contains the ID of the last block on the track and the number of usable bytes remaining on the track.

**CCW.** Channel command word.

**DADSM.** Direct-access device space management.

**DCB.** Data control block.

**DEB.** Data extent block.

**DECB.** Data event control block.

**DSCB.** Data set control block.

**dummy record.** A record, created when BSAM builds a BDAM data set containing format F records, whose purpose is to provide space in which new records can be added to the data set after it is created. The first byte in the key field of the dummy record contains X'FF', and the first byte in the data field has a value indicating the position of the dummy record on the track (the R in MBBCCHHR).

**DVCT.** Device characteristics table.

**dynamic buffering.** A user-specified option that requests that the system handle acquisition, assignment, and release of buffers.

**ECB.** Event control block.

**EOV.** End of volume.

**ERP.** Error recovery procedure.

**exclusive control.**

- When specified by the user, exclusive control requests that the system prevent the data block about to be read from being modified by other requests. When requested by the user, exclusive control is specified in a READ macro and released in a WRITE or RELEX macro.
- When a WRITE-add request is about to be processed, the system automatically gets exclusive control of either the data set or the track (all other WRITE-add requests). The purpose of getting exclusive control is the same in both of these cases: to prevent multiple WRITE-add requests from updating the same dummy record or writing over the same available space on a track.

**exclusive control list.** An area of storage containing the UCB address and actual address of resources under exclusive control, and the addresses of the first and last IOBs for requests waiting to get exclusive control of that resource. This list is described in

**Contains Restricted Materials of IBM  
Licensed Materials — Property of IBM**

detail in the "Data Areas" section of this manual.

**extended search.** A user-specified option that requests that the system search for the specified block or a place in which to add a new block, starting with the first block on the track containing the block address operand specified in the request macro, and continuing either for as many tracks or blocks (rounded up to a complete track) as are specified in the request macro, or until the search ends successfully.

Extended search is only applicable if relative addressing is being used.

**extent.** A continuous area of space on a direct-access device occupied by or reserved for a data set or part of a data set.

**F.** Fixed.

**feedback.** See block position feedback or next address feedback.

**format channel program.** A channel program that writes a new record to an already existing data set. See also preformat channel program and self-format channel program.

**GTF.** Generalized trace facility.

**I/O.** Input/output.

**IOB.** Input/output block.

**IOB buffer queue.** A queue containing the addresses of IOBs for requests for which a buffer is not available. The BCB contains the addresses of the first and last IOB in this chain, and the IOBDQPTR field in each IOB in the chain contains the address of the next IOB. This queue is used only when real storage is specified for a task.

**IOS.** Input/output supervisor.

**IRB.** Interrupt request block.

**next address feedback.** A user-specified option that causes the system to put the relative address (TTR) of the next data or capacity record into the area specified in the next address operand of the READ or WRITE macro. (If the type operand in the READ or WRITE macro terminated with an R, the address of the next data record is returned; if it terminated with an RU, the address of the next data or capacity record is returned, whichever occurs first.)

Next address feedback is only applicable for operations involving format VS records.

**O/C/EOV.** Open/close/end-of-volume.

**period.** A group of tracks in which the first track does not begin with an overflow block, and the last track does not contain a block that overflows to another track.

**preformat channel program.** A channel program that writes a new format F record to an already existing data set.

**processing program.** Any program that is not a control program; synonymous with problem program.

**PSW.** Program status word.

**READ-exclusive request.** A READ request specifying that exclusive control should be acquired for the record about to be read.

**relative address.** The position of a block in a data set relative to the first block of a data set. The relative address can be a relative track number or relative block number. See "relative track address" and "relative block address."

**relative block address.** A 3-byte binary number that indicates the position of a block in relation to the first block of a data set. The first block of a data set always has a relative block address of 0.

**relative extent.** An area in the DEB containing the number of blocks in each extent and the number of blocks in each track (if track overflow is not in effect) of a data set. Module IGG0193A builds the relative extent area when relative block addressing is specified in the processing program.

**relative track address.** A 3-byte binary number in the form TTR where:

**TT** is the position of the track relative to the first track of a data set. The first track has a relative position of 0.

**R** is the number of the block relative to the first block on the track TT. The first block of data on a track has a relative value of 1.

**RPS.** Rotational position sensing.

**R/TM.** Recovery/termination manager.

**RO.** Track capacity record.

**SAM.** Sequential access method.

**SDW.** Segment descriptor word.

**SDWA.** STAE diagnostic work area.

**search argument.** The field of a block that contains data identifying the block as unique from any other block in the



**data set.** Can be either the key field or the block ID in the count field.

**search limit.** The track following the last track that should actually be searched in a data set. The search limit is calculated and put in the IOBUPLIM field of the IOB when the DCB specifies the extended search option.

**self-format channel program.** A channel program that writes a new format U, V, or VS record to an already existing data set.

**SIO appendage.** Start I/O appendage.

**subroutine identification.** The 2 low-order bytes of each module's unique 8-byte name.

**SVC.** Supervisor call.

**SVRB.** Supervisor request block.

**SWA.** Segment work area.

**TCB.** Task control block.

**track overflow.** A user-specified option that will allow a format F record whose space requirements exceed the space remaining on the track to be partially written on that track and completed on the next track.

**U.** Undefined.

**UCB.** Unit control block.

**unposted queue.** A queue of IOBs that are waiting to get exclusive control of a resource currently under exclusive control. The unposted queue contains

only IOBs for the current task. (This list is described in detail under "Data Areas.")

**unscheduled list.** An area of virtual storage containing the addresses of IOBs for requests for which a buffer is not available. Used only when virtual storage is specified for a task. (This list is described in detail under "Data Areas.")

**Update channel program.** A channel program that reads or writes data for purposes other than adding a new block to an existing data set.

**USL.** Unscheduled list.

**V.** Variable.

**VS.** Variable spanned.

**WRITE-add request.** A request to write a new block to the data set.

**WRITE-release request.** A WRITE-update request that specifies exclusive control should be released for the record about to be written.

**WRITE-update request.** A request to write an already existing block to the data set.

**write-validity check.** A user-specified option that causes the system to verify the accuracy of any information written by the channel program.

**WTG.** Where to go.

**WTP.** Write to programmer.

INDEX

**A**

ABEND  
  001 51  
  235 54  
  239 54  
abnormal completion 27, 28, 38-41  
abnormal end routine 27  
abnormal termination  
  associated with FREEDBUF 55  
  caused by system or user error 56  
  cleaning up after abnormal  
  termination 11  
  during FREEDBUF or RELEX SVCs 4  
  during the exclusive control SVC 43  
  in CHECK macro processing 51  
  in dynamic buffering processing 49  
  in exclusive control processing 43  
  in FREEDBUF recovery routine 55  
  in open processing 15  
  in recovery routine processing 54-56  
actual addressing 9, 132  
actual extent 132  
address conversion  
  calculations involved in 95-98  
  overview 9  
  processing 20-21  
allocating buffers dynamically 9  
AMDPRDMP service aid 91  
appendage  
  channel end/abnormal end 24, 27  
  start I/O 24-26  
ASI routine  
  for nonspanned records 28-31  
  for spanned records 34-37  
  overview 8, 10  
assign buffer; handle I/O interrupts  
  (appendage processing) 24

**B**

base routine 8, 18  
basic direct access method 1  
BCB (buffer control block) 60  
BDW (block descriptor word) 2  
blkref field 132  
block descriptor word 2  
block identification 2  
block position feedback  
  explained 3  
  in ASI routines 28-31, 36  
BSAM (basic sequential access method) 1  
buffer control block 60-62  
buffer pools 62-64  
buffers 24  
  assignment 9  
  release 10, 48-49

build channel program 9, 22

**C**

capacity record 1-2, 132  
channel end condition 26  
channel end/abnormal end  
  appendage 25-27  
channel program generation 9, 22-23  
channel programs 99-126  
channel status word 35, 37  
CHECK macro 50-51  
checking for request completion 11  
CIRB macro 15  
cleanup routine 54-57  
CLOSE macro 11, 52-53  
close processing 11, 52-53  
completion codes  
  condition 28, 31, 35, 36  
  in ASI routines 30-31, 36  
  in base routine 18  
  in exclusive control processing 47  
  list of 87  
completion, abnormal 27, 28, 38-41  
contents supervisor routine 55  
control blocks  
  dummy 43, 47  
  list of 60-85  
conversion, address  
  See address conversion  
CSW (channel status word) 31, 35, 37  
current request 46

**D**

data areas 60-86  
data control block 65-70  
data event control block 72-74  
data extent block 70-72  
data record 1-2  
DCB (data control block) 65-70  
DCB macro 3, 19  
DEB (data extent block) 70-72  
DECBC (data event control block) 72-74  
DEQ macro 44, 53, 56  
dequeuing 38-41, 52-53  
device error 27  
diagnostic aids 87-92  
DMABCOND macro 15  
dummy control blocks 43  
dummy record 1-2  
dynamic buffering  
  buffer assignment 24-26  
  buffer release 48-49  
  explained 3, 10, 132  
  in ASI routines 30-31, 36-37  
  in open processing 15

work area 56, 91

**E**

end-of-data condition 25-27  
end-of-extent  
  See also extent  
  appendage 25  
  condition 28, 29, 35  
ENQ macro 56  
enqueueing 38-41  
EOV routine 11, 15, 51  
ERP (error recovery procedure) 27  
error information recorded in the SDWA  
  on SYS1.LOGREC 87  
error messages 87  
errors  
  device 27  
  user 54-56  
ESTAE macro 43, 49, 55  
exclusive control  
  explained 3, 10  
  getting and releasing 42-47  
  recovery routine 54-56  
exclusive control list 74  
EXCP macro 19, 24-27, 29, 33, 34-47,  
  48-49, 56  
exit effector routine 26-27  
extended search option  
  explained 3, 10  
  in address conversion 20-21  
  in channel program generation 22-23  
  when end-of-data condition  
    detected 25  
  when end-of-extent reached 25  
  with format U or V records 33-47  
  with format VS records 38, 49  
extent  
  actual 132  
  periods of an 93-94

**F**

feedback  
  explained 3  
  in address conversion 21  
  in ASI routine 28, 29, 36-37  
  with RELEX requests 42, 43  
force close executor 15, 53  
format F record 1, 3  
format U record 1  
format V record 1  
format VS record 1, 3  
free system resources after abnormal  
  termination 54  
FREEDBUF  
  macro 4, 9, 11, 31, 37, 48-49  
  recovery routine 54-56  
freeing a buffer 9

**G**

get or release exclusive control 10  
GETMAIN macro 19, 29, 35  
graphic symbols 12  
GTF data set 56, 91-92  
GTRACE  
  macro 56, 91

**I**

I/O request processing 8  
I/O supervisor 53  
ICBACREL macro  
  to allocate and stage space on  
  DASD 15  
IEAVVMSR 55  
IEAOVL00 55  
IEAOVL01 55  
IECPCNVT 21, 43  
IECPRLTV 21, 35, 47  
IFCEREPO service aid 87  
IGG020T1 15, 53  
IGxxxxxx  
  See module directory  
incorrect length condition 25-27  
input/output block 75-83  
IOB (input/output block) 75-83

**L**

link pack area 1

**M**

MACRF field 15  
macros 1  
Mass Storage System 15  
messages 87  
method of operation 12  
module directory 58  
MSS  
  check during open 15

**N**

next address feedback 3, 34, 35

**O**

O/C/EOV 4, 51  
OPEN macro 15  
Open processing 8, 15, 53  
operating system, relationship to  
  BDAM 4  
OPTCD parameter 19  
options 3  
overview of BDAM processing 4

P

periods of an extent 93-94  
permanent error 27  
problem determination 15, 87  
program check 55  
purge 52-53

R

R/TM 43, 49, 55, 57  
RDXLIST 74  
READ macro 3, 19, 35  
read or write a record—overview 17  
real storage 25, 49, 53  
record  
  capacity (R0) 1-2  
  data 1-2  
  dummy 2  
  format F, U, V, or VS 1  
recovery routine 54-56  
recovery/termination manager 43, 49,  
  55, 56, 57  
relative addressing option  
  conversion process 95-98  
  explained 3  
  in exclusive control processing 46,  
  47  
  in open processing 15, 18-19  
  with RELEX requests 42  
RELEX macro 3, 10, 42-43  
R0 3

S

scheduling further processing 10  
SDUMP macro 55  
SDW 2, 35  
SDWA (STAE diagnostic work area) 55-57  
search limit 33-39, 134  
service aid  
  AMDPRDMP 91  
  IFCEREPO 87  
STAE diagnostic work area 87-90  
supervisor 33, 55  
SVC  
  SVC 16 53  
  SVC 53 29, 33, 35, 37, 39, 41  
  SVC 55 11, 51  
SVC dump 55

switching extents 10  
SYNAD routine 11, 51  
SYNCH macro 43, 47  
system  
  completion codes 87  
  convert routine 21, 35  
  error 54-57  
  generation 1  
  validity check routine 55  
SYS1.DUMP data set 54-55  
SYS1.LOGREC data set 55-57, 87-90  
SYS1.LPALIB 1

T

task supervisor 24-27  
terminology associated with BDAM  
  requests 4  
track overflow 43  
track overflow option 3

U

unposted queue 84  
unscheduled list 84-86  
user error 54-57  
USL (unscheduled list) 84-86

V

virtual storage 15, 25, 49, 53

W

WAIT macro 31, 37  
waiting IOB 38-39  
waiting request 38-39  
where-to-go table 15, 53  
WRITE-add request 4  
WRITE macro 3, 8, 9, 10, 19, 23, 37  
WRITE-release request 4  
WRITE-update request 4  
write-validity check option 3  
WRITE-verify option 3, 9, 22-23  
WTG table 15, 53  
WTP message 54-56

**Contains Restricted Materials of IBM  
Licensed Materials—Property of IBM**  
(Except for Customer-Originated Materials)  
© Copyright IBM Corp. 1974, 1985  
LY26-3913-1

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

*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.*

Note: Staples can cause problems with automated mail sorting equipment.  
Please use pressure sensitive or other gummed tape to seal this form.

List TNLs here:

If you have applied any technical newsletters (TNLs) to this book, please list them here:

Last TNL \_\_\_\_\_  
Previous TNL \_\_\_\_\_  
Previous TNL \_\_\_\_\_

**Fold on two lines, tape, and mail. No postage stamp necessary if mailed in the U.S.A.**  
(Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.) Thank you for your cooperation.

Reader's Comment Form

Fold and tape

Please do not staple

Fold and tape



NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES

**BUSINESS REPLY MAIL**  
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation  
P.O. Box 50020  
Programming Publishing  
San Jose, California 95150



Fold and tape

Please do not staple

Fold and tape

MVS/370 BDAM Logic (File No. S370-30) Printed in U.S.A. LY26-3913-1





MVS/370  
BDAM Logic

Contains Restricted Materials of IBM  
Licensed Materials—Property of IBM  
© Copyright IBM Corp. 1974, 1985  
Order No. LY26-3913-1  
File No. S370-30

LY26-3913-01



Printed in U.S.A.