File Number S360-30
Order Number GY28-6618-5

Program Logic

OS ISAM Logic

Release 21

Program Number 360S–IO–526

This publication describes the program logic of the two indexed sequential access methods: the queued indexed sequential access method (QISAM) and the basic indexed sequential access method (BISAM). It also discusses the relationship of indexed sequential access method routines to other parts of the control program.

Sixth Edition (February 1972)

This is a major revision of, and makes obsolete, the edition of this manual identified as GY28–6618–4.

This edition applies to OS Release 21 and to all subsequent releases until otherwise indicated in new editions or technical newsletters. Changes to the information in this book may be made at any time; before using this publication in connection with the operation of IBM systems, consult the latest *SRL Newsletter,* GN20–0360, for the editions that are applicable and current.

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

Forms for readers' comments are provided at the back of this publication. If the forms have been removed, comments may be addressed to IBM Corporation, Programming Publications, Department D78, Monterey and Cottle Roads, San Jose, California 95114. Comments become property of IBM.

PREFACE

This publication describes the program structure of the two indexed sequential access methods: queued indexed sequential access method (QISAM) and basic indexed sequential access method (BISAM).

The manual is divided into seven sections:

"Section 1: Introduction" is an overview of indexed sequential access method organization and an overall description of ISAM operations.

"Section 2: Method of Operation" comprises four parts:

1. ISAM common open, common close, and validation modules — a discussion of the common processing operations for QISAM scan, QISAM load, and BISAM.
2. Queued Indexed Sequential Access Method, Load Mode — a discussion of the operations and routines unique to creating data sets with QISAM.
3. Queued Indexed Sequential Access Method, Scan Mode — a discussion of the operations and routines involved in retrieving and updating records sequentially using QISAM.
4. Basic Indexed Sequential Access Method — a discussion of the techniques and operations used in the direct storage and retrieval of records in an indexed sequential data set.

"Section 3: Program Organization" contains flowcharts of individual ISAM routines.

"Section 4: Director" contains a table of ISAM modules, by type, and module selection tables for QISAM load mode, open executors, and close executors.

"Section 5: Data Areas" contains descriptions of data management control blocks and work areas used by ISAM.

"Section 6: Diagnostic Aids" summarizes appendage, asynchronous, and exception codes set and used by ISAM routines.

"Section 7: Appendixes" supplements this manual and program listings with a description of ISAM data set organization (Appendix A) and the ISAM channel programs (Appendix B).

Prerequisite Knowledge

Before reading this book, you should understand the material presented under "Processing an Indexed Sequential Data Set" in *OS Data Management Services Guide,* GC26–3746.

Recommended Reading

The following publications contain information that you may need in conjunction with reading this book:

OS DADSM Logic, GY28–6607

OS Data Management Macro Instructions, GC26–3794

OS Data Management for System Programmers, GC28-6550

OS I/O Supervisor Logic, GY28-6616

OS MFT Guide, GC27-6939

OS MVT Guide, GC28-6720

OS Open/Close/EOV Logic, GY28-6609

OS Supervisor Services and Macro Instructions, GC28-6646

OS Sytem Control Blocks, GC28-6628

CONTENTS

iii **Preface**

xiii **Summary of Changes for Release 21**

1 **Section 1: Introduction**
1 Open Phase
2 Processing Phase
5 Close Phase

7 **Section 2: Method of Operation**
9 ISAM Common Open, Common Close, and Validation Modules
9 The ISAM Common Open Executors
12 The Validation Modules
13 Common Close Phase Executors
14 Queued Indexed Sequential Access Method, Load Mode
15 Load Mode Open Phase Operations
15 Initial Load or Reload Open Operations
15 Resume Load Open Operations
16 Full–Track–Index–Write Open Operations
16 The Final Load Mode Open Phase Operations
16 Load Mode Open Phase Organization
17 Initial Load Organization
20 Resume Load Open Organization
22 Full–Track–Index–Write Phase Organization
22 The Final Executors in Load Mode Open Phase Organization
23 Load Mode Processing Phase Operations
24 Put Routine
27 Beginning–of–Buffer Routine
27 End–of–Buffer Routine
28 Full Track–Index–Write
29 Appendages
30 Load Mode Processing Phase Organization
31 Channel Programs
33 Control Blocks and Work Areas
33 Load Mode Close Phase Operations
33 Load Mode Close Phase Organization
35 Queued Indexed Sequential Access Method, Scan Mode
36 Scan Mode Open Phase Operations
37 Scan Mode Open Phase Organization
38 Scan Mode Processing Phase Operations
39 Buffer Control Techniques
42 SETL Routine
43 Get Routine
45 EOB Routine
45 Scheduling Routine
47 PUTX Routine
48 ESETL Routine
49 RELSE Routine
49 Appendages

51 Scan Mode Processing Phase Organization
51 Processing Routines
51 Scan Mode Channel Programs
52 Scan Mode Control Blocks and Work Areas
52 Scan Mode Close Phase
55 Basic Indexed Sequential Access Method
55 BISAM Open Phase Operations
56 BISAM Open Phase Organization
60 BISAM Processing Phase Operations
61 An Example of BISAM Processing Flow
62 Privileged Macro–time Routines
64 Nonprivileged Macro–time Routines
65 Appendage and Asynchronous Routines
67 Dynamic Buffering Routines
68 Check Routine
71 BISAM Processing Phase Organization
71 BISAM Channel Programs
89 BISAM Control Blocks and Work Areas
91 BISAM Close Phase

93 Section 3: Program Organization

139 Section 4: Directory
141 ISAM Module Identified in Alphameric Sequence

147 Section 5: Data Areas
149 ISAM Control Blocks and Data Areas
149 Data Control Block (DCB)
159 Data Event Control Block (DECB)
160 Data Set Control Block (DSCB)
167 Data Extent Block (DEB)
170 Input/Output Block (IOB)
172 Buffer Control Block (BCB)—BISAM
174 Buffer Control Block (BCB)—QISAM
175 Buffer Control Table (IOBBCT)
179 QISAM Load Mode DCB Work Area
187 QISAM Scan Mode DCB Work Area
193 BISAM DCB Work Area
196 QISAM Track–Index Save Area
198 ISAM DCB Field Area

199 Section 6: Diagnostic Aids
201 Appendage Codes
201 QISAM Scan Mode Appendage Codes
201 BISAM READ and WRITE K Appendage Codes
201 BISAM WRITE KN Appendage Codes
202 Asynchronous Codes
202 BISAM READ and WRITE K Asynchronous Codes
203 BISAM WRITE KN Asynchronous Codes
204 Exception Codes
204 QISAM Exception Codes
205 BISAM Exception Codes

207 Section 7: Appendixes
209 Appendix A: ISAM Data Set Organization
209 Introduction
209 Data Set Structure
211 Prime Data Area
211 Index Areas
213 Adding Records to a Data Set
215 Detailed Index Description
222 Appendix B: ISAM Channel Programs

285 Index

ILLUSTRATIONS

Figures

3	Figure 1.	SIO Appendage for ISAM RPS
10	Figure 2.	ISAM Open Flow of Control
11	Figure 3.	RPS Identification Field in the Data Event Block
14	Figure 4.	Flow of control through the Close Executors
17	Figure 5.	Flow of Control through Load Mode Open Executors
25	Figure 6.	Load Mode Put Routine
26	Figure 7.	Load Mode BOB Routine
27	Figure 8.	Load Mode EOB Routine
28	Figure 9.	Load Mode Channel–end Appendage Routine
29	Figure 10.	Load Mode Abnormal–end Appendage Routine
30	Figure 11.	Load Mode Processing Modules
31	Figure 12.	QISAM—Load Mode Channel Program Flow (Fixed–Length Records)
32	Figure 13.	QISAM—Load Mode Channel Program Flow (Variable–Length Records)
34	Figure 14.	Load Mode Control Blocks and Work Areas
36	Figure 15.	The Flow of Control through QISAM Load Mode Close Executors
38	Figure 16.	Flow of Control through Scan Mode Open Executors
40	Figure 17.	Scan Mode Channel Program/Buffer Queues
40	Figure 18.	Buffer Queuing and Movement in Scan Mode
43	Figure 19.	Scan Mode SETL Routine
44	Figure 20.	Scan Mode GET Routine
46	Figure 21.	Scan Mode EOB Routine
47	Figure 22.	Scan Mode Scheduling Routine
48	Figure 23.	Scan Mode ESETL Routine
51	Figure 24.	QISAM Scan Mode Processing Modules
53	Figure 25.	Scan Mode Channel Program
54	Figure 26.	Scan Mode Control Blocks and Work Areas
56	Figure 27.	BISAM Open Executors
59	Figure 28.	Flow of Control through BISAM Open Executors
62	Figure 29.	Privileged Macro–time Routines
64	Figure 30.	Nonprivileged Macro–time Routines
66	Figure 31.	BISAM Appendage and Asynchronous Routines
67	Figure 32.	Dynamic Buffering Routines
68	Figure 33.	BISAM Check Routine
69	Figure 34.	BISAM Processing Flow (Not WRITE KN)
70	Figure 35.	BISAM Privileged Macro–time Modules
71	Figure 36.	BISAM Nonprivileged Macro–time Modules
71	Figure 37.	BISAM Asynchronous Modules
72	Figure 38.	BISAM Appendage Modules
73	Figure 39.	BISAM Channel Program Modules
76	Figure 40.	READ K, WRITE K, READ KU Channel Program Flow
77	Figure 41.	WRITE KN Channel Program Flow—Index Searching
78	Figure 42.	WRITE KN Channel Program Flow—Add to Prime (Fixed–Length Unblocked Records, System Work Area)

79 Figure 43. WRITE KN Channel Program Flow—Add to Prime (Fixed–Length Unblocked Records, User Work Area)
80 Figure 44. WRITE KN Channel Program Flow—Add to Prime (Fixed–Length Blocked Records, System Work Area)
81 Figure 45. WRITE KN Channel Program Flow—Add to Prime (Fixed–Length Blocked Records, User Work Area)
82 Figure 46. WRITE KN Channel Program Flow—Add to Prime (Variable–Length Records, System Work Area)
83 Figure 47. WRITE KN Channel Program Flow—Add to End (Fixed–Length Records, System Work Area)
84 Figure 48. WRITE KN Channel Program Flow—Add to End (Fixed–Length Records, User Work Area)
85 Figure 49. WRITE KN Channel Program Flow—Add to End (Variable–Length Records)
86 Figure 50. WRITE KN ChanneL Program Flow—Add to Overflow (Fixed–Length Records, System Work Area)
87 Figure 51. WRITE KN Channel Program Flow—Add to Overflow (Fixed–Length Records, User Work Area)
88 Figure 52. WRITE KN Channel Program Flow—Add to Overflow (Variable–Length Records)
89 Figure 53. Elements of a BISAM Request
90 Figure 54. BISAM Control Blocks and Processing Modules
91 Figure 55. BISAM Work Areas and Queues
142 Figure 56. ISAM Modules Identified by Function and Mode
143 Figure 57. ISAM Modules Identified by Alphameric Sequence
150 Figure 58. BISAM/QISAM DCB
159 Figure 59. Data Event Control Block
161 Figure 60. Format–2 DSCB
166 Figure 61. ISAM Extention to DEB
170 Figure 62. ISAM Extension to IOB
172 Figure 63. Fields of the BISAM Dynamic Buffering BCB
174 Figure 64. Fields of the QISAM BCB
175 Figure 65. QISAM Load Mode Buffer Control Table
180 Figure 66. QISAM Load Mode DCB Work Area
186 Figure 67. Area Y: QISAM Load Index Fields
187 Figure 68. QISAM Scan Mode DCB Work Area
193 Figure 69. BISAM Work Area
196 Figure 70. Track–Index Save Area
197 Figure 71. TISA Control Fields
198 Figure 72. DCB Field Area
204 Figure 73. QISAM Exception Code Summary
205 Figure 74. BISAM Exception Code Summary
210 Figure 75. Indexed Sequential Data Set Structure
211 Figure 76. Initial Structure of Prime Cylinder
212 Figure 77. Structure of Cylinder Index and Track Index
213 Figure 78. Structure of Prime Cylinder after Cylinder Overflow
215 Figure 79. Structure of Prime Cylinder after Independent Overflow
216 Figure 80. Format of ISAM Index Entry
219 Figure 81. Description of Track Indexes
220 Figure 82. Description of Cylinder Indexes
221 Figure 83. Description of Master Indexes
223 Figure 84. ISAM Channel Program Summary

Flowcharts

95	Chart AA	First Common Open Executor (IGG0192A)
98	Chart AB	Second Common Open Executor (IGG0192B)
100	Chart AC	Third Common Open Executor (IGG0192C)
101	Chart AD	Fixed–length Validation Open Executor (IGG01920)
102	Chart AE	First Load Mode Open Executor (IGG0192I)
105	Chart AF	First Initial Load Mode Open Executor (IGG0192D)
108	Chart AG	First Resume Load Open Executor (IGG0196D)
109	Chart AH	Last Scan Mode Open Executor (IGG01924)
110	Chart AI	First Scan Mode Open Executor (IGG01928)
114	Chart AJ	ISAM Common Close Executor Module (IGG0202D)
116	Chart AK	QISAM Scan Processing Module (IGG019HB)
130	Chart AL	Scan Mode Appendage (IGG019HG)
133	Chart AM	Scan Mode Close Executor Module (IGG02029)
134	Chart AN	BISAM Open Executor—Load Privileged Module (IGG0192I)
136	Chart AP	BISAM Nonprivileged Macro–time Processing— READ K, READ KU, WRITE K (IGG019JV)
137	Chart AQ	BISAM Privileged Macro–time Processing Module (WRITE KN, without Read, and Update) (IGG019JX)

SUMMARY OF CHANGES FOR RELEASE 21

Control Block Changes

Several fields containing addresses of ISAM routines have been moved to the DEB from the DCB and DCB work areas. These changes are for QISAM load and scan modes and for BISAM.

New QISAM Load Mode Open Executors

There are two new open executors for load mode — IGG01925 and IGG01927. They are executed when high-level indexes are created on 2301 and 2321 devices.

Summary of ISAM Modules

A table listing all ISAM modules in alphameric order has been added to the Directory. The table indicates the pages on which each module is described and replaces individual module names in the index.

Summary of ISAM Channel Programs

A table listing all ISAM channel programs has been added to the introductory text of Appendix B.

Miscellaneous Changes

- This manual is to be used with MFT and MVT systems. All information about PCP has been removed.
- New information has been added to channel programs 20 and VXCCW(1A) in Appendix B.
- Technical and editorial corrections have been made throughout the manual.

SECTION 1: INTRODUCTION

The indexed sequential access methods (ISAM) are data management techniques used for storing indexed sequential data sets on direct–access devices, or for retrieving those data sets.

A detailed description of the structure of an indexed sequential data set is provided in Appendix A of this manual. Detailed information on how to create and process an indexed sequential data set is in the publication *OS Data Management Services Guide,* GC26–3746.

ISAM routines are part of the operating system control program. They are grouped into modules that are placed in the supervisor call (SVC) library during system generation. Only the modules needed to perform those functions required by a processing program are loaded into main storage from the system–residence volume. Wherever possible, all processing programs use the same copy of a module.

There are two indexed sequential access methods: queued indexed sequential access method (QISAM) and basic indexed sequential access method (BISAM).

QISAM has routines for two modes: *load mode* routines, which are used to create an indexed sequential data set and to add records to the end of a data set; and *scan mode* routines, which are used to retrieve and update records from a previously created data set.

BISAM routines provide direct storage and retrieval of any logical record by its record key. The BISAM routines also permit records to be updated in place. The BISAM Write–Key–New (WRITE KN) routine provides the user a means of inserting new records into an indexed sequential data set.

Routines within QISAM load mode, QISAM scan mode, and BISAM are divided into three phases of execution: the open phase, the processing phase, and the close phase.

Open Phase

When a data control block (DCB) is opened to process an indexed sequential set, the Open routine gives control to ISAM open executors. (The Open routine is described in *OS Open/Close/EOV Logic,* GY28–6609.)

The ISAM open executors are modules that perform the initial ISAM processing. Open processing is performed in two stages: the first or *common open* stage which is executed for both QISAM and BISAM; and the second or *mode–oriented* stage which is executed by separate open modules for QISAM load mode, QISAM scan mode, and BISAM.

The common open executors receive control from the Open routine of I/O support when it is determined that an indexed sequential access method is to be used. The same executors are used for both QISAM and BISAM. These common open executors determine which mode of ISAM has been specified in the processing program and then select the required ISAM modules from the system–residence library. The common open executors determine storage requirements for the access method routines and also begin the building of control blocks and control lists for subsequent use by the processing and closing phases. When these operations are completed, the common open executors transfer control to the mode–oriented, second–stage open executors.

The common open executors are described in detail in the first part of the Method of Operation section of this manual; the mode–oriented executors are discussed in their respective QISAM and BISAM parts.

Processing Phase

During the processing phase of indexed sequential access method operations, several types of routines are invoked: these include input/output routines (in some cases, both privileged and nonprivileged) and their related channel programs, channel program appendage routines, asynchronous routines, and buffer management routines. Control blocks, work areas, and queues are used by the processing phase routines and by the corresponding channel programs.

When an input or output macro instruction is encountered in the processing program, ISAM routines construct the needed channel programs for processing the data and request the I/O supervisor to schedule those channel programs for execution. If an error occurs during the execution of the channel program, the ISAM appendage and asynchronous routines inform the processing program of the error. In the processing phase of ISAM, buffers are allocated, queued, and scheduled (buffer management); indications of whether or not the channel programs have been executed successfully are given by both the buffer management and appendage routines.

Processing Routines

The ISAM processing routines select and complete the channel programs that store, process, and retrieve an indexed sequential data set. These routines perform various operations and construct different channel programs depending on the characteristics of the data to be processed, the type of macro instruction issued by the processing (user) program, and the indexed sequential access method (or mode) being used.

For QISAM load mode, the primary processing routine is the Put routine. The load mode Put routine is used in creating or resuming the creation (see "Resume Load") of an indexed sequential data set.

In QISAM scan mode, five macro instruction routines are used for data retrieval and updating; the scan mode routines are described under Scan Mode Processing Phase in the Method of Operation section.

The BISAM processing routines consist of several variations of the basic Read and Write routines. In BISAM, both nonprivileged and privileged routines are used to facilitate channel program execution.

The QISAM load, QISAM scan, and BISAM processing routines are described fully in their respective sections of this manual.

Appendage Routines

The appendages are routines entered from the input/output supervisor when a channel program is to be started or when a channel program ends. The appendage routine determines if additional processing is necessary before an input/output operation has started or after it has been completed. For example, more than one channel program may be needed to satisfy completely a specific input or output request from the processing program. In such a case, the channel appendage would keep track of the channel programs needed and assist in initializing and scheduling these channel programs sequentially. Appendages may also schedule asynchronous routines to handle the additional processing of an I/O request. (Appendages and asynchronous routines are described in *OS Data Management for System Programmers,* GC28–6650.)

Rotational Position Sensing Start I/O Appendages

The rotational position sensing (RPS), start I/O (SIO) appendage routines decrease channel time by disconnecting the channel from RPS devices whenever possible. This is done by inserting channel command word (CCW) slots in the various ISAM channel programs.

When an ISAM data set is being used with an RPS device, the RPS start I/O appendages modify the channel command word slots dynamically to either an NOP, Set Sector, Read Sector, or a TIC, depending on the device type and the channel program.

Three RPS SIO appendages are used: one each for QISAM scan and load modes, and one for BISAM. These SIO appendages convert non–RPS channel programs to RPS channel programs and vice versa, as necessary.

Conversion of a non–RPS channel program to an RPS channel program involves:

- Conversion of the CCW slots from TICs or NOPs to Read or Set Sectors
- Possibly modifying a CCW's command–chaining flag so that the RPS CCWs are executed
- Interposing an RPS channel program prefix when the channel program starts with a search ID of five bytes
- Setting up sector values where necessary

Note: The rotational position sensing (RPS) devices referred to in this manual are the IBM 3330 and 2305 Direct–Access Storage Devices.

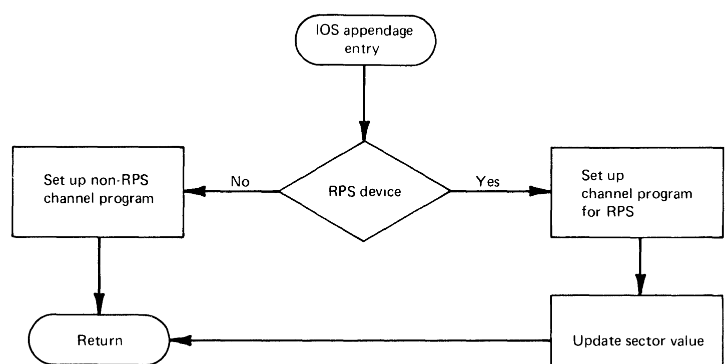


Figure 1. SIO Appendage for ISAM RPS

Asynchronous Routines

Asynchronous routines are used in QISAM scan mode and in BISAM to perform any additional processing of an I/O request required when a channel program ends.

Complete processing of an I/O request may require several channel programs. For BISAM, the asynchronous routines set up and schedule the requests as required. Also, when I/O request processing is complete, whether satisfactorily or in error, the completion must be posted. These routines do the posting. For QISAM scan mode, the asynchronous routine schedules the channel programs when the next record is to be read or written on another device.

The appendage routines of QISAM scan mode and BISAM select and schedule the appropriate asynchronous routines.

Further description of the scan mode asynchronous routines can be found in the discussion of "Appendages" under "Scan Mode Open Phase Operations" in Section 2. For more detail about the BISAM asynchronous routines, see "Appendage and Asynchronous Routines" under "BISAM Processing Phase Operations" in Section 2.

Buffer Handling Routines

Buffer handling or buffer management routines are provided in both modes of QISAM and, optionally, in BISAM.

In QISAM load mode, the Put routine has two subsidiary buffer handling routines: the *beginning–of–buffer* (EOB) routine and the *end–of–buffer* (EOB) routine. The BOB and EOB routines perform both the Put move mode and Put locate mode processing.

In move mode, the Put routine and its buffer handling routines move an output record from the user work area or input area to an output buffer.

In locate mode, the Put routine and its subsidiary routines give the address of an output buffer area to the user; the user must move the record to the buffer.

In QISAM scan mode, five buffer queues are used to control input/output operations. The queuing of buffers is handled primarily by the Get routine and its subsidiary routines—the scheduling routine and the end–of–buffer routine.

In scan mode, a copy of channel program 22 (CP 22) is allocated to each buffer. The buffers are manipulated among the queues and scheduled for I/O operations according to the macro instructions issued in the processing program. Refer to the discussion of "Buffer Control Techniques" under "Scan Mode Processing Phase Operations" in Section 2 for a description of the buffer queues.

Dynamic buffering may be used in BISAM to allow the queuing of multiple read requests. A buffer is automatically acquired from a buffer pool and assigned to the request just before data transfer begins. The buffer is returned automatically to the buffer pool when its contents are written, or it is returned under programmer control with the free dynamic buffer (FREEDBUF) macro instruction. Dynamic buffering requires relatively fewer buffers since the read requests waiting in the queue do not monopolize buffers.

Close Phase

When a DCB for an ISAM data set is closed, the Close routine gives control to ISAM close executor modules which terminate processing for the particular mode of ISAM being used. As do the open executors, the close executors have two stages: (1) the *mode–oriented* stage (that is, the load mode, scan mode, or BISAM close executors), and (2) the *common close* stage executor.

When invoked by the CLOSE macro, the CLOSE routines first determine that an ISAM data set is being processed. They then examine the DCBMACRF field in the DCB to determine which mode of ISAM is in use and which mode–oriented close executor should be given control. The close executors for load mode, scan mode, and BISAM are described in their respective sections.

SECTION 2: METHOD OF OPERATION

ISAM Common Open, Common Close, and Validation Modules

There are three distinct indexed sequential access methods: QISAM load mode, QISAM scan mode, and BISAM. Each comprises a group of modules.

In addition to the three separate groups of modules, certain ISAM modules are used for both QISAM and BISAM processing. In particular, the three common open executor modules (IGG0192A, IGG0192B, and IGG0192C), the common close executor module (IGG0202D), and the validation open executor modules (IGG01920, IGG01922, and IGG01950) are used in both modes of QISAM and in BISAM.

This part of the manual describes the common open and common close executors in detail, and generally describes the validation modules which are further detailed in the discussion of QISAM (load, scan) and BISAM.

The ISAM Common Open Executors

The first stage, or common, open executors receive control from the Open routine. A preexecutor module of Open (module IGG0190W):

1. Reads in the additional DSCBs for this data set (if multivolume)
2. Tests first volume for a format–2 DSCB
3. Checks DSCBs for ascending order on the same sequence in which space was allocated
4. Transfers control (XCTL) to the first ISAM open executor

The common executors, upon completion, pass control to second stage open executors required to initialize the specific form of QISAM or BISAM called for by the processing program.

The DCB Integrity Feature: ISAM routines maintain DCB integrity by preserving pertinent DCB fields and maintaining the current status of these fields during processing. The DCB integrity feature is invoked for the user whenever he opens with DISP=SHR.

This feature prevents multiple tasks, when sharing the same indexed sequential data set, from altering the data set without updating its attributes in the DCB. This could happen if one of the tasks opens the data set for Write–Key–New and modifies an area in order to change various DCB fields. For example, adding records to the last prime–data track would result in updating DCBLPDA and possibly DCBLIOV. Another task with concurrent access to the data set in QISAM scan mode would not process the added records.

With the DCB integrity feature, any change in the DCB caused by a modification of the data set causes a corresponding change in all DCBs currently open for that prticular data set. An ISAM common open module, IGG0192C, determines whether another ISAM data set has previously been opened, and if not, obtains space for a DCB field area (DCBFA) associated with each ISAM data set that is opened. The DCB field area is obtained (by a GETMAIN from subpool 255) by the ISAM open executor module, IGG0192C, when a data set is opened for the first time.

The DCBFA contains the DCB information that can be changed while processing the data set and is pointed to by all DCBs opened for that data set. The DCB fields that require this updating are DCBLIOV, DCBLPDA, DCBNOV, DCBNOREC, DCBNREC, DCBRORG1, DCBRORG2, DCBRORG3, DCBST, and DCBTDC. These fields require a 36–byte DCB field area.

During procesing of a data set opened for WKN or RU, ISAM routines gain access to the associated DCB fields and modify them from the DCBFA. This eliminates the possibility of a user's inadvertently and incorrectly modifying these fields.

The ISAM open executors are each 1024 bytes in length and overlay each other in the transient area.

The three common open executor modules are IGG0192A, IGG0192B, and IGG0192C. The flow of operations among these executors and to the second stage open executors is depicted in Figure 2.

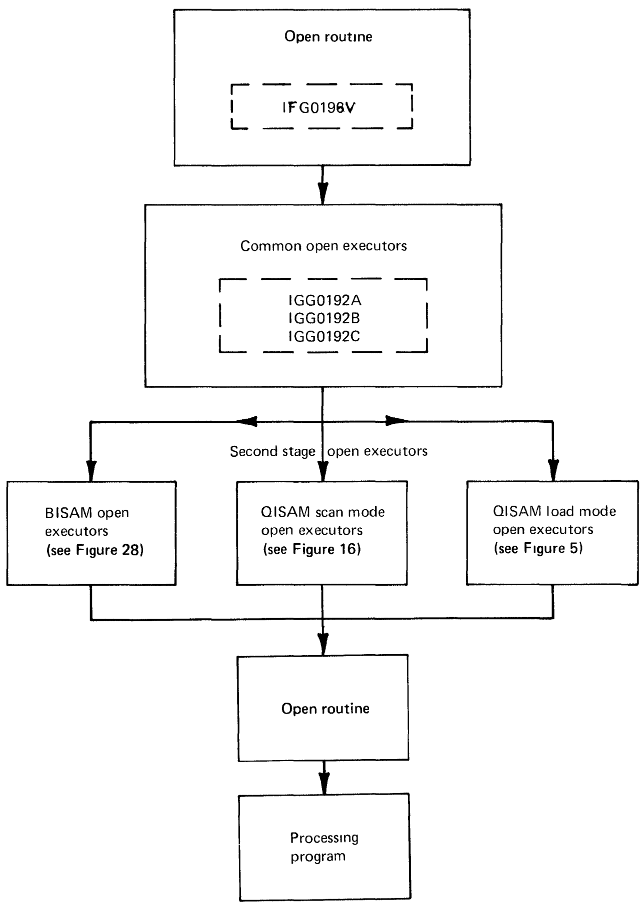


Figure 2. ISAM Open Flow of Control

Note: The second stage open executors return control to the Open routine of I/O support, which returns control to the processing program.

Common open executor IGG0192A receives control from the Open routine of input/output support. The primary functions of IGG0192A are:

1. It calculates the space needed for the DEB. (16 bytes are allocated for the DEB prefix, and 32 bytes for the basic section of the DEB.) The number of extents indicated by the user's data definition statements is picked up from the DSCBs (the data sets allocated must be online). The number of extents, plus 1, is multiplied by 16. Thus, each extent has 16 bytes.
2. It executes a GETMAIN macro instruction for the DEB.
3. It places a pointer to the DEB in the DCB and a pointer to the DCB in the DEB.
4. It sets the pointer to the UCB in each extent (there may be from 1 to 16 extents per volume.) The UCB in each extent points to the direct–access device where the data set (or extent) resides.
5. It checks the devices allocated to the data set to see if these devices have the rotational position sensing (RPS) feature and sets a bit in DSCCW1+4 accordingly. If bit 0, 1, or 2 is on and if the data set is being opened for either QISAM scan mode or BISAM, a count of 1 is added to the module count (DEBNMSUB) in anticipation of loading the necessary RPS start I/O appendage. (See the description of these bits in Figure 3, DEBRPSID.)

After the GETMAIN macro instruction has been performed for the DEB, IGG0192A moves the byte at DXCCW1+4 to DEBISAD in the DEB; the result is that DEBISAD has its high–order byte cleared to 0s if no RPS devices are being used. If RPS devices are being used, the bit is set as shown in Figure 3.

Field	Bit	Setting	Meaning
DEBRPSID	0	1	Prime is on an RPS device
	1	1	Index is on an RPS device
	2	1	Overflow is on an RPS device
	3	1	An SIO appendage has been loaded (set by IGG0192K)

Figure 3. RPS Identification Field in the Data Event Block

Upon completion, IGG0192A transfers control to the common open executor module IGG0192B. The primary functions of IGG0192B are outlined below:

1. IGG0192B uses the DCBBUFNO and DCBBUFL fields (plus 8 bytes for a control field) to develop the buffer pool.
2. It develops the buffer control block (BCB), using DCBBUFNO and DCBBUFL, and uses a GETMAIN from subpool 250 for the BCB space.
3. It also calculates the buffer lengths (using DCBBLKSIZE) and places the calculation in the DCBBUFL field (unless the user sets up his own buffers).
4. The number of buffers (DCBUFNO) field is checked, and if none have been specified, two buffers are allocated for the data set.
5. If the computed buffer length is inadequate, IGG0192B schedules an ABEND with a completion code of hexadecimal 37.

6. IGG0192B then returns to the initialization of the DEB, initializing the extent entries with the address and count fields already established in the DEB.

   The DEB now contains the UCB pointer, the starting addresses of the extents (cylinder, track, and head), and the number of cylinders per extent.

ISAM common open executor IGG0192B passes control to common open module, IGG0192C. The functions of IGG0192C are outlined below:

1. Frees the main storage space occupied by all data set control blocks (DSCBs) except the format–1 and the format–2 DSCBs.
2. Sets the device type fields (DCBDEVT and DCBOVDEV).
3. If the data set can be shared by two or more tasks (as indicated with a DISP=SHR parameter in the JCL), IGG0192C executes a GETMAIN macro instruction from subpool 255 for the DCBFA (DCB field area), unless the DCBFA was previously obtained for this same data set.

The Validation Modules

Modules IGG01920, IGG01922, and IGG01950 are open executors used to validate and maintain DSCB and DCB fields for resume load, scan mode, and BISAM. An initial load (or reload) in load mode does not cause execution of the validation modules.

The operations done in IGG01920, IGG01922, and IGG01950 are described in detail below. Thereafter the validation modules are referred to in the load, scan, and BISAM discussions.

Modules IGG01920 and IGG01922 process fixed–length records and module IGG01950 processes variable–length records.

The validation modules may not be executed, although they are entered if the user has specified that the data set may be shared by other tasks (DISP=SHR). They are not executed in that case because another DCB may have already been opened for the data set and a DCBFA (DCB field area) already set up for the purpose of maintaining the DCB fields.

Open Executor IGG01920

1. Validate and reset, if necessary, the following fields in the format–2 DSCB:
   a. DS2LPRAD—the address of the last record in the prime–data area. This address is in the form MBBCCHHR and subsequently moved to the DCBLPDA field.
   b. DS2PRCTR—the number of records in the prime–data area. This count is later moved to the DCBNREC field.

Open Executor IGG01922

1. Validate and reset, if necessary, the following fields in the format–2 DSCB:
   a. DS2LOVAD—the address of the last record in the current independent overflow area. This address is in the form of an MBBCCHHR address and subsequently moved to the DCBLIO field.
   b. DS2BYOVL—the number of bytes remaining on the current independent overflow track. This count is later moved to the DCBNOV field.
   c. DS2RORG2—the number of tracks remaining in the independent overflow area; subsequently merged into the DCBRORG2 field.
   d. DS2OVRCT—the number of records in all overflow areas; later merged to DCBNOREC.

These fields may be incorrect if the data set was previously closed improperly.

Open Executor IGG01950

1. Validate and reset, if necessary, the following fields in the format–2 DSCB:
   a. DS2LPRAD—the address of the last record in the prime–data area. This address will be in the form MBBCCHHR and subsequently moved to the DCBLPDA field.
   b. DS2LOVAD—the address of the last record in the current independent overflow area. This address will be in the form of an MBBCCHHR address and subsequently moved to the DCBLIOB field.
   c. DS2BYOVL—the number of bytes remaining on the current independent overflow track. This count is later moved to the DCBNOV field.
   d. DS2RORG2—the number of tracks remaining in the independent overflow area; subsequently merged into the DCBRORG2 field.
   e. DS2OVRCT—the number of records in all overflow areas; merged to DCBNOREC.

These fields may be incorrect if the data set was previously closed improperly.

Common Close Phase Executors

Like the open executors, the close executors are 1024 bytes in length and overlay each other in the transient area. The common close executor module is module IGG0202D; its functions are as follows:

1. Obtains main–storage space for the format–2 DSCB.
2. Reads the format–2 DSCB, updates it from the DCB, and writes it back into the volume table of contents (VTOC).
3. If operating with QISAM load mode, frees the main storage used for the load mode work area and channel programs.
4. If initial load, sets bit 2 of the DCB status byte field (DCBST).

The flow of control through the I/O support routines and the stages of ISAM close executors is shown in Figure 4.

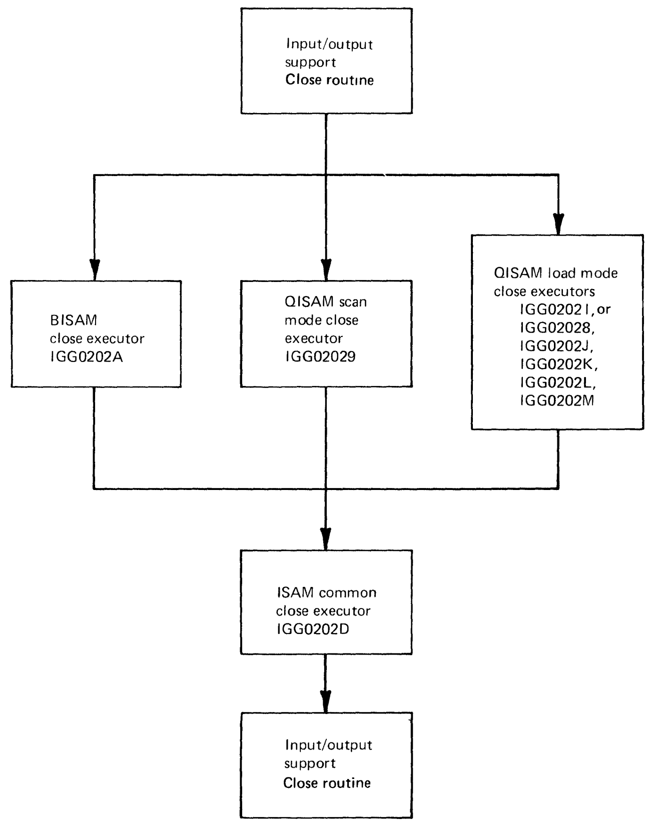


Figure 4. Flow of Control through the Close Executors

Queued Indexed Sequential Access Method, Load Mode

The load mode of QISAM is used to create (or recreate) indexed sequential data sets and may also be used to reopen existing data sets to add records to the end of the prime–data area. Creating a data set is called *initial loading;* recreating one is called *reloading;* and reopening a data set is called *resume loading.* (See *OS Data Management Services Guide,* GC26–3746, for a user–oriented discussion of resume loading.)

Since it is part of the queued access method, load mode handles all required buffering, blocking, and I/O activity synchronization.

There are three groups of QISAM load mode routines:

- The open phase
- The processing phase
- The close phase

The open phase routines include executor modules that perform tasks needed to open a data set, initialize data areas, and prepare to load other routines for the processing phase. The open phase executors receive control from the Open routine. The processing phase routines include the Put routine (which receives control and is executed when a PUT macro instruction is issued in the user's program), appendages, and channel programs. The processing phase routines perform the actual access method functions in QISAM load mode. The close phase routines perform functions essential to closing the indexed sequential data set when all processing phase operations are finished. The close phase routines are executor modules that receive control from the Close routine.

Load Mode Open Phase Operations

There are two stages of ISAM open executors. The first stage executors are entered for all indexed sequential access methods and are the *common open* executors (see Figure 2). The second stage open executors for load mode receive control from the common open executors. These second stage executors perform initialization operations required for load mode processing, whether creating, reloading, or resume loading the data set, with either variable or fixed–length records.

The *second–stage* executor for load mode (module IGG01921) is entered for both initial and resume loading to provide main storage space for the load mode work area. ISLCOMON is the load mode DCB work area and contains the input/output blocks (IOBs), location tables, counters, and various pointers. The load mode processing modules and channel programs refer to and modify the ISLCOMON area.

The IOBs, tables, and pointers in ISLCOMON are used in scheduling, controlling, checking the load mode processing operations, filling the buffers with records, loading these records into the ISAM data set, and referring to these records and their locations in the various ISAM indexes.

Besides obtaining main storage space for an initializing ISLCOMON, the beginning open executor for load mode determines if the user intends to create a new ISAM data set (initial load), to reload an old data set, or to reopen an existing data set.

Initial Load or Reload Open Operations

For the initial load or reload of an ISAM data set, the ISAM load mode open executors structure, allocate space for, and format the prime–data area, the track–index area, and, if specified, the high–level index areas. An initial load open module (IGG0192G) also initializes fields in the ISLCOMON area to be used by the load mode buffering routines.

The initial load or reload open routines of the load mode open executors also determine whether or not the last track of the track index for each cylinder will contain one or more data records, (that is, *shared track).* If there is to be a shared track, temporary records representing each track–index entry (preformat) must be written so the first data records can be written before the actual index entries are developed and written. Refer to the descriptions of modules IGG0192D and IGG0192S in the discussion of "Load Mode Open Phase Organization" for further information on the preformatting of shared tracks.

Resume Load Open Operations

When opening an existing ISAM data set to add records at the end of the prime–data area (resume load), the load mode open executors for resume load must ensure that the addressing control fields for prime, index, and overflow records are accurate and usable for locating the last records in each area and loading additional records into the data set. Control fields for buffering and record–moving logic must be initialized in accordance with the dimensions of the already created data set; this is also done as part of the resume load open operations. (Refer to "Resume Load Open Organization" for further details.)

Full–Track–Index–Write Open Operations

The full–track–index–write feature of load mode allows for accumulating and writing a full track of track–index entries as a group rather than singly (refer to "Appendix A: ISAM Data Set Organization"). The track–index entries are accumulated in the track–index save area (TISA) shown in Section 5. A full track of track index is written into the track–index area of the data set when the TISA is full, when end–of–cylinder is reached, or when the data set is closed.

When the user opens the DCB for load mode and specifies the full–track–index–write option (DCBOPTCD=U), the load mode open phase executors perform operations especially for the initialization of the full–track–index–write feature. These operations include acquiring the track–index save area, and initializing channel program 20 to write the track–index entries from the TISA to the direct–access storage device.

The Final Load Mode Open Phase Operations

The final load mode open phase operations are performed for all load mode open options. The final load mode open executors:

1. Load the needed ISAM load mode modules containing the appropriate routines, appendages, and channel programs.
2. Initialize and execute channel program 19 for preformatting shared track in Area Z of ISLCOMON when required.
3. Initialize channel programs 20 and 21 for writing track–index and high–level index entries.

Load Mode Open Phase Organization

Load Mode Open Executor IGG01921

As indicated in the load mode open operations discussion, the first load mode open executor, module IGG01921, is entered for both initial and resume load. The operations for this module are outlined below.

1. Obtains main–storage space for the load mode work area (ISLCOMON) and sets the work area pointers.
2. Fills in the load mode input/output blocks (IOBs) in ISLCOMON.
3. Determines from the DISP parameter the user's intent to reload the data set; resets the DCB status bits if necessary, and reinitializes the data set in accordance with DCB parameters supplied in the DD statement.
4. Calculates and sets the DCBHIRPD field (highest record that can be written in the prime area) and the DCBHIROV field (highest record of overflow).
5. Determines if track capacity of the independent overflow device is sufficient to contain the maximum length record for an overflow chain (the longest possible record in an overflow chain).
6. Checks the data control block for contradictory specifications; issues an ABEND macro instruction if RKP + key length is greater than LRECL.

Upon completion of module IGG01921, the selection of modules to continue load mode open operations depends on whether initial or resume loading is to take place: this is indicated by Figure 5 which shows the flow of control through the load open executors.

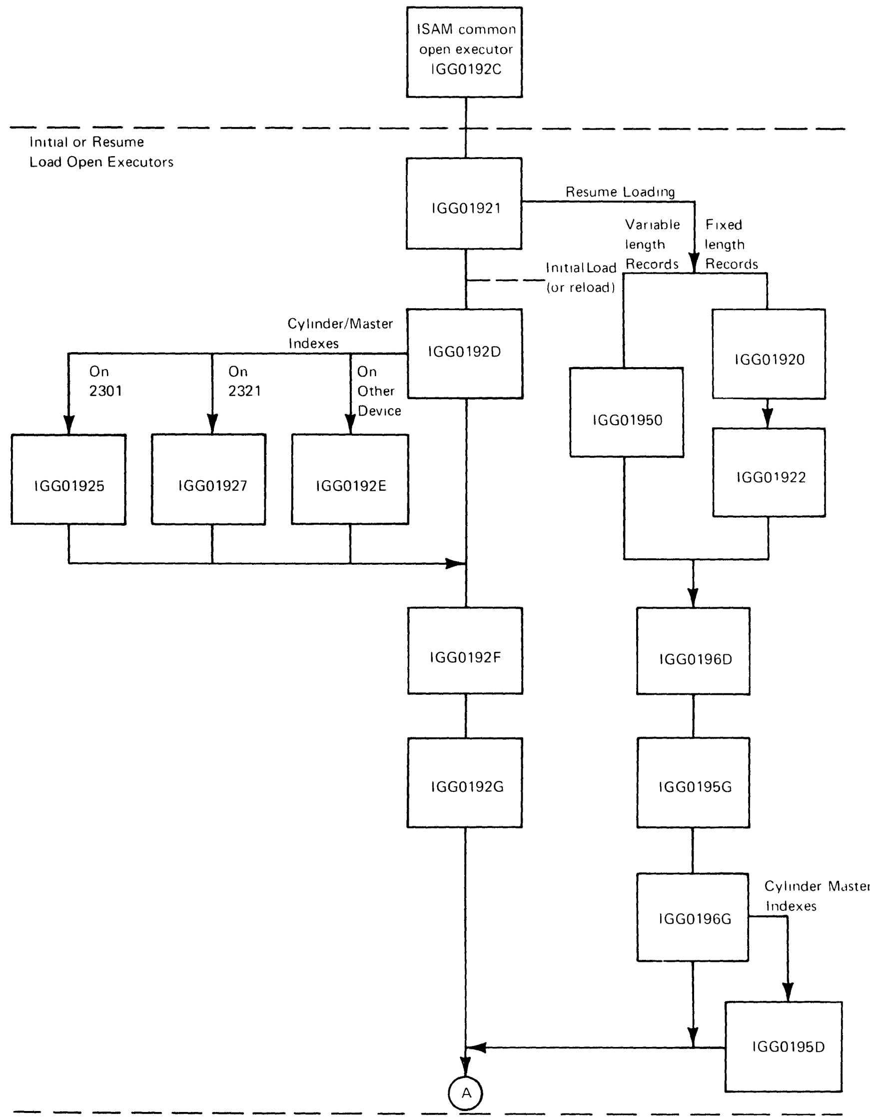


Figure 5 (Part 1 of 2). Flow of Control through Load Mode Open Executors

Initial Load Organization

If an indexed sequential data set is to be created, the first load mode open executor (IGG01921) passes control to module IGG0192D.

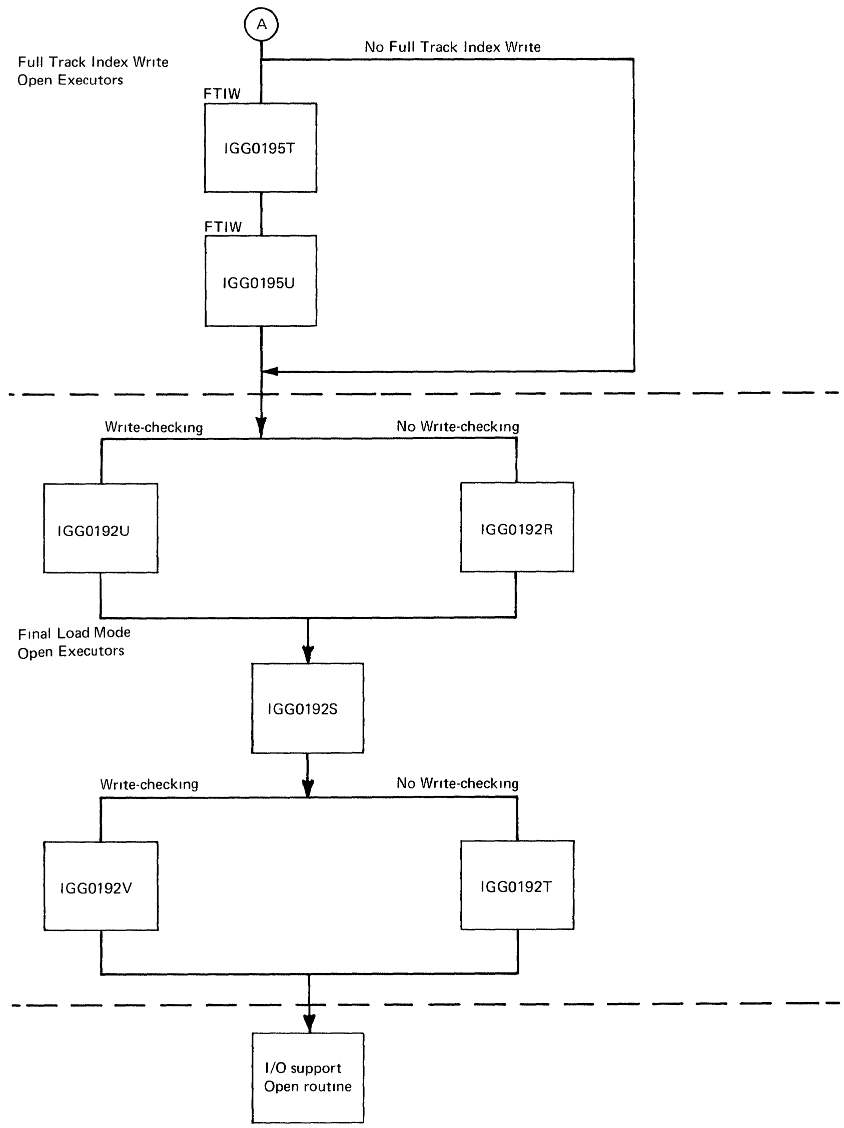


Figure 5 (Part 2 of 2). Flow of Control through Load Mode Open Executors

Load Executor IGG0192D

IGG0192D calculates several control fields needed in load mode processing. Listed below are some of the primary functions performed by module IGG0192D in structuring the prime–data area and calculating various DCB fields needed to allocate direct–access device storage for track, cylinder, and master indexes:

1. Determines if the higher levels of index are to be used and where they are to be located.
2. Determines whether the track index will share a track with prime–data records (shared track).
3. Uses the DEBFIEAD field (indicates if high–level indexes are to be used and set from the user–specified OPTCD parameter in the DCB) to determine whether high–level indexes are to be used. If the user has not specified an independent index area, the DEBNOEE field is used to determine whether an independent overflow area has been specified.
4. Module IGG0192D also sets indicators to specify whether the independent index, the independent overflow, or the prime area is to be used for the high–level indexes when they are requested by the user. The indicators are passed to module IGG0192E, module IGG01925, or module IGG01927 when high–level indexes are required. Module IGG0192D transfers control to module IGG0192F if high–level indexes are not needed.
5. Before transferring control, module IGG0192D establishes several fields in the DCB work area, ISLCOMON, to be used by other open modules.
6. Determines if the last index track can be shared by calculating the number of index entries required per cylinder and dividing by the number of entries that fit on a track, to yield the number of entries on the final track and the portion of the track available for data.
7. If a 3330 device is being used, IGG0192D treats the cylinder value on the device as a halfword. It also refers to the two halfwords, defined in IGG01921 (described previously), rather than to the I/O device table for its track capacity calculations for prime–data records. A similiar field is used during open processing for the analogous calculations on the index device. However, this field is already defined in the DSECT for the QISAM load mode work area and is returned to its normal usage at the completion of open operations. The index backup routine in IGG0192D set bits 1 or 2 of DEBRPSID, if necessary, as does IGG0195D.

The Load Mode Open Executors IGG0192E, IGG01925, and IGG01927

If in the initial loading (creation) or reloading of an ISAM data set, cylinder or master indexes are specified, then executor IGG0192D passes control to module IGG01925 if the indexes are on a 2301 device, module IGG01927 if the indexes are on a 2321 device, or module IGG0192E of the indexes are on any other device. The functions of these executors are outlined below:

1. Structures the high–level indexes, using information from the data fields established by module IGG0192D.

2. Allocates space for the cylinder and/or master indexes in the independent overflow, or prime areas depending on the user's specifications (in his DCB and data definition statements).

Load Mode Open Executor IGG0192F

If cylinder or master indexes are not required in the initial load for creating an ISAM data set, then module IGG0192D passes control directly to module IGG0192F, instead of IGG0192E, IGG01925, or IGG01927. Executor IGG0192F might also receive control from IGG0192E, IGG01925, or IGG01927 after the high–level index areas have been structured. The primary functions of IGG0192F are:

1. Initializes several index location table pointers (the ISLIXLT fields in ISLCOMON) to point to high–level indexes if these indexes have been created by module IGG0192E.
2. Initializes pointers in the DCB to the high–level index entries.
3. Places the calculated amount of storage needed for cylinder and master indexes in the DCBNCRHI field. This field of the DCB is useful to the user if he later needs to bring the high–level indexes into main storage to search them.
4. Module IGG0192F also computes the number of tracks available for independent and cylinder overflow and places this calculation in the DCB, the JFCB, and the DSCB.

Note: When the JFCB or DSCB are modified, they are scheduled for rewriting.

Load Mode Open Executor IGG0192G

During the initial loading of an ISAM data set, control is transferred from module IGG0192F to executor module IGG0192G.

1. Module IGG0192G sets up the buffer control table (IOBBCT) used by the Put macro processing modules.
2. Formats and initializes several fields in the DCB work area (ISLCOMON) which are used later in load mode processing. These fields include:
   - ISLCBF — a pointer to the buffer to be loaded next by the Put processing routine.
   - ISLBMPR — calculated by adding the logical record length to the key length and used to facilitate "stepping through" a series of records in blocked buffers.
   - ISLFBW — (equal to the number of buffers specified in the DCB minus 1) used to determine when buffers are filled and can be scheduled for writing.
   - ISLEOB — contains the end–of–block address calculated from adding the contents of the DEBBUFL field to the starting address of the buffer.

Resume Load Open Organization

If the user is adding new records to the prime area of a previously created data set (resume loading), then module IGG01921 doesn't pass control to module IGG0192D and the rest of the initial load modules; instead, control goes to the resume load modules beginning with IGG01920 (and IGG01922) or IGG01950.

The beginning open executors for resume load ensures the accuracy of the required DSCB and DCB fields. If the user is resume loading a data set containing fixed–length records, module IGG01920 is the first module entered. If variable–length records are being added to the prime area, module IGG01950 is entered first.

Load Mode Open Executor IGG0196D

From module IGG01922 or module IGG01950, module IGG0196D is given control during the opening of a DCB for resume load. The functions of IGG0196D follow:

1. Sets up the buffer control table.
2. Sets up the R, F, and P bytes for the current–normal and current–overflow track–index entries.
3. Initializes and executes channel program 31A which reads the key portion of the last overflow track–index entry of the last cylinder. CP 31A reads this last overflow track–index entry into the key save area of ISLCOMON.
4. If necessary, module IGG0196D initializes and executes channel program 31B. CP 31B is used when the last prime–data block allocated for the data set is not full. CP 31B reads this unfilled last prime–data block into the first buffer specified in the buffer control table.

Load Mode Open Executor IGG0195G

The next module, after IGG0196D, to be executed during open processing for resume loading is module IGG0195G. IGG0195G is the resume load counter — a part of the initial load module IGG0192G. Both modules calculate and initialize fields in the ISLCOMON area for buffer and record management in load mode. IGG0195G also:

1. Sets up ISLCBF, ISLEOB, ISLBMPR, and ISLFBW in the load mode DCB work area (ISLCOMON). (See module IGG0192G, and the ISLCOMON area in "Section 5: Data Areas.")
2. Sets the DCBMSWA field to the direct–access device address (MBBCCHH) of the next–to–last track in the last prime–data extent. The DCBMSWA field normally contains the address of a user–supplied work area used when records are being added to an existing data set.
3. Initializes record moving logic.
4. Initializes Area Y, the load mode processing work area containing a high–level index entry, and normal and overflow track–index entries. Area Y is shown in Figure 68. ISLVPTRS (in ISLCOMON) points to area Y.

Load Mode Open Executor IGG0196G

1. Sets the count fields in ISLCOMON as follows:
   - ISLNCNT — the count field for the current normal–track–index entry.
   - ISLOCNT — the count field for the current overflow–track–index entry.
   - ISLDCNT — the count field for the current dummy–track–index entry.
2. Sets the count field in the first buffer.
3. Checks the DCBST field to determine where the data set is loaded.
4. Reads in the last block to determine setting of appropriate IOBS field in buffer control block (BCB).

Load Mode Open Executor IGG0195D

If the user has no high–level indexes (cylinder or master indexes), then upon completion of module IGG0196G, all the open executors used for resume load only will have been executed; the flow of control will pass to the rest of the load mode open executors which are used for both initial and resume load.

However, if during the opening of a DCB for resume loading, high–level indexes are required, control is transferred from module IGG0196G to module IGG0195D.

The functions of IGG0195D, the last resume load open executor, are described below:

1. Initializes the index location table (ISLIXLT) in the load mode DCB work area (ISLCOMON). ISLIXLT contains the beginning and ending address for each level of index above the track index.
2. Corrects the bin number in the index location table if the direct–access device being used is a 2321.

Full–Track–Index–Write Phase Organization

If the full–track–index–write option has been selected by the user, two load mode open executors (used exclusively with full–track–index–write initialization) are entered. These modules are IGG0195T and IGG0195U. Both modules are executed during a resume load when the full–track–index–write option has been selected. For an initial load, module IGG0195U receives control from IGG0195T but is not executed.

Modules IGG0195T and IGG0195U are both described below.

Load Mode Open Executor IGG0195T

1. Calculates the size of the track–index save area (TISA). When the full–track–index–write feature is selected, the TISA is used by the full–track–index–write–put routine module (either IGG019I1 or IGG0192, see Figure 11) to accumulate track–index entries and write them as a group. This is done once for each track of track index. (The full–track–index–write is described in "Load Mode Processing Phase Operations.")
2. Calculates the size of the appropriate version of channel program 20.
3. Obtains main–storage space for both the TISA and CP 20 and initializes both. If main–storage space is not available, the full–track–index–write feature will not be employed.

Load Mode Open Executor IGG0195U

If the data set is being opened for resume loading, IGG1095U initializes the track–index save area and CP 20 to resume writing track–index entries. Otherwise, IGG0195U transfers control to the final load mode open executors.

The Final Executors in Load Mode Open Phase Organization

From the resume or initial load open modules, and from the full–track–index–write modules (if used), control is passed to the final load mode open modules which are used for all forms of load mode open processing.

Load Mode Open Executor IGG0192U

The first of the final open executors entered may be either module IGG0192U or IGG0192R. IGG0192U receives control if the user has specified that write–checking is used; module IGG0192R receives control if write–checking is not used.

1. Load the modules that contain the:
   - Macro–time routines — modules IGG019GB or IGG019IB for the Put routine, or module IGG019I2 for full–track–index–write routine
   - Appendage routines — module IGG019GD
   - Channel programs — module IGG019GF or IGG019IF
2. Module IGG0192U also obtains main–storage space for the channel programs needed by the processing routines.
3. Module IGG0192U builds channel program 18 from its skeleton brought in by module IGG019GF or IGG019IF.

Load Mode Executor IGG0192R

IGG0192R performs exactly those functions outlined above for module IGG0192U, except those necessary for write–checking.

Load Mode Executor IGG0192S

Module IGG0192S receives control from either IGG0192U or IGG0192R.

1. This module builds channel program 19 from its skeleton. CP 19 is used to initialize the cylinder overflow record and to preformat shared tracks when required with fixed–length records.
2. If a track is being shared, the temporary index entries on the shared track of the first cylinder are written. This is referred to as preformatting the first shared track. Channel program 19 is used to preformat shared index tracks and to write the cylinder overflow control record (COCR). The preformatting of shared tracks pertains to fixed–length records only. Area Z in ISLCOMON is used as a work area in preformatting the first shared track.

   The description of module IGG0192D also discusses the shared track feature.
3. This module loads the RPS SIO appendage module (IGG019GG).

Load Mode Processing Phase Operations

When loading or resuming the loading of an ISAM data set, the user issues a PUT macro instruction to place the record in the data set. The Put routine moves the record to the buffer. When a specified number of buffers are full, channel programs are scheduled to write the buffers into the prime–data area of the data set and to create or update any required index entries.

An appendage routine analyzes the results of each channel program execution. When necessary, the appendage routine will start a new channel program to continue or complete the request, or it will process and resolve errors resulting from the channel program execution. When necessary, the appendage routine will start a new channel program to continue or complete the request, or it will process and resolve errors resulting from the channel program execution. If the original request was successfully completed, the appendage routine returns control to the user.

Information about the data set is communicated among the processing routines and the channel programs in control blocks and work areas. These data areas are described in detail in "Section 5: Data Areas."

This part describes the processing routine logic, the flow of control through the channel programs, in addition to the relationships of the data areas to each other, the channel programs, and the processing routines.

Put Routine

Successive PUT macro instructions cause entries to the Put routine which places records into the data set and creates the necessary indexes. The records must be in data key sequence. The Put routine (shown in Figure 6) may operate in either of two modes: move or locate. In move mode, the routine actually moves a logical record from an input buffer or work area into an output buffer. In locate mode, the routine supplies the address of an output buffer to the processing program, which must then move the record to that buffer. The mode of PUT is specified in the DCBMACRF field of the DCB.

The Put routine utilizes the beginning–of–buffer and end–of–buffer subsidiary routines to accomplish buffer management. The Put routine initializes the various channel programs and requests their execution when writing data or indexes. The appendage modules gain control after channel program execution and indicate whether or not the writing was successful.

The Put routine first checks to see if the appendage routine has signaled (in DCBEXCD1) an uncorrectable write error on a previous attempt to write either data or index entries. If so, the Put routine takes the exit to the processing program's synchronous error routine, where the user may either issue a CLOSE macro instruction or terminate the task. In any event, no more records will be accepted. The results are unpredicatable if the programmer issues another PUT macro instruction.

The Put routine then performs a check on the data key. (In locate mode the key checked is that of the previous record.) If the keys are not in ascending sequence, control is given to the user's synchronous error routine. However, in this case, if the processing program is able to correct the sequence error, it may issue another PUT macro instruction for this record, and continue normal processing.

For variable–length records, the Put routine compares the length of the record with the maximum record length specified in DCBLRECL. If it is greater than the maximum record length, the Put routine sets bit 4 of DCBEXCD2 and enters the user's synchronous error routine. The user may either change the record length and reissue a PUT macro instruction for this record or he may reissue one for the next record.

The Put routine next determines if the processing mode is move or locate mode.

Move Mode Processing

Fixed–Length Records: If the current buffer is full, the routine links to the beginning–of–buffer routine to initialize a new buffer.

It then moves the user's record to the buffer. If this record completes the buffer, the routine links to the end–of–buffer routine to attempt to write the buffer. If the buffer is not full but a write–channel program is available, the routine uses the end–of–buffer routine to attempt to write any previously filled buffers which could not be written for lack of a channel program.

The routine then returns control to the user.

Variable–Length Records: If the record format is blocked and the record fits in the current buffer and/or on the current track, it is moved into the buffer and control is returned to the user. If the record format is unblocked or if the current buffer is full, control is passed to the end–of–buffer routine to schedule the current buffer for writing. The end–of–buffer routine will pass control to the beginning–of–buffer routine to initialize the next buffer. Then the record is moved into the new buffer and control is returned to the user.

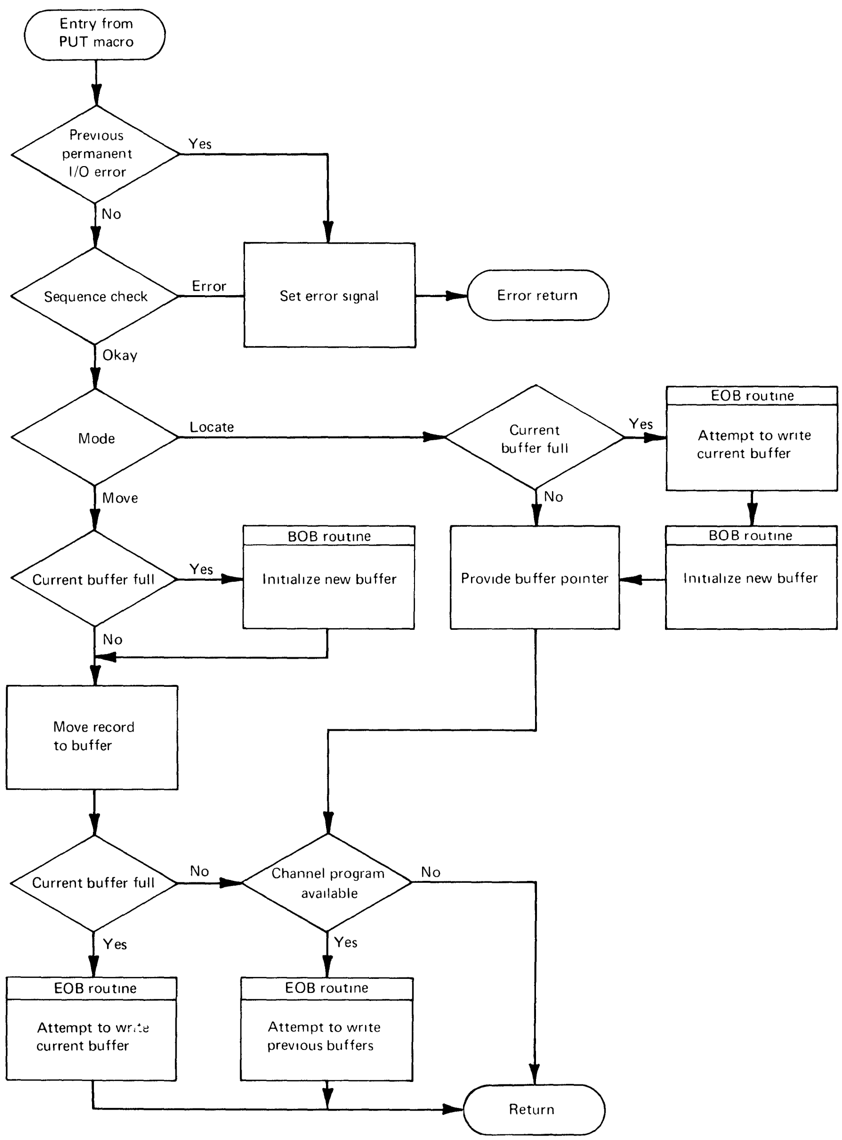


Figure 6. Load Mode Put Routine

If the record does not fit on the current track either as part of the current buffer or as another block, the current buffer is marked as the last for the current track. Control is then passed to the end–of–buffer routine to schedule the current buffer for writing. The end–of–buffer routine passes control to the beginning–of–buffer routine to initialize the next buffer. The record is moved into the new buffer and control is returned to the user.

Locate Mode Processing

Fixed–Length Records: If the current buffer is full, the Put routine links to the end–of–buffer routine to attempt to write the buffer just filled and then immediately links to the beginning–of–buffer routine to initialize a new buffer. If the current buffer is not full but channel program 18 is now available, the routine links to the end–of–buffer routine to attempt to write any buffers that could not be written previously because the channel program was in use.

The locate Put routine then provides the processing program with the address of an available buffer and returns control to the processing program.

Variable–Length Records: The Put routine computes the number of bytes remaining in the current buffer, using the buffer size and subtracting the sum of the logical record lengths of the records that have already been placed in the buffer by the user. Then the routine determines if another record of maximum LRECL can be placed into the address of the available position in the buffer. Otherwise, if the number of bytes remaining in the buffer is less than LRECL or if record format is unblocked, control is passed to the EOB and BOB routines, as described in the discussion of move mode. If it is determined the LRECL bytes added either to the current buffer or as another block exceeds the remaining capacity of the current track, the current buffer is marked as the last for the track. Control is then passed to the EOB and BOB routines.

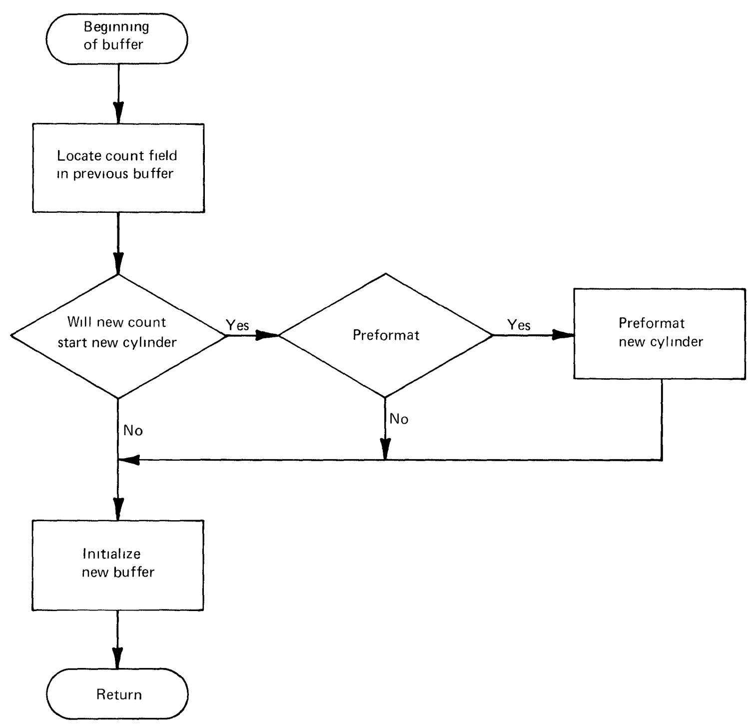


Figure 7. Load Mode BOB Routine

Beginning–of–Buffer Routine

The beginning–of–buffer routine (shown in Figure 7) initializes a new buffer and determines the device location into which the buffer will eventually be written. If the records are fixed–length and the location for this buffer proves to be the first location available for data records on a new cylinder, CP 19 may be called to preformat the track index of the cylinder if it is to contain a shared track and/or a cylinder overflow control record. In the preformatted records, only the count field is significant.

If writing this buffer causes the data set to exceed the prime–data space allocated to it, or if the appendage routine has indicated that an uncorrectable write error occurred during an attempt to add the previous contents of this buffer to the data set, the beginning–of–buffer routine takes the exit to the processing program's synchronous error routine.

The user may either issue a CLOSE macro instruction or terminate the task. In any event, no additional records will be accepted when either of these errors occurs. The end–of–buffer routine is entered when the Put routine has determined that the current buffer is full. The EOB routine initiates writing of the current buffer and any previously filled buffers not yet written under these conditions: when the current buffer is marked as the last one for the current tracks, or when the number of buffers ready for writing is equal to the value of ISLFBW.

End–of–Buffer Routine

The number of buffers that must be filled in order for a write to be scheduled (so that the number of writes per track is kept minimal) is maintained in the field ISLFBW. Its content depends on the number of buffers in the pool; however, it does not exceed the number of buffers necessary to fill an empty track if one is to be started or to fill a partially written track if one has already been started.

If a channel program is available and if the number of full buffers is equal to the content of ISLFBW, the end–of–buffer routine (shown in Figure 8) schedules a write channel program for that number of buffers and then recomputes the number. If a track or cylinder is to be completed, it also schedules channel programs to write index entries.

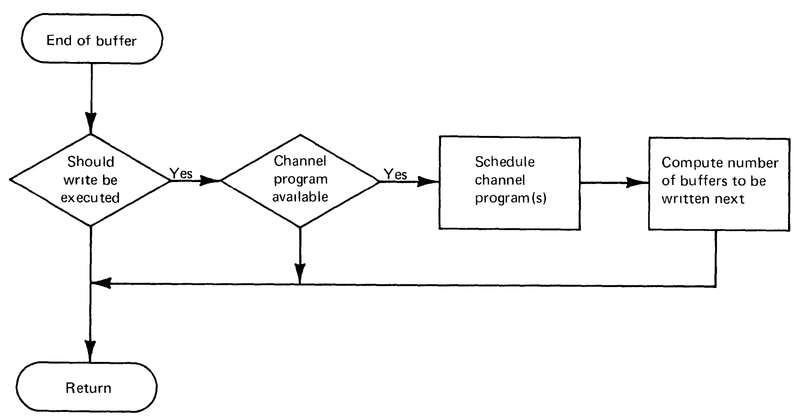


Figure 8. Load Mode EOB Routine

Full Track–Index–Write

The full track–index–write is an option for load mode that may be selected by specifying DCBOPTCD=U.

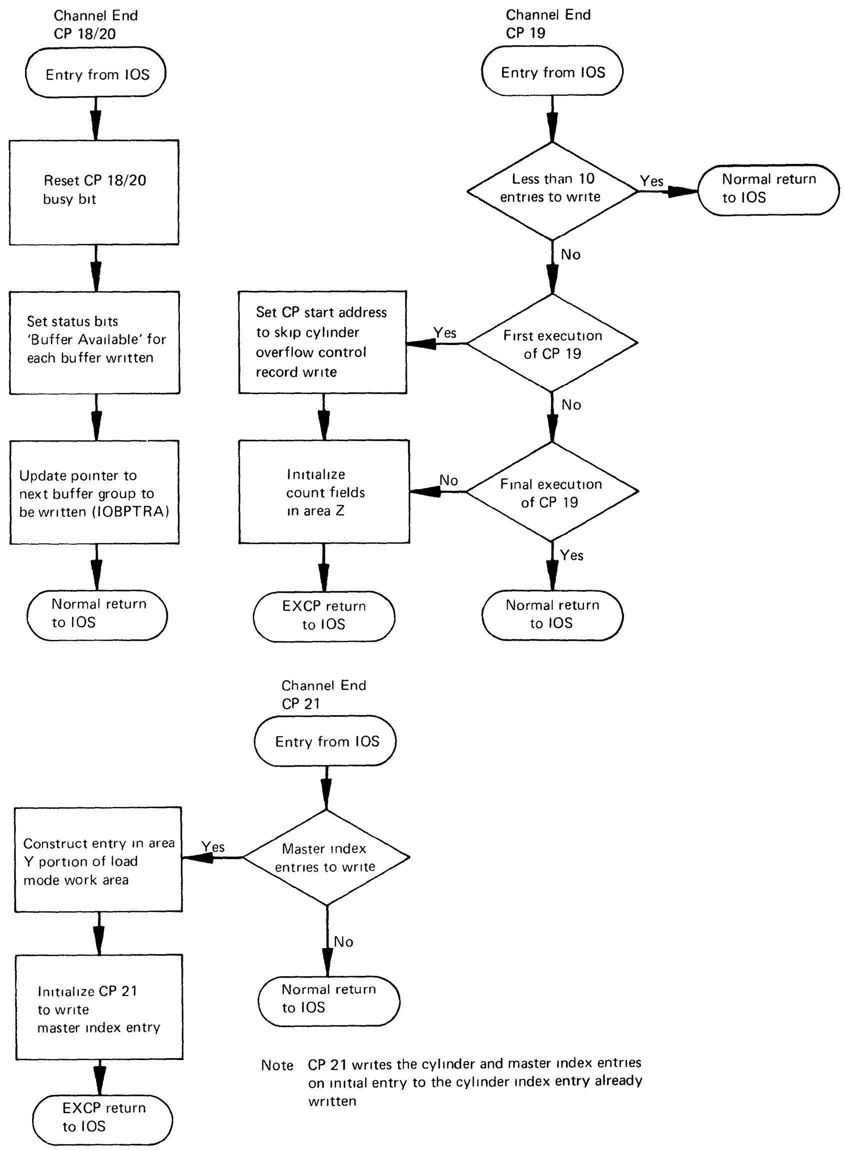


Figure 9. Load Mode Channel–end Appendage Routine

When the full–track–index–write option is specified, ISAM accumulates track–index entries in a track–index save area (TISA) obtained during open processing and writes these entries as a group, once for each track of track index.

The TISA obtained during open processing is preceded by a 20–byte control field which controls placement of entries. If an area of sufficient size is not available for the TISA, ISAM defaults to the usual mode of processing. (Normal and overflow entries written at the end of each prime–data track.)

The TISA is written when it is full, when end–of–cylinder is detected, or at processing time.

Appendages

There are both channel–end and abnormal–end appendages (shown in Figures 9 and 10) for the channel programs of load mode.

Channel–End Appendage: The channel–end appendage for CP 18 and CP 20 indicates successful completion of the channel program to the Put routines. The channel–end appendage of CP 21 indicates successful writing of an index record and determines whether a higher level index entry is needed. If so, it creates that index entry and issues an EXCP so that entry will be written. The channel–end appendage of CP 19 receives control after ten index entries have been written on a shared track and checks to see if more are needed. If the track is not yet full, it continues to issue EXCP commands until the track is properly formatted.

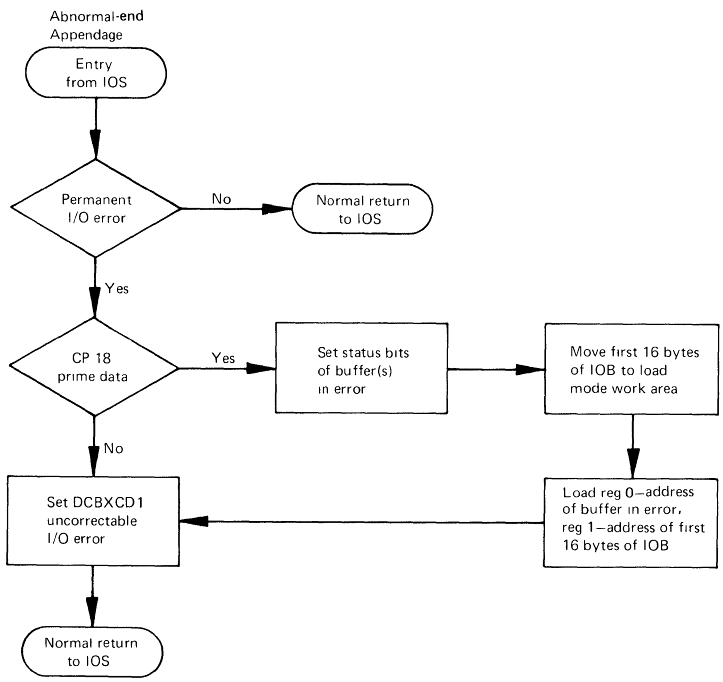


Figure 10. Load Mode Abnormal–end Appendage Routine

When write–checking has been specified, the CP 18 and CP 19 channel–end appendages reinitialize those channel programs to reread the data or index entry written before indicating successful completion. Appendages do not modify the channel programs when CP 20 and CP 21 are used with write–checking, because those channel programs are designed to read back without modifications.

Abnormal–End Appendage: The abnormal–end appendage for CP 18, upon finding a permanent error, identifies the buffer in error, saves the contents of the appropriate input/output block (IOB), and indicates the error to the Put routine. The abnormal–end appendages for CP 19, CP 20, and CP 21 also indicate permanent errors to the Put routine.

When write–checking has been specified, the CP 18 and CP 19 abnormal–end appendages have an additional function. If an error (for example, data check) is detected during read–back, the appendage reinitializes CP 18 or CP 19 for writing and issues the EXCP command.

Load Mode Processing Phase Organization

The processing routines of load mode include one module that contains the Put routine and its subsidiary routines: the beginning–of–buffer (BOB) routine and the end–of–buffer (EOB) routine. In addition, there is one module of appendages and one module of channel programs. Each of these modules exists in several versions; the version selected and executed depends on the options specified by the user. Load mode open executors, IGG0192U and IGG0192R, load the proper version according to the user's program options. Figure 11 shows the load mode processing modules.

<table>
<tr><th>Module Name</th><th colspan="2">Additional Considerations</th><th>Function</th></tr>
<tr><td>IGG019GA</td><td rowspan="2">Fixed-length Records</td><td>No write-check</td><td rowspan="4">Put processing contains Put routine, EOB routine, and BOB routine.</td></tr>
<tr><td>IGG019GB</td><td>Write-check</td></tr>
<tr><td>IGG019IA</td><td rowspan="2">Variable-length Records</td><td>No write-check</td></tr>
<tr><td>IGG019IB</td><td>Write-check</td></tr>
<tr><td>IGG019I1</td><td rowspan="2">Full track index write (Fixed-length records only)</td><td>No write-check</td><td rowspan="2"></td></tr>
<tr><td>IGG019I2</td><td>Write-check</td></tr>
<tr><td>IGG019GC</td><td colspan="2">No write-check</td><td rowspan="2">Put appendage routines—channel-end and abnormal-end.</td></tr>
<tr><td>IGG019GD</td><td colspan="2">Write-check</td></tr>
<tr><td>IGG019GE</td><td rowspan="2">Fixed-length Records</td><td>No write-check</td><td rowspan="4">Channel program skeletons—contains CP 18, CP 19, CP 20 and CP 21.</td></tr>
<tr><td>IGG019GF</td><td>Write-check</td></tr>
<tr><td>IGG019IE</td><td rowspan="2">Variable-length Records</td><td>No write-check</td></tr>
<tr><td>IGG019IF</td><td>Write-check</td></tr>
<tr><td>IGG019GG</td><td colspan="2"></td><td>RPS SIO appendage</td></tr>
</table>

Figure 11. Load Mode Processing Modules

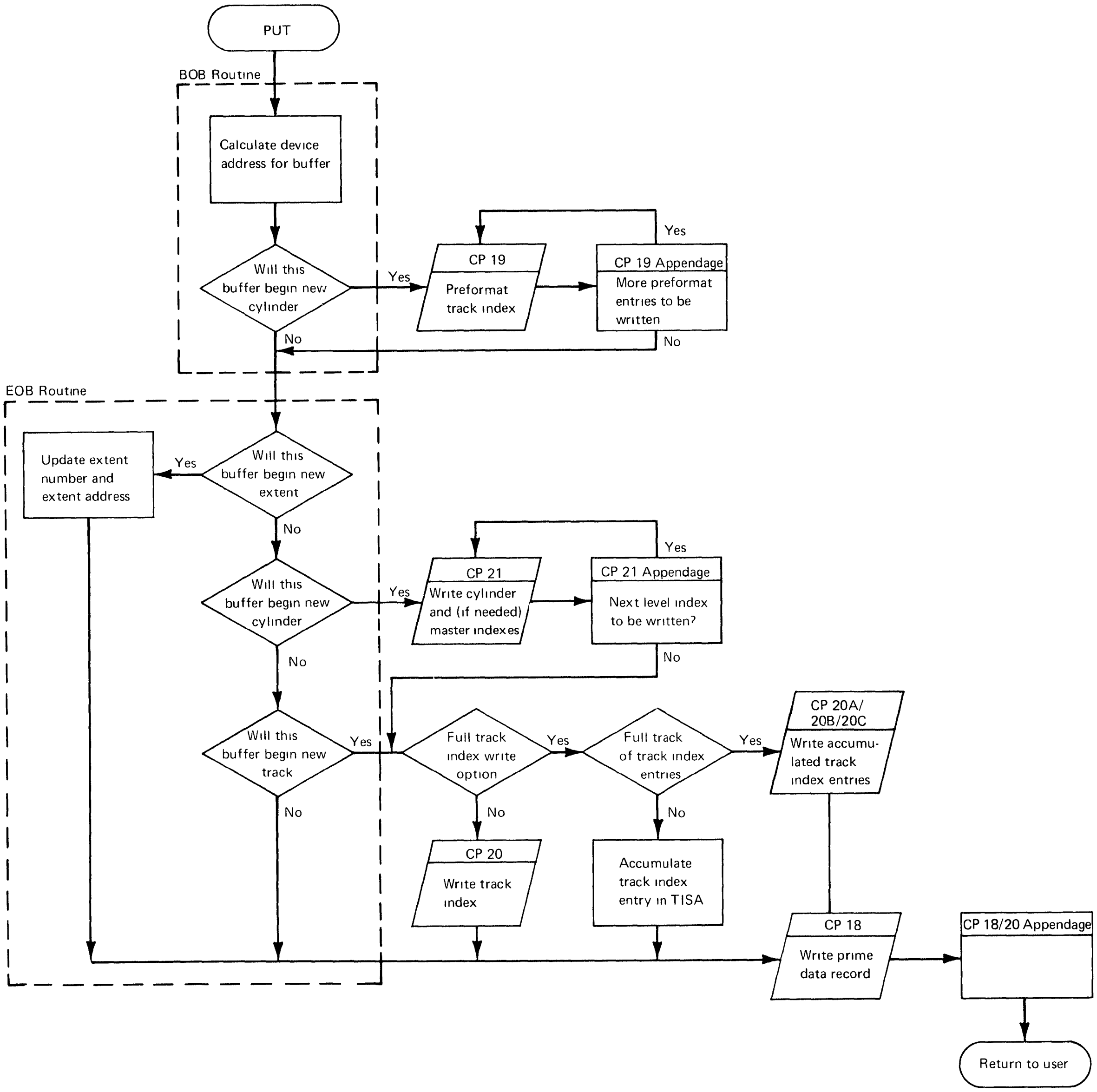


Figure 12. QISAM—Load Mode Channel Program Flow (Fixed–Length Records)

Channel Programs

The channel programs (except CP 31 and CP 91) exist in write–checking and no–write–checking versions. CP 19 and CP 20 also exist in different versions for fixed–length records and variable–length records. Figure 11 shows which channel program skeleton modules are loaded for each combination of user options. Flow of control through the channel programs is shown in Figure 12 for fixed–length records and in Figure 13 for variable–length records.

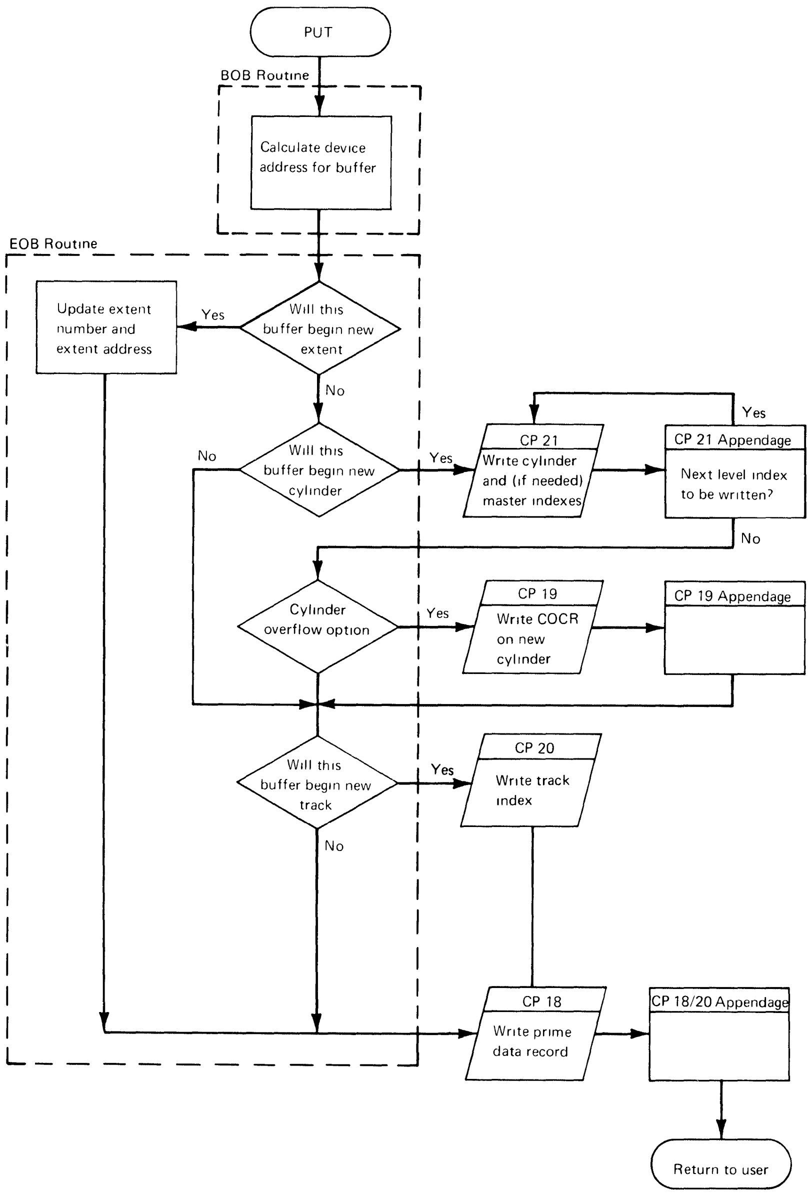


Figure 13. QISAM—Load Mode Channel Program Flow (Variable–Length Records)

CP 18 Used to write prime–data records.

CP 19 Fixed–length Records: used to initialize cylinder overflow record and shared index tracks (preformat).

Variable–length Records: used to initialize cylinder overflow control record.

CP 20 Used to write track–index entries.

CP 20A	Used to write a full track of track–index entries on a nonshared track of track–index entries.
CP 20B	Used to write a shared track of track–index entries.
CP 20C	Used to perform write–checking for CP 20A and CP 20B.
CP 21	Used to write cylinder and master–index entries.
CP 31A	Used to read the key portion of the last overflow track–index entry of the last prime–data cylinder into the key save area. (Resume loading only, located in IGG0196D.)
CP 31B	Used when the last prime–data block is not full enough to read it into the first buffer specified in the buffer control table. (Resume loading only, located in IGG0196D.)
CP 91	Used to fill unused index tracks with inactive and dummy entries. (CP 91 is located in IGG0202K.)

Control Blocks and Work Areas

Information about the data set and processing requests is carried in various control blocks and work areas. The relationship of these areas to each other and to the data set and processing programs is shown in Figure 14.

Load Mode Close Phase Operations

The first load mode close executor is entered from the Close routine. When all previously scheduled writes are finished, the load mode close executors complete the data set activity for load mode. The load mode close phase:

- Pads the last buffer
- Completes the writing of buffers
- Completes the writing of index entries
- Writes end–of–data mark
- Pads track indexes on unused cylinders
- Pads high–level indexes

Load Mode Close Phase Organization

The close phase of QISAM load mode comprises six executor modules that perform operations required to complete data set activity when a previously scheduled write operation is completed.

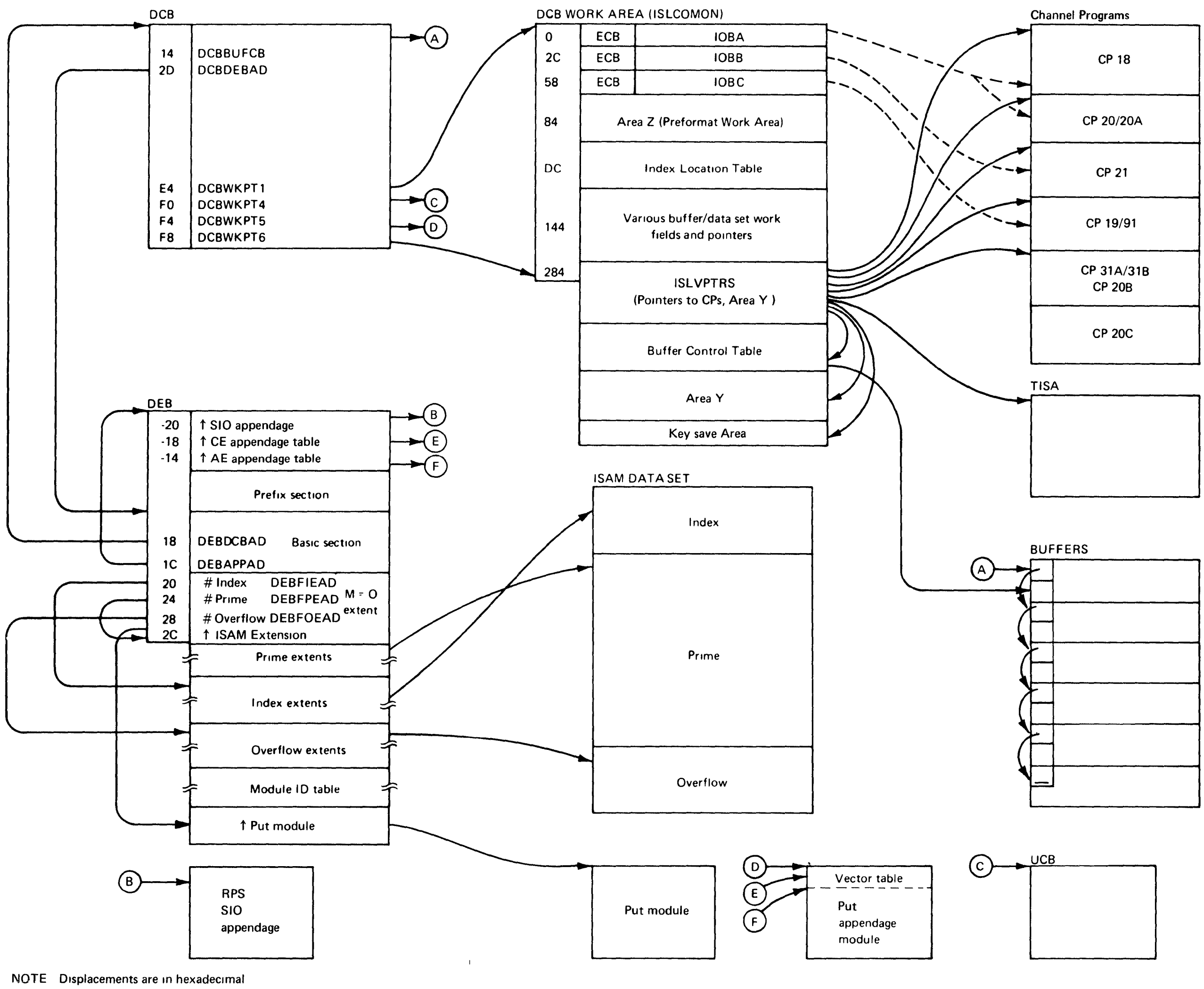


Figure 14. Load Mode Control Blocks and Work Areas

Load Mode Close Executor IGG0202I

After receiving control from the Close routine for a fixed–length record data set, IGG0202I does the following:

1. Pads (fills with dummy records) the last buffer, if necessary
2. Writes all filled but unwritten buffers
3. Completes the index entries

Load Mode Close Executor IGG02028

This module receives control from the Close routine for variable–length record data sets only. It then:

1. Pads the last buffer when necessary
2. Writes all buffers that are filled but not yet written into the data set
3. Completes the index entries so these reflect the complete data set

Load Mode Close Executor IGG0202J

1. Writes the end–of–data mark after the last data record
2. Writes the end–of–file mark in independent overflow

Load Mode Close Executor IGG0202K

1. Performs calculations for modules IGG0202L and IGG0202M in padding unused index space
2. Initializes channel program CP 91 which is used to fill unused index tracks with inactive dummy entries

Load Mode Close Executor IGG0202L

1. Writes the final dummy end–index entry.
2. Pads, with inactive entries, the unused track–index space of the cylinder containing the last prime–data record. Module IGG0202L uses ISLNIRT to signal the end–of–track index padding.

Load Mode Close Executor IGG0202M

1. Determines if higher level indexes exist and, if so, writes the final dummy entries for them.
2. Pads any unused index space with inactive entries. (See "Appendix A: ISAM Data Set Organization" for information on dummy entries and padding.)

The flow of control through the close executors is shown in Figure 15. After the mode–oriented close executors have completed their functions, the ISAM common close executor (IGG0202D) receives control. After completing the closing functions common to all ISAM, it returns control to the input/output support close routines.

Queued Indexed Sequential Access Method, Scan Mode

The scan mode of QISAM retrieves and updates the records of an indexed sequential data set in a manner similar to that of the queued sequential access method.

There are three phases of scan mode routines:

- The open phase
- The processing phase
- The close phase

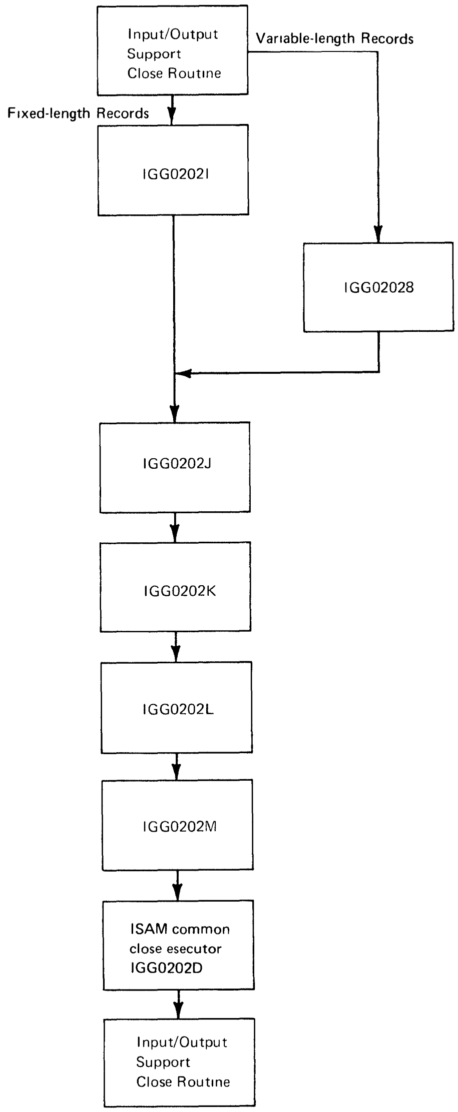


Figure 15. The Flow of Control through QISAM Load Mode Close Executors

Scan Mode Open Phase Operations

The ISAM common open executors are executed when an indexed sequential data set is opened and is to be processed by scan mode. The last ISAM common open executor passes control to the scan mode open executors. The scan mode open executors:

1. Move format–2 DSCB items to the DCB
2. Construct the DCB work area
3. Load the scan mode modules
4. Initialize channel programs and free queues.

Scan Mode Open Phase Organization

The scan mode open executor modules are IGG01920, IGG01922, IGG01950, IGG01928, IGG01929, and IGG01924.

The common open executor IGG0192C transfers control to the beginning open executors which are the validation modules, IGG01920, IGG01922, and IGG01950. The validation modules ensure that the DSCB and DCB fields needed are still accurate. If the data set contains fixed–length records, module IGG01920 will be the first module entered. For variable–length records, module IGG01950 is entered first. IGG01920, IGG01922, and IGG01950 are described in the common processing module description part of this manual.

Upon completion, the validation modules pass control to the first executor used exclusively in opening for scan mode, module IGG01928.

Scan Mode Open Executor IGG01928

1. Obtains main storage space for and structures the QISAM scan mode DCB work area (see "Section 5: Data Areas").
2. Loads scan mode processing modules processing routines.
3. Loads module IGG019HL which contains the channel program skeletons.
4. Moves the required channel program skeletons into the scan mode work area (see Figure 26). This includes moving one copy of the read/write channel program, CP 22, into the work area for each buffer.
5. Deletes the channel program skeleton module, IGG019HL, from main storage.
6. Tests the bits at DEBRPSID for an RPS device. If any of the bits are on, the scan mode RPS SIO appendage, IGG019HA, is loaded by executor IGG01924. A GETMAIN macro instruction for a 16–byte larger work area is issued to allow for the channel program prefix required RPS devices.

Scan Mode Open Executor IGG01929

1. Initializes the channel programs loaded by module IGG01928 in the DCB work area. If necessary, it initializes these channel programs to their non–RPS state.
2. Chains the copies of CP 22 together. Assigns a buffer to each copy of CP 22.

Scan Mode Open Executor IGG01924

1. Moves the format–2 DSCB fields needed into the DCB. (See modules IGG01950 and IGG01920 in Section 2.)
2. Loads the RPS SIO appendage if required. (See module IGG01928 above.)
3. Completes the initialization of the scan mode work area.
4. Obtains the interruption request block (IRB) that is used by the supervisor to maintain information concerning an asynchronous routine located in the Get appendage module (IGG019HG). Among the information in the IRB is the entry point address (RBEP—see the IRB as shown in Figure 26) of the asynchronous routine within module IGG019HG. (See the discussions of the scan mode Get routine and the appendages for further information on this asynchronous routine.)

5. Calculates W1ICNOT, which is equal to the integer that contains the number of buffers (DCBBUFNO) divided by (W1ICNOT=BUFNO/2).

   W1ICNOT is located in the scan mode DCB work area, and is used in scheduling input/output requests. The read/write channel program (CP 22) is only scheduled if the W1ICNOT field is set.

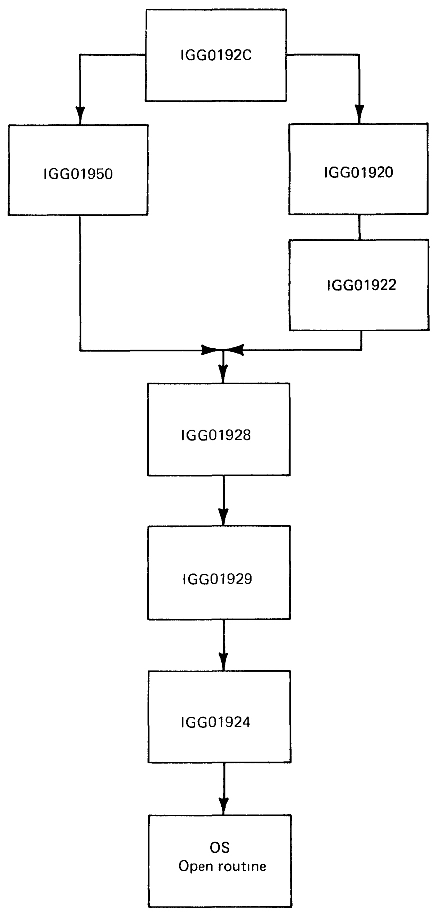


Figure 16. Flow of Control through Scan Mode Open Executors

Scan Mode Processing Phase Operations

QISAM scan mode is designed to read records from and/or write records back to an ISAM data set, selectively. Scan mode may be used to retrieve and update indexed sequential data records sequentially or randomly. The basic features of scan mode that make it able to retrieve and update records from any point in the data set are:

- A buffer controlling technique that allocates a copy of the read/write channel program (CP 22) to each buffer.
- Several logical buffer queues to which each copy of CP 22 and the buffer that the CP 22 points to may be moved. Figure 17 illustrates the chaining of channel program 22 and the buffers on these queues.

- Use of the W1ICNOT field in the scan mode DCB work area. W1ICNOT is equal to the number of buffers being used (DCBBUFNO/2) or the number of records on a prime track, whichever is less. W1ICNOT is especially important in the scheduling routine operations. (Refer to the scheduling routine description.)

The five macro instructions that cause scan mode processing routines to retrieve and update indexed sequential data records are SETL, GET, PUTX, ESETL, and RELSE. These macro instructions are described fully in *OS Data Management Macro Instructions,* GC26–3794.

The SETL routine sets the starting point of retrieval. The Get routine makes records available to the processing program. The PUTX routine restores the records to the data set. The ESETL routine terminates scanning of the data set. The RELSE routine causes the remaining records of the current buffer to be bypassed.

SETL initializes channel programs to search the indexes for the start–of–retrieval point and to read in the first buffer or buffers. GET initializes channel programs to read successive buffers, and PUTX causes the same channel programs to be reset and rescheduled to write the updated buffers back into the data set.

The channel programs for scan mode are described in detail in "Appendix B: ISAM Channel Programs." Appendage routines analyze the results of each channel program and initiate further processing operations depending on the status of the channel program's successful or unsuccessful execution.

Information about the data set is communicated among the processing routines and the channel programs in control blocks, work areas, and queues. This section shows the relationship of these areas to each other. They are described in detail in "Section 5: Data Areas."

This section describes the processing routine logic.

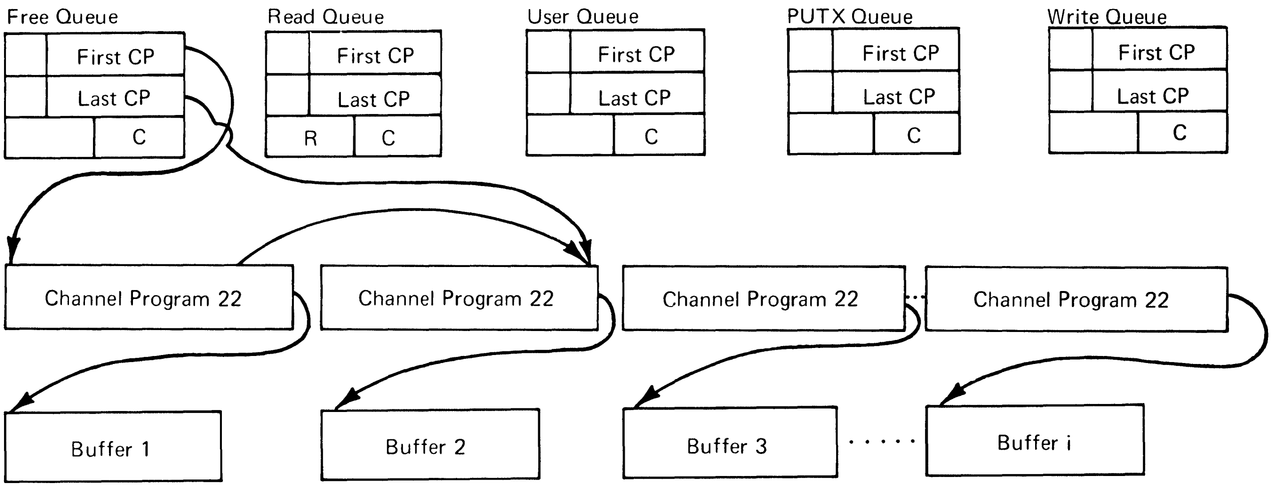
Buffer Control Techniques

Buffers are attached, by a copy of CP 22, to any one of the five buffer queues. (See Figure 17.) These queues are used in controlling input/output operations. The buffers are assigned to particular queues according to the current status of each buffer.

- Free queue buffer is not in use.
- Read queue buffer is scheduled to be filled (a version of CP 22 reads a record or records into the buffer).
- User queue buffer is made available for processing program use by the GET macro instruction.
- PUTX queue buffer is flagged as ready to be written.
- Write queue buffer is scheduled to be written.

The queuing on these buffer queues is handled by the Get routine and its subsidiary routines — the scheduling routine and the end–of–buffer (EOB) routine. However, all scan mode routines handle the buffer queuing to some degree. Figure 18 illustrates the buffer movement during scan mode processing.

The buffer queue movements of SETL and ESETL are shown in the upper portions of Figure 18, and the effects of Get and PUTX in the lower portion. The routines that process the queues are indicated on the flowlines to and from queues.



Note:

C= The number of buffers in the queue.

R= A residue of unused buffers in the Read queue. The R field is used to provide more efficient scheduling of overflow records.

Figure 17. Scan Mode Channel Program/Buffer Queues

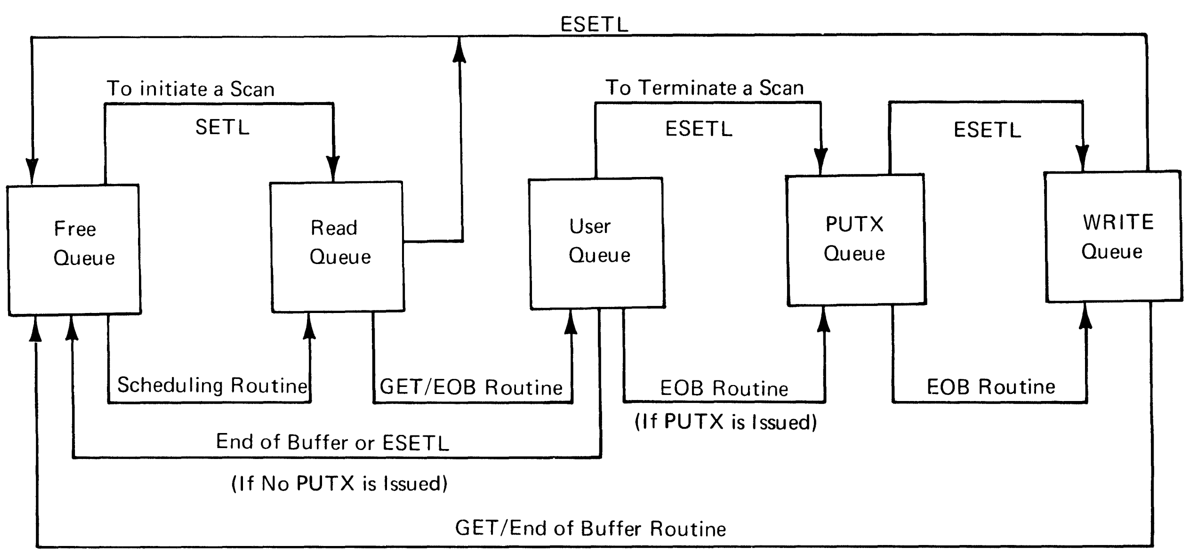


Figure 18. Buffer Queuing and Movement in Scan Mode

An Example of Buffer Movement in Scan Mode

For this example, it has been assumed that the number of buffers=3, the number of logical records per buffer=2, and each GET macro instruction issued is followed by a PUTX macro instruction.

Macro Instructions	Buffer Movement
1. OPEN	All buffers (3 buffers in this example) are placed on the Free queue.
2. SETL	a. Locate the starting record of the file (or string of records) specified in the SETL macro instruction.
	b. Place buffer 1 on the Read queue and schedule a read of the specified records into buffer 1; wait for completion of the read.
3. GET (1st GET)	a. Move buffer 1, which has been filled, to the User queue.
	b. Move buffers 2 and 3 to the Read queue and schedule a read operation.
	c. Return the address of the first retrieved record to the user.
4. PUTX	Any PUTX will set an indicator that the current record is to be written back to the data set and returned.
5. GET (2nd GET)	a. If the outstanding reads from the previous GET are completed, move those buffers to the User queue.
	b. Return the address of the next input record to the user.
6. GET (3rd GET)	a. On the third GET macro instruction, move the processed buffer (buffer 1) to the PUTX queue. (It is assumed that a PUTX macro instruction follows each GET macro instruction in the processing program.)
	b. Move buffers 2 and 3 from the Read queue to the User queue, unless these buffers were moved to the User queue by the Get routine in step 5.
	c. Return the address of the next input record in the file to the user.
7. GET (4th GET)	Return the address of the next input record to the processing program.
8. GET (5th GET)	a. Move the processed buffer (buffer 2, in this instance) to the PUTX queue.
	b. Move two buffers from the PUTX queue to the Write queue and schedule a write operation. Since the PUTX has been executed for two buffers, a Write may now be scheduled. (See "Scheduling Routine" and "EOB Routine.")
	c. Return the address of the next input record.

9.	GET (6th GET)	a. If the scheduled write is completed (step 8), move the two buffers from the Write queue to the Read queue and schedule a read.
		b. Return the address of the next input record.
10.	GET (7th GET)	a. On the seventh GET, the processed buffer (buffer 3, in this example) is moved to the PUTX queue.
		b. When the scheduled read is completed (step 9), move two buffers to the User queue. (It may be necessary to wait for the last scheduled write, move the buffers to the Read queue, issue a Read, and wait for that Read before this step can be executed.)
		c. Return the address of the next input record.
11.	GET/PUTX	The succeeding GET and PUTX macro instructions repeat steps 7 through 10. Every time a read takes place, 2 blocks will have been filled. For a write to occur, 2 buffers must be filled.
12.	ESETL	a. Wait for any outstanding read or write to be completed.
		b. Move buffers from the Read or Write queue to the Free queue.
		c. Move any buffers from the User queue to the PUTX queue or to the Free queue.
		d. Move any buffers on the PUTX queue to the Write queue and schedule a write.
13.	CLOSE	a. Wait for any scheduled, but uncompleted writes to be completed.
		b. Return all buffers to the buffer pool.

SETL Routine

The SETL routine (shown in Figure 19) determines the start of a scan by executing a channel program (dependent on the SETL option used) to search the indexes for the first record or block to be retrieved. In scan mode, records are retrieved from the beginning of the data set unless a SETL macro instruction is used.

In addition to determining the starting point, the SETL routine initializes the buffer queues. When scanning is initiated, all buffers are on the Free queue. (See "Scan Mode Open Phase Operations.") However, when subsequent scans are to be initiated, it is possible that buffers remain on the Write queue from the previous scan. When this is the case, the SETL routine moves these buffers to the Free queue after awaiting the completion of any writes in progress. The SETL routine then moves a buffer from the Free queue to the Read queue, initiates a read operation, and upon completion of the read operation, returns control to the processing program.

If the SETL routine detects any error condition, it sets the corresponding bit for that error in the DCB exceptional condition (DCBEXCD1) field. (The exceptional condition codes are described in "Section 6: Diagnostic Aids.") After setting this bit, SETL passes control to the processing program's synchronous error routine (SYNAD). If no synchronous error routine is present, the task is abnormally terminated.

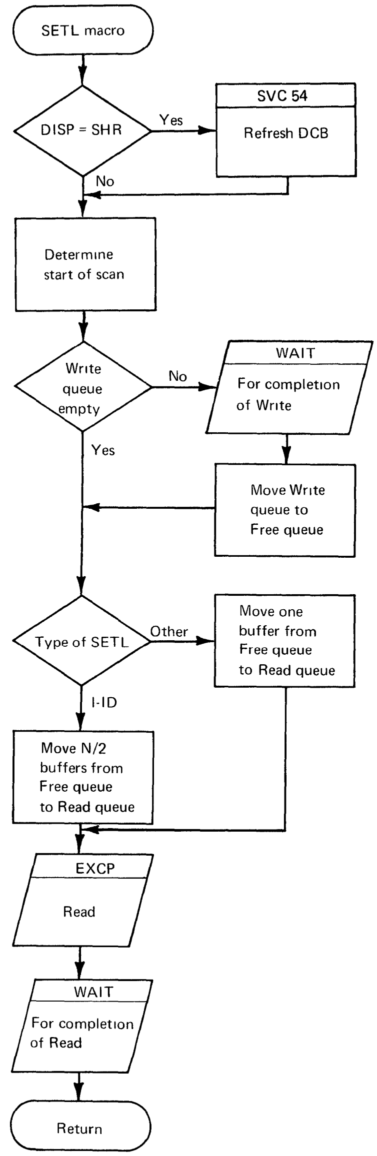


Figure 19. Scan Mode SETL Routine

When the data set is shared (DISP=SHR), the SETL routine issues an SVC 54 instruction to update the DCB field area (DCBFA). (See "The DCB Integrity Feature" under "The ISAM Common Open Executors.")

Get Routine

The Get routine (shown in Figure 20) retrieves records from the data set sequentially and gives the processing program access to a record in the current buffer on the User queue. (SETL fills the first buffer.) The Get routine has two subsidiary routines: the end–of–buffer routine and the scheduling routine.

If, on entry from the macro instruction, the user has already been given access to the last record of the User queue buffer currently being scanned, the routine links to the end–of–buffer routine to advance to a new buffer.

Then, if a write has been initiated and is complete, the Get routine moves the buffers on the Write queue to the Free queue. If the Get routine finds that an appendage routine has indicated unsuccessful completion of a previous write, the exit to the processing program's synchronous error routine is taken. Another GET macro instruction must be issued before a record becomes available for processing.

If the previous attempt to schedule a read has been unsuccessful because of a shortage of available buffers (refer to "Scheduling Routine" for criteria for determining the minimum number of buffers necessary), the scheduling routine is used to make another attempt to execute the read.

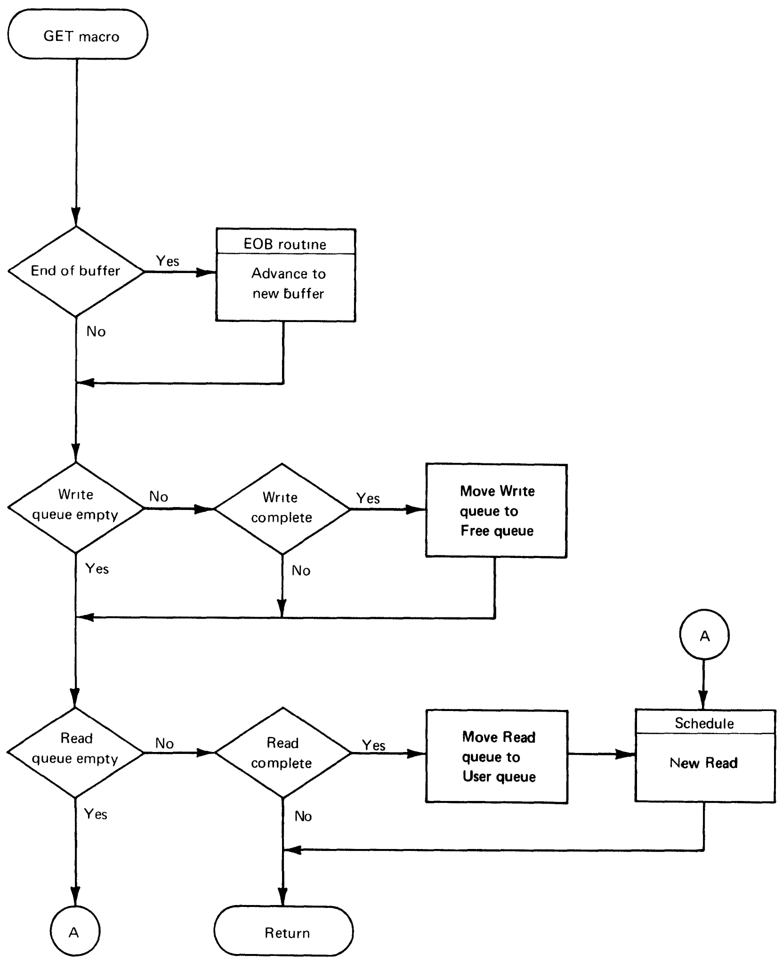


Figure 20. Scan Mode Get Routine

If a read has been initiated and is complete, the routine moves the buffers on the Read queue to the User queue and uses the scheduling routine (refer to "Scheduling Routine") to attempt to schedule a new read.

If a buffer on the User queue has been incorrectly read, each GET command issued to that buffer causes control to pass to the synchronous error routine. For blocked records, successive GET commands to the buffer give the synchronous error routine access to each record of the buffer in turn. When the buffer is exhausted and another GET macro instruction is issued, the return to the processing program is normal unless another read error occurred.

EOB Routine

The end–of–buffer (EOB) routine, which is shown in Figure 21, moves the buffer just completed from the User queue to either the PUTX queue or the Free queue. It moves the buffer to the PUTX queue if the user has issued a PUTX macro instruction for any of the records in that buffer; otherwise, it moves the buffer to the Free queue.

If there is a minimum of N/2 buffers on the PUTX queue and a previous write has been completed, the routine moves the Write queue buffers to the Free queue, the PUTX queue buffers to the Write queue, and initiates a write.

If at this point there are buffers on the User queue, the routine returns control to the calling routine. Otherwise, the routine must move buffers from the Read queue to the User queue. If the Read queue is empty, the routine waits for completion if a write is in progress, moves the Write queue to the Free queue and uses the scheduling subroutine to initiate a read and, on completion of that read moves the Read queue to the User queue. If the Read queue is not empty, the routine moves the Read queue to the User queue. It then returns control to the calling routine.

Before moving a buffer from the Write queue to the Free queue, the routine ensures that the write operation of that buffer was successfully completed. If not, the synchronous error routine is given control.

Scheduling Routine

Processing in the scheduling routine (shown in Figure 22) depends primarily on whether the next record to be read is on a prime–data or overflow track.

If an overflow record is to be read, a read may be scheduled if there are at least two buffers on the Free queue. It may also be scheduled if there is only one buffer and that buffer is on the Free queue. Before initiating the read, the routine moves the Free queue to the Read queue. It then returns control to the calling routine.

If prime data is to be read, it attempts to schedule a read of N/2 buffers. Provided N/2 buffers are available and at least N/2 blocks remain on the track, this can be done. It can also be done with fewer than N/2 blocks remaining on the track if the track is not the last one of a cylinder and no overflow chain is associated with the track. If these conditions are met, the routine moves N/2 buffers from the Free queue to the Read queue, initiates a read, and returns control to the calling routine.

If these conditions are not met, the scheduling routine initiates a read operation to complete the last track of a cylinder or a track having an overflow chain associated with it, provided that sufficient buffers are available on the Free queue. As before, it moves the buffers required to the Read queue, initiates a read, and returns control to the calling routine.

If a read cannot be initiated, the routine returns control to the calling routine.

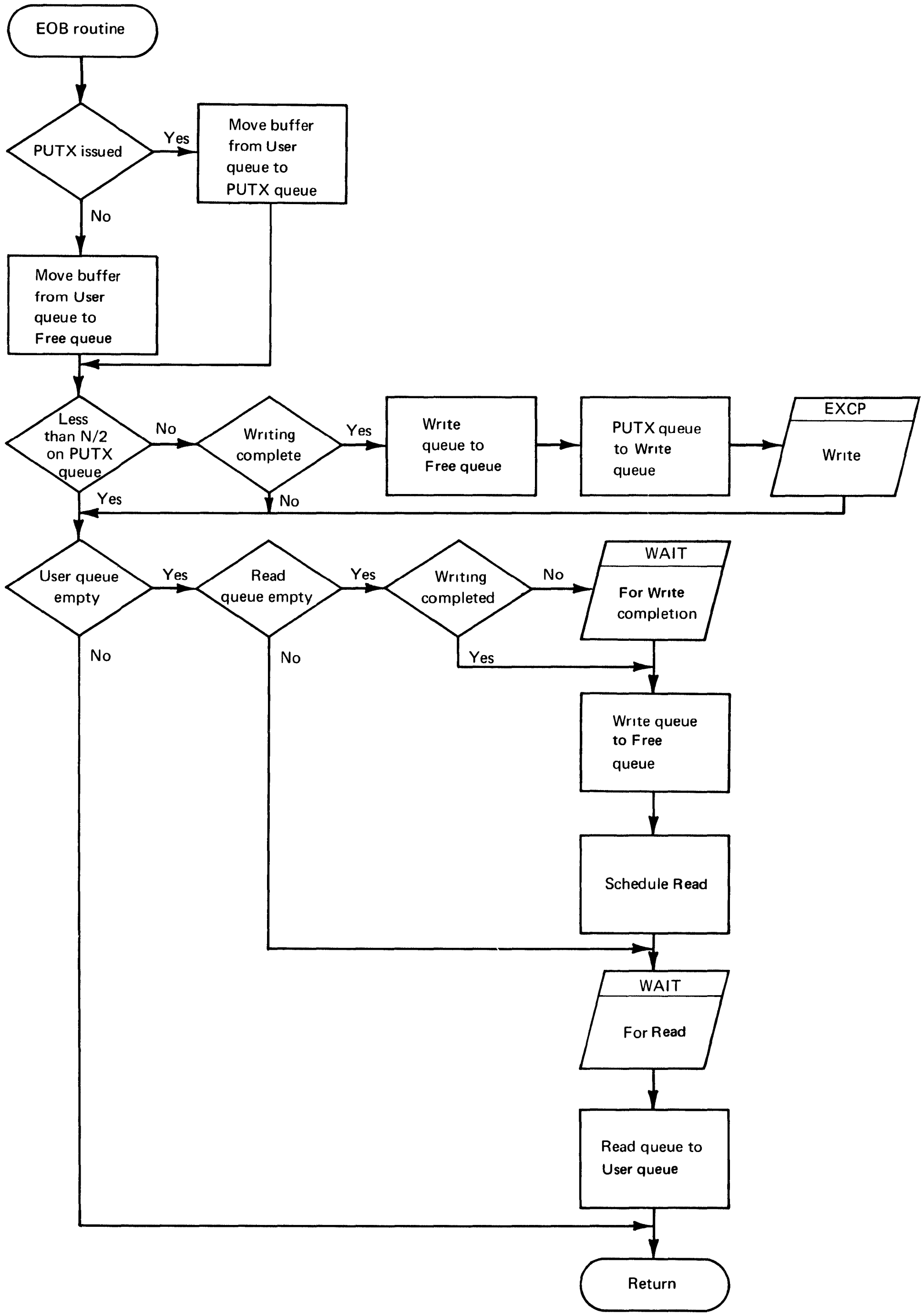


Figure 21. Scan Mode EOB Routine

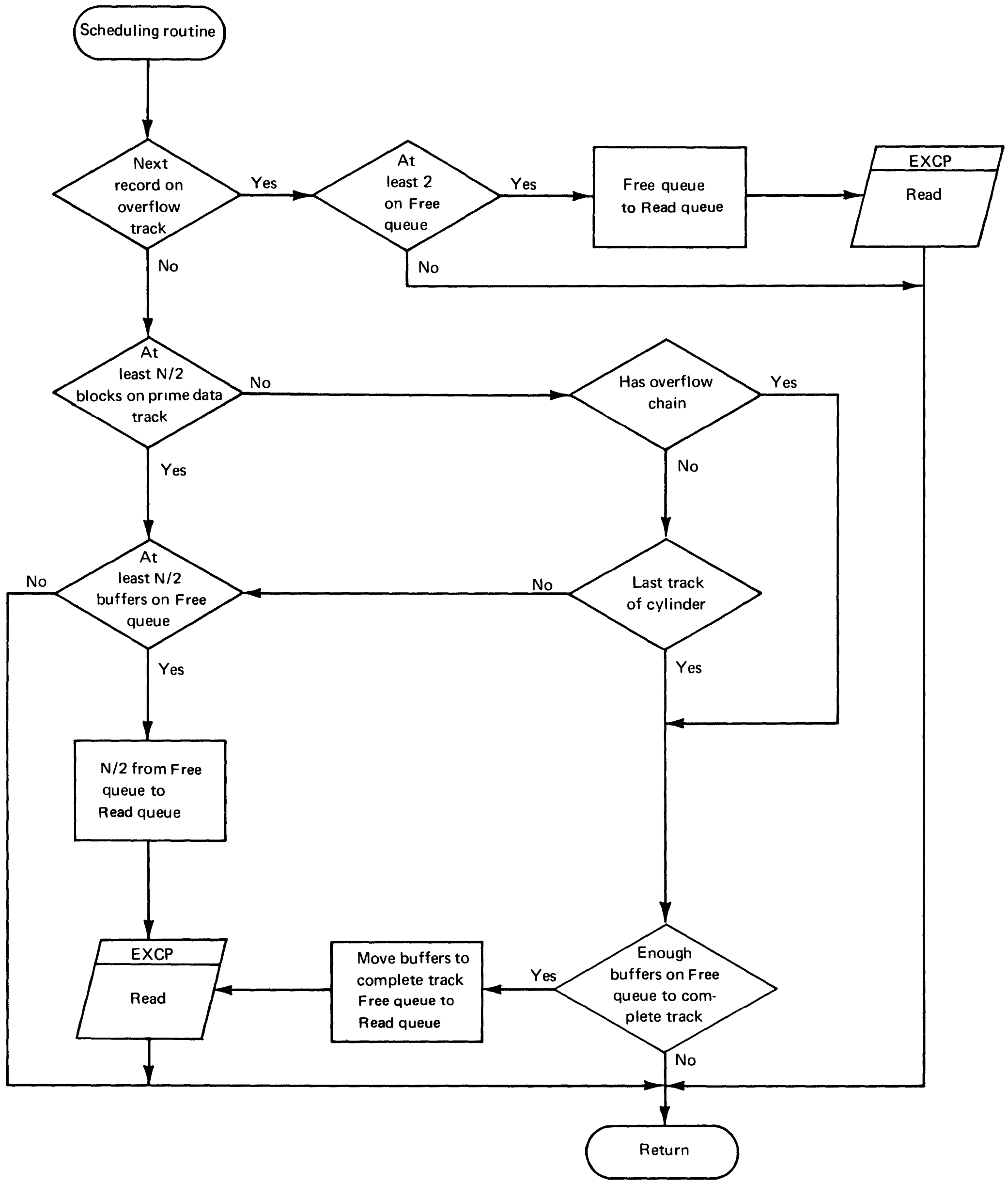


Figure 22. Scan Mode Scheduling Routine

PUTX Routine

The PUTX macro is used in updating data sets. When the PUTX macro instruction is issued in the processing program, the PUTX routine of scan mode will be used (see Processing Routines — Figure 24). The PUTX routine causes records obtained by the locate mode GET macro instructions to be written back to the data set.

The PUTX routine sets an indicator flag associated with the current buffer on the User queue. The GET macro instruction's end–of–buffer (EOB) routine uses this indicator to determine if the User queue buffer should be moved to the PUTX queue. Eventually, the buffer will be moved from the PUTX queue to the Write queue (it is moved either by the EOB routine for GET or by the ESETL routine when an ESETL is issued in the processing program). Once on the Write queue, the buffer is scheduled to be written — that is, the channel program used to read or write the buffer (a copy of CP 22 is used with each buffer) is reset and scheduled to write the udpated buffer back into the data set.

ESETL Routine

The ESETL routine (shown in Figure 23) ends scanning of the data set.

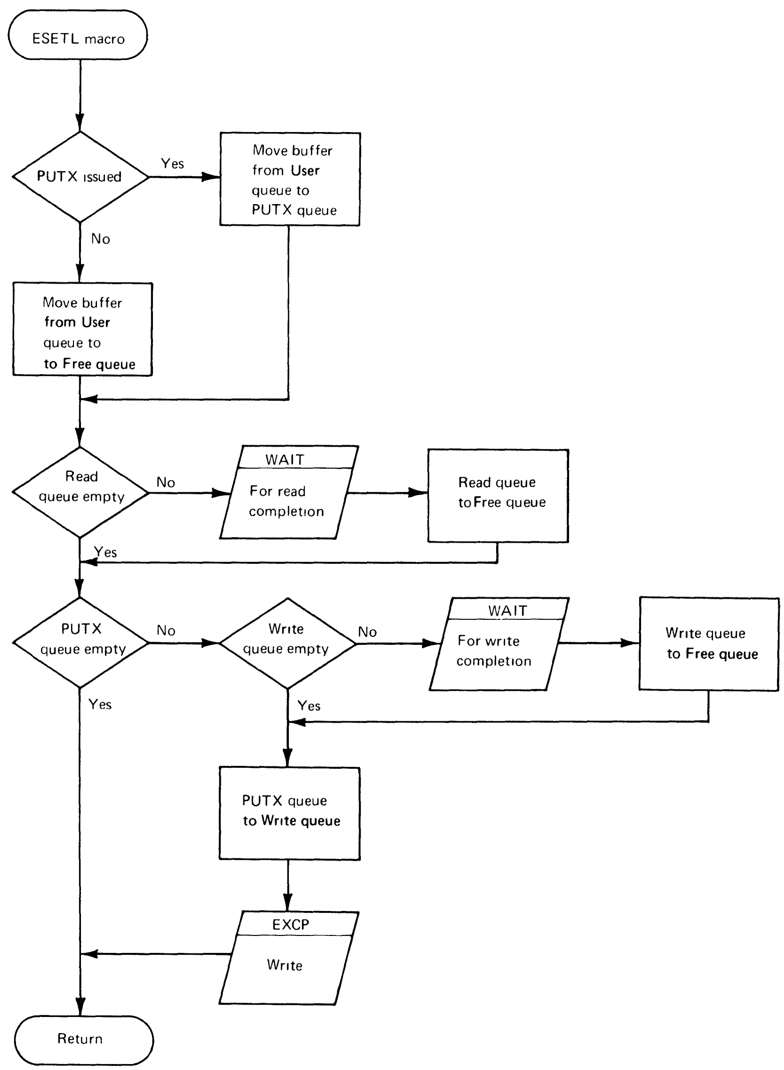


Figure 23. Scan Mode ESETL Routine

If the user has issued a PUTX macro instruction for any of the records in the current buffer on the User queue, the routine moves the buffer to the PUTX queue. If the Read queue is not empty, the routine awaits completion of pending reads and then moves the Read queue to the Free queue.

If the PUTX queue is empty, the routine returns control to the processing program. Otherwise, the routine awaits completion of pending writes and moves the Write queue to the Free queue if the write was successful. (If the write was not successful, the synchronous error routine is entered, and another ESETL macro instruction must be issued to end this scan.) It then moves the PUTX queue to the Write queue, initiates a write, and returns control to the user.

RELSE Routine

The RELSE routine links to the end–of–buffer routine causing the current buffer to be released and a new buffer to be initialized. If the current record is the first or last logical record in the buffer, the request is ignored. The RELSE routine then returns to the user.

The RELSE routine also determines if there were any write errors for those buffers on the Write queue whose writing had been completed. If so, the processing program's synchronous error routine is given control and another RELSE must be issued to release this buffer.

Appendages

There are both channel–end and abnormal–end appendages for those routines that cause input/output operations. (Refer to Figure 24.)

The channel–end appendage of the SETL I routine causes a normal return to the I/O supervisor if CP 25 was completely executed. If CP 25 was not completely executed, either the channel–end or abnormal–end appendage of the SETL I routine may be entered, depending on the setting of the CSW status bits. In the case of incomplete execution, an indicator is set so that the SETL I routine can later inform the processing program that the record was unreachable. A normal return to the I/O supervisor is issued.

The channel–end and abnormal–end appendages of the SETL K (or SETL KC) routine examine CP 23 to find out where and why the channel program terminated. Based on this examination, either CP 23 is reinitialized to continue searching for the desired key by issuing an EXCP return, or an indicator is set to inform the processing program that the key could not be found and a normal return is issued. Whether the examination is performed by the channel–end or abnormal–end appendage depends on the setting of the CSW status bits and the contents of the higher level indexes.

The channel–end appendage of the Get routine issues a normal return to the I/O supervisor if there are no more buffers on the Read queue, or the last record on a track has been read, or the buffers on the Read queue were filled with records read from a prime–data area. This channel–end appendage issues an EXCP return to the I/O supervisor, or schedules an asynchronous routine to issue an EXCP return if an overflow record was read after it modified CP 22 to continue reading the records in the overflow chain. When the last record of an overflow chain has been read, a normal return is issued. The abnormal–end appendage of the Get routine sets an indicator to mark the buffer that contains the record in error and issues an EXCP return if there are more records to be read. Otherwise, it issues a normal return.

The channel–end appendage of the PUTX routine (without write–checking) makes a normal return to the I/O supervisor if there are no more buffers on the Write queue. An EXCP return is issued if there are more buffers on the queue to be written. The abnormal–end appendage makes the same returns under the same conditions, but, in addition, it sets both a write–error indicator and an indicator to inform the processing program which buffer contains the record in error.

When a write–checking is in effect, the PUTX routine channel programs are command–chained to write the contents of a set of buffers at a time, rather than writing all the buffers on the Write queue. For prime–data records, a set of buffers is the number of buffers on the queue or the number needed to complete the current track, whichever is lower. For overflow records, a set is one buffer. The contents of a set of buffers is written and checked before the next set is written.

If return is to the channel–end appendage after the inital write of a set, CP 22 is modified to accomplish readback, and an EXCP return to the I/O supervisor is issued.

If return is made to the abnormal–end appendage after the initial write of any buffer in the set, that buffer is marked unreachable or unwritable and an EXCP return is issued to write the remaining buffers in the set; if no buffers remain in the set, CP 22 is modified to accomplish readback of the successfully written buffers, and an EXCP return is issued. No attempt will be made to rewrite the buffer in error; the processing program will be informed of the error the next time a GET macro instruction is issued for the buffer.

If channel–end return is made for both writing buffers and reading them back, an EXCP return is issued if there is another set to be written. Otherwise, a normal return is issued.

If a return to the abnormal–end appendage occurs when reading back a buffer that was successfully written, an EXCP return is issued to rewrite, and an additional EXCP return is issued to recheck the buffer in error. Up to ten rewrites and rechecks per buffer are permitted; CP 22 must be modified for each readback and rewrite. If a successful readback cannot be accomplished, or if an abnormal–end return is made on any of the attempts to rewrite the buffer, the buffer is marked as unwritable and an EXCP return is issued to start writing the next set. If there are no more sets to be written, a normal return is issued.

When an EXCP return is to be issued and the next record to be written or searched is on another device, the appendage routine cannot issue the EXCP command itself. Instead, it schedules an asynchronous routine (located in the GET appendage). When the asynchronous routine receives control, it issues the EXCP macro instruction.

Scan Mode Processing Phase Organization

Processing Routines

The modules containing the scan mode processing routines are shown in Figure 24.

Module Name	Function
IGG019HB (Fixed-length records)	Get, PUTX, RELSE, ESETL, SETL B processing routines
IGG019HN (Variable-length records)	
IGG019HD	SETL K processing routines
IGG019HF	SETL I processing routines
IGG019HG	Get channel-end and abnormal-end appendages and asynchronous routine
IGG019HH	PUTX channel-end and abnormal-end appendages, no write-check
IGG019HI	PUTX channel-end and abnormal-end appendages, write-check
IGG019HJ	SETL I channel-end and abnormal-end appendages
IGG019HK	SETL K channel-end and abnormal-end appendages
IGG019HL	channel program skeletons
IGG019HA	RPS SIO Appendage

Figure 24. QISAM Scan Mode Processing Modules

Scan Mode Channel Programs

The scan mode channel program skeletons are contained in module IGG019HL. The channel program skeletons are moved to a work area and completed during the open phase of scan mode.

In processing and updating an ISAM data set, the following scan channel programs are used:

Channel Program 22 (CP 22)	The two versions of CP 22 are used to read or write data records. *Version 22A (CP 22A)* is used to read the key and data fields of unblocked records. *Version 22B (CP 22B)* is used to read either the data field of unblocked records, or any blocked records.
Channel Program 23 (CP 23)	Used to locate the data record by SETL K or KC; searches the index and data tracks.
Channel Program 24 (CP 24)	Used to read count and data fields of the track–index entries.
Channel Program 25 (CP 25)	Used with SETL I to obtain track–index entries.
Channel Program 26 (CP 26)	Used on overflow chains as an extension of CP 23 (SETL K).

If the user has allocated enough buffers and is reading a full track at a time, as many CP 22s as are needed (one for each buffer) are chained together for reading the track; the same is true for writing a full track at one time, that is, all copies of CP 22 are chained together.

Assuming the use of a file with no overflow, CP 23 is used by SETL to locate the proper record; then CP 22 is used to read the record; CP 24 then reads the next level of track–index entries and schedules the next CP 22.

Figure 25 illustrates the operations of one scan mode channel program (CP 23). Channel program 23 is used by SETL to position to the first record of the specified file. For this example, it is assumed that no master indexes are being used.

Scan Mode Control Blocks and Work Areas

Information about the data set and processing requests is carried in various control blocks, work areas, and queues. The address relationships of these areas to each other and processing routines and channel queues are shown in Figure 26.

Scan Mode Close Phase

The QISAM scan mode close phase has only one close executor, module IGG02029, which is entered from the I/O support Close routine. Module IGG02029 uses the ESETL routine to terminate scanning and clear the buffer queues. (Refer to "ESETL Routine" and "Buffer Control Techniques.")

Even if the user has already issued an ESETL, the close executor issues another one. The close executor then awaits completion of any outstanding writes. If any of these writes are unsuccessful, the user synchronous error is entered. The user must return to the close executor to complete the release of buffers and work areas to the operating system.

If the oustanding writes or the return from the synchronous error routine to the close executor have been completed successfully, then the close executor:

1. Returns all buffers to the buffer pool.

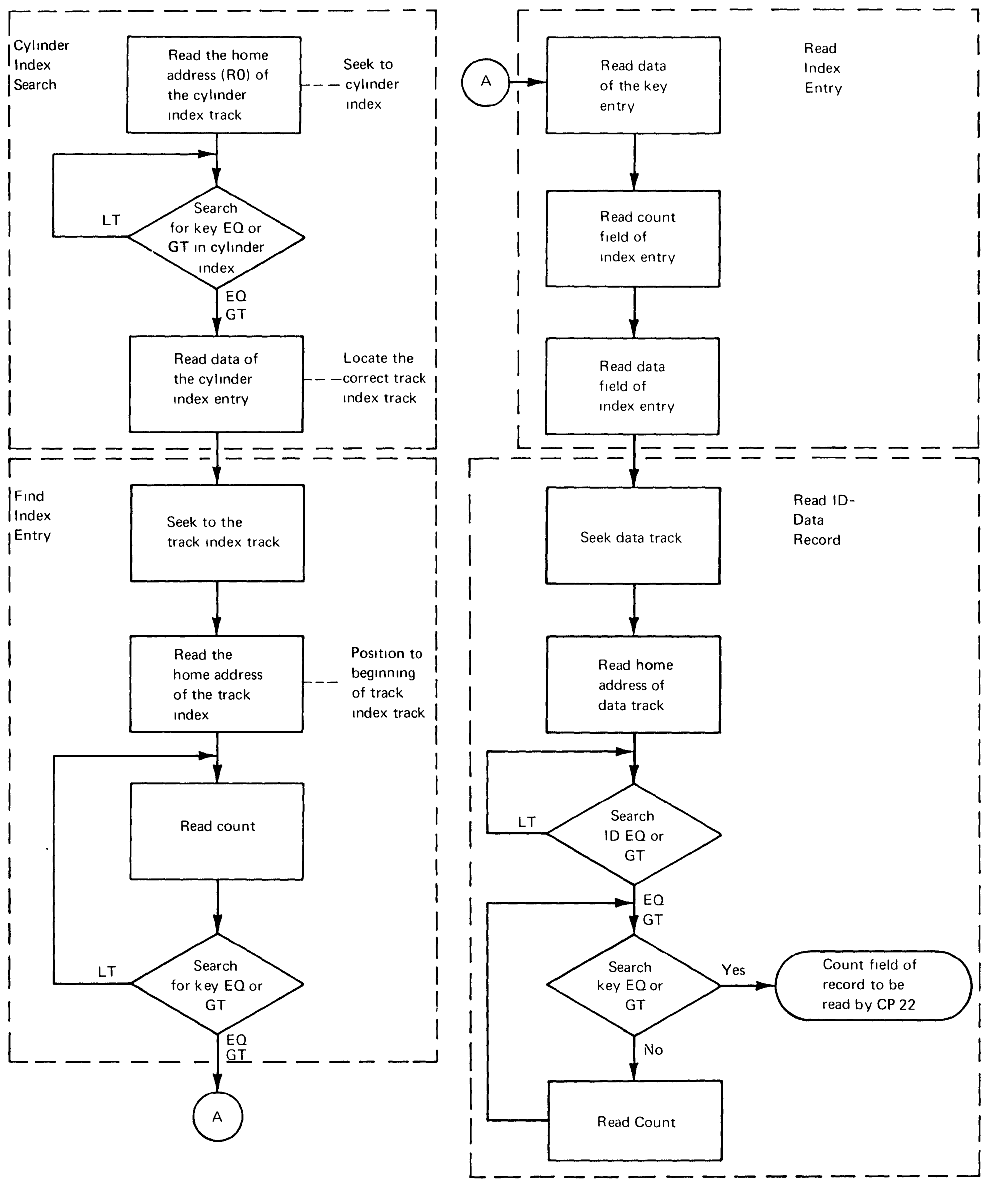


Figure 25. Scan Mode Channel Program 23

2. Releases the work area.
3. Updates the DCB tag deletion count, DCBTDC.
4. Updates the number–of–overflow–references field in the DCB, DCBRORG3.
5. Moves the DCB fields that may have been changed during processing from the DCB field area (DCBFA) to the DCB if the data set was opened for DISP=SHR. Frees the DCB field area if this is the last DCB open for the data set.

When finished, the scan close executor, module IGG02029, passes control to the ISAM common close executor.

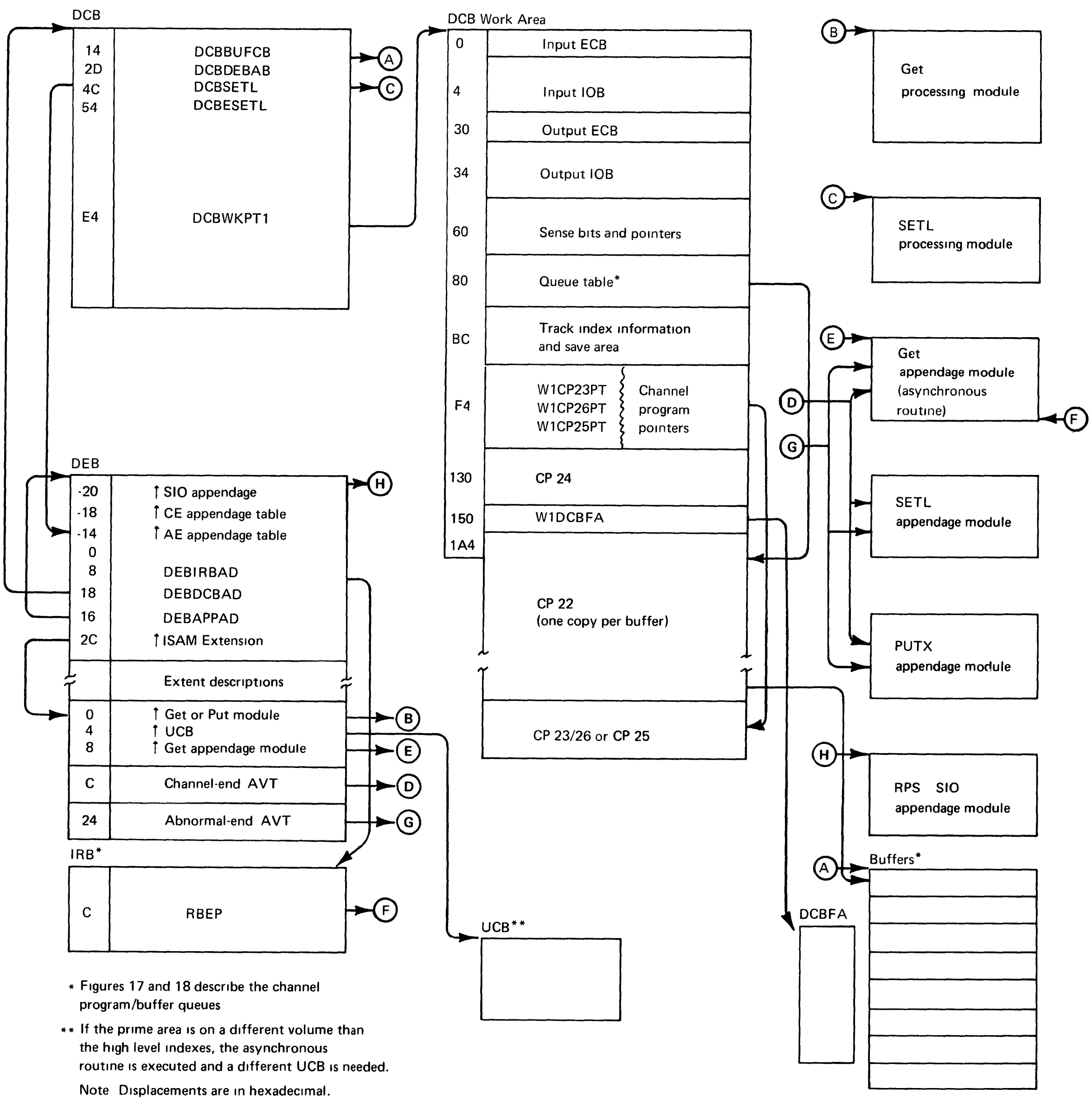


Figure 26. Scan Mode Control Blocks and Work Areas

Basic Indexed Sequential Access Method

The basic indexed sequential access method (BISAM) provides direct storage and retrieval of the records in an indexed sequential data set. The READ K macro instruction permits the retrieval of a logical record from main storage by its record key. The READ KU and WRITE K macro instructions, when used together, provide the ability to update logical fixed–length (or variable–length if the record length does not change) records in place. The WRITE K macro instruction, when used without READ KU, allows the user to replace unblocked fixed–length (or variable–length if the record length does not change) logical records. The WRITE KN macro instruction is used with the READ KU macro instruction to update variable–length records when the record length can change. The WRITE KN macro instruction allows the user to insert new logical records into the data set or to replace a variable–length logical record with one having the same key and possibly a different record length.

Since storage and retrieval of records are direct in BISAM, the BISAM routines are not able to read ahead as the QISAM scan mode Get routine can. Consequently, the user must issue a WAIT or CHECK macro instruction in order to determine whether a read operation has been completed.

As in QISAM, there are three phases of BISAM routines:

- The open phase
- The processing phase
- The close phase

BISAM Open Phase Operations

The first BISAM open executor is entered from the last common ISAM open executor. The BISAM open executors load the BISAM processing routines, selecting the processing phase modules according to the processing program options. Particular processing modules are selected depending on such options and considerations as:

- The number of levels of index to be searched on the direct–access device (NLSD)
- Whether the records are blocked or unblocked
- Whether work areas are supplied by the user or by the access method routines
- Whether or not write–checking is to be used
- Are buffers controlled by the user program or by the ISAM dynamic buffering routine (module IGG019JI)
- The user's intent to add new records to the data set with the WRITE KN macro instruction

Some of these considerations also affect the sequence in which the BISAM open executors are called. Figure 27 illustrates the flow of control through the BISAM open executors.

Those BISAM open executors that initialize channel programs include conversion to a non–RPS state as part of their processing.

BISAM Open Phase Organization

When a DCB is being opened for BISAM processing, one or two of the validation modules are selected to correlate format–2 DSCB and DCB fields. The validation modules (IGG01920, IGG01922, and IGG01950) are also used in open processing for resume load and scan mode.

If the records are fixed–length records, modules IGG01920 and IGG01922 are selected for validation and initial BISAM open processing.

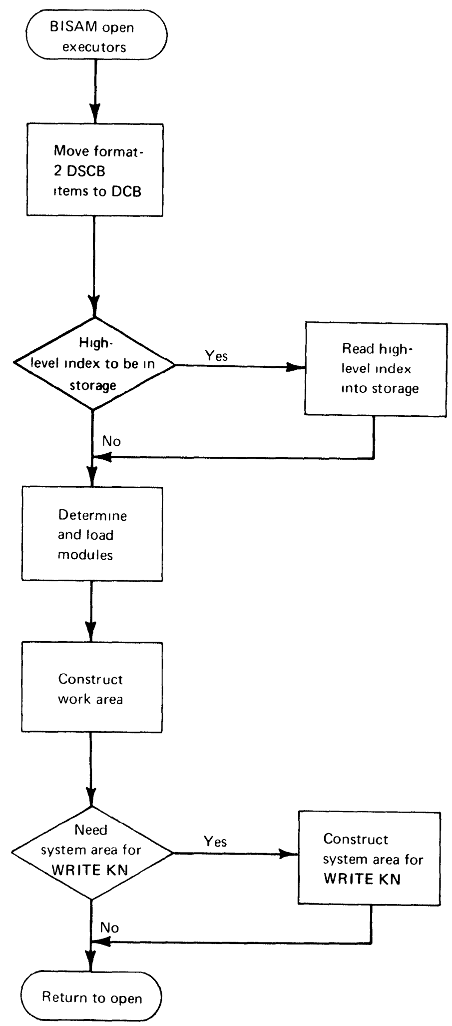


Figure 27. BISAM Open Executors

These two modules reset certain fields in the DCB and format–2 DSCB which may be incorrect if the data set was previously closed improperly.

If variable–length records are used, module IGG01950 is selected to merge end pointers from the format–2 DSCB to the DCB and adjust, if necessary, the independent overflow control information in the DCB.

IGG01950 is the VLR counterpart to modules IGG01920 and IGG01922. It is the first BISAM open module entered when variable–length records are being added.

The validation module may not be executed, although it will be entered, if the user has specified that the data set may be shared by other tasks (DISP=SHR). It will not be executed in that case if another DCB has already been opened for the data set and a DCB field area (DCBFA) set up for the purpose of maintaining the DCB fields. (See "The DCB Integrity Feature" under "The ISAM Common Open Executors" and the description of the DCBFA.)

Module IGG0192W or IGG0192H receives control from modules IGG01920 and IGG01922, or module IGG01950 during the opening of a DCB for BISAM.

BISAM Open Executor IGG0192H (Fixed–length records)

1. Moves the format–2 DSCB fields needed for BISAM into the DCB.
2. Obtains and structures the work areass and provides pointers to the work area.

BISAM Open Executor IGG0192W (Variable–length records)

1. Moves the format–2 DSCB fields needed for BISAM into the DCB.
2. Obtains and structures the work areas and provides pointers to the work areas.

BISAM Open Executor IGG0192P

1. When the high–level indexes are to be searched in main storage, module IGG0192P schedules CP 87 to read the high–level index into the user–specified work area. The work area is specified in the DCB at DCBMSHI. Channel program 87 is contained in module IGG0192P.
2. After reading the high–level index into the user work area, module IGG0192P saves the address of the last active entry in the high–level index.

BISAM Open Executor IGG0192I

1. Selects and loads the proper privileged module, according to the options specified in DCBMACRF by the user. (See Figure 35 for the privileged macro–time module.)
2. Selects, loads, and initializes CP 1 when cylinder and master indexes are to be searched on the direct–access device.
3. Selects, loads, and initializes CP 2 when the cylinder index is the highest level index to be searched on the device.

4. If an RPS device is being used, IGG0192I saves and restores the high-order byte of DEBISAD when storing the address of the privileged macro-time module. (See step 1.) This is done to preserve the RPS bits at DEBRPSID.
5. This module also initializes RPS fields in the DCB work area.
6. Initializes the error queue counter to 2(NCP) + DCBBUFNO.

BISAM Open Executor IGG0192K (READ K, READ KU, WRITE K)

1. Selects and loads CP 4, CP 5, CP 6, and CP 7; initializes these channel programs.
2. Selects and loads the nonprivileged macro-time routine, module IGG019JV, for READ K, READ KU, and WRITE K.
3. If dynamic buffering is specified, loads the dynamic buffering module, IGG019JI.
4. If RPS is used and the dynamic buffering module loaded, IGG0192K also sets bit 3 of DEBRPSID.

BISAM Open Executor IGG0192L (WRITE KN)

1. Loads the set of WRITE KN channel programs needed with the data set being processed — blocked or unblocked records, user work area or system work area, etc. (See BISAM channel programs, Figures 40–52.)
2. Loads the nonprivileged macro-time routines for WRITE KN, module IGG019JW.
3. Initializes CP 8 and CP 10B.

BISAM Open Executor IGG0192M (WRITE KN with fixed-length records)

1. Initializes CP 14 which is used to update the cylinder overflow control record (COCR) and writes overflow records. There are six different versions of this channel program, which are described in "Appendix B: ISAM Channel Programs."

BISAM Open Executor IGG0192X (WRITE KN with variable-length records)

1. Performs the same functions as IGG0192M as described above. See CP 14 in "Appendix B: ISAM Channel Programs."

BISAM Open Executor IGG0192Q (WRITE KN)

1. Initializes CP 1 or CP 2, CP 10A, CP 15, CP 16, CP 17.

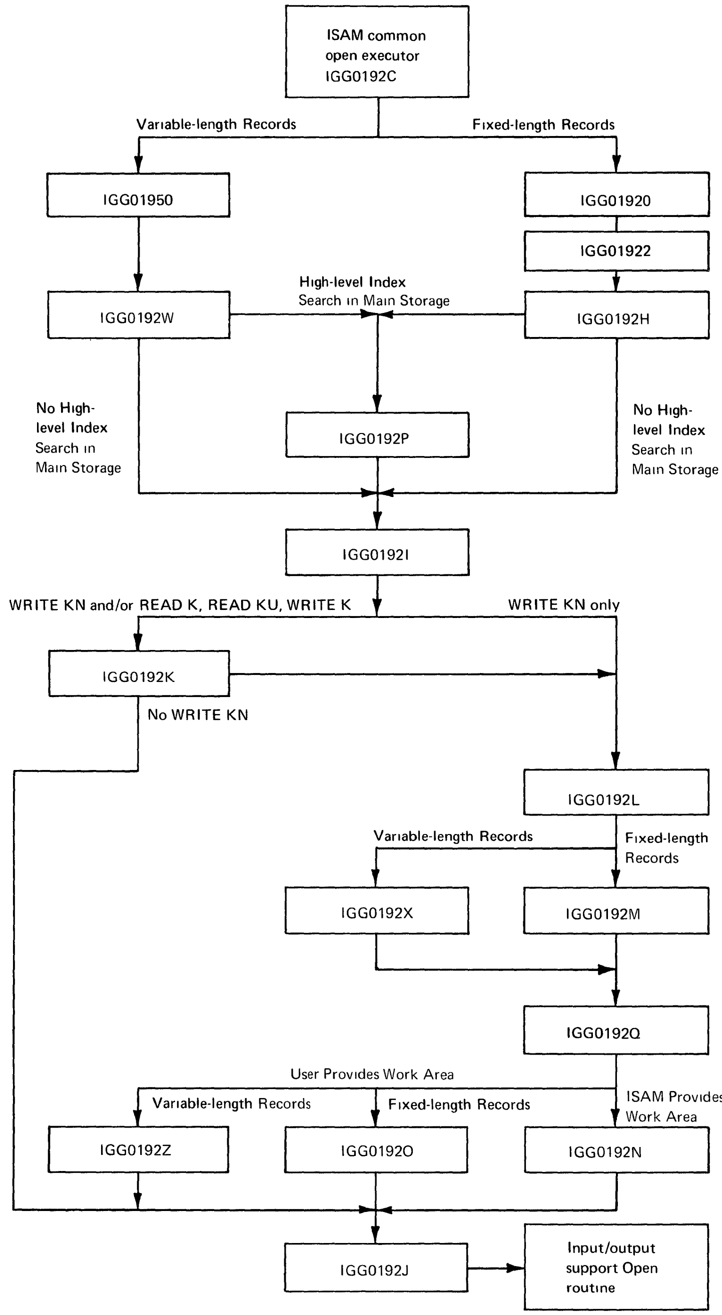


Figure 28. Flow of Control through BISAM Open Executors

BISAM Open Executor IGG0192O (WRITE KN, fixed–length records, user work area)

1. Initializes CP 12 or CP 13 series, and CP 123W; deletes skeleton channel program modules.

BISAM Open Executor IGG0192N (WRITE KN, fixed–length records, system work area)

1. Initializes CP 9 series or CP 11 series; deletes skeleton channel program modules.

BISAM Open Executor IGG0192Z (WRITE KN, variable–length records)

1. Initializes CP 12AV, CP 12BV, and CP 123WV; deletes skeleton channel program modules.

BISAM Open Executor IGG0192J

1. Module IGG0192J selects and loads the proper appendage modules and one asynchronous module. Refer to the BISAM appendage and asynchronous modules tables shown in Figures 37 and 38.
2. Initializes the interrupt request block (IRB) used by the asynchronous routine.
3. If any of the RPS bits at DEBRPSID in the DEB are set, IGG0192J loads the RPS SIO appendage, IGG019JH.

During processing, if bit 3 of DEBRPSID is on, control is passed to IGG019JH.

BISAM Processing Phase Operations

BISAM processing is performed by channel programs that read and search indexes, prime–data tracks, and overflow chains. They also write prime–data and overflow records and index entries. The channel programs are set up and controlled by the BISAM processing routines.

All BISAM READ and WRITE macro instructions enter a nonprivileged macro–time routine, which enters a privileged macro–time routine where I/O interruptions may be readily enabled or disabled. The privileged routine returns to the nonprivileged routine upon completion. The nonprivileged routine then starts a channel program, if possible, and returns control to the user.

When a channel program ends, the I/O supervisor passes control to an appendage routine that analyzes the manner in which the channel program ended and determines the action to be taken as a result. This involves either an EXCP return to the I/O supervisor or the scheduling of an asynchronous routine. The overall control flow through these routines is shown in Figure 7.

The user can supply his own buffers or use the dynamic buffering option of BISAM. In the latter case, the dynamic buffering routine obtains and frees buffers for each processing request.

A check routine is available to all BISAM requests to allow the user to analyze processing errors.

Information about the data set and the processing requests is communicated among the processing routines and the channel programs in control blocks, work areas, and queues. This section describes the processing routine logic, the flow of control through the channel programs, and the relations of the data areas to each other and to the processing routines and channel programs.

Descriptions of the channel programs are in "Appendix B: ISAM Channel Programs." "Section 5: Data Areas" contains detailed layouts of the data areas.

An Example of BISAM Processing Flow

Whenever a BISAM macro is issued, a nonprivileged macro–time module is entered. In this example the nonprivileged module entered will be IGG019JW after a WRITE KN macro instruction is issued.

1. The WRITE KN is issued from the processing program.
2. The nonprivileged module is entered; module IGG019JW issues an SVC 54 to disable interrupts and link to the privileged macro–time routine. In the case of a WRITE KN without READ K, WRITE K, or READ KU, the privileged routine module entered is IGG019JX. (See Figure 35.)
3. Module IGG019JX:
   a. Initializes the IOB.
   b. Determines if another WKN is in progress; if so, the IOB is added to the *on–schedule* queue and the on–schedule switch is set on.
   c. If another WKN is *not* in progress and it is necessary to search the high–level index in main storage, the following operations are done:
      (1) The first WKN channel program is initialized.
      (2) The Seek address for the channel program is determined, using the DCBFTHI field.
      (3) If the track index is the highest level of index (this is assumed for this example), the appendage code is set to 8.
4. Channel program 8 is initialized — CP 8 is used to determine where the new record should be inserted.
5. Return to the SVC 54 issued by IGG019JW.
6. The SVC 54 exits to the original nonprivileged module.
7. Module IGG019JW tests the on–schedule switch; if it is set, return is made to the processing program. If the on–schedule switch is off, an EXCP is issued using the IOB just created.
8. When the channel program ends, the appendage routine uses the appendage code in the IOB and the appendage vector table in the appendage module to select the needed appendage routine for this particular channel program.

Privileged Macro–time Routines

A privileged macro–time routine (shown in Figure 29) schedules the first channel program for a given macro instruction. BISAM has several modules of privileged macro–time routines (refer to Figure 35). However, no more than one of these modules is loaded into storage by the BISAM open executor, IGG0192I, for a single DCB.

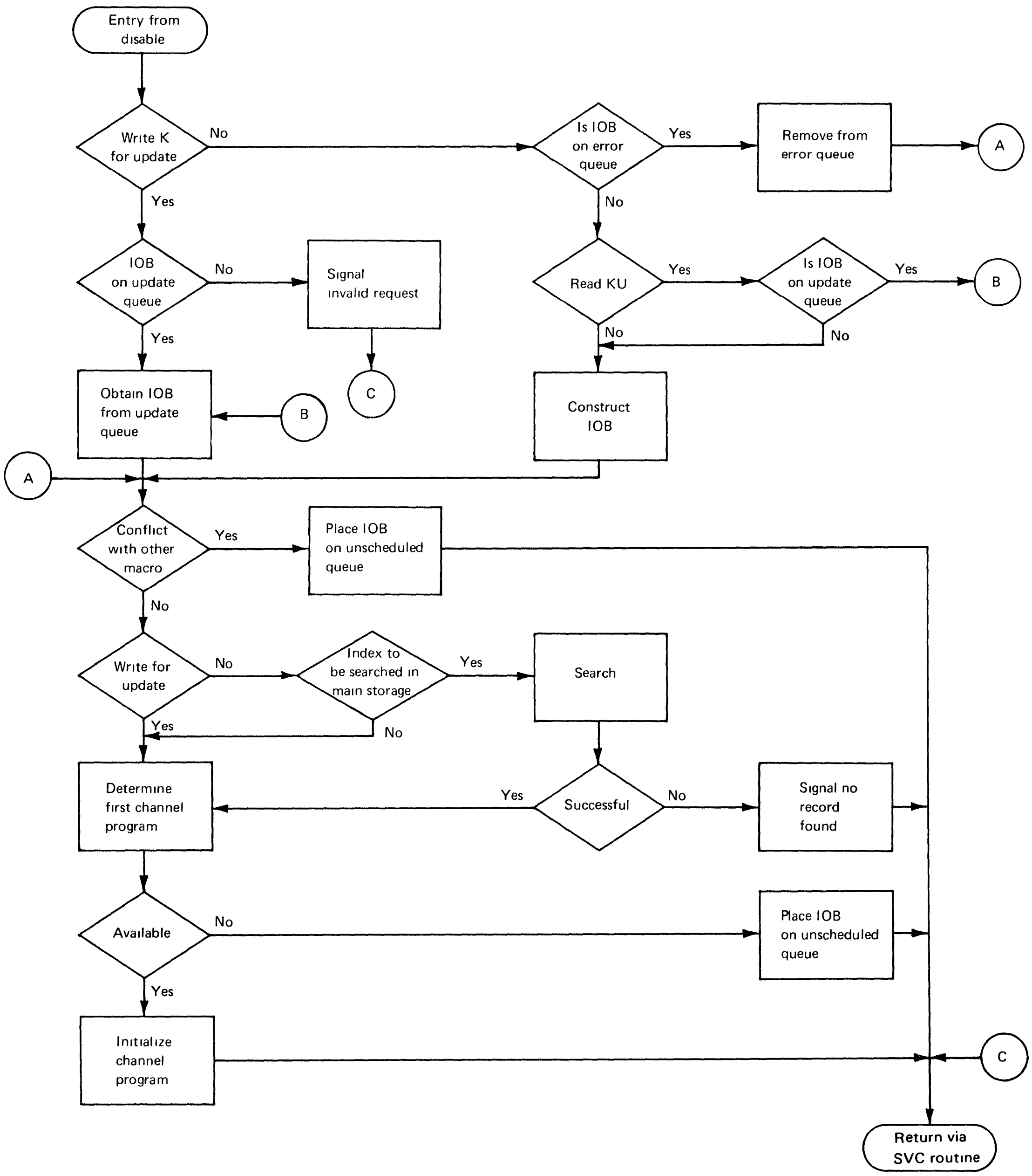


Figure 29. Privileged Macro–time Routines

Selection of the macro–time routine module to be loaded depends on the BISAM macro instructions specified in the DCB, the record format, and the number of levels of index searched on a direct–access device (rather than searched in main storage). These factors determine the choice of channel programs needed in a macro–time routine.

A nonprivileged macro–time routine enters a privileged macro–time routine by means of an SVC 54 (disable) instruction to disable I/O interruptions. If the IOB being reused has a dynamic buffer associated with it, the buffer is returned to the dynamic buffer pool.

For any read or write request, the routine checks the error queue and the update queue to see if any existing IOB refers to the data event control block (DECB) of the present request. If so, the old IOB is reused for the current request. If the IOB being reused has a dynamic buffer associated with it, the buffer is returned to the dynamic buffer pool unless the request requires a dynamic buffer. If no IOB is found that refers to the DECB of the present request, and a dynamic buffer must be assigned to the request, DECBAREA is zeroed to force the assignment of a dynamic buffer in function 1 of the dynamic buffer module (IGG019JI).

When a WRITE K macro instruction is issued after a READ KU, both with the same DECB, an IOB for the DECB should be on an update queue (as the result of the READ KU). If the IOB is not on the update queue, an invalid request condition exists and the privileged routine returns to the calling nonprivileged routine. Otherwise, the privileged routine for the WRITE K associated with a previous READ KU removes the IOB from the update queue. In all other cases, the routine constructs an IOB for the request.

Subsequently, the privileged routine attempts to schedule the first channel program needed for the user's request. If the channel program is available and the high–level index is to be searched in main storage, the routine performs this search. If the search is unsuccessful, a *record–not–found* condition exists and the routine posts the DECB as complete, sets the appropriate exceptional condition bit in DECBEXCD, and returns control to the nonprivileged routine. (Searching is always successful in the case of WRITE KN.) If the search is unsuccessful or no search in main storage is necessary, the routine determines the first channel program to be scheduled. If it is available, the routine schedules it. If it is unavailable, an unscheduled condition exists, and the routine queues a request for the channel program by placing the IOB on a queue called the unscheduled queue. The routine then returns to the nonprivileged routine.

A special case exists if the WRITE KN macro instruction is being used with other READ or WRITE macro instructions. Possible conflicts between these macro instructions are avoided because WRITE KN changes indexes and record positions. Its channel programs are not scheduled if another WRITE KN, WRITE K, READ K, or READ KU has been scheduled but not completed, or if a READ KU has been completed but a FREEDBUF or a WRITE K for that DECB has not. The WRITE KN channel programs are not scheduled if there are IOBs on the update queue, or if there are IOBs on the unscheduled queue for reasons other than those associated with WRITE KN. Similarly, WRITE K, READ K, and READ KU are not scheduled if a WRITE KN has been scheduled but not completed, or if a previous WRITE KN cannot be scheduled.

Note: Entry to the privileged routine from the asynchronous routine is also possible. In this case, the return will be to the asynchronous routine.

Nonprivileged Macro–time Routines

There are two modules of nonprivileged macro–time routines. (Refer to Figure 36.) The READ K, READ KU, and WRITE K macro instructions link to one routine and the WRITE KN macro instruction links to the other. The nonprivileged routine is shown in Figure 30.

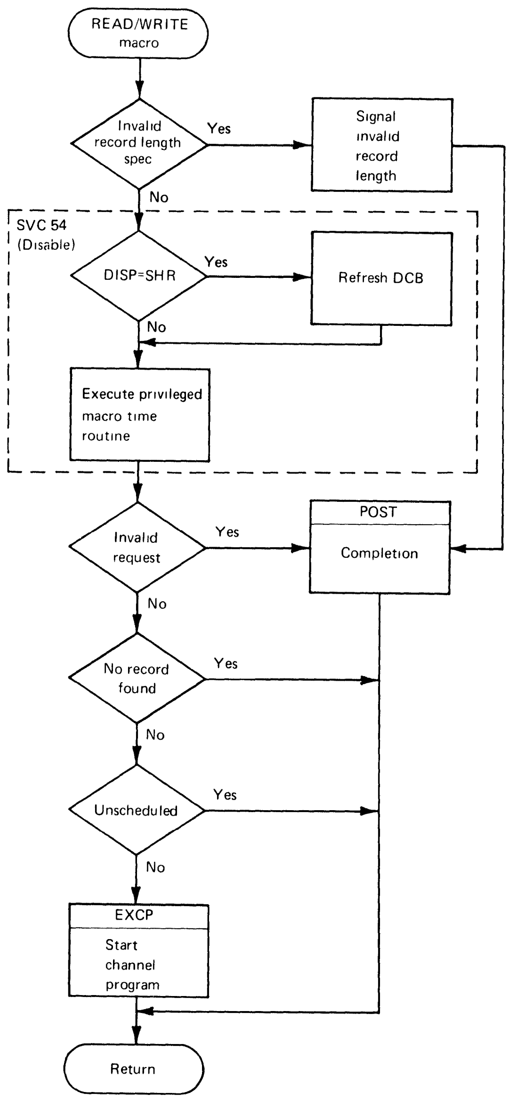


Figure 30. Nonprivileged Macro–time Routines and SVC 54

If the user has specified a record length in a READ K, READ KU, or WRITE K macro instruction, the respective macro instruction routine checks the record length specified against the logical record length supplied by the user in the DCB (DCBLRECL). If the length specified in the macro instruction is invalid or if the user has specified a record length in a WRITE KN macro instruction, the nonprivileged macro–time routines set the record length check indicator in the DECB exceptional condition code field (DECBEXCD1) and return control to the user. Otherwise, an SVC 54 is issued to link to a privileged macro–time routine. The privileged routine, upon completion, returns to the nonprivileged routine.

If no channel program was scheduled, the nonprivileged macro–time routine issues the EXCP and returns to the user. When the channel program is completed, an I/O interruption takes place and the I/O supervisor links to an appendage routine. (Appendage routines are described in the BISAM "Appendage and Asynchronous Routines" section.)

If no channel program was scheduled because of an invalid request, a no–record–found condition, or an unscheduled condition, the nonprivileged routine returns to the user. In the case of an invalid request, the routine posts the DECB as complete and returns to the user.

Appendage and Asynchronous Routines

The BISAM appendages and asynchronous routines are shown in Figure 31. The asynchronous modules are listed in Figure 37; the appendage modules are listed in Figure 38.

Appendage routines determine the action to be taken when a channel program ends. Asynchronous routines perform that action except in certain cases, which are explained below. Appendage modules consist of an appendage vector table and a group of appendage routines. Asynchronous modules consist of an asynchronous vector table and a group of asynchronous routines.

When a channel program ends, a general appendage routine uses a combination of the appendage code in the IOB and the appendage vector table for the module to select the appropriate appendage routine. A list of appendage and asynchronous codes is contained in "Section 6: Diagnostic Aids."

If the channel program is complete, the appendage routine schedules an asynchronous routine that sets up the next channel program. If the channel program is not complete, the appendage routine returns to IOS to reschedule that channel program.

If the channel program did not end in error, the action taken depends on whether (1) it is the final channel program needed to satisfy the user's request, (2) an additional channel program is needed to satisfy the request and no other requests are waiting for the channel program just completed, or (3) neither of the above conditions exists.

In the first case, the appendage routine schedules an asynchronous routine to report completion to the user. If the data set is shared (DISP=SHR), the DCBFA (DCB field area) is reset as needed before completion is posted. In the second case, the appendage routine schedules the additional channel program by a special return to the I/O supervisor. In the third case, the appendage schedules an asynchronous routine which in turn schedules an additional channel program for the current request and, if possible, reschedules the channel program just completed for a waiting request.

If the present request used a dynamic buffer, the address of the buffer is saved in the IOB before the IOB is placed on either the update or error queue.

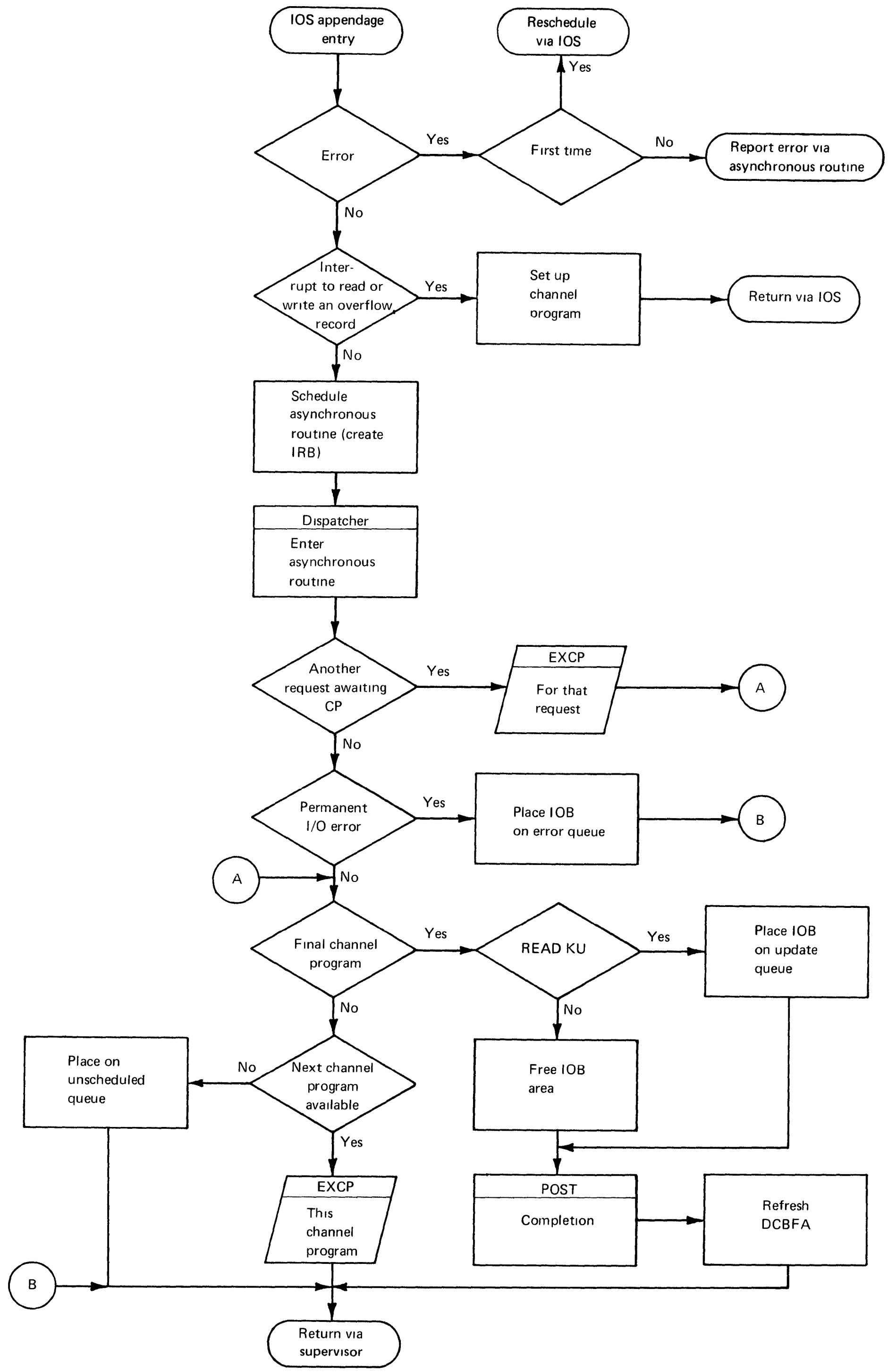


Figure 31. BISAM Appendage and Asynchronous Routines

The first time a channel program ends in error, the appendage routine returns control to the I/O supervisor to retry the operation. If the I/O supervisor finds the error is permanent, it reenters the appendage routine which schedules an asynchronous routine to report the error to the user and place the request on the error queue.

Dynamic Buffering Routines

The READ K and READ KU macro instructions require an area into which a block can be read. The user may supply this area or use BISAM routines to provide the area through the dynamic buffering option of the macro instruction. Figure 32 shows the dynamic buffering routines.

When the dynamic buffering option is used, BISAM routines release the buffer when a corresponding WRITE K macro is completed. If no WRITE K is issued, the processing program may release the area obtained with dynamic buffering for a READ K or READ KU by issuing a free dynamic buffer (FREEDBUF) macro instruction.

Also, the privileged macro routine automatically releases the buffer if a READ macro instruction is followed by a WRITE KN or another READ. The buffer is released, reusing a DECB, without an intervening WRITE K or FREDBUF.

The dynamic buffering module contains two routines. The first, called *function 1*, obtains buffers in response to the dynamic buffering option of a READ K or READ KU macro instruction. The second routine, called *function 2*, frees the buffers.

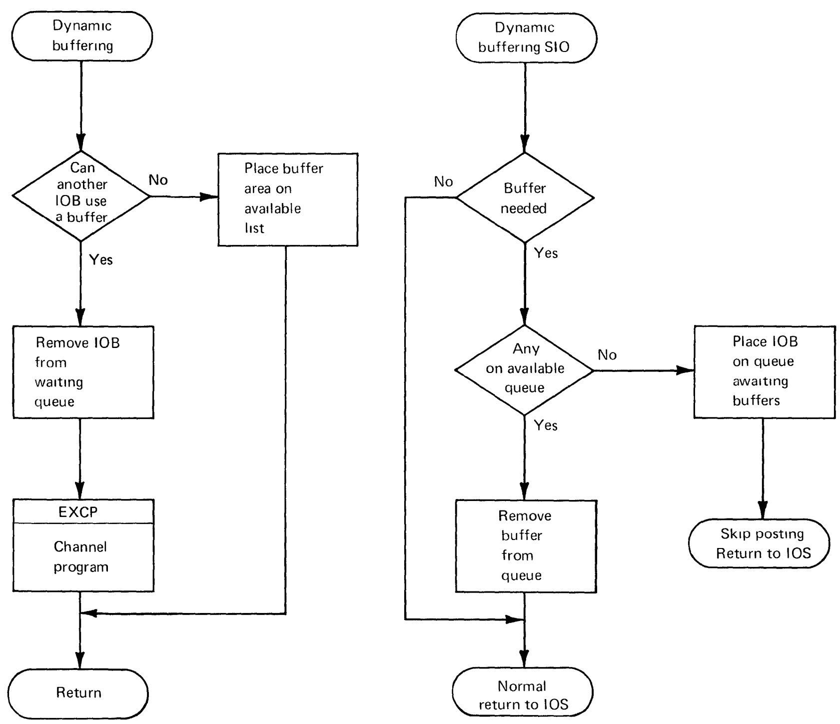


Figure 32. Dynamic Buffering Routine

Function 1 is an appendage routine entered by the I/O supervisor just prior to executing the scheduled channel program. When used by the FREEDBUF macro instruction, function 2 is considered a macro–time routine. When used on completion of a WRITE K macro instruction, function 2 is considered an asynchronous routine. The function 2 routine of IGG019JI, when executed from FREEDBUF, also frees any IOB on the error or update queue that is associated with the DECB, regardless of whether a dynamic buffer is also associated with the DECB.

Rather than returning to IOS, IGG019JI passes control to the RPS SIO appendage (IGG019JH) if bit 3 of DEBRPSID is set.

A description of the BISAM dynamic buffering buffer control block appears in "Section 5: Data Areas."

Check Routine

The check routine module (shown in Figure 33), loaded when check is specified in the DCBMACRF field, gets control each time the user issues a CHECK macro instruction.

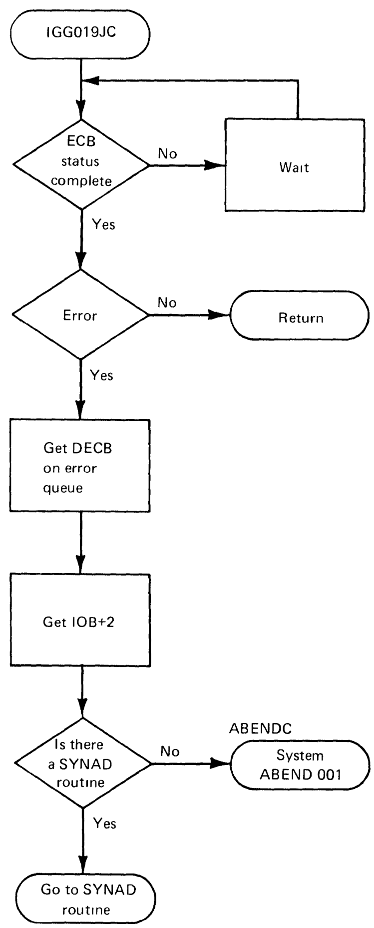


Figure 33. BISAM Check Routine

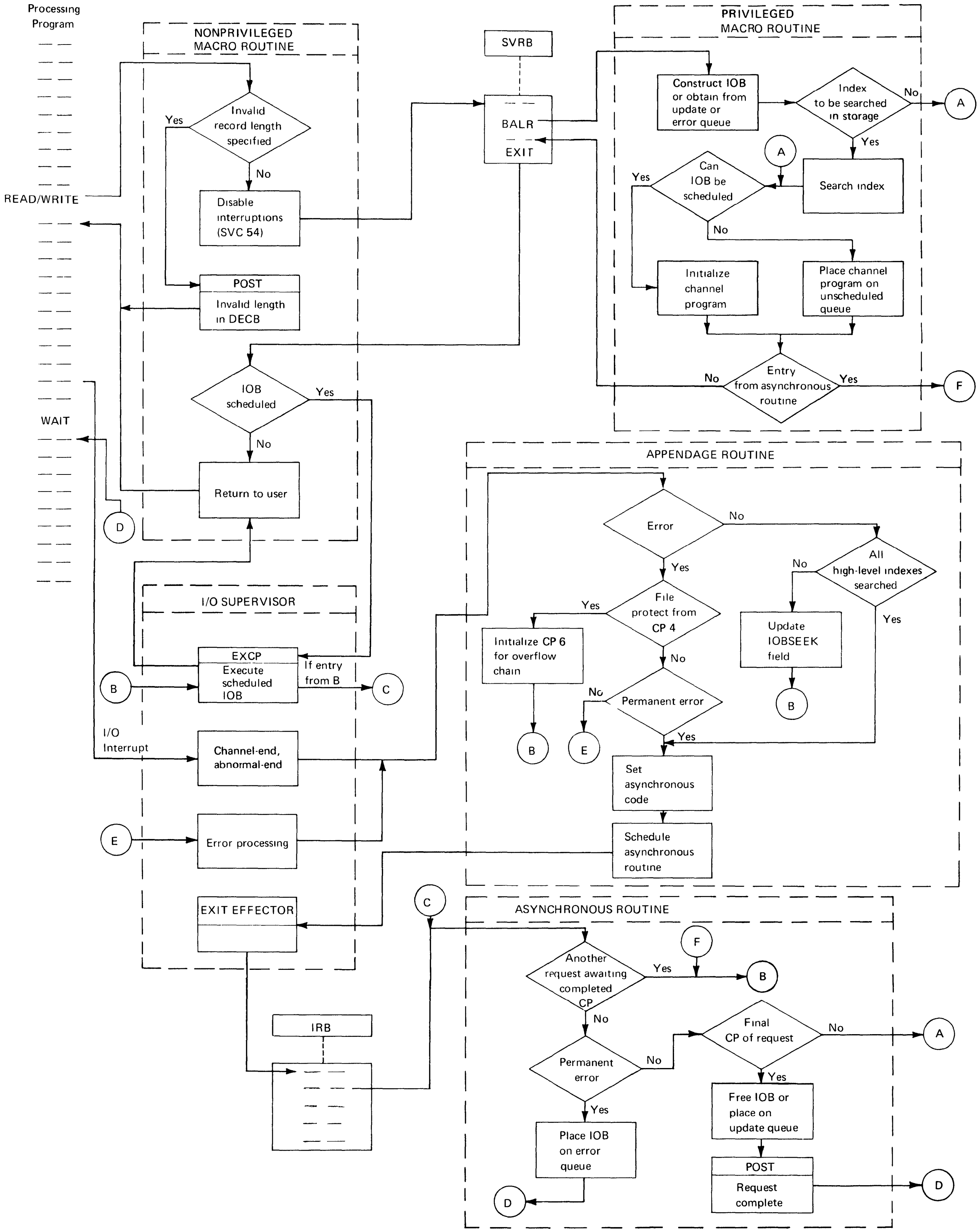


Figure 34. BISAM Processing Flow (Not WRITE KN)

The check routine examines the DECB exception code (DECBEXCD) fields. If a permanent error has been posted, it searches the error queue for the corresponding IOB. The check routine then either gives control to the user's synchronous error (SYNAD) routine or, if the user has no SYNAD routine, issues SVC 55 (EOV) to request an ABEND with a code of 001.

Upon entry to the SYNAD routine, register 0 contains the address of the first sense byte of the IOB (sense information is valid only when a unit check has occurred) and register 1 contains the address of the DECB. In the SYNAD routine, the user can issue a SYNADAF macro instruction. It places all pertinent information on the request in a buffer and returns the buffer's address to the user. For a description of the SYNADAF macro instruction, refer to *OS Data Management Macro Instructions,* GC26–3794.

Macro Instructions	Additional Considerations		Module Names
READ K, WRITE K READ KU	Fixed-length Records	*NLSD=0	IGG019J6
		NLSD≠0	IGG019J7
	Variable-length Records		IGG019H7
WRITE KN	None		IGG019JX
READ K, WRITE K READ KU in combination with WRITE KN	Fixed-length Records	NLSD=0	IGG019J0
		NLSD≠0	IGG019J3
	Variable-length Records		IGG019H3

*NLSD represents the number of levels of indexing (cylinder or master indexes) that are searched on the device.

NLSD=0 represents the case where the data set was allocated no more than one cylinder and has no cylinder or master indexes or there is only a cylinder index and it is searched in main storage.

NLSD≠0 means: (1) there is only a cylinder index that is searched on the device and (2) there are at least two levels of indexing, one of which is searched in main storage and the other is searched on the device.

Figure 35. BISAM Privileged Macro–time Modules

Macro Instructions	Additional Considerations	Module Names
READ K, WRITE K, READ KU	None	IGG019JV
WRITE KN	None	IGG019JW

Figure 36. BISAM Nonprivileged Macro–time Modules

<table>
<tr><th>Macro Instruction</th><th colspan="2">Additional Considerations</th><th>Modules</th></tr>
<tr><td rowspan="2">READ K, WRITE K, READ KU</td><td colspan="2">Fixed-length Records</td><td>IGG019GX</td></tr>
<tr><td colspan="2">Variable-length Records</td><td>IGG019IX</td></tr>
<tr><td rowspan="3">WRITE KN</td><td rowspan="2">Fixed-length Records</td><td>No Write Check</td><td>IGG019GY</td></tr>
<tr><td>Write Check</td><td>IGG019GV</td></tr>
<tr><td colspan="2">Variable-length Records</td><td>IGG019IY</td></tr>
<tr><td rowspan="3">READ K, WRITE K, READ KU
in combination with
WRITE KN</td><td rowspan="2">Fixed-length Records</td><td>No Write Check</td><td>IGG019GZ</td></tr>
<tr><td>Write Check</td><td>IGG019GW</td></tr>
<tr><td colspan="2">Variable-length Records</td><td>IGG019IZ</td></tr>
</table>

Figure 37. BISAM Asynchronous Modules

BISAM Processing Phase Organization

BISAM Channel Programs

BISAM uses the channel programs that are enumerated below and described in Appendix B. The flow of control through the READ K, WRITE K, and READ KU channel programs is shown in Figure 40 and the flow for WRITE KN channel programs is shown in Figures 41 through 52. Channel program modules are indicated in Figure 39.

Note: Figures 40 through 52 show only the normal (nonerror) flow of control through the channel programs. For WRITE KN, two different methods are used to add records to the data set. For fixed–length records with a system work area, the prime track is rewritten and the index entries are updated before the overflow record is written. For fixed–length records with a user–supplied work area and for variable–length records, the overflow record is written before the prime track and index entries. This requires two different methods for executing CP 14 as explained in "Appendix B: ISAM Channel Programs."

Macro Instructions	Additional Considerations		Module Names
READ K, WRITE K, READ KU	Fixed-Length Records	No Write Check	IGG019G8
		Write Check	IGG019G9
	Variable Length Records		IGG019I9
WRITE KN	Fixed-Length Records	Unblocked, System Work Area, No Write Check	IGG019G0 and IGG019GL
		Unblocked, System Work Area, Write Check	IGG019G1 and IGG019GM
		Unblocked, User Work Area, No Write Check	IGG019G2 and IGG019GL
		Unblocked, User Work Area, Write Check	IGG019G3 and IGG019GM
		Blocked, System Work Area, No Write Check	IGG019G4 and IGG019GL
		Blocked, System Work Area, Write Check	IGG019G5 and IGG019GM
		Blocked, User Work Area, No Write Check	IGG019G6 and IGG019GL
		Blocked, User Work Area, Write Check	IGG019G7 and IGG019GM
	Variable-Length Records		IGG019IO and IGG019IM
READ K, WRITE K, READ KU in combination with WRITE KN	Fixed-Length Records	Unblocked, System Work Area, No Write Check	IGG019G0 and IGG019GN
		Unblocked, System Work Area, Write Check	IGG019G1 and IGG019GO
		Unblocked, User Work Area, No Write Check	IGG019G2 and IGG019GN
		Unblocked, User Work Area, Write Check	IGG019G3 and IGG019GO
		Blocked, System Work Area, No Write Check	IGG019G4 and IGG019GN
		Blocked, System Work Area, Write Check	IGG019G5 and IGG019GO
		Blocked, User Work Area, No Write Check	IGG019G6 and IGG019GN
		Blocked, User Work Area, Write Check	IGG019G7 and IGG019GO
	Variable-Length Records		IGG019IO and IGG019IN
RPS SIO Appendage			IGG019JH

Figure 38. BISAM Appendage Modules

<table>
<tr><th colspan="2">Macro Instructions</th><th>Additional Considerations</th><th>Module Names</th><th>Channel Programs</th></tr>
<tr><td colspan="2" rowspan="2">Any READ or WRITE</td><td>NLSD = 1</td><td>IGG019JK</td><td>2</td></tr>
<tr><td>NLSD > 1</td><td>IGG019JJ</td><td>1</td></tr>
<tr><td colspan="2" rowspan="2">READ K, WRITE K, READ KU</td><td>None</td><td>IGG019JL</td><td>4 5 6 7</td></tr>
<tr><td>Write Check</td><td>IGG019JM</td><td>4 5W 6W 7W</td></tr>
<tr><td rowspan="9">WRITE KN</td><td rowspan="8">Fixed-Length Records</td><td>Unblocked, System Work Area, No Write Check</td><td>IGG019JN</td><td>8 9A 9B 9C 10A 10B 14 15 16 17</td></tr>
<tr><td>Unblocked, System Work Area, Write Check</td><td>IGG019JP</td><td>8 9A 9BW 9CW 10AW 10BW 14W 15 16 17W</td></tr>
<tr><td>Unblocked, User Work Area, No Write Check</td><td>IGG019JR</td><td>8 10A 10B 12A 12B 12C 14 15 16 17</td></tr>
<tr><td>Unblocked, User Work Area, Write Check</td><td>IGG019JT</td><td>8 10AW 10BW 12A 12B 12CW 14 15 16 17W 123W</td></tr>
<tr><td>Blocked, System Work Area, No Write Check</td><td>IGG019JO</td><td>8 10A 10B 11A 11B 14 15 16 17</td></tr>
<tr><td>Blocked, System Work Area, Write Check</td><td>IGG019JQ</td><td>8 10AW 10BW 11A 11BW 14W 15 16 17W</td></tr>
<tr><td>Blocked, User Work Area, No Write Check</td><td>IGG019JS</td><td>8 10A 10B 13A 13B 13C 14 15 16 17</td></tr>
<tr><td>Blocked, User Work Area, Write Check</td><td>IGG019JU</td><td>8 10AW 10BW 13A 13B 13CW 14W 15 16 17W 123W</td></tr>
<tr><td colspan="2">Variable-Length Records</td><td>IGG019HP</td><td>8 12AV 12BV 14/14W 15 16 17 123WV</td></tr>
</table>

Figure 39. BISAM Channel Program Modules

CP 1	Used to search master and cylinder indexes.
CP 2	Used to search a cylinder index when it is the highest level to be searched on a device.
CP 4	Used to search a track index. CP 5 and CP 5W are always appended to this channel program.
CP 5	Used to search prime–data tracks and to read or write prime–data records.
CP 5W	Write–checking version of CP 5.

CP 6	Used to search an overflow chain and read or write overflow records.
CP 6W	Write–checking version of CP 6.
CP 7	Used to write data records when WRITE K is associated with READ KU.
CP 7W	Write–checking version of CP 7.
CP 8	Used to search track indexes and search prime–data tracks for the place to insert a new record. There are separate versions for fixed–length records and variable–length records.

The following channel programs are used for insertion of fixed–length unblocked prime–data records when the work area is provided by the system.

CP 9A	Used to read into the work area the record occupying the position at which an insertion is to be made.
CP 9B	Used to read an even–numbered record after writing a record into the previous slot and write back the last record of a non–EOF track when the number of records bumped is odd.
CP 9BW	Used instead of CP 9B when write–checking is specified.
CP 9C	Used to read an odd–numbered record after writing a record into the previous slot and write back the last record of a non–EOF track when the number of records bumped is even.
CP 9CW	Used instead of CP 9C when write–checking is specified.

The following channel programs are used for fixed–length records regardless of whether they are blocked or unblocked or whether the work area is obtained by the system or the user.

CP 10A	Used to write a record or block to replace an EOF mark.
CP 10AW	Used instead of CP 10A when write–checking is specified.
CP 10B	Used to write an EOF mark.
CP 10BW	Used instead of CP 10B when write–checking is specified.

The following channel programs are used for insertion of fixed–length prime–data records into blocks when the work area is provided by the system.

CP 11A	Used to read into the work area a block to be bumped.
CP 11B	Used to write back a rearranged block.
CP 11BW	Used instead of CP 11B when write–checking is specified.

The following channel programs are used for insertion of fixed–length unblocked prime–data records when the work area is supplied by the user.

CP 12A	Used to read all records from the track following the slot into which a new record is to be inserted.

CP 12B Used to write a new record followed by the records read by CP 12A.

CP 12C Used to write a new record with a key identical to that of a record which, although logically deleted, is still physically present on the track.

CP 12CW Used instead of CP 12C when write–checking is specified.

The following programs are used for insertion of blocked or unblocked variable–length records.

CP 12AV Used to read all records from the track following the slot into which a new record is to be inserted.

CP 12BV Used to write a new record and the records read by CP 12AV.

The following channel programs are used for insertion of fixed–length prime–data records into blocks when the work area is provided by the user.

CP 13A Used to read all blocks from the track following and including the slot into which a record is to be inserted.

CP 13B Used to write back the rearranged blocks read by CP 13A.

CP 13C Used to write back a block if the insertion is a record with a key identical to that of a record which, although logically deleted, is still physically present within the block.

CP 13CW Used instead of CP 13C when write–checking is specified.

The following channel programs are used regardless of whether records are fixed–length or variable–length, blocked or unblocked, or whether the work area is obtained by the system or the user.

CP 14 Used to update track–index entries, update the cylinder overflow control record (COCR), and write overflow records. The six different setups for this channel program are explained in "Appendix B: ISAM Channel Programs."

There are separate versions for fixed–length records and for variable–length records.

For variable–length records and fixed–length records with a user–supplied work area, CP 14 is divided into two parts. Part I writes the overflow record and Part II udpates the COCR and index entries. See "Appendix B: ISAM Channel Programs" for details.

CP 14W Used instead of CP 14 when write–checking is specified.

CP 15 Used to read in the cylinder overflow control record and the overflow track–index entry when a new record is added to the end of a data set.

CP 16 Used to search an overflow chain for the record that logically precedes or is equal to the new record to be added, or the last record in the chain.

CP 17 Used to change the key in a normal or normal–and–overflow track–index entry or in a higher–level index entry.

CP 17W Used instead of CP 17 when write–checking is specified.

CP 87 Used to read a high–level index into main storage.

CP 123W	Addendum to CP 12A and CP 12B or to CP 13A and CP 13B when write–checking is specified (fixed–length records).
CP 123WV	Addendum to CP 12BV when write–checking is specified (variable–length records).

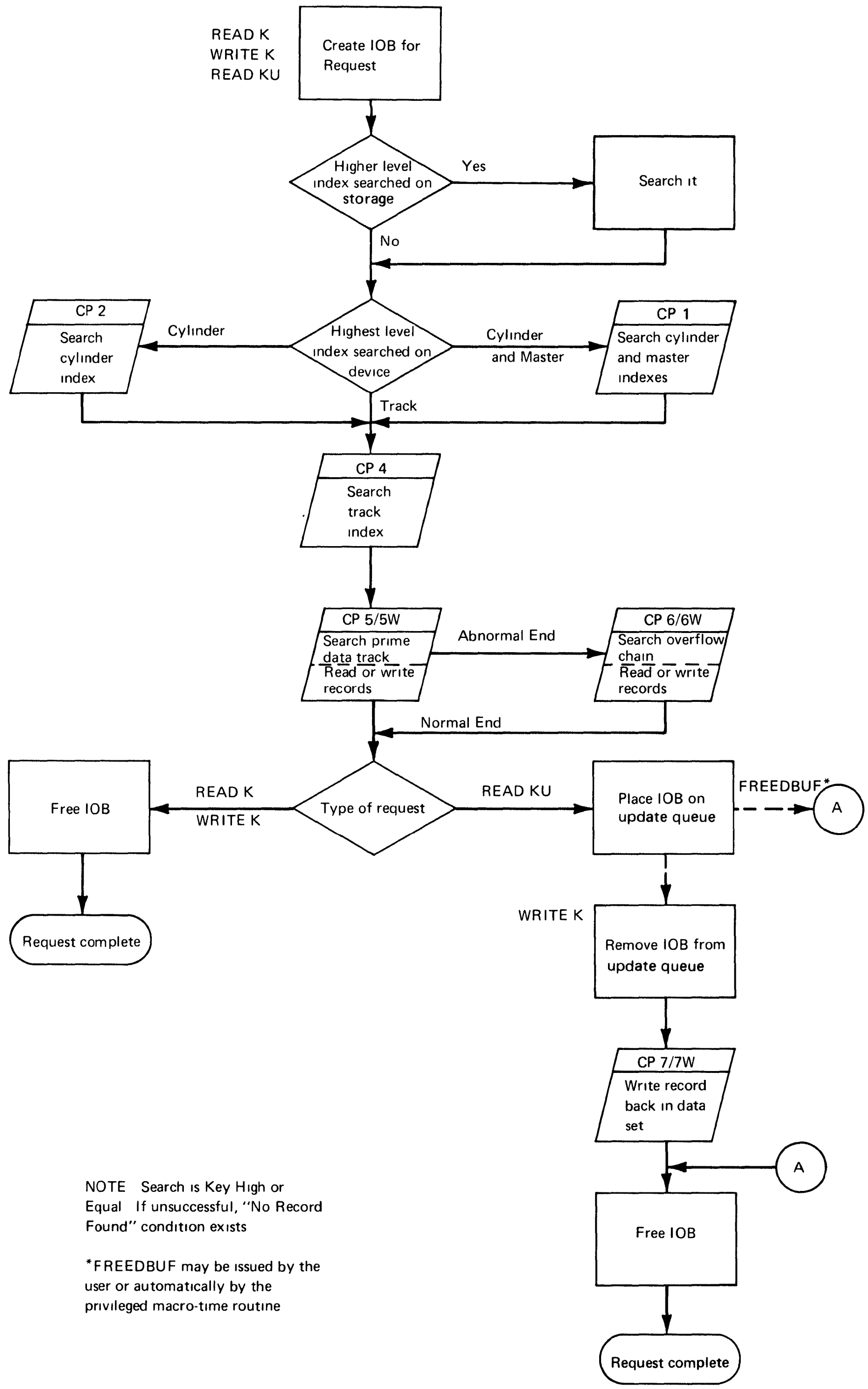


Figure 40. READ K, WRITE K, READ KU Channel Program Flow

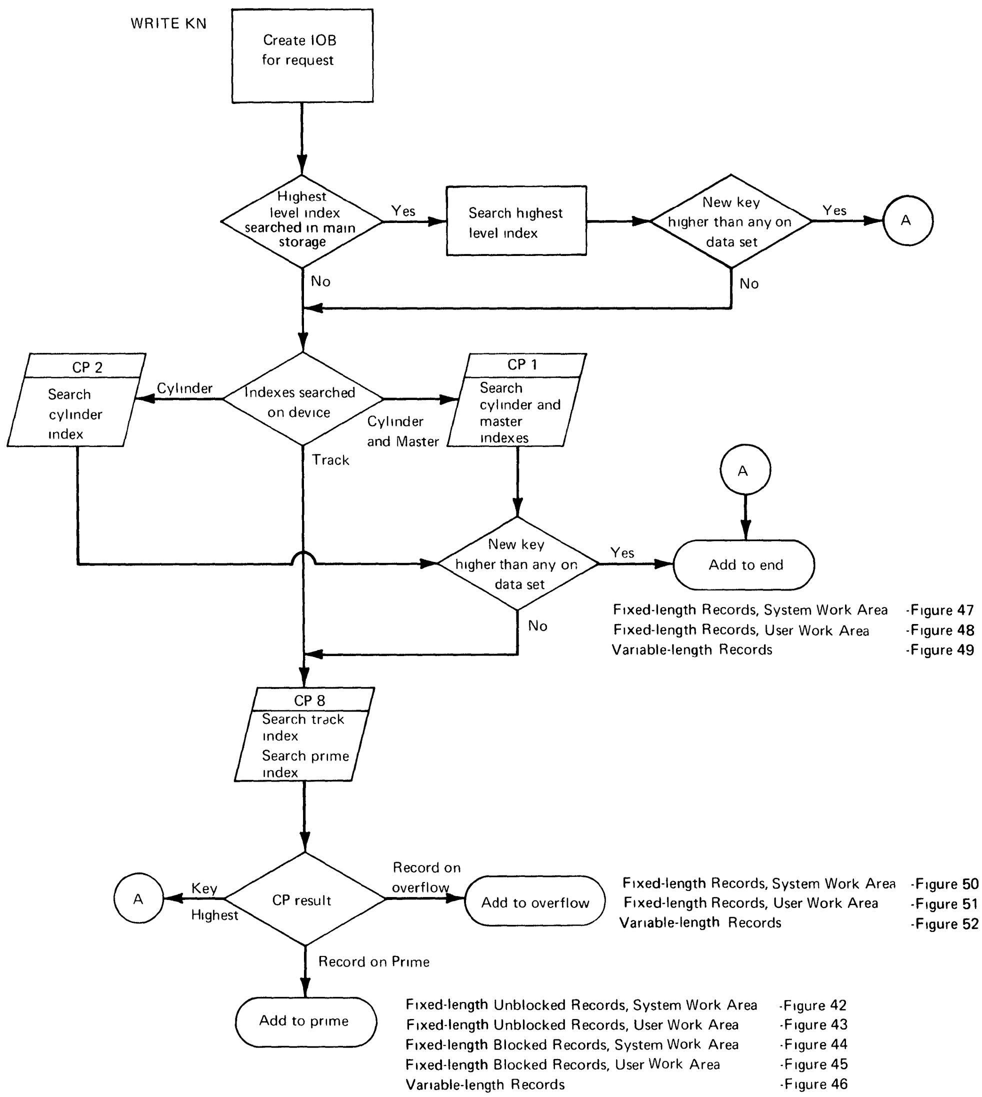


Figure 41. WRITE KN Channel Program Flow — Index Searching

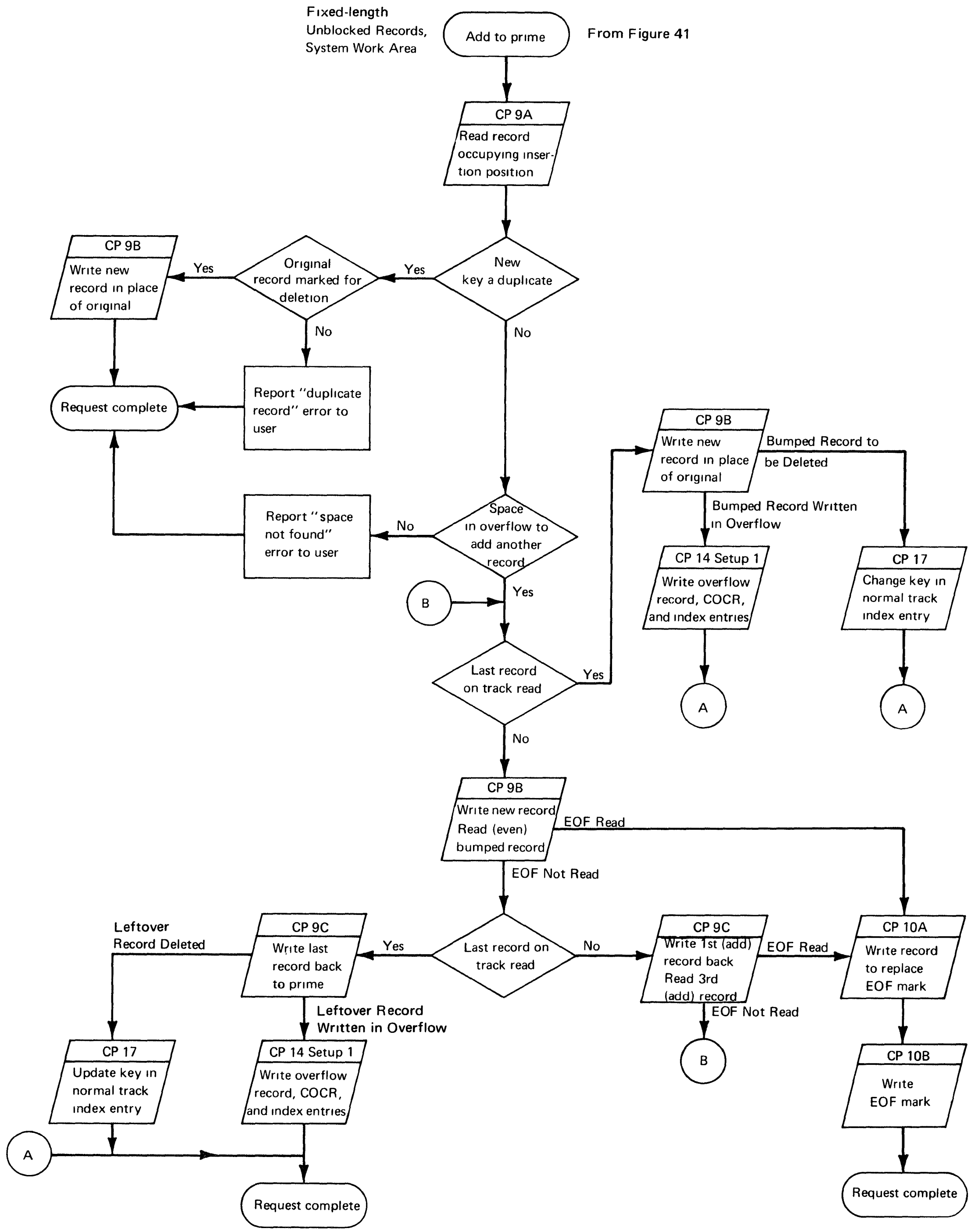


Figure 42. WRITE KN Channel Program Flow — Add to Prime (Fixed–Length Unblocked Records, System Work Area)

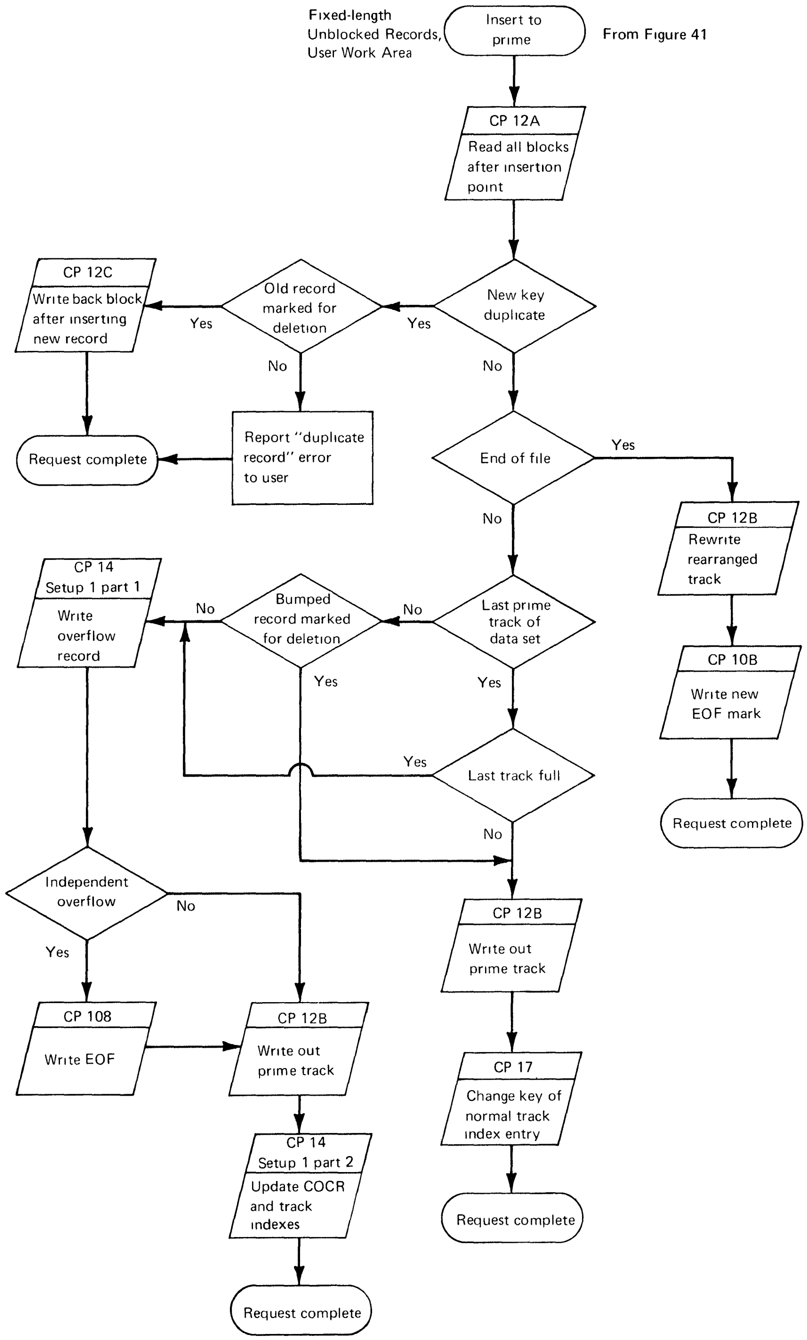


Figure 43. WRITE KN Channel Program Flow — Add to Prime (Fixed– Length Unblocked Records, User Work Area)

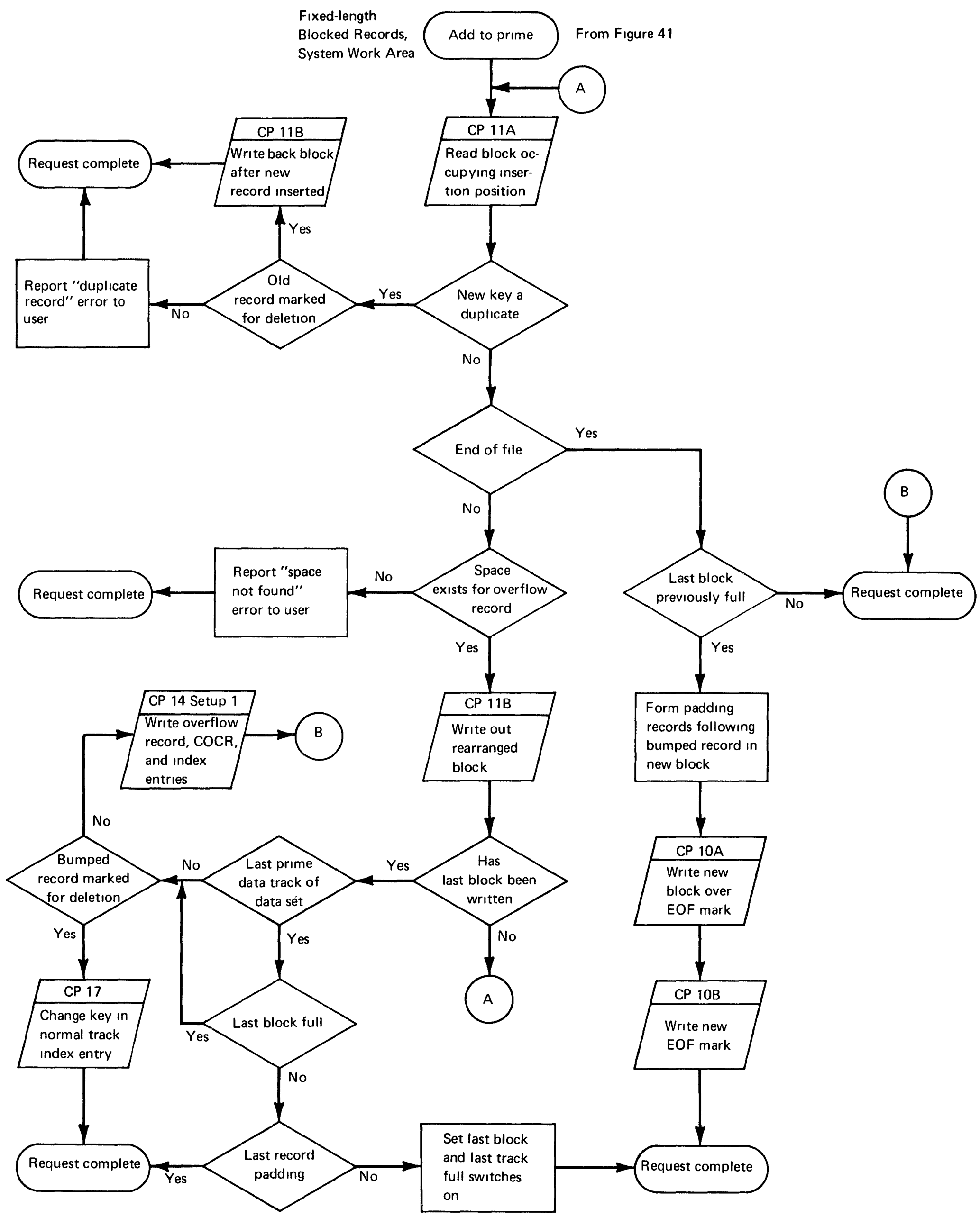


Figure 44. WRITE KN Channel Program Flow — Add to Prime (Fixed–Length Blocked Records, System Work Area)

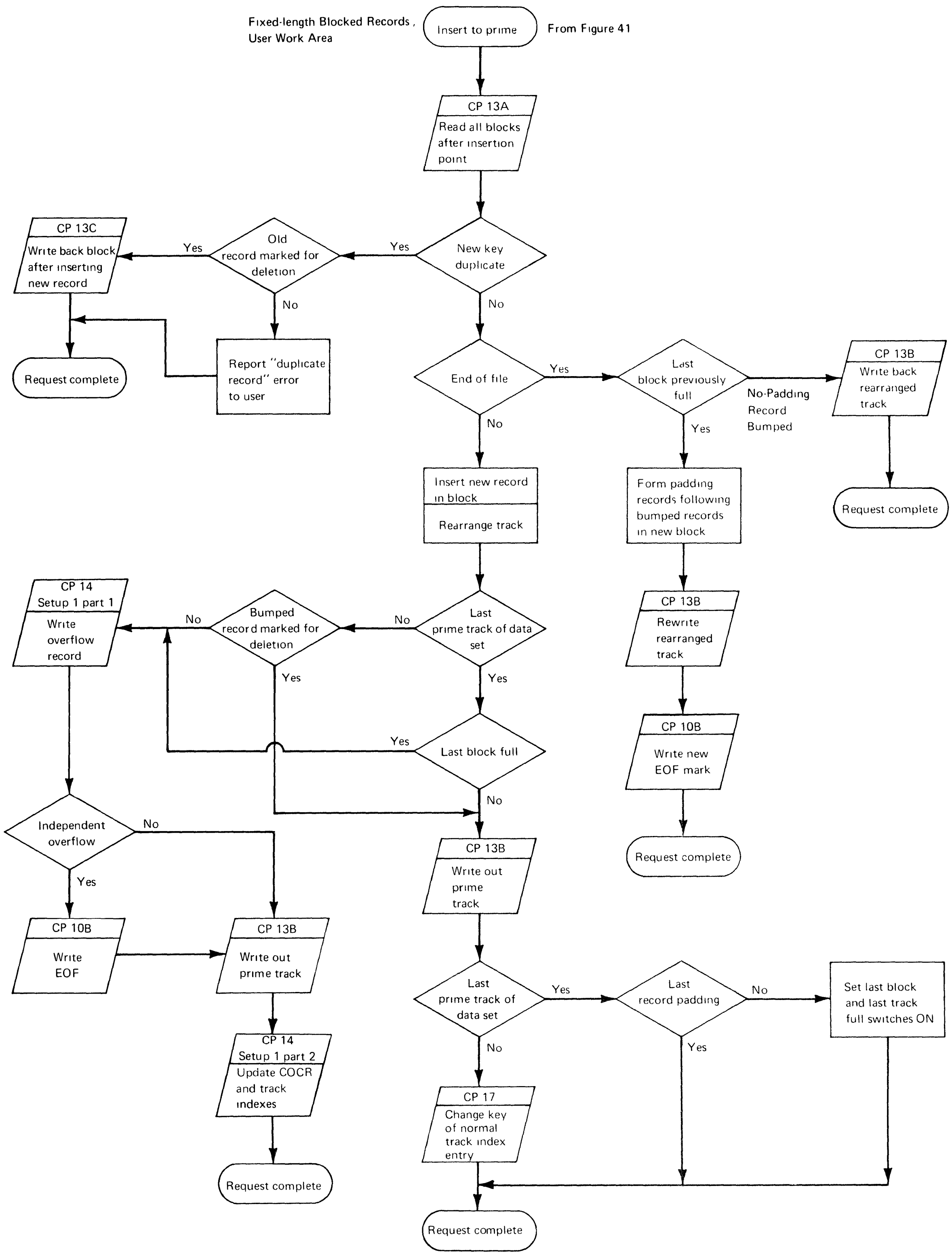


Figure 45. WRITE KN Channel Program Flow — Add to Prime (Fixed– Length Blocked Records, User Work Area)

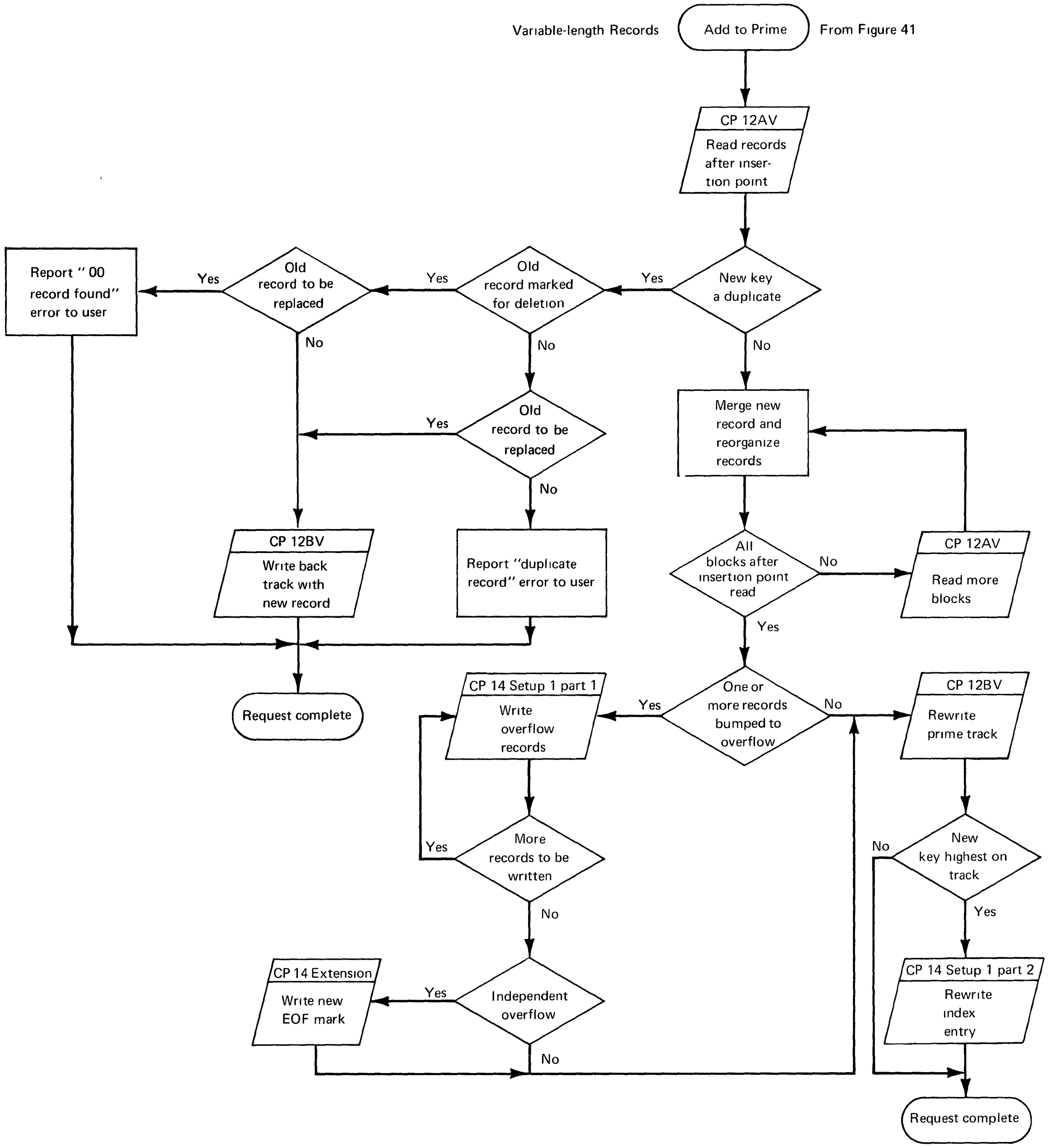


Figure 46. WRITE KN Channel Program Flow — Add to Prime (Variable– Length Records)

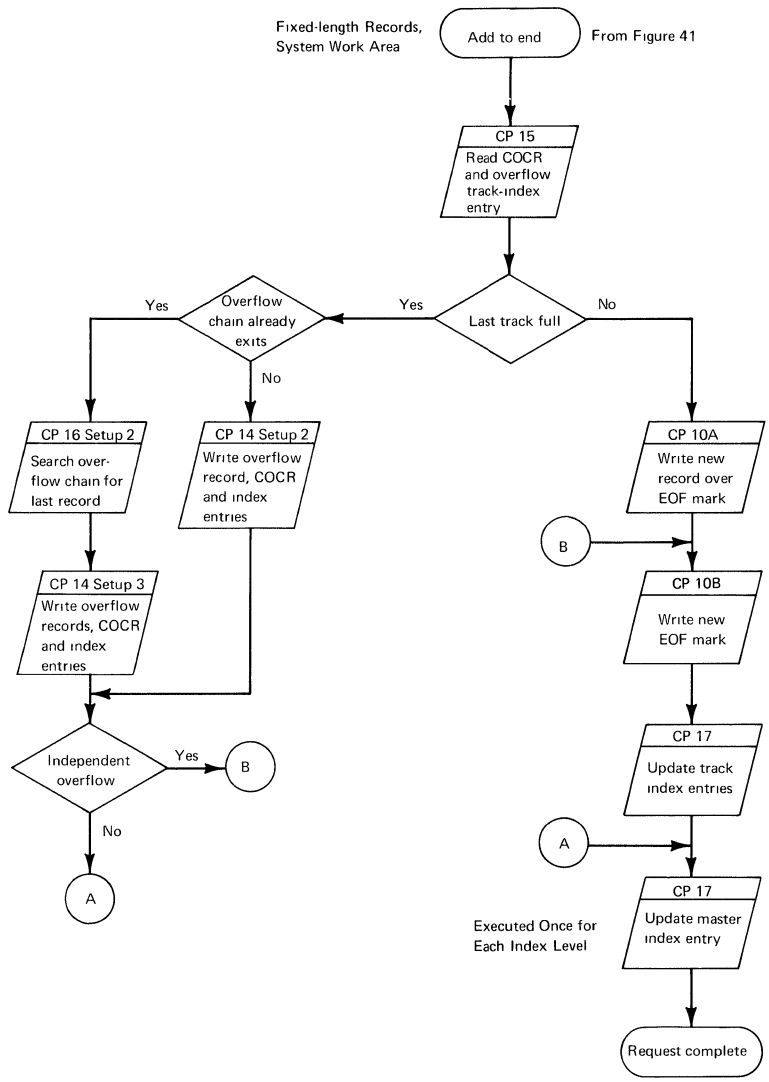


Figure 47. WRITE KN Channel Program Flow — Add to End (Fixed–Length Records, System Work Area)

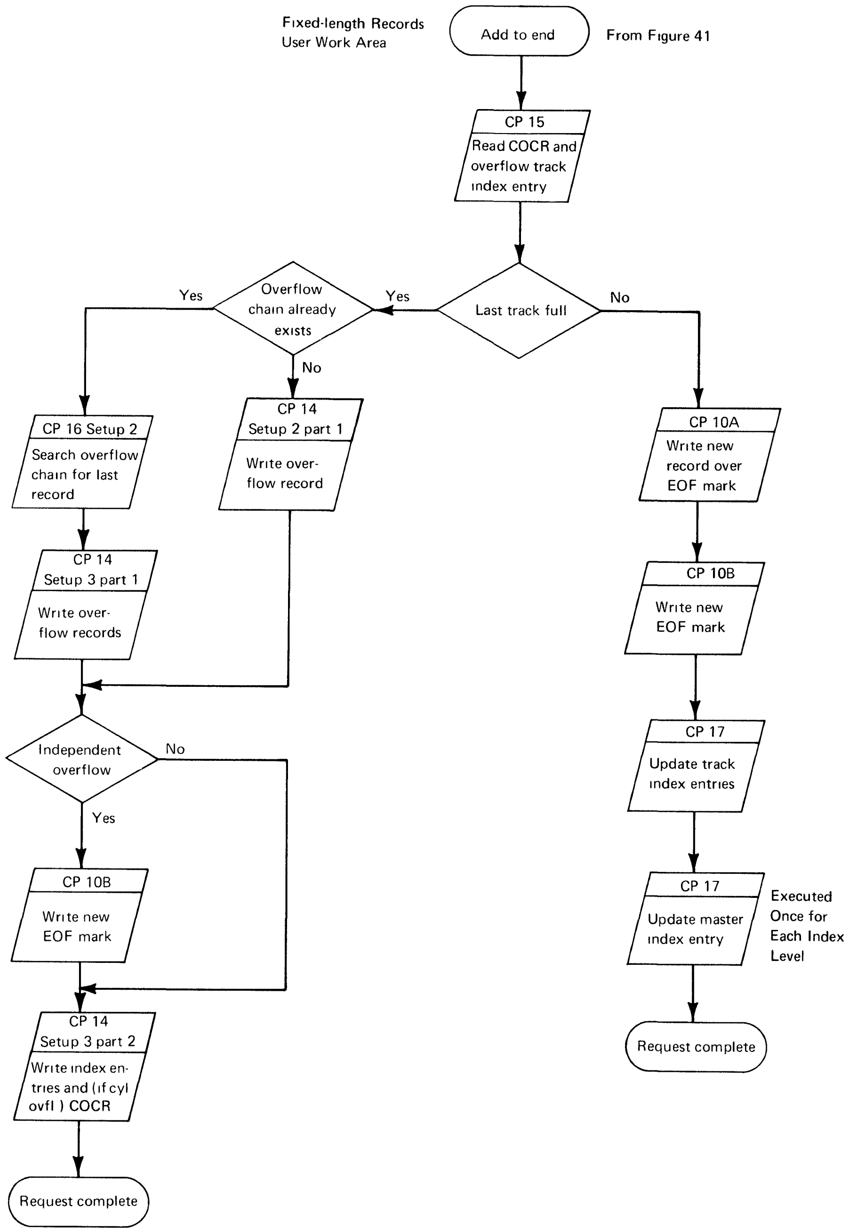


Figure 48. WRITE KN Channel Program Flow — Add to End (Fixed–Length Records, User Work Area)

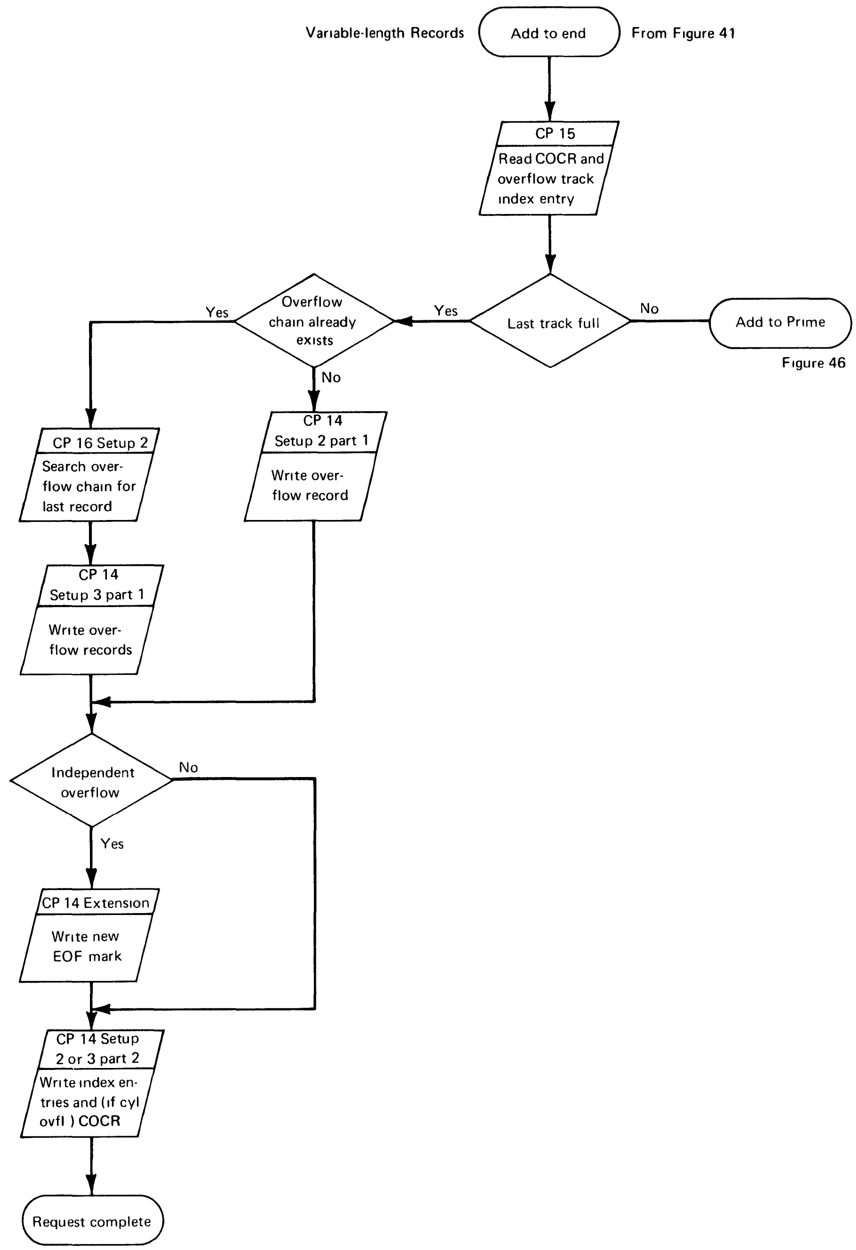


Figure 49. WRITE KN Channel Program Flow — Add to End (Variable- Length Records)

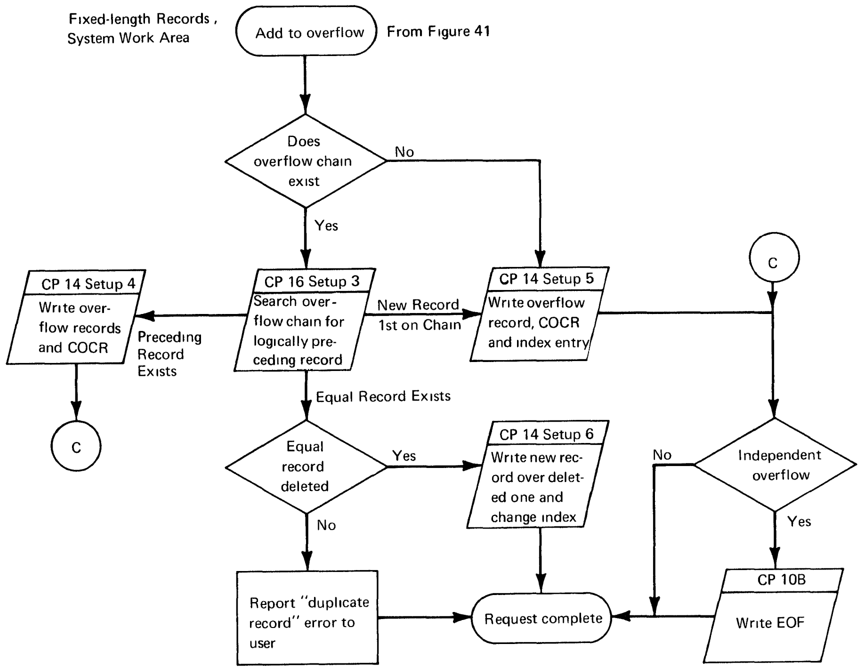


Figure 50. WRITE KN Channel Program Flow — Add to Overflow (Fixed-Length Records, System Work Area)

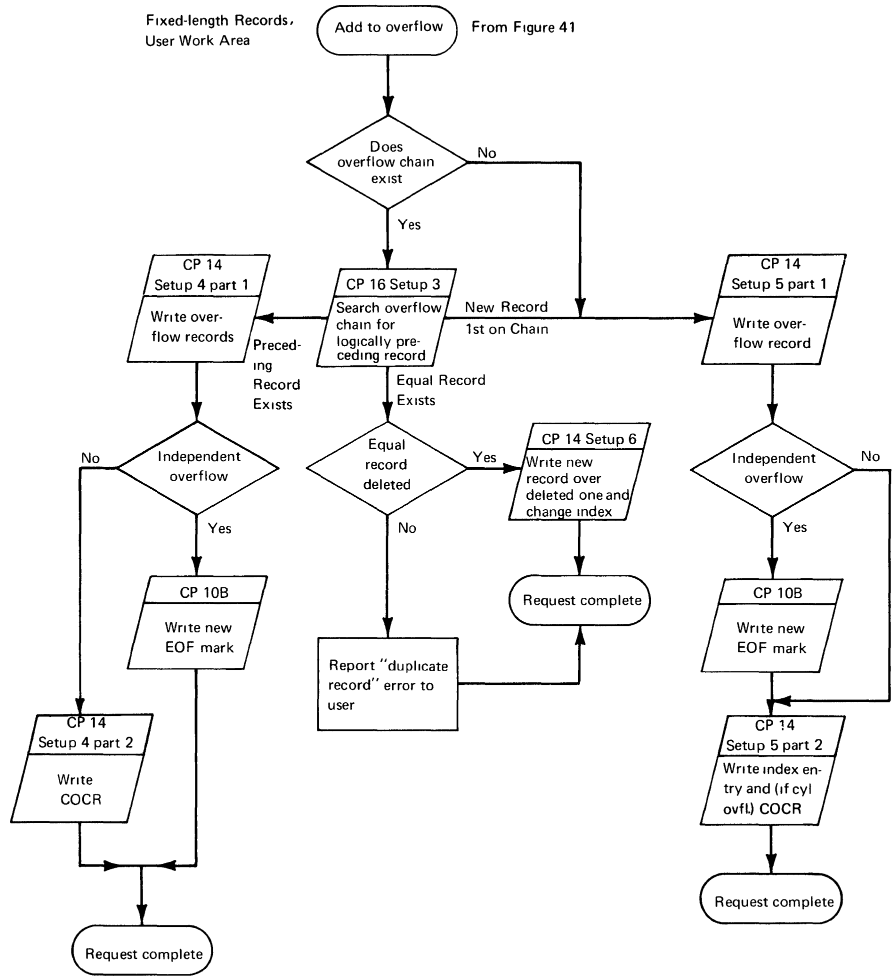


Figure 51. WRITE KN Channel Program Flow — Add to Overflow (Fixed–Length Records, User Work Area)

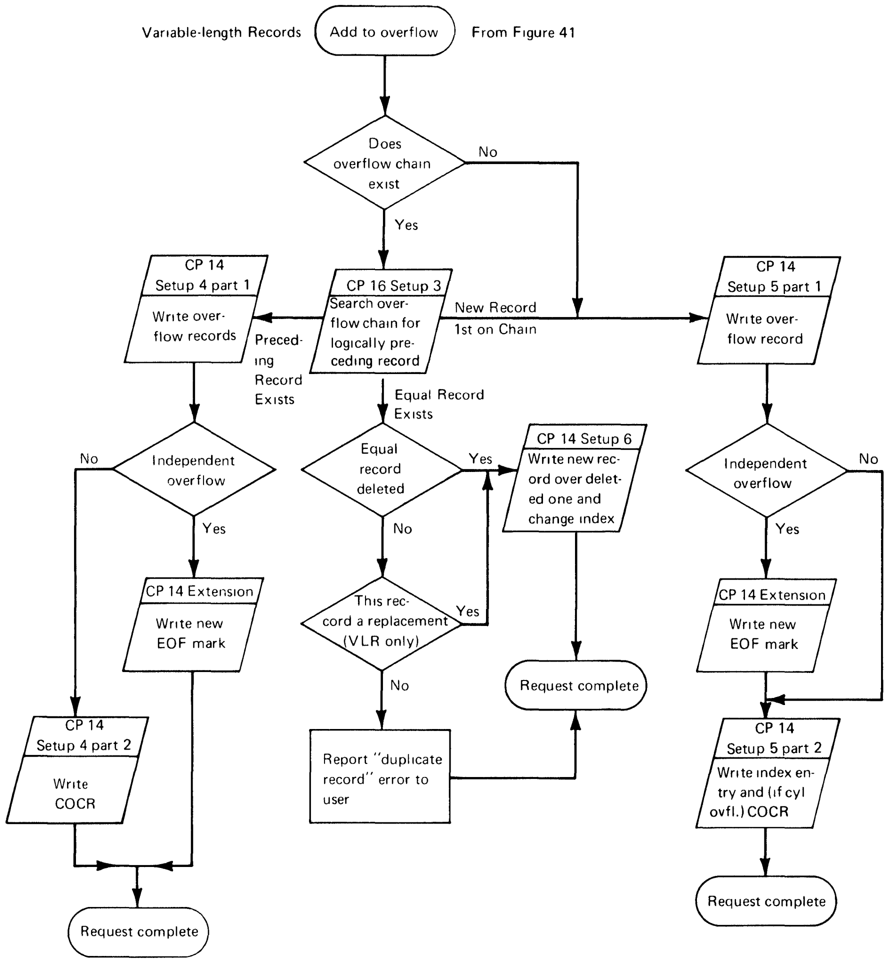


Figure 52. WRITE KN Channel Program Flow — Add to Overflow (Variable–Length Records)

BISAM Control Blocks and Work Areas

Information about the data set and processing requests is carried in control blocks, work areas, and queues. The address relationships of the control blocks to the processing modules, work areas, buffers, channel programs, IOB, and channel program queues are shown in Figures 54 and 55. Figure 53 below shows the elements of a BISAM READ or WRITE request.

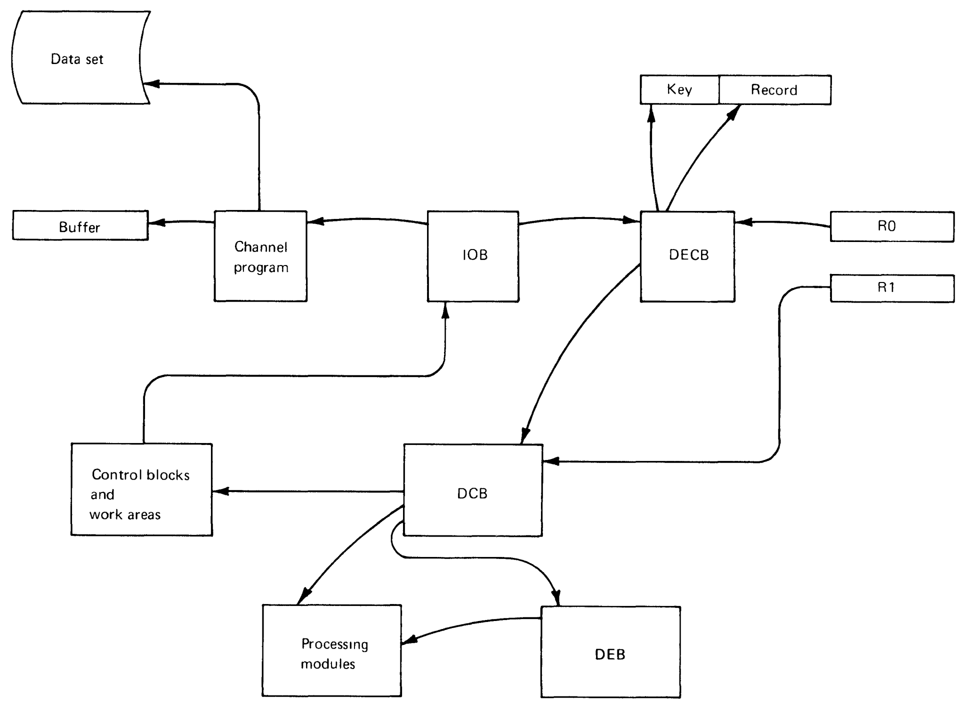


Figure 53. Elements of a BISAM Request

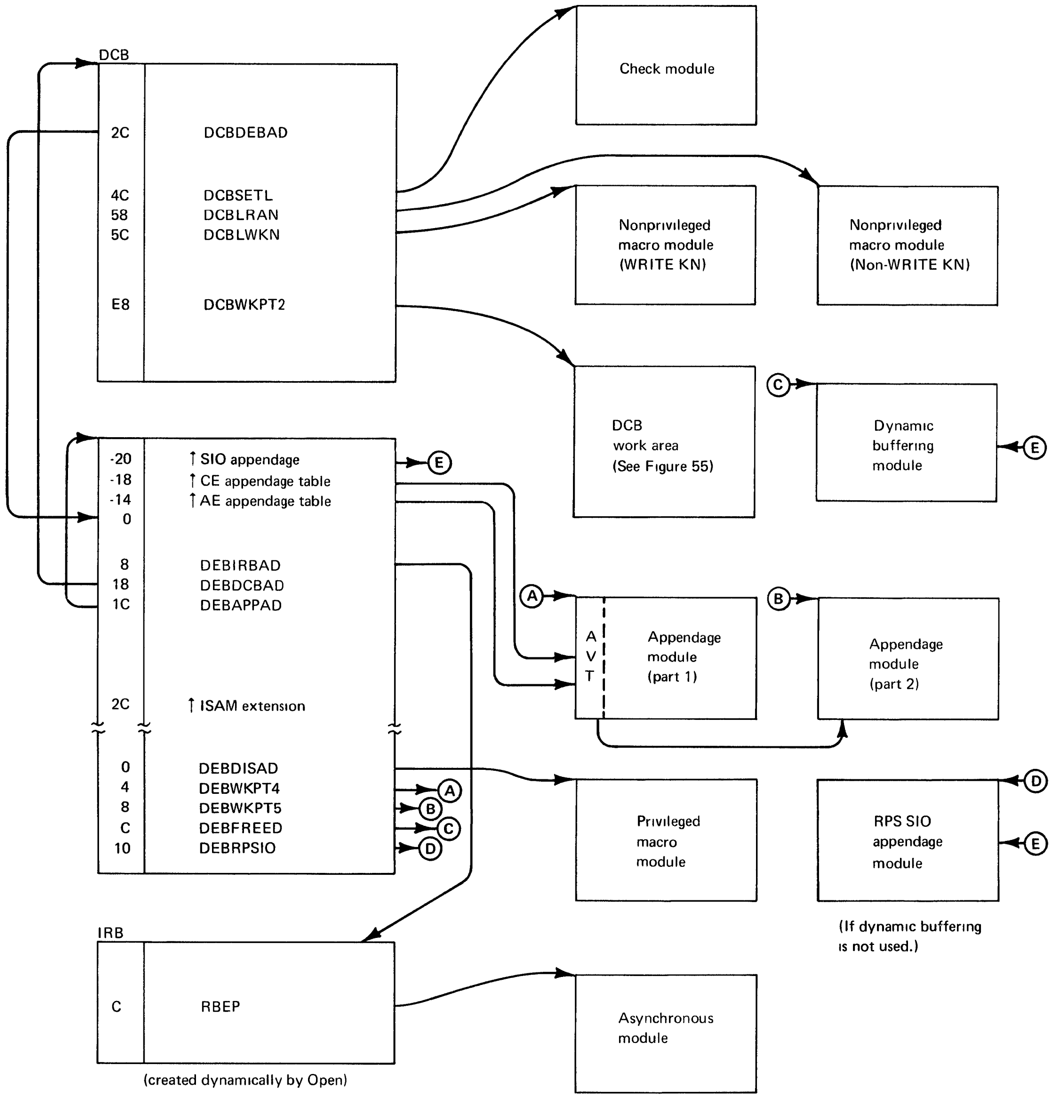


Figure 54. BISAM Control Blocks and Processing Modules

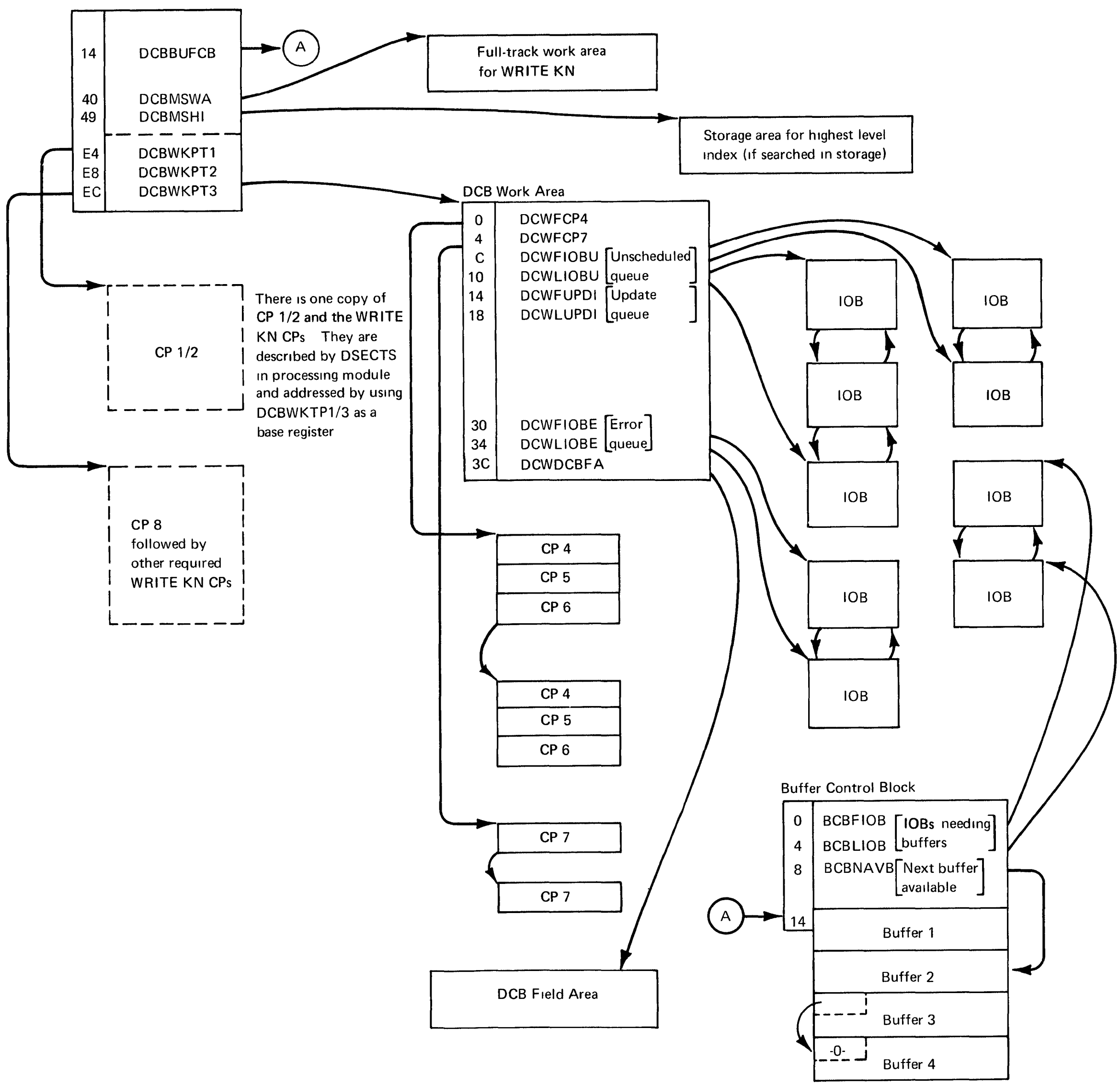


Figure 55. BISAM Work Areas and Queues

BISAM Close Phase

The BISAM close executor (module IGG0202A) is entered from the I/O support Close routine. It terminates outstanding I/O requests and releases main storage obtained for the work area and for channel programs. If dynamic buffering was used, it releases the system–obtained buffer area. If the data set was opened for DISP=SHR, move the DCB fields that may have been changed during processing from the DCB field area (DCBFA) to the DCB. If this is the last DCB open for the data set, free the DCB field area. The BISAM close executor passes control to the ISAM common close executor.

SECTION 3: PROGRAM ORGANIZATION

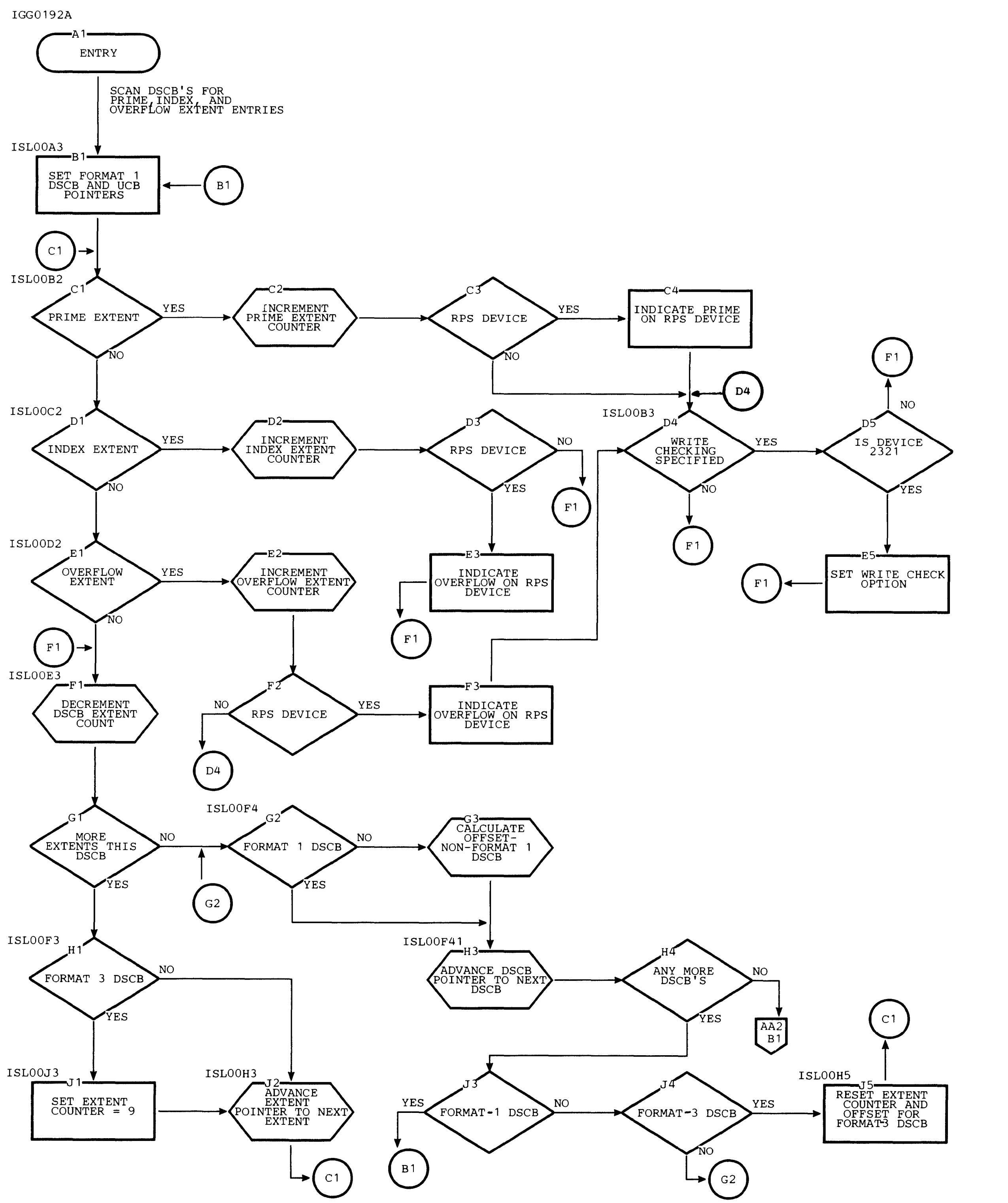


Chart AA1 First Common Open Executor (IGG0192A) (Part 1 of 3)

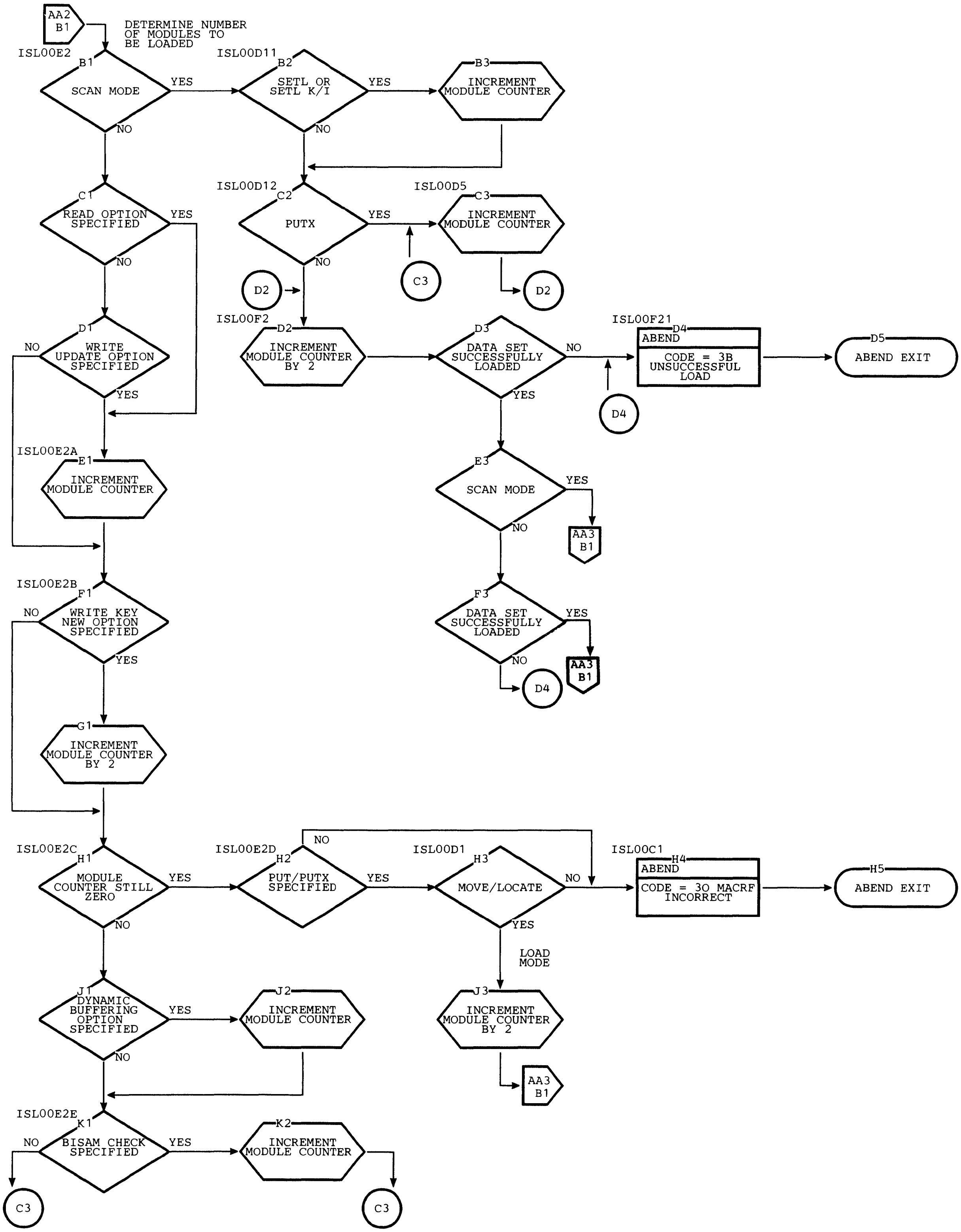


Chart AA2 First Common Open Executor (IGG0192A) (Part 2 of 3)

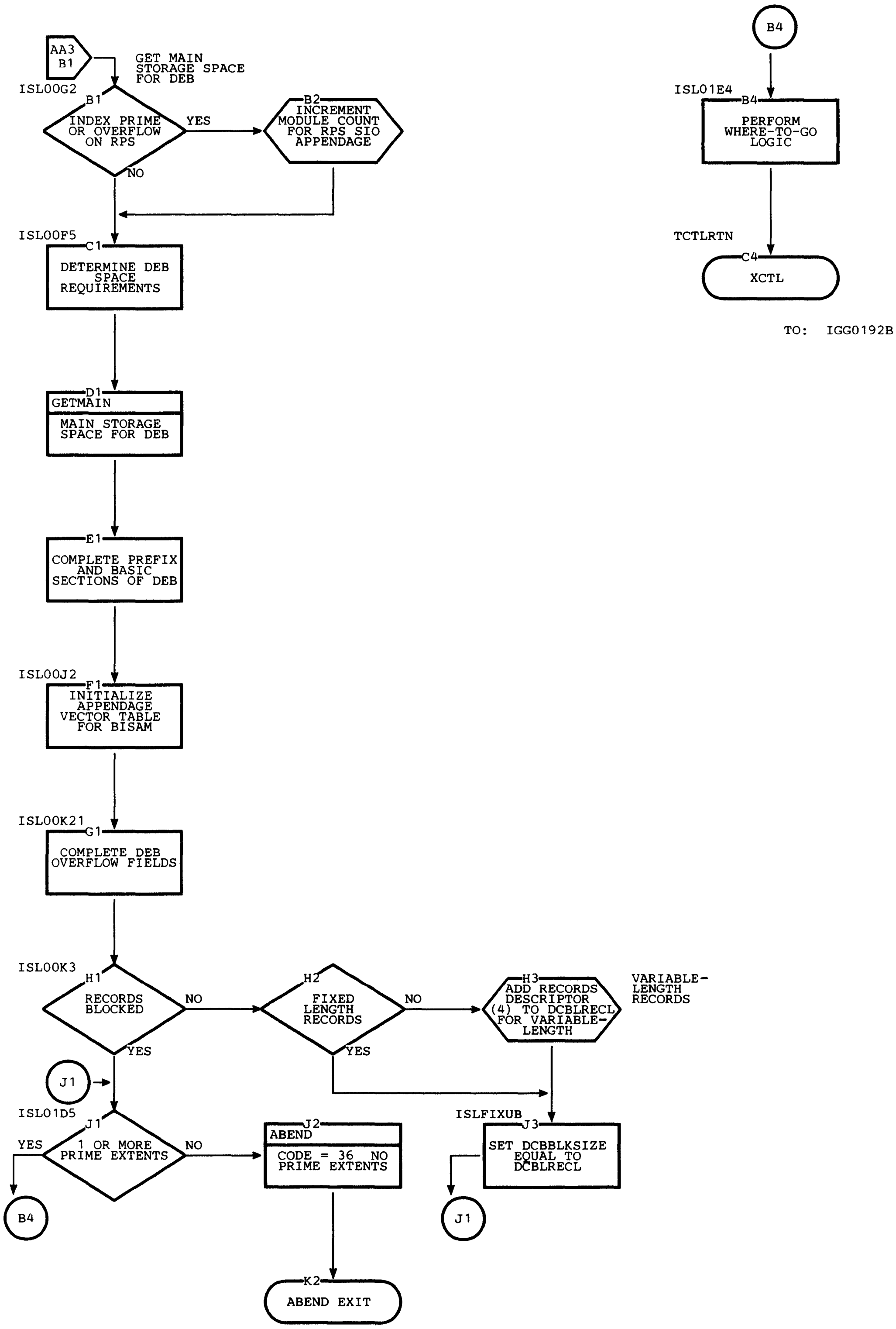


Chart AA3 First Common Open Executor (IGG0192A) (Part 3 of 3)

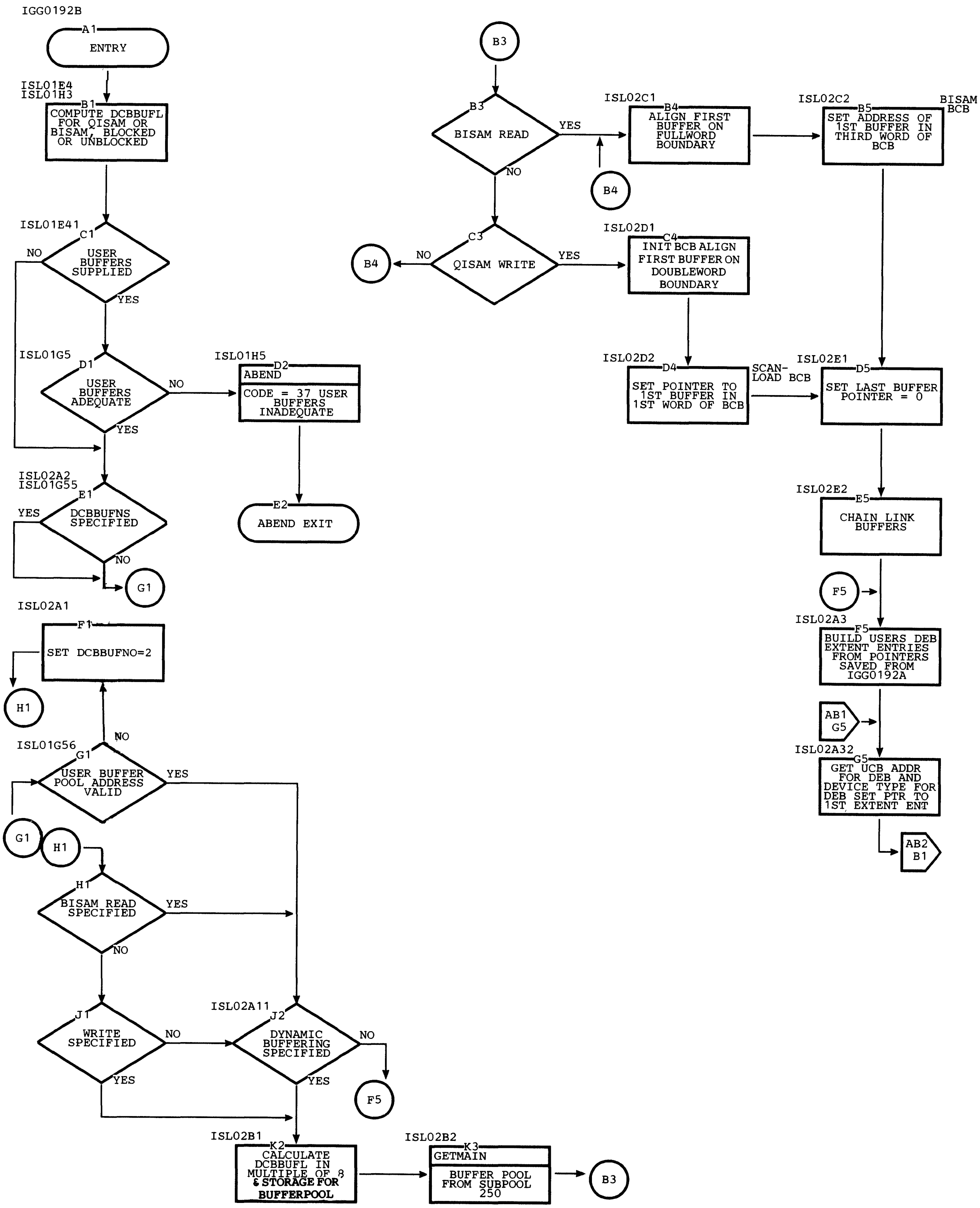


Chart AB1 Second Common Open Executor (IGG0192B) (Part 1 of 2)

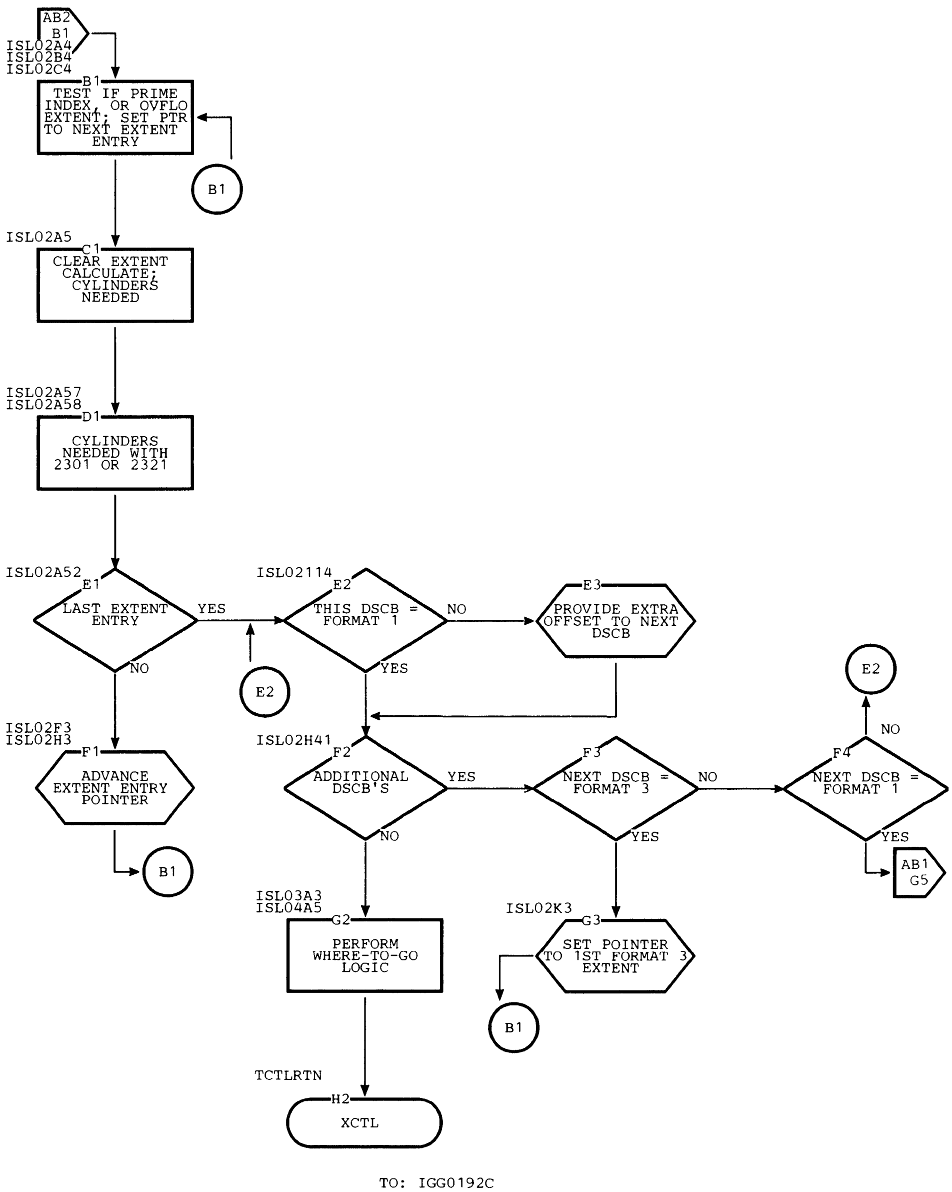


Chart AB2 Second Common Open Executor (IGG0192B) (Part 2 of 2)

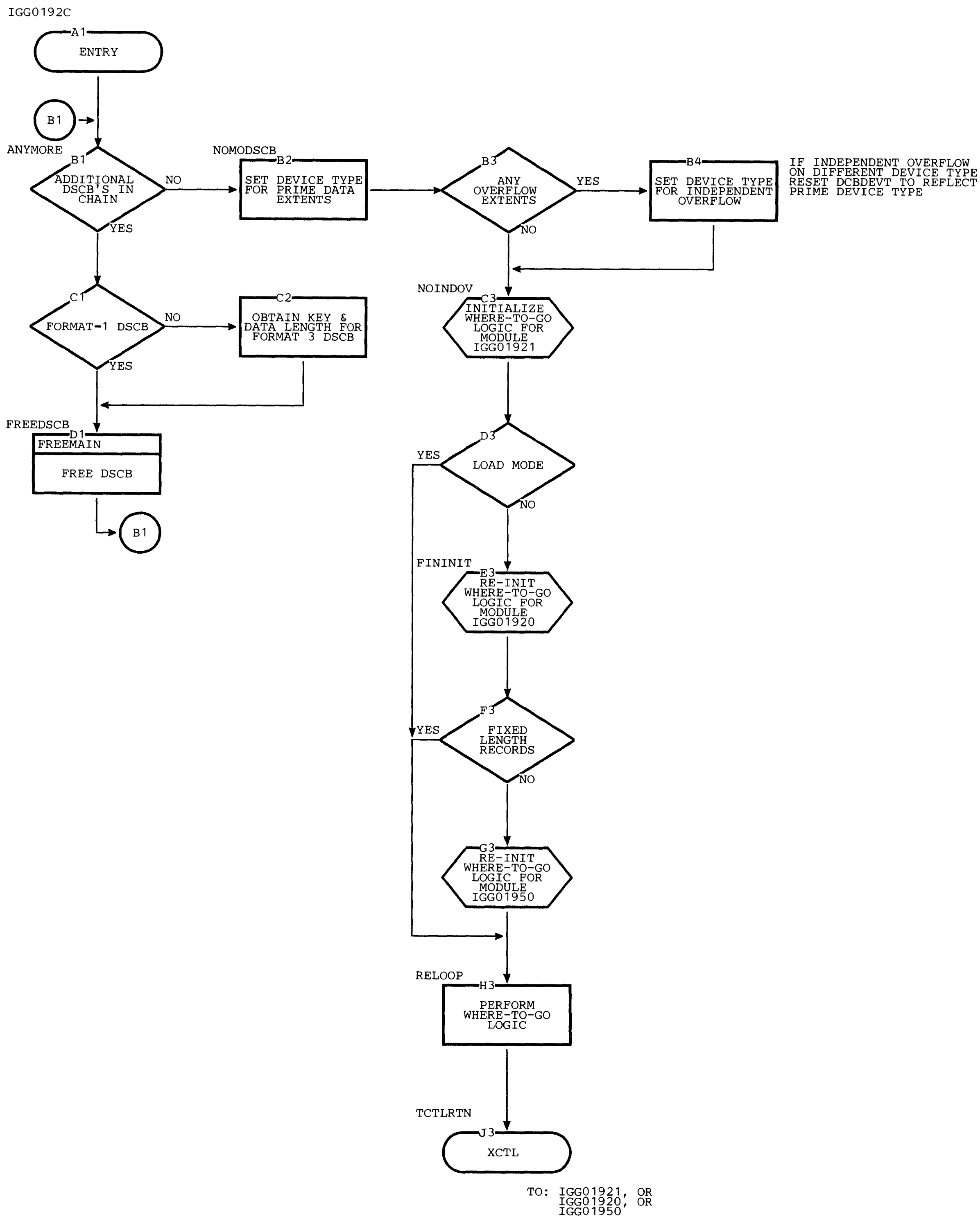


Chart AC1 Third Common Open Executor (IGG0192C)

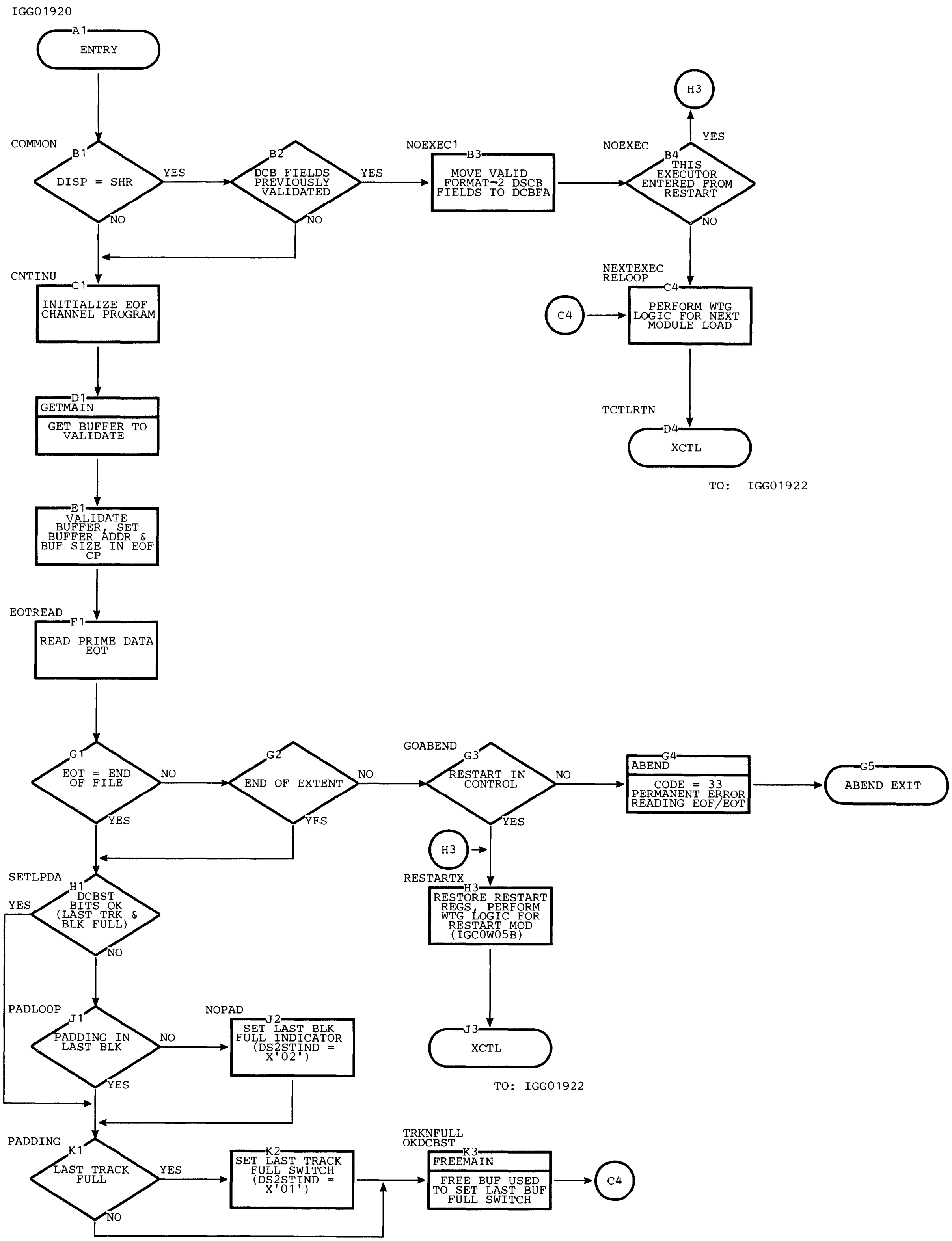


Chart AD1 Fixed-length Validation Open Executors (IGG01920)

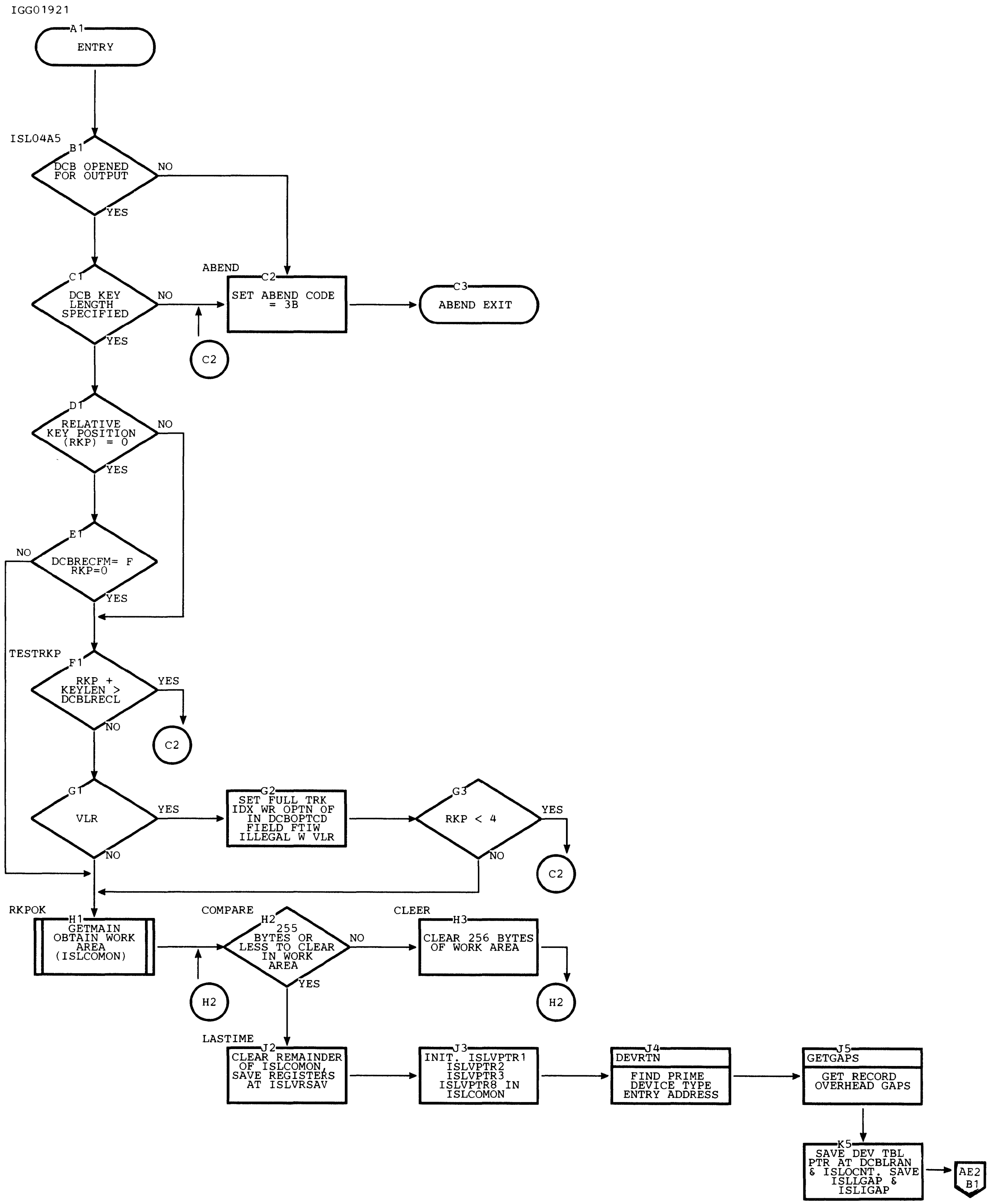


Chart AE1 First Load Mode Open Executor (IGG01921) (Part 1 of 3)

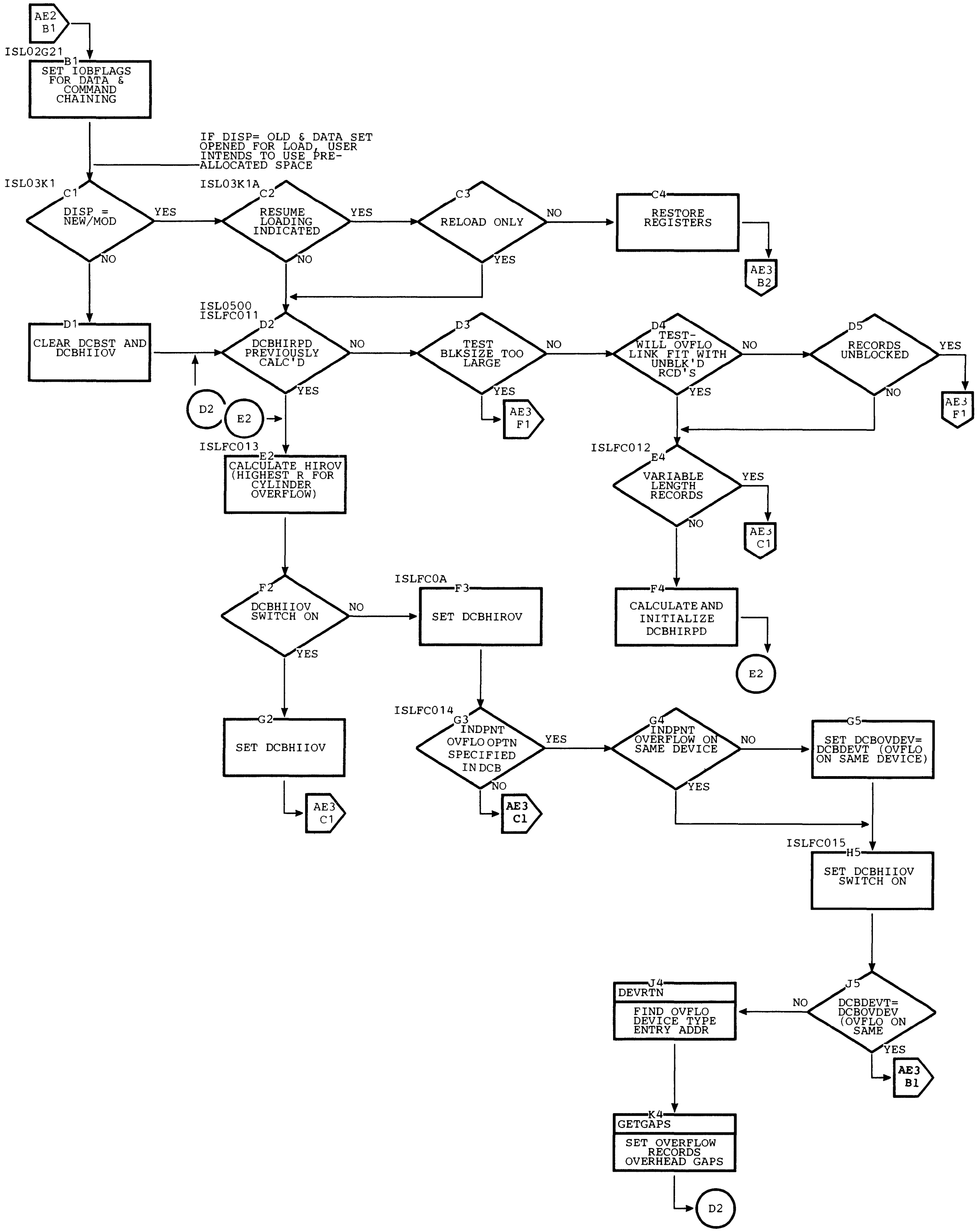


Chart AE2 First Load Mode Open Executor (IGG01921) (Part 2 of 3)

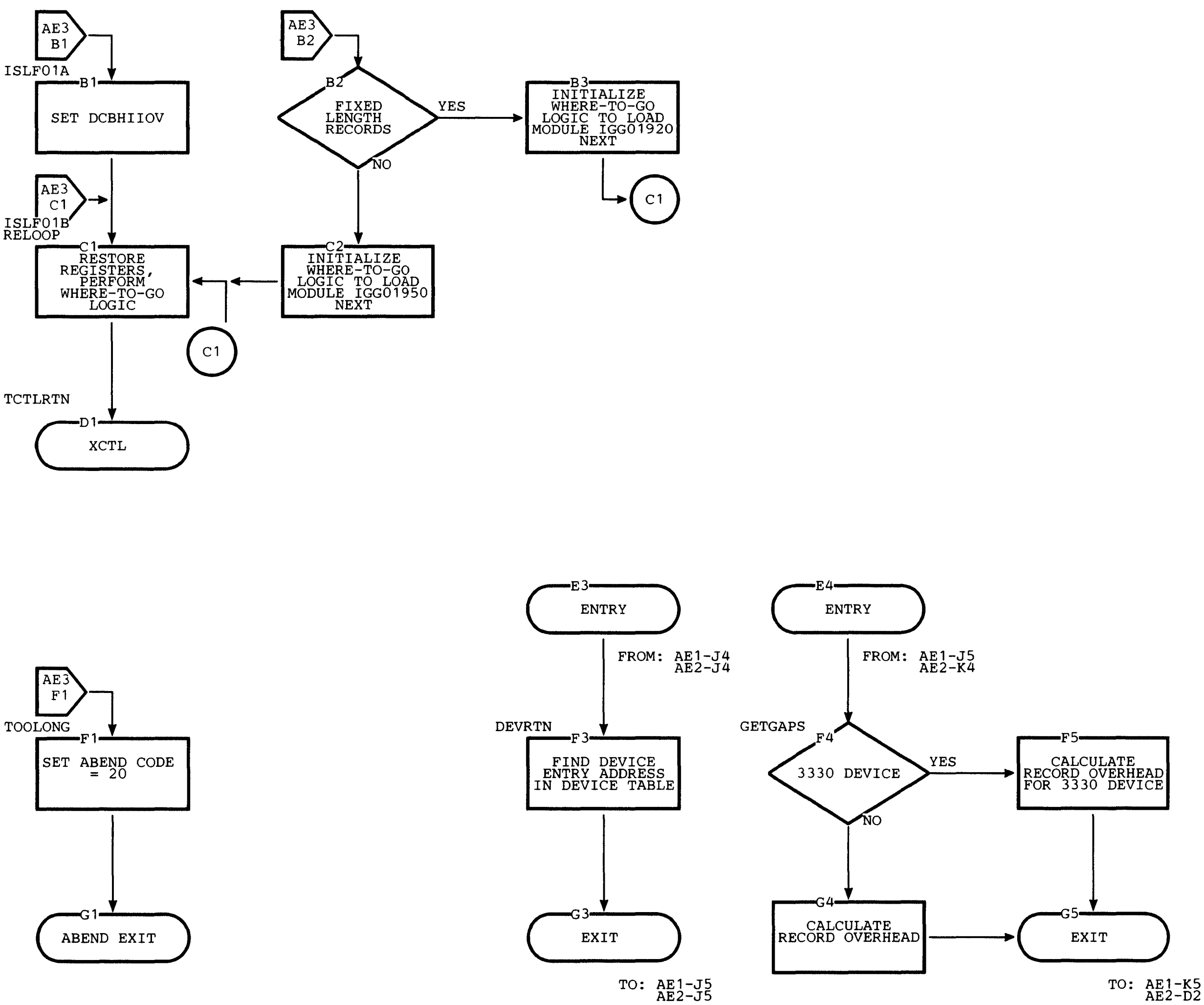


Chart AE3 First Load Mode Open Executor (IGG01921) (Part 3 of 3)

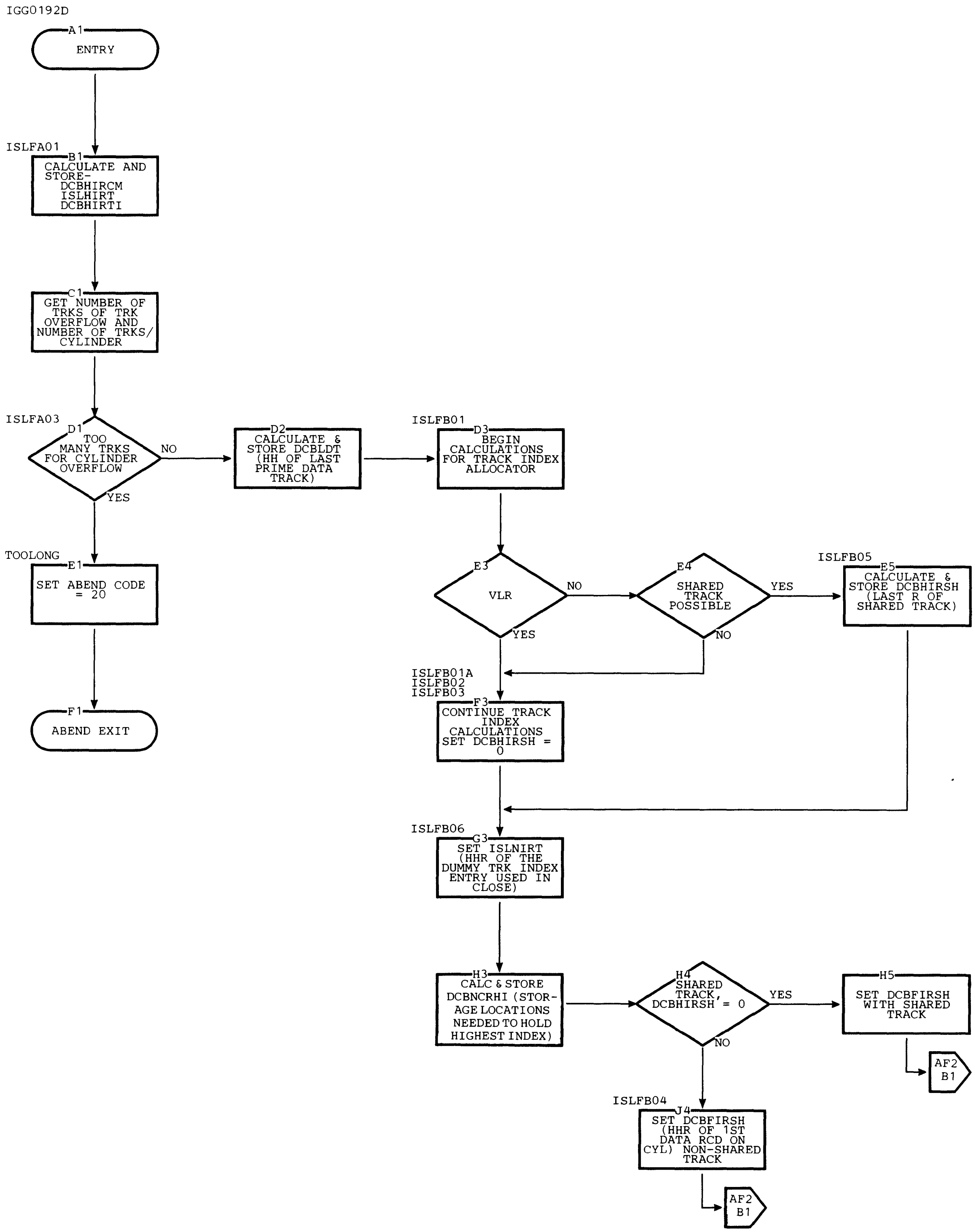


Chart AF1 First Initial Load Mode Open Executor (IGG0192D) (Part 1 of 3)

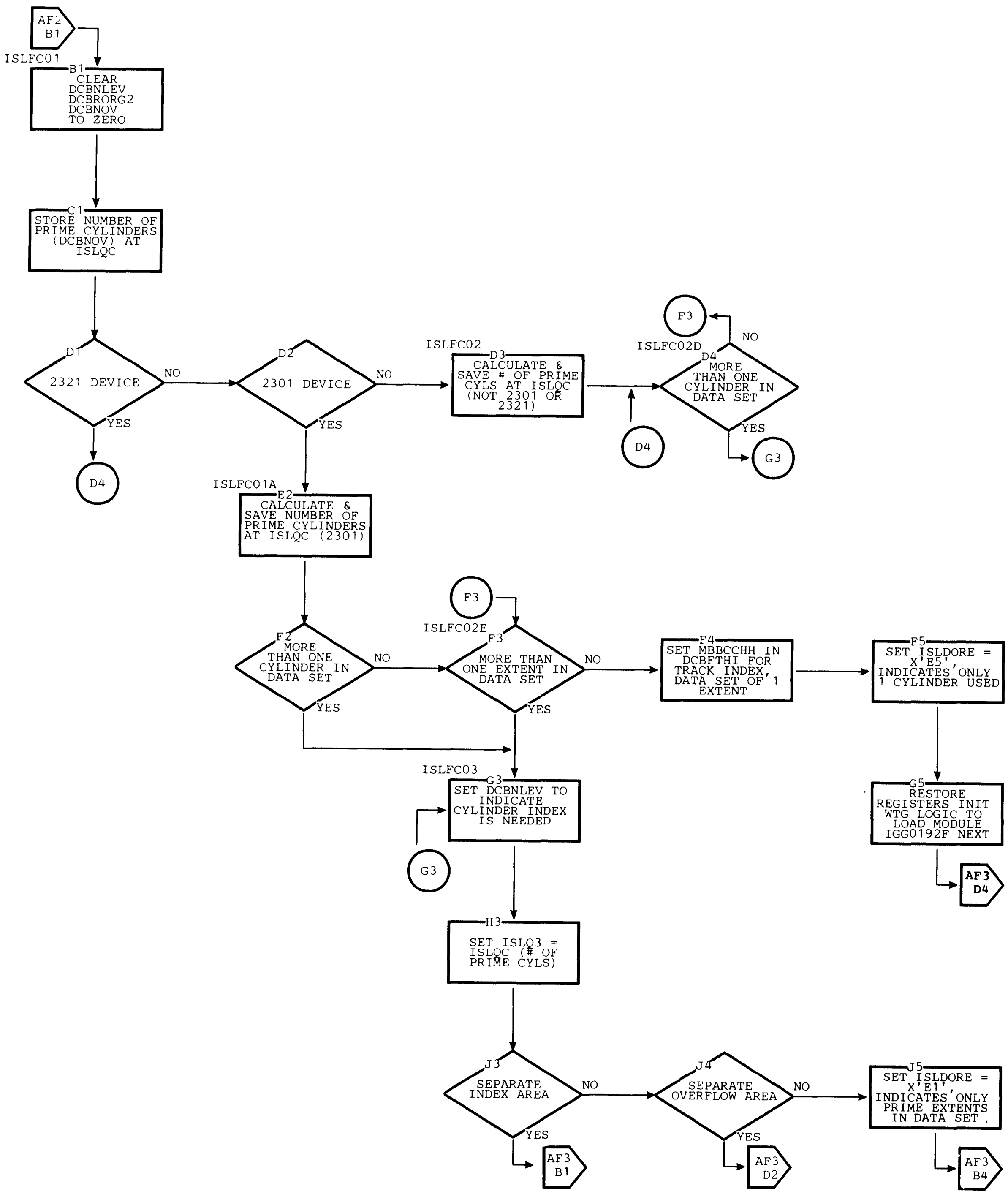


Chart AF2 First Initial Load Mode Open Executor (IGG0192D) (Part 2 of 3)

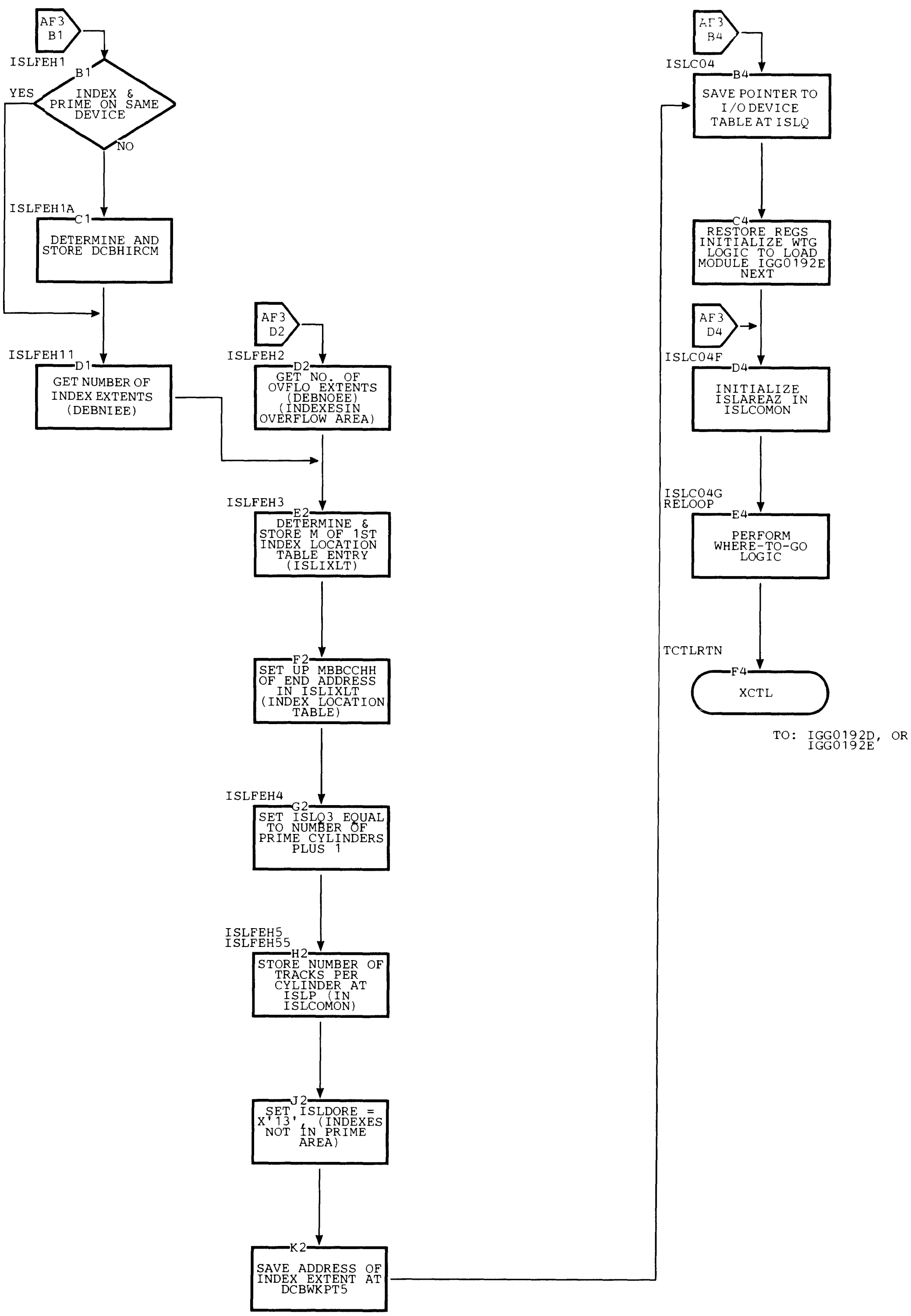


Chart AF3 First Initial Load Mode Open Executor (IGG0192D) (Part 3 of 3)

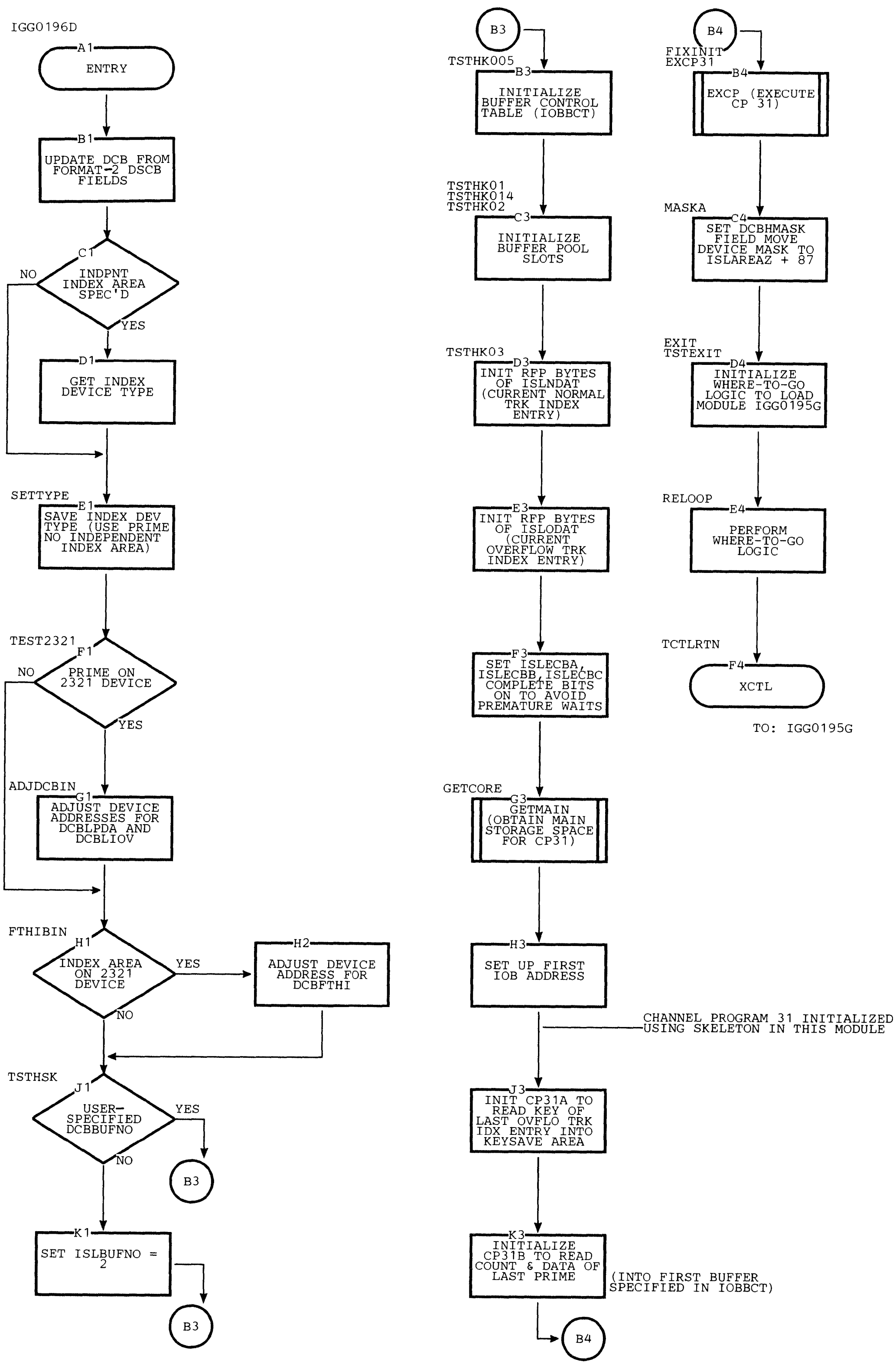


Chart AG1 First Resume Load Open Executor (IGG0196D)

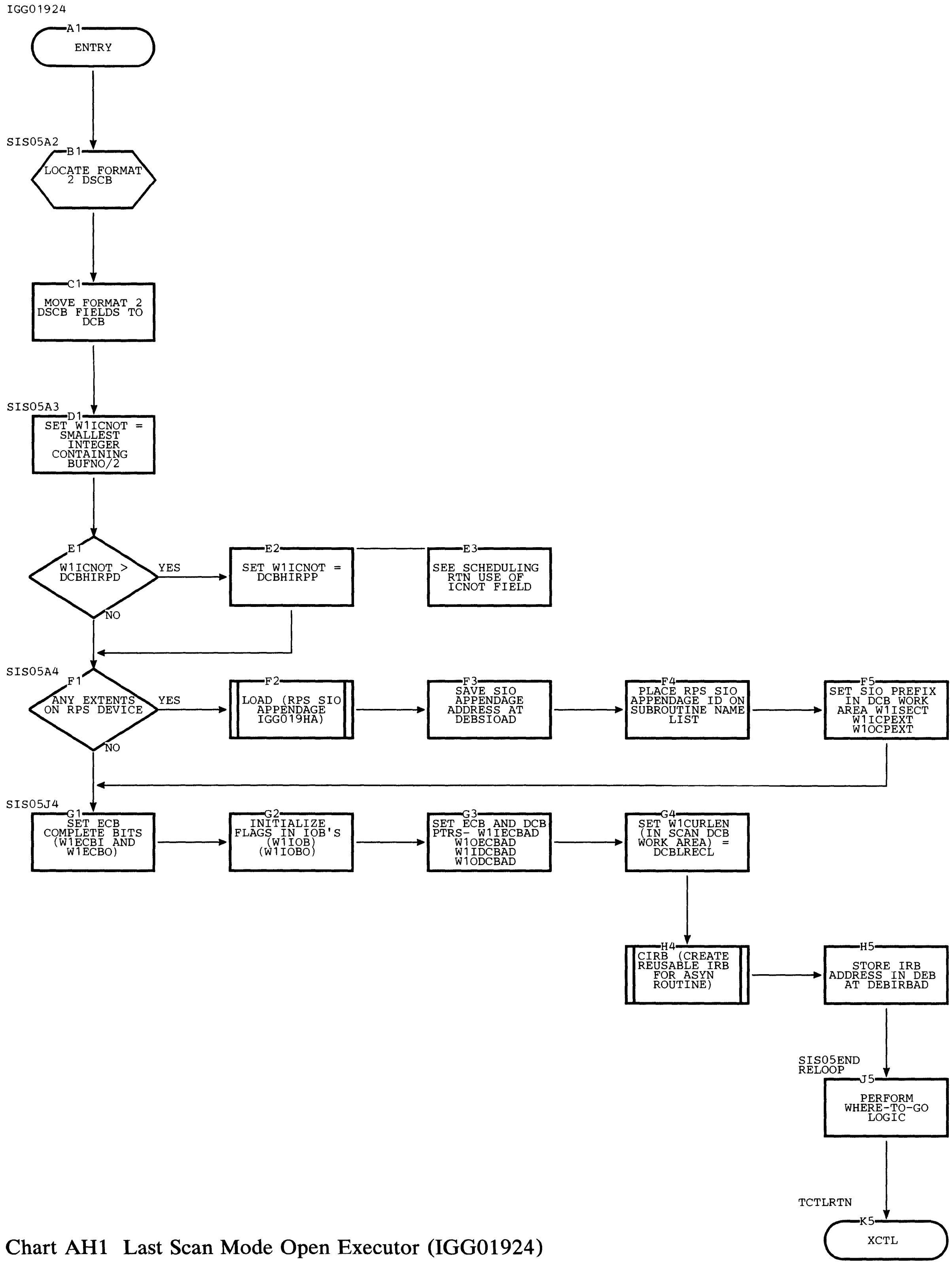


Chart AH1 Last Scan Mode Open Executor (IGG01924)

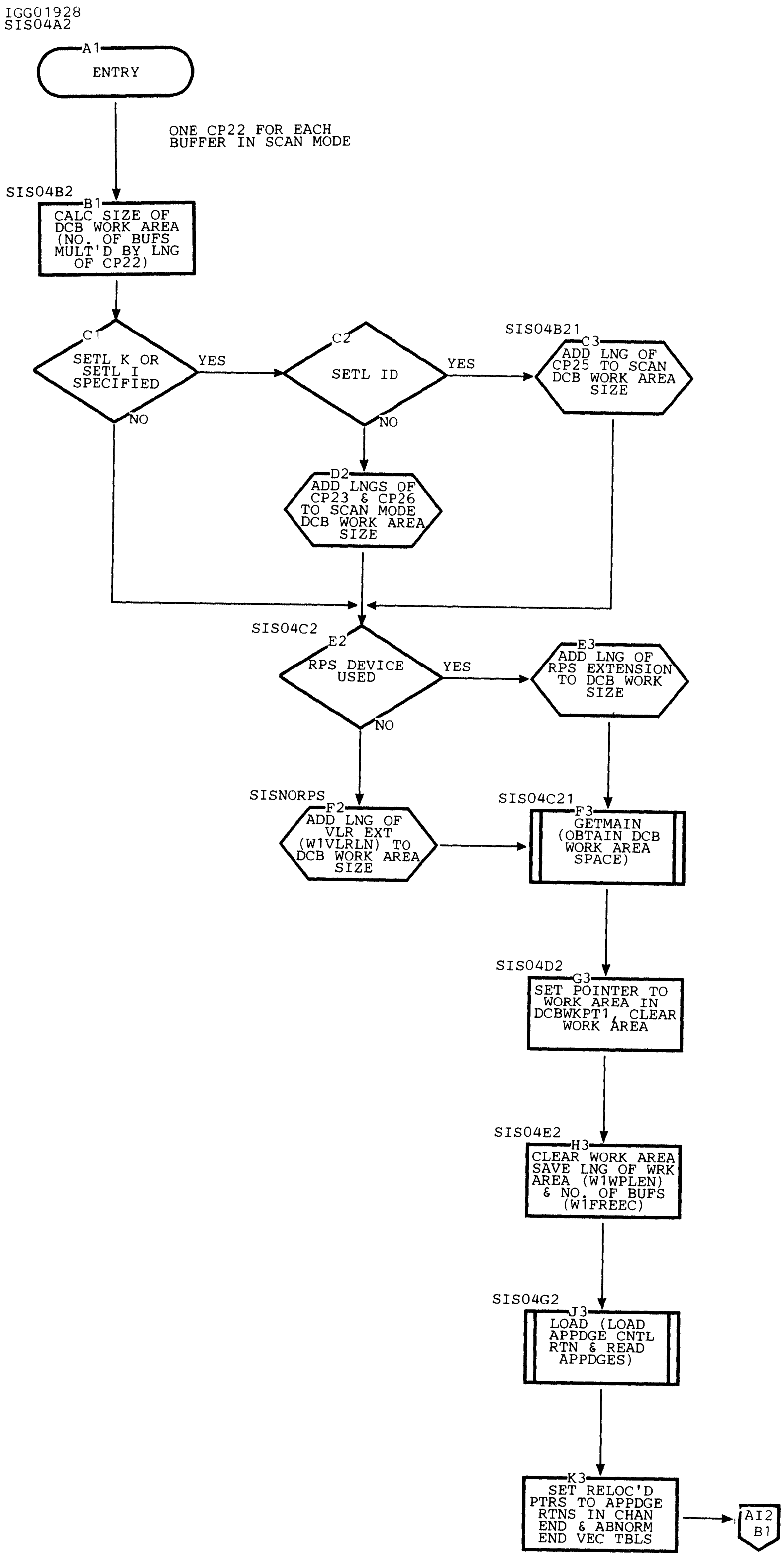


Chart AI1 First Scan Mode Open Executor Module (IGG01928) (Part 1 of 4)

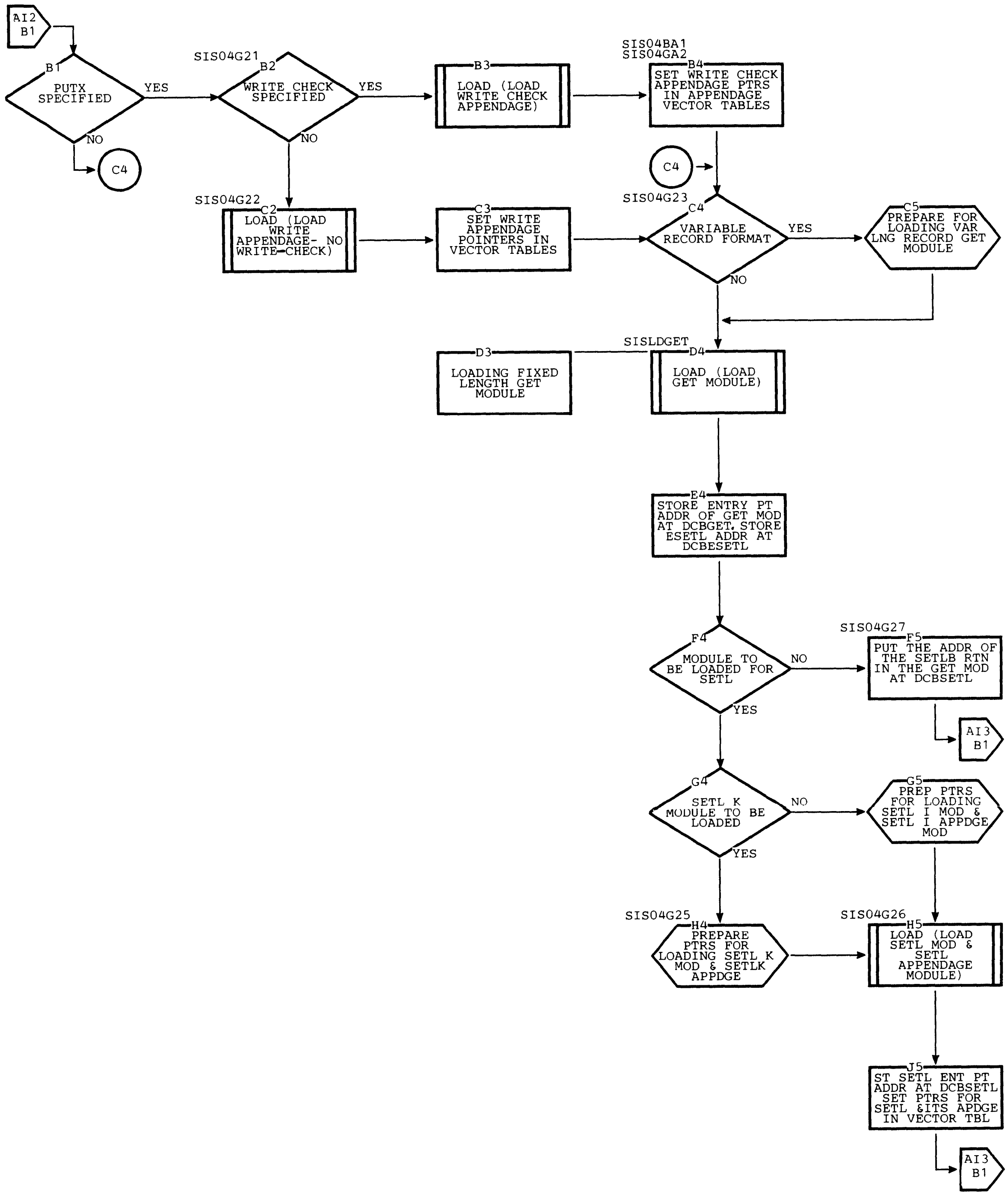


Chart AI2 First Scan Mode Open Executor Module (IGG01928) (Part 2 of 4)

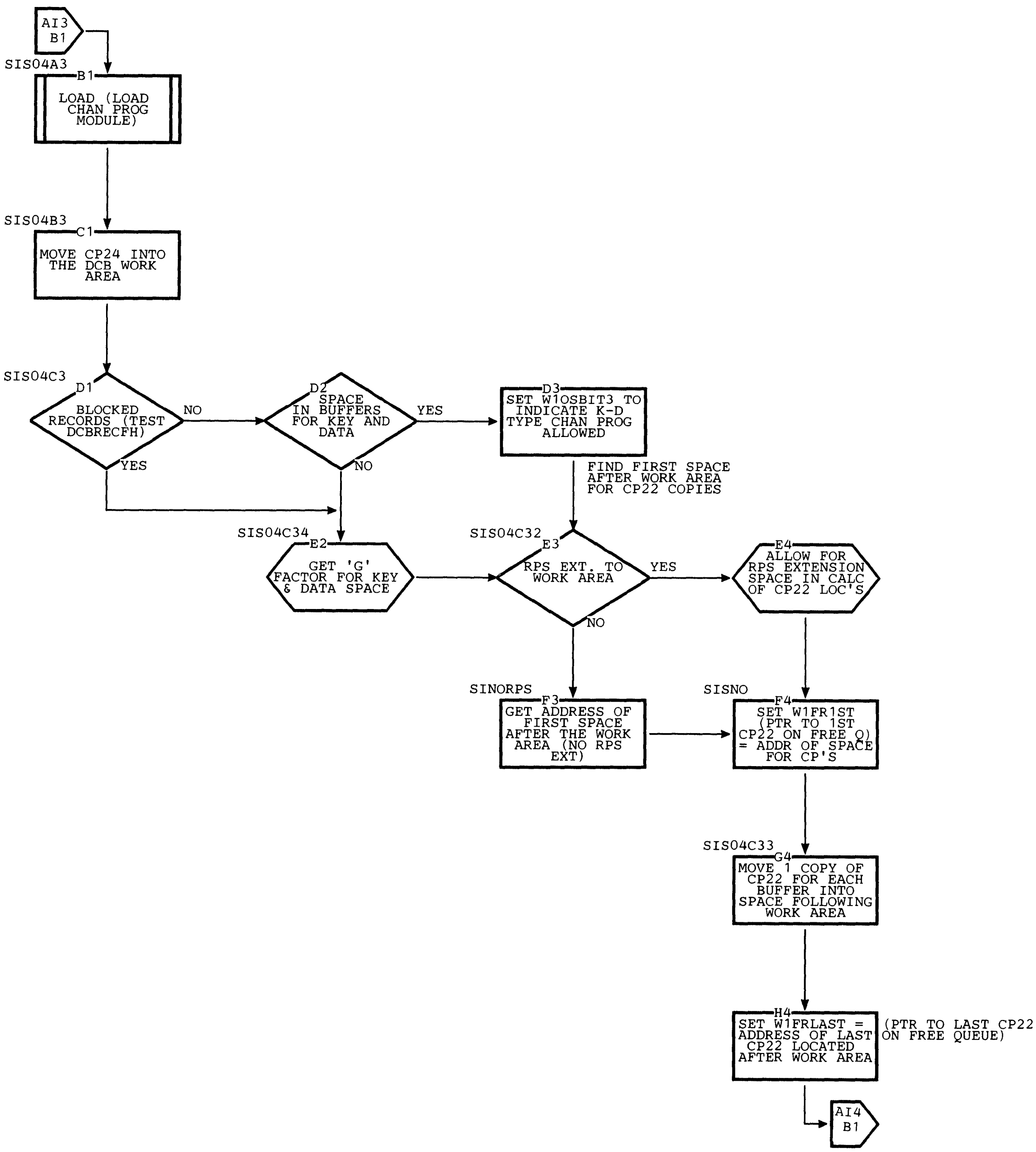


Chart AI3 First Scan Mode Open Executor Module (IGG01928) (Part 3 of 4)

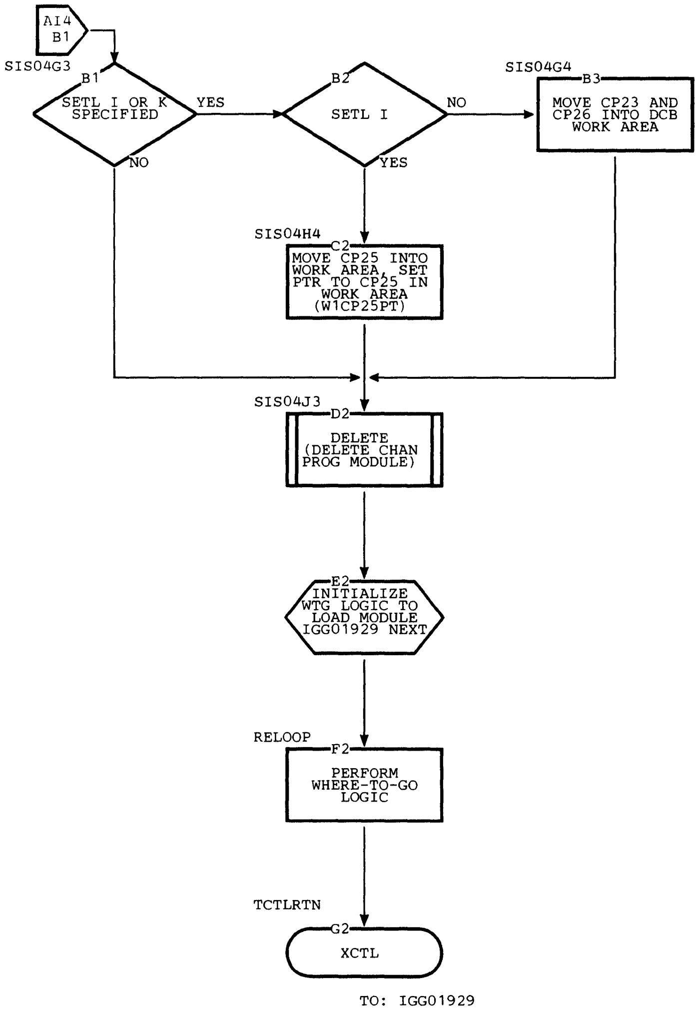


Chart AI4 First Scan Mode Open Executor Module (IGG01928) (Part 4 of 4)

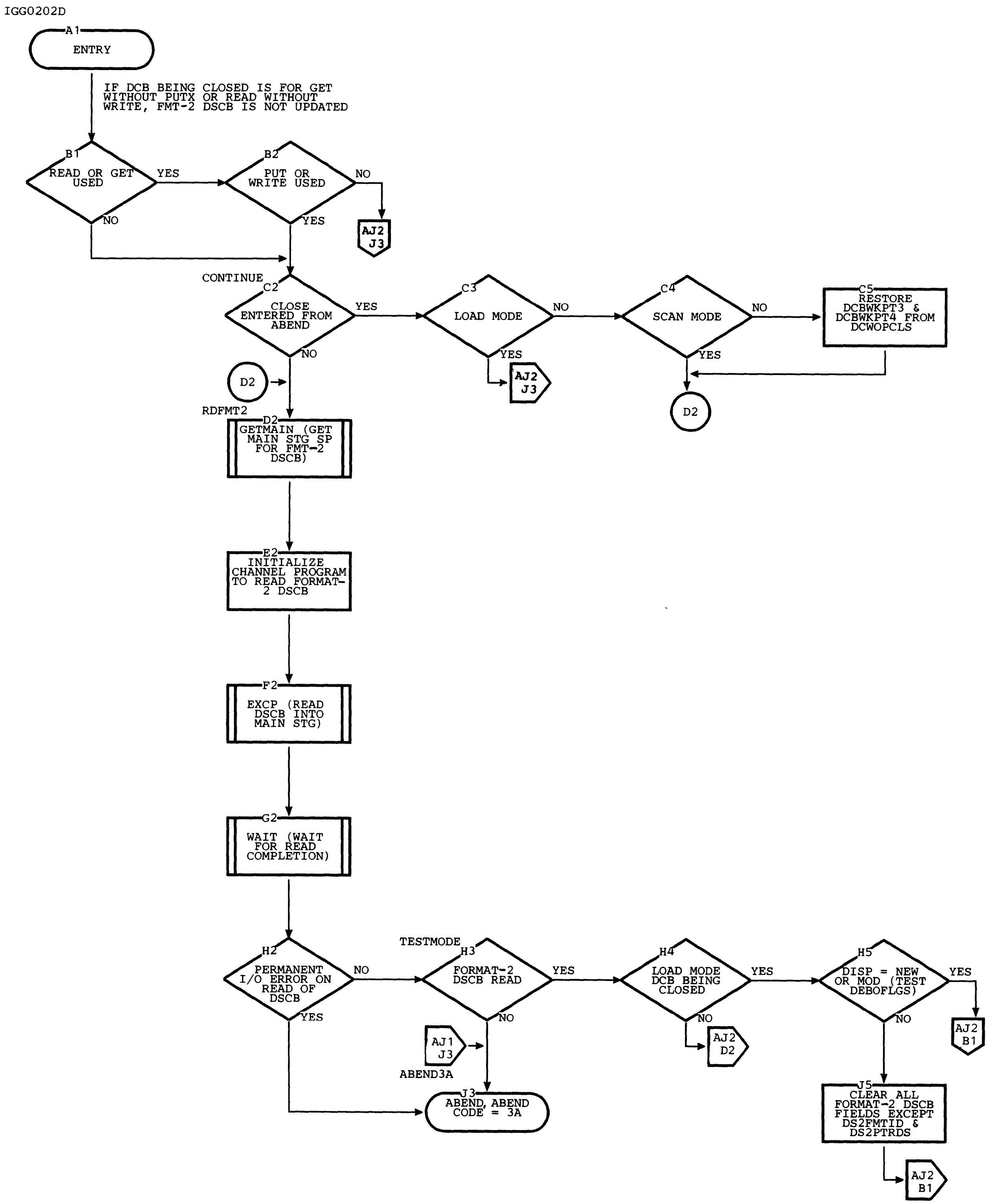


Chart AJ1 ISAM Common Close Executor Module (IGG0202D) (Part 1 of 2)

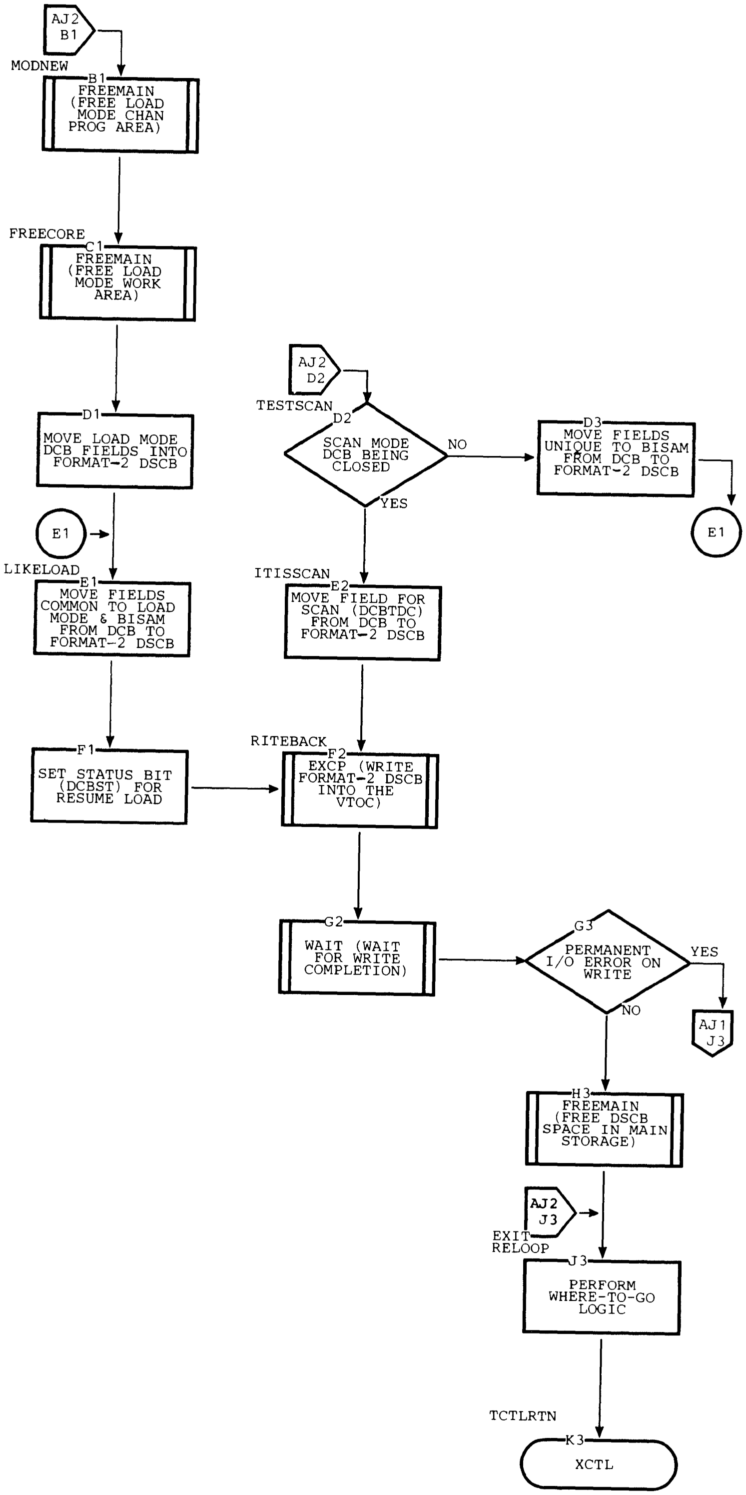


Chart AJ2 ISAM Common Close Executor Module (IGG0202D) (Part 2 of 2)

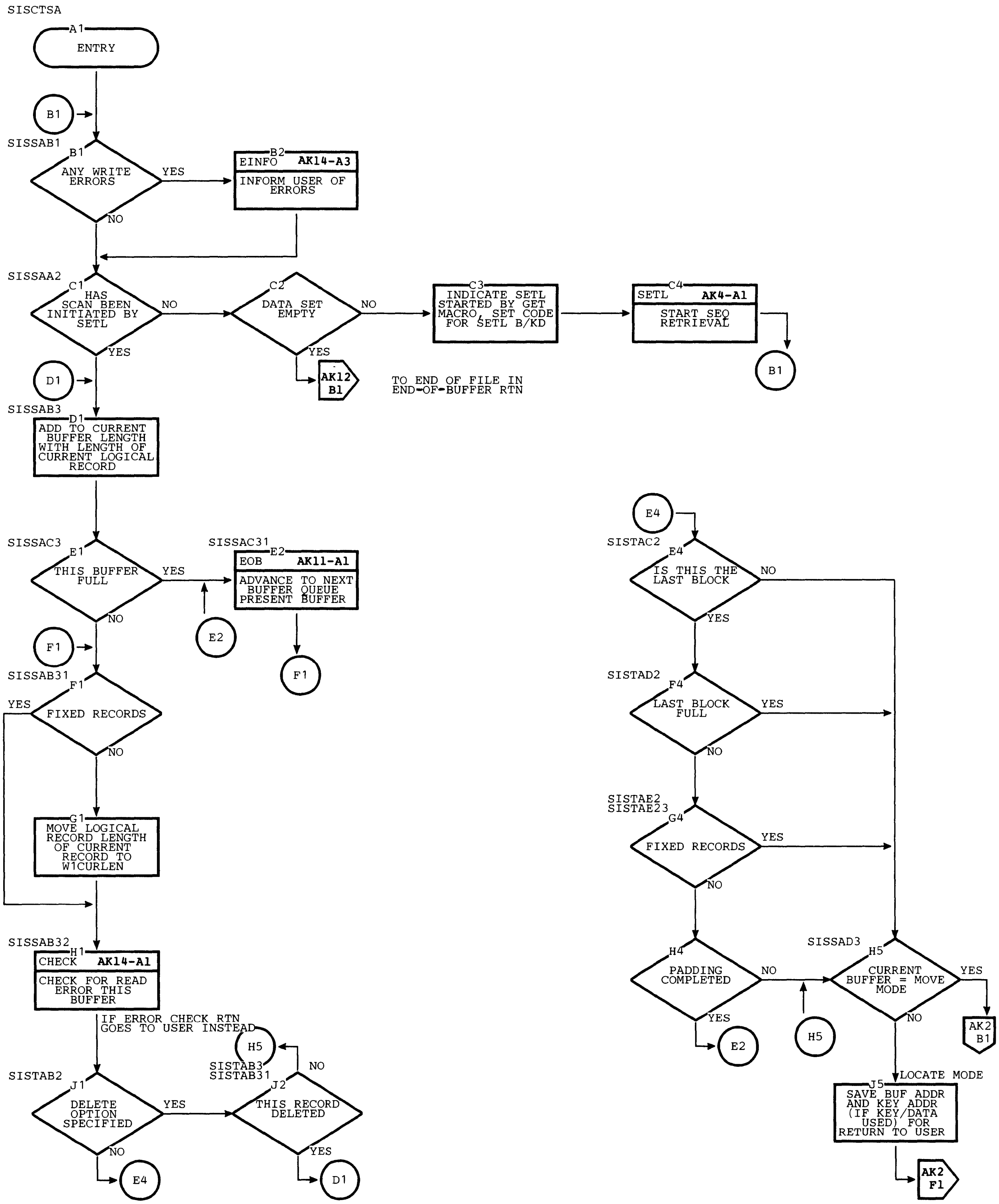


Chart AK1 QISAM Scan Processing Module (IGG019HB) GET Macro Routine (Part 1 of 14)

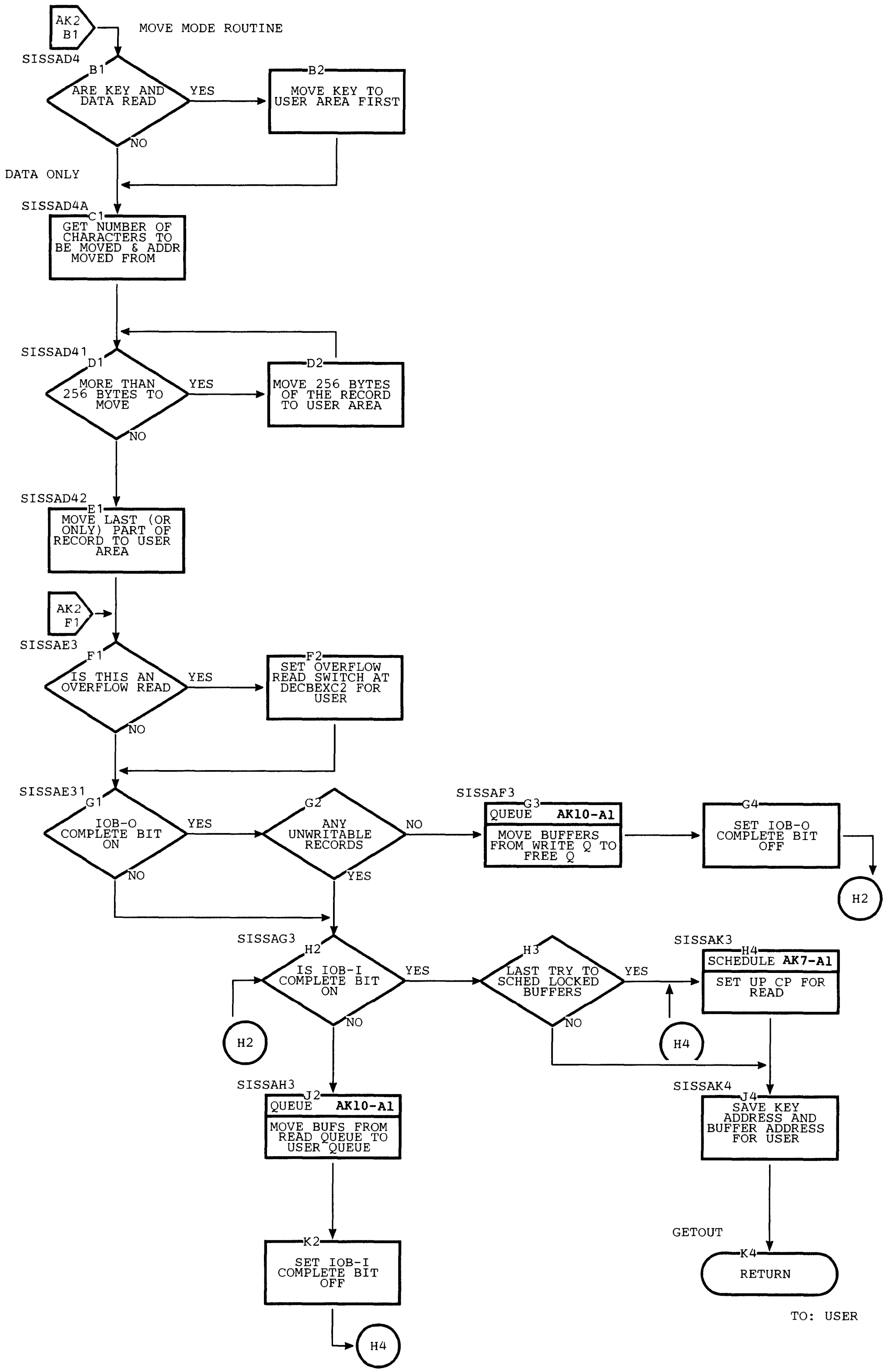


Chart AK2 QISAM Scan Processing Module (IGG029HB) GET Macro Routine (Part 2 of 14)

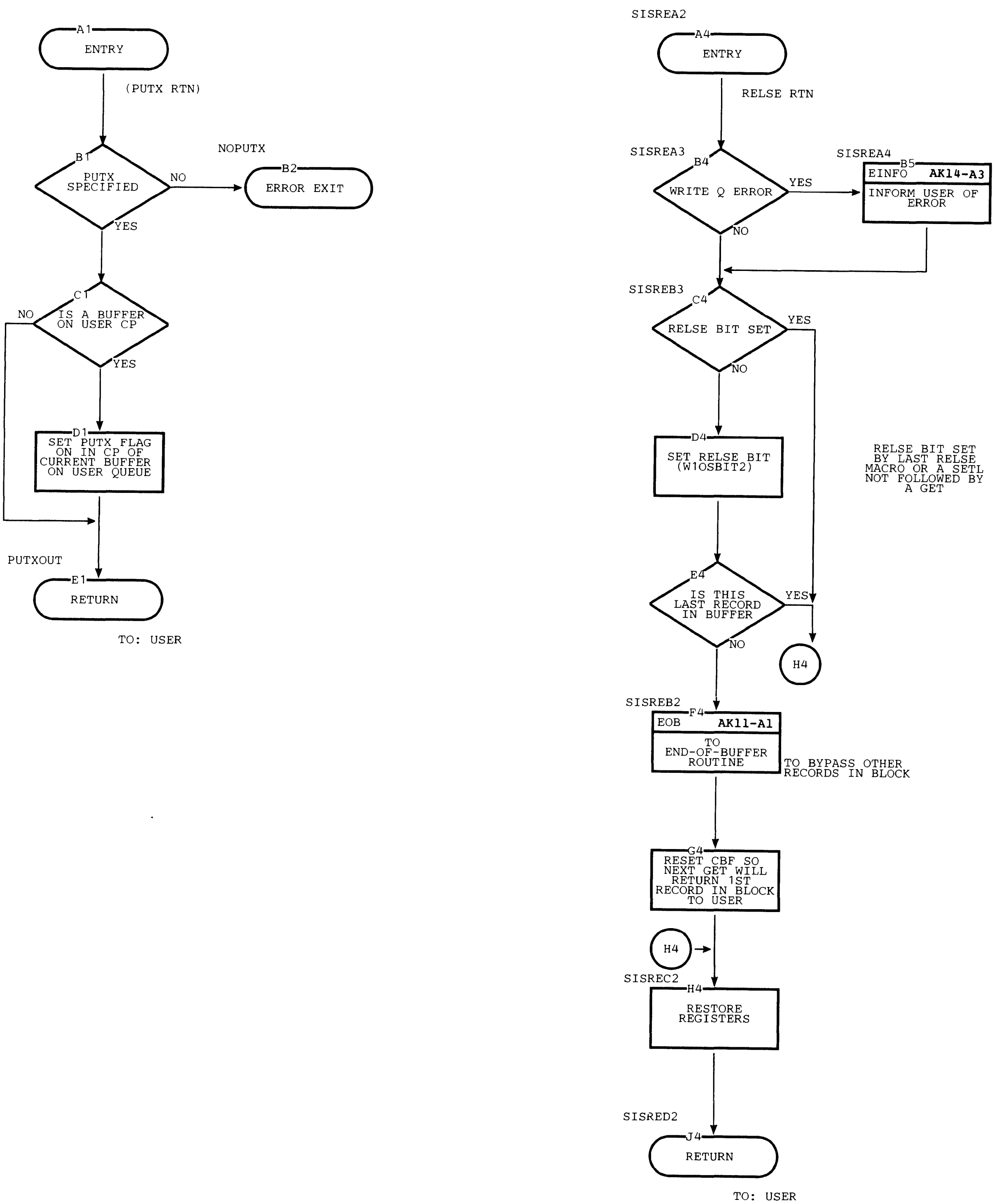


Chart AK3 QISAM Scan Processing Module (IGG019HB) PUTX Macro Routine, RELSE Macro Routine (Part 3 of 14)

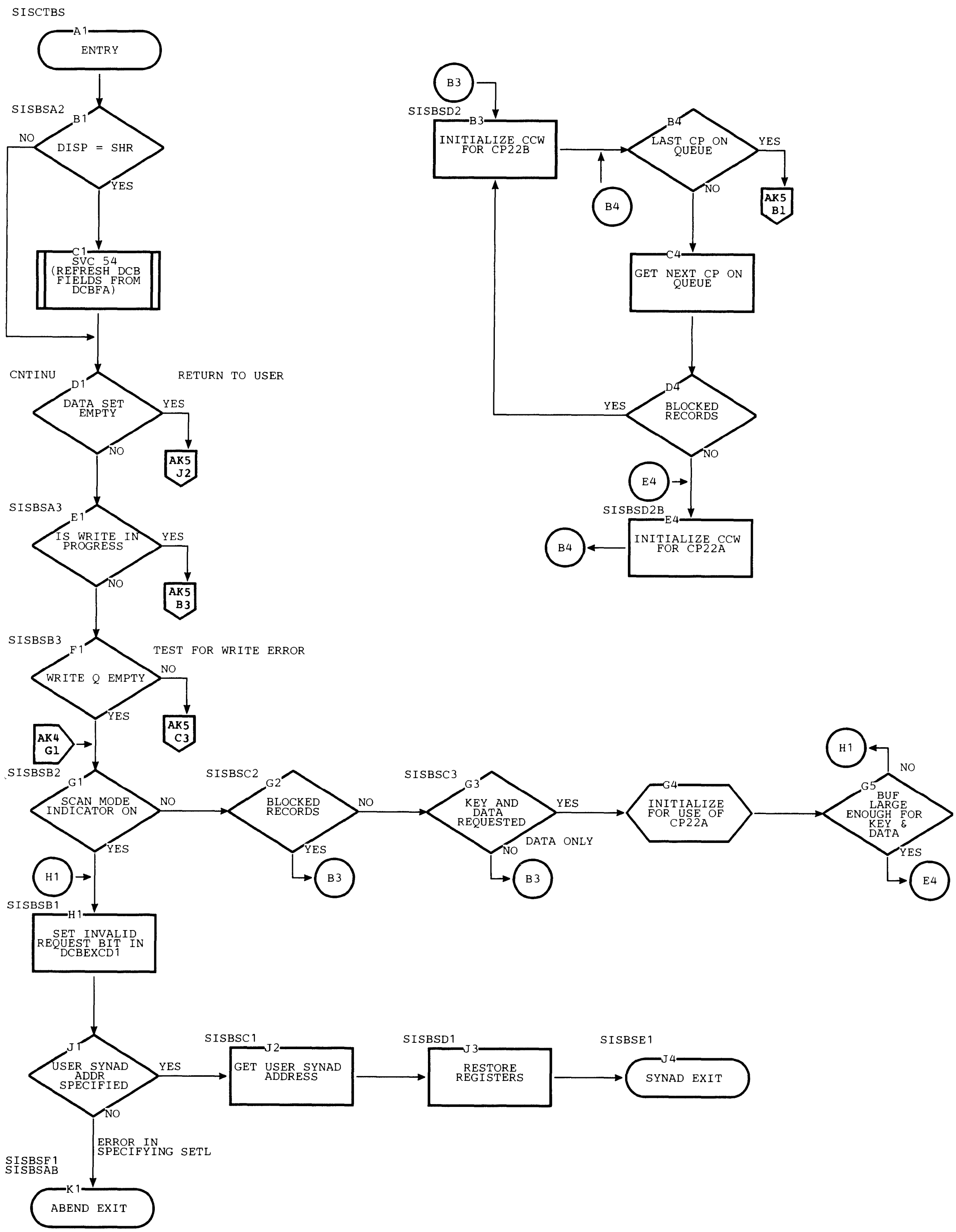


Chart AK4 QISAM Scan Processing Module (IGG019HB) SETL B Macro Routine (Part 4 of 14)

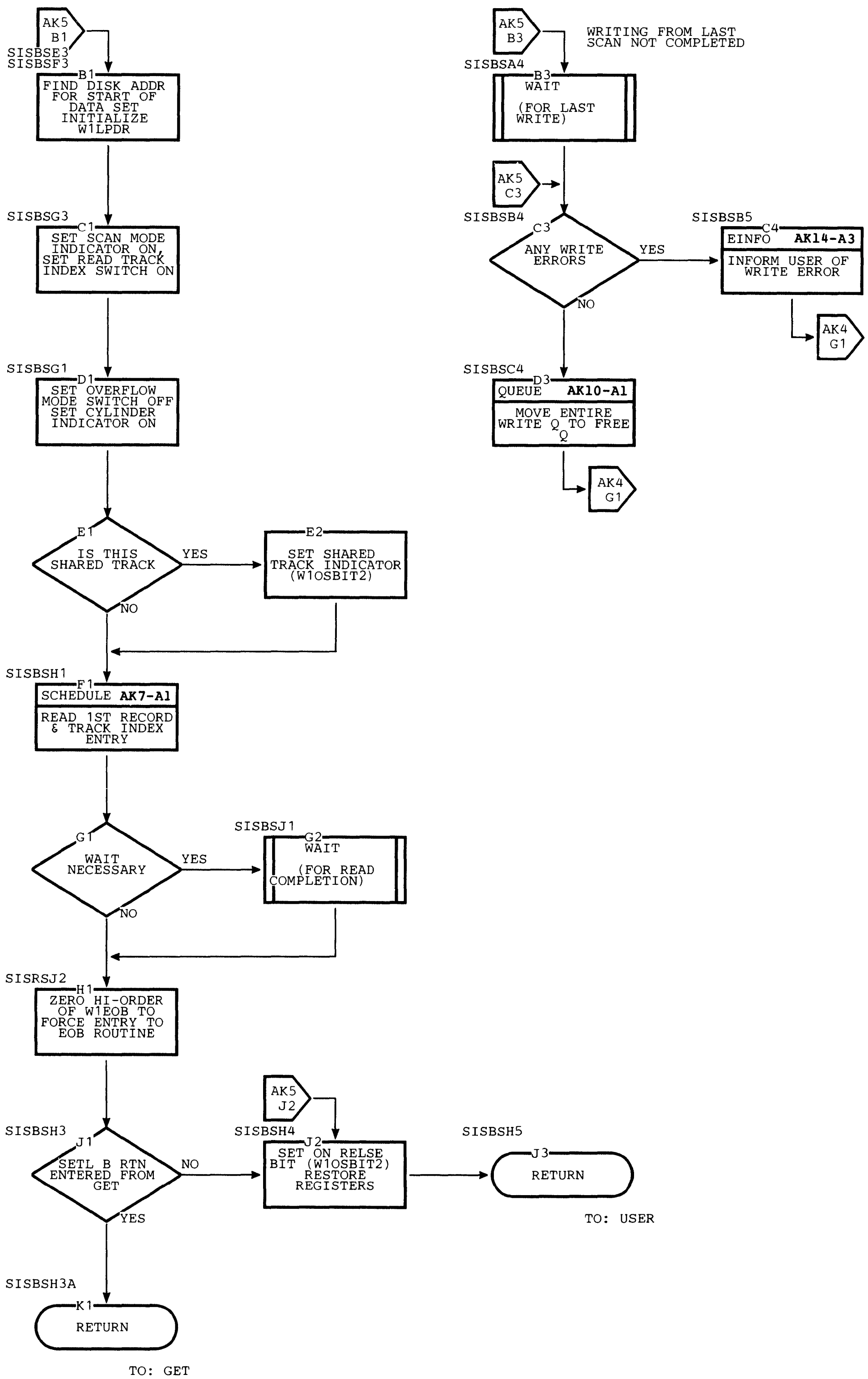


Chart AK5 QISAM Scan Processing Module (IGG019HB) SETL B Macro Routine (Part 5 of 14)

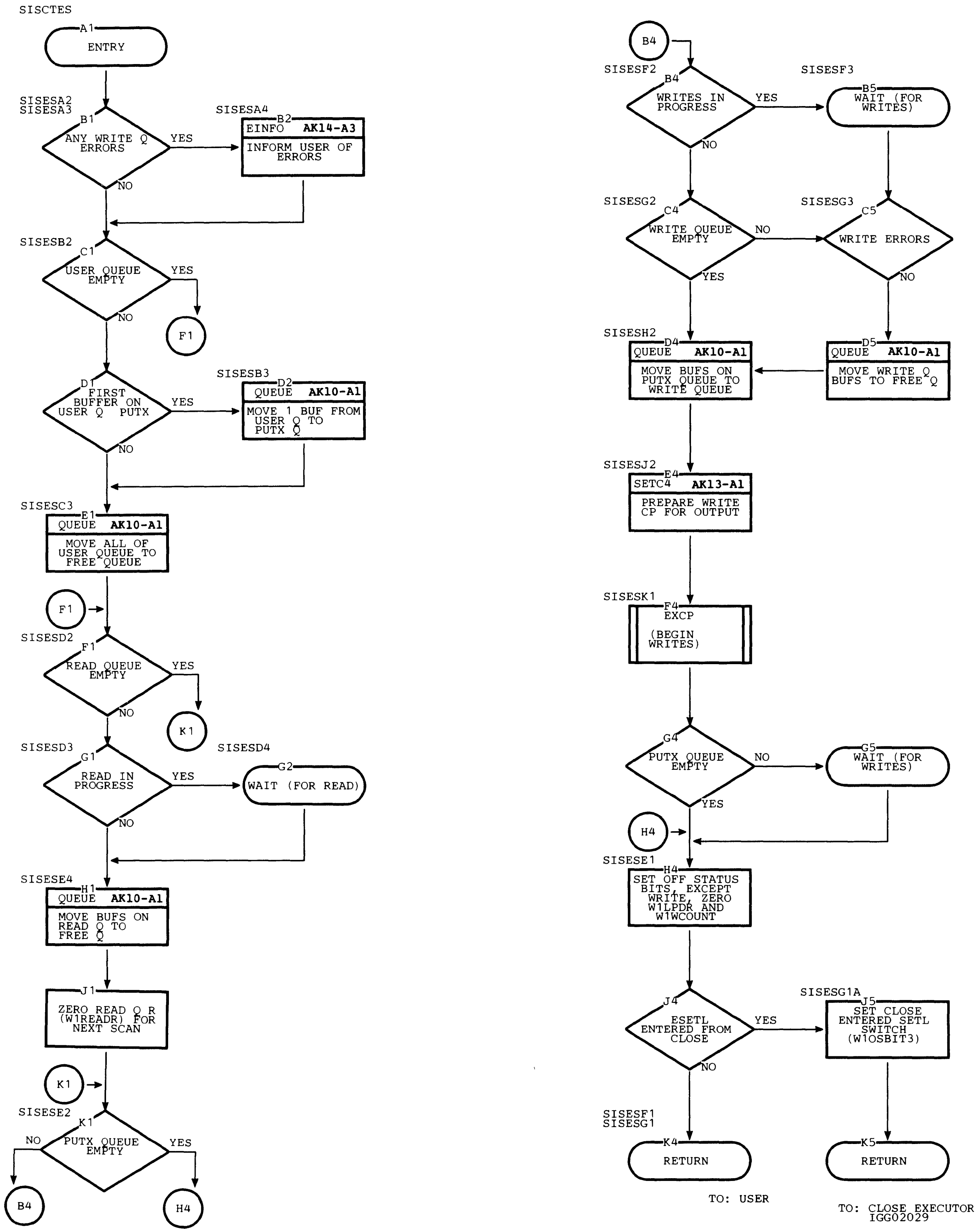


Chart AK6 QISAM Scan Processing Module (IGG019HB) ESETL Macro Routine (Part 6 of 14)

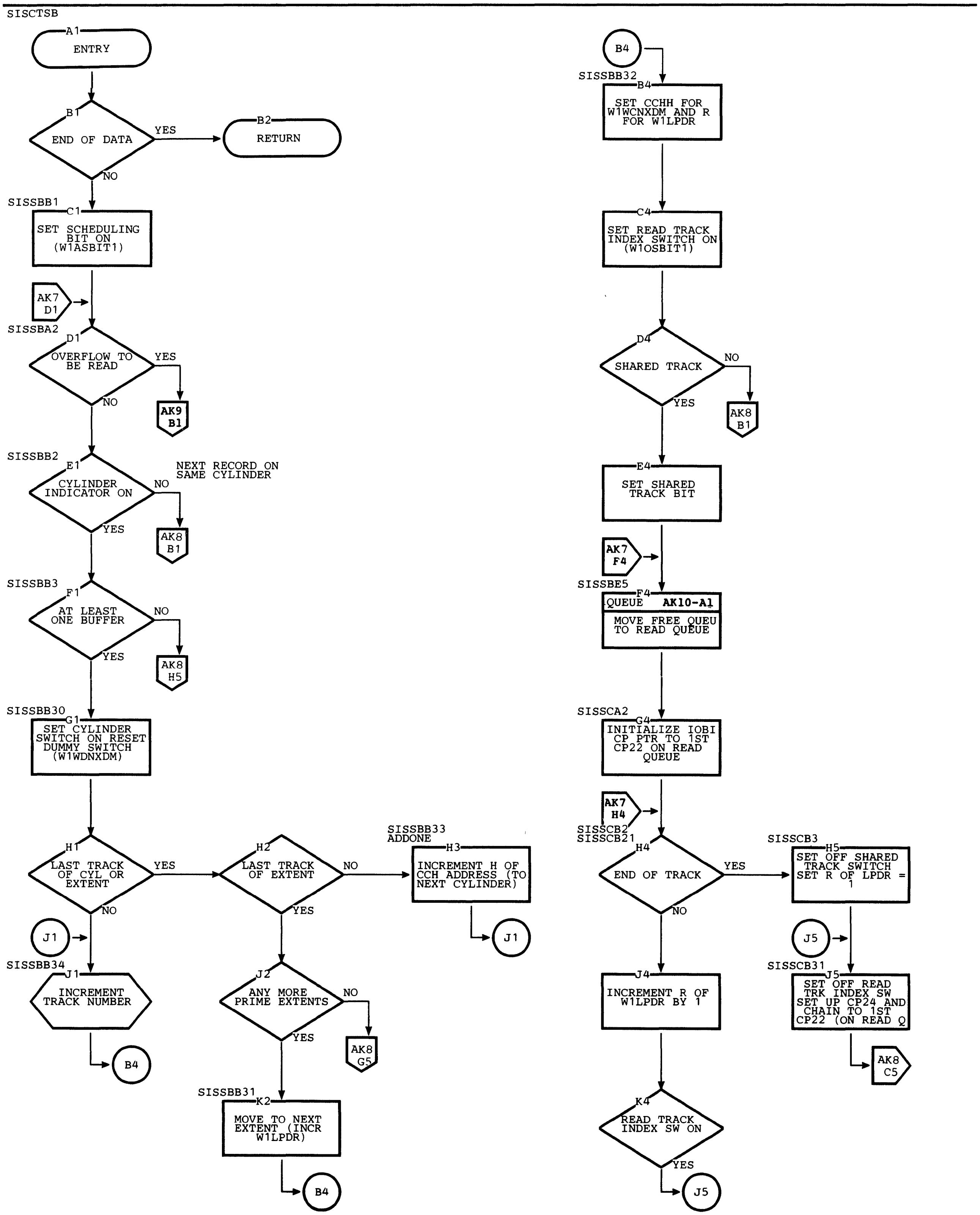


Chart AK7 QISAM Scan Processing Module (IGG019HB) Schedule Routine (Part 7 of 14)

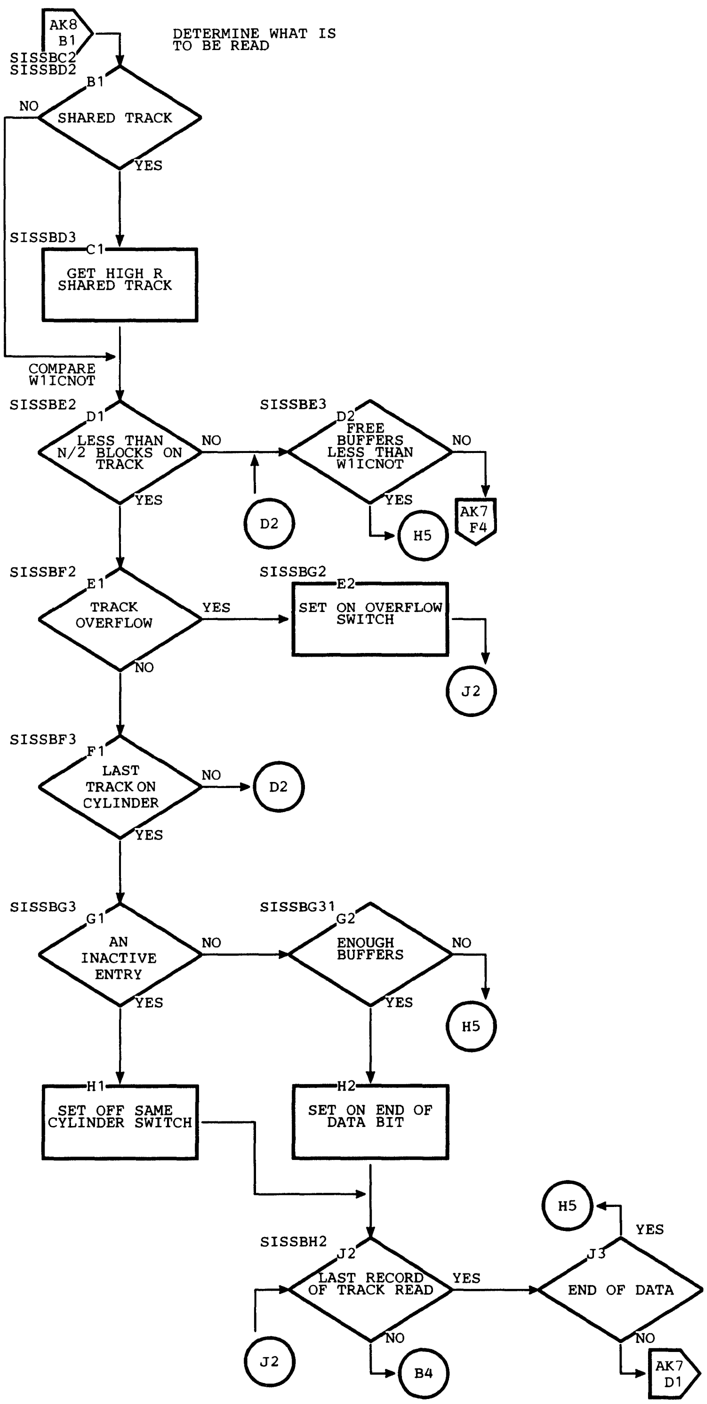


Chart AK8 QISAM Scan Processing Module (IGG019HB) Schedule Routine (Part 8 of 14)

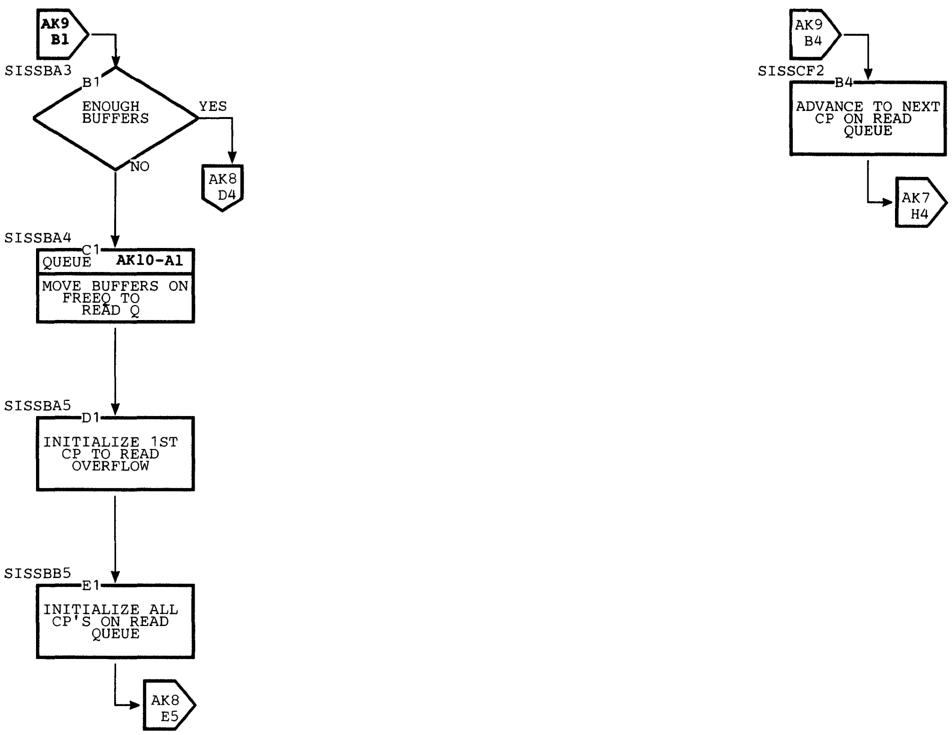


Chart AK9 QISAM Scan Processing Module (IGG019HB) Schedule Routine (Part 9 of 14)

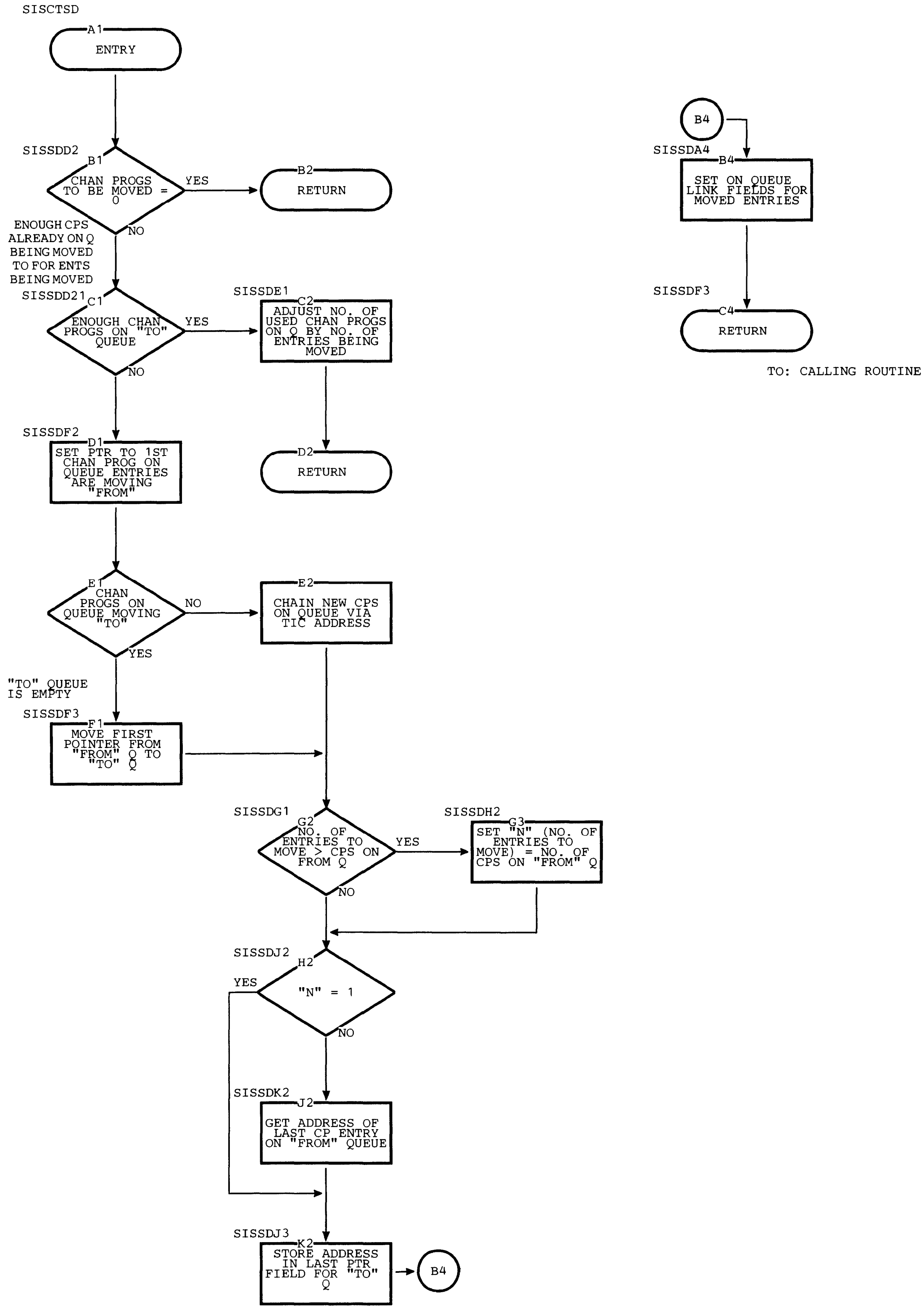


Chart AK10 QISAM Scan Processing Module (IGG019HB) Queue Routine (Part 10 of 14)

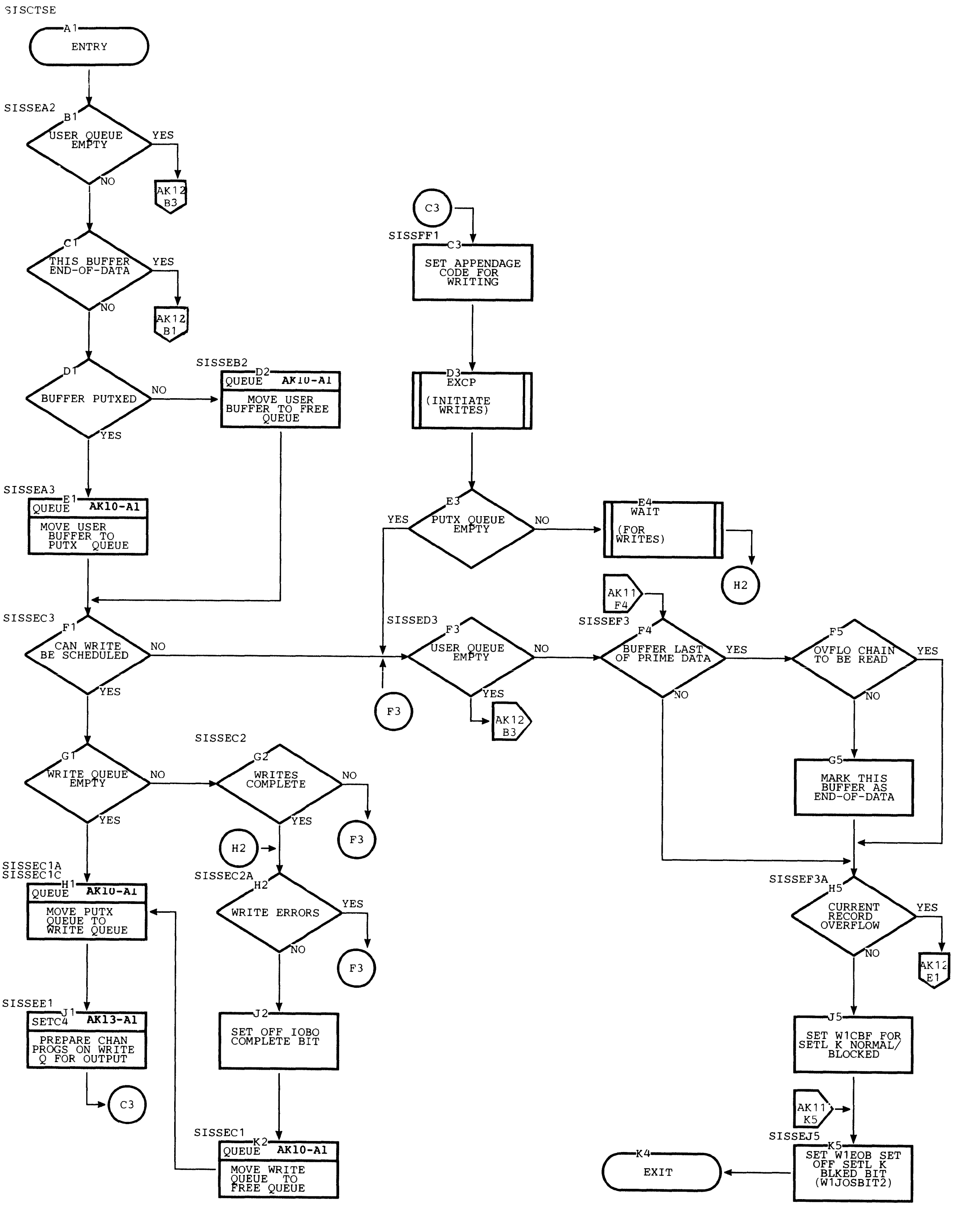


Chart AK11 QISAM Scan Processing Module (IGG019HB) End–of–Buffer Routine (Part 11 of 14)

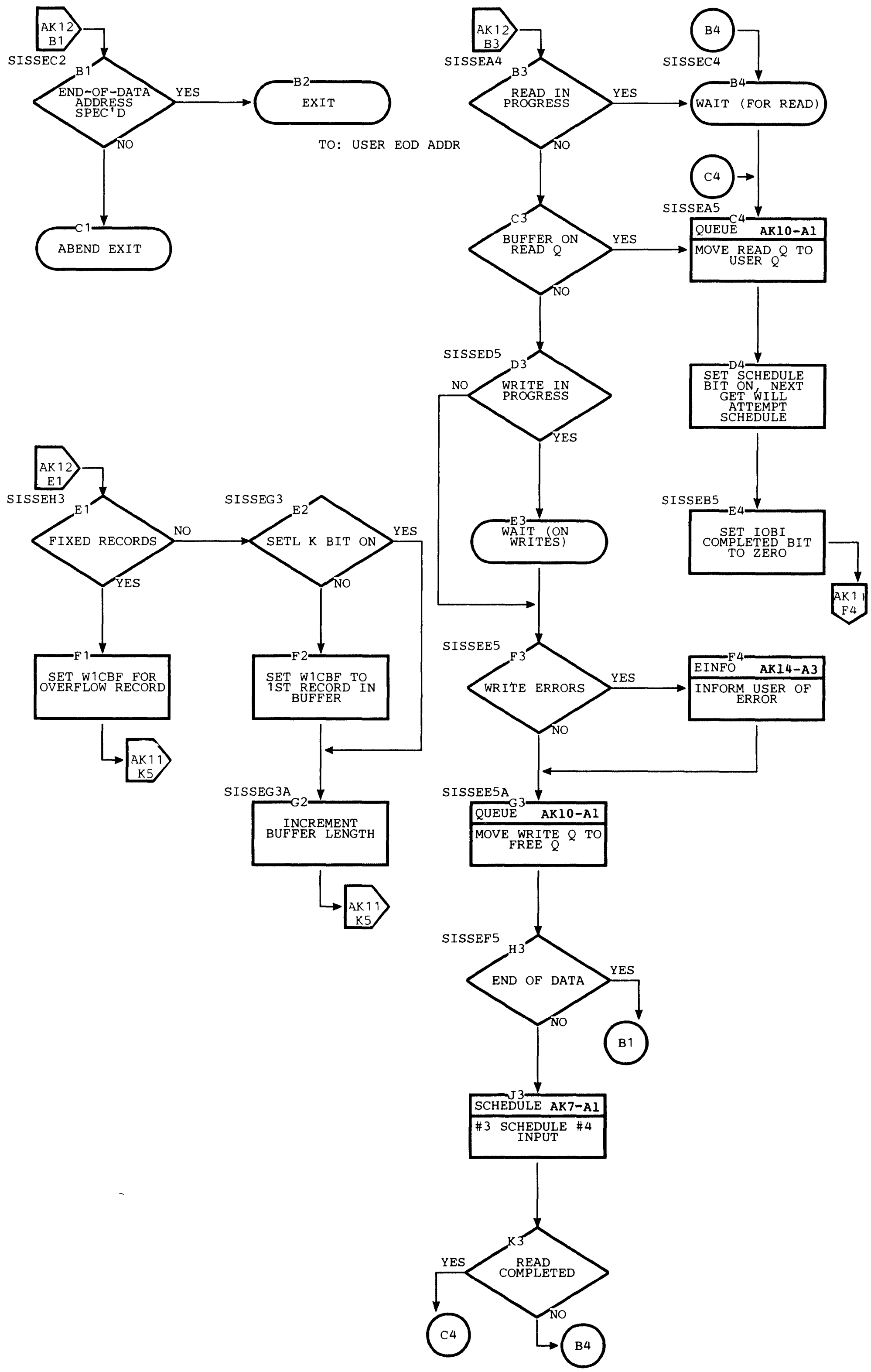


Chart AK12 QISAM Scan Processing Module (IGG019HB) End–of–Buffer Routine (Part 12 of 14)

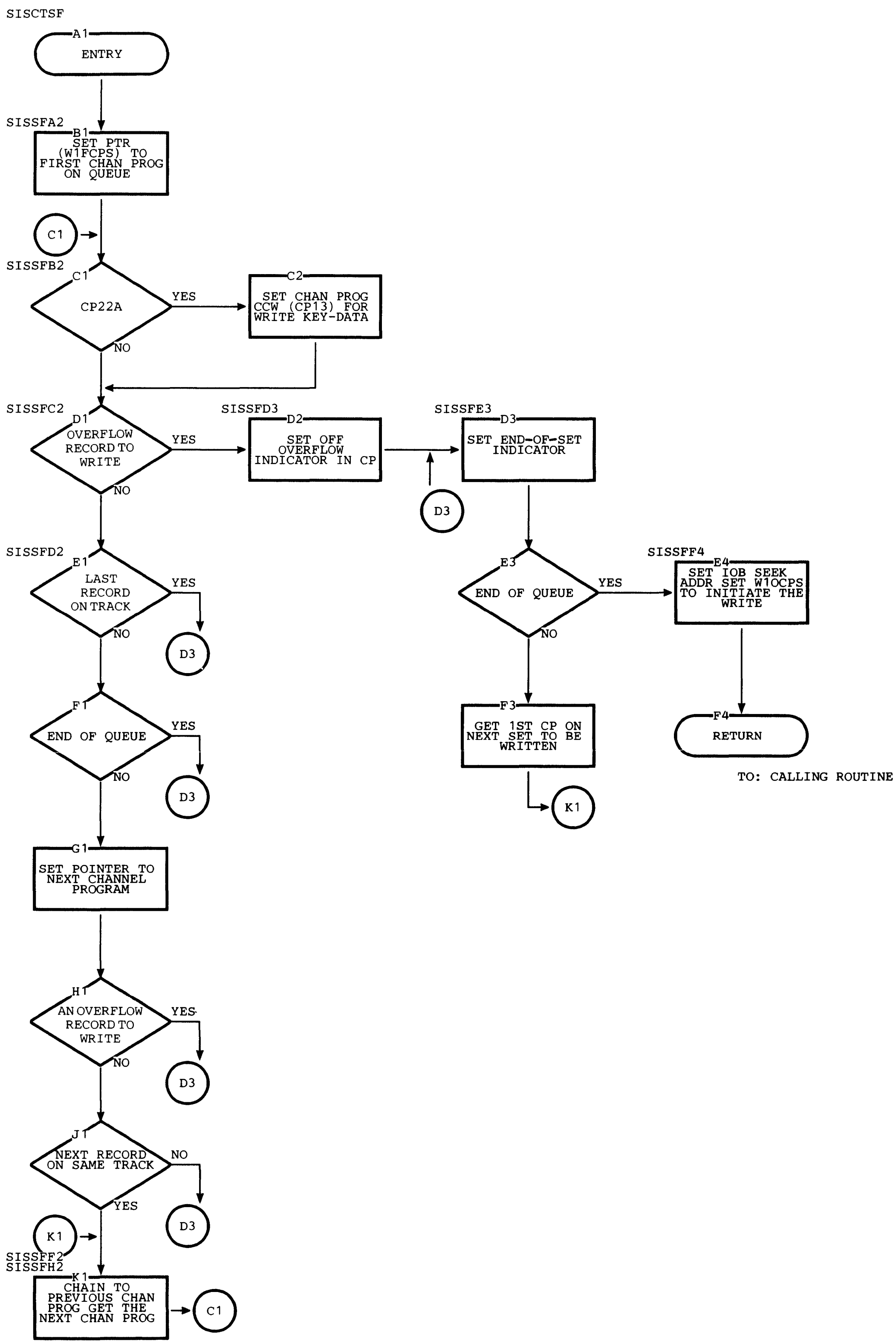


Chart AK13 QISAM Scan Processing Module (IGG019HB) SETC4 Subroutine (Part 13 of 14)

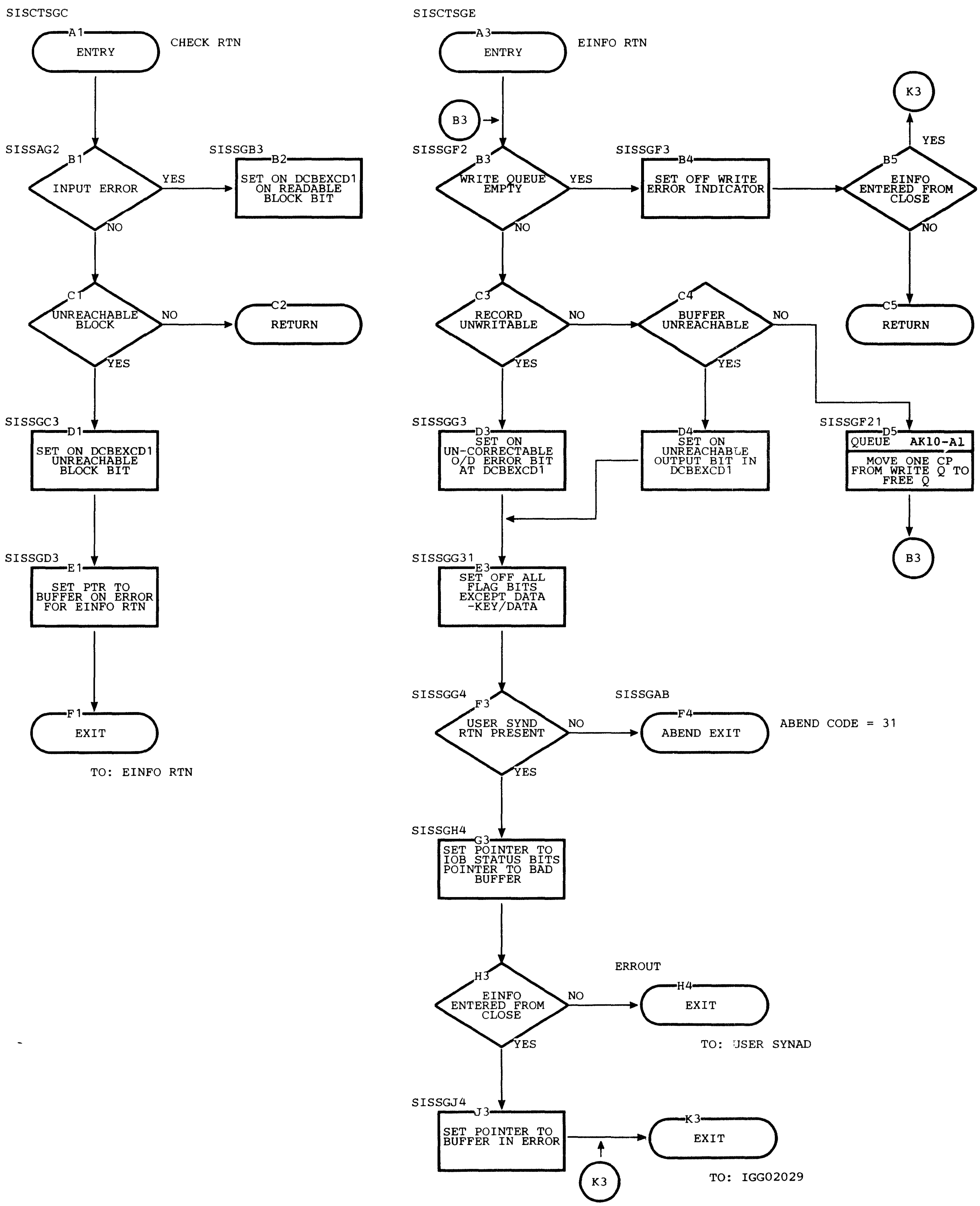


Chart AK14 QISAM Scan Processing Module (IGG019HB) Check Routine and EINFO (Error Information) Routine (Part 14 of 14)

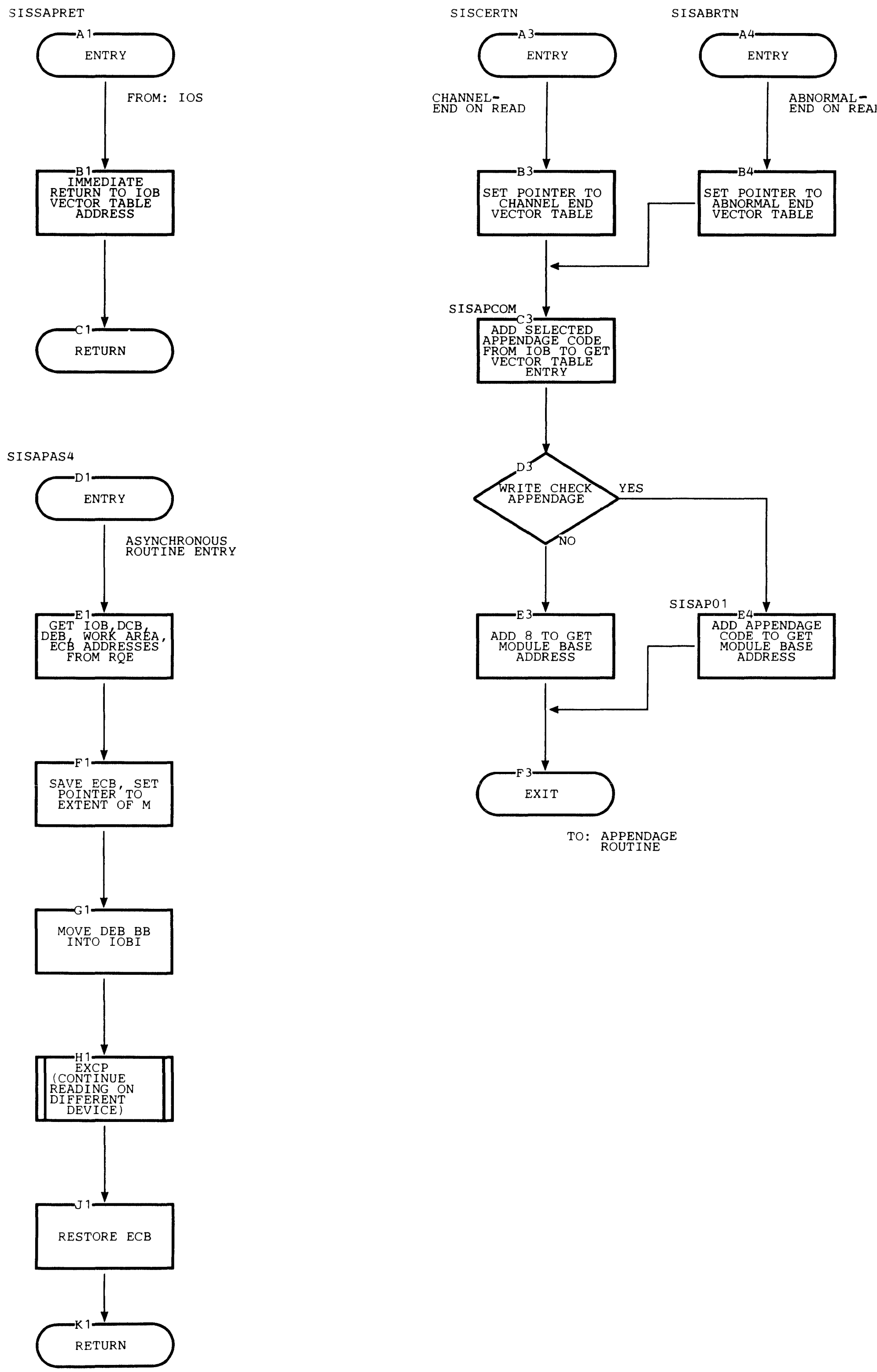


Chart AL1 Scan Mode Appendage (IGG019HG) (Part 1 of 3)

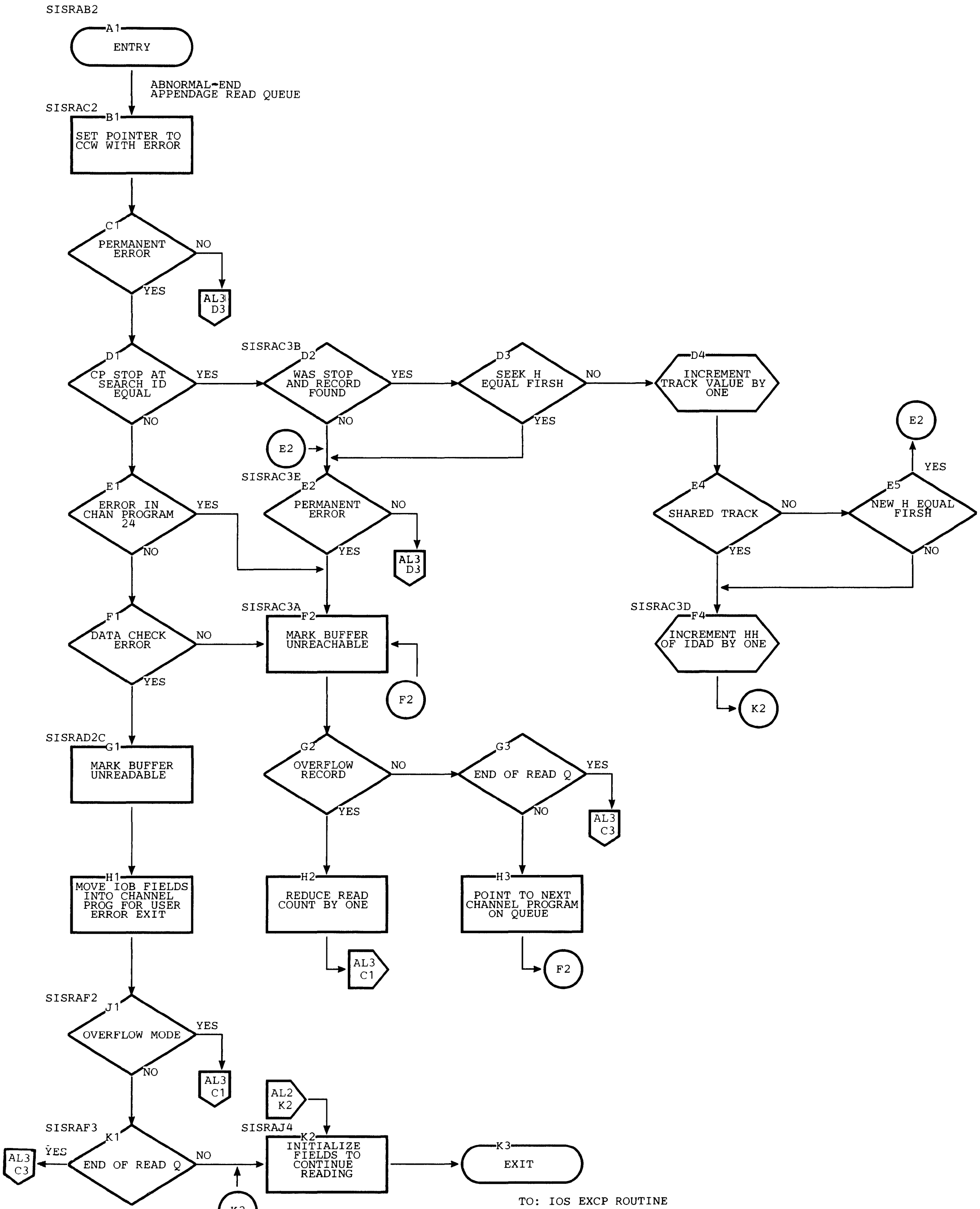


Chart AL2 Scan Mode Appendage (IGG019HG) Abnormal–end, Read Queue (Part 2 of 3)

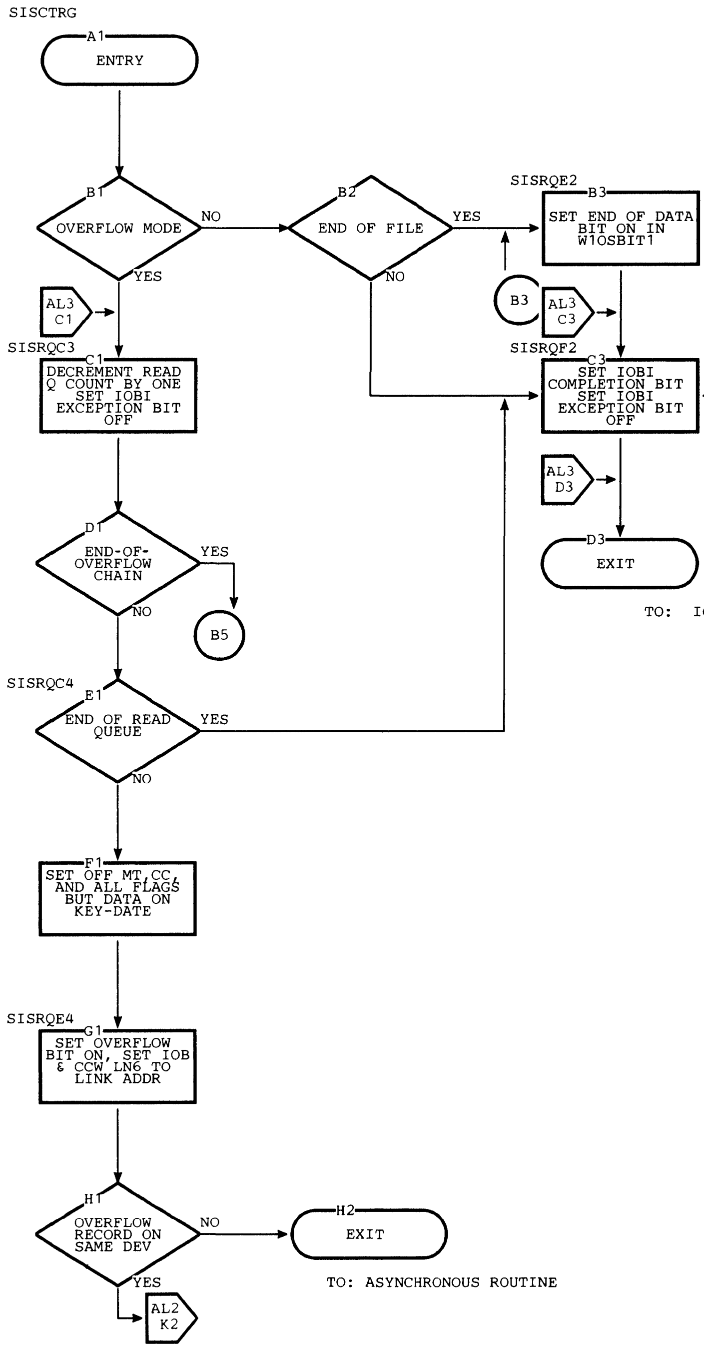


Chart AL3 Scan Mode Appendage (IGG019HG) Channel–end, Read Queue (Part 3 of 3)

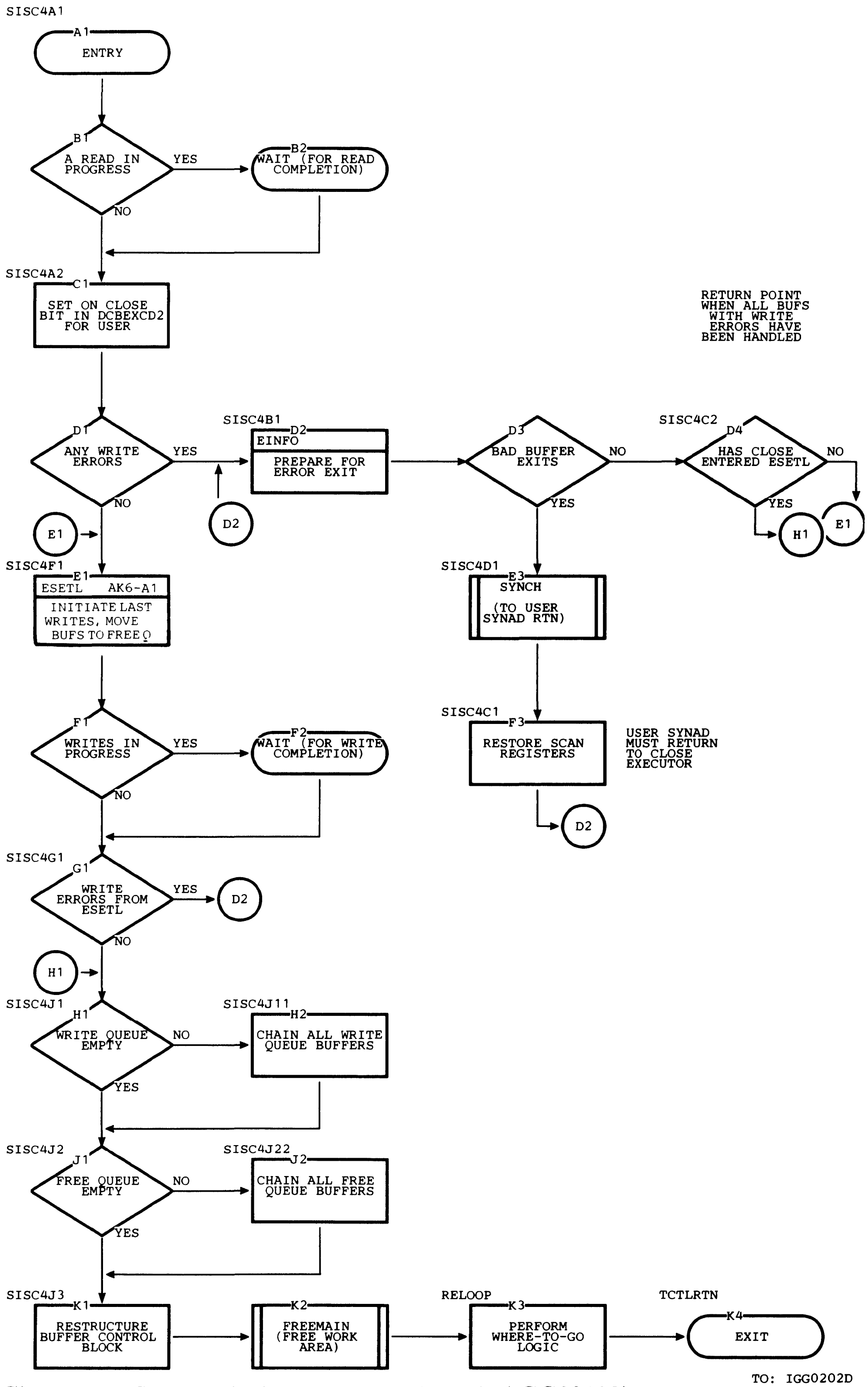


Chart AM1 Scan Mode Close Executor Module (IGG02029)

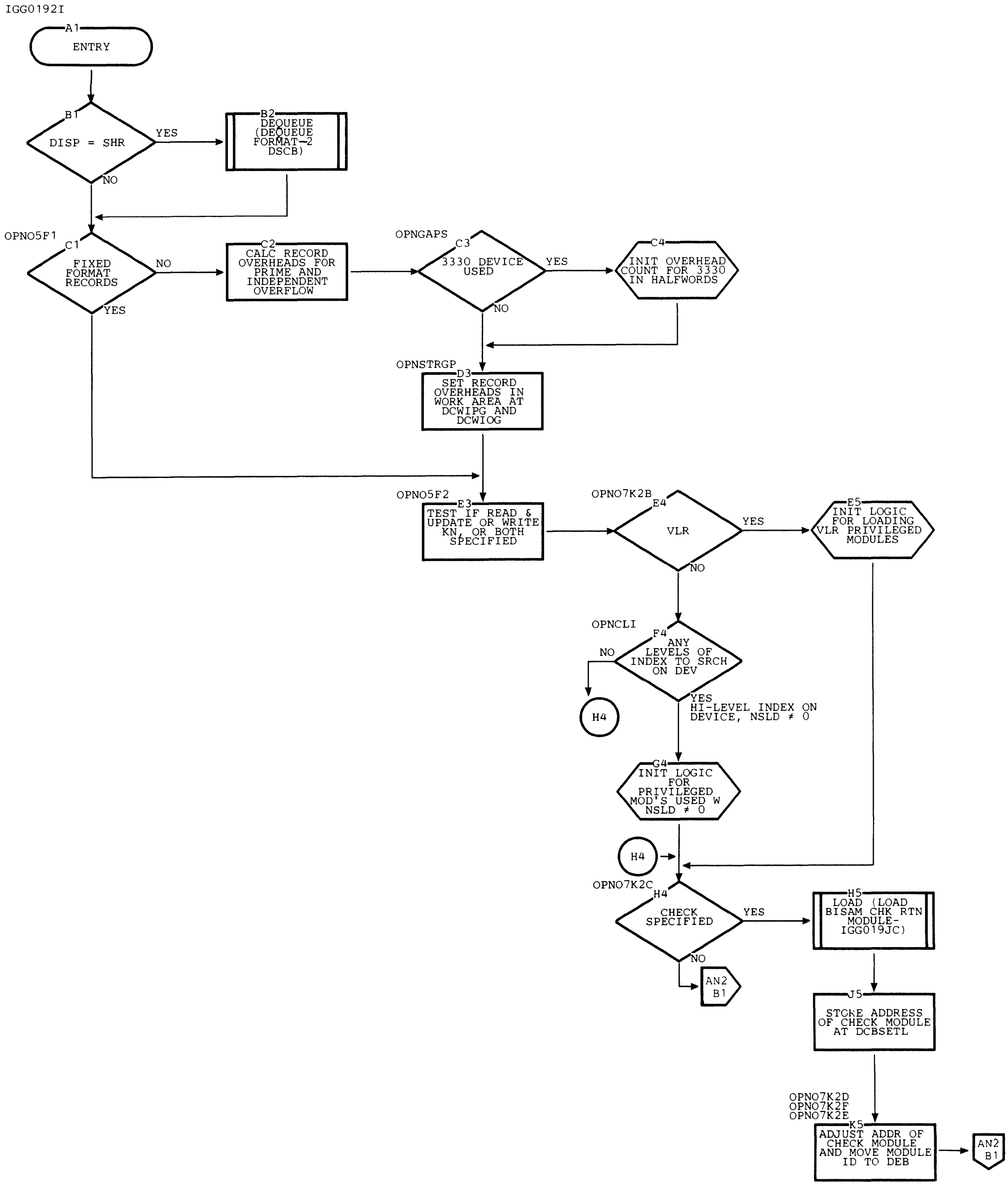


Chart AN1 BISAM Open Executor — Load Privileged Module (IGG0192I) (Part 1 of 2)

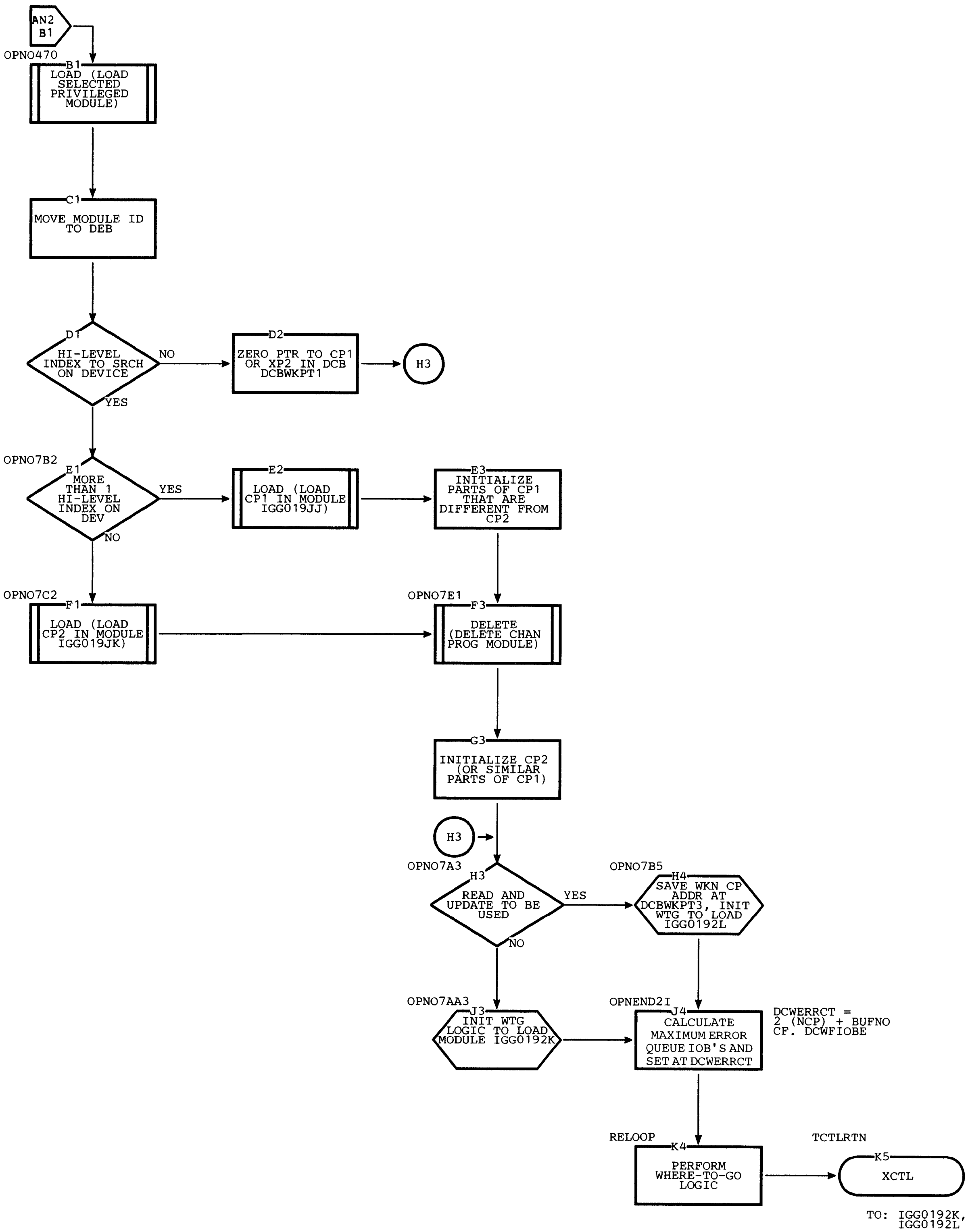


Chart AN2 BISAM Open Executor — Load Privileged Module (IGG0192I) (Part 2 of 2)

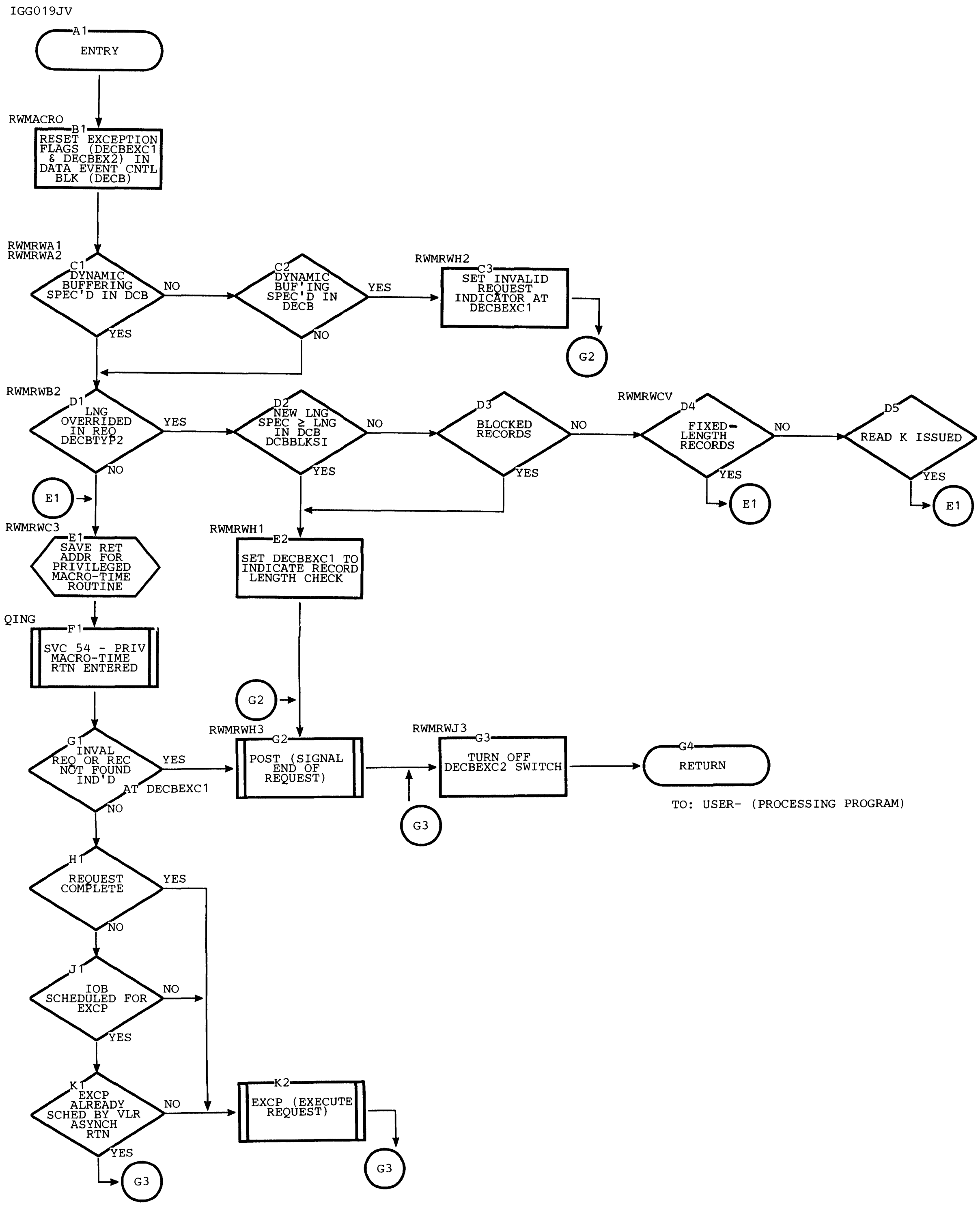


Chart AP1 BISAM Nonprivileged Macro-time Processing — READ K, READ KU, WRITE K (IGG019JV)

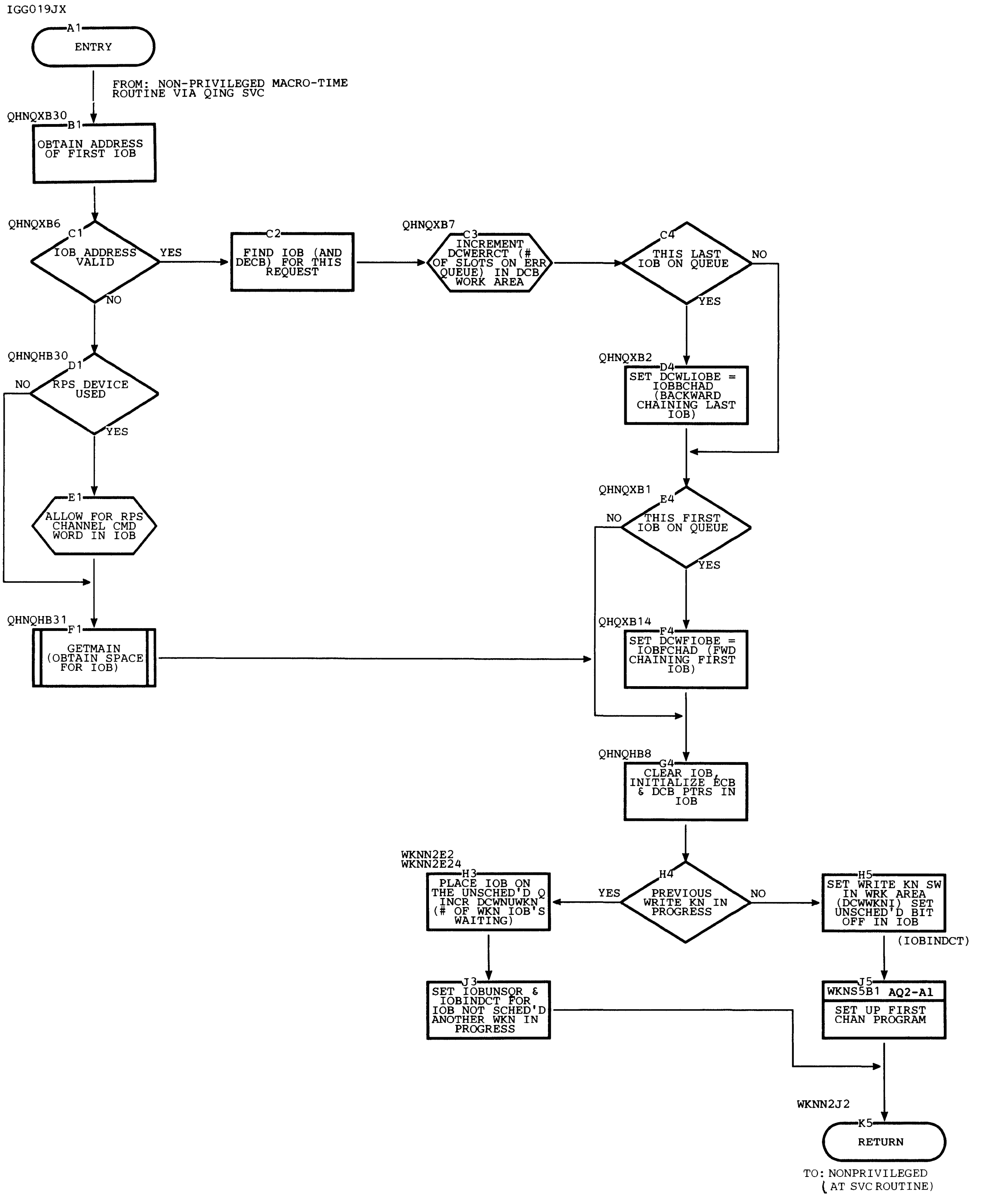


Chart AQ1 BISAM Privileged Macro–time Processing Module (WRITE KN, without Read and Update) (IGG019JX) (Part 1 of 2)

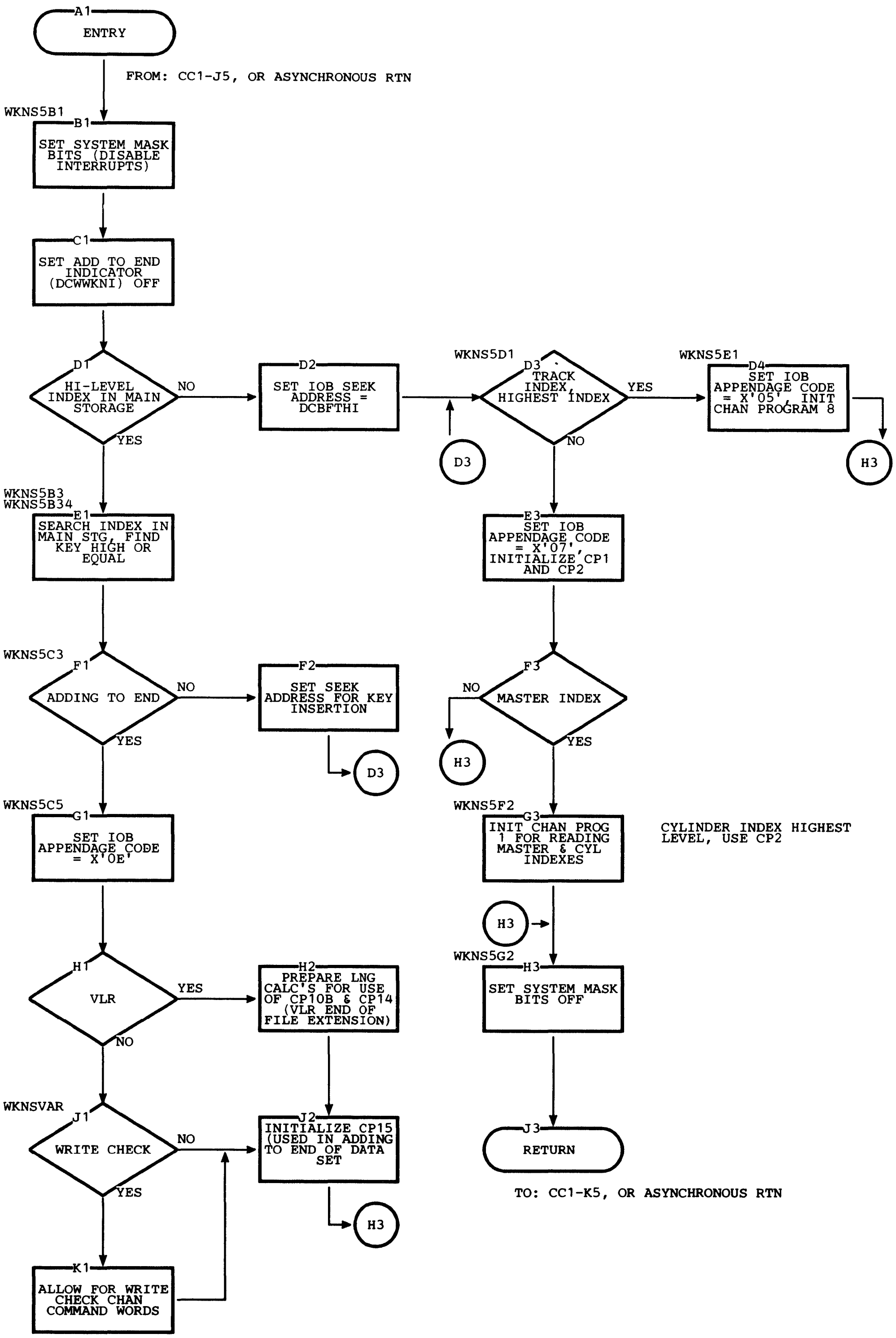


Chart AQ2 BISAM Privileged Macro–Time Processing Module (WRITE KN, without Read and Update) (IGG019JX) (Part 2 of 2)

SECTION 4: DIRECTORY

ISAM Modules Identified in Alphameric Sequence

All ISAM modules are listed according to function and mode in Figure 56 and in alphameric order in Figure 57.

Function \ Modes		QISAM Load Mode	QISAM Scan Mode	BISAM
Open Executor	Common	192A 192B 192C	192A 192B 192C	192A 192B 192C
	Validation Modules	1920 1950 1922	1920 1950 1922	1920 1950 1922
	Mode-oriented	192D 192T 195D 192E 192U 195G 192F 192V 195T 192G 1921 195U 192R 1925 196D 192S 1927 196G	1924 1928 1929	192H 192N 192Z 192I 192O 192J 192P 192K 192Q 192L 192W 192M 192X
Processing Modules	Macro-time	19GA 19IA 19I1 19GB 19IB 19I2	19HB 19HD 19HF 19HN	19JV 19J0 19H3 19JW 19J3 19H7 19JX 19J6 19J7
	Channel-end and Abnormal-end Appendages	19GC 19GD	19HG 19HH 19HI 19HJ 19HK	19GL 19G3 19IM 19GM 19G4 19IN 19GN 19G5 19IO 19GO 19G6 19I9 19G0 19G7 19G1 19G8 19G2 19G9
	SIO Appendage	19GG	19HA	19JH
	Channel Program	19GE 19GF 19IE 19IF	19HL	19HP 19JO 19JU 19JJ 19JP 19JK 19JQ 19JL 19JR 19JM 19JS 19JN 19JT
	Asynchronous			19GV 19GY 19IX 19GW 19GZ 19IY 19GX 19IZ
	Other		054(SVC54)	054(SVC54) 19JC (CHECK) 19JI (Dynamic Buffer)
Close Executor	Mode-oriented	202I 202K 202M 202J 202L 2028	2029	202A
	Common	202D	202D	202D

Figure 56. ISAM Modules Identified by Function and Mode

Module	Mode and Function	Text Pages	References Figures Pages	Flowcharts Pages
IGG019GA	QISAM load (macro routines)		30	
IGG019GB	QISAM load (macro routines)		30	
IGG019GC	QISAM load (appendage routines)		30	
IGG019GD	QISAM load (appendage routines)		30	
IGG019GE	QISAM load (channel programs)		30	
IGG019GF	QISAM load (channel programs)		30	
IGG019GG	QISAM load (RPS appendage routine)		30	
IGG019GL	BISAM (appendage routines)		72	
IGG019GM	BISAM (appendage routines)		72	
IGG019GN	BISAM (appendage routines)		72	
IGG019GO	BISAM (appendage routines)		72	
IGG019GV	BISAM (asynchronous routines)		71	
IGG019GW	BISAM (asynchronous routines)		71	
IGG019GX	BISAM (asynchronous routines)		71	
IGG019GY	BISAM (asynchronous routines)		71	
IGG019GZ	BISAM (asynchronous routines)		71	
IGG019G0	BISAM (appendage routines)		72	
IGG019G1	BISAM (appendage routines)		72	
IGG019G2	BISAM (appendage routines)		72	
IGG019G3	BISAM (appendage routines)		72	
IGG019G4	BISAM (appendage routines)		72	
IGG019G5	BISAM (appendage routines)		72	
IGG019G6	BISAM (appendage routines)		72	
IGG019G7	BISAM (appendage routines)		72	
IGG019G8	BISAM (appendage routines)		72	
IGG019G9	BISAM (appendage routines)		72	
IGG019HA	QISAM scan (RPS appendage routines)		51	
IGG019HB	QISAM scan (macro routines)		51	116
IGG019HD	QISAM scan (macro routines)		51	
IGG019HF	QISAM scan (macro routines)		51	
IGG019HG	QISAM scan (appendage routines)		51	130
IGG019HH	QISAM scan (appendage routines)		51	
IGG019HI	QISAM scan (appendage routines)		51	
IGG019HJ	QISAM scan (appendage routines)		51	
IGG019HK	QISAM scan (appendages)		51	
IGG019HL	QISAM scan (channel programs)		51	
IGG019HN	QISAM scan (macro routines)		51	
IGG019HP	BISAM (channel programs)		73	
IGG019H3	BISAM (macro routines)		70	
IGG019H7	BISAM (macro routines)		70	
IGG019IA	QISAM load (macro routines)		30	
IGG019IB	QISAM load (macro routines)		30	
IGG019IE	QISAM load (channel programs)		30	
IGG019IF	QISAM load (channel programs)		30	
IGG019IM	BISAM (appendage routines)		72	
IGG019IN	BISAM (appendage routines)		72	
IGG019IO	BISAM (appendage routines)		72	
IGG019IX	BISAM (asynchronous routines)		71	
IGG019IY	BISAM (asynchronous routines)		71	
IGG019IZ	BISAM (asynchronous routines)		71	
IGG019I1	QISAM load (macro routines)		30	
IGG019I2	QISAM load (macro routines)		30	

Figure 57 (Part 1 of 3). ISAM Modules Identified in Alphameric Sequence

Module	Mode and Function	References Text Pages	Figures Pages	Flowcharts Pages
IGG019I9	BISAM (appendage routines)		72	
IGG019JC	BISAM (check routine)		68	
IGG019JH	BISAM (RPS appendage routine)	68	72	
IGG019JI	BISAM (dynamic buffering routine)	67-68		
IGG019JJ	BISAM (channel programs)		73	
IGG019JK	BISAM (channel programs)		73	
IGG019JL	BISAM (channel programs)		73	
IGG019JM	BISAM (channel programs)		73	
IGG019JN	BISAM (channel programs)		73	
IGG019JO	BISAM (channel programs)		73	
IGG019JP	BISAM (channel programs)		73	
IGG019JQ	BISAM (channel programs)		73	
IGG019JR	BISAM (channel programs)		73	
IGG019JS	BISAM (channel programs)		73	
IGG019JT	BISAM (channel programs)		73	
IGG019JU	BISAM (channel programs)		73	
IGG019JV	BISAM (macro routines)		71	136
IGG019JW	BISAM (macro routines)		71	
IGG019JX	BISAM (macro routines)		70	137
IGG019JO	BISAM (macro routines)		70	
IGG019J3	BISAM (macro routines)		70	
IGG019J6	BISAM (macro routines)		70	
IGG019J7	BISAM (macro routines)		70	
IGG0192A	Common open executor	9	10	95
IGG0192B	Common open executor	9	10	98
IGG0192C	Common open executor	9	10,17,38,59	100
IGG0192D	QISAM load (open executor)	19	17	105
IGG0192E	QISAM load (open executor)	19	17	
IGG0192F	QISAM load (open executor)	20	17	
IGG0192G	QISAM load (open executor)	20	17	
IGG0192H	BISAM (open executor)	58	59	
IGG0192I	BISAM (open executor)	57	59	134
IGG0192J	BISAM (open executor)	60	59	
IGG0192K	BISAM (open executor)	58	59	
IGG0192L	BISAM (open executor)	58	59	
IGG0192M	BISAM (open executor)	58	59	
IGG0192N	BISAM (open executor)	60	59	
IGG0192O	BISAM (open executor)	60	59	
IGG0192P	BISAM (open executor)	57	59	
IGG0192Q	BISAM (open executor)	58	59	
IGG0192R	QISAM load (open executor)	23	17	
IGG0192S	QISAM load (open executor)	23	17	
IGG0192T	QISAM load (open executor)		17	
IGG0192U	QISAM load (open executor)	23	17	101
IGG0192V	QISAM load (open executor)		17	
IGG0192W	BISAM (open executor)	57	59	
IGG0192X	BISAM (open executor)	58	59	
IGG0192Z	BISAM (open executor)	60	59	
IGG01920	Common open executor (validation)	12	17,38,59	101
IGG01921	QISAM load (open executor)	16	17	102
IGG01922	Common open executor (validation)	12	17,38,59	103
IGG01924	QISAM scan (open executor)	37	38	109

Figure 57 (Part 2 of 3). ISAM Modules Identified in Alphameric Sequence

Module	Mode and Function	References Text Pages	Figures Pages	Flowcharts Pages
IGG01925	QISAM load (open executor)	19	17	
IGG01927	QISAM load (open executor)	19	17	
IGG01928	QISAM scan (open executor)	37	38	110
IGG01929	QISAM scan (open executor)	37	38	
IGG0195G	QISAM load (open executor)	22	17	
IGG0195G	QISAM load (open executor)	21	17	
IGG0195T	QISAM load (open executor)	22	17	
IGG0195U	QISAM load (open executor)	22	17	
IGG01950	Common open executor (validation)	12	17,38,59	
IGG0196D	QISAM load (open executor)	21	17	108
IGG0196G	QISAM load (open executor)	21	17	
IGG0202A	BISAM (close executor)	13	14	
IGG0202D	Common close executor	13	14,36	114
IGG0202I	QISAM load (close executor)	34	14,36	
IGG0202J	QISAM load (close executor)	35	14,36	
IGG0202K	QISAM load (close executor)	35	14,36	
IGG0202L	QISAM load (close executor)	35	14,36	
IGG0202M	QISAM load (close executor)	35	14,36	
IGG02028	QISAM load (close executor)	35	14,36	
IGG02029	QISAM scan (close executor)	52	14	133

Figure 57 (Part 3 of 3). ISAM Modules Identified in Alphameric Sequence

SECTION 5: DATA AREAS

ISAM Control Blocks and Data Areas

Indexed sequential access method (ISAM) routines use a number of control blocks that are common to all of data management.

The control blocks are:

Data control block (DCB)
Data event control block (DECB)
Data set control block (DSCB)
Data extent block (DEB)
Input/output block (IOB)

ISAM routines also use certain work areas and buffer control areas.

The ISAM work areas are:

QISAM load mode work area
QISAM scan mode work area
BISAM work area
QISAM load mode track–index save area (TISA)
ISAM DCB field area

The ISAM buffer control areas are:

BISAM dynamic buffering buffer control block (BCB)
QISAM buffer control block (BCB)
QISAM load mode buffer control table (IOBBCT)

Data Control Block (DCB)

The data control block (DCB) is the major means of communication between the problem program and the control program. The sources for ISAM DCB information are: the open executors, the DCB macro instruction, the problem program, the data definition (DD) statement, and the data set control block (DSCB). Figure 58 shows the portion of the DCB that is unique to ISAM.

	49(31) DCBGET/DCBPUT		
52(34) DCBOPTCD	53(35) DCBMAC	54(36) DCBNTM	53(17) DCBCYLOV
56(38) DCBSYNAD			
60(3C) DCBRKP		62(3E) DCBBLKSI	
64(40) DCBMSWA			
68(44) DCBSMSI		70(46) DCBSMSW	
72(48) DCBNCP	73(49) DCBMSHI		
76(4C) DCBSETL			
80(50) DCBEXCD1	81(51) DCBEXCD2	82(52) DCBLRECL	
84(54) DCBESETL			
88(58) DCBLRAN			
92(5C) DCBLWKN			
96(60) DCBRELSE			
100(64) DCBPUTX			
104(68) DCBRELX			
108(6C) DCBFREED			
112(70) DCBHIRTI	113(71) DCBFTMI2		
120(78) DCBLEMI2			
	125(7D) DCBFTMI3		
132(84) DCBLEMI3			
	137(89) DCBNLEV	138(8A) DCBFIRSH	
DCBFIRSH (cont.)	141(8D) DCBHMASK	142(8E) DCBLDT	
144(90) DCBHIRCM	145(91) DCBHIRPD	146(92) DCBHIROV	147(93) DCBHIRSH
148(94) DCBTDC		150(96) DCBNCHRI	

Figure 58 (Part 1 of 2). BISAM/QISAM DCB

Offset	Field	Offset	Field	Offset	Field
152(98)	DCBRORG3				
156(9C)	DCBNREC				
160(A0)	DCBST	161(A1)	DCBFTCI		
168(A8)	DCBHIIOV	169(A9)	DCBFTMI1		
176(B0)	DCBNTHI	177(B1)	DCBFTHI		
184(B8)	DCBLPDA				
192(C0)	DCBLETI	197(C5)	DCBOVDEV	198(C6)	DCBNBOV
200(C8)	DCBLECI	205(CD)	Reserved	206(CE)	DCBRORG2
208(D0)	DCBLEMI1	213(D5)	Reserved	214(D6)	DCBNOREC
216(D8)	DCBLIOV				
224(E0)	DCBRORG1	226(E2)	Reserved		
228(E4)	DCBWKPT1				
232(E8)	DCBWKPT2				
236(EC)	DCBWKPT3				
240(F0)	DCBWKPT4				
244(F4)	DCBWKPT5				
248(F8)	DCBWKPT6				

Figure 58 (Part 2 of 2). BISAM/QISAM DCB

Offset	Field Name	Bytes	Field Description
49(31)	DCBGET/DCBPUT	3	Address of Get module or address of Put module.

Note: This field is not used by ISAM routines. See the extension of the data extent block (DEB).

53(34)	DCBOPTCD	1	Option codes: Bit 0 W — Write validity–check 1 U — Full track–index write 2 M — Master index(es) 3 I — Independent overflow area 4 Y — Cylinder overflow area 5 Reserved 6 L — Delete option 7 R — Reorganization criteria
53(35)	DCBMAC	1	MACRF Extension for ISAM Bit 0 3 — Reserved 4 U — Update type of READ 5 U — Update type of WRITE 6 A — Add type of WRITE 7 Reserved
54(36)	DCBNTM	1	The number of tracks that determine the development of a master index. If the number of tracks in the cylinder index exceeds this number, a master index is developed. If the number of tracks in the master index in turn exceeds this number, then a higher level master index is developed, and so forth. Maximum permissible value: 99.
55(37)	DCBCYLOV	1	The number of tracks to be reserved on each prime–data cylinder to hold records that overflow from other tracks on that cylinder. Refer to the section on allocating space for an ISAM data set in the *OS Data Management Services Guide,* GC28–3746, to determine how to calculate the maximum number.
56(38)	DCBSYNAD	4	Address of user's synchronous error routine to be entered when uncorrectable errors are detected in processing data records.

Offset	Field Name	Bytes	Field Description
60(3C)	DCBRKP	2	The relative position of the first byte of the key within each logical record. Maximum permissible value: logical record minus key length.
62(3E)	DCBBLKSI	2	Blocksize. For fixed–length record formats, this must be an integral multiple of DCBLRECL. For variable–length record formats, it must be maximum blocksize and must include the 4–byte block length field.
64(40)	DCBMSWA	4	Address of a work area supplied by the user when new records are being added to an existing data set.
68(44)	DCBSMSI	2	Number of bytes in an area reserved to hold the highest level index. The address of this area is in DCBMSHI. Maximum size allowed is 65,535 bytes.
70(46)	DCBSMSW	2	Number of bytes in work area used by the control program when new records are being added to the data set. The address of this area is in DCBMSWA. Maximum size allowed is 32,767 bytes.
72(48)	DCBNCP	1	Number of copies of the READ/WRITE type K channel programs that are to be established for this data control block (99 maximum).
73(49)	DCBMSHI	3	Address of a main–storage area to hold the highest level index.
76(4C)	DCBSETL	4	Address of SETL module for QISAM. Address of Check module for BISAM.
80(50)	DCBEXCD1	1	First byte in which exceptional conditions detected in processing data records are reported to the user (see "Appendix B: ISAM Channel Programs"). Bit 0 —Lower key limit not found 1 —Invalid device address for lower limit 2 —Space not found 3 —Invalid request 4 —Uncorrectable input error 5 —Uncorrectable output error 6 —Unreachable block (input) 7 —Unreachable block (update)

Offset	Field Name	Bytes	Field Description
81(51)	DCBEXCD2	1	Second byte in which exceptional conditions detected in processing data records are reported to the user (See "Appendix B: ISAM Channel Programs"). Bit 0 —Sequence check 1 —Duplicate record 2 —DCB closed when error was detected 3 —Overflow record 4 —The logical record length specified in the record field is greater than that specified in DCBLRECL. (Variable–length records only).
82(52)	DCBLRECL	2	Logical record length for fixed–length record formats. For variable–length record formats, may either be maximum logical record length or an actual logical record length changed dynamically by the user when creating the data set.
84(54)	DCBESETL	4	QISAM: Address of the ESETL routine in the Get module.
88(58)	DCBLRAN	4	Address of READ/WRITE K module.
92(5C)	DCBLWKN	4	Address of WRITE KN module.
96(60)	DCBRELSE	4	Work area for temporary storage of register contents.
100(64)	DCBPUTX	4	Work area for temporary storage of register contents.
104(68)	DCBRELX	4	Reserved.
108(6C)	DCBFREED	4	Address of dynamic buffering module.

Note: This field is not used by ISAM routines. See the extension of the data extent block (DEB).

112(70)	DCBHIRTI	1	Highest number of index entries that fit on a prime–data track.

Offset	Field Name	Bytes	Field Description
113(71)	DCBFTMI2	7	Direct–access device address of the first track of the second level master index (in the form MBBCCHH). If the second level master index crosses an extent boundary, the first B byte holds the M of the last active entry in this master index (LEMI2). Otherwise, the first B byte will be 0.
120(78)	DCBLEMI2	5	Direct–access device address of the last active entry in the second level master index (in the form CCHHR). The M for this address is the same as the M contained in the field DCBFTMI2 (above) if the first B byte of that field is 0. Otherwise, the M for the address is contained in the first B byte of DCBFTMI2.
125(7D)	DCBFTMI3	7	Direct–access device of the first track of the third level master index (in the form MBBCCHH). As for FTMI2, the first B byte will either be 0 or will hold the M of the last active entry in the index (in this case, the M for LEMI3).
132(84)	DCBLEMI3	5	Direct–access device address of the last active entry in the third level master index (in the form CCHHR). The M for this address is the same as the M for FTMI3 if the first B byte is contained in the first B byte of FTMI3.
137(89)	DCBNLEV	1	Number of levels of index. Has a maximum value of 4, corresponding to the case where there is a cylinder index and three master indexes. If the track index is the highest level index, then NLEV=0.
138(8A)	DCBFIRSH	3	HHR of the first data record on each cylinder. The first data record on each cylinder may be on the last track of the track index for that cylinder (in which case, the track is said to be shared).
141(8D)	DCBHMASK	1	If the device is a 2301 drum, HMASK = X‘37’; otherwise, HMASK = X‘FF’.

Offset	Field Name	Bytes	Field Description
142(8E)	DCBLDT	2	HH of the last prime–data track on each cylinder. This differs from the last physical track on a cylinder when the user has reqeusted cylinder overflow areas.
144(90)	DCBHIRCM	1	Highest possible R for tracks of the cylinder and master indexes. This is the number of index entries that fits on a track. Note that these indexes may be on a different type of device than the rest of the data set.
145(91)	DCBHIRPD	1	Highest possible R for any prime–data track. This is the number of records or blocks that fits on a prime–data track.
146(92)	DCBHIROV	1	Highest possible R for overflow data tracks, fixed–length record formats only. This is the number of fixed–length records or blocks that fits on an overflow data track.
147(93)	DCBHIRSH	1	R of the last data record on a shared track, if applicable (fixed–length records only).
148(94)	DCBTDC	2	Tag deletion count. A field reserved for the user in which he may keep the number of records that have been tagged for deletion. It is merged to and from the format–2 DSCB for BISAM, scan mode, and load mode resume load.
150(96)	DCBNCHRI	2	Number of storage locations needed to hold the highest level index. This is equal to (KL + 10) (N), where N is the total number of index entries, including dummy entries. Note that the track index may be the highest level index, and the track index is never held and searched in main storage.
152(98)	DCBRORG3	4	For each use of the data set, the number of Read or Write accesses to an overflow record which is not the first in a chain of such records.
156(9C)	DCBNREC	4	Number of logical records in the prime–data area.

Offset	Field Name	Bytes	Field Description
160(A0)	DCBST	1	Status indicators. Bit 0 —Single schedule mode 1 —Key sequence to be checked 2 —Initial load has been completed 3 —Data set extension (resume loading) will begin on new cylinder 4 —Reserved 5 —First macro not yet received 6 —Last block full 7 —Last track full
161(A1)	DCBFTCI	7	Direct–access device address of the first track of the cylinder index (in the form MBBCCHH). As for FTMI2, the first B byte will either be 0 or will hold the M of the last active entry in the index (in this case, the M for LEMI).
168(A8)	DCBHIIOV	1	Highest R for independent overflow track.
169(A9)	DCBFTMI1	7	Direct–access device address of the first track of the first level master index (in the form MBBCCHH). As for FTMI2, the first B byte will either be 0 or will hold the M of the last active entry in the index (in this case, the M for LEMI1).
176(B0)	DCBNTHI	1	Number of tracks of the high–level index.
177(B1)	DCBFTHI	7	Direct–access device address of the first track of the highest level index (in the form MBBCCHH). Note that this may be the track index.
184(B8)	DCBLPDA	8	Direct–access device address of the last prime–data record in the prime–data area (in the form MBBCCHHR).
192(C0)	DCBLETI	5	Direct–access device address of the last active normal entry of the track index on the last active cylinder (in the form CCHHR). The M of this entry is the same as the M of LPDA.
197(C5)	DCBOVDEV	1	Independent overflow device type (field description same as DCBDEVT).

Offset	Field Name	Bytes	Field Description
198(C6)	DCBNBOV	2	Number of bytes remaining on current overflow track (variable–length records only).
200(C8)	DCBLECI	5	Direct–access device address of the last active entry in the cylinder index (in the form CCHHR). The M for this address is the same as the M for FTCI if the first B byte in FTCI is 0. Otherwise the M for this address is contained in the first B byte of FTCI.
205(CD)		1	Reserved for future use.
206(CE)	DCBRORG2	2	Number of tracks (partially or wholly) remaining in the independent overflow area.
208(D0)	DCBLEMI1	5	Direct–access device address of the last active entry in the first level index (in the form CCHHR). The M for this address is the same as the M for FTMI1 if the first B byte in FTMI1 is 0. Otherwise the M for this address is contained in the first B byte of FTMI1.
213(D5)		1	Reserved for future use.
214(D6)	DCBNOREC	2	Number of logical records in an overflow area.
216(D8)	DCBLIOV	8	Direct–access device address of the last record written in the independent overflow area (in the form MBBCCHHR).
224(E0)	DCBRORG1	2	Number of cylinder overflow areas that are full.
226(E2)		2	Reserved for future use.
228(E4)	DCBWKPT1	4	BISAM: pointer to CP 1 or CP 2. QISAM: pointer to DCB work area.
232(E8)	DCBWKPT2	4	BISAM: pointer to DCB work area.
236(EC)	DCBWKPT3	4	BISAM: pointer to CP 8.
240(F0)	DCBWKPT4	4	BISAM: pointer to appendage module (part 1). QISAM: pointer to UCB.

Note: This field is not used by ISAM routines. See the extension of the data extent block (DEB).

Offset	Field Name	Bytes	Field Description
244(F4)	DCBWKPT5	4	BISAM: pointer to appendage module (part 2). QISAM: pointer to appendage module.

Note: This field is not used by ISAM routines. See the extension of the data extent block (DEB).

Offset	Field Name	Bytes	Field Description
248(F8)	DCBWKPT6	4	QISAM: pointer to DCB work area vector pointers (ISLVPTRS).

Data Event Control Block (DECB)

The data event control block is constructed as part of the expansion of a READ or WRITE macro instruction. The DECB contains a parameter list, an event control block, a pointer to the desired logical record, and an exception code. Figure 59 shows the format of the DECB.

← 4 bytes →

0(0) DECBECB		
4(4) DECBTYP1	5(5) DECBTYP2	6(6) DECBLGTH
8(8) DECBDCBA		
12(C) DECBAREA		
16(10) DECBLOGR		
20(14) DECBKEY		
24(18) DECBEXC1	25(19) DECBEXC2	

Figure 59. Data Event Control Block

Offset	Field Name	Bytes	Field Description
0(0)	DECBECB	4	Standard ECB
4(4)	DECBTYP1	1	First byte of macro type field Bit 0–5 – Reserved 6 – Length coded as 'S' (take length from DCBBLKSI) 7 – Area coded as 'S' (dynamic buffer option)
5(5)	DECBTYP2	1	Second byte of macro type Bit 0 – READ K 1 – Reserved 2 – READ KU 3 – Reserved 4 – WRITE K 5 – WRITE KN 6–7 – Reserved
6(6)	DECBLGTH	2	Number of bytes read or written
8(8)	DECBDCBA	4	Data control block address

Offset	Field Name	Bytes	Field Description
12(C)	DECBAREA	4	Address of storage area for record
16(0)	DECBLOGR	4	Pointer to logical record
20(14)	DECBKEY	4	Record key address
24(18)	DECBEXC1	1	Exceptional condition code byte (see "Appendix B: ISAM Channel Programs") Bit 0 – Record not found 1 – Record length check 2 – Space not found in which to add a record 3 – Invalid request 4 – Uncorrectable I/O error 5 – Unreachable block 6 – Overflow record 7 – Duplicate record presented for inclusion in data set
25(19)	DECBEXC2	1	Exceptional condition code byte (see "Appendix B: ISAM Channel Programs") Bit 0–5 – Reserved 6 – Channel program initiated by an asynchronous routine (variable–length records only) 7 – Previous macro was READ KU

Data Set Control Block (DSCB)

Data sets on direct–access devices use a control block called a data set control block (DSCB) as their data set label. There are actually three kinds of DSCBs used to describe the attributes and extents of an ISAM data set. The information in the attribute fields of the DSCBs includes data set organization, record format, and other information needed to refer to and use a data set. The extent entries in the DSCBs describe the physical boundaries of a data set.

The three kinds of DSCBs used to describe ISAM data sets are:

- The identifier (format–1) DSCB contains such items as the data set name, the number of extents on the volume, creation and expiration dates, block length, logical record length, and three extent entries that are used to build the DEB. There is one format–1 DSCB for each volume of a data set. (*OS DADSM Logic,* GY28–6607, provides additional details on the construction of the DSCBs at allocation of the data set.)
- The index (format–2) DSCB is used only for ISAM data sets. There is one format–2 DSCB for each data set; it is used in constructing the ISAM DCB interface. The format–2 DSCB resides in the VTOC of the first volume on which the data set was allocated. When the QISAM scan mode open executor module (IGG01928) or the BISAM open executor module (IGG0192H) is executed, data

in the associated format–2 DSCB are moved to the BISAM/QISAM interface portion of the DCB. The DCB field corresponding to each DSCB field is shown in the following detailed description of the format–2 DSCB. The format–2 DSCB is shown in Figure 60.

- The extension (format–3) DSCB is required on each volume of a data set that contains more than three extents. It contains as many as 13 additional extent entries, permitting a maximum of 16 extent entries per volume.

Detailed descriptions of DSCBs are given in *OS System Control Blocks,* GC28–6628.

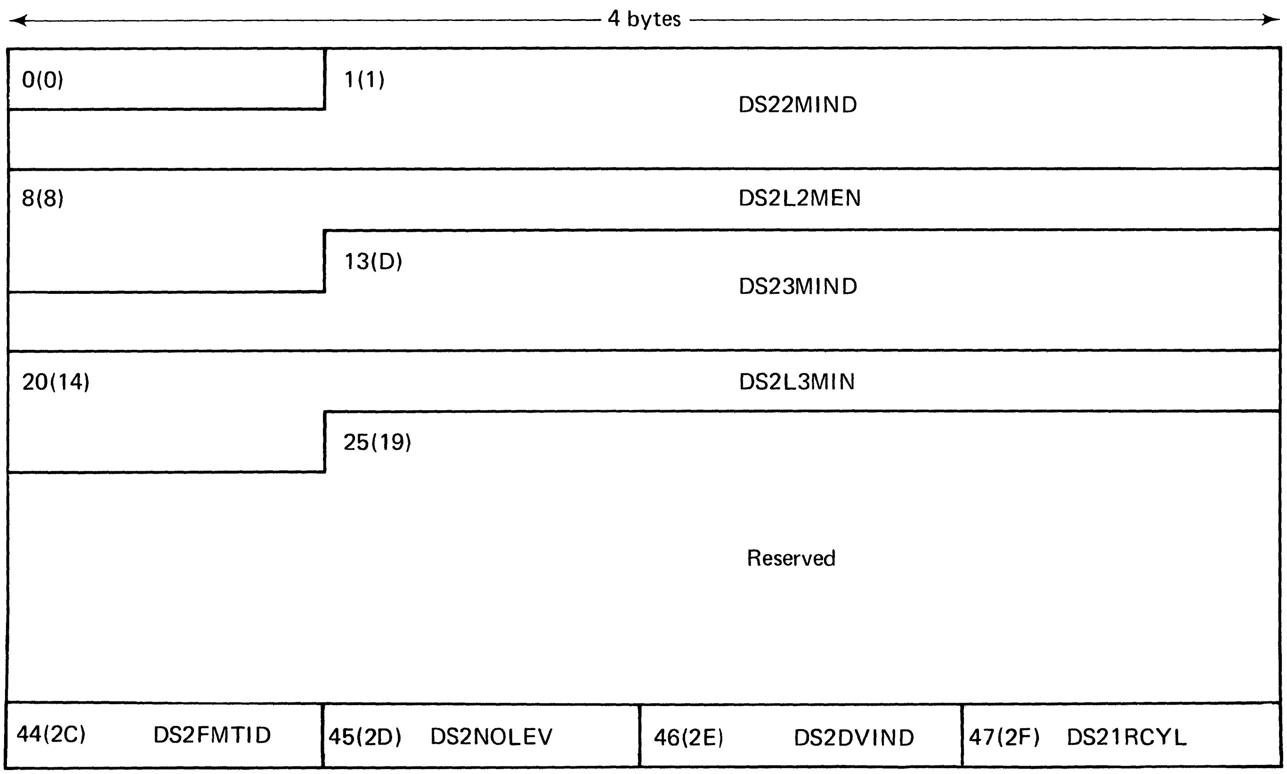


Figure 60 (Part 1 of 2). Format–2 DSCB

(Continued)

DS21RCYL (cont.)		50(32) DS2LTCYL	
52(34) DS2CYLOV	53(35) DS2HIRIN	54(36) DS2HiRPR	55(37) DS2HIROV
56(38) DS2RSHTR	57(39) DS2HIRTI	58(3A) DS2HIIOV	59(3B) DS2TAGDT
DS2TAGDT (cont.)	61(3D) DS2RORG3		
64(40) DS2NOBYT		66(42) DS2NOTRK	67(43) DS2PRCTR
DS2PRCTR (cont.)			71(47) DS2STIND
72(48) DS2CYLAD			
			79(4F)
DS2ADLIN			
		86(56)	
DS2ADHIN			
	93(5D) DS2LPRAD		
	101(65) DS2LTRAD		
DS2LTRAD (cont.)		106(6A)	
DS2LCYAD			111(6F)
DS2LMSAD			
116(74) DS2LOVAD			
124(7C) DS2BYOVL		126(7E) DS2RORG2	
128(80) DS2OVRCT		130(82) DS2RORG1	
132(84) DS2NIRT			135(87)
DS2PTRDS			

Figure 60 (Part 2 of 2). Format–2 DSCB

Offset	Field Name	Bytes	Field Description	DCB Field to Which Moved
0(0)		1	Contains hexadecimal code 02 in order to avoid conflict with a data set name.	
1(1)	DS22MIND	7	Address of the first track of the second–level master index (in the form MBBCCHH).	DCBFTMI2
8(8)	DS2L2MEN	5	Contains the CCHHR of the last active index entry in the second–level master index.	DCBLEMI2
13(D)	DS23MIND	7	Address of the first track of the third–level master index (in the form MBBCCHH).	DCBFTM3
20(14)	DS2L3MIN	5	Contains the CCHHR of the last active index entry in the third–level master index.	DCBLIMI3
25(19)		19	Reserved.	
44(2C)	DS2FMTID	1	Format identification for format–2 DSCB (EBCDIC "2").	
45(2D)	DS2NOLEV	1	Number of index levels.	DCBNLEV
46(2E)	DS2DVIND	1	Number of tracks determining development of the master index.	DCBNTM
47(2F)	DS21RCYL	3	Contains the HHR of the first data record on each cylinder.	DCBFIRSH
50(32)	DS2LTCYL	2	Contains the HH of the last data track on each cylinder.	DCBLDT
52(34)	DS2CYLOV	1	Number of tracks of cylinder overflow area on each cylinder.	DCBCYLOF
53(35)	DS2HIRIN	1	Highest possible R on a track containing high–level index entries.	DCBHIRCM
54(36)	DS2HIRPR	1	Highest possible R on prime–data tracks for format–F records.	DCBHIRPD

Offset	Field Name	Bytes	Field Description	DCB Field to Which Moved
55(37)	DS2HIROV	1	Highest possible R on overflow data tracks for format–F records.	DCBHIROV
56(38)	DS2RSHTR	1	Contains the R of the last data record on a shared track.	DCBHIRSH
57(39)	DS2HIRTI	1	Highest number of index entries that fit on a prime–data track.	DCBHIRTI
58(3A)	DS2HIIOV	1	Highest R for independent overflow track.	DCBHIIOV
59(3B)	DS2TAGDT	2	The number of records that have been tagged for deletion. This field is updated by the user during BISAM, scan mode, and load mode resume loading.	DCBTCD
61(3D)	DS2RORG3	3	The number of random references to overflow records other than the first overflow record in a chain.	DCBRORG3
64(40)	DS2NOBYT	2	The number of bytes needed to hold the highest–level index in main storage.	DCBNCRHI
66(42)	DS2NOTRK	1	The number of tracks occupied by the highest–level index.	DCBNTHI
67(43)	DS2PRCTR	4	The number of records in the prime–data area.	DCBNREC
71(47)	DS2STIND	1	Status indicators.	DCBST

Bits	Bit Setting	Meaning
0	0	Reserved
1	1	Key sequence to be checked
2	1	Initial load has been completed
3–5	1	Reserved (must remain zero)
6	1	Last block full
7	1	Last track full

Offset	Field Name	Bytes	Field Description	DCB Field to Which Moved
72(48)	DS2CYLAD	7	Address of the first track of the cylinder index (in the form MBBCCHH).	DCBFTCI

Offset	Field Name	Bytes	Field Description	DCB Field to Which Moved
79(4F)	DS2ADLIN	7	Address of the first track of the lowest–level master index (in the form MBBCCHH).	DCBFTMI1
86(56)	DS2ADHIN	7	Address of the first track of the highest–level master index (in the form MBBCCHH).	DCBFTHI
93(5D)	DS2LPRAD	8	Address of the last record in the prime–data area (in the form MBBCCHHR).	DCBLPDA
101(65)	DS2LTRAD	5	Contains the CCHHR of the last normal entry in the track index on the last cylinder.	DCBLETI
106(6A)	DS2LCYAD	5	Contains the CCHHR of the last index entry in the cylinder index.	DCBLECI
111(6F)	DS2LMSAD	5	Contains the CCHHR of the last index entry in the master index.	DCBLEMI1
116(74)	DS2LOVAD	8	Address of the last record written in the current independent overflow area (in the form MBBCCHHR).	DCBLIOV
124(7C)	DS2BYOVL	2	The number of bytes remaining on the current independent overflow track.	DCBNBOV
126(7E)	DS2RORG2	2	The number of tracks remaining in the independent overflow area.	DCBRORG2
128(80)	DS2OVRCT	2	The number of records in the overflow area.	DCBNOREC
130(82)	DS2RORG1	2	The number of cylinder overflow areas that are full.	DCBRORG1
132(84)	DS2NIRT	3	HHR of the dummy track–index entry.	
135(87)	DS2PTRDS	5	If there are more than 3 extent segments for the data set on this volume, this field contains the address of a format–3 DSCB (in the form CCHHR). Otherwise, this field contains binary 0s.	

ISAM-dependent Section (Occurs only once)

32(20)	DEBNIEE	33(21)	DEBFIEAD
36(24)	DEBNPEE	37(25)	DEBFPEAD
40(28)	DEBNOEE	41(29)	DEBFOEAD
44(2C)	DEBRPSID	45(2D)	DEBEXPTR

Device-dependent Section (Occurs once for each extent)

+0(0) DEBDVMOD		+1(1) DEBUCBAD	
+4(4)	DEBBINUM	+6(6)	DEBSTRCC
+8(8)	DEBSTRHH	+10(A)	DEBENDCC
+12(C)	DEBENDHH	+14(E)	DEBNMTRK

+0	DEBSUBID

Subroutine Name Section (Occurs once for each subroutine)

ISAM Extension

Load Mode Extension

+0(0)	DEBPUT

Scan Mode Extension

+0(0)	DEBGET, DEBPUT	+4(4)	DEBWKPT4
+8(8)	DEBWKPT5	+12(C)	DEBCREAD
+16(10)	DEBCSETL	+20(14)	DEBCWRIT
+24(18)	DEBCCHK	+28(1C)	DEBCREWT
+32(20)	DEBCRECK	+36(24)	DEBAREAD
+40(28)	DEBASETL	+44(2C)	DEBAWRIT
+48(30)	DEBACHK	+52(34)	DEBAREWT
+56(38)	DEBARECK		

BISAM Extension

+0(0)	DEBDISAD	+4(4)	DEBWKPT4
+8(8)	DEBWKPT5	+12(C)	DEBFREED
+16(10)	DEBRPSIO		

Figure 61. ISAM Extensions to DEB

Data Extent Block (DEB)

The ISAM open executors construct the data extent block (DEB). The DEB contains the extents of the opened data set, pointers to the unit control blocks (UCBs) for the extents, and the names of access method routines to be used. The ISAM–dependent, device–dependent, and subroutine name sections of the DEB are shown in Figure 61.

ISAM–DEPENDENT SECTION

Offset	Field Name	Bytes	Field Description
32(20)	DEBNIEE	1	Number of extents of independent index area
33(21)	DEBFIEAD	3	Address of first index extent
36(24)	DEBNPEE	1	Number of extents of prime–data area
37(25)	DEBFPEAD	3	Address of the first prime–data extent
40(28)	DEBNOEE	1	Number of extents of independent overflow area
41(29)	DEBFOEAD	3	Address of the first overflow extent
44(2C)	DEBRPSID	1	Identifiers for prime, index, or overflow areas on an RPS direct–access storage device. Bits Meaning 0 Prime area is on an RPS device. 1 Index area is on an RPS device. 2 Overflow area is on an RPS device. 3 An SIO appendage for RPS has been loaded. (This bit set by IGG0192K.) 4–7 Reserved.
45(2D)	DEBEXPTR	3	Address of ISAM DEB extension. The device–dependent sections (one for each extent) are in the following order: prime extents, index extents, overflow extents.

DEVICE–DEPENDENT SECTION

Offset	Field Name	Bytes	Field Description
+0(0)	DEBDVMOD	1	Device modifier: file mask.

Offset	Field Name	Bytes	Field Description
+1(1)	DEBUCBAD	3	Address of UCB associated with this data extent.
+4(4)	DEBBINUM	2	Bin number if the device is a 2321 data cell drive, 0 for other devices.
+6(6)	DEBSTRCC	2	Cylinder address for the start of an extent limit.
+8(8)	DEBSTRHH	2	Read/write track address for the start of an extent limit.
+10(A)	DEBENDCC	2	Cylinder address for the end of an extent limit.
+12(C)	DEBENDHH	2	Read/write track address for the end of an extent limit.
+14(E)	DEBNMTRK	2	Number of tracks allocated to a given extent.

SUBROUTINE NAME SECTION

	DEBSUBID	2n	Subroutine identification. Each access method subroutine, appendage subroutine, and IRB routine has a unique 8–byte name. The low–order two bytes of each routine name are in this field if the subroutine is loaded by the open routine.

ISAM EXTENSION

Load Mode Extension

+0(0)	DEBPUT	4	Address of the PUT processing module

Scan Mode Extension

+0(0)	DEBGET, DEBPUT	4	Address of the Get processing module
+4(4)	DEBWKPT4	4	Address of the UCB
+8(8)	DEBWKPT5	4	Pointer to the Get appendage module
+12(C)	DEBCREAD	4	Address of channel–end appendage for Read
+16(10)	DEBCSETL	4	Address of channel–end appendage for SETL
+20(14)	DEBCWRIT	4	Address of the channel–end appendage for Write
+24(18)	DEBCCHK	4	Address of the channel–end appendage for Write–validity–check

Offset	Field Name	Bytes	Field Description
+28(1C)	DEBCREWT	4	Address of the channel–end appendage for Rewrite
+32(20)	DEBCRECK	4	Address of the channel–end appendage for Recheck
+36(24)	DEBAREAD	4	Address of the abnormal–end appendage for Read
+40(28)	DEBASETL	4	Address of the abnormal–end appendage for SETL
+44(2C)	DEBAWRIT	4	Address of the abnormal–end appendage for Write
+48(30)	DEBACHK	4	Address of the abnormal–end appendage for Write–validity–check
+52(34)	DEBAREWT	4	Address of the abnormal–end appendage for Rewrite
+56(38)	DEBARECK	4	Address of the abnormal–end appendage for Recheck

BISAM Extension

Offset	Field Name	Bytes	Field Description
+0(0)	DEBDISAD	4	Address of the privileged module entered when a BISAM macro instruction is executed.
+4(4)	DEBWKPT4	4	Address of the Part 1 appendage module (abnormal–and channel–end appendages).
+8(8)	DEBWKPT5	4	Address of the Part 2 appendage module (abnormal–and channel–end appendages).
+12(C)	DEBFREED	4	Address of the dynamic buffering module.
+16(10)	DEBRPSIO	4	Address of the RPS SIO appendage module if dynamic buffering is used. (If dynamic buffering is not used, the appendage vector table of the DEB contains the address of the RPS SIO appendage module.)

Input/Output Block (IOB)

The input/output block (IOB) contains information required by the I/O supervisor to perform an input/output operation. The ISAM routine constructs an IOB for each such operation.

The IOB consists of 40 bytes of standard information as described in *OS System Control Blocks,* GC28-6628. The standard information is common to all access methods. BISAM and QISAM (scan mode) use extensions of the standard IOB, and QISAM uses an IOB prefix. The ISAM extensions and prefix are shown in Figure 62.

QISAM Prefix

-4(-4)	Event Control Block

BISAM Extension

40(28)	IOBCCWAD						
44(2C)	IOBINDCT	45(2D)	IOBUNSQR	46(2E)	IOBAPP	47(2F)	IOBASYN
48(30)	IOBCOUNT	49(31)	IOBFCHAD				
52(34)	IOBBCHAD						
56(38)	IOBCCW1						
64(40)	IOBCCW2						

QISAM Extension (scan mode)

40(28)	Q1IEXTEN-W1OEXTEN	

Figure 62. ISAM Extensions to IOB

Offset	Field Name	Bytes	Field Description
	QISAM Prefix		
–4(–4)		4	Event control block.
	BISAM Extension		
40(28)	IOBCCWAD	4	Address of first CCW of channel program or address of buffer after completion of a READ KU (BISAM dynamic buffering).

Offset	Field Name	Bytes	Field Description
44(2C)	IOBINDCT	1	Indicators.
			Bit / Bit Setting / Meaning
			0 / 1 / Remove channel program from queue.
			1 / 1 / IOB is on the unscheduled queue.
			2 / 0 / DECBAREA (+6) points to overflow record data; DCBMSWA points to the key and data of an overflow record.
			3 / 0 / DECBKEY points to overflow record key.
			1 / DCBMSWA (+8) points to overflow record key.
			4–6 / 0 / Reserved.
			7 / 0 / Normal channel end has occurred.
			1 / Abnormal channel end has occurred.
45(2D)	IOBUNSQR	1	Reason for unscheduled or error queue.
			Bit / Bit Setting / Meaning
			0 / 1 / CP 1 or CP 2 busy.
			1 / 1 / No CP 4, CP 5, or CP 6.
			2 / 1 / No CP 7.
			3 / 1 / WRITE KN is in effect (unscheduled IOB is for WRITE KN).
			4 / 1 / WRITE KN is in effect (unscheduled IOB is for READ or WRITE K).
			5 / 1 / An error condition is associated with this IOB.
			6–7 / 0 / Reserved.
46(2E)	IOBAPP	1	Appendage code (see "Section 6: Diagnostic Aids").
47(2F)	IOBASYN	1	Asynchronous routine code (see "Section 6: Diagnostic Aids").
48(30)	IOBCOUNT	1	Write–check counter.
49(31)	IOBFCHAD	3	Forward chain address.
52(34)	IOBBCHAD	4	Backward chain address.
56(38)	IOBCCW1	8	Set sector CCW for use with RPS direct–access storage devices.
64(40)	IOBCCW2	8	TIC CCW to the channel program, used with RPS devices.

Offset	Field Name	Bytes	Field Description
	QISAM Extension (scan mode)		
40(32)	Q1IEXTEN W1OEXTEN	2	Appendage codes (see "Section 6: Diagnostic Aids").

Buffer Control Block (BCB)—BISAM

The buffer control block (BCB) used to control dynamic buffering in BISAM is structured by the stage 2 Open executor IGG0293B if the problem program has requested dynamic buffering. If the user does not specify the number of buffers he desires, two buffers are provided. The fields of the BISAM BCB are shown schematically in Figure 63.

Offset	← 4 bytes →
0(0)	BCBFIOB
4(4)	BCBLIOB
8(8)	BCBNAVB
12(C)	BCBSIZE
16(10)	Reserved (for doubleword alignment)
20(14)	First Buffer (Link Field)[1]
24(18)	First Buffer (continued)
	Second Buffer (Link Field)
	Second Buffer (continued)
	Nth Buffer (Link Field)
	Nth Buffer (continued)

[1] The first buffer begins at 20(14) if buffer alignment specified was fullword; it begins at 24(18) if alignment was at doubleword.

Figure 63. Fields of the BISAM Dynamic Buffering Buffer Control Block

The following describes the contents and uses of the fields of the BISAM BCB.

Field Name: BCBFIOB

Offset: 0(0)

Bytes: 4

Field Description: If there are not enough buffers available for the number of READ K or READ KU requests issued, the dynamic buffering routine, entered from the start I/O appendage routine, activates this field as a pointer to the first IOB that needs a buffer. Later, when a buffer has become available (because it was released by either the WRITE K macro instruction or the FREEDBUF macro instruction), the dynamic buffering routine, entered through one of those macro routines, updates BCBFIOB to point to the next IOB that needs a buffer. If there are no more IOBs on queue for a buffer, this field is then reset to 0. Initially, this field is set to 0 by the ISAM open module IGG0192B.

Field Name: BCBLIOB

Offset: 4(4)

Bytes: 4

Field Description: If there are not enough buffers available for the number of READ K or READ KU requests issued, the dynamic buffering routine, entered from the start I/O appendage routine, activates this field as a pointer to the last IOB that needs a buffer (the IOB of the latest Read requested). The IOB forward chain address (IOBFCHAD) of the IOB previously pointed to by this field, if BCBLIOB has been previously activated, is also set to point to this latest IOB. IOBFCHADs thus provide the linkage between BCBFIOB and BCBLIOB. BCBLIOB is initialized and reset whenever BCBFIOB is.

Field Name: BCBNAVB

Offset: 8(8)

Bytes: 4

Field Description: Points to the next buffer available to a READ K or READ KU request. Initially, BCBNAVB is set to point to the first buffer by ISAM Open module IGG0192B. The dynamic buffering routine is entered from the start I/O appendage routine to select the buffer pointed to by this field when a read is issued. The link field of the buffer selected is placed into BCBNAVB. When a buffer has been released either by a FREEDBUF macro instruction or because it has been written back into the data set, entry is made to the dynamic buffering routine. If an IOB is waiting for a buffer (see BCBFIOB), the buffer just released is assigned to the IOB, and an EXCP is issued. If, however, the IOB queue is empty, the buffer is placed on the available queue. This is accomplished by placing a pointer to the buffer in BCBNAVB after moving the contents of BCBNAVB into the link field of the buffer. When there are no buffers on the available queue, BCBNAVB contains 0.

Field Name: BCBSIZE

Offset: 12(C)

Bytes: 4

Field Description: Total main–storage size of the BCB and the attached buffers. Calculated by open module IGG0192B. Used by Close module IGG0202A to free the buffer control block and the associated buffers.

Field Name: First Buffer (Link Field)

Offset: 20(14)

Bytes: 4 (first 4 bytes of each buffer)

Field Description: If a buffer is on the available queue, its link field contains the address of the following buffer to be made available. When a buffer is the last buffer on the available queue, its link field contains 0. When a buffer is not on the available queue, these 4 bytes are used as a part of the buffer.

Buffer Control Block (BCB)—QISAM

The BCB used in QISAM differs in format from the BISAM BCB. Figure 64 pictures schematically the fields of the QISAM BCB. This BCB may result from a GETPOOL or BUILD macro instruction issued by the processing program, or it may be constructed by the stage 1 open executors. The information it contains is needed by the stage 2 open executors.

0(0) ADDRESS OF FIRST BUFFER	
4(4) NUMBER OF BUFFERS	6(6) LENGTH OF EACH BUFFER

Figure 64. Fields of the QISAM Buffer Control Block

The following is a description of the contents and uses of the fields of the QISAM BCB.

Field Name: Address of first buffer

Offset: 0(0)

Bytes: 4

Field Description: Load mode open module IGG0192G uses this address to initialize the load mode buffer control table field named IOBABUF. Scan mode open module IGG01929 uses the address (in conjunction with the link field of each buffer) to initialize its channel programs.

Field Name: Number of buffers

Offset: 4(4)

Bytes: 2

Field Description: The number of buffers in this buffer pool.

Field Name: Length of each buffer

Offset: 6(6)

Bytes: 2

Field Description: Scan mode open module IGG01929 uses this field to ensure the buffer size is adequate for the records to be retrieved.

Buffer Control Table (IOBBCT)

The buffer control table, used by QISAM load mode to control the filling of buffers, is initialized by Stage 2 Open executor module IGG0192G. The area for the IOBBCT is obtained by Stage 1 Open executor module IGG0192B. The fields of the buffer control table are shown schematically in Figure 65.

0(0)	IOBFLAGS	1(1)	IOBPTRA
4(4)	IOBB	5(5)	IOBPTRB
8(8)	IOBS (1st Buffer)	9(9)	IOBABUF (1st Buffer)
2N+10	IOBS (Nth Buffer)	2N+11	IOBABUF (Nth Buffer)

Figure 65. QISAM Load Mode Buffer Control Table

The following is a description of the contents and uses of the fields of the IOBBCT.

Field Name:	IOBFLAGS
Offset:	0(0)
Bytes:	1
Field Description:	General I/O conditions pertaining to all buffers. IOBFLAGS is initialized by open executor IGG0192G. At this time, bit 4 is set; all other bits are reset.
Bit 0:	When the end–of–buffer routine schedules an EXCP to use CP 18/CP 20 (to write data records and the associated track indexes), the bit is set on to indicate CP 18/CP 20 are busy.

The CP 18/CP 20 appendage routine resets the bit.

Bit 1:	When the end–of–buffer routine cannot schedule the EXCP because CP 18/CP 20 are busy (bit 0 = 1), this bit is set. It is interrogated after every PUT macro instruction and, if set, another attempt is made to schedule the EXCP. If the attempt is successful, the bit is reset.
Bit 2:	When bit 1 = 1 and an attempt is being made to write previously filled buffers, but the current buffer is not full, this bit must be set to tell the end–of–buffer routine, which schedules the EXCP, to return to the Put routine.
Bit 3:	This bit is set by close executor module IGG0202I. It ensures return to closing routines after using channel programs to complete processing of the final buffers.
Bit 4:	This bit is set by the Put routine (in move mode only) when the last record PUT filled a buffer. It is interrogated by the Put routine to determine if a new buffer must be initialized before moving the current record and is reset by the beginning–of–buffer routine after the new buffer has been intialized.
Bit 5:	When the Put routine determines that there is enough space on the current track–index track for only one more normal and overflow track–index entry, it sets this bit. Prior to this determination, it has reset this bit. If the Put routine determines that an end–of–cylinder condition exists, it interrogates the bit to see if the extra track–index dummy entry will fit on the current track (bit 5 = 0), or whether a new track is needed (bit 5 = 1).
Bit 6:	This bit is set by close executor module IGG0202I. It ensures return to closing routines after completing the data set's high–level index.

Bit 7:	Set by open executor module IGG0192R (or IGG0192U) if the data set consists of unblocked records whose relative key position (RKP) is 0. The bit is interrogated during initialization of CP 18.

Field Name:	IOBPTRA
Offset:	1(1)
Bytes:	3
Field Description:	This field serves as a pointer to the address of the first buffer of the group that is written next. During the execution of CP 18, it points to the address of the first buffer of the group currently being written. When CP 18 is completed, the appendage routine updates this field to point to the address of the first buffer of the next group. IOBPTRA is needed to initialize CP 18 before CP 18 is executed. IOBPTRA is initialized by open executor module IGG0192G to point to the address of the first buffer.

Field Name:	IOBB
Offset:	4(4)
Bytes:	1
Field Description:	IOBB contains the number of buffers filled but not yet scheduled for writing. It is updated by the Put routine as each buffer is filled and reset to 0 by the end–of–buffer routine when the buffers are scheduled for writing. IOBB is initialized to 0 by open executor module IGG0192G.

Field Name:	IOBPTRB
Offset:	5(5)
Bytes:	3
Field Description:	This field serves as a pointer to the address of the buffer currently being filled. It is updated when the beginning–of–buffer routine is called to prepare a new buffer before executing a PUT command. IOBPTRB is initialized by open executor module IGG0192G to point to the address of the first buffer.

Field Name:	IOBS
Offset:	2n+10 where n is the buffer number.
Bytes:	1

Field Description:	There is one status byte (IOBS) for each buffer. The bits are used to indicate conditions peculiar to each buffer. All status bits (except bit 0) are initially reset by open executor module IGG0192G.
Bit 0:	Set (by open executor module IGG0192G) if this is IOBS field for buffer N (last buffer); otherwise reset. Interrogated to ensure proper sequence of buffering when going from last to first buffer.
Bits 1 and 2:	A 2–bit code indicating buffer availability as follows: 00 — buffer available — set by CP 18/CP 20 appendage routine after writing; interrogated by beginning–of–buffer routine prior to using buffer again. 01 — contents of buffer caused permanent write error — set by CP 18/CP 20 appendage routine; interrogated by beginning–of–buffer routine prior to using buffer again. 10 — buffer full, but not yet scheduled for writing — set by Put routine when buffer becomes full; prevents refilling of buffer before writing. 11 — buffer scheduled for writing — set by end–of–buffer routine when scheduled; interrogated by appendage routine to reset these bits and to update IOBPTRA.
Bit 3:	This bit is set by the beginning–of–buffer routine when it determines that this buffer, when written, will begin a new extent. Interrogated, then reset, by end–of–buffer routine before scheduling writing of this buffer in the new extent.
Bit 4:	This bit (the T–Bit) is set by the beginning–of–buffer routine when it determines that this buffer will be the last written on a track. Interrogated by end–of–buffer routine so that CP 20 is executed to write the track index. The T–Bit is reset by the CP 18/CP 20 appendage routine.
Bit 5:	This bit (the C–Bit) is set by the beginning–of–buffer routine when it determines that this buffer, in addition to being the last written on a track, is also the last written on a cylinder. Interrogated by end–of–buffer routine so that CP 21 is executed to write the cylinder index when necessary. The C–Bit is reset by the CP 21 appendage routine.
Bit 6:	This bit (the PF–Bit) is set by the beginning–of–buffer routine when it determines that this buffer is the first buffer written on a cylinder and track–sharing is in effect. CP 19 is used to preformat the shared track. The end–ofbuffer–routine interrogates this bit and does not schedule a write on the new cylinder until the CP 19 appendage routine has reset the bit.

Bit 7: Not used.

Field Name: IOBABUF

Offset: 2n+11 where n is the buffer number.

Bytes: 3

Field Description: There is one IOBABUF field for each buffer, and it contains the address of its associated buffer. Stage 1 open executor module IGG0192B provides the address of the first buffer (through DCBBUFCB) and Stage 2 open executor module IGGG0192G uses the buffer link field of each buffer to fill out the remaining IOBABUFs. (When buffers are structured, the first four bytes of each buffer — the buffer link field — contain the address of the next buffer in the chain. After these addresses are put into the IOBBCT, these four bytes become part of the buffer.) Buffer addresses are used for initialization of CP 18 and provide the storage location into which records are to be moved.

QISAM Load Mode DCB Work Area

The QISAM load mode DCB work area is pointed to by the DCBWKPT1 field of the DCB. The DCB work area format is shown in Figure 66.

Offset	Field Name	Bytes	Field Description
0(0)	ISLECBA	4	The ECB for CP 18 and CP 20.
4(4)	ISLIOBA	40	The IOB for CP 18 and CP 20.
44(2C)	ISLECBB	4	The ECB for CP 21.
48(30)	ISLIOBB	40	The IOB for CP 21.
88(58)	ISLECBC	4	The ECB for CP 19 and CP 91.
92(5C)	ISLIOBC	40	The IOB for CP 19 and CP 91.
132(84)	ISLAREAZ	88	This area contains the data field for cylinder overflow records and the count field for ten index entries. These are used to preformat shared tracks during the Put load mode function and to pad dummy track indexes on unused cylinders during the Close routine.

Area Z appears as follows:

CYL.OVL. CTRL.RCD. HHRYYT	COUNT 1	COUNT 2		COUNT 10
Z	Z+6(6)	Z+14(E)		Z+78(4E)

8 bytes

0(0)	ISLECBA	4(4)	ISLIOBA
		44(2C)	ISLECBB
48(30)	ISLIOBB		
88(58)	ISLECBC	92(5C)	ISLIOBC
		132(84)	ISLAREAZ
		220(DC)	ISLIXLT
		324(144) ISLNIRT	327(147) ISLHIRT
328(148)	ISLCBF	332(14C)	ISLBMPR
336(150)	ISLFBW	340(154)	ISLEOB
344(158)	ISLNCNT		
352(160)	ISLOCNT		

Figure 66 (Part 1 of 2). QISAM Load Mode DCB Work Area

(Continued)

360(168)	ISLDCNT		
368(170)	ISLNDAT		
378(17A)	Reserved	380(17C)	ISLODAT
290(176)	Reserved	391(187)	ISLBUFNO
392(188)	ISLBUFN	396(18C)	ISLMVC
400(190)	ISLMVCT	404(194)	ISLVRSAV
476(1DC)	ISLAPSAV		
516(204)	ISLWRSAV		
		580(244)	TSTWK1C
584(248)	TSTWK2C	588(24C)	Reserved
592(250)	ISLNOENT	596(254)	ISLOFFST
600(258)	ISLD	604(25C)	ISLFSTBF
608(260)	ISLLSTBF	612(264)	ISLCCFAD
616(268)	ISLKEYAD	620(26C)	CL1AD/ISLF8AD
624(270)	CM1AD/ISLFXAD	628(274)	CQ1AD/ISLFYAD
632(278)	CQT1AD/ISLFZAD	636(27C)	CQ40AD/ISLPAAD
640(280)	CQ45AD/ISLF1AD	644(284)	ISLVPTRS (pointed to by DCBWKPT6)
704(2C0) ISLIGAP	706(2C2) ISLLGAP	708(2C4)	ISLRPSSS

Variable-length areas follow:
Pointed to by 1SLVPTRS
Area Y (See Figure 67)
Key save area
Buffer control table
Channel programs

Figure 66 (Part 2 of 2). QISAM Load Mode DCB Work Area

Offset	Field Name	Bytes	Field Description
220(DC)	ISLIXLT	104	The index location table contains the direct–access device addresses for high–level indexes.

IND.	BEGIN	STEPPING	END	
0(0)	MBBCCHHR	MBBCCHHR	MBBCCHHR	CYL
26(1A)	MBBCCHHR	MBBCCHHR	MBBCCHHR	M1
52(34)	MBBCCHHR	MBBCCHHR	MBBCCHHR	M2
78(4E)	MBBCCHHR	MBBCCHHR	MBBCCHHR	M3

There is an indicator byte and three device addresses for each level of index; cylinder, and up to three master index levels.

The begin and end addresses are set during the Open routine according to formulas based on space allocation. The stepping addresses are used during data set creation to point to the current index entry location at each level. The indicator byte is as follows:

Bit 0 = 1 for last level
= 0 otherwise
1 = 1 for dummy switch on
= 0 for dummy switch off
2 = 1 for current level
= 0 otherwise
3 = 1 during Close
= 0 otherwise
4 = 1 when track index has been written but not cylinder index
= 0 when cylinder index has been written

Indicator bit 4 only applies to the first level of the index location table.

Offset	Field Name	Bytes	Field Description
324(144)	ISLNIRT	3	HHR of the dummy track–index entry. It is used in Close to signal the end–of–track index padding.
327(147)	ISLHIRT	1	The number of index entries that fit on a prime–data track.
328(148)	ISLCBF	4	Buffer control pointer. This field contains the address of the current record in the current buffer. It is used to move records into a buffer.
332(14C)	ISLBMPR	4	Size of individual records (equal DCBLRECL or DCBLRECL + DCBKEYLE). This field is used to bump ISLCBF to next record location in a buffer.

Offset	Field Name	Bytes	Field Description
336(150)	ISLFBW	4	The number of buffers scheduled to be written. This number is determined immediately following each execution of CP 18. It is the number of buffers (DCBBUFNO) minus one, or the number of buffers that completes a track, whichever is smaller.
340(154)	ISLEOB	4	End–of–buffer address. When ISLCBF and ISLEOB are equal, a buffer has been filled.
344(158)	ISLNCNT	8	CCHHRKDD. This is the count field for the current normal track–index entry.
352(160)	ISLOCNT	8	CCHHRKDD. This is the count field for the current overflow track–index entry.
360(168)	ISLDCNT	8	CCHHRKDD. This is the count field for the current dummy track–index entry.
368(170)	ISLNDAT	10	MBBCCHHRFP. This is the data field for the current normal track–index entry.
378(17A)		2	Reserved.
380(17C)	ISLODAT	10	MBBCCHHRFP. This is the data field for the current overflow track–index entry.
390(186)		1	Reserved.
391(187)	ISLBUFNO	1	Number of buffers. ISLBUFNO equals DCBBUFNO.
392(188)	ISLBUFN	4	Address of Slot N in buffer control table.
396(18C)	ISLMVC	4	The count used for the Executed Move at ISLFX21 when moving a record from the user's work area into a buffer. This count equals R–1 where R is the remainder when dividing ISLBMPR by 255. If R=0, ISLMVC is set decreased (see ISLMVCT).
400(190)	ISLMVCT	4	The count used for the BCT at ISLFX21 when moving a record from the user's work area into a buffer. This is the number of 255–byte moves, plus one, needed to move the record. This count equals Q+1 where Q is the quotient when dividing ISLBMPR by 255. When R, alone, equals 0, ISLMVCT is set to equal Q.
404(194)	ISLVRSAV	72	Index register save area. This area is used during load mode macro time to save index registers within load mode.
476(1DC)	ISLAPSAV	40	Index register save area. This area is used during load mode appendage time to save index registers belonging to either the I/O supervisor or load mode Close.
516(204)	ISLWRSAV	64	Index register save area. This area is used during load mode Close to save index registers belonging to common Close.

Offset	Field Name	Bytes	Field Description
580(244)	TSTWK1C	4	Open work field.
584(248)	TSTWK2C	4	Open work field.
588(24C)		4	Reserved.
592(250)	ISLNOENT	4	Number of spaces for track–index entries remaining on the current track–index track.
596(254)	ISLOFFST	4	Size of WRITE channel commands in CP 18. If unblocked records, RKP=0, ISLOFFST=8. Otherwise, ISLOFFST=24.
600(258)	ISLD	4	At Macro Time: ISLD is the displacement from the start of CP 18 to the CC flag in the first WRITE CCW in the chain. If unblocked records, RKP=0, ISLD=28. Otherwise, ISLD=44. (ISLOFFST+20) During Close: ISLD is a set of switches used when padding indexes: Bit 0 = 1 for new cylinder; 0 otherwise 1 = 1 for end entry; 0 otherwise 2 = 1 for chained entry; 0 otherwise
604(250)	ISLFSTBF	4	Pointer to first buffer scheduled for writing. This is the slot number in the buffer control table associated with the first buffer to be written in the current output chain.
608(260)	ISLLSTBF	4	Pointer to last buffer scheduled for writing. This is the slot number in the buffer control table associated with the last buffer to be written in the current output chain.
612(264)	ISLCCFAD	4	Address of CC flag in the last WR CKD CCW in CP 18 chain. This CC flag is turned off to stop the write chain.
616(268)	ISLKEYAD	4	Address of the key in the last record that is placed on the current prime–data track. This key becomes the track–index key for the given track.
620(26C)	CL1AD ISLF8AD	4	Address of the CP 18 skeleton (Open). Address of instruction at ISLF800+6=PUT base (Close).
624(270)	CM1AD ISLFXAD	4	Address of the CP 19 skeleton (Open). Address of the instruction at ISLFY01 (Close).
628(274)	CQ1AD ISLFYAD	4	Address of the CP 20 skeleton (Open). Address of the instruction at ISLFY01 (Close).
632(278)	CQT1AD ISLFZAD	4	Address of CP 20 write–check extension skeleton (Open). Address of the instruction at ISLFZ01 (Close).
636(27C)	CQ40AD ISLPAAD	4	Address of the CP 21 skeleton (Open). Address of the instruction at ISLPA01 (Close).

Offset	Field Name	Bytes	Field Description
640(280)	CQ45AD ISLF1AD	4	Address of CP 21 write–check extension skeleton (Open). Address of the instruction at ISLF110 (Close).
644(284)	ISLVPTRS	60	Address of variable–length areas and channel programs. 0(0) — A(Area Y) (Figure 67) + 4(4) — A(Key save) + 8(8) — A(IOBBCT) + 12(C) — A(CP 18) + 16(10) — A(CP 19) + 20(14) — A(CP 20A or 0s) — full track–index write option + 24(18) — A (CP 21) + 28(1C) — Size of DCB work area — ISLCOMON (for FREEMAIN in Close) + 32(20) — Size of channel program area for FREEMAIN + 36(24) — A (TISA) Bit 0 — full track–index write Bit 1 — successful GETMAIN + 40(28) — A (CP 31A/31B) — resume load A (CP 20B or 0s) — full track–index write option + 44(2C) — A (CP 20C or 0s) — full track–index write option + 48(30) — ISLFXWK1 (macro work field) + 52(34) — ISLFXWK2 (macro work field) + 56(38) — ISLF9WK1 (work field) **Note:** When there is a permanent I/O error, ISLVPTRS+36 is overlaid with the address of the buffer that caused the error if CP 18 failed; otherwise, it is set to 0. ISLVPTRS+40 is overlaid with the SYNAD address and ISLVPTRS+44 is overlaid with the second word of the IOB.
704(2C0)	ISLIGAP	2	Overhead (record gap) for other than the last record. Used in RPS device space allocation calculations for VLR track capacity of prime–data records.
706(2C2)	ISLLGAP	2	Last record overhead for RPS devices. Used to calculate VLR track capacity of prime–data records.
708(2C4)	ISLRPSSS	4	Sectors values used in CP 18, CP 19, CP 20, and CP 21 for RPS devices.

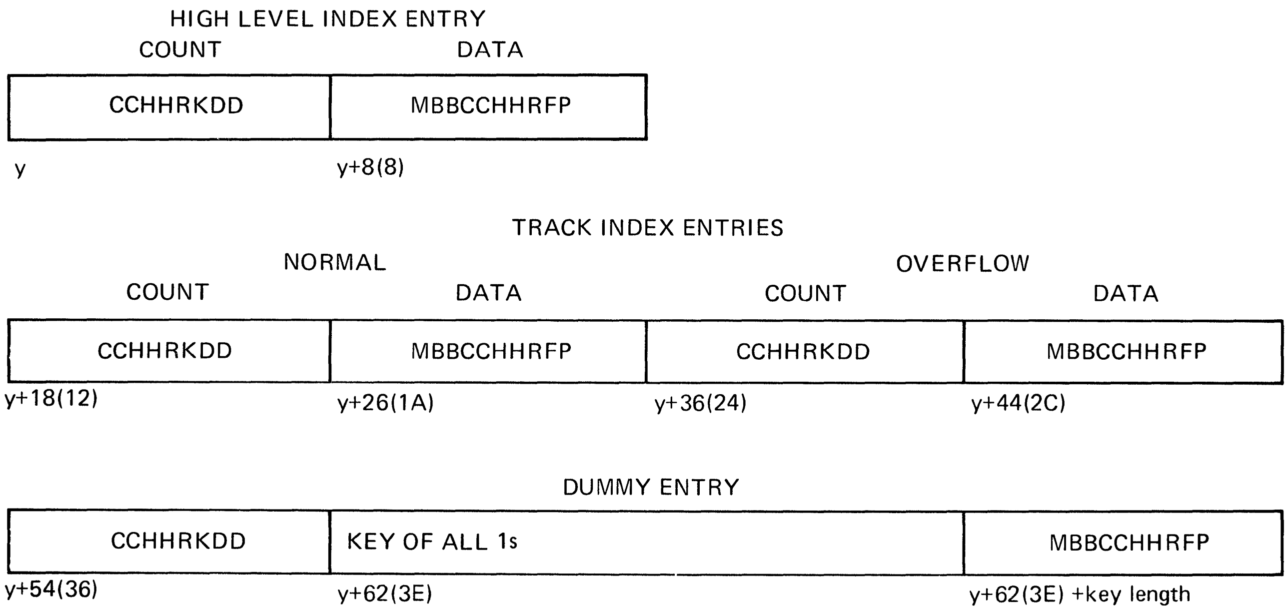


Figure 67. Area Y: QISAM Load Index Fields

QISAM Scan Mode DCB Work Area

The QISAM scan mode DCB work area is pointed to by the DCBWKPT1 field of the DCB. The DCB work area format is shown in Figure 68.

8 bytes

0(0) W1ECBI				4(4) W1IOB
W1IOB				44(2C) W1IEXTEN | 46(2E) W1CPNUP
48(30) W1ECBO				52(34) W1IOBO
W1IOBO				92(5C) W1OEXTEN | 94(5E) W1SAV7
96(60) W1OSBIT1	97(61) W1OSBIT2	98(62) W1OSBIT3	99(63) W1ICNOT	100(64) W1KEYBLK
104(68) W1LPDR				
112(70) W1CBF				116(74) W1EOB
120(78) W1COUNTR		PRIMEIND	FIXIND	124(7C) W1FCPS

W1QTABLE

128(80) W1FR1ST		132(84) W1FRLAST	
136(88) Reserved	W1FREEC	140(8C) W1RDIST	
144(90) W1RDLAST		148(94) W1READR	150(96) W1READC
152(96) W1US1ST		156(9C) W1USLAST	
160(A0) Reserved	162(A2) W1USERC	164(A4) W1PX1ST	
168(A8) W1PXLAST		172(AC) Reserved	174(AE) W1PUTXC
176(B0) W1WR1ST		180(B4) W1WRLAST	
184(B8) Reserved	186(BA) W1WRITEC		

Figure 68 (Part 1 of 2). QISAM Scan Mode DCB Work Area

(Continued)
W1WAREA

		188(BC) W1WCOUNT	
W1WCOUNT (cont.)		196(C4) W1WCNXDM	
W1WCNXDM (cont.)		204(CC) W1WOVFL	
W1WOVFL (cont.)			214(D6) W1WDNXDM
W1WDNXDM (cont.)			

224(E0) W1WPLEN	226(E2) W1CURLEN	228(E4) W1TEMPSA	
232(E8) W1REGSV2		236(EC) W1REGSAV	
240(F0) W1REGSV3		244(F4) W1CP23PT	
248(F8) W1CP26PT		252(FC) W1CP25PT	
256(100) W1CP24			
324(144) W1WDCXDM			
	334(14E) W1ISECT	335(14F) W1OSECT	336(150) W1DCBFA
340(154) W1ICPEXT			
356(164) W1OCPEXT			
372(174) W1RDCNT			
380(17C) W1RDSECT			
388(184) W1CN5SAV		392(188)	
W1RPSSA			
		408(198) W1TOTAL	410(19A) W1RECLEN
412(19C) W1OVLEN	414(19E) W1FSTSH	416(1A0) W1RPSC1 / 417(1A1) W1RPSC2	418(1A2) W1RPSI1 / 419(1A3) W1RPSI2

Figure 68 (Part 2 of 2). QISAM Scan Mode DCB Work Area

Offset	Field Name	Bytes	Field Description
0(0)	W1ECBI	4	Input ECB.
4(4)	W1IOBI	44 40	Input IOB and extension. This includes: IOB.
44(2C)	W1IEXTEN	2	Input appendage code.
46(2E)	W1CPNUP	2	Save area for schedule routine.
48(30)	W1ECBO	4	Output ECB.
52(34)	W1IOBO	44 40	Output IOB and extension. This includes: IOB.
92(5C)	W1OEXTEN	2	Output appendage code. 8 — Write C — Check 10 — Rewrite 14 — Recheck
94(5E)	W1SAV7	2	Save area for schedule routine.
96(60)	W1OSBIT1	1	Overall status, byte 1. Bit 0 Scan mode 1 End of data set 2 Overflow 3 Read track index 4 Key found (for SETL K) 5 Unreachable record 6 IOBI completion 7 IOBO completion
97(61)	W1OSBIT2	1	Overall status, byte 2. Bit 0 Unwritable record 1 Work bit for write appendage 2 Same–cylinder indicator 3 Shared track 4 GET — SETL communication 5 Scheduling 6 RELSE 7 SETL K blocked

Offset	Field Name	Bytes	Field Description
98(62)	W1OSBIT3	1	Overall status, byte 3. Bit 0 Buffer size 1 CLOSE — ESETL communication 2 Bad set indicator for write–checking 3–7 Unused
99(63)	W1ICNOT	1	BUFNO/2— used to schedule input/output.
100(64)	W1KEYBLK	4	Used by SETL K for address within the block of the requested record.
104(68)	W1LPDR	8	Seek — Search address of the last prime–data record read.
112(70)	W1CBF	4	Current buffer address.
116(74)	W1EOB	4	End–of–buffer address.
120(78)	W1COUNTER	2	Counter used to count number of retries for Write–validity–checking.
122(7A)	PRIMEIND	1	Switch for testing same device.
123(7B)	FIXIND	1	Temporary storage.
124(7C)	W1FCPS	4	First Write channel program scheduled.
128(80)	W1QTABLE	60	Queue table (comprising the following fields)
128(80)	W1FR1ST	4	Pointer to first channel program on the Free queue.
132(84)	W1FRLAST	4	Pointer to last channel program on the Free queue.
136(88)		2	Reserved.
138(8A)	W1FREEC	2	Number of buffers on the Free queue.
140(8C)	W1RD1ST	4	Pointer to first channel program on the Read queue.
144(90)	W1RDLAST	4	Pointer to last channel program on the Read queue.
148(94)	W1READR	2	Number of unusued buffers on the Read queue.
150(96)	W1READC	2	Number of buffers on the Read queue.
152(98)	W1US1ST	4	Pointer to the first channel program on the User queue.

Offset	Field Name	Bytes	Field Description
156(9C)	W1USLAST	4	Pointer to the last channel program on the User queue.
160(A0)		2	Reserved.
162(A2)	W1USERC	2	Number of buffers on the User queue.
164(A4)	W1PX1ST	4	Pointer to first channel program on the PUTX queue.
168(A8)	W1PXLAST	4	Pointer to last channel program on the PUTX queue.
172(AC)		2	Reserved.
174(AE)	W1PUTXC	2	Number of buffers on the PUTX queue.
176(BO)	W1WR1ST	4	Pointer to the first channel program on the Write queue.
180(B4)	W1WRLAST	4	Pointer to the last channel program on the Write queue.
184(B8)		2	Reserved.
186(BA)	W1WRITEC	2	Number of buffers on the Write queue.
188(BC)	W1WAREA	36	Area for track–index entries (comprising the following three fields).
188(BC)	W1WCOUNT	8	Count of current index entry.
196(C4)	W1WCNXDM	8	Count of next normal or dummy entry.
204(CC)	W1WOVFL	10	Data of current overflow entry.
214(D6)	W1WDNXDM	10	Data of next normal or dummy entry.
224(E0)	W1WPLEN	2	Byte length of work area.
226(E2)	W1CURLEN	2	Length of current logical record.
228(E2)	W1TEMPSA	4	Temporary storage.
232(E8)	W1REGSV2	4	Area to save contents of a register.
236(EC)	W1REGSAV	4	Area to save contents of a register.
240(F0)	W1REGSV3	4	Temporary storage.
244(F4)	W1CP23PT	4	Address of CP 23.
248(F8)	W1CP26PT	4	Address of CP 26.

Offset	Field Name	Bytes	Field Description
252(FC)	W1CP25PT	4	Address of CP 25.
256(100)	W1CP24	68	CP 24 — read track indexes.
324(144)	W1WDCXDM	10	Data of current normal track–index entry (variable–length records only).
334(14E)	W1ISECT	1	Current input channel program sector value.
335(14F)	W1OSECT	1	Current output channel program sector value.
336(150)	W1DCBFA	4	Pointer to DCB field area.
340(154)	W1ICPEXT	16	Extension to the input channel program used with an RPS device. Set sector and TIC to input channel program.
356(164)	W1OCPEXT	16	Extension to the output (PUTX) channel program used with an RPS device.
372(174)	W1RDCNT	8	Read count of next block for channel program.
380(17C)	W1RDSECT	8	Read Sector of next block for channel program.
388(184)	W1CN5SAV	4	Save area to restore TIC address CN5 during overflow processing.
392(188)	W1RPSSA	16	Register save area for RPS processing.
408(198)	W1TOTAL	2	Byte count on track.
410(19A)	W1RECLEN	2	Minimum record length, prime records.
412(19C)	W1OVLEN	2	Minimum record length, overflow records.
414(19E)	W1FSTSH	2	Byte count to first shared track.
416(1A0)	W1RPSC1	1	Lower limit cylinder overflow.
417(1A1)	W1RPSC2	1	Upper limit cylinder overflow.
418(1A2)	W1RPSI1	1	Lower limit independent overflow.
419(1A3)	S1RPSI2	1	Upper limit independent overflow.

BISAM DCB Work Area

The BISAM DCB work area is pointed to by the DCBWKPT2 field of the DCB. The DCB work area format is shown in Figure 69.

0(0) DCWFCP4			
4(4) DCWFCP7			
8(8) DCWNUCPS	9(9) DCWNUCP4	10(A) DCWNUCP7	11(B) DCWNLSD
12(C) DCWFIOBU			
16(10) DCWLIOBU			
20(14) DCWFUPDI			
24(18) DCWLUPDI			
28(1C) DCWHIAV	29(1D) DCWWKNI	30(1E) DCWLEVC	31(1F) DCWNUWKN
32(20) DCWMSHIL			
36(24) DCWHIRPS	37(25) DCWNACT	38(26) DCWSIZE	
40(28) DCWOPCLS			
48(30) DCWERRCT	49(31) DCWFIOBE		
52(34) DCWLIOBE			
56(38) DCWSIOA			
60(3C) DCWDCBFA			
64(40) DCWIPG		66(42) DCWLPG	
68(44) DCWIOG		70(46) DCWLOG	

Figure 69. BISAM Work Area

Offset	Field Name	Bytes	Field Description
0(0)	DCWFCP4	4	Pointer to the first available set of channel programs in the CP 4–CP 5–CP 6 or CP 4–CP 5W–CP 6W queue. The second word of the second CCW in the channel program set points to the next set of channel programs. The pointer is 0 in the last set on the queue. If no set of channel programs is available, this field is 0.
4(4)	DCWFCP7	4	Pointer to the first available CP 7 or CP 7W. This queue is handled similarly to the one pointed to by DCWFCP4.
8(8)	DCWNUCPS	1	The number of IOBs awaiting CP 1 or CP 2.
9(9)	DCWNUCP4	1	The number of IOBs awaiting CP 4–CP 5–CP 6 or CP 4–CP 5W–CP 6W.
10(A)	DCWNUCP7	1	The number of IOBs awaiting CP 7 or CP 7W.
11(B)	DCWNLSD	1	The number of high–level indexes searched on a device. This number equals DCBNLEV unless the highest level index is searched in main storage in which case the number equals DCBNLEV minus 1.
12(C)	DCWFIOBU	4	Address of the first IOB in the queue of unscheduled IOBs. This field is 0 if no IOBs are unscheduled.
16(10)	DCWLIOBU	4	Address of the last IOB in the queue of unscheduled IOBs. This field is 0 if no IOBs are unscheduled.
20(14)	DCWFUPDI	4	Address of the first IOB in the update queue, that is, the queue of IOBs for which a READ KU has been successfully completed, but for which no WRITE K has yet been issued. This field is 0 when the queue is empty.
24(18)	DCWLUPDI	4	Address of the last IOB in the update queue. This field is 0 when the queue is empty.
28(1C)	DCWHIAV	1	Switches Bit Meaning 0 CP 1 or CP 2 is available. 1 Highest–level index must be searched in main storage. 2–7 Reserved.
29(1D)	DCWWKNI	1	0 WRITE KN is in process. 1 First time switch (used with various WRITE KN channel programs which are executed repetitively). 2 Same module switch. 3 Add to the end of the data set. 4 CP 12A or CP 13A detected an end–of–file mark. 5 CP 11A—First use by a given WRITE KN. 6 Work area for WRITE KN was obtained by Open (VLR only) 7 Reserved.

Offset	Field Name	Bytes	Field Description
30(1E)	DCWNLEVC	1	Counter used when rewriting high–level indexes.
31(1F)	DCWNUWKN	1	The number of WRITE KN IOBs awaiting completion of WRITE KN.
32(20)	DCWMSHIL	4	Address of the last active high–level index entry in main storage. This field is 0 when the high–level index is not searched in main storage.
36(24)	DCWHIRPS	1	Used with WRITE KN. It contains DCBHIRPD if the current track of prime data being processed is not shared with a track index or DCBHIRSH if it is.
37(25)	DCWNACT	1	The number of READ or WRITE K IOBs awaiting completion of WRITE KN.
38(26)	DCWSIZE	2	The total size, in doublewords, of (1) the DCB work area, (2) all the channel programs, and (3) the minimum size work area used by WRITE KN if the user has not supplied a work area.
40(28)	DCWOPCLS	8	Data saved by common ISAM open executor in DCBWKPT3 and DCBWKPT4. This data will be restored in these two fields by the BISAM Close routine and used by the common ISAM close executor. (The data saved is the address of the format–2 DSCB and the UCB address of the device on which the volume containing the DSCB is mounted. This address has 5 bytes for CCHHR and 3 bytes for UCB address.)
48(30)	DCWERRCT	1	Number of positions left for IOBs to be placed on the error queue. Maximum value = 2(NCP)+DCBUFNO.
49(31)	DCWFIOBE	3	Address of the first IOB on the error queue, which contains requests that ended with a permanent error or used a dynamic buffer. This address is 0 if the queue is empty.
52(34)	DCWLIOBE	3	Address of the last IOB on the error queue. This address is 0 if the queue is empty.
56(38)	DCWSIOA	4	Address of the RPS SIO appendage.

Note: This field is not used by ISAM routines. See the ISAM extension of the DEB.

Offset	Field Name	Bytes	Field Description
60(3C)	DCWDCBFA	4	Pointer to DCB field area.
64(40)	DCWIPG	2	Prime record (other than the last) overhead (variable–length records only).
66(42)	DCWLPG	2	Last prime record overhead (variable–length records only).
68(44)	DCWIOG	2	Overflow record (other than the last) overhead (variable–length records only).
70(46)	DCWLOG	2	Last overflow record overhead (variable–length records only).

QISAM Track–Index Save Area

Calculations for the track–index save area

The size of the track–index save area (TISA) is equal to the total of the following five items:

1. TISA control fields — 20 bytes.
2. Area for the track–index entries
   a. Number of entries equal to the maximum number of entries on a track. This is ISLNIRT if the track index is on one track; otherwise, ISLHIRT is used. If ISLHIRT is odd, then the calculations are performed with the number of entries equal to ISLHIRT + 1 to allow the save area enough space for the last pair of entries.
   b. Size of each entry equals COUNT + KEY + DATA

      COUNT=8

      KEY=KEY LENGTH

      DATA=10

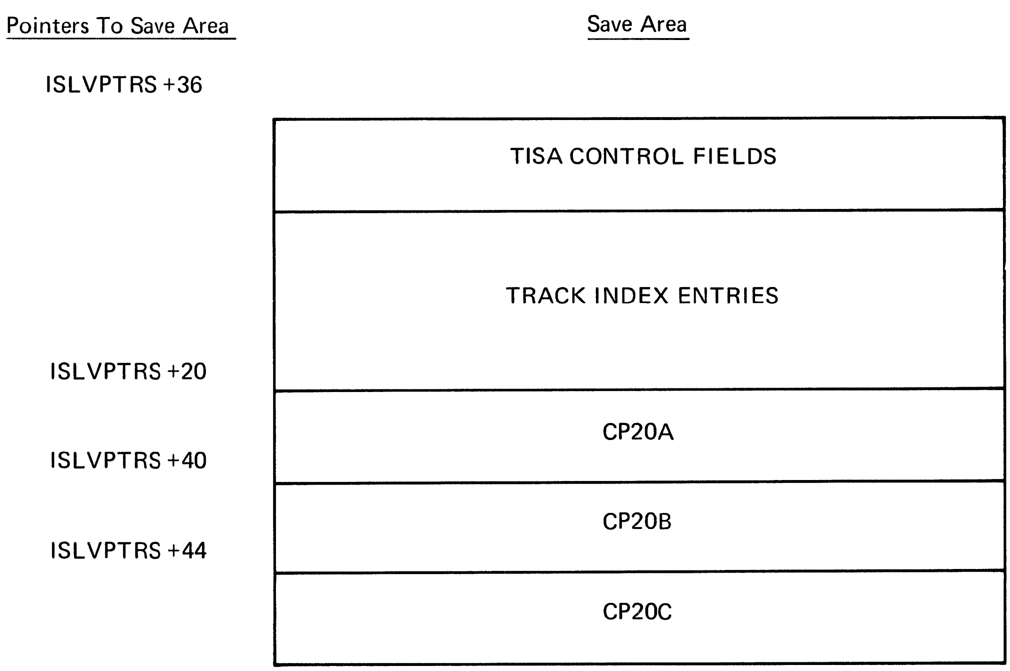


Figure 70. Track–Index Save Area

3. Channel program 20A if no shared track.
4. Channel program 20B if shared track.
5. Channel program 20C if write–check.

Track–Index Save Area (TISA)

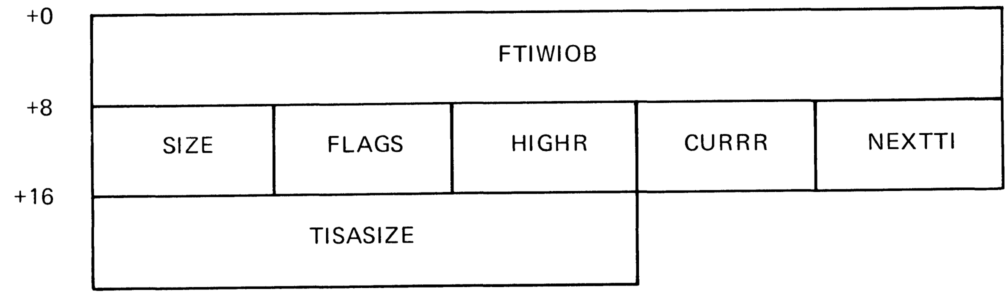


Figure 71. TISA Control Fields

Field Name	Bytes	Description
FTIWIOB	8	MBBCCHHR for the prime–data track which is pointed to by the seek CCW in CP 20 and the search CCW in CP 18.
SIZE	2	Length of one track–index entry (8+KL+10).
FLAGS	1	X'80' — Resume load. Turned on for the first track index write.
		X'40' — Close. Turned on by 202I to force writing of the track index.
		X'20' — End of track–index track.
		X'10' — End of cylinder.
		X'08' — Execute CP 20 alone (with one CP 18).
		X'04' — Close. Track–index entries previously generated.
HIGHR	1	Highest record number for the current track of track index (either ISLHIRT or ISLNIRT).
CURRR	1	Current record number (last record moved to TISA). Initialized to 0.
NEXTTI	3	Address in TISA where the next track–index entry will be placed. Initialized to TISA + 20.
TISASIZE	4	Size of TISA saved for the Close routine to issue a FREEMAIN.

00(00) DFATDC	02(02) DFARORG3	06(06) DFANREC	
DFANREC (cont.)	10(0A) DFAST	11(0B) DFALPDA	
DFALPDA (cont.)	19(13) DFANBOV	21(15) DFARORG2	23(17) DFANOREC
DFANOREC (cont.)	25(19) DFALIOV		
DFALIOV (cont.)	33(21) DFARORG1	35(23) Not used	36(24) DFACOUNT

Figure 72. DCB Field Area

Offset	Field Name	Bytes	Field Description
00(00)	DFATDC	2	Tag deletion count. User's count field for records marked for deletion. (Refer to DCBTDC in the data control block.)
02(02)	DFARORG3	4	The number of times an overflow record was referred to by a READ or WRITE instruction.
06(06)	DFANREC	4	Number of logical records in the prime–data area.
10(0A)	DFAST	1	Status indicators. Bit 0 — Single schedule mode 1 — Key sequence to be checked 2 — Initial load has been completed 3 — Data set extension (resume loading) will begin on new cylinder. 4 — Reserved 5 — First macro not yet received 6 — Last block full 7 — Last track full
11(0B)	DFALPDA	8	Direct–access device address of the last prime–data record in the prime–data area (in the form MBBCCHHR).
19(13)	DFANBOV	2	Number of bytes remaining on current overflow track (variable–length records only).
21(15)	DFARORG2	2	Number of tracks (partially or wholly) remaining in the independent overflow area.
23(17)	DFANOREC	2	Number of logical records in a overflow area.
25(19)	DFALIOV	8	Direct–access device address of the last record written in independent overflow area (in the form MBBCCHHR).
33(21)	DFARORG1	2	Number of full cylinder overflow areas.
35(23)			Not used.
36(24)	DFACOUNT	4	Number of open DCBs on this data set.

SECTION 6: DIAGNOSTIC AIDS

Appendage Codes

Before an EXCP command is issued, QISAM scan mode and BISAM enter an appendage code into the IOB extension. When the appendage is entered from the I/O supervisor, the appendage routine tests the code to determine which functions to perform to complete processing for the input/output request.

When an appendage routine schedules an asynchronous routine, it puts an asynchronous code into the IOB extension. When the asynchronous routine gains control it tests the asynchronous code to determine the functions it must perform.

QISAM Scan Mode Appendage Codes

The following codes apply under both channel–end and abnormal–end conditions:

Code	Meaning
0	Completion of READ
4	Completion of SETL (K or I)
8	Completion of WRITE (with or without write–checking)
12	Completion of CHECK (read–back for write–checking)
16	Completion of REWRITE (write–back when write–checking)
20	Completion of RECHECK (read–back after REWRITE during write–checking)

BISAM READ and WRITE K Appendage Codes

The following codes apply under both channel–end and abnormal–end conditions:

Code	Meaning
0	Completion of CP 4–5–5W for READ
1	Completion of CP 4–5–5W for WRITE
2	Completion of CP 7 or 7W
3	Completion of CP 1 or 2
5	Completion of CP 6 or 6W
6	Completion of CP 5W for write–checking after WRITE

BISAM WRITE KN Appendage Codes

The following codes apply under both channel–end and abnormal–end conditions:

Code	Meaning
4	Completion of CP 14 part 2 (fixed–length records with user work area)
7	Completion of CP 1 or CP 2 for WRITE KN

Code	Meaning
8	Completion of CP 8
9	Completion of CP 10A for true insert or part 2 of CP 14 (variable–length records), for EOF extension
10	Completion of CP 10B for true insert or part 2 of CP 14 (variable–length records), when part 1 has been executed
11	Completion of CP 10B for addition to end–of–data set
12	Completion of CP 14 or part 1 of CP 14 (fixed–length records with user work area and variable–length records), for setups 1, 2, and 5 (asynchronous routine codes 9, 10, and 13)
13	Completion of CP 14 or part 1 of CP 14 (fixed–length records with user work area and variable–length records), for setups 3, 4, and 6 (asynchronous routine codes 11, 12, and 14)
14	Completion of CP 15
15	Completion of CP 16 for setup 2 (search overflow chain for last overflow record in the chain: addition to end–of–data set)
16	Completion of CP 16 for setup 2 (search overflow chain for record which logically precedes or is equal to new record to be added: true insertion)
17	Completion of CP 17 when used for track index only or part 2 of CP 14 (variable–length records) when part 1 has not been executed (no overflow)
18	Completion of CP 17 when used for track index and when it is to be continued for higher level indexes
19	Completion of CP 17 when it is to be started or continued for higher level indexes
20	Completion of CP 9A, CP 11A, CP 12A, CP 13A, or CP 12AV
21	Completion of CP 9B, CP 11B, CP 12B, CP 13B, or CP 12BV
22	Completion of CP 9C, CP 123W, or CP 123WV
23	Completion of CP 10A for addition to end of data set
24	Completion of CP 12C or CP 13C

Asynchronous Codes

BISAM READ and WRITE KN Asynchronous Codes

The following codes direct asynchronous coding to the proper routines:

Code	Condition
0	Successful completion of CP 4–5–6

Code	Condition
1	EXCP macro instruction to be issued
2	Successful completion of CP 7
3	Successful completion of CP 1 or CP 2
4	Unsuccessful completion of CP 4–5–6
6	Unsuccessful completion of CP 1
7	Unsuccessful completion of CP 1 or CP 2

BISAM WRITE KN Asynchronous Codes

The following codes direct asynchronous coding to the proper routines:

Code	Condition
1	Scheduled to issue an EXCP which could not be done in an appendage routine because a different device (UCB) was involved.
8	Scheduled upon the successful or unsuccessful completion of a WRITE KN macro instruction.
9	Scheduled to set up and execute CP 14 when a record is bumped from a prime–data track as a result of a new record being placed on that track (setup 1).
10	Scheduled to set up and execute CP 14 when a new record is to be added to the end of the data set, the last track is full, and no overflow chain currently exists for the last track (setup 2).
11	Scheduled to set up and execute CP 14 when a new record is to be added to the end of the data set, the last track is full, but an overflow chain already exists for the last track (setup 3).
12	Scheduled to set up and execute CP 14 when a new record is a true insert and is to go in the middle of an overflow chain (setup 4).
13	Scheduled to set up and execute CP 14 when a new record is a true insert and it is to become the first record in an already existing overflow chain (setup 5).
14	Scheduled to set up and execute CP 14 when a new record is a true insert and it has a key equal to that of the key of a record in the overflow chain (the record is marked for deletion). The new record simply replaces the deleted record (setup 6).
15	Scheduled to set up and execute CP 14 (for variable–length records only) when more than one record is bumped from a prime–data track (setup 1).
16	Scheduled to set up and execute the CP 14 extension (the variable–length records only) to write an EOF mark in independent overflow.

Exception Codes

QISAM Exception Codes

QISAM exception codes and the macro instructions which set them are summarized in Figure 73.

Exception Code		Code Set By					Condition if On
Field	Bit	CLOSE	GET	PUT	PUTX	SETL	
DCBEXCD1	0					Type K	Record is not found
	1					Type I	Invalid actual address for lower limit
	2			X			Space is not found in which to add a record
	3					X	Invalid request
	4		X				Uncorrectable input error
	5	X		X	X		Uncorrectable output error
	6		X			X	Block could not be reached (input)
	7	X	X				Block could not be reached (update)
DCBEXCD2	0			X			Sequence check
	1			X			Duplicate record
	2	X					Data control block is closed when error routine is entered
	3		X				Overflow record[1]
	4			X			Length of logical record is greater than DCBLRECL (Variable length records only)
	5-7						Reserved for future use

[1]The SYNAD routine is entered only if bit 4, 5, 6, or 7 of DCBEXCD1 is also on.

Figure 73. QISAM Exception Code Summary

BISAM Exception Codes

BISAM exception codes and the macro instructions which set them are summarized in Figure 74.

Exception Code		Code Set By		Condition if On
Field	Bit	READ	WRITE	
DECBEXCD1	0	X	Type K	Record is not found
	1	X	X	Record length is checked
	2		Type KN	Space is not found
	3		Type K	Invalid request
	4	X	X	Uncorrectable I/O error
	5	X	X	Unreachable block
	6	X		Overflow record
	7		Type KN	Duplicate record
DECBEXCD2	0-5			Reserved for future use
	6	X	X	Channel program initiated by an asynchronous routine (variable length records only)
	7	X		Previous macro was READ KU

Figure 74. BISAM Exception Code Summary

SECTION 7: APPENDIXES

Appendix A: ISAM Data Set Organization

Introduction

The indexed sequential access methods (ISAM) can be defined as the combination of data set organization and the techniques used to process the data. With the indexed sequential organization, data records are arranged in logical sequence by a key field. An indexed sequential data set resides on direct–access storage devices and can occupy up to three different areas:

- Prime area

  This area contains data records and related track indexes. It exists for all ISAM data sets.

- Overflow area

  This area contains overflow from the prime area when new data records are added. It is optional.

- Index area

  This area contains master and cylinder indexes associated with the data set. It exists for a data set that has a prime area occupying more than one cylinder.

The indexes of an ISAM data set are analogous to the card index in a library. For example, if the library user knows the name of the book or the author, he can look in the card index and obtain a catalog number which will enable him to locate the book in the book files. He would then go to the shelves and proceed through each row until he found the shelf containing the book. Usually each row contains a sign to indicate the beginning and ending numbers of all books in that particular row. Thus, as he proceeded through the rows, he would compare the catalog number obtained from the index with the numbers posted on each row. Upon locating the proper row, he would then search that row for the shelf that contained the book. Then he would look at the individual book's numbers on that shelf until he found the particular book.

ISAM uses the indexes in much the same way to locate records in an indexed sequential data set. The operating system provides both the queued and basic access techniques to process an indexed sequential data set. The queued access technique is used to create the data set and add records to the end. It can also be used to sequentially process or update the records. The basic technique is used to read or update records and to insert new records at any place in the data set.

Data Set Structure

The overall structure of an indexed sequential data set is shown in Figure 75. The prime area contains data records arranged according to the collating sequence of a key field in each record. As the records are stored (written) in the prime area, the system prepares a track index. Each entry in the track index identifies the key of the last record on each track. There is a track index for each cylinder in the data set. If more than one cylinder is used, the system develops a higher level index called a cylinder index. Each entry in the cylinder index identifies the key of the last record in the cylinder.

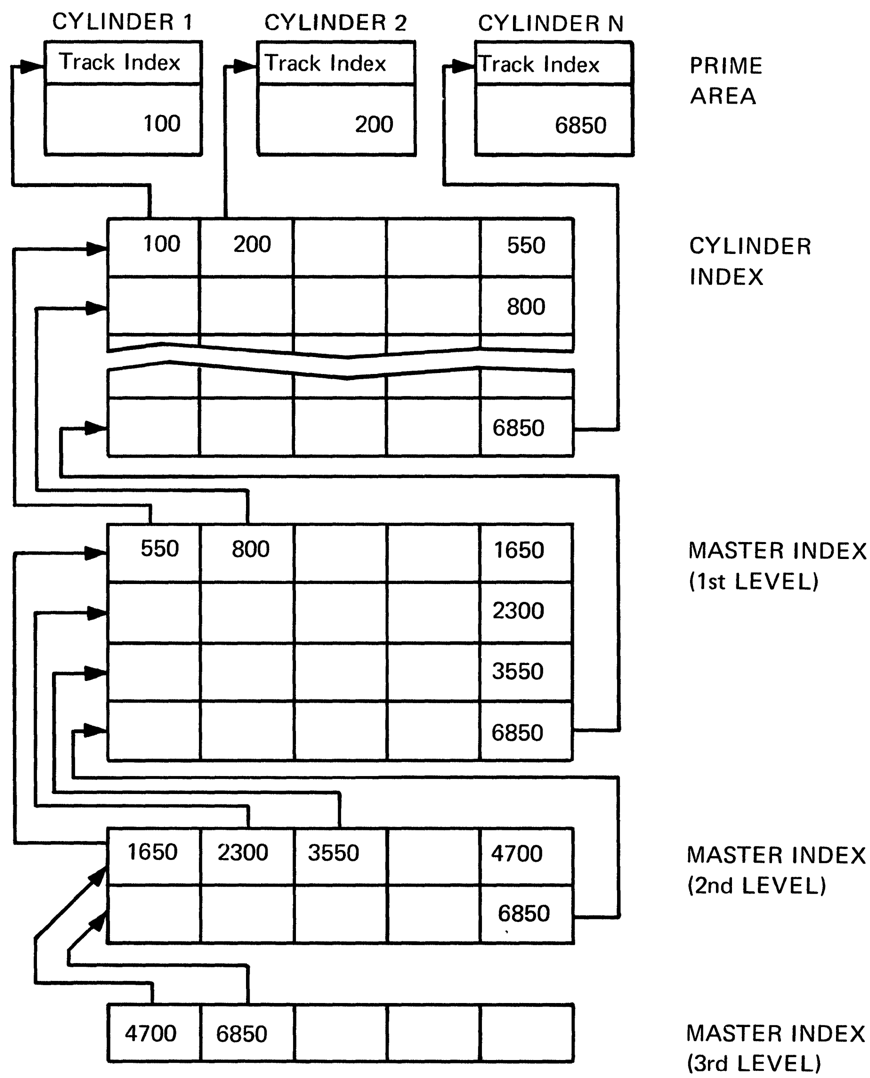


Figure 75. Indexed Sequential Data Set Structure

To increase the speed of searching the cylinder index, you can request the system to create a master index that indexes the cylinder index. You can specify through the data control block (NTM and OPTCD operands) that, if the size of a cylinder index exceeds a certain number of tracks, a master index should be created. The example in Figure 75 shows an entry in the master index (first level) for each one track of cylinder index entries. If the size of the master index exceeds the number of tracks specified in the data control block the master index is automatically indexed by a higher level master. This is illustrated in Figure 75 by the second level master. Three such higher level master indexes can be constructed.

Prime Data Area

Records are written in the prime area when the data set is created or updated. Figure 76 illustrates the initial structure of a cylinder of the prime area. The track index is contained on the first track of the cylinder. Note that a pair of track index entries is associated with each prime track in the cylinder. In this example, the last track of the cylinder is reserved for a cylinder overflow area.

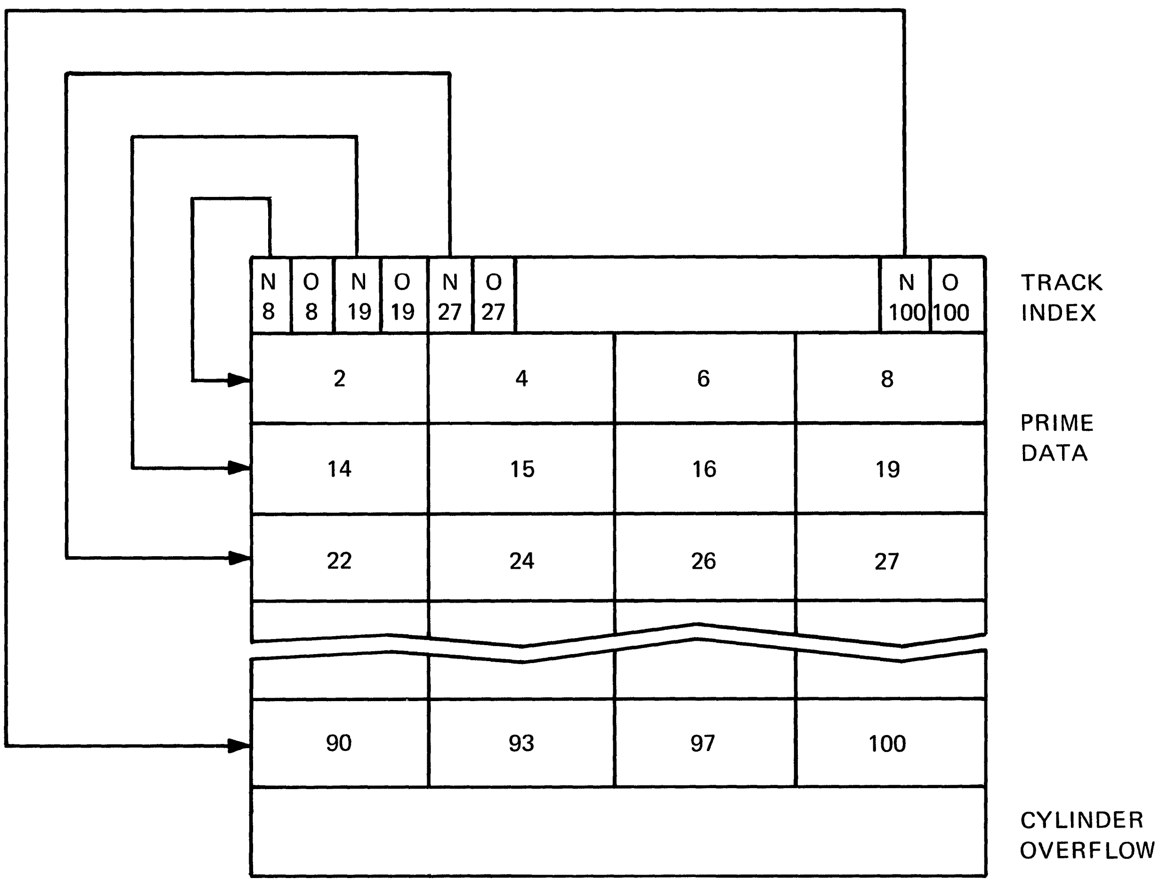


Figure 76. Initial Structure of Prime Cylinder

Index Areas

The operating system automatically generates at least two levels of indexes: a track index and a cylinder index. (Up to three levels of master indexes are created if requested.)

Track Index: This is the lowest level of index and is always present. There is one such index for each cylinder in the prime area; it is written on the first track of the cylinder that contains the indexes. The index consists of a series of paired entries; that is, a normal and an overflow entry for each prime track. The normal entry contains the home address of the prime track and the key of the highest record on the track. The overflow entry is originally the same as the normal entry but is changed when records are added to the data set.

In Figure 77, the track index is an expanded detail of the index shown in Figure 76. Note that the data area of the first normal entry points to track 01 and the key area represents the highest key on track 01. Since this figure illustrates the initial structure of the data set, the first overflow entry is the same as the normal entry.

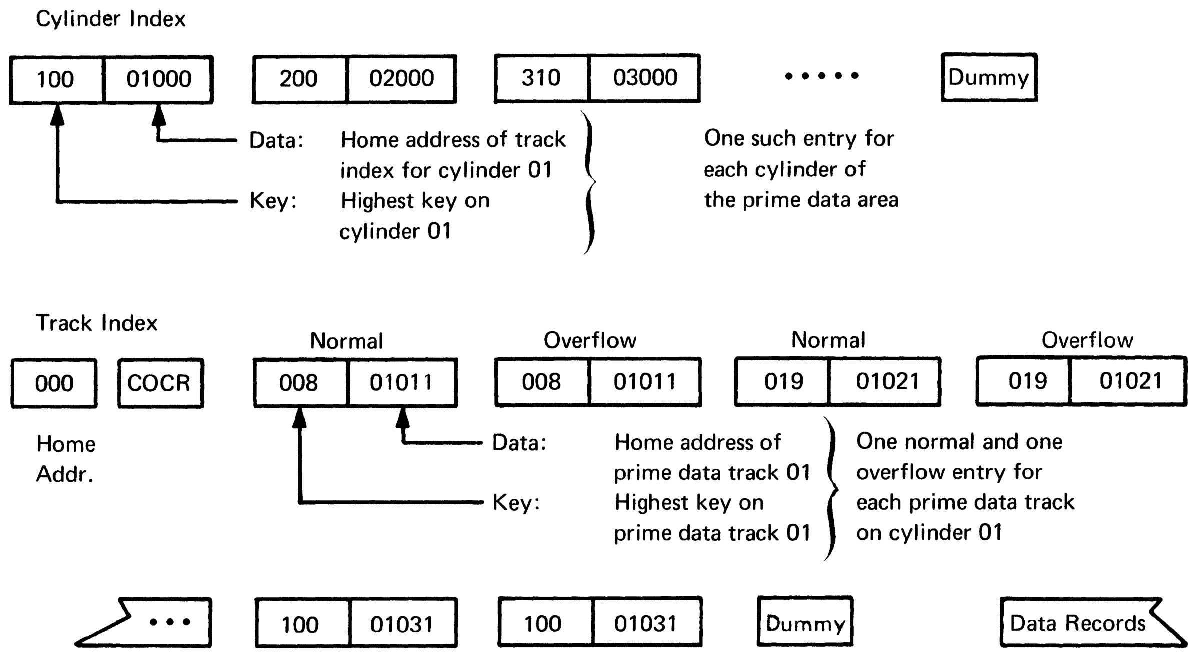


Figure 77. Structure of Cylinder Index and Track Index

Cylinder Index: For every track index created, the system generates a cylinder index entry. There is one cylinder index for a data set, each entry of which points to a track index. Since there is one track index per cylinder, there is one cylinder index entry for each cylinder in the prime area. In Figure 77, the data area of the first cylinder index entry points to the home address of the track index for cylinder 01. The key area contains the number 100 which represents the highest key on the cylinder. For simplicity, in Figure 77 only the cylinder, track, and record number portion of the address in the data areas is shown.

Overflow Areas: As records are added to an indexed sequential data set, space is required to contain those records that do not fit on the prime data track on which they belong. You can request that a number of tracks be set aside as a cylinder overflow area to contain overflows from prime tracks in each cylinder. When a cylinder overflow area is specified, record 0 of the track index is used as a cylinder overflow control record (see Figure 77). ISAM uses this record to keep such information as the address of the last the overflow record in cylinder and the number of bytes remaining on the current overflow track.

An advantage of using cylinder overflow areas is a reduction of search time required to locate overflow records. To access the cylinder overflow area requires only a seek to another track within the cylinder. This can be performed with less system overhead than a seek to another cylinder as is required to access an independent overflow area.

Instead of, or in addition to, cylinder overflow areas, you can request an independent overflow area. Overflow from anywhere in the prime data area is placed in a specified number of cylinders reserved for this area. An advantage for having an independent overflow area is a reduction in unused space reserved for overflow. A disadvantage is the increased search time required to locate overflow records in an independent area (see Figure 79).

It is good practice to request cylinder overflow areas large enough to contain a reasonable number of additional records, and an independent overflow area to be used as the cylinder overflow areas are filled.

Adding Records to a Data Set

A new record added to an indexed sequential data set is placed into a location on a track determined by the value of its key field. If records were inserted (added) in precise physical sequence, insertion would require shifting all records of the data set with keys higher than that of the one inserted. However, because an overflow area

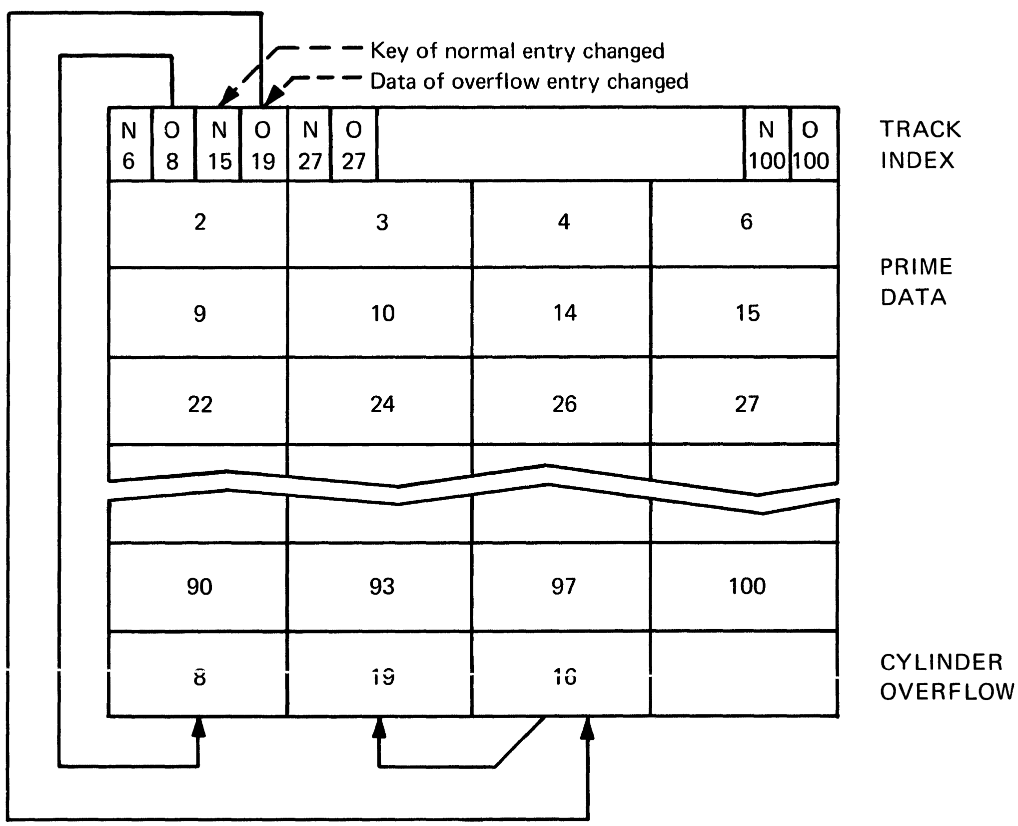


Figure 78. Structure of Prime Cylinder After Cylinder Overflow

exists, the indexed sequential data organization allows a record to be inserted into its proper position with only the records on the track in which the insertion is made being shifted. When a record is to be inserted, the records already on the prime track that are to follow the new record are written back on the track after the new record. If the addition of records results in insufficient track space for all the records to be written onto the track, the records that do not fit are written onto an overflow track. This technique maintains the sequential order of records on the prime track. Three situations may occur when a record is added to a data set. Each is discussed below.

First Addition to a Prime Track: When a data set is created, its records are placed on the prime tracks in the storage area allocated to the data set as shown in Figure 76. If a record (for example, record 3) is to be inserted into the data set, the indexes indicate that record 3 belongs on prime track 01. Record 3 is written immediately following record 2, and records 4 and 6 are retained on prime track 01 (see Figure 78). Since record 8 no longer fits on this track, it is written on track 09 (cylinder overflow track).

The key area of the normal index entry is changed, since record 6 is now the highest record on the track. The data area of the overflow index entry is changed; it now points to record 8 as the first record on track 09. The first addition to a track is always handled in this way.

When records 9 and 10 are added, prime track 02 receives these records as shown in Figure 78. Record 19 is shifted to track 09 (cylinder overflow track). Record 16 is also shifted to the overflow track after record 19. Note that records 16 and 19 are chained together to show the logical sequence and to indicate that they are associated with the same prime track. (Overflow records are chained through a link field which forms the first 10 bytes of each overflow record.)

Subsequent Additions to a Track: Subsequent additions are written either on the prime track where they belong or as part of the overflow chain from that track. If the addition belongs between the last prime record on a track and a previous overflow from that track, it is written in the first available location in the overflow area, with its link field containing the address of the next record in the chain. Because the data area of the overflow index entry always refers to the address of the lowest key in a chain, it is changed.

If subsequent additions belong on a prime track, they are written in proper sequential location on the prime track. For example, records 11 and 13, as shown in Figure 79, are written in proper sequential position on track 01. Record 15 (previously the highest record on the prime track) is shifted to the cylinder overflow area with its link field chaining to record 16. Record 14 is shifted to the independent overflow area since the cylinder overflow area is full. The link field in record 14 points to record 15, the next record in the chain. The key area of the normal index entry is changed to indicate that record 13 is the highest on the prime track. The data area of the overflow index entry is changed to point to record 14 in the independent overflow area as the first record in the overflow chain.

Addition of High Keys: A record with a key higher than the current highest key in the data set is placed at the end of the prime area, if there is room. Such an addition is handled, in effect, as if it had been presented when the file was first created.

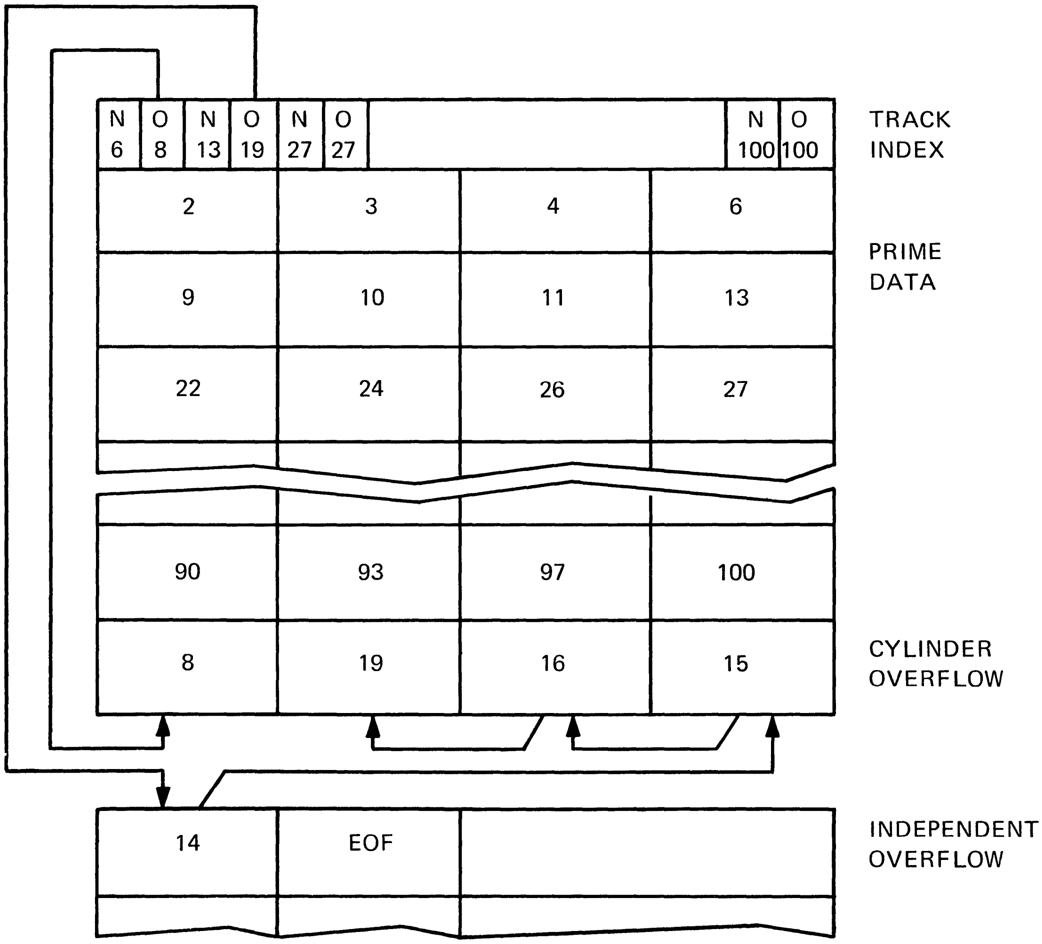


Figure 79. Structure of Prime Cylinder After Independent Overflow

If the prime area is full, the new record is written in the overflow area and linked to the overflow chain from the last prime track. The key area of higher level indexes is changed to reflect the addition.

Detailed Index Description

All index records have three sections: count, key, and data (except the cylinder overflow control record, which has no key section). Index records are formed in main storage and written on direct–access devices by QISAM load mode channel programs operating with I/O supervisor. BISAM channel programs may later cause sections of the indexes to be updated when deleting and/or adding records to the data set. In all records (index and data), the BB portion of MBBCCHHR is 0. The BB portion of the IOB is filled prior to EXCP from the DEB. This avoids having to mount 2321 bins back into their original position. Figure 80 shows the ISAM index entry format.

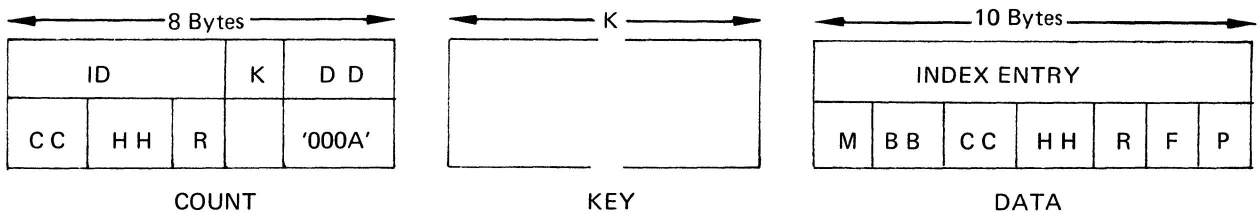


Figure 80. Format of ISAM Index Entry

The count section is 8 bytes in length, in the following format: CC HH R K D D.

CC HH R
: is the direct–access device address of this index entry; the components of this address vary with the type of device.

K
: is the length of the key of each record in the data set. It is also the length of the key section of each index entry.

DD
: is the length of the data section of each index record. It is always hexadecimal '000A' (indicating 10 bytes) except for the cylinder overflow control record, whose data section is 8 bytes long.

The key section is always the same length as the key of each record in the data set and has a value equal to the highest key referenced by this entry.

The data section is always (except for the cylinder overflow control record) 10 bytes in length, in the following format:

M BB CC HH R F P.

The first 8 bytes contain the direct–access device address of the data record whose key is equal to the key section of this index entry.

This address is represented as follows:

M
: is the DEB extent serial number.

BB CC HH R
: is the direct–access device address of the data record. The components of the address vary with the type of device.

F, the flag reference code byte, is broken down into bits, as follows:

Bit	0	1	2	3	4	5	6	7
	C	C	C	C	C	I	I	I

where CCCCC is the index entry type code and I I I indicates the level of index entry.

The following are valid index entry type codes:

CCCCC =	00000	normal entry	data record resides on unshared track
	00001	normal entry	data record resides on shared track
	00010	overflow entry	end (last entry in chain)
	00011	overflow entry	chained (not last entry in chain)
	00100	dummy entry	end of index
	00101	dummy entry	chained
	00110	inactive entry	

Inactive entries are written by QISAM load mode Close executors to fill out allocated, but unused, space at the end of each index.

The following are valid codes for level of index entry:

I I I =	000	The track index
	001	The cylinder index
	010	The first level master index
	011	The second level master index
	100	The third level master index

P, the command code byte, is referenced by channel programs. The three valid hexadecimal command codes are 1B, 0B, and 07.

1B =Seek HH	These are used for entries whose data records are on the same volume as the index entry.
0B =Seek CC HH	
07 =Seek BB CC HH	This is used when the data record is on a volume other than the one on which the index entry resides. For the 2321 data cell drive, the seek code must be 07 if the data set crosses a strip. It is also used in all overflow and dummy index entries. Its purpose is to cause an interrupt during the execution of ISAM channel programs (protection check) so that the ISAM appendage routines can issue another EXCP or check for an error or special procedure.

Track–Index Records: Track–index entries consist of a series of paired entries; that is, a normal and an overflow entry for each track. A dummy end entry indicates the end of the index, which may be padded with inactive entries. The first track of a track index may contain a cylinder overflow control record.

Track Capacity Record: The track capcity record is R0 of each prime–data track for variable–length records. Bytes 0–1 of the data portion contain the number of unused bytes currently left on the track. Byte 2 contains the highest record ID currently on the track.

Cylinder Overflow Control Record: The cylinder overflow control record is the R0 record on the first track of the track index, if the DCBOPTCD field has specified the cylinder overflow option. It has no key section. The 8–byte data section is in the following format:

HH R YY T 00

Initially,

HH R
: indicates the first track of the cylinder overflow area, and R = 0.

After overflow has occurred,

HH R
: indicates the track and record number of the last overflow record.

YY
: indicates the number of unused bytes remaining on the current overflow track, but is not maintained when the data records are of fixed length.

T
: indicates the number of tracks remaining unused in the cylinder overflow area.

00
: indicates that these two bytes are not used.

Figure 81, which follows, contains a detailed explanation of track–index records.

Overflow Linkage: On the first overflow from a prime–data track:

1. The data portion of that track's overflow index entry is written onto the overflow track as a link field in front of the data section of the overflow record.
2. The key of the prime–data track's normal index entry is updated to contain the key of the last record remaining on the prime–data track.
3. M BB CC HH R in the data portion of the prime–data track's overflow index entry is updated to contain the address of the overflow record. The F byte is changed from CCCCC = 00010 to CCCCC = 00011 to indicate that this overflow index entry is pointing to an overflow chain.

On subsequent overflows from the prime–data track:

1. The link fields of all but the highest overflow record are modified to contain the location of the next higher overflow record. The F byte indicates CCCCC = 00011 (overflow chain).
2. The link field of the highest overflow record will contain a meaningless address and the F byte indicates CCCCC = 00010 (end of the overflow chain).
3. The key of the overflow index entry for the prime–data track is modified, if necessary, to contain the highest overflow key. This occurs only when adding a record to the end of the data set.

4. The key of the normal index entry for the prime–data track is modified to contain the key of the last record on the prime–data track.

5. The data portion of the overflow index entry for the prime–data track is modified, if necessary, to contain the location of the lowest overflow record.

Type of Entry	Key	Data			
		M BB CC HH	R	F	P
Normal, Data Record on Unshared Track	Highest key on prime data track pointed to by data portion of this index entry.	Location of track whose highest key equals the key field of this index entry. (The cylinder is the same cylinder on which this index entry resides.)	Hexadecimal '00'	CCCCC = 00000, III = 000	Hexadecimal '1B'
Normal, Data Record on Shared Track	Same as Normal, Data Record on Unshared Track.	Same as Normal, Data Record on Unshared Track.	Record number of first data record on the shared track. For variable length records, R equals the highest record ID currently on the track that the index entry references.	CCCCC = 00001, III = 000	Hexadecimal '1B'
Overflow, End and Chained	End—same as preceding normal index entry. Chained—highest key to overflow from the track referenced by this entry.	End—same as preceding normal index entry. Chained—location of record with lowest key to overflow from the track referenced by this entry.	End—Hexadecimal 'FF'. Chained—record number with lowest key to overflow the track referenced by this entry.	End—CCCCC = 00010, III = 000 Chained—CCCCC = 00011, III = 000	Hexadecimal '07'
Dummy, End of Index	Maximum value (each byte equal to hexadecimal 'FF').	Minimum Value (each byte equal to hexadecimal '00').	Hexadecimal '00'	CCCCC = 00100, III = 000	Hexadecimal '07'
Inactive	Maximum value (each byte equal to hexadecimal 'FF').	Minimum value (each byte equal to hexadecimal '00').	Hexadecimal '00'	CCCCC = 00110, III = 000	Hexadecimal '07'

Figure 81. Description of Track Indexes

Type of Entry	Key	Data			
		M BB CC HH	R	F	P
Normal	Highest key on the cylinder whose track index begins at location specified by data portion of this index entry.	Location of start of track index on the cylinder whose highest key equals the key of this index entry.	Record number of first data record on first track of the track index. If no data records on that track (an unshared track), R = hexadecimal '00'.	CCCCC = 00000, III = 001	Hexadecimal '07' if this cylinder index entry references a track entry on either a different volume, or on a different strip if the device is a 2321 data drive. Hexa-decimal '0B' if same volume or strip.
Dummy, End	Maximum value, (each byte equal to hexadecimal 'FF').	Minimum value, (each byte equal to hexa-decimal '00').	Hexadecimal '00'	CCCCC = 00100, III = 001.	Hexadecimal '07'
Dummy, Chained	Maximum value (each byte equal to hexadecimal 'FF').	Location of next track of this cylinder index.	Hexadecimal '00'	CCCCC = 00101, III = 001	Hexadecimal '07'
Inactive	Maximum value (each byte equal to hexadecimal 'FF').	Minimum value (each byte equal to hexa-decimal '00').	Hexadecimal '00'	CCCCC = 00110, III = 001	Hexadecimal '07'

Figure 82. Description of Cylinder Indexes

Cylinder Index Records: A cylinder index is created for the data set if the processing program has requested space that extends over more than one cylinder. Figure 82 contains a detailed explanation of cylinder index records.

Master Index Records: One or more levels of master indexes are created if the DCBOPTCD field has specified this option.

Figure 83 contains a detailed explanation of master index records.

Type of Entry	Key	Data			
		M BB CC HH	R	F	P
Normal	Highest key on a track of the next lower level index. That track is pointed to by the data portion of this index entry.	Location of the track within next lower level index, whose highest key equals the key of this index entry.	Hexadecimal '00'	CCCCC = 00000 III = 010, 011, or 100	Hexadecimal '1B' if next lowest level index is on same cylinder as this index entry. Hexadecimal '0B' if not on same cylinder, but, for 2321 data cell drive, on same strip. Hexadecimal '07' for 2321 data cell drive if indexes cross strip boundaries.
Dummy, End	Maximum value (each byte equal to hexadecimal 'FF').	Minimum value (each byte equal to hexa-decimal '00').	Hexadecimal '00'	CCCCC = 00100, III = 010, 011, or 100	Hexadecimal '07'
Dummy, Chained	Maximum value (each byte equal to Hexadecimal 'FF').	Location of next track of this level master index.	Hexadecimal '00'	CCCCC = 00101, III = 010, 011, or 100	Hexadecimal '07'
Inactive	Maximum value (each byte equal to hexadecimal 'FF').	Minimum value (each byte equal to hexa-decimal '00').	Hexadecimal '00'	CCCCC= 00110 III = 010, 011, or 100	Hexadecimal '07'

Figure 83. Description of Master Indexes

Appendix B: ISAM Channel Programs

The channel program for each request using ISAM is constructed by the appropriate module. All ISAM channel programs are listed in Figure 84. The address of the channel program is placed in the IOB for that request. A channel program consists of a group of channel command words (CCWs), each word of which has the following format:

Command Code (1 byte)	Address (3 bytes)	Flags (5 bits)	000 (3 bits)	(ignored) (1 byte)	Count (2 bytes)

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

(In some TIC CCWs, these bytes contain flags or a chain address.) The entry in the address field is one of the following:

- The main–storage address where data is to be placed or found; for a Read or a Write command word
- The location of the seek or search argument; for a Seek or Search command word
- The CCW to which a transfer is made; for a transfer–in–channel command word

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

CC Command chaining

DC Data chaining between gaps of a record

SK Skip the transferring of data

SLI Suppress incorrect length indication

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

The function or purpose of each command word or group of words 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.

The following abbreviations are used in the address and count fields:

WA Work area

KL Key length

DL Data length

CF Storage area for count fields (8 x DCBHIRPD bytes)

Those BISAM or QISAM scan mode channel programs beginning with a Search ID with a count of 5 bytes are executed with a channel program prefix if a rotational position sensing (RPS) device is being used. The prefix will be a Set Sector followed by a TIC to the regular channel program. The channel command words that vary depending on the presence of RPS are shown in the following channel programs with both possible command codes.

Channel Program	Description	Mode
1	Searches cylinder and master indexes.	BISAM (all)
2	Searches a cylinder index when it is the highest–level index searched on the device.	BISAM (all)
4	Searches a track index.	BISAM (no WRITE KN)
5/5W	Searches prime–data tracks and reads or writes prime–data records.	BISAM (no WRITE KN)
6/6W	Searches an overflow chain and reads or writes overflow records.	BISAM (no WRITE KN)
7/7W	Writes data records when WRITE K is associated with READ KU.	BISAM (no WRITE KN)
8	Searches the track index and the prime–data track for place to insert new record.	BISAM (WRITE KN)
9A	Reads the record occupying the position at which a new record is to be inserted into the work area.	BISAM (WRITE KN)
9B/9BW	Reads an even–numbered record after writing a record into the previous slot and writes back the last record of a non–EOF track when the number of records bumped is odd.	BISAM (WRITE KN)
9C/9CW	Reads an odd–numbered record after writing a record into the previous slot and writes back the last record of a non–EOF track when the number of records bumped is even.	BISAM (WRITE KN)
10A/10AW	Writes a record or block to replace an EOF mark.	BISAM (WRITE KN)
10B/10BW	Writes an EOF mark.	BISAM (WRITE KN)
11A	Reads an odd–numbered record after writing a record into the previous slot.	BISAM (WRITE KN)
11B/11BW	Writes a rearranged block back onto the prime–data track.	BISAM (WRITE KN)
12A	Reads data records following slot into which new records are to be inserted.	BISAM (WRITE KN)
12AV	Records variable–length data records or blocks following point at which a new record is to be inserted.	BISAM (WRITE KN)

Figure 84 (Part 1 of 3). ISAM Channel Program Summary

Channel Program	Description	Mode
12B	Writes back prime–data records.	BISAM (WRITE KN)
12BV	Writes back variable–length prime–data records or blocks.	BISAM (WRITE KN)
12C/12CW	Writes a new record which has replaced a deleted record.	BISAM (WRITE KN)
13A	Reads all blocks from the track following and including the slot into which a record is to be inserted.	BISAM (WRITE KN)
13B	Writes back the blocks read by CP 13A after they have been rearranged.	BISAM (WRITE KN)
13C/13CW	Writes back a block if the record inserted has the same key as a record which has been logically deleted but is still physically present in the block.	BISAM (WRITE KN)
14/14W	Writes some combination of COCR, normal and over–flow track–index entries, and overflow records.	BISAM (WRITE KN)
15	Reads in the COCR and the overflow track–index entry when a new record is added to the end of a data set.	BISAM (WRITE KN)
16	Searches an overflow chain for (1) the record that logically precedes or is equal to the new record to be added or (2) the last record in the chain.	BISAM (WRITE KN)
17/17W	Changes the key in a normal or overflow–track–index entry or in a higher level index entry.	BISAM (WRITE KN)
18	Writes prime–data records or blocks.	QISAM load
19	Preformats the shared track and/or writes the COCR.	QISAM load
20	Writes track–index entries.	QISAM load
20A	Writes a full track of a nonshared track index.	QISAM load
20B	Writes a full track of a shared track index.	QISAM load
20C	Write–check for CP 20A and 20B.	QISAM load
21	Writes high–level (cylinder and master) index entries and end–of–data marks.	QISAM load
22A	Reads or writes prime–data records (key and data, unblocked records).	QISAM scan
22B	Reads or writes prime–data records (data only, unblocked records, and all blocked records).	QISAM scan
23	Searches high–level indexes, the track index, and the prime–data track when a SETL K is issued.	QISAM scan

Figure 84 (Part 2 of 3). ISAM Channel Program Summary

Channel Program	Description	Mode
24	Reads track–index entries.	QISAM scan
25	Reads track–index entries when SETL I is issued.	QISAM scan
26	Extension of CP 23 to read overflow chains	QISAM scan
31A	Reads the key of the last overflow track–index entry into the Key save area (resume loading only).	QISAM load
31B	Reads the count and data of the last prime–data block into the first buffer specified in the buffer control table (resume loading only).	QISAM load
87	Reads the highest level index into the user work area (specified by DCBMSHI).	BISAM (all)
91	Fills unused index tracks with inactive and dummy (end–of–index) entries (same as CP 19).	QISAM load
123W	Extension of CP 12A and CP 12B or CP 13A and CP 13B when write–checking is specified.	BISAM (WRITE KN)
123WV	Extension of CP 12AV and CP 12BV when write–checking is specified.	BISAM (WRITE KN)
CLOSE CCW(1)	Reads the format–2 DSCB for updating by close phase.	Common Close
CLOSE CCW(2)	Writes the format–2 DSCB back in the volume table of contents (VTOC).	Common Close
VXCCW (1A)	Reads to the end of the file or the end of the last track in the prime–data area.	Common Open (validation)
VXCCW (1B)	Reads to end of file of independent overflow area.	Common Open (validation)
VXCCW (2)	Reads to the end of the prime–data track.	Common Open (validation)

Figure 84 (Part 3 of 3). ISAM Channel Program Summary

CHANNEL PROGRAM 1

Searches cylinder and master indexes								
CCW No.	Command Code		Address	Flags		Count	Comments	
	Hex	Description		Hex	Description			
C01	31	Search ID equal	IOBSEEK+3	60	CC, SLI	4	Search for equal CCHH to verify seek—IOBSEEK set from either DCBFTHI or index entry in main storage	
C02	08	TIC	C01	00		0		
C1	69	Search key high or equal	Contents of DECBKEY	60	CC, SLI	KL	Too far along index?	Search for master index entry
C2	08	TIC	C4	00		0	No	
C2B	03 23	NOP Set sector	C2B+5	60	CC,SLI	1	Set sector to zero	
C3	1A	Read home address	C8	50	CC, SK	5	Yes, position to start of track	
C4	E9	Search key high or equal (MT)	Contents of DECBKEY	40	CC	KL	Search for entry	
C5	08	TIC	C4	00				
C6	06	Read data	C8+7	00 40	CC (lowest master)	10	When found, read master index, CC off if lower level master index is to be searched	
C7	08	TIC	C10	00		0	Go search cylinder index	
C8	— — — — — — — M						Master index entry—IOBSEEK set to C8+7 when this CP is restarted for lower level master index	
C9	B B C C H H R F							
C10	P	Seek	C9	40	CC	6	Seek cylinder index (Figure 83)	
C10A	31	Search ID equal	C9+2	40	CC	4	Search for equal CCHH to verify seek	
C10B	08	TIC	C10A	40	CC	0		
C11	69	Search key high or equal	Contents of DECBKEY	40	CC	KL	Too far along index?	Search for cylinder index entry
C12	08	TIC	C14	00		0	No	
C12B	03 23	NOP Set sector	C12B+5	60	CC, SLI	1	Set sector to zero	
C13	1A	Read home address	C8	50	CC, SK	5	Position to start of track	
C14	E9	Search key high or equal (MT)	Contents of DECBKEY	40	CC	KL	Search for entry	
C15	08	TIC	C14	00		0		

CHANNEL PROGRAM 1 (continued)

Searches cylinder and master indexes							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
C16	06	Read data	C17	00		DL	Read in cylinder index entry
C17	M B B C C H H R						Cylinder index entry—IOBSEEK for CP4 set to C17
C18	F P — — — — — —						

CHANNEL PROGRAM 2

Searches a cylinder index when it is the highest level index searched on the device

CCW No.	Command Code		Address	Flags		Count	Comments	
	Hex	Description		Hex	Description			
C28	31	Search ID equal	IOBSEEK+3	60	CC, SLI	4	Search for equal CCHH to verify seek—IOBSEEK set from either DCBFTHI or index entry in main storage	
C29	08	TIC	C28	00		0		
C30	69	Search key high or equal	Contents of DECBKEY	60	CC, SLI	KL	Too far along index?	Search for cylinder index entry
C31	08	TIC	C33	00		0	No	
C31B	03 23	NOP Set sector	C31B+5	60	CC, SLI	1	Set sector to zero	
C32	1A	Read home address	C37	50	CC, SK	5	Yes, position to start of track	
C33	E9	Search key high or equal (MT)	Contents of DECBKEY	40	CC	KL	Search for entry	
C34	08	TIC	C33	00		0		
C35	06	Read data	C36	00		10	Read in cylinder index entry.	
C36			M B B C C H H R				Cylinder index entry—IOBSEEK set to C36 when CP4 is executed	
C37			F P -- — — — — —					

CHANNEL PROGRAM 4

Searches a track index								
CCW No.	Command Code		Address	Flags		Count	Comments	
	Hex	Description		Hex	Description			
CA01	31	Search ID equal	IOBSEEK+3	60	CC	4	Search for equal CCHH to verify seek–IOBSEEK set from C17 (CP1), C36 (CP2), DCBFTHI or entry in main storage	
CA02	08	TIC	CA01	Address of CP5 in CP 4-5-6 chain (see Figure 55)				
CA03	08	TIC	CA1 or CA5	00		0	TIC to CA1 if shared track is present. Otherwise, TIC to CA5.	
CA1	71	Search ID high or equal	IOBSEEK+3	40	CC	5	In prime data part of track?	Search track index
CA2	08	TIC	CA5	00		0	No	
CA4	08	TIC	CA7 or CA6B	00		0	Yes	
CA5	69	Search key high or equal	Contents of DECBKEY	60	CC, SLI	KL	Too far along in index?	
CA6	08	TIC	CA8	00		0	No	
CA6B	03 23	NOP Set sector	CA6B+5	60	CC, SLI	1	Set sector to zero	
CA7	1A	Read home address		50	CC, SK	5	Yes, position to start of track	
CA8	E9	Search key high or equal (MT)	Contents of DECBKEY	40	CC	KL	Search for entry	
CA9	08	TIC	CA8	00		0		
CA10	06	Read data	CA12+7	40	CC	10	If found, read index entry	
CA11	08	TIC	CA14	00		0		
CA12			– – – – – – – M				Track index entry	
CA13			B B C C H H R F					
CA14	P	Seek	CA13	40	CC (to CP5)	6	**Seek prime-data track (see Figure 81)**	

CHANNEL PROGRAM 5/5W

Searches prime data tracks and reads or writes prime data records							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CA15	23 03	Set sector NOP	CA15+5	60	CC, SLI	1	Position to beginning of track if RPS device. Set sector to zero if RPS.
CA16A	31	Search ID equal	CA13+2	40	CC	5	Search past index on shared track or past R0 on normal track. Should be RHA and TIC to CA20 for VLR.
CA16B	08	TIC	CA16A	00		0	
CA16C	08	TIC	CA21	00		0	Avoid read count of FIRSH+1. (CA25+3 set to FIRSH prior to execution.)
CA20	12	Read count	CA25+3	60	CC, SLI	5	Read count of record (see CA25)
CA21	29 69	Search key equal Search key equal or high	Contents of DECBKEY	60	CC, SLI	KL	Search (29) if Read, Records Unblocked, or Write. Search (69) if Read, Records Blocked.
CA22	08	TIC	CA20	00		0	
CA23	06 05	Read data Write data	Contents of DECBAREA	40	CC	DL	Read prime data or write prime data
CA24	03 22	NOP Read sector	IOBSECT	60	CC, SLI	1	Obtain address of record just read or written. No CC if read.
CA240*	03 23	Set sector	IOBSECT	40	CC	1	
CA24A*	31	Search ID equal	CA25+3	40	CC	5	Search for record again
CA24B*	08	TIC	CA24A	00		0	
CA24C*	06	Read data		10	SK	DL	Read it back
CA24D*	31	Search ID equal	IOBSEEK+3	40	CC	5	Rewrite record if necessary
CA24E*	08	TIC	CA24D	00		0	
CA24F*	05	Write data	Contents of DECBAREA	40	CC	DL	
CA24G*	08	TIC	CA24A or CA240	40	CC	0	Write check again
CA25		— — — C C H H R					If Read KU, CHHR of count is moved into IOBSEEK+4 (without destroying MBBC in IOBSEEK) when record is written back (CP7)

*Write Validity Check

CHANNEL PROGRAM 6/6W

Searches an overflow chain and reads or writes overflow records								
CCW No.	Command Code		Address	Flags		Count	Comments	
	Hex	Description		Hex	Description			
CA26*	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for first record in overflow chain—IOBSEEK set from CA12+7 (CP4)	
CA27	08	TIC	CA26	00		0		
CA28	69	Search key equal or high					RKP=0 and blocked or RKP≠0; read	
	29	Search key equal	Contents of DECBKEY	40	CC	0	Check key in overflow record. If equal, read (CA31) or write (CA40) record; otherwise, go to next one in chain	
CA29	08	TIC	CA32	00		0		
CA30	08	TIC	CA31 CA40	00		0		
CA31	06	Read data	Contents of DECBAREA (+6)	00	**	DL40	Read the overflow record (end of CP)	
CA31B	22	Read sector	IOBSECT	00		1		
CA32	06	Read data	CA34+7	60	CC, SLI	10	Read link field to next record	
CA33	08	TIC	CA36	00		0		
CA34	— — — — — — — — M						Link field from overflow entry	
CA35	B B C C H H R F							
CA36	P(07)	Seek	CA35	40	CC	6	Seek next record in overflow chain (see Figure 81 for value of P—seek command code)	
CA36B	03 23	NOP Set sector	IOBSECT	60	CC, SLI	1	NOP if CP unbroken. Set sector if stop at CA32 or CA30 (estimate if VLR).	
CA37	31	Search ID equal	CA35+2	40	CC	5	Search for overflow record	
CA38	08	TIC	CA37	00		0		
CA39	08	TIC	CA28	00		0	If found, check key	
CA40	06	Read data	Contents (+6) of DECBAREA	60	CC, SLI	10	Read link field	Write overflow record
CA40A	08 22	TIC Read sector	CA41 IOBSECT	40	CC	1	Position to record again	
CA40B	23	Set sector	IOBSECT	40	CC	1		

*This channel program is preceded by a set sector—TIC if RPS is present. This prefix is located in the IOB extension.

**CC if RPS

(continued)

CHANNEL PROGRAM 6/6W (continued)

Searches an overflow chain and reads or writes overflow records

CCW No.	Command Code		Address	Flags		Count	Comments	
	Hex	Description		Hex	Description			
CA41	31	Search ID equal	CA35+2	40	CC	5	Position to record again	Write overflow record
CA42	08	TIC	CA41	00		0		
CA43	05	Write data	Contents (+6) of DECBAREA	40	CC		Write record	
CA430*	03 23	NOP Set sector	IOBSECT	60	CC, SLI	1	Reposition to correct record	
CA43A*	31	Search ID equal	CA35+2	40	CC	5	Find record again	
CA43B*	08	TIC	CA43A	00		0		
CA43C*	06	Read data		10	SK	0	Read it back	

*Write Validity Check

CHANNEL PROGRAM 7/7W

Writes data records when WRITE K is associated with READ KU							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CA44*	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for record to be updated—See CA25 (CP5)
CA45	08	TIC	CA44		Address of next CP7 in queue (see Figure 55)		
CA46	05	Write data	Contents of DECBAREA	40	CC	DL	Write updated record
CA46O**	03 23	NOP Set sector	IOBSECT	60	CC,SLI	1	
CA46A**	31	Search ID equal	IOBSEEK+3	40	CC	5	Find record again
CA46B**	08	TIC	CA46A	00		0	
CA46C**	06	Read data		10	SK		Read it back

*This channel program is preceded by a prefix if RPS is present. The prefix consists of a set sector and TIC which are located in the IOB extension.

**Write Validity Check

CHANNEL PROGRAM 8

Searches track index and prime data track to determine first record to be moved and position to insert it.							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CB1*	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for (COCR) R0
CB2	08	TIC	CB1	00		0	
CB3	06	Read data	CB22	60	CC, SLI	6	Read R0 COCR (HHRYYT) into CB22
CB4	92	Read count (MT)	CB22+6	60	CC, SLI	5	Read count of index entry
CB5	69	Search key equal or high	Contents of DECBKEY	40	CC	KL	Search for index entry
CB6	08	TIC	CB4	00		0	
CB7	06	Read data	CB10+7	40	CC	10	Read data of track index entry
CB8	92	Read count (MT)	CB24	40	CC	8	Read count of following entry
CB8A	06	Read data	CB25	40	CC**	10	Read data of next entry
CB9	08	TIC	CB12	00		0	
CB10	– – – – – – – M						Track-index entry contains search address for prime or overflow data
CB11	B B C C H H R F						
CB12	P	Seek	CB11	40	CC	6	Seek prime or overflow track. See Figure 81 for value of P (Seek Command Code).
CB16	03 23	NOP Set sector	CB16+5	60	CC, SLI	1	Position to beginning of track if RPS. Set sector to 0 if RPS.
The following versions of CB17-CB20 are used with fixed-length records							
CB17	31	Search ID equal	CB11+2	40	CC	5	Search for prime record
CB18	08	TIC	CB17	00		0	
CB18A	08	TIC	CB19	00		0	Avoid skipping first record
CB18B	12	Read count	CB23+3	60	CC,SLI	5	Get count of insertion record
CB19	69	Search key equal or high	Contents of DECBKEY	60	CC,SLI	KL	Search track for insertion block
CB20	08	TIC	CB18B	00		0	

*This channel program is preceded by . . . IOB extension.

CHANNEL PROGRAM 8– (Continued)

Searches track index and prime data track to determine first record to be bumped and place to insert it.							
CCW No.	Command Code Hex	Command Code Description	Address	Flags Hex	Flags Description	Count	Comments
The following versions of CB17-CB20 are used with variable-length records							
CB17	16	Read home address	0	70	CC,SK,SLI	1	Position to beginning of track
CB18	08	TIC	CB18B	00		0	Avoid skipping first record
CB18A	06	Read data	WATKL	60	CC,SLI	0	Read in block prior to insertion block
CB18B	12	Read count	CB23+3	60	CC,SLI	5	Get count, probable insertion block
CB19	69	Search key equal or high	Contents of DECBKEY	40	CC	KL	Search for probable insertion block
CB20	08	TIC	CB18A	00		0	
CB21	03 22	NOP Read sector	IOBSECT	20	SLI	1	Read insert-block sector for RPS
CB22	H H R Y Y T C C						COCR--start of count of index entry
CB23	H H R C C H H R						Finish count of index entry and count of record after insertion (record to be bumped)
CB24		C C H H R KL DL DL					Count of the index entry following the entry that meets the search conditions
CB25		M B B C C H H R					Data field of the index entry following the entry that meets the search conditions
CB26		F P – – – – – –					

CHANNEL PROGRAM 9A

Read into work area an unblocked record occupying the position at which an insertion is to be made							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CB30	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for record
CB31	08	TIC	CB30	00		0	
CB32	0E	Read key and data	WA	80	DC	KL	Read record into work area
CB33	00		WA+KL+16	00		DL	

CHANNEL PROGRAM 9B/9BW

Reads an even numbered record after writing a record into the previous slot and writes back the last record of a non-EOF track when the number of records bumped is odd.							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CB34*	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for record
CB35	08	TIC	CB34	00		0	
CB36	0D	Write key and data	Contents of DECBKEY	80	DC	KL	Write new record or record pointed to by DECB
CB37	00		Contents of DECBAREA	00		DL	
CB370**	03 23	NOP Set sector	IOBSECT	60	CC, SLI	1	
CB37A**	31	Search ID equal		40	CC	5	Search for record again
CB37B**	08	TIC	CB37A	00		0	
CB37C**	0E	Read key and data		10	SK	KL+DL	Read it back
CB38	0E	Read key and data	Contents of DECBKEY	80	DC	KL	Read next record
CB39	00		Contents of DECBAREA	00		DL	

*This channel program is preceded by a set sector—TIC if RPS is present. This prefix is located in the IOB extension.
**Write Validity Check

CHANNEL PROGRAM 9C/9CW

Reads an odd numbered record after writing a record into the previous slot and writes back the last record of a non-EOF track when the number of records bumped is even.

CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CB40*	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for record
CB41	08	TIC	CB40	00		0	
CB42	0D	Write key and data	WA	80	DC	KL	Write record into work area
CB43	00		WA+KL+16	00		DL	
CB430**	03 23	NOP Set sector	IOBSECT	60	CC, SLI	1	
CB43A**	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for record again
CB43B**	08	TIC	CB43A	00		0	
CB43C**	0E	Read key and data		10	SK	KL+DL	Read it back
CB44	0E	Read key and data	WA	80	DC	KL	Read record and point DECB to that area
CB45	00		WA+KL+16	00		DL	

*This channel program is preceded by a set sector—TIC if RPS is present. This prefix is located in the IOB extension.
**Write Validity Check

CHANNEL PROGRAM 10A/10AW

Writes a record or block to replace an EOF mark							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CB46*	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for last data record
CB47	08	TIC	CB46	00		0	
CB48	ID	Write count, key, and data	CB51	80	DC	8	Write record or block over EOF mark
CB49	00		Contents of DECBKEY	80	DC	KL	
CB50	00		WA+KL+16	40	CC	DL	
CB500**	03 23	NOP Set sector	IOBCCW2+4	60	CC, SLI	1	
CB50A**	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for record again
CB50B**	08	TIC	CB50A	00		0	
CB50C**	1E	Read count, key, and data		10	SK	8+KL +DL	Read it back
CB51	C C H H R KL DL DL						Count of record or block which replaces EOF

*This channel program is preceded by a set sector–TIC if RPS is present. This prefix is located in the IOB extension.

**Write Validity Check.

CHANNEL PROGRAM 10B/10BW

Writes an EOF mark							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CB52*	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for last data record
CB53	08	TIC	CB52	00		0	
CB54	1D	Write count, key, and data	CB55	40	CC	8	Write EOF mark
CB540**	03 23	NOP Set sector	IOBSECT	60	CC, SLI	1	Search for EOF mark
CB54A**	31	Search ID equal	IOBSEEK+3	40	CC	5	
CB54B**	08	TIC	CB54A	00		0	
CB54C**	1E	Read count, key, and data		10	SK	8	Read it back
CB55	C C H R R 0 0 0						EOF mark (count field)

*This channel program is preceded by a set sector–TIC if RPS is present. This prefix is located in the IOB extension.
**Write Validity Check

CHANNEL PROGRAM 11A

Reads an odd numbered record after writing a record into the previous slot							
CCW No	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CC1	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for block
CC2	08	TIC	CC1	00		0	
CC2A	0E	Read key and data	WA	80	DC	KL	Read in block
CC3	00		WA+KL+RL	00		DL	

CHANNEL PROGRAM 11B/11BW

Writes a re-arranged block back onto the prime data track							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CC4*	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for insertion point
CC5	08	TIC	CC4	00		0	
CC6	0D	Write key and data	WA	40	CC	KL+DL	Write block
CC60*	03 23	NOP Set sector	IOBSECT	60	CC, SLI	1	
CC6A*	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for block again
CC6B*	08	TIC	CC6A	00		0	
CC6C*	0E	Read key and data		10	SK	KL+DL	Read it back

*This channel program is preceded by a set sector - TIC if RPS is present. This prefix is located in the IOB extension.
**Write Validity Check

CHANNEL PROGRAM 12A

Reads data records following slot in which new record is to be inserted							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CD1	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for block prior to insert
CD2	08	TIC	CD1	00		0	
CD3	0E	Read key and data	WA+10	60	CC, SLI	KL+DL	Read first prime data block
CD4	1E	Read count, key, and data	WA+10+ KL+DL	60	CC, SLI	DL	Read successive prime data record. There is one copy of CD4 for each record on a prime data track; the CC bit is set off in the appropriate copy depending on how many blocks are to be read.

CHANNEL PROGRAM 12AV

Reads variable length data records or blocks following point at which new record is to be inserted							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CD0	C C H H R 0 0 8						Capacity record for prime data track
CD0A	Y Y R – – – – –						
CD0A1*	31	Search ID equal	CD0	40	CC	5	Search for R0 (track capacity record)
CD0A2	08	TIC	CD0A1	00		0	
CD0B	06	Read data	CD0A	60	CC, SLI	3	Read capacity record
CD0C	08	TIC	CD0D or CD3	00		0	TIC to CD3 if a full track is to be read or prior block full
CD0D	03 23	NOP Set sector	IOBSECT+1	60	CC, SLI	1	
CD1	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for record prior to insert point
CD2	08	TIC	CD1	00		0	
CD2A	08	TIC	CD2B or CD3	00		0	TIC to CD2B if this is first execution of channel program**
CD2B	0E	Read key and data	WA	60	CC, SLI	KL	Read key of record prior to insert point
CD3	06	Read data	WA+KL+CF +LRECL	60	CC,SLI	DL	Read data portion of record. There is one copy of CD3 for each record which can be read in a single execution.*

*This channel program is preceded by a set sector-TIC if RPS is present. This prefix is located in the IOB extension.

**With unblocked records and a large HIRPD, the WRITE KN work area (DCBMSWA) may not be large enough to contain all records past the insertion point. CP 12AV is then executed more than once. "ISAM Buffer and Work AREA Requirements" in Data Management Services Guide, GC26-3746, tells how to determine the best size for the work area.

CHANNEL PROGRAM 12B

Writes back prime data records							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CE1	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for block prior to insert
CE2	08	TIC	CE1	00		0	
CE3	1D	Write count, key, and data	WA+2	80	DC	8	Write prime data records. There is one set of CE6-CE7 for each record on a prime data track; the CC bit is set off in the appropriate copy of CE7 depending on how many records are written back.
CE4	00		DECBKEY	80	DC	KL	
CE5	00		DECBAREA	40	CC	DL	
CE6	1D	Write count, key, and data	WA+KL+ DL+10	80	DC	8	
CE7	00		WA+10	40	CC	KL+DL	

CHANNEL PROGRAM 12BV

Writes back variable length prime data records or blocks							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CE0*	31	Search ID equal	CD0	40	CC	5	Search for R0
CE0A	08	TIC	CE0	00		0	
CE0B	05	Write data	CD0A	60	CC, SLI	3	Write updated track capacity record
CE0C	03 23	NOP Set sector	IOBSECT+1	60	CC, SLI	1	Search for record prior to insert point
CE1	31	Search ID equal	IOBSEEK+3	40	CC	5	
CE2	08	TIC	CE1	00		0	
CE3	08	TIC	CE4	00		0	TIC to CE4 to write partial track
CE3A	39	Search home address	CD0	40	CC	4	Search for start of track
CE3B	08	TIC	CE3A	00		0	
CE3C	15	Write R0	CD0	60	CC, SLI	11	Write updated track capacity record again
CE4	1D	Write count, key, and data	WA+KL	80	DC	8	Write prime data record. The number of sets of CE4-CE6 equals DCBHIRPD; the CC bit is set off in the appropriate copy of CE6 depending on how many records are written back
CE5	00		WA+KL+CF +(DL-LRECL) +RKP	80	DC	KL	
CE6	00		WA+KL+CF	40	CC	DL	

*This channel program is preceded by a set sector—TIC if RPS is present. The prefix is located in the IOB extension.

CHANNEL PROGRAM 12C/12CW

Writes a new record which has replaced a deleted record							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CL1*	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for deleted record
CL2	08	TIC	CL1	00		0	
CL3	05	Write data	Contents of DECBAREA	40	CC	DL	Replace deleted record
CL30**	03 23	NOP Set sector	IOBSECT	60	CC, SLI	1	Search for record again
CL3A**	31	Search ID equal	IOBSEEK+3	40	CC	0	
CL3B**	08	TIC	CL3A	00		0	
CL3C**	06	Read data		10	SK	DL	Read it back

*This channel program is preceded by a set sector - TIC if RPS is present. This prefix is located in the IOB extension.
**Write Validity Check

CHANNEL PROGRAM 13A

Reads all blocks from the track following and including the slot into which a record is to be inserted							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CF1	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for first record to be read
CF2	08	TIC	CF1	00		0	
CF3	06	Read data	Data address	00		DL	Read first prime data block
CF4	12	Read count	WA	40	CC	8	Read successive prime data block. There is one copy of CF4-CF5 for each block on a prime data track; the CC bit is set off in the appropriate copy of CF5 depending on how many blocks are to be read.
CF5	06	Read data	Data address	40	CC	DL	

CHANNEL PROGRAM 13B

Writes back the rearranged blocks read by CP13A							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CG1	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for record before insertion point
CG2	08	TIC	CG1	00		0	
CG3	1D	Write count, key, and data	WA	80	DC	8	Write back prime data block. There is one copy of CG3-CG4-CG5 for each block on a prime data track; the CC bit is set off in the appropriate copy of CF5 depending on how many blocks are to be written.
CG4	00		Key address	80	DC	KL	
CG5	00		Data address	00		DL	

CHANNEL PROGRAM 13C/13CW

Writes back a block if the insertion is a record with a key identical to that of a record, which although logically deleted, is still physically present within the block.							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CL5*	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for block insertion point
CL6	08	TIC	CL5	00		0	
CL7	05	Write data	Data address	40	CC	DL	Replace block
CL70**	03 23	NOP Set sector	IOBSECT	60	CC, SLI	1	Find record again
CL7A**	31	Search ID equal	IOBSEEK+3	40		5	
CL7B**	08	TIC	CL7A	00		0	
CL7C**	06	Read data		10	SK	DL	Read it back

*This channel program is preceded by a set sector—TIC if RPS is present. The prefix is located in the IOB extension.
**Write Validity Check

CHANNEL PROGRAM 14/14W – Fixed Length Records

Writes some combination of COCR, normal and overflow track index entries, and overflow records. (See *BISAM Write KN Asynchronous Codes* in Section 6 for descriptions of the setups of this channel program.)

Part II–Rewrites COCR and track index *

CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CH1**	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for COCR Entry point for Setups 1-5 (add to cylinder overflow)
CH2	08	TIC	CH1	00			
CH3	05	Write data	CB22	60	CC, SLI	6	Write updated COCR from CP8
CH3A1***	23 03	Set sector NOP	CH3A1+5	60	CC, SLI	1	Set sector to zero if RPS
CH3A***	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for COCR again
CH3B***	08	TIC	CH3A	00		0	
CH3C***	06	Read data		70	CC, SK, SLI		Read it back
CH4	08	TIC	CH5, CH9, CH55, CH14, or CH8D	00		0	TIC to CH5 for Setup 1, CH9 for Setups 2, 3 5; CH14 for Setup 4
CH5	03 23 1B	NOP Set sector Seek head	IOBSECT CI5	60	CC, SLI	6	
CH55	31	Search ID equal	CB22+6	40	CC	5	Search for prime index entry; entry point for Setups 1-2 (add to independent overflow)
CH6	08	TIC	CH55	00		0	
CH7	0D	Write key and data	Contents of DECBKEY	80	DC	0	Write new hi-key prime data chain
CH8	00		CB10+7	40	CC	10	Write prime index entry
CH80***	03 23	NOP Set sector	IOBSECT	60	CC, SLI	1	
CH8A***	31	Search ID equal	CB22+6	40	CC	5	Search for entry again
CH8B***	08	TIC	CH8A	00		0	
CH8C***	0E	Read key and data		50	CC, SK	0	Read it back
CH8D	31	Search ID equal	CB24	40	CC	5	Search for overflow track index entry
CH8E	08	TIC	CH8D	00		0	
CH8F	05	Write data	CB25	10	SK	10	
CH8G	08	TIC	CH13+8	00		0	

*CP14 is executed in two parts only when the work area is provided by the user.
**This channel program is preceded by a set sector–TIC if RPS is present. This prefix is located in the IOB extension.
***Write Validity Check

(Continued)

CHANNEL PROGRAM 14/14W – Fixed Length Records (continued)

Writes some combination of COCR, normal and overflow track index entries, and overflow records (See *BISAM Write KN Asynchronous Codes* in Section 6 for descriptions of the setups of this channel program.)

CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
			Part II—Rewrites COCR and track index **				
CH9	03 23	NOP Set sector	IOBSECT+1	60	CC, SLI	1	Search overflow track index entry
CH95	31	Search ID equal	CB24	40	CC	5	
CH10	08	TIC	CH95	00		0	
CH12	0D	Write key and data		80	DC	0	Write new overflow key-data chain
CH13	05	Write data	CB25	40	CC	10	Write overflow index entry
CH130*	03 23	NOP Set sector	IOBSECT+1	60	CC, SLI	1	Search for entry again
CH13A*	31	Search ID equal	CB24	40	CC	5	
CH13B*	08	TIC	CH13A	00		0	
CH13C*	0E	Read key and data		50	CC, SK	KL+DL	Read it back
CH14	07 0B 1B 03	Seek Seek cylinder Seek head NOP	CH23+1	40	CC		Seek new overflow record (seek is set by appendage routine). For user work area this CCW is a NOP.
			Part I—Writes overflow record.**				
CH150	03 23	NOP Set sector	IOBSECT+2	60	CC, SLI	1	Entry point for Setup 6
CH15	31	Search ID equal	CH23+3	40	CC	5	Search for overflow slot
CH15A	08	TIC	CH15	00		0	
CH16	1D	Write count, key, and data	CH24	80	DC	8	Write new overflow record
CH17	00		Contents of DECBKEY	80	DC	KL	
CH18	00		Contents of DECBAREA	40	CC	DL	
CH180*	03 23	NOP Set sector	IOBSECT+2	60	CC, SLI	1	Search for new overflow record again
CH18A*	31	Search ID equal	CH23+3	40	CC	5	
CH18B*	08	TIC	CH18A	00		0	
CH18C*	1E	Read count, key, and data		10	SK	0	Read it back. Termination for Setups 1, 2, 5, 6
CH19	07 0B 1B	Seek Seek cylinder Seek head	CJ11+1	40	CC	6	Seek previous overflow record (appropriate seek set by appendage routine).

*Write Validity Check
**CP14 is executed in two parts only when the work area is provided together

(continued)

CHANNEL PROGRAM 14/14W – Fixed Length Records (continued)

Writes some combination of COCR, normal and overflow track index entries, and overflow records. (See *BISAM Write KN Asynchronous Codes* in Section 6 for descriptions of the Setups of this channel program.)

CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CH200	03 23	NOP Set sector	IOBSECT+3	60	CC, SLI	1	Search for record
CH20	31	Search ID equal	CJ11+3	40	CC	5	
CH21	08	TIC	CH20	00		0	
CH22	05	Write data	WA	40	CC	0	Write back previous overflow record
CH220*	03 23	NOP Set sector	IOBSECT+3	60	CC, SLI	1	Search for previous overflow record again
CH22A*	31	Search ID equal	CJ11+3	40	CC	5	
CH22B*	08	TIC	CH22A	00		0	
CH22C*	06	Read data		10	SK	DL	Read it back. Termination for Setups 3-4.
CH23	M B B C C H H R						Search address of new overflow record
CH24	C C H H R KL DL DL						Count of new overflow

*Write Validity Check

CHANNEL PROGRAM 14/14W—Variable Length Records

Writes some combination of COCR, normal and overflow track index entries, and overflow records. (See BISAM Write KN Asynchronous Codes in Section 6 for descriptions of the Setups of this channel program.)

CCW No.	Command Code Hex	Command Code Description	Address	Flags Hex	Flags Description	Count	Comments
			Part II—Rewrites COCR and Track Index				
CH1*	31	Search ID equal	CH23+3	40	CC	5	Search for COCR Entry point for Setups 1-5 (add to cylinder overflow)
CH2	08	TIC	CH1	00			
CH3	05	Write data	CB22	60	CC, SLI	6	Write updated COCR from CP8
CH3A1*	23 03	Set sector NOP	CHA1+5	60	CC, SLI	1	Set sector to zero if RPS
CH3A**	31	Search ID equal	CH23+3	40	CC	5	Search for COCR again
CH3B**	08	TIC	CH3A	00		0	
CH3C**	06	Read data		70	CC, SK, SLI		Read it back
CH4	08	TIC	CH50, CH5, CH3F0,CH3GV or CH14	00		0	TIC to CH5 for Setup 1; CH8G for Setups 2, 3, 5; CH14 for Setup 4
CH5	03 23 1B	NOP Set sector Seek head	IOBSECT CI5	60	CC, SLI	6	
CH55	31	Search ID equal	CB22+6	40	CC	5	Search for prime index entry; Entry point for Setups 1-2 (add to independent overflow)
CH6	08	TIC	CH55	00		0	
CH7	0D	Write key and data	Contents of DECBKEY	80	DC	0	Write new hi-key prime data chain
CH8	00		CB10+7	40	CC	10	Write prime index entry
CH80**	03 23	NOP Set sector	IOBSECT	60	CC, SLI	1	
CH8A**	31	Search ID equal	CB22+6	40	CC	5	Search for entry again
CH8B**	08	TIC	CH8A	00		0	
CH8C**	0E	Read key and data		50	CC, SK	0	Read it back

*This channel program is preceded by a prefix if RPS is present. The prefix consists of a set sector and TIC, which are located in the IOB extension.

**Write Validity Check

(continued)

CHANNEL PROGRAM 14/14W—Variable Length Records (continued)

Writes some combination of COCR, normal and overflow track index entries, and overflow records. (See BISAM Write KN Asynchronous Codes in Section 6 for descriptions of the setups of this channel program.)

CCW No.	Command Code Hex	Command Code Description	Address	Flags Hex	Flags Description	Count	Comments
CH8D	08	TIC	CH8G5	00		0	
CH8F							This CCW not used
Part II—Rewrites COCR and Track Index							
CH8G	23 03	Set sector NOP	IOBCCW2+5	60	CC,SLI	1	Search overflow track index entry
CH8G5	31	Search ID equal	CB24	40	CC	5	
CH9	08	TIC	CH8G5	00		0	
CH10	08	TIC	CH12 or CH13	00		0	TIC to CH13 to write data only of overflow record
CH12	0D	Write key and data		80	DC	0	Write new overflow key-data chain
CH13	05	Write data	CB25	40	CC	10	Write overflow index entry
CH130*	03 23	NOP Set sector	IOBSECT+1	60	CC, SLI	1	Search for entry again
CH13A*	31	Search ID equal	CB24	40	CC	5	
CH13B*	08	TIC	CH13A	00		0	
CH13C*	0E	Read key and data		50	CC, SK	KL+10	Read it back
CH14	03	NOP		20	SLI	1	

*Write Validity Check

(continued)

CHANNEL PROGRAM 14/14W—Variable Length Records (continued)

Writes some combination of COCR, normal and overflow track index entries, and overflow records. (See BISAM Write KN Asynchronous Codes in Section 6 for descriptions of the setups of this channel program.)

CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
Part I—Writes Overflow Record							
CH150	03 23	NOP Set sector	IOBSECT+2		CC, SLI		
CH15	31	Search ID equal	CH23+3	40	CC	5	Search for overflow slot
CH15A	08	TIC	CH15	00		0	
CH16	1D	Write count, key, and data	CH24	80	DC	8	Write new overflow record
CH17	00		Contents of DECBKEY	80	DC	KL	
CH18	00		Contents of DECBAREA	40	CC	DL	
CH180*	03 23	NOP Set sector	IOBSECT+2	60	CC, SLI	1	Search for new overflow record again
CH18A*	31	Search ID equal	CH23+3	40	CC	5	
CH18B*	08	TIC	CH18A	00		0	
CH18C*	1E	Read count, key, and data		10	SK	0	Read it back. Termination for Setups 1, 2, 5, 6
CH19	07 0B 1B	Seek Seek cylinder Seek head	CJ11+1 SECT+3	40	CC	6	Seek previous overflow record (appropriate seek set by appendage routine).
CH200	03 23	NOP Set sector	IOBCCW2+7	60	CC, SLI	1	Search for record
CH20	31	Search ID equal	CJ11+3	40	CC	5	
CH21	08	TIC	CH20	00		0	
CH22	05	Write data	WA	40	CC	0	Write back previous overflow record
CH220*	03 23	NOP Set sector	IOBSECT+3	60	CC, SLI	1	Search for previous overflow record again
CH22A*	31	Search ID equal	CJ11+3	40	CC	5	
CH22B*	08	TIC	CH22A	00		0	
CH22C*	06	Read data		10	SK	DL	Read it back. Termination for Setups 3-4

*Write Validity Check

(continued)

CHANNEL PROGRAM14/14W–Variable Length Records (continued)

CH23	M B B C C H H R						Search address of new overflow record
Writes some combination of COCR, normal and overflow track index entries, and overflow records. (See BISAM Write KN Asynchronous Codes in Section 6 for descriptions of the Setups of this channel program.)							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CH24	C C H H R KL DL DL						Count of new overflow
EOF Extension							
CH25	31	Search ID equal	CH31+3	40	CC	5	Search for last overflow record
CH26	08	TIC	CH25	00		0	
CH27	1D	Write count, key, and data	CH32	40	CC	8	Write EOF mark
CH280*	03 23	NOP Set sector	IOBSECT	60	CC, SLI	1	Search for record again
CH28*	31	Search ID equal	CH31+3	40	CC	5	
CH29*	08	TIC	CH28	00		0	
CH30*	1E	Read count,key, and data		30	SK, SLI	8	Read it back
CH31	M B B C C H H R						Address of last overflow record
CH32	C C H H R K K D						EOF mark

*Write Validity Check

CHANNEL PROGRAM 15

Reads in the cylinder overflow control record and the overflow track index entry when a new record is added to the end of a data set

CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CI1*	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for COCR
CI1A	08	TIC	CI1	00		0	
CI1B	06	Read data	CB22	60	CC, SLI	6	Read R0 (COCR) into CP8
CI1C	1B	Seek head	CI5	40	CC	6	Find last active index track
CI1D	03 23	NOP Set sector	IOBSECT+1	60	CC, SLI	1	Search for last active normal track index entry
CI1E	31	Search ID equal	CI5+2	40	CC	5	
CI2	08	TIC	CI1E	00		0	
CI3	92	Read count	CB24	40	CC	8	Read count of last overflow entry into CP8
CI4	06	Read data	CB25	00		10	Read data of last overflow entry into CP8
CI5	B B C C H H R –						ID of last active normal track index entry

*This channel program preceded by a set sector—TIC if RPS is present. This prefix is located in the IOB extension.

CHANNEL PROGRAM 16

Searches an overflow chain for (1) the record that logically precedes or is equal to the new record to be added or (2) the last record in the chain.

CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CJ1**	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for next overflow record in chain
CJ2	08	TIC	CJ1	00		0	
CJ3	69	Search key equal or high	Contents of DECBKEY	40	CC	KL	Is this the desired record?
CJ4	08	TIC	CJ10	00		0	No
CJ4A	03 23	NOP Set sector	IOBSECT	60	CC, SLI	1	
CJ5	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for overflow record
CJ6	08	TIC	CJ5	00		0	
CJ7	29	Search key equal	Contents of DECBKEY	40	CC	0	Test if key equals user key
CJ8	03	NOP	0	20	SLI	1	No, stop here
CJ9	06	Read data	WA	20	SLI	11	Yes, read 11 bytes of equal key record
CJ10	06	Read data	WA*	00†		DL +10***	Read next overflow record in chain
CJ11	M B B C C H H R						Address of record in chain before insert

*The address is WA+20 for variable length records

**This channel program preceded by a prefix if RPS is present. The prefix consists of a set sector and TIC which are located in the IOB extension.

***DL+14 if VLR

†SLI if VLR

CHANNEL PROGRAM 17/17W

CCW No.	Command Code Hex	Command Code Description	Address	Flags Hex	Flags Description	Count	Comments
Changes the key in a normal or overflow track index entry or in a higher level index entry							
CK1*	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for last entry in index
CK2	08	TIC	CK1	00		0	
CK3	06	Read data	CK8	40	CC	10	Read data of last entry
CK30	03 23	NOP Set sector	IOBSECT	80	CC, SLI	1	Search for entry again
CK4	31	Search ID equal	IOBSEEK+3	40	CC	5	
CK5	08	TIC	CK4	00		0	
CK6	0D	Write key and data	Contents of DECBKEY	80	DC	KL	Write new high key and rewrite data of entry
CK7	00		CK8	40	CC	10	
CK70**	03 23	NOP Set sector	IOBSECT	60	CC,SLI	1	Search for updated entry
CK7A**	31	Search ID equal	IOBSEEK+3	40	CC	5	
CK7B**	08	TIC	CK7A	00		0	
CK7C**	0E	Read key and data		10	SK	KL+10	Read it back
CK8	M B B C C H H R						Data of index entry
CK9	F P – – – – – – –						

*Write Validity Check

**This channel program preceded by a prefix if RPS is present. The prefix consists of a set sector and TIC which are located in the IOB extension.

CHANNEL PROGRAM 18

Write Prime Data Blocks—Load Mode, ISAM.							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CL0	23	Set sector	ISLRPSSS	40	CC	1	Position for first record
CL1$_1$	31	Search ID equal	IOBSEEK+3 CQ1, CQ14A	40	CC	5	Search for count field of the block preceding the block to be written next
CL2$_1$	08	TIC	CL1$_1$	00		0	The count field contains the address of the write check segment of this channel program (CL1$_2$)
CL3$_1$	08	TIC	CL4 or CL6	00		0	Transfer to the first CCW of the group of write CCWs to be executed next. The count field contains the address of the last read CCW in the write check segment of this channel program +8.
One copy of CL4 for each buffer. CL4 is used to write blocks for fixed length, unblocked record formats where RKP = 0 because count, key, and data are contiguous.							
CL4∂	1D	Write count, key data	Buffer N	40	CC	8+KL +DL	**Write prime data records when RECFM=F, RKP=0**
One copy of CL6, CL7, CL8 for each buffer. CL6, CL7, CL8 are used to write blocks for fixed length, unblocked formats where RKP≠0, fixed length, blocked formats because count, key, and data are not contiguous.							
CL6∂	1D	Write count	Buffer N	80	DC	8	**Write prime data records when RECFM=F; RKP≠0 or RECFM=FB; RKP–N/A**
CL7	00	Write key	Buffer N+8+RKP	80	DC	KL	
CL8	00	**Write data**	Buffer N+8	40	CC#	DL	
The next CCW follows each copy of CL4 or CL8 except the last. It transfers to the beginning of the Write Validity Check segment of this channel program (CL1$_2$), if this is the last of the current group of write CCWS; otherwise it transfers to the next copy of CL4 or CL6. This CCW is omitted if Write Validity Check is not specified.							
	08	TIC	CL1$_2$, CL4$_n$, or CL6$_n$	00		0	The count field of this CCW contains the address of the next sequential copy of CL4 or CL6
The next CCW (CL5) follows the last copy of CL4 or CL8. It transfers to the beginning of the Write Validity Check segment of this channel program (CL0$_2$), if this is the last of the current group of write CCWs; otherwise it transfers to the first copy of CL4 or CL6. If Write Validity Check is not specified, this CCW points to the first copy of CL4 or CL6.							
CL5	08	TIC	CL1$_2$, CL0$_2$, CL4$_1$, or CL6$_1$	00		0	The count field of this CCW contains the address of CL4$_1$ or CL6$_1$

(continued)

CHANNEL PROGRAM 18 (continued)

Write Prime Data Blocks—Load Mode, ISAM.							
CCW No.	Command Code Hex	Command Code Description	Address	Flags Hex	Flags Description	Count	Comments
CL0_2	23 03	Set Sector NOP	ISLRPSSS	60	CC, SLI	1	Position for first record
CL1_2*	31	Search ID equal	IOBSEEK+3 or Buffer N	40	CC	5	Search for the count field of block preceding the first block of the group last written; Buffer N is the address of the count field if this is a shared track.
CL2_2*	08	TIC	CL1_2	00		0	
The following CCW (CL3_2) transfers to the first read CCW to be executed.							
CL3_2*	08	TIC	CL9	00		0	
One copy of CL9 is generated for each buffer. Each copy of CL9 is command chained except the last. CL3 transfers to the copy of CL9 whose position in relation to the last copy of CL9 is equal to the number of blocks written by this execution of channel program 18.							
CL9*	1E	Read count, key, and data		50	CC, SK#	0	

#Command chain is off if this is the last read or write of a group to be executed.
*Write Validity Check
∂For shared (preformatted) tracks. The count field is not written.

CP19—Preformat shared track and/or write cylinder overflow control record (COCR)
CP91—Fill unused index tracks with inactive and dummy (end of index) entries

CCW No.	Command Code Hex	Command Code Description	Address	Flags Hex	Flags Description	Count	Comments
CM0#†	23	Set sector	CM0+5	40	CC	1	Position for COCR
CM1#	31	Search ID equal	DCBLPDA	40	CC	5	When CP is being generated, DCBLPDA contains the DADAD of the record preceding the first prime data record
CM2#	08	TIC	CM1	00			
CM3#	05	Write data	Area Z	60	CC, SLI	8	Write COCR
CM4#	1B	Seek head	DCBLPDA or CM27+1	40	CC	6	DCBLPDA if COCR and DCBFIRSH are same track, otherwise CM27+1
CM40	23 03	Set sector NOP	ISLRPSSS+1	60	CC, SLI	1	Position to index entries
CM5	31	Search ID equal	DCBLPDA or CM27+3	40	CC	5	DCBLPDA if COCR and DCBFIRSH are same track, otherwise CM27+3
CM6	08	TIC	CM5	00			
CM7	1D	Write count,key, data	Area Z+6	80	DC	8	Write inactive track index entries
CM8	00		Buffer	40	CC	KL+10	
CM9	1D	Write count,key, data	Area Z+14	80	DC	8	
CM10	00		Buffer	40	CC	KL+10	
CM11	1D	Write count, key, data	Area Z+22	80	DC	8	
CM12	00		Buffer	40	CC	KL+10	
CM13	1D	Write count, key, data	Area Z+30	80	DC	8	
CM14	00		Buffer	40	CC	KL+10	
CM15	1D	Write count, key, data	Area Z+38	80	DC	8	
CM16	00		Buffer	40	CC	KL+10	
CM17	1D	Write count, key, data	Area Z+46	80	DC	8	

#Cylinder Overflow Control Record (COCR) to be written. With variable length records, CP19 consists of CM1 through CM4 only because the track index is not preformatted.
†Set sector to zero if RPS.

(continued)

CHANNEL PROGRAM 19/91 (continued)

CP19—Preformat shared track and/or write cylinder overflow control record (COCR)
CP91—Fill unused index tracks with inactive and dummy (end of index) entries

CCW No.	Command Code		Address	Flags		Count	Comments	
	Hex	Description		Hex	Description			
CM21	1D	Write count, key, data	Area Z+62	80	DC	8		
CM22	00		Buffer	40	CC	KL+10		
CM23	1D	Write count, key, data	Area Z+70	80	DC	8		
CM24	00		Buffer	40	CC	KL+10		
CM25	1D	Write count, key, data	Area Z+78	80	DC	8		
CM26	00		Buffer	00		KL+10		
CM27	M B B C C H H R						If the COCR and the Shared Track are not the same track; this field is used to store the Seek and Search arguments for CM4 and CM5.	
CM27	08	TIC	CM5	00		0	This CCW resides in the skeleton only and replaces CM1 when COCR is not to be written.	CP91 only
CM28	0D	Write key and data	Buffer	00		0	This CCW can replace CM8	
CM29	1D	Write count, key, and data	Area Z+6	80	DC	8	This CCW can replace CM7	

CHANNEL PROGRAM 20--Fixed Length Records

CCW No.	Command Code Hex	Command Code Description	Address	Flags Hex	Flags Description	Count	Comments
Writes Track Index Entry(s)							
The following segment of CP20 is executed for fixed length record formats when shared tracks are in effect. CP19 has preformatted the track index by writing a **count field for each entry.**							
CQ0	23	Set sector	ISLRPSSS+2	40	CC	1	Position for normal track index entry
CQ1	31	Search ID equal	ISLIOBA	40	CC	5	Search for normal track index entry to be written next
CQ2	08	TIC	CQ1	00		0	
CQ3	0D	Write key, data	Buffer N+8 +RKP	80	DC	KL	Write normal track index entry
CQ4	00		Area Y+26	40	CC	10	
CQ5	B1	Search ID equal (MT)	Area Y+36	40	CC	5	Search for track to write overflow track index entry
CQ6	08	TIC	CQ5	00		0	
CQ7	0D	Write key, data	Buffer N+8 +RKP	80	DC	KL	Write overflow track index entry
CQ8	00		Area Y+44	40	CC	10	
CQ9	08	TIC	CQ10, CQT1, or CQ13	00		0	**Transfer to write dummy track index entry (CQ10) or to CQT1 if Write Validity Check is specified, or transfer to to CQ13 if CP18 (write prime data) is to be executed next**
CQ10	B1	Search ID equal (MT)	Area Y+54	40	CC	5	Search for dummy track entry to be written next
CQ11	08	TIC	CQ10	00		0	
CQ12	0D	Write key, data	Area Y+62	40	CC	KL+10	Write key, data fields of dummy track index entry
CQ13	1B	Seek HH	ISLIOBA+33	40	CC	6	
CQ14	08	TIC	CQT1 or CL1	20	SLI	5	**Transfer to CQT1 if Write Validity Check is specified, or to CL1 (CP18); this CCW is a NOP during Close processing.**

(continued)

CHANNEL PROGRAM 20–Fixed Length Records (continued)

Writes Track Index Entry(s)							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CQ14A	M B B C C H H R						Seek address for CP18
CQ14B	23	Set sector	ISLRPSSS+2	40	CC	1	Position to next index entry
CQ15	31	Search ID equal	Area Y+18 (R=R-1)	40	CC		Index entry to be written next
CQ16	08	TIC	CQ15	00		0	
CQ17	1D	Write count, key, data	Area Y+18	80	DC	8	Write count, key, and data fields of normal track index entry ISLKEYAD points to key
CQ18	00		Buffer N+8 +RKP	80	DC	KL	
CQ19	00		Area Y+26	40	CC	10	
CQ20	08	TIC	CQ21 or CQ27	00		0	Transfer to CQ21 if normal and overflow entries are on the same track, or to CQ27 if normal and overflow entries are on different tracks
CQ21	1D	Write count, key, data	Area Y+36	80	DC	8	Write overflow index entry ISLKEYAD points to key
CQ22	00		Buffer N+8	80	DC	KL	
CQ23	00		Area Y+44	40	CC	10	
CQ24	08	TIC	CQT1 CQ13 CQ25 CQ27	00		0	Transfer to CQT1 if Write Validity Check is specified, or to CQ13 if CP18 is to be executed next, or to CQ25 if overflow and dummy track index entries are on the same tracks, or to CQ27 if overflow and dummy track index entries are on different tracks
CQ25	1D	Write count, key, data	Area Y+54	40	CC	8+KL+10	Write count, key, and data of dummy of index entry
CQ26	08	TIC	CQT1 or CQ13	00		0	Transfer to CQT1 if Write Validity Check is specified, or to CQ13 if CP18 is to be executed next
CQ27	B1	Search ID equal	CQ30+3	40	CC	5	Index entries are split across tracks. Search for next physical track
CQ28	08	TIC	CQ27	00		0	

(continued)

CHANNEL PROGRAM 20—Fixed-length Records (continued)

Writes Track-index Entry(s)							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CQ29	08	TIC	CQ21 or CQ25	00		0	Transfer to write overflow track index entry (CQ21), or to write dummy track index entry (CQ25)
CQ30	M B B C C H H R						Search argument for next track if index entries are split across track boundary
CQT0*	23	Set sector	ISLRPSSS+2	40	CC	1	Position for track index
CQT1*	31	Search ID equal	Area Y+18	40	CC	5	Find last normal entry written
CQT2*	08	TIC	CQT1	00		0	
CQT3*	0E	Read key and data		50	CC,SK	KL+10	Read entry back
CQT4*	B1	Search ID equal (MT)	Area Y+36	40	CC	5	Find last overflow entry written
CQT4A*	08	TIC	CQT4	00		0	
CQT5*	0E	Read key and data		50	CC,SK	KL+10	Read entry back
CQT5A*	08	TIC	CQT7	60	CC,SLI	1	No inactive entry written
	08	TIC	CQT7	60	CC,SLI	1	Inactive entry written
CQT5B*	B1	Search ID equal (MT)	Area Y+54	40	CC	5	Find inactive entry
CQT5C*	08	TIC	CQT5B	00		0	
CQT6*	0E	Read key and data		50	CC,SK	KL+10	Read entry back
CQT7*	1B	Seek head	CQ14A+1	40	CC	6	FLR – Prime track
CQT8*	08	TIC	CL1	0		0	FLR — Transfer to write prime—CP 18

*Write-validity-check

CHANNEL PROGRAM 20–Variable Length Records

Writes Track Index Entry(s)							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CQ0†	23	Set sector	CQ0+5	40	CC	1	Position for R0
CQ1	31	Search ID equal	CQ5+3	40	CC	5	Search for R0 on current prime track
CQ2	08	TIC	CQ1	00		0	
CQ3	05	Write data	CQ7	40	CC	3	Write track capacity record
CQ4	08	TIC	CL1	00		0	TIC to CP18 to write prime data
CQ5	L L – C C H H R						Maximum record length (LL) and R0 ID for current prime track
CQ6							This CCW not used
CQ7	Y Y R – – – – –						Data of track capacity record (R0)
CQ8							This CCW not used
CQ9	– – Y Y R – – –						Running capacity
CQ10							This CCW not used
CQ11	P P L L – – – –						PP–pointer to last used CCW in CP18, LL–length of current record
CQ12							This CCW not used
CQ13	1B	Seek HH	ISLIOBA	40	CC	6	
CQ14	08	TIC	CQT1 or CL1	20	SLI	5	**Transfer to CQT1 if Write Validity Check is specified, or to CL1 (CP18) if it is not specified; this CCW is a NOP during close processing**
CQ14A	M B B C C H H R						Seek address for CP18
CQ14B	23	Set sector	ISLRPSSS+2	40	CC	1	Position for next entry
CQ15	31	Search ID equal	IOBSEEK+3	40	CC	5	Index entry to be written next
CQ16	08	TIC	CQ15	00		0	
CQ17	1D	Write count, key, and data	Area Y+18	80	DC	8	Write count, key, and data fields of normal track index entry ISLKEYAD points to key
CQ18	00		Buffer N+8 +RKP	80	DC	KL	
CQ19	00		Area Y+26	40	CC	10	

†Set sector to zero if RPS

(continued)

CHANNEL PROGRAM 20—Variable-length Records (continued)

Writes Track Index Entry(s)							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CQ20	08	TIC	CQ21 or CQ27	00		0	Transfer to CQ21 if normal and overflow entries are on the same track, or to CQ27 if normal and overflow entries are on different tracks
CQ21	1D	Write count, key, data	Area Y+36	80	DC	8	Write overflow index entry
CQ22	00		Buffer N+8 +RKP	80	DC	KL	ISLKEYAD points to key
CQ23	00		Area Y+44	40	CC	10	
CQ24	08	TIC	CQT1 or CQ13 or CQ25 or CQ27	00		0	Transfer to CQT1 if Write Validity Check is specified, or to CQ13 if CP18 is to be executed next, or to CQ25 if overflow and dummy track index entries are on the same tracks , or to CQ27 if overflow and dummy track index entries are on different tracks
CQ25	1D	Write count, key, data	Area Y+54	40	CC	8+KL+10	Write count, key, and data of dummy index entry
CQ26	08	TIC	CQT1 or CQ13	00		0	Transfer to CQT1 if Write Validity Check is specified, or to CQ13 if CP18 is to be executed next
CQ27	B1	Search ID equal (MT)	CQ30+3	40	CC	5	Index entries are split across tracks. Search for next physical track
CQ28	08	TIC	CQ27	00		0	
CQ29	08	TIC	CQ21 or CQ25	00		0	Transfer to write overflow track index entry (CQ21), or to write dummy track index entry (CQ25)
CQ30	M B B C C H H R						Search argument for next track, if track entries are split across track boundary
CQT0*	23	Set sector	ISLRPSSS+2	40	CC	1	Position for track index
CQT1*	31	Search ID equal	Area Y+18	40	CC	5	Find last normal entry written
CQT2*	08	TIC	CQT1	00		0	

*Write-validity-check

CHANNEL PROGRAM 20—Variable length Records (continued)

Writes Track Index Entry(s)							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CQT3*	0E	Read key and data	*	50	CC,SK	KL+10	Read entry back
CQT4*	B1	Search ID equal (MT)	Area Y+36	40	CC	5	Find last overflow entry written
CQT4A*	08	TIC	CQT4	00		0	
CQT5*	0E	Read key and data	*	50	CC,SK	KL+10	Read entry back
CQT5A*	08	TIC	CQT7	60	CC,SLI	1	No inactive entry written
	08	TIC	CQT7	60	CC,SLI	1	Inactive entry written
CQT5B*	B1	Search ID equal (MT)	Area Y+54	40	CC	5	Find inactive entry
CQT5C*	08	TIC	CQT5B	00		0	
CQT6*	0E	Read key and data	*	50	CC,SK	KL+10	Read entry back
CQT7*	1B	Seek head	CQ5+2	40	CC	6	VLR-Track capacity record
CQT8*	08	TIC	CQ1	0		0	VLR-Write track capacity record

*Write-validity Check

CHANNEL PROGRAM 20A

Write a non-shared track of track index							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CQ0	23	Set sector	ISLRPSSS+2	40	CC	1	Position for the track index entry
CQ1	31	Search ID equal	IOBASEEK+3	40	CC	5	Search for the Count Field of the record preceding the record to be written next
CQ2	08	TIC	CQ1	00			The count field contains the address of the CCW that TICs to CP18 when non-write check
CQ3	08	TIC	CQ4	00			TIC to the first write CCW to be executed, as follows: 1. CQ4 2. Resume Load write CCW (some CQ4) 3. Non-shared last track of track index. The address of some CQ4 is stored in the count portion of this TIC (may be CQ4)
One copy of CQ4 for each track index entry							
CQ4	1D	Write count, key, and data	TISA+20 or TISA+20+N (8+KL+10)	40	CC	8+KL+10	Write a track index entry
For non-write checking, the following two CCW's are at the end of CP20A							
	1B	Seek head	TISA+1	40	CC	6	Seek on the prime data track to be written
	08	TIC	CP18	00		0	TIC to CP18
For write checking, the following CCW is at the end of CP20A							
	08	TIC	CP20C	00		0	TIC to CP20C

CHANNEL PROGRAM 20B

Write a shared track of track index							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CQ0	23	Set sector	ISLRPSSS+2	40	CC	1	Position for the next index entry
CQ1	31	Search ID equal	IOBASEEK+3	40	CC	5	Search for the count field of the record to be written next
CQ2	08	TIC	CQ1	00			The count field contains the address of the CCW that TICs to CP18 for non-write check
CQ3	08	TIC	CQ4	00			TIC to the first write key, data CCW to be executed, as follows: 1. CQ4 2. Resume Load write KD CCW (some CQ7)
CQ4	0D	Write key, data	TISA+20+8 or TISA+20+8+N (8+KL+10)	40	CC	KL+10	Write the first track index entry on a shared track
One copy of CQ5, CQ6, and CQ7 for each remaining track index entry							
CQ5	31	Search ID equal	TISA+20+N (8+KL+10)	40	CC	5	Search for the count field of the record to be written next
CQ6	08	TIC	CQ5	00		0	TIC to CQ5
CQ7	0D	Write key, data	TISA+20+8+ N (8+KL+10)	40	CC	KL+10	Write the key and data portion of a track index entry
For non-write checking, the following two CCW's are at the end of CP20B							
	1B	Seek head	TISA+1	40	CC	6	Seek on the prime data track to be written
	08	TIC	CP18	00		0	TIC to CP18
For write checking, the following CCW is at the end of CP20B							
	08	TIC	CP20C	00		0	TIC to CP20C

CHANNEL PROGRAM 20C

Write check for CP20A and B							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CQ0	03 23	NOP Set sector	ISLRPSSS+2	60	CC,SLI	1	Position for the next index entry
CQ1	31	Search ID equal	IOBASEEK+3	40	CC	5	Search for the count field of the record to be written next
CQ2	08	TIC	CQ1	00		CQ9	The count field contains the address of the CCW that TICs to CP18
CQ3	08	TIC	CQ4	00			TIC to the first read CCW to be executed as follows: 1. CQ4 2. Resume Load read CCW (some CQ7) 3. Read CCW for non-shared last track or shared track. The address of this CCW is stored in the count portion of this TIC (may be CQ4).
CQ4	0E	Read key, data	TISA+20+8 or TISA+20+ 8+N (8+KL+10)	50	CC,SK	KL+10	Read back a track index entry
One copy of CQ5, CQ6, and CQ7 for each remaining track index entry.							
CQ5	31	Search ID equal	TISA+20+N (8+KL+10)	40	CC	5	Search for the count field of the record to be written next
CQ6	08	TIC	CQ5	00		0	TIC to CQ5
CQ7	0E	Read key, data	TISA+20+8+ N (8+KL+10)	50	CC,SK	KL+10	Read back a track index entry
CQ8	1B	Seek head	TISA+1	40	CC	6	Seek on the prime data track to be written
CQ9	08	TIC	CP18	00		0	TIC to CP18

CHANNEL PROGRAM 21

Write High Level Index and End of Data (EOD) Mark(s)'							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CQ39A	23	Set sector	ISLRPSSS+3	40	CC	1	Position for entry
CQ40	31	Search ID equal	Area Y	40	CC	5	Search for ID of index entry to be written with R=R-1
CQ41	08	TIC	CQ40	00			
CQ42	1D	Write count, key, data	Area Y	80	DC#	8	Write count field of current under entry
CQ43	00		ISLKEYA or Area Y+62	80	DC	KL	ISLKEYAD is used for normal entry area Y+62 is used for dummy and inactive entry
CQ44	00		Area Y+8	00 40	CC (Write validity check)	10	Write data field of high level index entry
CQ44A*	03 23	NOP Set sector	ISLRPSSS+3	60	CC, SLI	1	Position for entry
CQ45*	31	Search ID equal	Area Y	40	CC	5	Search for ID (CCHHR) of current index entry with R=R-1
CQ46*	08	TIC	CQ45	00		0	
CQ47*	1E	Read count, key, data		10	SK	KL+18	Read back current high level index entry

#Close processing utilizes CP21 to write end of data marks in the prime data area and independent overflow area. ISL area Y is initialized with the 'KDD' portion of the count field set to zero. The data chain bit is turned off.
*Write Validity Check

CHANNEL PROGRAM 22A

Read/Write data record — key and data, unblocked records							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CN1*	B1	Search ID equal (MT)	CN6+3	40	CC	5	MT set off for 1st CP 22 in chain
CN2	08	TIC	CN1	XX	CN2+4 used as buffer flags	0	See description of CN2+4 and CN2+5 below
CN3	08	TIC	CN4	00		0	Transfer is set when records are blocked or when data only (instead of key and data) is read or written
	0E 0D	Read key and data Write key and data	Buffer address and offset	80	DC	KL	SKIP bits set on in CN3 and CN4 for write check processing
CN4	06 05	Read data Write data	Buffer address and offset	40	CC (off when end of chain) unless CN5 is used for RPS)	DL	Fixed-length records: the blocksize (DL) is constant so the count field is set at open Variable-length records: the actual block size is set in the count field by the EOB routine each time this CP is executed
CN5	08	TIC	Next CN1	00		0	Transfer to next CP 22 in chain if record is not last or not RPS
	88	TIC	WIREADSC**	00		0	If RPS, and record is not last on track and last in chain, transfer to RDCNT and RDSECTOR for read only.
	22	Read sector	CN2+6	00		1	Save sector of record read for PUTX.
CN6	M B B C C H H R						Set from W1LPDR or link field in overflow record
CN7	Address buffer and offset						Set from DCBBUFCB init.

*If RPS is present and this channel program is not chained from CP 24, it will be preceded by a set sector and a TIC. The set sector and TIC are located in the work area. If the channel program is chained from CP 24, the set sector will be performed in CP 24.

**W1OSECT of channel program is writing.

The following is a description of buffer flags at CN2+4 and CN2+5.

CN2+4

BIT		
0	1	Buffer marked for PUTX
1	. 1	Overflow record
2	. . 1	Key and data to be read
	. . 0	Data only to be read
3	. . . 1	End of data buffer
4	. . . . 1 . . .	Input error
5	. . . . . 1 . .	Unwritable block
6	. . . . . . 1 .	Unreachable block
7	. . . .	Reserved

CN2+5

BIT		
0	1	End of track

CHANNEL PROGRAM 22B

Read/Write data records—data only, unblocked records; all blocked records							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CN1*	B1	Search ID equal (MT)	CN6+3	40	CC	5	MT is set for first CP 22 in chain
CN2	08	TIC	CN1	XX	CN2+4 used as flags for buffer de-scription	0	See description of CN2+4 and CN2+5 below, CP 22A
CN3	08	TIC	CN4	80	DC (ignored)	KL	
CN4	06 05	Read data Write data	Buffer address and offset	40	CC (off when last in chain unless CN5 is used for RPS)	DL	Fixed length records: the block size (DL) is constant so the count field is set at open time. Variable length records: the actual block size is set in the count field by the EOB routine each time this CP is executed.
CN5	08	TIC	Next CN1	00		0	Transfer to 1st CCW in next CP22 in chain if not lost in chain or if not RPS
	88	TIC	WIREADSEC**	00		0	If RPS, and record is not last on track and last in chain, transfer to RDCNT and RDSECTOR for read only.
	22	Read sector	CN2+6	00		1	Save sector of record read for PUTX
CN6	M B B C C H H R						Set from WILPDR or link field in overflow record
CN7	Address buffer and offset						Set from DCBBUFCB

*See note to CP22A.

**W10SECT if channel program is writing.

CHANNEL PROGRAM 23

Search hi-level indexes, track index, and data track for SETL K or KC							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CS1	31	Search ID equal	W1IMBBCC+3	40	CC	4	Position read head to first index track
CS1A	08	TIC	CS1	00		0	
CS1B	69	Search key high or equal	Key address	60	CC, SLI	KL	Too far along index
CS1C	08	TIC	CS2	00		0	No
CS1D	03 23	NOP Set sector	CS1D+5	60	CC, SLI	1	Set sector to zero if RPS Yes, position to index point.
CS1E	1A	Read home address		50	CC, SK	5	Position to home address
CS2	E9	Search key high or equal (MT)	Key address	40	CC	KL	Key address passed in register 0
CS3	08	TIC	CS2	00		0	
CS4	06	Read data	CS6+7	40	CC (off for master indexes)	10	CC set on when read cylinder index; read data of current index entry
CS5	08	TIC	CS8	00		0	
CS6 CS7	– – – – – – – M B B C C H H R F						Address of next lower level index
CS8	P	Seek	CS7	40	CC	6	Seek track index. See Figure 82 for value of P (seek command code).
CS80	03 23	NOP Set sector	CS80+5	60	CC, SLI	1	
CS9	31	Search ID equal	CS7+2	40	CC	5	Starting CCW when only track index; position read head to R0 to track index
CS9A	08	TIC	CS9	00		0	
CS10	92	Read count (MT)	W1WCOUNT	40	CC	8	Read count of current index entry (normal or overflow)
CS11	69	Search key high or equal	Key address	40	CC	KL	Key address passed in register 0
CS12	08	TIC	CS10	00		0	
CS13	06	Read data	CS17+7	40	CC	10	Read data of current index entry (normal or overflow)
CS14	92	Read count (MT)	W1WCNXDM	40	CC	8	Read count of next index entry (normal or overflow)

(continued)

CHANNEL PROGRAM 23 (continued)

Search hi-level indexes, track index, and data track for SETL K or KC							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CS15	06	Read data	W1WDNXDM	60	CC, SLI	10	Read data of next index entry (normal or overflow)
CS16	08	TIC	CS19	00		0	
CS17 CS18	— — — — — — — M B B C C H H R F						**Track index entry contains address of prime data or overflow track containing record**
C19	P	Seek	CS18	40	CC	6	**Seek data track. (Figure 81)**
CS19A	03 23	NOP Set sector	CS19A+5	60	CC, SLI	1	Set sector to zero if RPS Position to start of track if RPS
CS20	31	Search ID equal	CS18+2	40	CC	5	Search to the first data record on track
CS21	08	TIC	CS20	00		0	
CS26	08	TIC	CS22	00		0	
CS25	12	Read count	First CN6+3	60	CC, SLI	5	Read count (CCHHR) of record into first CP22; R set to 0
CS22	29 69	Search key equal Search key high or equal	Key address	60	CC, SLI (on for KC)	KL	Search for desired record (29) or search for desired block (69)
CS23	08	TIC	CS25	00		0	
CS24	03 22	NOP Read sector	00 W1ISECT	20	SLI	1	Exit when record found

CHANNEL PROGRAM 24

Read track index entries							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CN8*	31	Search ID equal	W1WCOUNT	40	CC	5	W1WCOUNT – count of current index entry; set from W1WCNXDM
CN9	08	TIC	CN8	00		0	
CN10	06	Read data	W1DCXDM	40	CC	10	Read data of current normal index entry
CN11	86	Read data (MT)	W1WOVFL	40	CC	10	Read data of current overflow index entry
CN12	92	Read count (MT)	W1WCNXDM	40	CC	8	Read count of next normal or dummy entry
CN13	06	Read data	W1WDNXDM	40	CC	10	Read data of next normal or dummy entry
CN14	1B	Seek HH	CN6+1	40	CC	6	Seek to track in W1LPDR
CN14A	03 23	NOP Set sector	CN14A+5	60	CC, SLI	1	Set sector to zero Position to first record next track
CN15	08	TIC	CN1	00		0	Transfer to read or write the record

*If RPS is present this channel program will be preceded by a set sector - TIC located in the work area.

CHANNEL PROGRAM 25

Read track index entries for SETL I							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CN20*	31	Search ID equal	W1IDAD	40	CC	5	Search to record at actual direct-access address
CN21	08	TIC	CN20	00		0	
CN22	0E	Read key and data	CN7+5	60	CC, SLI	KL	Read record key into 1st buffer
CN23	1B	Seek head	CN31+1	40	CC	6	Seek to beginning of track index
CN23A	03 23	NOP Set sector	CN23A+5	60	CC, SLI	1	Set sector to zero Position to first record of next track
CN24	1A	Read home address	CN31	50	CC, SK	5	Position read head to start of track
CN25	E9	Search key high or equal (MT)	CN7+5	40	CC	KL	Serially search index tracks for index entry containing key
CN26	08	TIC	CN25	00		0	
CN27	06	Read data	W1WDCXDM	40	CC	10	Read data of current normal index entry
CN28	86	Read data (MT)	W1WOVFL	40	CC	10	Read data of current overflow index entry
CN29	92	Read count (MT)	W1WCNXDM	40	CC	8	Read count of next normal or dummy entry
CN30	06	Read data	W1WDNXDM	00		10	Read data of next normal or dummy entry
CN31	M B B C C H H R						Address of track index; set from lower entry with HH=0, R=1

*If RPS is present this channel program will be preceded by a set sector-TIC located in the work area.

CHANNEL PROGRAM 26

Extension of CP23 to read overflow chains							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CS27*	31	Search ID equal	W1IMBBCC+3	40	CC	5	Search to first record of overflow chain
CS28	08	TIC	CS27	00		0	
CS29**	69	Search key high or equal	Key address	40	CC	KL	SLI on when KC, search for desired record in chain
CS30	08	TIC	CS32	00		0	
CS31	03	NOP		20	SLI	1	Exit when record found if RKP = 0, unblocked
	08	TIC	CN4 of buffer	20		1	Read in record if RKP=0 or blocked format
CS32	06	Read data	CS34+7	60	CC, SLI	10	Read link field of overflow record
CS33	08	TIC	CS36	00		0	
CS34	– – – – – – – – M						Address of overflow record
CS35	B B C C H H R F						
CS36	P	Seek	CS35	40	CC	6	**Seek overflow track containing next record in chain. See Figure 81 for value of P (seek command code).**
CS37	31	Search ID equal	CS35+2	40	CC	5	Search for overflow record
CS38	08	TIC	CS37	00		0	
CS39	08	TIC	CS29	00		0	

*If RPS is present this channel program will be preceded by a set sector—TIC located in the work area.
**Search key equal if RKP=0, RECFM=F and not SETL KH or SETL KDH.

CHANNEL PROGRAM 31A

Reads the key of the last overflow track index entry into the Key save area							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CA1	31	Search ID equal	IOBASEEK+3	40	CC	5	Search for the last normal track index entry
CA2	08	TIC	CA1	00			
CA3	9E	Read count, key, data		90	DC, SK	8	Read last overflow track index entry
CA4	00		Key save area	80	DC	KL	Read key of last overflow track index entry into key save area
CA5	00			10 50	SK CC, SK is turned on if CP31B is executed	10	

CHANNEL PROGRAM 31B

Reads the count and data of the last prime data block into the first buffer specified in the Buffer Control Table							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CA1	1B	Seek head	CA6+1	40	CC	6	Seek to the head of the last prime data block
CA2	31	Search ID equal	CA6+3	40	CC	5	Search for the next to last prime data record
CA3	08	TIC	CA2				
CA4	12	Read count	First buffer	40	CC	8	Read count of the last prime data block into the first buffer (buffer control table + 9)
CA5	06	Read data	First buffer +8	00		DL	Read data of the last prime data block into the first buffer + 8
CA6	M B B C C H H R						MBBCCHHR of DCBLPDA, R is set to R-1

CHANNEL PROGRAM 87

Reads high-level index into user work area (specified by DCBMSHI)—this channel program is in module IGG0192P							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CZ1	31	Search ID equal	IOBSEEK+3	40	CC	5	Search for first entry of high level index
CZ2	08	TIC	CZ1	00		0	
CZ3	8E	Read key and data (MT)	DCBMSHI	40	CC	0	Read it into the work area. There are several copies of CZ3. The channel program is executed as many times as needed to read in the entire index.

CHANNEL PROGRAM 123W

Addendum to CP 12A and CP 12B or to CP 13A and CP 13B when write-checking is specified							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CEA00	03 23	NOP Set sector	IOBSECT	60	CC, SLI	1	Search for record or block again
CEA	31	Search ID equal		40	CC	5	
CEB	08	TIC	CEA	00		0	
CEE	1E	Read count, key, and data		10	SK	0	Read it back

CHANNEL PROGRAM 123WV

Addendum to CP 12AV and CP 12BV when write-checking is specified							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
CEA00	03 23	NOP Set sector	CEA00+5	40	CC	1	Set sector to 0
CEA0	31	Search ID equal	CD0	40	CC	5	Search for track capacity record (R0)
CEA05	08	TIC	CEA0	00		0	
CEA1	06	Read data		70	CC, SK, SLI	3	Read capacity record
CEA2	08	TIC	CED or CEA3	00		0	Transfer to CED if the full track is being checked
CEA3	03 23	NOP Set sector	IOBSECT+1	40	CC, SLI	1	Search for first data record written
CEA	31	Search ID equal	IOBSEEK+3	40	CC	5	
CEB	08	TIC	CEA	00		0	
CED	1E	Read count, key,and data		90	DC, SK	8	Read record back. The number of CEE-CEF sets equals DCBHIRPD, the CC flag is set off in the appropriate CCW depending on how many records are read.
CEE	0E	Read key and data		50	CC, SK	KL+DL	
CEF	1E	Read count, key, and data		90	DC, SK	8+KL+DL	

CHANNEL PROGRAM CLOSECCW(1)

Reads format-2 DSCB—this channel program is in module IGG0202D							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
DXCCW1	31	Search ID equal	Format-2 DSCB address	60	CC, SLI	5	Search for format-2 DSCB
DXCCW2	08	TIC	DXCCW1	00		0	
DXCCW3	0E	Read key and data	DXDADDR	00		140	Read format-2 DSCB into work area

CHANNEL PROGRAM CLOSECCW(2)

Writes format-2 DSCB back in the VTOC- this channel program is in module IGG0202D							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
DX CCW1	31	Search ID equal	Format-2 DSCB address	60	CC, SLI	5	Search for format-2 DSCB position
DX CCW2	08	TIC	DXCCW1	00		0	
DX CCW3	0D	Write key and data	DXDADDR	40	CC	140	Write format-2 DSCB back in VTOC
DX CCW4*	31	Search ID equal	Format-2 DSCB address	60	CC, SLI	5	Search to format-2 DSCB again
DX CCW5*	08	TIC	CCW4	00		0	
DX CCW6*	0E	Read key and data		10	SK	140	Read back

*Write-validity-check

CHANNEL PROGRAM VXCCW (1A)

Reads to EOF or end of LPDA track for prime data—this channel program is in module IGG01920							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
VX CCW1	31	Search ID equal	DS2LPRAD+3	40	CC	5	Search to the last prime data record
VX CCW2	08	TIC	VXCCW1	00		0	
VX CCW2A	08	TIC	VXCCW3A	00		0	Skip first read count
VX CCW3	92	Read count, (MT)	VXCCW6	60	CC, SLI	8	Read count field (normally, count of EOF)
VX CCW3A	06	Read data	WA*	60	CC,SLI	DL	Read in block
VX CCW4	92	Read count, (MT)	VXCCW7	60	CC, SLI	8	Executed when DS2LPRAD is incorrect
VX CCW4A	06	Read data	WA*	60	CC,SLI	DL	Read in block
VX CCW5	08	TIC	VXCCW3	00		0	
VX CCW6	C C H H R KL DL DL						Count field
VX CCW7	C C H H R KL DL DL						Count field

*The work area is obtained by a GETMAIN.

CHANNEL PROGRAM VXCCW(1B)

Reads to EOF for independent overflow or end of LPDA track for prime data — this channel program is in modules IGG01922 and IGG01950							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
VX CCW1	31	Search ID equal	DS2LOVAD+3	40	CC	5	Search to the last overflow record
VX CCW2	08	TIC	VXCCW1	00		0	
VX CCW3	9E	Read count, key, and data (MT)	VXCCW6	60	CC, SLI	8	Read count field (should be count of EOF)
VX CCW4	9E	Read count, key, and data (MT)	VXCCW7	60	CC, SLI	8	Executed when DS2LOVAD is incorrect
VX CCW5	08	TIC	VXCCW3	00		0	
VX CCW6	C C H H R KL DL DL						Count field
VX CCW7	C C H H R KL DL DL						Count field

CHANNEL PROGRAM VXCCW(2)

Reads to end of track - this channel program is in module IGG01920							
CCW No.	Command Code		Address	Flags		Count	Comments
	Hex	Description		Hex	Description		
VX CCW4	12	Read count	SAVEREG	60	CC, SLI	8	Read count of each record on track
VX CCW5	08	TIC	VXCCW4	00		0	CP will end with count of last record on track in SAVEREG

INDEX

Indexes to program logic manuals are consolidated in the publication *IBM System/360 Operating System: Program Logic Manual Master Index,* GY28-6717. For additional information about any subject listed below, refer to other publications listed for the same subject in the *Master Index.*

A

abnormal end appendages
  (*see* Appendages)
adding records to data set
  basic description 213-214
allocating space on ISAM data set 152
appendage codes 201-202
appendage definition 2
appendages
  BISAM
    codes 201-202
    diagram 66
    modules 72
    pointers to 90
    processing 65,69
    vector table 65,90
  QISAM (load mode)
    abnormal end 29
    channel-end 28
    pointers to 34
    processing 23,30
    vector table 34
    write checking functions 29
  QISAM (scan mode)
    abnormal end 49
    channel-end 49
    codes 201
    GET 49,44
    modules 52
    pointers to 54
    processing 49,50
    PUTX 50,47
    SETL 49,42
    vector table 54
    write-checking function 50
Area Y 185,181,34
Area Z 180,179,34
asynchronous codes
asynchronous routines -- BISAM
  codes 202
  flow diagram 66,69
  modules 72
  pointers to 90
  vector table 65

B

BCB
  (*see* buffer control block)
BCT
  (*see* buffer control table)
beginning-of-buffer (BOB) routine
  flow diagrams 26,30,31,32
  processing 27
BISAM
  channel programs
    (*see* Channel programs, BISAM)
  code phase 91
  control blocks and work areas 89-91
  DCB work area 193-195
  flowcharts
    processing routines 134-138
    channel program flow 76-88
  open phase 55
  processing flow 69
  processing phase 61
buffer control block
  BISAM
    format 172-174
    pointers to 91
    use by dynamic buffering routine 67
    use by open routines 172
  QISAM 174
buffer control table (load mode)
  format 175-179
  pointers to 34
  use by open routines 175
Buffers
  BISAM
    control block 172-174
    dynamic buffering 67-68
    pointers to 90
    queues 91
  QISAM(scan mode)
    control block 174
    control technique 39
    initialization 37
    pointers to 40,54
    queues and processing 39-42
    scheduling 45

Buffers (continued)
QISAM (load mode)
closing functions 33
control block 174
control table 175-179
pointers to 34
processing 15,34,35
scheduling 23-27

C

C-bit 178
CCWs, explanation of 222
Chaining
channel program 22
scan mode 38,45
Chains
(*see* overflow chains)
Channel program descriptions and formats 226-290
CLOSECCW(1) 286
CLOSECCW(2) 287
VXCCW(1A) 288
VXCCW(1B) 289
VXCCW(2) 290
1 226-227
2 226
4 229
5/5W 230
6/6W 231-232
7/7W 233
8 234-235
9A 236
9B/9BW 236
9C/9CW 237
10A/10AW 238
10B/10BW 239
11A 240
11B/11BW 241
12A 242
12B 243
12C/12CW 244
12AV 245
12BV 246
13A 247
13B 248
13C/13CW 249
14/14W (fixed length records) 250-252
14/14W (variable length records) 253-256
15 257
16 258
17/17W 259
18 260-261
19/91 260-261
20 (fixed length records) 262-264
20 (variable length records) 265-267
20A 268
20B 269
20C 270
21 271
22A 272
22B 273
23 274-275
24 276
25 277
26 278
31A 279
31B 279
87 280
123W 281
123WV 281
Channel programs
BISAM
flow-of-control (non write KN) 76
flow-of-control (write KN) 77-78
functions 71-76,60
modules 73
list of 143
QISAM (load mode)
flow-of-control 31-32
functions 32-33
modules 30
QISAM (scan mode)
functions 51-52
modules 51
queues 55
Check routine – BISAM
description 68
flow diagram 68
Close phase executors and modules
common 13-14
BISAM 91
errors during 205-205,53
flow-of-control 14
QISAM
load mode 33-35
scan mode 52,204
COCR
(*see* cylinder overflow control record)
codes
appendage 201-202
asynchronous 202-203
exception (error) 204,205
common close 13-14,9
channel programs used 286-287
flow diagram 14
module 13
common open 9-12
channel programs used 288-289
modules 9
count field 216
CP
(*see* channel programs)
cylinder index
BISAM processing 77,71
definition 212
direct access extents 157,158,182
format 216
load mode processing 31-32
Cylinder overflow area 212
cylinder overflow control record (COCR)
definition 212

cylinder overflow control record (COCR) (continued)
BISAM processing 77-88
format 218

D

data control block (DCB)
BISAM processing use 89-91
format 149-158
initialization
BISAM 56
common 9
QISAM
load mode 16
scan mode 37
integrity feature 9
QISAM – load mode processing use 34
QISAM – scan mode processing use 54
data extent block (DEB)
BISAM processing use 89-91
format 166-169
initialization 11
QISAM
load mode processing use 34
scan mode processing use 54
data event control block (DECB)
BISAM processing use 89,69
format 159-160
data set control block (DSCB)
format 161-165
use by open routines 9,12
use by close routines 13
data set organization 209
adding records of data set 213
indexes 211
detail description 215
overflow area 212
prime data area 211
DCB
(*see* data control block)
DCB work area
BISAM
format 193-195
initialization 57
pointers to 90-91
QISAM
load mode
format 180-185
pointers to 34
QISAM
scan mode
format 187-192
pointers to 54
DCW
(*see* DCB work area – BISAM)
DEB
(*see* data extent block)
DECB
(*see* data event control block)
deletion, record
BISAM asynchronous code 202
count fields tagged for deletion 156,164
processing 78-88
disable SVC 63-64
DSCB
(*see* data set control block)
DS2
(*see* data set control block)
dummy index entries
creation 35
format 212,216-217
duplicate records
error indications 204-205
processing 78-88
dynamic buffering routine – BISAM
description 67,60
control block 170-172
flow diagram 67
initialization 58
pointers to 90

E

ECB
(*see* event control block)
enable, BISAM I/O interruptions 61
end-of-buffer (EOB) routine
load mode
description 27,30
fields used 174-179
flow diagram 27
scan mode
description 45,46
flowchart 126
end-of-cylinder processing
fields used
flowcharts 31,32
end-of-extent processing
fields used 174-179
flowcharts 31,32
end-of-file (EOF) mark processing 78-88
end-of-track processing
fields used 174-179
flowcharts 31,32
end index entries, format 212
cylinder 220
master 221
track 219
EOB
(*see* end-of-buffer routine)
EOF
(*see* end-of-file mark)
error codes
BISAM 205
QISAM 204

error descriptions
duplicate record 78-88
record length – BISAM 65
sequence error 24
write K with read KU 63
error queue – BISAM
format 91,195
flowchart references 137
use in processing 65,69
ESETL macro instruction 39
ESETL routine – scan mode
description 48
flowchart 121
event control block
BISAM 159
QISAM
load mode 170,34
scan mode 170,54
exception codes
BISAM 205
QISAM 204
EXCP
BISAM 69,201
QISAM
load mode 28,29
scan mode 49
executors
(*see* open executors and close executors)
extents 157,163

F

flowcharts
BISAM macro time routines 136-138
BISAM open executor 134-136
common close executor 114-115
common open executors 95-100
load mode open executors 101-108
scan mode appendage routines 130-132
scan mode close executors 133
scan mode open executors 109-113
scan mode processing routines 116-129
format, data set
(*see* data set organization)
free queue – scan mode
format 40
flow diagram references 43-48
use in processing 43-48
FREEDBUF macro instruction 63,173
(*see* also dynamic buffering routine)
full track index
full track index write 16
track index save area 196

G

GET appendage routine – scan mode
description 49
module 51
pointers to 54
GET macro instruction 39,204
GET routine – scan mode
description 43-44
flowchart 116
module 51
pointers to 54,149

I

inactive index entries 219-221
index
(*see* cylinder, master, or track)
index location table – load mode
format 180,182
initialization
pointers to 34
input/output block (IOB)
BISAM
pointers to 90-91
processing use 61-62,106,172-174
queues 91,193
format 170-172
channel program use 226
codes 201-203
QISAM
load mode 34,179
scan mode 54,189
IOB
(*see* buffer control table)
integrity feature, DCB
(*see* data control block integrity feature)
ISAM data set
(*see* data set organization)
ISL
(*see* DCB work area – load mode)

K

keysave area – load mode 33,34

L

levels of indexes
description 211-212
format 216-221
library, SVC 1
load mode 14
channel programs 31
descriptions flow of control 31-32
close phase 33-35
control block and work areas 34

load mode (continued)
DCB work area 180-185,34
flow diagrams 25-29
open phase 15-23
processing phase 23-33
locate mode processing 26

M

M=0 DEB extent 34,167
macro instructions
(*see* GET, PUT, etc.)
Macro-time routines
(*see* privileged and nonprivileged)
master indexes
format 221
BISAM processing 76,77
direct access extents 149,163,182
QISAM load mode processing 31-32
MBBCCHHRFP 216
modules directory 143-145
move mode processing 24

N

N/2 buffers 45
new high key records
BISAM 78,83-85
QISAM load mode 24
nonprivileged macro-time routine – BISAM
description 64
flow diagram 64,69
modules 70
pointers to 90
normal track index entry
description 212
format 216-219

O

organization, data set
(*see* data set organization)
open phase executors and modules
BISAM 59-60
common 9-12
QISAM
load mode 15-16
scan mode 37-38
overflow records and chains
BISAM processing 76-77
description 212
format 218
QISAM – scan mode processing 38,53,54
overflow track index entry
description 212
format 216-219

P

padding records 35
PF-bit 178
phase
(*see* open, close, or processing)
pointer diagrams
BISAM 89-92
QISAM
load mode 34
scan mode 54
prime data area
adding records to 213
pointers to 31
prime data track, shared
(*see* shared track)
privileged macro-time routine – BISAM
description 62-64
flow diagrams 62,69
modules 70
pointers to 90
processing phase
BISAM 60
QISAM
load mode 23
scan mode 38
PUT appendage
(*see* appendage routines – load mode)
PUT macro instruction 23
exception codes set 204
PUT routine – load mode
description 24-25
flow diagrams 25
pointers to 34
PUTX appendage
(*see* appendage routines scan mode)
PUTX macro instruction 39
exception codes set 204
PUTX queue – scan mode
format 40,54,191
flow diagram references 46,48
use in processing 39-42
PUTX routine – scan mode
description 47
flowchart 118
pointers to 54

Q

QISAM modes
(*see* load mode and scan mode)
queues
BISAM load mode 34
QISAM
scan mode 40-42,54

R

reopen data set
(*see* resume loading)

Read appendages
(*see* appendage routines – BISAM)
READ macro instruction 56
exception codes set 205
Read queue – scan mode
format 40-42,54
flow diagram references 43-48
use in processing 38-48
RELSE macro instruction 39
RELSE routine
description 49
flowchart 118
pointers to 54
resume loading 20
channel programs 31-33
initialization 21-22
rotational position sensing
devices 3
identification in DEB 11
start I/O appendages 2-3

S

scan mode
channel programs 51,52
close phase 52
control blocks and work areas 52,54
DCB work area 187-192,54
flowcharts 116-133
open phase 37
processing phase 38
queues 40-42,54
schedule routine – scan mode
description 45
flowchart 122
pointers to 54
scheduling of BISAM
channel programs 61-63
SELT macro instruction 39
exception codes set 204
SETL routine – scan mode
description 42
flowchart 119
pointers to 54
shared track
channel programs used 51,33
fields used
BCB 178
DCB 156
DCB work area (load) 180
DSCB 164
initialization 23
index format 216-217
processing 33
stages of open and close executors 1-4
status indicators
buffers – load mode 173
DCB 157
DSCB 164
scan mode 189
SYNAD macro instruction
(*see* synchronous error routine)
SYNADAF macro instruction 70
synchronous error routine
address 152
BISAM use 68-70
QISAM
load mode use 24,27
scan mode use 42-49,52

T

T-bit 178
TISA
(*see* track index save area)
track index
BISAM processing 76-89
description 219
format 216-217
QISAM
load mode processing 31-32
track index save area (TISA) 196-197
track, shared
(*see* shared track)

U

unit control block (UCB), pointers to 34,54
unreachable block error 205
unscheduled queue – BISAM
format 91
pointers to 91,194
use in processing 62,64,66,171
update processing – BISAM 76,60
update queue – BISAM
format 91
pointers to 91,194
use in processing 60
User queue – scan mode
format 40,54,190-191
flowchart references 44,46,47
use in processing 39-42

W

WAIT macro instruction – BISAM 55
Write appendages
(*see* appendage routine – BISAM)
WRITE macro instructions 55
exception codes set 205
WRITE K processing 58,76
channel programs 73
flow of control 77,88
differing methods of adding records to a data set 71-72
Write queue – scan mode
format 40,54,191
flowchart references 44,46,47,48
use in processing 39-42

READER'S COMMENT FORM

OS ISAM Logic

Order Number GY28-6618-5

Your comments about this publication will help us to produce better publications for your use. If you wish to comment, please use the space provided below, giving specific page and paragraph references.

Please do not use this form to ask technical questions about the system or equipment or to make requests for copies of publications. Instead, make such inquiries or requests to your IBM representative or to the IBM Branch Office serving your locality.

Reply requested

Yes ☐

No ☐

Name ____________________

Job Title ____________________

Address ____________________

____________________ Zip ____________________

No postage necessary if mailed in the U S A

Order Number GY28-6618-5

YOUR COMMENTS, PLEASE . . .

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

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

fold fold

FIRST CLASS
PERMIT NO. 2078
SAN JOSE, CALIF.

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

POSTAGE WILL BE PAID BY . . .

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

Attention: Programming Publications, Dept. D78

fold fold

IBM®

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

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

Order Number GY28-6618-5

IBM

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

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