GY30-2029-0

Systems

IBM System/360 Operating System Telecommunications Access Method (TCAM) Program Logic Manual

Program Number 360S - CQ - 548

The IBM System/360 Telecommunication Access Method (TCAM) allows high-level, device-independent communication with telecommunications equipment. This program provides a flexible message control language that can be used to achieve installation-oriented message control.

This publication describes the internal logic of TCAM. It identifies and discusses the parts of the program that perform specific functions and relates these parts to the program listing. It is directed to the IBM customer engineers and system engineers, who need information on the internal organization and logic of TCAM in order to provide program maintenance.

In order to understand the logic of TCAM, the reader must have a general understanding of IBM System/360 operating system. In addition, the following are prerequisite publications:

- *IBM System/360 OS TCAM Concepts and Facilities,* GC30-2022, to gain familiarity with the overall concepts and structure of TCAM.
- *IBM System/360 OS TCAM Programmer's Guide,* GC30-2024, to learn how to construct and modify a TCAM message control program and a TCAM-compatible application program.

In addition, the *IBM System/360 OS System Control Blocks* publication, GC28-6628, provides co-requisite information on system control blocks used by TCAM.

- The information relating to the Time Sharing Option (TSO) in this manual is preliminary and should be used accordingly.

IBM

PREFACE

The *Organization and Use of the TCAM Program Logic Manual* section of this book defines the audience for which this program logic manual was intended, explains how the book is organized, and suggests how the reader might best familiarize himself with its contents. In order to understand the logic of TCAM, the reader must have a general understanding of System/360 OS. In addition, the following prerequisite publications are applicable:

- *IBM System/360 OS TCAM Concepts and Facilities*, Order No. GC30-2022, to gain familiarity with the overall concepts and structure of TCAM.
- *IBM System/360 OS TCAM Programmer's Guide*, Order No. GC30-2024, to learn how to construct and modify a TCAM MCP and a TCAM-compatible application program.

The *IBM System/360 OS System Control Blocks* publication, Order No. GC28-6628, provides corequisite information on system control blocks that are used by TCAM.

The information relating to the Time Sharing Option (TSO) in this manual is preliminary and should be used accordingly.

First Edition (January 1971)

This edition applies to release 20.0 of IBM System/360 Operating System.

The contents of this publication are subject ot change from time to time. Changes will be reflected in periodically updated editions. Before using this publication, consult the latest *System/360 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.

A form is provided at the back of this publication for your comments. If the form is missing, comments may be addressed to IBM Corporation, Publications Center, Dept. E01, P. O. Box 12275, Research Triangle Park, North Carolina 27709.

CONTENTS

ALPHABETIC LISTING OF TCAM MODULES BY CSECT NAME

FLOWCHARTS

FIGURES

FOLDOUT CHARTS

Organization and Use of the TCAM Program Logic Manual

This seven-part publication covers the internal logic of the IBM System/360 OS Telecommunications Access Method (TCAM). The TCAM PLM is directed to the IBM customer engineers and system engineers who provide program maintenance and who need information on the internal organization and logic of TCAM.

Section 1 is the Introduction to the TCAM system. The general information presented in the Introduction is basic to an understanding of TCAM. This information places TCAM in the proper perspective to the Operating System (OS) and points out the special concepts and control areas used by TCAM in order to operate as a component of OS.

Section 2, the Method of Operation section, describes the functional flow of each operation in a TCAM system. When possible, the operations are discussed in sequential order by time of occurrence as a message is being processed by TCAM. Each discussion is accompanied by Method of Operation diagrams, which depict the operation. (These diagrams are foldout charts and are located between Appendix D and the Glossary at the back of this manual.) The main-line processing operations are discussed in the following order:

1. Disk message queue initialization

2. Initialization of a Message Control Program (MCP)

3. Message handling in an MCP

4. Closedown of an MCP

The other functional operations occur intermittently with the main-line processing and except for system control, are discussed after the MCP sections. System control is discussed after the MCP initialization section. These operations include:

1. System control

2. Application program processing

3. Operator Control processing

4. Checkpoint processing

5. Error recovery procedures

6. Time Sharing Option interface

Section 3 covers the program organization and operation, both in textual descriptions and in flowcharts. Each TCAM module is described within its functional area of operation. The functional areas are organized exactly as in the Method of Operation section and thus allow the reader to relate actual modules to general functions. When a module name ends in two letters or in one or two letters followed by a number, the flowchart identification is the same as those characters. When multiple flowcharts are necessary for a module, these two or three characters are followed by a dash and then a number (HM1-1). When a module name ends in twc numbers, the flowchart identification is arbitrarily assigned.

The informaticn on a TCAM-TSO mixed environment is located in two places in Section 3. When a TCAM module contains logic necessary to identify that TSO is in operation and to activate special TSO routines, that module description and flowchart describe the tests. The special TSO routines that operate under the TCAM Dispatcher, but that perform TSO-only functions, are described in a section devoted solely to TSO routines. There is also a general discussion of the TCAM-TSO interface in Section 2.

Section 4 is the TCAM Microfiche Directory. This directory is a list of all TCAM modules. Each entry contains the correspcnding entry point or entry points, its generic name, its flowchart identification, and its CSECT name.

Section 5 is a composite of the data areas that are used by TCAM. Each data area is described in terms of purpose, internal references, allocation, and initialization. Both a visual and a tabular description of the DSECT for each area are also given, where applicable.

Section 6 contains tables of information to aid in debugging and analyzing the activity of TCAM.

The seventh section consists of information to aid in the use of TCAM. This information is in four appendixes: a list of TCAM queues and OCBs, a list of TCAM modules by library, a list of TCAM relative priorities, and the TCAM channel programs.

SECTION 1: INTRODUCTION

This section provides general information describing the purpose, organization, and internal operation of the Telecommunications Access Method (TCAM), and its relationship to the operating system.

PURPOSE OF TCAM

TCAM is a component of the IBM System/360 Operating System. The primary purpose of TCAM is to provide a high-level access method to communicate with telecommunications equipment while maintaining the greatest possible amount of device independence. In addition to supporting the transfer of data (messages) between both local and remote terminals and the system, TCAM provides a high-level, flexible message control language that can be used to direct the processing of the data. By using the TCAM macro instructions, installation-oriented message control is achieved.

SYSTEM STRUCTURE

TCAM operates under OS MFT or MVT in System/360 Model 40 or above processors. The minimum main storage requirement is 128K bytes. In addition to the system timer and normal OS requirements, TCAM requires a 2701, 2702, or 2703 on a multiplexer channel (unless only the 7770 or 2260 Local terminals are used, in which case the 7770 or 2848 is attached to the channel). Secondary storage for libraries and main or secondary storage for queuing are also required.

This section describes the various parts of TCAM and explains what they are, where they come from, how they get into the system, their relationships to each other, and how they pass control back and forth.

Figure 1 shows the steps necessary to begin processing in the TCAM environment.

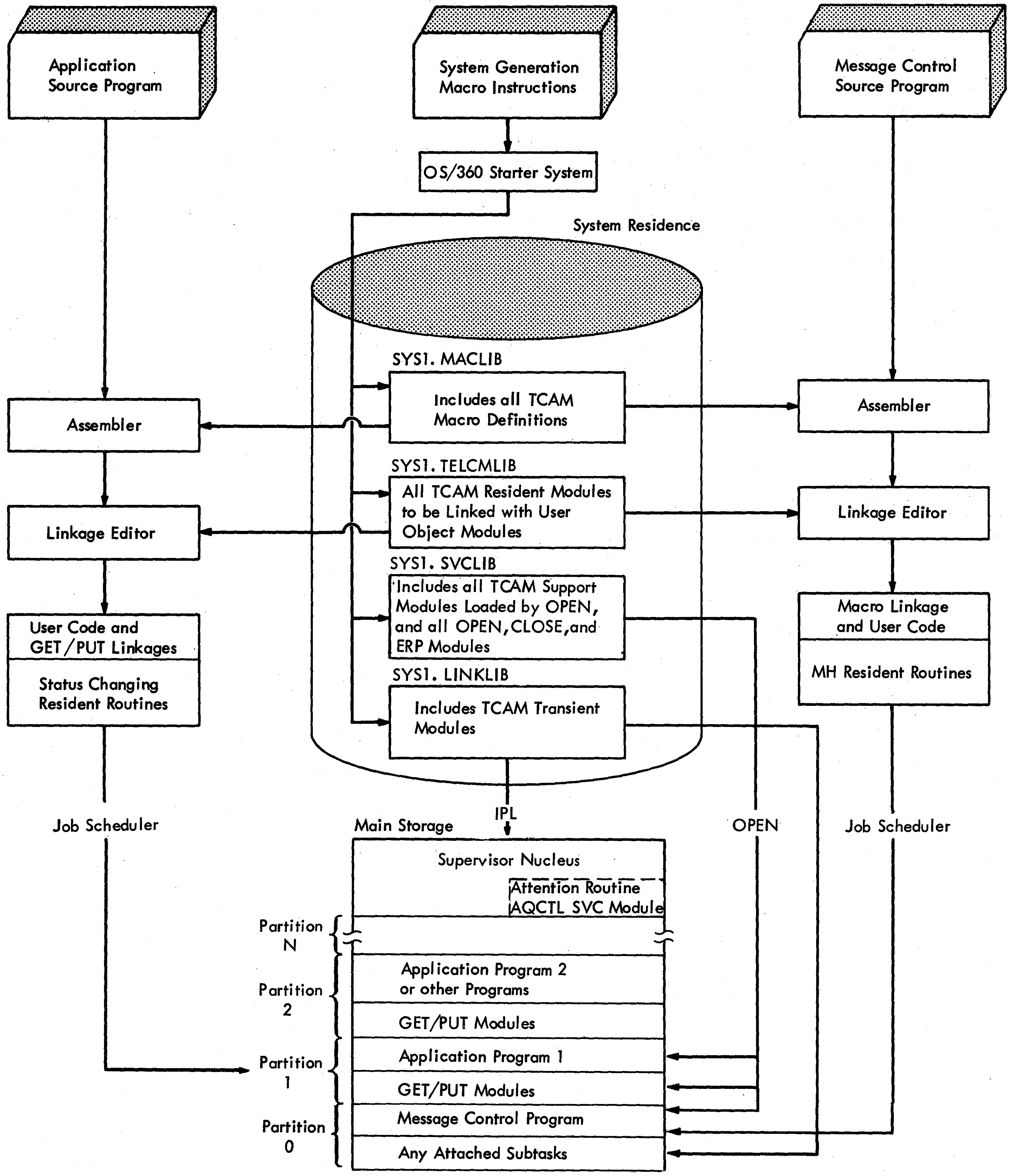

Application Source Program
System Generation Macro Instructions
Message Control Source Program
OS/360 Starter System
System Residence
SYS1. MACLIB
Includes all TCAM Macro Definitions
Assembler
Assembler
SYS1. TELCMLIB
All TCAM Resident Modules to be Linked with User Object Modules
Linkage Editor
Linkage Editor
SYS1. SVCLIB
Includes all TCAM Support Modules Loaded by OPEN, and all OPEN, CLOSE, and ERP Modules
User Code and GET/PUT Linkages
Status Changing Resident Routines
Macro Linkage and User Code
MH Resident Routines
SYS1. LINKLIB
Includes TCAM Transient Modules
IPL
Job Scheduler
Main Storage
OPEN
Job Scheduler
Supervisor Nucleus
Attention Routine
AQCTL SVC Module
Partition N
Partition 2
Application Program 2 or other Programs
GET/PUT Modules
Partition 1
Application Program 1
GET/PUT Modules
Partition 0
Message Control Program
Any Attached Subtasks

Figure 1. Physical Organization of TCAM

SYSTEM GENERATION

When TCAM is called for during a system generation procedure (via the ASCMETH operand in the DATAMGT system generation macro instruction), the TCAM modules are included in four libraries: SYS1.MACLIB, SYS1.TELCMLIB, SYS1.SVCLIB, and SYS1.LINKLIB. An Attention routine and a Type I SVC module (the AQCTL SVC 102 routine) are incorporated in the Supervisor Nucleus (SYS1.NUCLEUS). Using these modules, the user can assemble, linkage edit, and execute TCAM message control and application programs.

TCAM Macro Definitions

The operating system macro definition library (SYS1.MACLIB) includes the macro definitions necessary for the assembly of TCAM message control and application programs.

TCAM Resident Modules

When performing a system generation to include TCAM, the user must define a special library area named SYS1.TELCMLIB. During the generation run, modules that can later be linkage edited with message control and application object modules are copied from SYS1.CQ548 into SYS1.TELCMLIB. In this publication, these modules are defined as the TCAM resident modules. Appendix A contains a list of the modules in SYS1.TELCMLIB.

TCAM Support Modules

During the system generation run, all modules that are loaded into main storage by the various system open executors and the TCAM open and close executors are copied from SYS1.CQ548 into SYS1.SVCLIB. The TCAM Dispatcher, the Command Scheduler, the Type IV SVC modules, and the Error Recovery Procedure routines are also placed in SYS1.SVCLIB. In this publication, these modules are defined as TCAM support modules. Appendix A contains a list of the TCAM support modules in SYS1.SVCLIB.

The Error Recovery Procedure routines and the TCAM open and close routines can, at the option of the user at system generation, be resident or transient during program execution. In either case, these routines reside in SYS1.SVCLIB.

TCAM Transient Modules

At system generation time, modules that can be called into main storage for a limited length of time during the execution of a TCAM message control or application program are copied from SYS1.CQ548 into SYS1.LINKLIB. In this publication, these modules are defined as TCAM transient modules. Appendix A contains a list of the modules in SYS1.LINKLIB.

The Operator Control, Checkpoint, and On-line Test routines stored in SYS1.LINKLIB can optionally be specified to be resident during program execution. However, in this publication they are defined as transient modules.

System Nucleus Modules

At system generation time, the Attention routine and the AQCTL SVC 102 routine (a Type I SVC) are copied from SYS1.CQ548 into SYS1.NUCLEUS. In this publication these two modules are defined as the *system nucleus modules*.

THE MESSAGE CONTROL PROGRAM IN THE SYSTEM

Assembling and Linkage Editing a Message Control Program

The user codes the TCAM macro instructions necessary to design a message control program. When these instructions are entered for assembly, the output of this assembly includes: several tables and control blocks, linkages to TCAM resident and support routines, message handler (MH) macro instruction expansions, and any user-written routines that were included.

The assembled object module is then linkage edited to include the referenced resident routines from SYS1.TELCMLIB. These resident routines are the MCP routines used to process header information, to translate from one transmission code to another, to direct messages to the proper lines and queues, to manage system resources, etc.

The resulting load module is stored in a system library to be loaded for execution.

Execution of a Message Control Program

The TCAM message control program (MCP) is normally executed as the highest priority task in the highest priority partition or region in the system. The OS Initiator/Terminator routine loads and transfers control to the MCP. The first TCAM macro instruction executed must be INTRO. The initial functions of INTRO are to establish the TCAM Address Vector Table (AVT), addressability and entry linkages for the MCP, the Cross-Reference Table, the Channel Program Block (CPB) pool, the buffer unit pool, and main storage queues. INTRO also attaches the Operator Control, FE Common Write, and On-line Test tasks and provides override of some INTRO parameters via the Write to Operator with Reply (WTOR) Interpreter routine. These functions are discussed in detail under Functions of INTRO in the Method of Operation section of this publication.

The MCP runs under the control of the OS task management routines. It is scheduled and dispatched according to the priorities included in the Task Control Block (TCB) in the partition in which it is being executed. The MCP includes:

1. The object module output from the assembly of the user's code.
2. The resident routines linkage edited with the assembly output.

In order to understand the operation of an MCP, it is necessary to become acquainted with the use of save areas in the MCP and the way in which control is passed from one level of operation to another. Five save areas are located at the beginning of the AVT, which is assembled at the beginning of the MCP. The MCP is that portion of the user's CSECT that contains the INTRO, OPEN, READY, and CLOSE macros, the MH routines and macro expansions, and constant areas.

Save area management occurs when a subroutine returns to the routine that called it. A save area "belongs" to a routine when that routine sets register 13 to point to the save area. A subroutine of the routine can then store the registers of the routine in the specified save area. If a routine does not call a subroutine, it does not have a save area, since it does not modify the contents of register 13.

TCAM maintains four 18-word save areas and one 10-word save area in the AVT. After the standard entry linkage of a routine that uses save area management, certain words of the save area contain specific addresses:

- The second word of the save area points to the address of the save area for the calling routine.

- The third word of the save area for the calling routine has the address of the save area for the called routine.

- Register 13 has the address of the save area for the called routine.

During the standard exit linkage of a routine that uses save area management, the save area address for the calling routine is restored from the second word of the save area for the called routine. The registers of the calling routine are also restored from this area, and the calling routine can regain control.

As stated previously, when OS Job Management initiates an MCP, the MCP gains control at the INTRO macro expansion. In performing standard entry linkages, the INTRO macro expansion sets register 13 to point to the first field of the AVT, AVTSAVE1, which is the save area that belongs to the MCP. When the functions of the READY macro are executed, the MCP calls the TCAM Dispatcher. The TCAM Dispatcher performs standard entry linkage, saving the registers of READY in AVTSAVE1 and setting register 13 to point to the Dispatcher save area, AVTSAVE2.

Routines, subroutines, and subtasks use the AVTSAVE3 and AVTSAVE4 save areas if they need to perform save area management.

When a disabled routine, an appendage, gains control, it uses AVTSAVEX, the ten-word save area, to store the I/O Supervisor registers.

THE APPLICATION PROGRAM IN THE SYSTEM

Assembling and Linkage Editing an Application Program

A TCAM application program processes messages obtained from a TCAM MCP. The application program can run in a partition or region different from the MCP, or it can run as an attached task in the same partition or region.

An application program needs only the OPEN, CLOSE, GET, and PUT macro instructions and some data set definition macro instructions. When this is the case, no resident routines need to be linkage edited with the object module. However, the user may wish to write application programs that use the following macro instructions to examine and modify the status of the MCP:

- CHECK
- CKREQ
- ICOPY
- MCPCLOSE
- MRELEASE
- POINT
- QCOPY
- TCHNG
- TCOPY

When any of these macro instructions are used, the linkage editor includes the corresponding resident modules in the load module. The load module is stored in a system library from which it is loaded for execution.

Execution of an Application Program

It is possible to run an MCP with no application program, but there may be one or more application programs being executed asynchronously with the MCP.

In most cases an application program is loaded into the next highest priority partition to the MCP. However, application programs may also be executed in the same partition as the MCP after being brought in by the system ATTACH facility.

Application programs, like the MCP, run under the control of the OS task management routines. They are scheduled and dispatched according to the priorities indicated in the Task Control Blocks (TCBs) for the partitions in which they are being run.

An application program includes:

1. The object module output from the assembly of the user's code.

2. Any resident routines linkage edited with the assembly output.

3. The CHECK, POINT, GET/READ, and PUT/WRITE routines.

The primary difference between a TCAM application program and any other processing program is the requirement for and the implementation of inter-partition communication.

The various macro instructions that can be used in an application program are handled as follows:

1. TCOPY, ICOPY, and QCOPY. The corresponding resident routine for each of these macro instructions copies the requested information from the MCP partition, using address pointers stored in the AVT and in the Terminal Table. These tables are located via the Communications Vector Table (CVT).

2. All other macro instructions. The routines invoked by the remaining macro instructions cause SVC TYPE I interruptions to the supervisory routines. A module within a partition can move data or control information from another partition into its own partition; however, that module must use an SVC either to move data from its own partition into another partition or to move data within another partition.

RELATIONSHIP OF THE OS DISPATCHER TO TCAM

The Operating System (OS) gains control from the TCAM task when the TCAM Dispatcher finds no elements on its ready queue and subsequently issues an OS WAIT macro. This indicates that the MCP has no work to perform. When OS gains control, it examines all the ready tasks in the system and passes control to the one with the highest priority.

When a TCAM appendage has work for the MCP, it invokes the OS Post routine via a branch entry point to post the MCP Event Control Block (ECB). This indicates to the OS Dispatcher that the MCP now has work to do and is vying for control of the system. OS can pass control to the TCAM task when it is the highest priority task that is ready to be activated. TCAM resumes execution at the instruction following the WAIT that gave control to OS.

TCAM posts the ECBs for its attached tasks when they are to be activated. When TCAM subsequently issues a WAIT, the attached tasks can vie to gain control from OS.

THE TCAM DISPATCHER

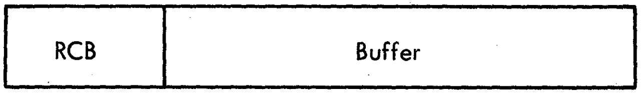
The following sections describe the tools and mechanisms by which the TCAM Dispatcher, or control module, allocates and schedules system resources, that is, CPU processing time, main storage, I/O paths, and elements (primarily buffers and lines). The key to the mechanism is the *ready queue*, through which a resource is allocated to a subtask.

The mechanisms of allocation are the "twait" and "tpost" functions performed by the TCAM subtasks. A *twait* schedules a subtask to be activated when a specific resource is available; a *tpost* passes an available resource to the ready queue. The actual implementation of twait and tpost are not exclusive functions of the subtasks; rather, the subtasks return to specific entry points in the TCAM Dispatcher to indicate the status of the resource. *Dispatching* is the process of providing a routine with an element and giving the routine control to handle the element.

A detailed discussion of the TCAM Dispatcher is included under System Control in the Method of Operation section of this publication.

Elements, Queues, and Subtasks

The physical resources of the system are composed of *elements* (for example, the buffer pool, a resource, is broken into individual buffers, the elements) with each element represented by a resource control block (RCB). An RCB is an 8-byte prefix to an element. The first four bytes are a pointer to the queue control block (QCB) that the element is to be associated with; the last four bytes contain a priority byte and a link field.



There is at least one *subtask* that works with every type of element in the system. These subtasks are represented by subtask control blocks (STCBs).

The elements, and the subtasks that operate on these elements, are associated with one another by a third control block, the queue control block (QCB). Thus, a QCB has a pointer to the chain of elements under its control and a pointer to the chain of STCBs for subtasks waiting to operate on these elements. The chains are referred to as *queues*. Figure 2 illustrates the linkage of these queues to a QCB.

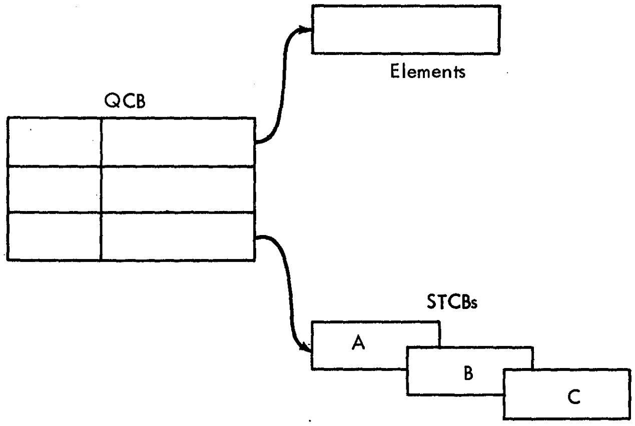


Figure 2. TCAM QCB Linkage

When a subtask needs an element, it can request one from the QCB that handles that particular element by tpcsting a request element to that QCB or it can insert its STCB into the STCB chain of the QCB to twait for the element. When the element is available, the subtask is dispatched.

When a subtask has finished using an element, it gives (tposts) the element to the appropriate QCB. The TCAM Dispatcher gives this element to the first (highest priority) subtask in the STCB chain of the QCB. In this case, Subtask A in Figure 3 is dispatched. The subtask associated with STCB B in Figure 3 can be dispatched if Subtask A indicates to the TCAM Dispatcher that it does not need to process the element. The STCB chain ends with a permanent STCB. STCB C in Figure 3 remains the last STCB in the chain. STCB C might point to a routine that does nothing more than chain elements into the QCB element chain. Subtask C has a lower priority than any other subtask that might use the element and, therefore, is dispatched only if each of the higher priority subtasks bypasses processing.

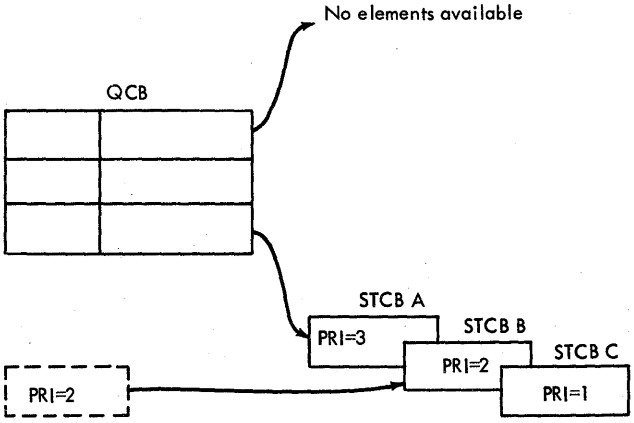


Figure 3. Priority of Subtasks on a QCB

Figure 4 demonstrates the linkage when an element processed by Subtask X is tposted to the QCB and placed on the element chain by Subtask C. Subtask C can place the element in the QCB element chain only if Subtask A and Subtask B do not need the element and pass it down the chain to Subtask C.

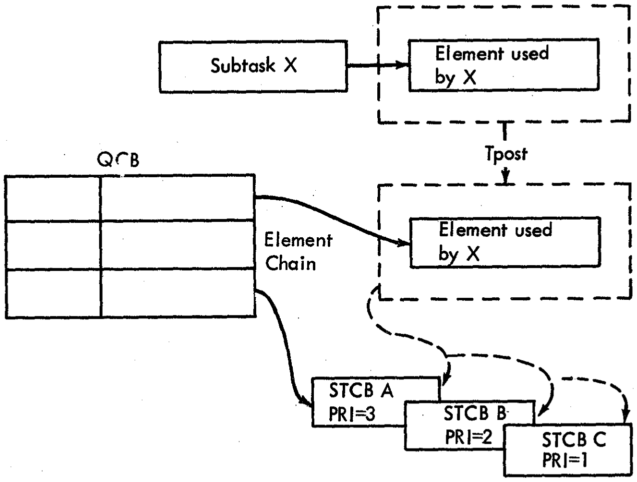


Figure 4. Passing Elements to a QCB

To illustrate the basic sequence of events involved when the TCAM Dispatcher processes an element, the procedure can be compared to a postal service system. The people that mail and receive letters are subtasks. Each letter is an element, the address on a letter is its QCB, the post box is the ready queue, and the mail box at the destination is the appropriate STCB. When a letter is mailed (tposted), it becomes the property of the post office (the TCAM Dispatcher). The post office examines the address (the QCB) and directs it to its destination (the STCB). When the letter is delivered (dispatched), the person represented by the address (the subtask) can examine it. Figure 5 illustrates this analogy.

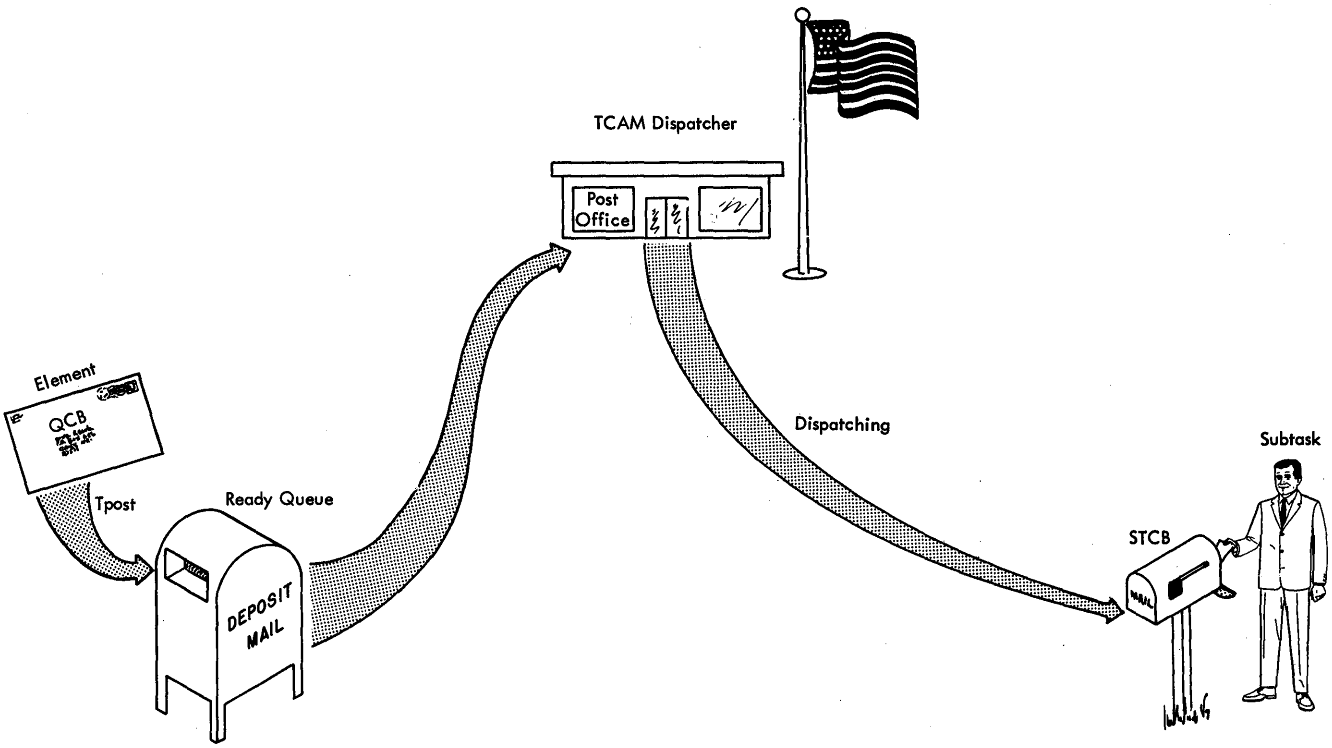


Figure 5. TCAM Dispatcher Analogy

The Readv Oueue

The previous discussion points out that subtasks gain control from the TCAM Dispatcher dependinq on:

1. The availabilitv of elements, and

2. The prioritv of the STCB for the subtask.

The TCAM messaqe control program is responsible for allocating CPU processinq time to the various tasks under its control. The mechanism it uses is called the readv queue (as discussed later, there are actually twc ready queues).

The ready queue is a chain of elements that represent all the work to be done in the TCAM system. The work to be done is represented by the various elements (RCBs) that appear on the ready queue in priority order. The purpose of the ready queue is to ensure that all elements are processed and dispatched with full respect to priority and without one impacting the resources of another.

To support dispatching while enabled for interruption, TCAM uses two readv queues. One is designated to be used by disabled appendages or by the disabled AQCTL SVC 102 routine for tposting elements, while the other is used by enabled routines. Although the two ready queues are not manaqed by the same technique, each is a ready queue because it contains elements (RCBs) to be processed by the various subtasks.

TCAM manages the disabled ready queue by the first-in-first-out (FIFO) technique. The queue itself consists of twc words: a one-word pointer to the first and a one-word pointer to the last element on the queue. Disabled appendages place an element (RCB) on the disabled ready queue by linking the new element to the element pointed to by the second word of the queue and by then updating the second word to point to the new element.

TCAM manages the enabled ready queue by the priority-FIFO technique. The TCAM Dispatcher has the responsibility for merging the disabled into the enabled ready queue just prior to dispatching. Dispatching is always handled from the enabled ready queue, and unless specified otherwise, this is the cne referred to as the ready queue.

The TCAM Dispatcher manages the ready queue by attempting to execute the subtask associated with the highest priority element on its chain. Since the element has an RCB as its prefix, the Dispatcher can refer to the correct QCB in order to pass control to the first subtask represented in the STCB chain cf the QCB. The subtask processes the element and then returns control to the TCAM Dispatcher, which can then examine the next element cn the ready queue. A discussion of the way the TCAM Dispatcher manages the ready queue is included under System Control in the Method of Operation section of this publication.

Principle of Tpost and Twait

The technique fcr passing an element from cne queue to another queue is called tposting. When the subtask that an STCB points to finishes processing an element and wishes to allow another routine to process that same element, the subtask tposts the element to the second routine. The subtask achieves the tpost by placing in the RCB of the element a pointer to the QCB that ccntrols the STCB for the new routine, and by then returning to the TCAM Dispatcher with an indication that the element is to be placed cn the ready queue.

The second technique for handling resources is called twaiting. When a subtask needs elements to process, it returns control to the TCAM Dispatcher indicating that it has finished the processing that it can do at this time. The twait is implemented by the TCAM Dispatcher. The Dispatcher places the STCB for this subtask in the STCB chain of the QCB to which the resource that the subtask needs tc complete processing will be tposted. When an STCB is in the STCB chain of a QCB and the subtask for that STCB does not have control, the subtask is twaiting.

When an application program needs either to place an element on the disabled ready queue, to post an Event Control Block (ECB) complete, or tc move data from one partiticn to another, a special technique is used. This technique is performed by the AQCTL SVC 102 routine, which uses pointers in the AVT to refer to the disabled ready queue. Since AQCTL is a resident Type I SVC, the actual processing occurs in the OS Supervisor, out of the control of either the application prcgram or the MCP. A detailed discussion cf the AQCTL SVC 102 routine is included under System Control in the Method of Operation secticn of this publication.

TCAM CONTROL AREAS

A TCAM control area is a storage area through which a particular type of informaticn required for control of the TCAM system is communicated among its parts. There are several principal control areas used by TCAM:

- TCAM Address Vector Table
- Invitation List
- Termname Table
- Terminal Table
- Cption Table
- Option Characteristics Table
- Device Characteristics Table
- Special Characters Table

- Translation Tables

- Resource Ccntrol Block

- Subtask Ccntrcl Blcck

- Queue Control Block

- Line Contrcl Elock

- Station Ccntrcl Blcck

- Channel Prcqram Blcck

- Element Reauest Block

- Process Contrcl Block

- Operator Control Address Vector Table

TCAM ADDRESS VECTCR TABLE

The TCAM Address Vector Table (AVT) is a local constant area assembled in the MCP. When the functions of the INTRO macro expansion are executed , the AVT is initialized and fcrmatted. At message queues open time, a pcinter to the word that contains the address of the AVT is placed in the system Communication Vector Table (CVT). The first entries in the AVT are initialized from the parameters of INTRO, and other entries are made durinq the assembly of other macros coded by the user.

The AVT provides work areas in which TCAM rcutines can store variables. The AVT also contains constant areas shared by more than one macro expansicn or TCAM subroutine. The AVT contains five save areas - one for the MCP, one for each level of control in the MCP, and one for disabled code. (The levels cf control in the MCP are discussed under System Structure in the Introduction section of this publication.) For efficient internal ccntrol, the AVT also contains module addresses, special elements, contrcl bytes/tits, and the two readv queues.

The format of the AVT is in the Data Area Layouts section of this publication.

INVITATION LIST

The INVLIST=(name of list,...) operand of a DCB macrc specifies the names of the invitation lists for the lines of the line qroup represented by the DCB. There is one invitation list for each line in a line qroup, and the DCB contains a pointer tc the ccntrol word of each of its invitation lists. An INVLIST macro specifies the actual entries in each invitation list.

An invitation list is a list of the invitation (polling) characters for terminals that may generate messages to the CPU on the same line. The order in which the invitation characters of the terminals are listed determines the crder in which the terminals on the line are pclled.

Invitation lists may contain both active and inactive entries. Active entries are those invited to enter a message on each pass through the list; an X'FE' follows the last active entry. An inactive entry is one that is not currently being invited to enter messages. Inactive entries in the list are located after the X'FE' indicator. The methods of establishing and altering the status of the entries in the invitation list are discussed in the section on Invitation in the System/360 OS TCAM Programmer's Guide, Order No. GC30-2024.

The general fcrmat of an invitation list is eight bytes of control information, fcllowed by an invitation list entry for each active terminal on the line, followed by an end-cf-list indicator (X'FE'), followed by an entry for each inactive terminal on the line.

An invitation list with 'n' active entries has the following format:

-2n		-4	-2	0	+4	+8					
Reln	•••	Rel 2	Rel 1	Control Word	CPU ID	Invchars	1	Invchars	2	n	X 'FE'

Rel1-Reln are the two-byte relative pcsiticns in the Termname Table for the entries represented by the invitation characters. There is one two-tyte field for each entry in the invitation list, in reverse order.

Control Word is a field defining the status of the invitation list. (See format belcw.)

CPU ID, for dial terminals, is the address of a field that contains the ID sequence assigned to the computer. The referenced field contains a length byte, which specifies the number of bytes in the ID sequence, followed by the ID sequence itself. For buffered terminals, the CPU ID field in an invitation list has the following format:

Offset	+4	+5	+6	+7
	Active Count	UCB Status	Reserved	Terminal Count

Active Count is the number of active terminals on the line to which TCAM is currently sending. This field is initialized to zero at line open time.

UCB Status is set to X'01' at line open time if the UCB for the line indicates Autc Poll. Otherwise, this field contains X'00'.

Terminal Count is the total number cf terminals on this line. This field is initialized at line open time.

Invchars are the invitation or polling characters to be used for the terminal. The one-byte index following "Invchars" points to the corresponding relative position field that precedes the control word.

X'FE' is the end-of-list indicator, which is used to separate active and inactive entries. An EOT character precedes the X'FE' as an end of transmission character in an invitation list for BSC Auto Poll terminals.

The control word of an invitation list has the following format:

Offse:	0	+1	+2	+3
	Total Entries	Active Entries	Width	Status

Total entries indicates the number of active and inactive entries in the list (if this byte is equal to zero, the list is for an output-only line; there is no message traffic from the terminals).

Active entries indicates the number cf entries currently being invited. If byte 1 is equal tc zero, all the entries in the list are inactive.

Width indicates the size of each entry in the list (the size includes the one-byte index that follows the invitation characters).

Status indicates whether the list is active or inactive and whether it is being autopolled.

Status bits	Meaninq
0	ON - 'EOT=' was specified on the INVLIST macro OFF - 'EOT=' was not specified on the INVLIST macro
1	ON - Offsets to the Termname Table entries have been sorted OFF - Offsets to the Termname Table entries have not been sorted
2-4	Reserved
5	Indicates whether the list has been processed by Checkpoint/Restart
6	ON - Active list OFF - Inactive list
7	ON - List is beinq autopolled OFF - Proqrammed pcll is in effect

The invitation list entries have the same format whether the terminals are under control of the Auto Poll facility, the proqrammed poll facility, cr otherwise (e.q., contenticn). The width of each entry is indicated in byte 2 of the control word.

The format cf each entry in an invitaticn list is:

Invitation Characters	K

The invitation characters (polling characters) are in the hexadecimal fcrm cf the transmissicn code. K is the one-byte index field used to indicate the relative position of the entry in the list and to find the two-byte pointer tc the correspondinq entry in the Termname Table.

TERMNAME TABLE

The Termname Table contains the names cf all the terminals in the system in collatinq sequence.

The table is qenerated at assembly time from the names of the Terminal Table entries in the TERMINAL macros in the order in which they are named. The names of queues for applicaticn programs, of queues for lcqqinq media, and of certain lists of terminals are included, in addition to the names of terminals and terminal components. See the publication <u>System/360 OS TCAM Proqrammer's</u>

Guide, Order Nc. GC30-2024, for informaticn about specifying the names of the terminals, terminal components, etc.

During the execution of the functions of the INTRC macro expansion at MCP initialization time, the names in the Termname Table are sorted into collating sequence to permit binary searches for locating terminal names and for finding terminal-dependent information.

The beginning of the Termname Table contains code (the Termname Table Code - IEDQTNT) that is used to ccnvert the invitation list relative position field to the address of the corresponding entry in the Terminal Table. After the code there are two bytes of control information for the Binary Search routine. The next fields in the Termname Table contain the number of bytes in the name of an entry, the address of the middle entry in the table, and the total number of entries in the table. Each entry consists of the terminal name and the three-byte address of the Terminal Table entry for that terminal. The length of the field for the terminal name is determined by the longest terminal name; each terminal name field is as long as the longest name (the names are padded with blanks on the right, if needed).

From the address field of a Termname Table entry, TCAM can locate the corresponding Terminal Table entry, which consists of blocks of information about each terminal.

Figure 6 shows the relationship of pointers from an invitation list to the Termname Table. The format of the Termname Table is shown in the Data Area Layouts section of this publication.

TERMINAL TABLE

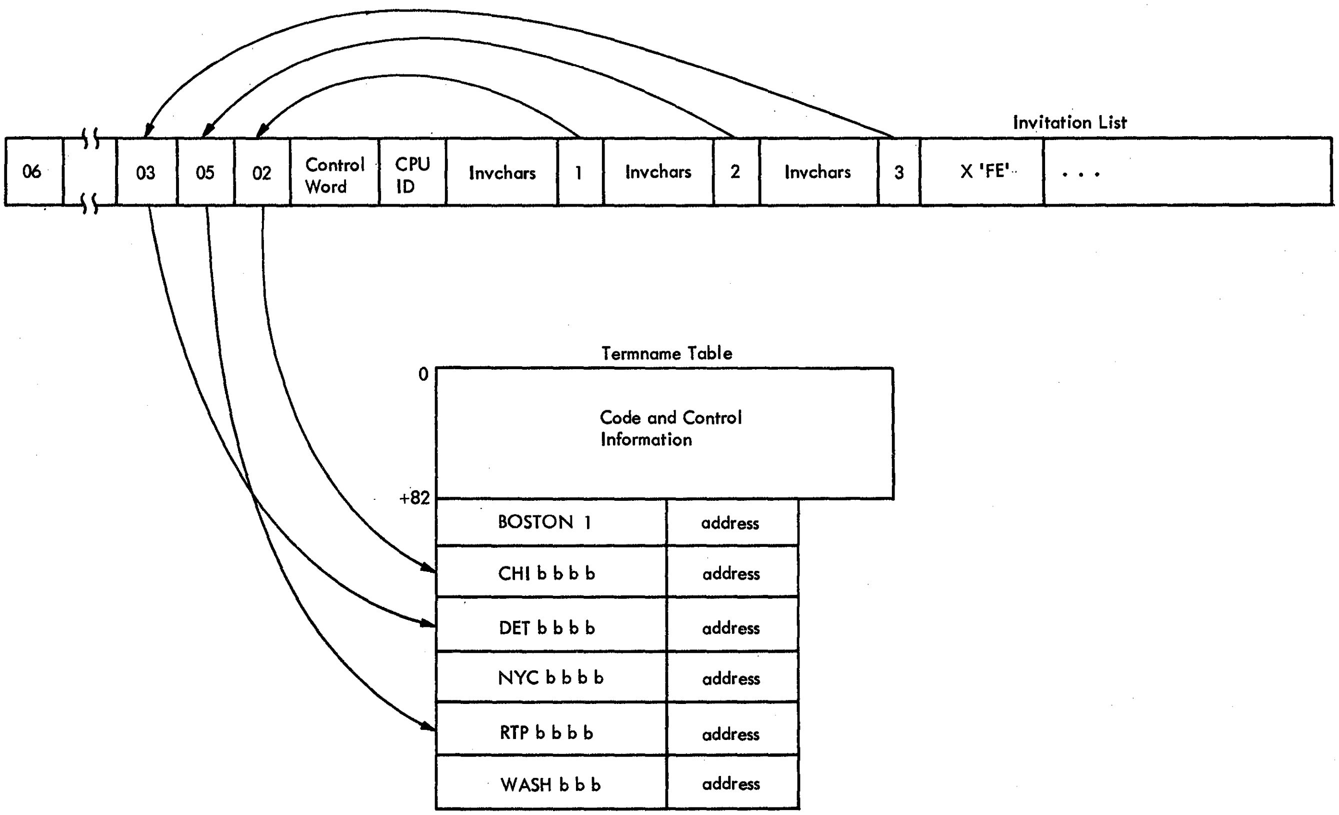
The Terminal Table consists of blocks of device-dependent information about each terminal in the TCAM system; each such block is called a terminal entry. There are six types of terminal entries, each of which is discussed later in this section.

The size, structure, and contents of the Terminal Table are based on informaticn prcvided by the user thrcugh the TTABLE, OPTION, TERMINAL, TLIST, TPROCESS, and LOGTYPE macro instructions. TTABLE is specified once and defines the limits of the table. One TERMINAL macro is issued to create each single cr group entry. OPTION macros and data supplied by TERMINAL and TPROCESS operands cause storage to be allocated for any option fields to be included in the Option Table for a Terminal Table entry. The option fields can contain information needed to perfcrm varicus optional functicns provided by TCAM or the user. The initial contents of each opticn field are specified by the TERMINAL or TPRCCESS macro that defines the entry. TLIST defines a distribution or cascade entry (defined below). TPROCESS creates an entry for an application program. LOGTYPE creates an entry for logging messages.

Each entry in the Terminal Table begins on a fullword boundary.

The formats of the various types of terminal entries, with notes concerning Option Table implicaticns, are included in the Data Area Layouts section of this publication.

There is one terminal entry for each terminal in the system, and each Terminal Table entry is referred tc via a pointer from the Termname Table. Fiqure 7 shows the relaticnship between the Termname Table and the Terminal Table.



Fiqure 6. Pointers from an Invitation List to the Termname Table.

Sinqle Entry

A single entry in the Terminal Table defines a sinqle terminal or component. A sinqle entry must be defined for each terminal or component that can enter only, accept only, or both enter and accept messaqes (except for a terminal in a qroup entry, defined below). If a terminal compcnent is to be selected individually, the component must have a separate sinqle entry.

The format of a sinqle entry is the same as the qeneral Terminal Table format defined in the Data Area Layouts section of this publication. Bits 0 through 2 of byte 0 of the control information field are set to binary 000 to indicate a sinqle (or qroup) entry. If there is no option area for an entry, the offset and count fields are omitted. The required selection sequence field contains the selection characters for the terminal and, if it is a switched terminal, its telephone number and the number of dial digits.

A single entry in the Terminal Table is defined by a TERMINAL macro.

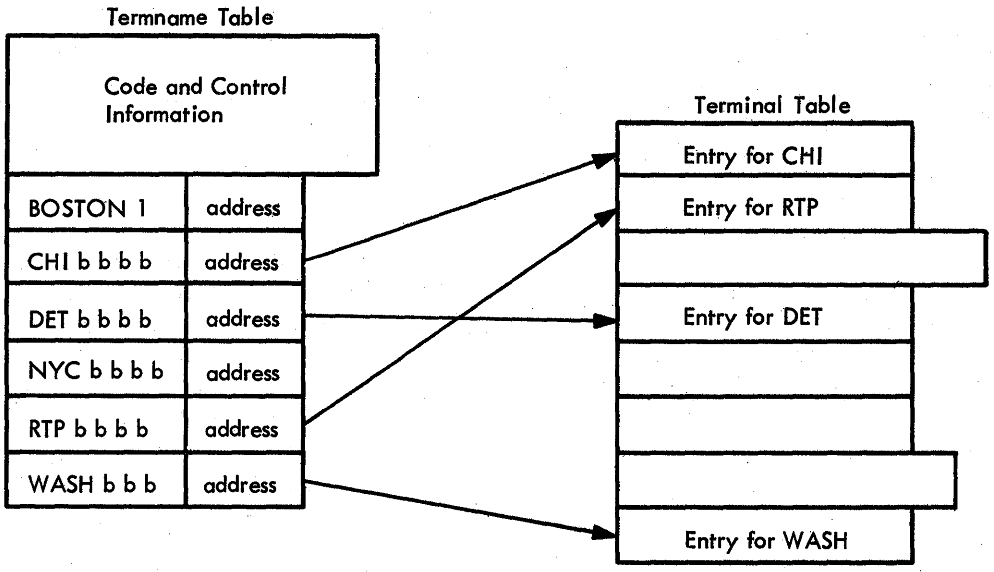


Figure 7. Pointers from the Termname Table to the Terminal Table

Group Entry

A group entry represents a prespecified group of terminals on a line that has special equipment to permit simultaneous transmission of a message to the group. A single set of unique addressing characters is used to contact the group. Several combinations of prespecified terminals can be grouped for this purpose. Each group has a group terminal name and a corresponding group entry in the Terminal Table. A group entry in the Terminal Table has the same format as a single entry, except that, since the entry is for output transmissions only, the input sequence counter field is not used.

A group entry is defined by a TERMINAL macro.

Distribution Entry

A distribution entry contains a list of pointers to single, process, or group entries. The pointers are grouped under the entry name. When a message contains a distribution entry name as its destination code, TCAM sends the message via separate transmissions to all destinations indicated by the list. Each terminal on the list must have a corresponding single or group entry in the Terminal Table. The TCAM MCP can only send messages through the distribution list method.

The format of a distribution entry in the Terminal Table is the same as that for a single entry, except that the setting of the status bits is binary 010, and the input sequence number field (bytes 4 and 5) contains a count of the entries in the list. Two-byte pointers to the single or group entries that make up the list follow this count field.

For distribution and cascade entries, bytes 1 to 3 contain the address of a distribution or cascade Destination QCB.

A distribution entry in the Terminal Table is defined by a TLIST macro.

Cascade Entry

A cascade entry is identical in appearance to a distribution entry, except for the status byte, but is handled differently. The message is queued for the available terminal that has the fewest messages queued for it in the list. An available terminal is one that is currently capable of accepting a message. The terminal must not be held. To be available, a dial terminal must not be involved in a time delay. If more than one of the available terminals have the same number of messages queued and that number is the fewest number of messages queued, the message is sent to the first of these terminals. If the message cannot be sent to any terminal at this time, it is queued for the first terminal in the list. The TCAM MCP can only send messages through a cascade list.

The format of a cascade entry is the same as that for a single entry, except that the setting of the status bits is binary 010 and the input sequence number field contains a count of the entries in the list. Two-byte pointers to the single or group entries that make up the list follow this count field.

A cascade entry in the Terminal Table is defined by a TLIST macro.

Process Entry

A process entry in the Terminal Table represents a queue of messages for an application program. There must be a process entry for each queue to which an application program can issue a GET or READ macro and at least one for all the PUT or WRITE macros from the same application program. The format for a process entry in the Terminal Table is the same as that for a single entry, except that the setting of the status bits is binary 001. Also, for a GET/READ operation, bytes 1 to 3 contain the address of the Destination QCB.

A process entry is defined by a TPROCESS macro.

Logtype Entry

A logtype entry in the Terminal Table represents a queue of messages for a logging medium. The setting of the status bits for a log entry is binary 011.

A logtype entry is defined by a LOGTYPE macro.

Line Entry

A line entry in the Terminal Table defines a switched line that is used for input operations. A line entry contains the device characteristics for stations that call in on a switched line before supplying identification and for stations that call in and never supply identification data.

The format of a line entry is the same as for a single or group entry except that the setting of the status bits is binary 100.

A line entry is defined by the UTERM operand on a TERMINAL macro.

OPTION TABLE

The user may specify an area to correspond to any entry in the Terminal Table for use by the COUNTER, ERRORMSG, FORWARD, MSGLIMIT, INSERT, PATH, REDIRECT, STARTMH, and other MH delimiter macro instructions issued in a message handler. The fields are generated by OPTION macros, which must be issued before the TERMINAL and TPROCESS macros that define the Terminal Table. One-byte offsets to these fields are placed in the terminal entry beginning at the TRMOPT label. The routine for the LOCOPT macro uses these offsets to locate the option field.

An OPTION macro defines each field in the Option Table. The macro names the option field and defines the type and length of the field. The OPTION macro generates a CSECT to contain the actual option data and another CSECT to contain the field name and characteristics.

Initial values for the option fields are specified via parameters of the TERMINAL or TPROCESS macros.

Each option field requires one OPTION macro. The order of the fields within the Option Table is determined by the order in which the OPTION macro instructions are specified. The first option field is generated on a doubleword boundary. The maximum size of the option fields for a given terminal is 254 bytes, including required boundary alignment.

For each OPTION specified, space for a one-byte offset is reserved in the offsets field of the Terminal Table entry. When the TERMINAL or TPROCESS macro that initializes the fields of the Option Table is issued, a two-byte offset is generated to the option table for this entry. If initial data is supplied, the option field is generated for the terminal cr process entry; if a comma is coded, the option field is not generated. If the field is generated, its offset is placed in the offset field cf the terminal entry; if the field is not generated, the offset field contains X'FF' to indicate that there is no field.

Each single, group, or process entry in the Terminal Table contains a one-byte offset in the offset field for each OPTION macro issued. The space needed for the Option Table depends on the number of fields initialized by the TERMINAL or TPROCESS macros, and on the size of the fields as specified by the OPTION macros.

All OPTION names are kept in a table with their numeric values. This table enables an cption field named in an Operator Control message to be located.

OPTION CHARACTERISTICS TABLE

The Opticn Characteristics Table is a variable length table that contains one entry for each OPTION macro issued by the MCP. The table allows TCAM routines to use the assembled name for an OPTION macro to locate the data for a specific terminal in the Option Table. Each entry in the Option Characteristics Table contains the length of the corresponding Option Table entry, the type of option field specified, and the user-specified name of the OPTION macro.

A field in the AVT contains the address of the Option Table, and the second word of the Opticn Table contains the address of the Option Characteristics Table. Storage is allocated for and the table is initialized at assembly time.

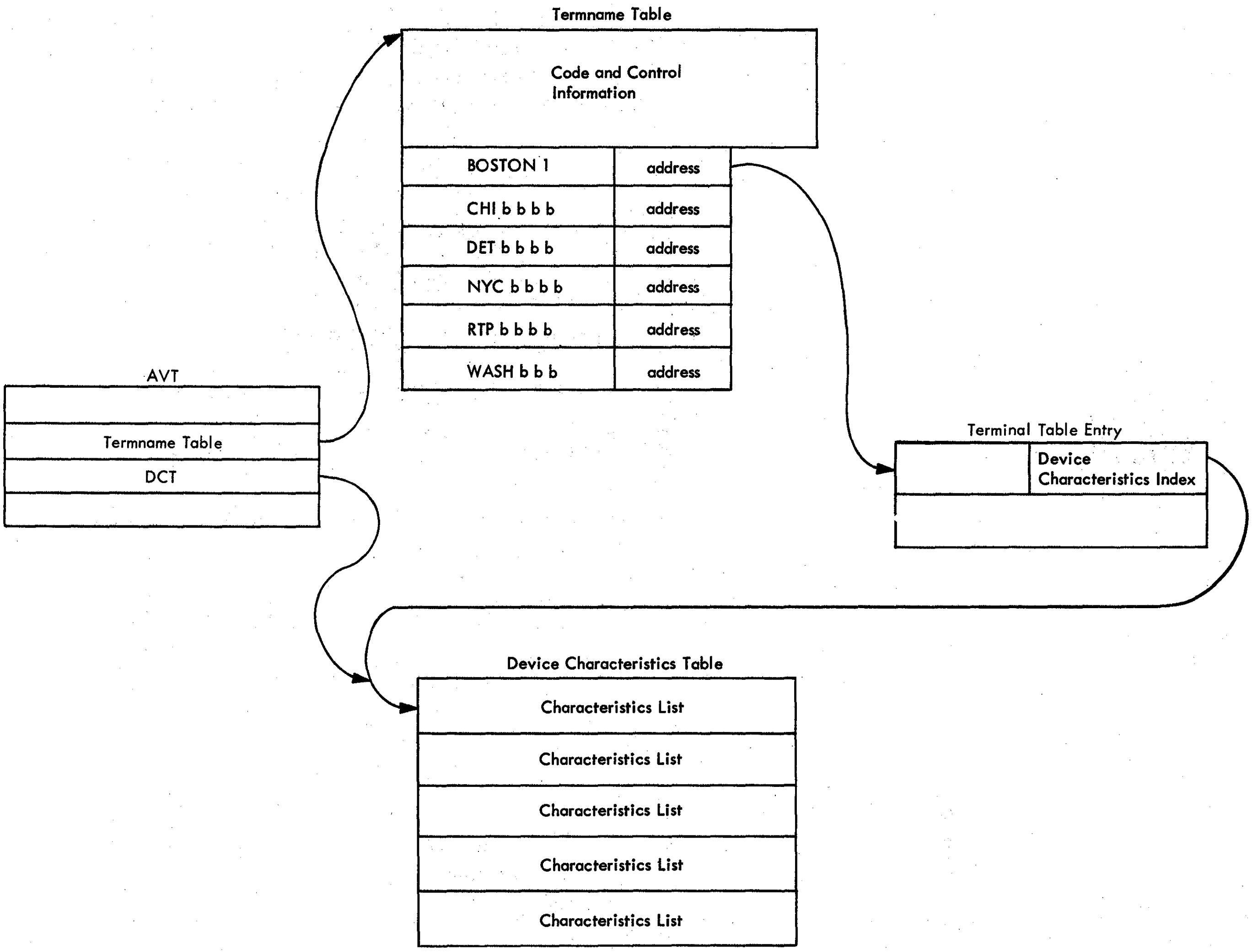
DEVICE CHARACTERISTICS TABLE

The Device Characteristics Table (DCT) consists of entries that describe the characteristics of the terminals in the system. A pointer in the AVT and a one-byte index in the Terminal Table entry are used to gain access to the entries in the DCT. A single four-byte entry is generated for all terminals that have identical characteristics.

The DCT is generated by the specifications of the TERMINAL macros. Figure 8 shows the relationships amcng the AVT, the Termname Table, the Terminal Table, and the DCT.

SPECIAL CHARACTERS TABLE

A special characters table (SCT) consists of entries that contain the special characters required for device I/O for a specific line group. The SCT for a line group is located via a three-byte address in the DCB for that line group. The DCB for the line group is located through a pointer in the LCB.

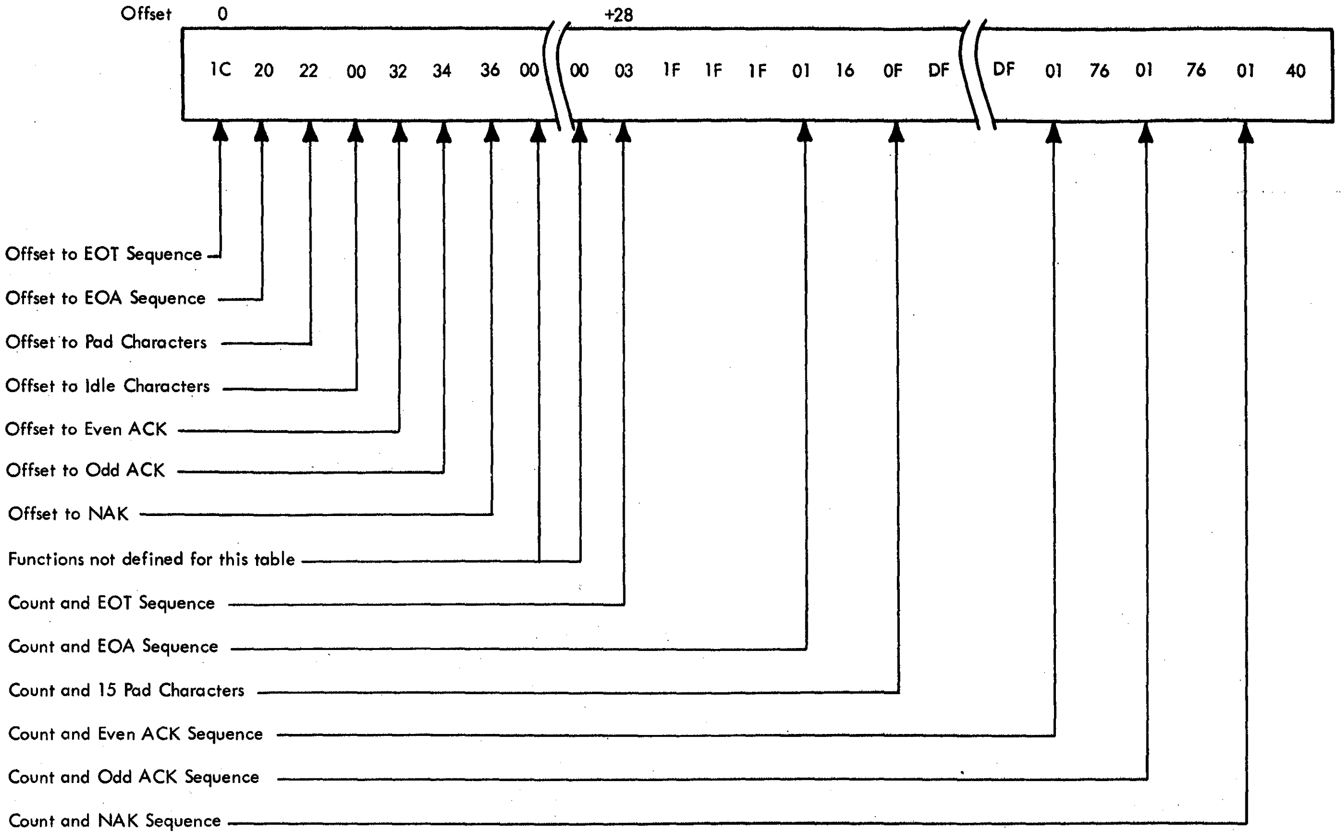


Fiqure 8. Relationships among the AVT, the Termname Table, the Terminal Table, and the DCT

An SCT is variable in lenqth since the special characters needed by each terminal type vary.

The beqinninq of an SCT consists of 28 one-byte offsets, each of which when added to the SCT pcinter in the DCB, points to a one-byte lenqth field followed by a special characters entry. There are as many entries in an SCT as there are different sets of special characters needed. If a function is not defined for the associated line qroup, the one-byte offset field ccntains X'00'.

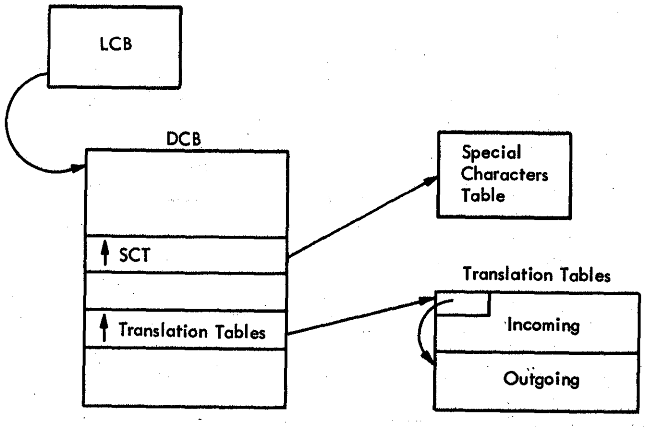
Fiqure 9 provides an example of a special characters table entry. Fiqure 10 describes the relationship among an LCB, a DCB, the Translation Tables, and an SCT.



Fiqure 9. Fxample of a Special Characters Table Entry

TRANSLATION TAELES

The Translaticn Tables ccnsist of entries that qive the transmission codes for incominq and cutqoinq messaqes. The Translation Tables are found throuqh a three-byte address in the DCB for the line qroup.



Fiqure 10. Relationshir amcnq the LCB, SCT, DCB, and Translation Tables

PESOURCE CONTRCL ELOCK

Each element in the TCAM system is represented by a resource control block (RCB). An PCB is actually a two-word prefix to an element. The first word is a pcinter to the QCB that the element is to be associated with; the second word is a link field that, when the element is on a chain, points to the next item on the chain. Fiqure 11 shows the qeneral format of an RCB.

Offset		+1
0	Reserved	QCB Address
+4	Priority	Link Address

OCB address is a pointer to the QCB to which the element has been tposted.

Priority is of the element represented.

Link address is a pointer to the next element in the chain.

Fiqure 11. Rescurce Control Block

There are two types of permanent RCBs:

1. Buffer PCBs

2. Communicaticn line PCBs

Buffers are areas of main storaqe used to contain messaqe data and/or control information. The first 8 bytes of each buffer comprise an RCB. As with all TCAM elements, the identity of a buffer depends solely upon the queue that its representative RCB is chained to at a particular time. The buffer itself is always physically identifiable as a fixed number of bytes of main storaqe. If the RCB representinq the buffer is chained into a Destination QCE, the tuffer is full; that is, it contains a messaqe seqment to be transmitted to a destination. When the same RCB is subsequently chained into the element chain of the Buffer Pequest QCP, the element involved is an available buffer, even thouqh there has been no chanqe in the physical storaqe location of the buffer.

A line ccntrol block (LCB) represents communication lines to the TCAM MCP. There is an LCB for each line in the system. When a subtask has ccntrol of an LCB, it has control of the line; therefore, the LCB itself is treated as the rescurce element. The RCB is contained within the first two words of the LCB.

There are two special types of RCBs:

1. Oueue contrcl block RCBs
2. Flement request block RCBs

When a queue control block (QCB) appears on the ready queue, it may represent a special case in which the QCB is tposted to itself. The QCB is acting as a special element rather than as a system resource, in that the first subtask on the STCB chain of the QCB gains control withcut an element to process. The subtask must be self-contained and able to locate any data it needs for execution. If there are no elements to process, the QCB has gained the system resource, time.

An element request block (ERB) on the ready queue can act as a request for a rescurce or as an actual element itself.

SUBTASK CCNTRCI FIOCK

Subtask control blocks (STCBs) represent the modules that perform the work of the TCAM system. The purpcse of an STCB is to cause a module to be executed. The format cf a full STCB is shown in Figure 12.

Offset		+1
0	Activation Key	
+4	Priority	Link Address

Figure 12. Format of a Full STCB

When the TCAM Dispatcher examines the QCB associated with the element on the top of the ready queue, the third word of the QCB points to the highest-priority STCB on the STCB chain of the QCB. The TCAM Dispatcher uses the activation key of the STCB to determine the type of STCB present. The way of determining the actual address of the subtask varies according to the type of STCB. When the address is available, the TCAM Dispatcher exits tc the routine itself. More details concerning the actual dispatching of a routine are presented under System Ccntrol in the Method cf Operation section of this publication.

The four types of STCBs are discussed under Functions of the TCAM Dispatcher in the Method of Operation secticn of this publication.

QUEUE CONTROL BLOCK

A queue control block (QCB) is used to regulate the sequential use of elements among requesting tasks. Every queue, or item, that is waiting for service in the system is associated with a QCB. Figure 13 gives the general format of a QCB.

Offset		+1
0	Key	Element Chain Pointer
+4	Priority	Link Address
+8		STCB Chain Pointer

Figure 13. General Format of a QCB

A QCB has three primary fields: a pointer to the element chain, a link address, and a pointer to the STCB chain. The element chain consists of any elements, other than the requesting resource on the ready queue, that the subtask represented by the STCB chain might need to process. If this is the Buffer Request QCB, the element chain consists of buffers (actually the buffer unit pool). The link field is used to point to another item when a QCB is on a higher queue. For example, if a QCB is on the ready queue, the link field points to the next item on the ready queue. The STCB chain consists of pointers to the routines that are associated with the QCB.

For each attached task (Operator Control, On-Line Test, Checkpoint, and FE Common Write) there is a special QCB that has an ECB in the second word. The TCAM Dispatcher posts the ECB when the attached task is to vie for control of the system. An element that is to be passed to the attached task is chained into the QCB element chain.

There is a detailed list of the QCBs in the TCAM system in Appendix B.

LINE CONTROL BLOCK

There is one line control block (LCB) for each line in the TCAM system. An LCB contains all the information pertaining to the status of the communications line that it represents. The format of an LCB is given in the Data Area Layouts section of this publication.

STATION CONTROL BLOCK

There is at least one station control block (SCB) associated with each LCB in the TCAM system. With buffered terminals there is one SCB per terminal on a line. A buffered terminal sends a block or a part of an entire transmission at a time. While that terminal is preparing to send a subsequent block, TCAM examines the SCBs and sends to and receives from other terminals on the same line. TCAM uses the SCB for a terminal to keep track of one transmission from that buffered terminal on the line.

If the terminals on a line are not buffered, one terminal at a time completes its transmission. There is no need to keep track of many transmissions in parallel, thus one SCB is sufficient for the entire line.

CHANNEL PROGRAM BLOCK

A channel program block (CPB) contains a disk I/O channel program that contains a pointer to the buffer to be processed. In disk queuing, CPBs are used to read to or write from the destination queues. If disk queuing is utilized, the pool of CPBs is created by a nonresident routine called by the INTRO macro expansion. The user specifies the number of CPBs to be built to handle the message queues buffers in the CPB=integer operand of the INTRO macro. Each CPB is built in main storage and is allocated a work area equal in size to one buffer unit (including the 12-byte unit control area).

ELEMENT REQUEST BLOCK

TCAM uses an element request block (ERB) to request buffers for a line. There is one ERB in each LCB. An ERB is posted to the appropriate QCB to obtain filled buffers for a send operation or empty buffers for a receive operation. The format of an ERB is shown in Figure 14.

Offset		+1
0	Key	QCB Pointer
+4	Priority	Link Address
+8	Status	Chain Pointer
+12	Count 1	Count 2

Figure 14. Format of an ERB

The QCB pointer refers to the queue control block to which the ERB is tposted. The link address points to the next element on the queue that contains the ERB. The status field indicates the status of the ERB (for example, tposted for a buffer, available, etc.). The chain field contains a pointer to the first buffer in a chain of buffers to be used in the operation. If the buffer unit pool is empty (all buffer units are in use), the ERB is placed in a chain of ERBs waiting for buffers and remains there until a buffer is returned and assigned to it. The two count fields indicate the number of buffers requested for an operation. Two fields are needed because a disabled routine may need to increment the count and an enabled routine to decrement the count.

PROCESS CONTROL BLOCK

A process control block (PCB) is a control area in an MCP that provides an interface between the MCP and an application program. The PCB contains information needed for communication between the two programs.

A PCB macro instruction in the MCP defines a PCB. There must be one PCB, hence one PCB macro, for each active application program to be used with the MCP.

OPERATOR CONTROL ADDRESS VECTOR TABLE

The Operator Control Address Vector Table is a constant area assembled at the beginning of the Resident Operator Control module. This table is used by the Resident Operator Control module, by the operator control processing modules, and by the checkpoint/restart modules.

SELECTED OPTIONS

TCAM has certain optional features available. These features are optional in one of three possible ways:

1. Some of the functions of the feature are optional.

2. The presence or absence of the feature itself is optional.

3. The feature may be either resident or transient.

The following sections discuss each of the optional featues of TCAM.

OPERATOR CONTROL

The TCAM Operator Control facility provides a way for the user to dynamically examine or alter the status of his telecommunications network. A detailed description of the functions of this facility is included in the Operator Control Facility section of the System/360 OS TCAM Programmer's Guide, Order No. GC30-2024.

The TCAM user specifies at SYSGEN time whether he wants the Operator Control facility in his system to be supported by resident or transient routines. The control module of the Operator Control facility is always resident. If the user indicates that he wants the operator control support routines to be transient, these routines are called in whenever they are needed. If the routines are specified to be resident, they are all present in the system at all times.

APPLICATION PROGRAM PROCESSING

The application program services of TCAM enable a programmer to process messages from a telecommunications network with the same macro instructions that he uses for local input/output devices. Because the TCAM MCP performs the I/O operations, a completely device-independent application program can be written. The programmer need not be concerned with the time and device-dependent aspects of the telecommunications environment.

A TCAM MCP can operate in the System/360 without an application program or programs. However, if the user wishes to examine and process the data coming in from his terminals to a greater extent than is allowed by the macro instructions of the MCP, he must use one or more application programs. The macros specific to application programs are discussed in detail in the System/360 OS TCAM Programmer's Guide, Order No. GC30-2024.

LINE QUEUING OPTIONS

The TCAM user has the option of queuing either by line or by terminal, as specified in the TERMINAL macro for each terminal or group of terminals. The only exceptions are in the cases of buffered terminals and of dial lines, where queuing by terminal is required. Since queuing by terminal requires one Destination QCB per terminal rather than one per line group, this method requires more main storage space.

MESSAGE QUEUING OPTIONS

There are three types of queuing for messages:

- Main storage queuing

- Reusable disk queuing

- Nonreusable disk queuing

The message queues may be maintained by any one of the three methods or by a combination of main storage queuing with backup on either reusable or nonreusable disk.

In an MCP there are at most two message queues data sets: reusable disk with or without main storage queues, and nonreusable disk with or without main storage queues. The user specifies the type of queuing for a given data set by coding specified keyword operands of the macros that build the Terminal Table. The way in which the types of queuing are specified is discussed in detail in the System/360 OS TCAM Programmer's Guide, Order No. GC30-2024. The way that the various queuing types function is discussed under Queue Management in the Method of Operation section of this publication.

LOGGING

The logging option allows the user to maintain a record of incoming or outgoing message traffic on a sequential medium. Message segments or full messages, as determined by the placement of LOG macros in an MH, are placed on an output device. The various types of logs, and the corresponding MH subgroups in which a LOG macro appears, are:

1. Incoming header segments only (Inheader)

2. All incoming segments (Inbuffer)

3. Complete incoming messages (Inmessage)

4. Outgoing header segments only (Outheader)

5. All outgoing segments (Outbuffer)

6. Complete outgoing messages (Outmessage)

When segments of messages are logged separately, they are logged in the sequence in which they are handled by the message handlers. Segments of different multi-segment messages handled about the same time are likely to be intermixed on the logging medium. When the first segment of a message is logged, the TCAM header prefix (except the first twelve bytes) and the segment itself are recorded in that order on the logging device. Each subsequent message segment logged is preceded by all except the first twelve bytes of the TCAM subsequent-buffer prefix for that segment.

CHECKPOINT/RESTART

Checkpoint/Restart is provided as an optional facility for the TCAM MCP at user-specified intervals (every 30 seconds to 65,535 seconds). By using the TCAM Checkpoint/Restart facility for the MCP and other TCAM facilities, such as sequence numbers, an effective restart can be accomplished in an application program.

The checkpoint routines store tables and other control information necessary for a restart subsequent to a system failure or normal closedown. Restart of the TCAM job after a system failure is accomplished by initial program loading (IPL) the system again, and loading the TCAM MCP. TCAM reinitializes the tables and pointers from the latest checkpoint record on the disk, unless "CY" is specified on the STARTUP parameter of the INTRO macro to suppress continuation start-up. After a system failure, the STARTUP=C or STARTUP=W operand on the INTRO macro causes TCAM to perform a continuation restart with a scan of the message queues. If STARTUP=WY is specified, a continuation restart with no message queues scan is performed.

After a normal closedown, TCAM can either reconstruct the environment that existed before closedown (a warm restart) or it can reinitialize the system (a cold restart). A warm restart is specified by STARTUP=W on INTRO; a cold restart is specified by STARTUP=C.

To include the Checkpoint/Restart facility in an MCP, the user has only to specify an OPEN for the checkpoint data set. As a result of this, the Checkpoint Executor is attached in the same region as the MCP. The other checkpoint modules can be either resident or transient, dependent on what the user specifies at SYSGEN time.

TCAM AS A STARTABLE PROCEDURE

The user has the option of starting a TCAM MCP or application program either via JCL in the system input device or via the START operator command at the system console. If the START command is to be used, the JCL for the MCP and the different TCAM problem programs must be cataloged on SYS1.PROCLIB under individual procedure names. The user may then type START and the "procname" for the program he wants, and job management immediately fetches the JCL at "procname" and subsequently starts the program.

ERROR RECOVERY PROCEDURES

The Error Recovery Procedure (ERP) routines are designed to diagnose and recover, if possible, from line errors occurring during a telecommunications operation. The error routines provide the following basic functions:

- Automatic retry of all errors not involving data transfer. Data transfer is handled by the EOB/ETB Handling subtask.

- Automatic retry of text errors during a receive operation when the data is still available; that is, the PCI Appendage has not tposted the buffers containing the data following the last good EOB/ETB.

- Statistical recording of all terminal errors.

- Error messages to the primary TCAM operator console for all permanent errors.

The ERP routines are optional in that they may be either resident or transient. The user specifies this option at SYSGEN time.

SUBTASK TRACE

The Subtask Trace facility maintains a time-sequential table of the dispatching activity of the TCAM Dispatcher. Each time the Dispatcher activates a subtask, it completes an entry in the Subtask Trace Table.

The presence of the Subtask Trace facility in the TCAM system is determined by the DTRACE operand of the INTRO macro in the MCP. If the operand is coded DTRACE=0, the facility is not included. If the operand is coded with a numerical value, that value determines the number of four-word entries reserved for the Subtask Trace Table.

CROSS REFERENCE TABLE

The TCAM Cross Reference Table is formatted if the CROSSRF=integer operand of the INTRO macro is assembled with a nonzero value. The numerical value of integer determines the number of four-word entries reserved for this table. Each time that a line is successfully opened, the Line Group Open routine (IGGC1940) completes an entry in the table.

TCAM IN A MULTIPROCESSING ENVIRONMENT

TCAM operating in a multiprocessing environment increases throughput, availability, and flexibility. All TCAM appendages and SVC 102 cause the TCAM task to become not eligible to be dispatched in order to prevent TCAM disabled code from modifying TCAM control blocks while enabled TCAM code is executing. These modules set a flag in the TCAM TCB to indicate that the task is not eligible to be dispatched and then call the OS Task Removal routine. When the Task Removal routine issues an external interrupt to the other CPU, the other CPU loops on the supervisor lock. When the TCAM module completes its functions, it resets the TCB flag and zeros the supervisor lock before exiting. The other CPU then obtains the lock and dispatches the task of the highest priority on its ready queue.

To prevent two enabled tasks from attempting to enqueue/dequeue on the same resource at the same time, each task issues a test-and-set instruction on a specific byte in the QCB before referring to the queue. The byte must be equal to zero before the task can update the queue, and the task must reset the byte to zero after completing the update.

TIME SHARING OPTION

TCAM provides terminal support for the Time Sharing Option (TSO) under MVT when this option is requested on the INTRO macro. There are special macros to generate an MCP with MH routines to handle TSO messages. TCAM also supports application programs that are run under TSO in the foreground region. If the TSO option is specified, TCAM provides a conversational approach to terminal support - this includes support of the transmit and receive interrupt features, modifications to the scheduling of I/O operations, and editing of the data in TSO messages to make the data compatible with disk or tape.

TCAM and the TSO control program run in different partitions. Smaller buffer prefixes and a modified message flow allow TCAM to route the messages to the TSO region.

TCAM support for TSO also includes the ability to use 1050s and 2741s on the same dial line, the ability to simulate receive interrupts when they are not a feature of the hardware, and the ability to have the transmission code dynamically determined.

MODULE ATTRIBUTES

TCAM modules are designed to possess certain defined attributes concerning structure, content, and logical format. These attributes determine how a module is to be loaded, what it contains, if it is executable, whether it is executable more than once without reloading, and if it can be executed by concurrent tasks.

The attributes are included in the description of each module in the Program Organization section of this publication. The attributes applicable to TCAM modules are:

- Reentrant. A reentrant module can be executed by more than one task concurrently and cannot be modified by itself or by any other module during execution; that is, a task may begin executing a reentrant module before the previous task has finished executing it.

- Refreshable. A refreshable module cannot be modified by itself or by any other module during execution; that is, a refreshable module can be replaced by a new copy during execution by a recovery management routine without changing either the sequence or the results of processing. (See *IBM System/360 OS Concepts and Facilities*, Order No. GC28-6536, for an explanation of recovery management routines.)

- *Serially Reusable*. A serially reusable module can be executed by only one task at a time. The module either initializes itself and/or it restores any instructions or any data in the module that was altered during its execution.

- *Enabled*. An enabled module can be interrupted at any time by an appendage or external event. When the interruption occurs, the enabled module waits for the appendage to complete its processing and then continues as though the interruption had never occurred. The interruption has no effect on the execution of the enabled module.

- *Disabled*. A disabled module cannot be interrupted during its execution. It must execute from beginning to end once it has gained control.

- *Resident*. A resident module resides in main storage of the TCAM system at all times.

- *Transient*. A transient module is a nonresident module that resides in a system library on some type of storage device until it is called into the TCAM system for a limited length of time during the execution of a problem program.

- *Problem Program Mode*. A module that operates in problem program mode is operating under control of the message control or application program, rather than under the control of the OS supervisor.

- *Supervisor Mode*. A module that is operating in supervisor mode is operating under the control of the system supervisor.

This section contains an introduction to the logic of TCAM. The flow of messages and control information through the buffers and tables and the detailed functional descriptions of the modules are emphasized.

LOGIC OF TCAM

TCAM can be functionally divided into four major phases:

- Disk message queue initialization
- Initialization of an MCP
- Message handling in an MCP
- Closedown of an MCP

In addition to the four phases listed above, there are other phases that are functionally independent, yet necessary to complete a discussion of the logic of TCAM:

- System control
- Application program processing
- Operator control processing
- Checkpoint processing
- Error recovery procedures
- Time Sharing Option interface

This section of the TCAM PLM presents the above phases of the program in the order as they would logically occur in a TCAM system. Since the application program, Operator Control, Checkpoint, error recovery, and Time Sharing phases have no clear place in this time-frame organization, they are presented as separate discussions at the end of the section.

The foldout operation diagrams associated with this section illustrate the functional operation of TCAM. The foldout diagrams are located after Appendix D and are accompanied by a description in the text of this section.

THE DISK MESSAGE QUEUE INITIALIZER

The Disk Message Queue Initializer is a utility routine that is used to pre-format the data sets for the disk message queues for a TCAM MCP. This routine is run before executing the TCAM MCP job.

A TCAM MCP can use either reusable or nonreusable disk message queues data sets, or both. If both are used, each one must reside on a separate data set. The Disk Message Queue Initializer must be executed for each data set.

The variables used to define the disk data set are entered as Job Control Language (JCL) parameters. In the JCL the user defines the size of each extent, the number of extents, the volumes to contain the data set, the type of disk used, and the size of each fixed-length record. In this data set there is one extent per volume, and all the records on a volume must be contiguous. There is no difference in the creation of a reusable or nonreusable data set.

The data set formatted by the Disk Message Queue Initializer is fixed length and physically sequential. Each record has a key and data field initialized to zero. The size of the data field is fixed at six bytes; the size of the key field is specified by the user in the "KEYLEN=mm" parameter of the IEDQDATA job control statement. The key field must be less than or equal to 255 bytes, but greater than or equal to 33 bytes (three plus the size of the prefix of the first buffer). At the end of writing each extent of records, the Disk Message Queue Initializer lists on the system console typewriter a statement that contains the total record count from the beginning of the data set through the volume just completed. If an error condition is encountered in writing the records, the initializer sends an error message to the system console and then terminates.

INITIALIZATION OF A MESSAGE CONTROL PROGRAM

Upon receiving control from System/360 OS Job Management, the TCAM MCP performs certain initialization functions in preparation for subsequent processing. The initial processing operations include:

- Allowing the user to alter the contents of certain AVT data fields that were initialized and formatted at assembly time.
- Initializing and allocating storage for buffers, tables, control blocks, and work areas.
- Sorting the Termname Table.
- Opening data sets, initializing LCBs, and modifying DCBs.
- Preparing the communications lines for transmission.
- Attaching any required tasks in the TCAM partition.

The INTRO, OPEN, and READY macro expansion instructions in the MCP initiate the initialization functions cf TCAM. Foldout Charts 1 through 5 show the flow of control during the initialization of an

FUNCTIONS OF INTRO

The INTRO macrc is the first instruction ccded by the user in a TCAM MCP. When OS Job Management is alerted to the presence of a TCAM MCP in the system, the INTRO macrc expansicn, as a subroutine of job management, is called to execute its specific initialization functions. Foldout Chart 1 presents a summary of these functions.

FUNCTIONS OF THE OPEN RCUTINES

After the initialization functions of the INTRO macro expansion have been completed, the functions of any OS or TCAM macros can be executed. However, in an MCP, the user must open various data sets before he can begin processing any data. He must open these data sets in a certain crder: first, the message queues data sets (optional), then the checkpcint data set (optional), and then the data sets for the line groups.

Foldout Charts 2 through 5 illustrate the flow cf control during the opening of each of these data sets, respectively.

The following general points apply to each of the figures:

1. When an OPEN macro is issued in an MCP, the OS Cpen routine gains control. It, in turn, issues an XCTL command to bring in the first load of the apprcpriate open module. The first routine, upon completion of its functions, issues an XCTL to the appropriate subsequent routine.

2. When any given routine is to load a module, it activates OS, which checks the OS Contents Directory to determine whether that module has already been loaded. If there is an entry for the module in the directcry, OS adds one to the directory usage count. If there is no entry in the directory, OS makes a two-byte entry in the directory, adds one to the usage count, and loads the module.

3. If the user issues an OPEN macro for multiple message queues or line group data sets, each individual routine performs its functions fcr each data set before issuing an XCTL to the next routine. However, if there is a separate OPEN for each data set, each routine is loaded individually fcr each data set.

PREPARATION OF COMMUNICATIONS LINES FOR TRANSMISSION

The initial channel programs to enable the lines in the TCAM network are built by the Line Group Open routines. The content of each channel program depends on the type of control unit used with the devices on the line.

Control Unit	Line	Channel Program
2702	Leased	DISABLE, SAD, ENABLE
2702	Dial	DISABLE, SAD
2701 or 2703	Leased BSC	DISABLE, SETMODE, ENABLE
2701 or 2703	Dial BSC	DISABLE, SETMODE
2701 or 2703	Dial	DISABLE
2701 with IBM Type III Adapter		DISABLE
2701 or 2703	Leased	DISABLE, ENABLE
7770	Dial	NOP
2848	Leased	NOP

FUNCTIONS OF READY

The READY macro instruction must be the last instruction in the initialization section of an MCP. After the functions of READY have been executed, the system is ready to handle message traffic. The expansion of this macro ends with an instruction to branch to the routine (the TCAM Dispatcher) in the MCP where arrival of the first element on the ready queue is awaited. When the first message enters the system, control is transferred to the MH section of the MCP.

When the user codes a READY macro in his MCP, he has the option of specifying the addresses of routines to handle "Good Morning" and "Restart in Progress" messages. The assembly of the READY macro places these addresses in the AVT.

Foldout Chart 5 presents a functional flow for the READY macro expansion and routine.

SYSTEM CONTROL

Two primary routines maintain control among the parts of TCAM:

- The TCAM Dispatcher

- The AQCTL SVC 102 routine

FUNCTIONS OF THE TCAM DISPATCHER

The TCAM Dispatcher is the control module of the TCAM system. The primary purpose of this module is to allocate and schedule system resources. The section on the TCAM Dispatcher in the Introduction to this publication contains a discussion of the tools and mechanisms used by the Dispatcher to perform its functions.

Each queue in the TCAM system is represented by a queue control block (QCB), which is the connecting link between elements and the subtasks waiting for the elements. A QCB consists of a pointer to a chain of elements and a pointer to a chain of STCBs. Elements and STCBs are inserted in their respective chains on the QCB in priority-FIFO order, that is, first-in-first-out within priority class.

A subtask control block (STCB) represents each waiting subtask to the Dispatcher. An STCP contains the data necessary to activate the subtask it represents. A full STCB consists of a subtask entry code or activation key (MCPL), a priority field, and a link field for STCB chaining. (There is a complete discussion of the four formats of STCBs later in this section.)

A resource control block (RCB) represents each element to the Dispatcher. An RCB contains three fields: the address of the QCB to which the element is or is to be tposted, a priority field, and a link field to be used for element chaining. When elements are on the ready queue, they are maintained in priority-FIFO order. The TCAM Dispatcher activates a subtask for the element on the top of the ready queue. The RCB for the element points to a QCB, and the activated subtask is represented by the highest priority STCB on the STCB chain of the QCB.

Figure 15 illustrates the chain of linkage from the ready queue to a subtask when an element is on the ready queue.

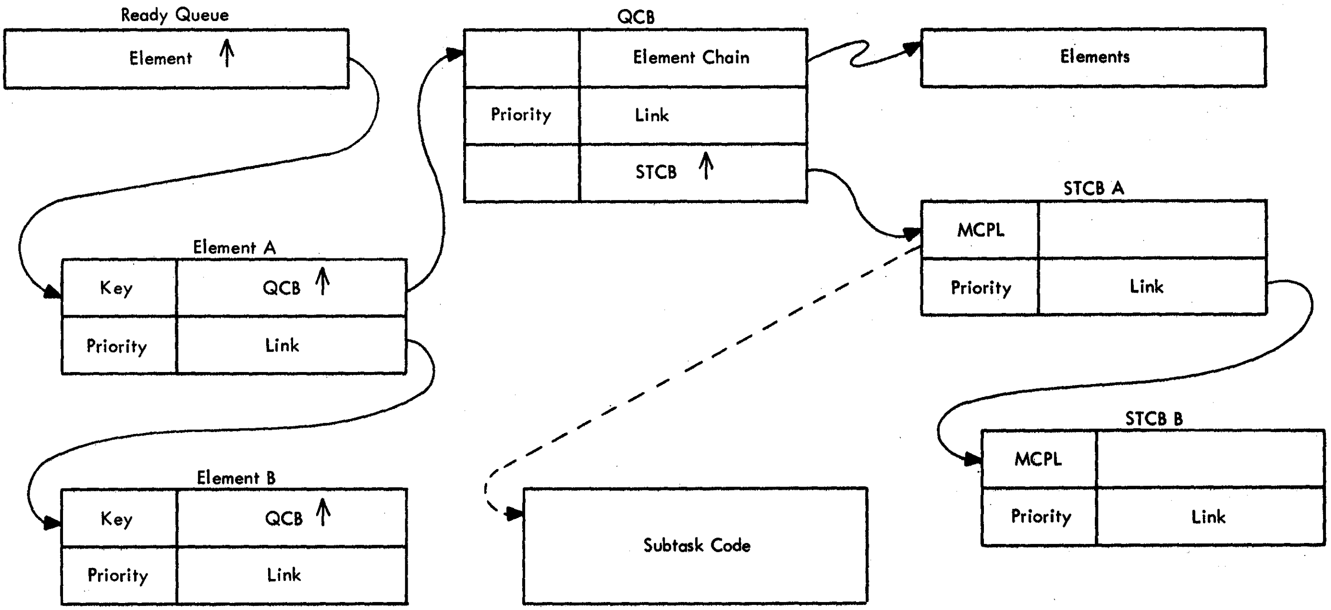


Figure 15. Linkage from the Ready Queue to Subtask Code

When the Dispatcher examines the highest priority element on the ready queue, it removes that element from the ready queue by placing the address of the element in register 1. The Dispatcher then inserts the link field of the element in the ready queue, so that the next element can be examined. When there are no elements for the ready queue, it pcints to the "dummy last element" in the AVT (AVTDELEM). This element has a priority of zero. Figure 16 demonstrates the change in linkage between the ready queue and its elements during an update of the ready queue by the Dispatcher.

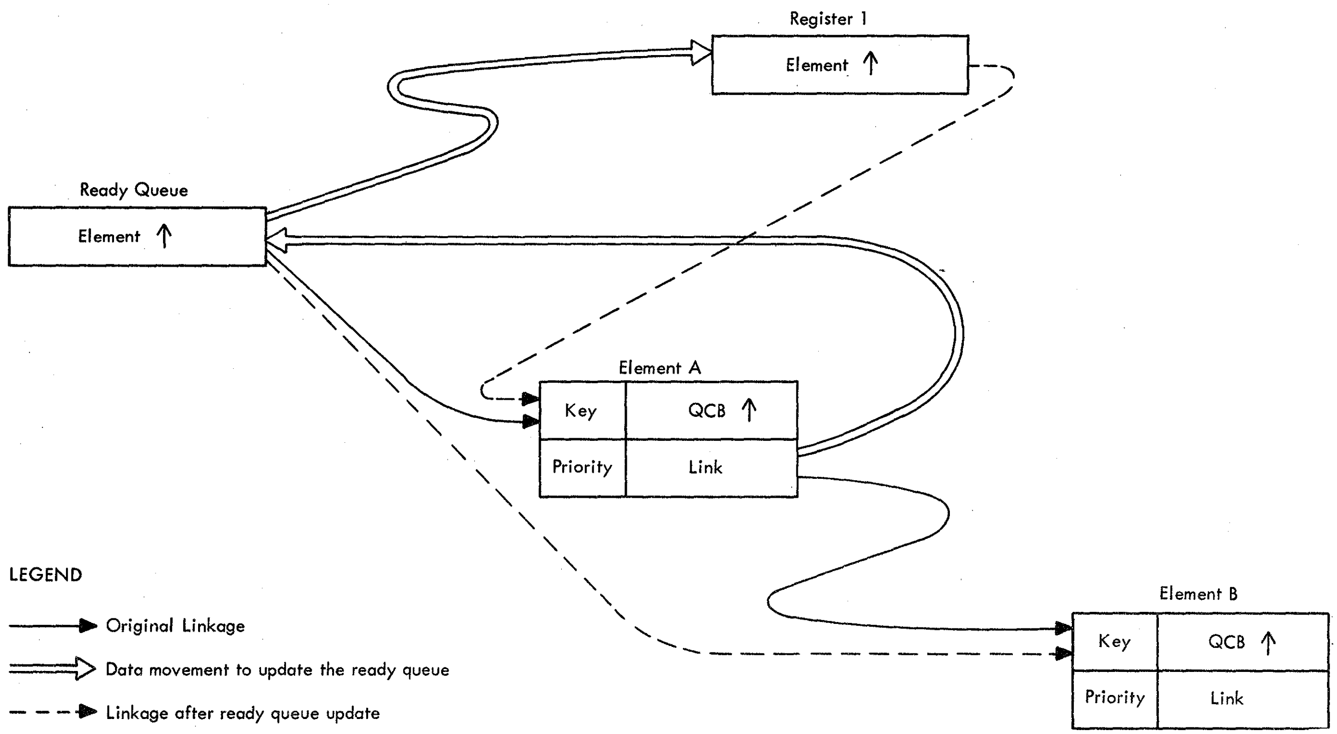


Figure 16. Pointers during a Ready Queue Update

After the ready queue has been updated, the TCAM Dispatcher examines the element pointed to by register 1. There are three situations that can exist:

- If the element pcints to a QCB that has an STCB with a MCPL field of zero, the element indicates to the Dispatcher that there are no real elements currently tposted to the ready queue. This is a "dummy" element that causes the Dispatcher to issue a system WAIT command. The activity of the Dispatcher resumes when an I/O routine or an application program tposts an element tc the ready queue and causes an interruption in the operating system.

- If the element is tposted to a QCB that represents an attached TCAM task (Operator Control, Checkpoint, On-line Test, or FE Common Write), the MCPL field of the STCB is equal to X'02'. This causes the Dispatcher to link the element to the element chain of the QCB and to post complete the event control block (ECB, the second word of the QCB) of the attached task. This allows the attached task the opportunity to vie for control of the system when TCAM issues a system WAIT command.

- If neither of the above situations exists, the Dispatcher computes the entry point for the highest priority subtask represented on the STCB chain of the QCB referred to by the RCB of the element. The Dispatcher then branches to that subtask.

  The TCAM Dispatcher calculates the subtask entry point according to the value of the MCPL field in the STCB. If the MCPL field is equal to X'04', the subtask entry point immediately follows a two-byte STCB. If the MCPL value is X'06', the subtask entry point immediately follows a four-byte STCB; and an MCPL value of X'08' indicates a six-byte STCB. An MCPL value of X'0A' indicates a subtask entry point immediately following an eight-byte STCB. If the MCPL value is greater than X'0A', the TCAM Dispatcher activates the associated subtask by using the MCPL field as an index into the AVT branch table at AVTDISP. The following values of MCPL cause the Dispatcher to activate the associated subtasks:

X'0C'	-	Leased Receive Scheduler
X'0E'	-	Send Scheduler
X'10'	-	Get Scheduler
X'12'	-	Put Scheduler
X'14'	-	Get FIFO Scheduler
X'16'	-	Log Scheduler
X'18'	-	Dial Receive Scheduler
X'1A'	-	Buffered Terminal Scheduler
X'1C'	-	Retrieve Scheduler
X'1E'	-	Local Receive Scheduler

Figure 17 shows the linkage from register 1 to the highest priority STCB when the Dispatcher is examining an element.

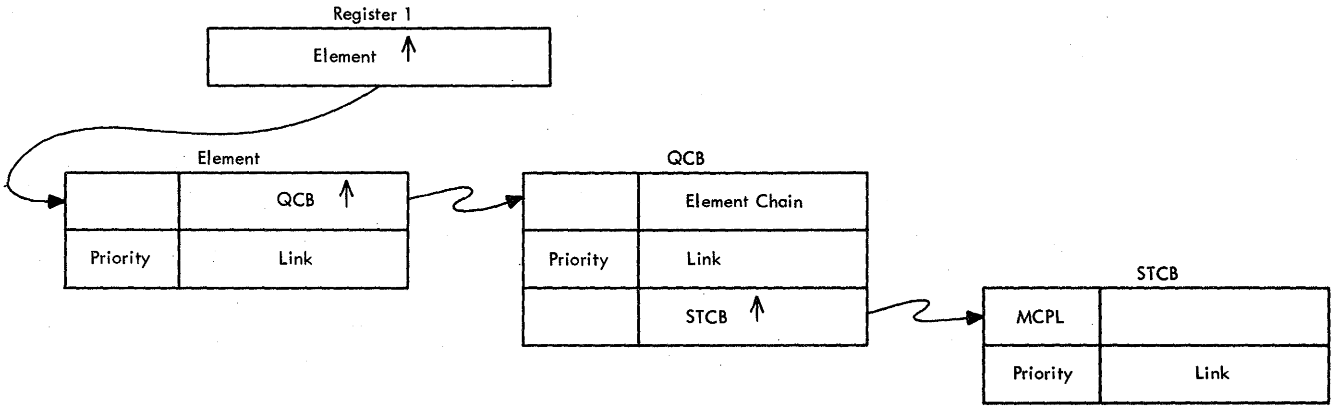


Figure 17. Linkage from Register 1 when a Subtask Gains Control

Note: If a subtask is activated without an element to process, its STCB is tposted to the ready queue, as if it was an RCB, with the MCPL field containing the correct entry code for the subtask and the next three bytes containing the address of AVTREADY-8.

There are four possible formats for STCBs. The way the subtask entry point is calculated depends on the type of STCB, and the MCPL field indicates the type. Each type of STCB has a different length.

A two-byte STCB is used when its QCB is located in the AVT or elsewhere in main storage, the STCB is the only one that ever appears in the STCB chain of the QCB, and the STCB is never placed in the STCB chain of any other QCB. The Dispatcher examines the QCB to find the STCB pointer. The MCPL field of the STCB contains the value X'04', and the Dispatcher adds 2 bytes to the address of the STCB to find the subtask entry point. The second byte of the STCB is unused. The format of a two-byte STCB is shown in Figure 18.

A four-byte STCB has an MCPL value of X'06' and is used when it is convenient to have the QCB as a part of the subtask code. The QCB and STCB are combined by making the STCB the third word of the QCB. The STCB must be the only one for this QCB, and the STCB must never be transferred to the STCB chain of another QCB. The Dispatcher calculates the subtask entry point by adding four bytes to the STCB address. The format of a four-byte STCB is shown in Figure 18.

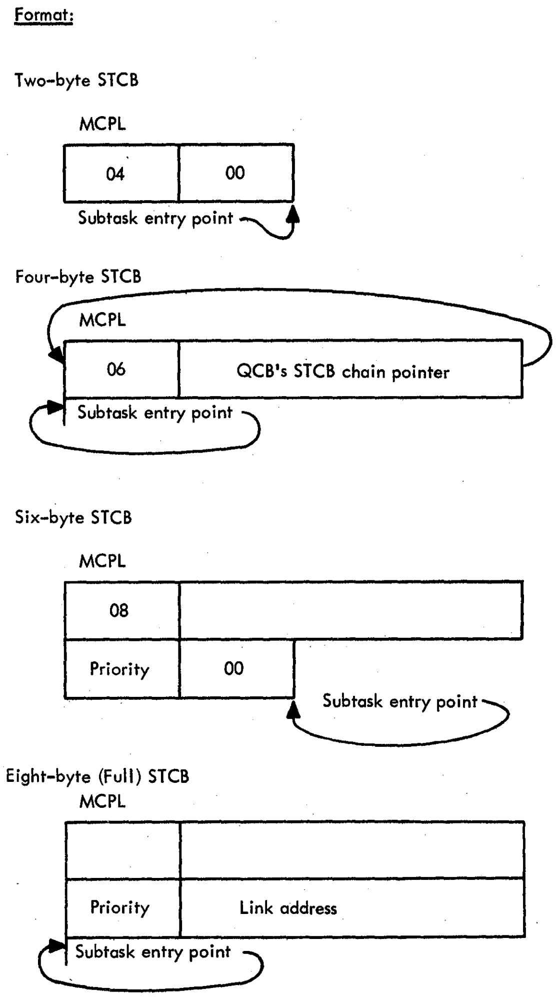
A six-byte STCB has an MCPL value of X'08' and is used when an STCB always appears as the last STCB in the STCB chain of a QCB. In this situation, the priority field, but not the link field, is needed. The Dispatcher calculates the subtask entry point by adding six bytes to the STCB address. The format of a six-byte STCB is shown in Figure 18.

An eight-byte STCB is used when an STCB can appear in any position in the STCB chain of a QCB. When the MCPL field is X'0A', the Dispatcher calculates the subtask entry point by adding eight bytes to the STCB address. If the MCPL value is greater than X'0A', the STCB is for one of the TCAM schedulers, for each of which the Dispatcher uses the MCPL field as an offset into the AVTDISP table of addresses. The format of an eight-byte STCB, which is a full STCB, is shown in Figure 18.

Foldout Chart 6 presents a summary of the dispatching functions of the TCAM Dispatcher.

The TCAM Dispatcher also functions as a queue manager. The Dispatcher performs queue management functions when a subtask branches to a particular entry point in an entry point table in the Dispatcher. The function performed by the Dispatcher depends on which label a subtask branches to.

Entry point labels that do not end in "R" result in loss of control by the branching subtask. Entry point labels that end in "R" result in an immediate return of control to the branching subtask after the queue management function has been performed.



Attributes:

- QCB located in the AVT or assembled in main storage
- QCB has only one STCB
- STCB is never chained to any other QCB

- QCB is part of the subtask code
- QCB and STCB are combined – the STCB is the third word of the QCB
- QCB has only one STCB
- STCB is never chained to any other QCB

- STCB is always the last STCB in the STCB chain of a QCB

- STCB can appear in any position of the STCB chain of a QCB

Fiqure 18. Formats for Different Tyces of STCBs

The queue manaqement functions cf the varicus entry point labels are described in foldout Chart 7.

There are actually two TCAM Dispatchers available for a TCAM MCP. They are the same except that IGG019RO performs the additional function of buildinq a Subtask Trace Table. If the DTRACE keyword of the INTRO macrc is coded with a ncnzero numerical value, the IGG019RO TCAM Dispatcher is loaded into the MCP. Ctherwise, the IGG019RB TCAM Dispatcher is used.

FUNCTIONS OF THE AQCTL SVC 102 ROUTINE

The AOCTL SVC 102 routine is a multipurpose system service routine that performs the followinq functions:

- Cross-partition data movement tetween the MCP and application proqrams.

- Postinq ECBs for attached tasks and applicaticn proqrams.

- Tpostinq elements frcm attached tasks and application proqrams to the disabled ready queue in the MCP.

- Flaqqinq the Task Control Block (TCB) that represents a Time Sharinq Opticn (TSO) application proqram as either available or not available for swap.

- Flaqqinq the TCB that represents an application program as either eliqible cr nct eliqible fcr rcllcut.

The AQCTL SVC 102 routine is a Type I SVC resident in the Operatinq System nucleus. It qains control when an SVC 102 call is issued from any task in the system.

When the AQCTL SVC 102 routine is called by a routine anywhere in the system, reqister 1 must point to a variable lenqth standard parameter list. The AQCTL SVC 102 routine examines the first byte (byte 0) of this list to determine which of the possible functions is to be performed. The contents of the parameter list vary accordinq to the action code settinq in byte 0.

If more than one bit in the acticn code byte is turned on, the AQCTL SVC 102 routine performs the acticns specified for each bit. The combinaticns of bits used, however, must be compatible so that the parameter list satisfies all the requirements.

When the AQCTL SVC 102 routine relinquishes ccntrol, it stores a return code in reqister 15. For a successful operaticn, the return code is binary zero. If the SVC is issued when there is nct an active MCP in the system, the requested acticn is not perfcrmed and the return code is binary four.

The followinq paraqraphs discuss the method used by calling routines to effect the functions cf the AQCTL SVC 102 routine.

Cross-partition Data Movement: When a rcutine needs to move data across a partition boundary, it turns on action code bit 4 in byte 0 of the parameter list beinq built for the AQCTL SVC 102 rcutine.

To effect cross-partition data movement, the callinq routine provides a three-word parameter list. The first word contains the address of the data to be moved. The seccnd word contains the address of the place the data is to be moved to (the target field), and the third word points tc a halfword that contains the lenqth in bytes of the data field. Fiqure 19 defines this particular parameter list format.

The AQCTL SVC 102 routine, upon findinq bit 4 cf byte 0 set to 1, moves the data to the specified lccation.

Offset		+1
0	Action code	ECB address
+4	X'00'	TSO Job Identifier address
+8	X'80'	TCB address

Figure 19. Format of a Cross-Partition Data Movement Parameter List

Post an ECB of a Different Task: When a routine needs to issue an OS POST on the ECB of another task, either bit 1 or bit 2 of the action code byte is set to 1. Bit 1 is turned on if the ECB of a task that is eligible for rollout (RORI) is to be posted complete; bit 2 is turned on if the ECB of a standard (ECB always in main storage) or Time Sharing Option (TSO) task is to be posted complete.

Depending on the type of ECB to be posted, the parameter list built for the AQCTL SVC 102 routine is either two or three words long. The parameter list for a standard or TSO task is two words long; the parameter list for a RORI task is three words long. The formats of these two parameter lists are shown in Figure 20.

TSO or Standard Task:

Offset		+1
0	X'20'	ECB address
+4	X'80'	TSO Job Identifier address

Rollout/Rollin Task:

Offset		+1
0	X'40'	ECB address
+4	X'00'	TCB address
+8	X'80'	DEB address

Figure 20. Formats of an ECB Post Parameter List

To effect an ECB post, the AQCTL SVC 102 routine interfaces with the OS Post routine (IEAQSY50) at a special entry point (IEAOPT01) that performs nc validity checking. The address of this entry point is in the CVT. The AQCTL SVC 102 routine supplies input to the OS Post routine in the following general registers:

- Register 15 - the address of the branch entry, IEAOPT01.
- Register 14 - the return address.
- Register 13 - in the low-order 16 bits, the TSC Jcb Identifier for the ECB to be posted (for standard ECBs, this field is binary zeros).
- Register 11 - the ECB address, with the lcw-order bit set to one.
- Register 10 - the completion code (always zero).

If the task to be posted is currently rolled out, the AQCTL SVC 102 routine sets a bit in the TCB (TCBFLTRN) to designate to the Rollout/Rollin routine at rollin time that there is a POST pending for this task.

If the ECB to be posted is for a TSO task, the AQCTL SVC 102 routine branches to the Time Sharing Interface program in the nucleus task to be flagged either eligible or not eligible for swap. The interface is accomplished via the TSEVENT macro.

Tpost An Element to the Disabled Ready Queue: When a routine needs to tpost an element to the disabled ready queue in the MCP, bit 5 of the acticn code byte is set to 1.

The calling routine builds the same format three-word parameter list used for cross-partition data movement (see Figure 19). The address of the target field, in this case, is the address of the disabled ready queue in the AVT. There is no actual data movement, because both the data field and the target field are elements - only the pointers are changed.

The AQCTL SVC 102 routine chains the element onto the disabled ready queue and posts the ECB for the MCP complete.

Flag the TCB for a TSO Program: When a routine needs to flag the TCB of a TSO application program either eligible or not eligible for swap, bit 3 or bit 6 of the acticn code byte is used.

If bit 3 is equal to one, the AQCTL SVC 102 routine flags the TCB of the TSO program not eligible for swap; if bit 6 is equal to one, the TCB of the program is flagged eligible for swap.

The three-word parameter list created by the calling routine is illustrated in Figure 21.

Offset		+1
0	Action code	ECB address
+4	X'00'	TSO Job Identifier address
+8	X'80'	TCB address

Figure 21. Format of a Parameter List to Flag the TCB of a TSO Program

Flag the TCB for a RORI Program: When a routine needs to flag the TCB of an RORI application program as eligible or not eligible for rollout, bit 0 or bit 7 of the action code byte is used.

If bit 0 is equal to one, the AQCTL SVC 102 routine flags the TCB of the task as not eligible for rollout; if bit 7 is equal to one, the TCB is flagged as eligible for rollout.

The three-word parameter list created by the calling routine has the same format as the parameter list for posting the ECB of an RORI task (see Figure 20).

MESSAGE HANDLING IN A MESSAGE CONTROL PROGRAM

Data enters the TCAM system randomly in the form of messages from remote terminals or programs that generate messages. Data is ultimately delivered to one or more terminals or programs that process the data. The MCP controls the routing of the messages as well as a limited amount of processing. These functions of an MCP are referred to as "message handling" functions.

In order to present an overview of the way an MCP performs its message handling (MH) functions, this section contains discussions of the functional areas involved:

- Line management

- Buffer management

- Message handling routines

- Queue management

Foldout Chart 8 illustrates message flow through a TCAM system. Note the area of influence for each of the functional parts to be discussed.

LINE MANAGEMENT

TCAM schedules line operations to allow data to travel over a line in a single direction at any one point in time. A line can be used for both sending and receiving, and in order to schedule this two-way activity TCAM uses two mechanisms. The first of these, a receive scheduler, allows data to be received from a remote station; the other, a send scheduler, allows data to be sent to a remote station.

Each line in a TCAM system is represented by an LCB, and at line open time, each LCB (except a send-only line) has a receive scheduler STCB built in it. This STCB can be for the Leased Receive Scheduler, the Dial Receive Scheduler, the Local Receive Scheduler, or the Buffered Terminal Scheduler, depending on the characteristics of the line.

At assembly time each Destination QCB in a TCAM system has an STCB starting in its third word. This STCB can represent either the Send Scheduler or the Buffered Terminal Scheduler and is used to schedule sending operations.

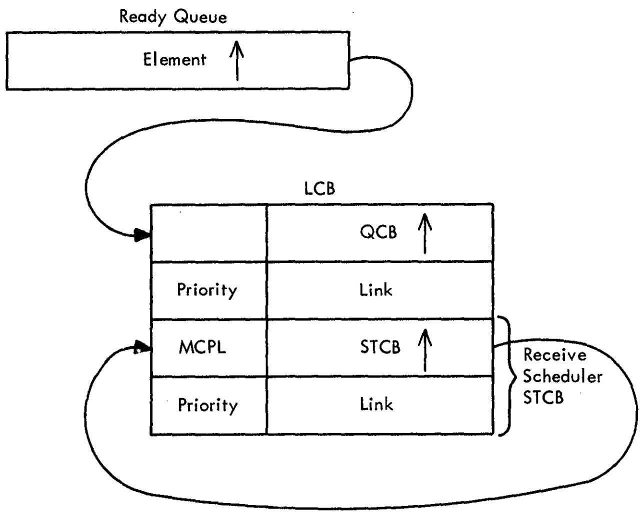
The priorities of the receive and send scheduler STCBs are determined when the user specifies whether he wants receive, equal, or send priority for a line. As the address of an STCB is moved from the STCB chain of the LCB to the STCB chain of the Destination QCB and back, that STCB is inserted in the respective STCB chains by FIFO-priority.

An LCB is tposted to the ready queue when the line that it represents is free to either receive or send data. The STCB that has the highest priority in the STCB chain of the LCB has its subtask dispatched.

A Receive Operation

At open time, either a Leased Receive Scheduler STCB, a Local Receive Scheduler STCB, a Buffered Terminal Scheduler STCB, or a Dial Receive Scheduler STCB is built in each LCB. Since there is one LCB for each line in the system, there is also one receive scheduler STCB for each line that can receive data in the system. If a line is intended for sending only, there is no receive scheduler STCB and the only STCB in the STCB chain for the line points to the QEVENT routine, which frees the LCB instead of attempting to initiate a receive operation. (The QEVENT routine is part of the Receive Scheduler CSECT.)

The receive scheduler in control inspects a line to determine whether a receive operation is possible. A message can enter the TCAM system only after the receive scheduler for a line has recognized that the line is available so that a receive operation can be started. The scheduler is activated by the Dispatcher when its STCB is the next STCB in the STCB chain of an LCB at the top of the ready queue. (See Figure 22.)



Fiqure 22. A Peceive Scheduler STCB in an ICB on the Ready Queue

The primary function of each cf the receive schedulers is to solicit data from the terminals on a line. For contention lines this is done by preparinq the line to receive; fcr multipcint lines this is done by pollinq the terminals on the line. The point at which a receive scheduler releases a line is qenerally when the end of an invitation list is reached, althouqh it can be after receipt of a messaqe or, in the absence of the Auto Poll feature, when a neqative response is received.

Each receive scheduler, in order to solicit a messaqe, requests buffers to ccntain the messaqe by tposting the ERB for the LCB to the Buffer Request QCB. The Buffer Request routine (IEDQGA) gets the requested number of buffers from the buffer unit pool, and chains the units from the chain field of the ERB. Buffer Request branches to Buffer Association, which builds in the buffers a channel program that is appropriate to the characteristics of the line. Buffer Association returns to Buffer Request, which tposts the ERB tc the Activate QCB. As a result, the TCAM Dispatcher dispatches the Activate-I/C Generator subtask (IEDQKA, IEDQKB, IEDQKC, IEDQKD, cr IEDQKE).

The Activate-I/O Generator subtask builds the initial ccntrol CCW sequence. This subtask then issues an EXCP to accept a message and relinquishes ccntrol to IOS.

After IOS has accepted the EXCP request, the subsequent I/O interrupt with device ending status causes the TCAM Line End Appendaqe to qain control. If there is a messaqe ready to be processed, Line End Appendaqe tposts the buffers to the STARTMH QCB for message handlinq. If there is no messaqe available, Line End Appendaqe tposts the LCB to Buffer Disposition where the buffers are returned to the buffer unit pcol and the line is freed (the LCB tposted to itself and placed on the ready queue).

Foldout Chart 9 illustrates the general flow cf control during a receive operaticn. Foldout Chart 16 shows how a receive scheduler operates in a ccmplete receive operation.

The specific functions of each of the receive schedulers are described in the Program Organization section of this publication.

A Send Operaticn

There is a send scheduler STCB assembled in every Destination QCB in TCAM. (If the Destination QCB is for an application program, the Get Scheduler STCB assembled for it is the equivalent of sending to an application program.) The purpose of a send scheduler is to attempt to find a line for sending when a message is tposted to a Destination QCB and to initiate sending of the messages on the QCB. The line is initialized for sending when the send scheduler is dispatched as a subtask of the LCB.

A send scheduler is activated by the Dispatcher when its STCB has top priority in the STCB chain of a Destinaticn QCB or an LCB. This send scheduler can be either the Send Scheduler or the Buffered Terminal Scheduler.

The number of send schedulers that can contend for a line is determined by the type cf queuing requested in the TERMINAL macro for the line. If queuing by line is specified, one send scheduler STCB is generated for the line. However, if queuing by terminal or by component is specified, there is one send scheduler STCB for each terminal. The relative priority of the send schedulers is established at assembly time by the CPRI operand of the line group DCB.

A send scheduler, in order to prepare to read a message from a message queues data set and to direct the message to the appropriate terminal, tposts the ERB in the LCB to the Disk I/O QCB. The Disk I/O QCB has the CPB Initialization STCB in its STCB chain. When the tposted ERB gets to the top of the ready queue, the TCAM Dispatcher activates CPB Initialization. This routine starts reading a message for the line and gets enough full buffers to satisfy the ERB request. CPB Initialization chains the buffers off the ERB and tposts the ERB to the Activate QCB. As a result, the TCAM Dispatcher dispatches the Activate-I/O Generator subtask.

The Activate-I/O Generator subtask builds the selection CCW sequence and a send channel program that is appropriate to the characteristics of the device to receive the message. This module then issues an EXCP to address the terminal and relinquishes control to IOS.

After IOS has addressed the terminal and received a response to addressing, the resulting I/O interrupt activates the TCAM Line End Appendage. The Line End Appendage examines the response to addressing; and if the response is positive, the appendage tposts the buffers to the STARTMH QCB for outgoing message handling and restarts

I/O on the Write Idles loop. For terminals that do not have a selection sequence (cannot be addressed by TCAM), the Activate-I/O Generator subtask tposts the outgoing buffers directly to MH. If reserve (idle) characters exist for the device, Line End Appendage restarts the channel program on the Write Idles loop; otherwise Buffer Association (IEDQGD) issues the EXCP command. If the response to addressing is negative, the appendage tposts a buffer with an error indicator to MH in order to route control to the outmessage subgroup for user consideration via optional OUTMSG macros. Also, if the negative response to addressing is due to a hardware error, Line End Appendage activates the error recovery procedure.

The general flow of control during a send operation is illustrated in foldout Chart 10. Foldout Chart 17 shows how a send scheduler operates in a complete send operation.

The specific functions of the Send Scheduler and of the Buffered Terminal Scheduler are discussed in the Program Organization section of this publication.

BUFFER MANAGEMENT

The TCAM network has one buffer unit pool that contains buffer units of one size. These buffer units are the basic building blocks from which *logical buffers* are constructed. Henceforth, in this publication *unit* refers to a buffer unit and *buffer* refers to a logical buffer.

Messages entering a TCAM network are placed in buffers, which are user-defined areas of main storage used for handling, queuing, and transferring message segments between all lines and queuing media. (A message segment is that portion of a message contained in one buffer.) A buffer has two parts, one that contains control information (the *buffer prefix*) and the other that contains all or part of the message. Buffers must be at least 33 bytes long, and may be no longer than 65,535 bytes.

The size of a unit is specified in the KEYLEN operand of the INTRO macro of an MCP, and the number of units in the buffer unit pool is equal to the sum of the numbers specified by the LNUNITS and MSUNITS operands of INTRO. For internal management purposes, TCAM adds 12 bytes as a prefix to the user-specified unit size. These 12 bytes are called a *unit control area*. Thus, if a user defines a unit size of 60 bytes (KEYLEN=60), the size of the unit is actually 72 bytes.

The size of a buffer for a line group is specified by the BUFSIZE operand of the DCB macro for a line group data set. All buffers used by a given line group are the same size, but each line group may utilize buffers that differ in size from those assigned to other line groups. (The buffer size can be overridden on a terminal basis for send operations by using the BUFSIZE operand of the TERMINAL macro.)

TCAM constructs buffers by linking together the number of units necessary to create a buffer that contains a number of usable bytes equal to or greater than that specified by the BUFSIZE operand of the DCB macro for a given line group. (The 12 bytes added to each unit by TCAM are not considered in defining BUFSIZE; the user should consider only the number of bytes he specified in the KEYLEN operand of INTRO). For example, if KEYLEN=60 in the INTRO macro and BUFSIZE=120 in a line group DCB macro are specified, TCAM links together two units in building each buffer for that line group.

There are two types of buffers - header buffers and text buffers. A *header buffer* contains all or any part of a message header. A *text buffer* contains message text only.

A *buffer prefix* is a control area contained within each buffer of the system. The user must allow room for the buffer prefix in defining his buffers. TCAM fills the buffer prefix area with buffer control information.

There are two kinds of buffer prefix. The *first buffer prefix* is 30 bytes long and is contained within the first buffer of a message. Any *subsequent buffer prefix* is 23 bytes long and is contained within all buffers after the first.

Thus, there are two kinds of control areas associated with buffers: the twelve-byte unit control area associated with each *buffer unit* and assigned automatically by TCAM, and the 30-byte or 23-byte buffer prefix assigned to each *buffer* by TCAM in an area allowed for by the user. Each unit must be big enough to contain a header prefix plus three bytes of message text (33 bytes) and may be no larger than 255 bytes. A subsequent buffer contains more bytes of actual message than the first buffer, since a subsequent buffer prefix is 7 bytes shorter than the first buffer prefix.

The twelve-byte unit control area that TCAM assigns to each unit is used to manage multi-unit buffers. This control area has different functions dependent on the status of its buffer - it may contain pointers, be used as an RCB, or be used to generate a channel program. The initial format of this 12-byte area is defined in Figure 23.

Offset

0	1	4	8
Key	QCB address	Address of the first unit of the next logical buffer that is assigned	Address of the next unit of this buffer

Figure 23. Unit Control Area

Figure 24 shows how two buffers assigned to a line group look on an initial request if the user specifies the following:

```
INTRO     KEYLEN=60
DCB       BUFSIZE=100,BUFIN=2
```

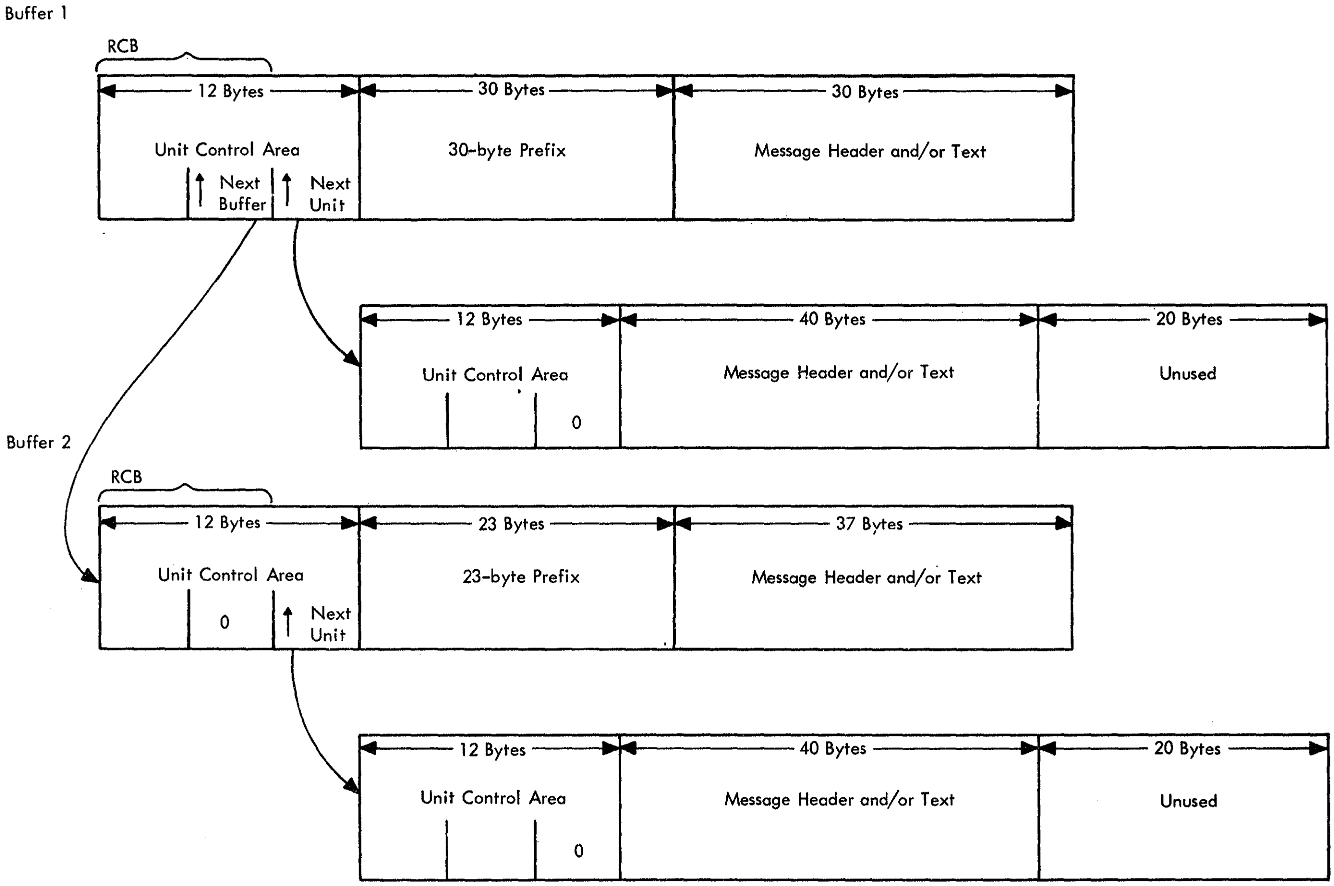


Figure 24. Buffer Units Chained to Form Logical Buffers.

In Figure 24, each buffer consists of two units linked together by the pointer in the third word of the twelve-byte unit control area. The two buffers are linked together by the second word of the twelve-byte unit control area. Note that in this situation the first eight bytes of the unit control area of the first unit in each buffer is functioning as an RCB.

When the user's program requests and obtains buffers, they look like the ones in Figure 24. However, when a line is ready to read or write, the function of the twelve-byte control area changes. TCAM then uses the area to contain the channel program that operates on the unit. The Buffer Association routine places a CCW in each RCB field, and the pointer in the third word becomes a TIC to the next unit. The 30-byte prefix contains a count of the number of units in a logical buffer; this indicates where one buffer stops and another starts.

To tpost a buffer, TCAM places only the first unit of that buffer on the ready queue. All other units can be located through the chain created in the TIC field of the unit control area.

Buffer Requesting and Allocating

TCAM uses an element request block (ERB) to make requests for buffers for a line group. A description of the physical characteristics of an ERB is included under Control Areas in the Introduction section of this publication.

Initial requests for buffers for a line are made when a scheduler tposts its ERB, which contains the number of buffers requested, to the Buffer Request QCB for a receive operation, or to the Disk I/O QCB for a send operation.

Subsequent requests for buffers are handled by the TCAM Program-Controlled Interruption (PCI) Appendage. When the PCI operand of the DCB for a line group is coded to allow program-controlled interruption, a PCI may occur during the filling or emptying of the first and each subsequent buffer assigned to that line group. When the PCI is received, the PCI Appendage gains control.

When PCI=A is coded on the DCB macro and the first interruption occurs, PCI Appendage assigns to the line group a number of buffers equal to the difference between the maximum number assigned to the line group (specified by the BUFMAX operand of the DCB) and the number initially assigned to the line group (specified by the BUFIN operand of the line group DCB for a receiving operation and by the BUFOUT operand for a sending operation). On subsequent PCIs, the appendage deallocates the buffer immediately preceding the one being filled or emptied and requests a new buffer in order to keep the number of buffers assigned to the line group equal to that specified by BUFMAX. (For a sending operation, the buffer units are returned via the Buffer Return QCB to the buffer unit pool - the element chain of the Buffer Request QCB; for a receiving operation, the buffer is sent to the message handler for the line group for that DCB.)

When PCI=R is coded, the appendage deallocates the previous buffer when the second and subsequent PCIs occur, but makes no requests for additional buffers. If program-controlled interruptions are not permitted (PCI=N) or additional allocation is not allowed (PCI=R), the number of buffers assigned must be sufficient to handle the entire transmission, since no new buffers are allocated until the transmission is complete. If PCI=N, there is no deallocation of buffers until the transmission is complete.

- Initial Request - Receive Operation

When a line group in the TCAM system needs a buffer or buffers for a receive operation, a receive scheduler must tpost an ERB that contains the number of buffers requested to the Buffer Request QCB. Foldout Chart 11 shows the complete flow of control for an initial buffer request in a receive operation.

Figure 25 shows the result of an ERB with a count of three being tposted to the Buffer Request QCB. The ERB chain of the LCB points to the first buffer. This figure demonstrates the change in linkage after units have been transferred from the buffer unit pool to form a buffer chain off the requesting ERB. The physical location of the units in main storage does not change - the various pointers are changed to reflect the new organization.

Figure 26 shows the contents of the buffers after Buffer Association has been executed.

If the initial request for buffers cannot be satisfied, the ERB is chained by priority into the element chain of the Buffer Return QCB. This ERB has a high priority; therefore, as soon as the buffers are available, the initial request is satisfied and the line can start receiving messages.

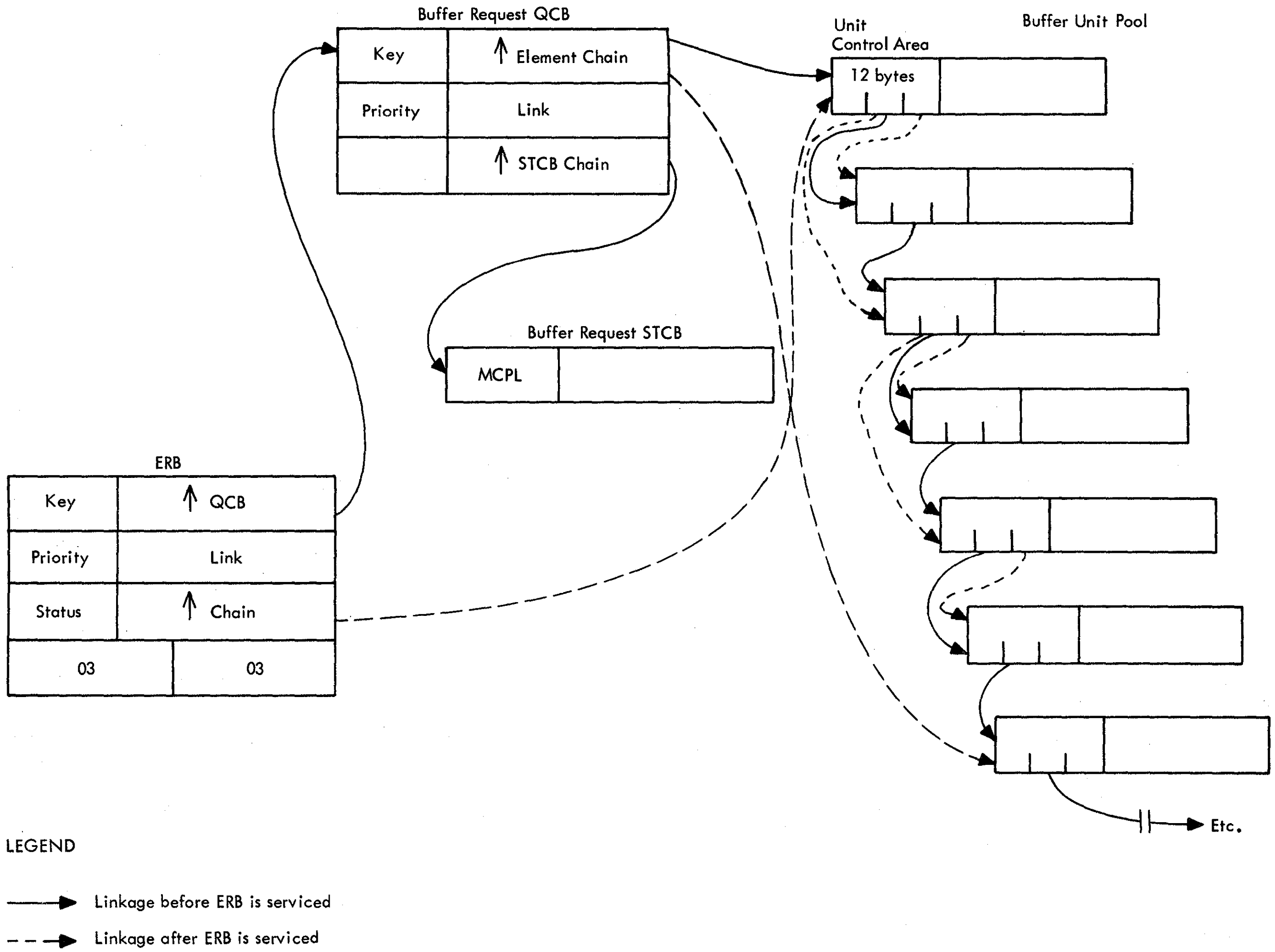


Figure 25. Effect of an ERB on Buffer Unit Linkage

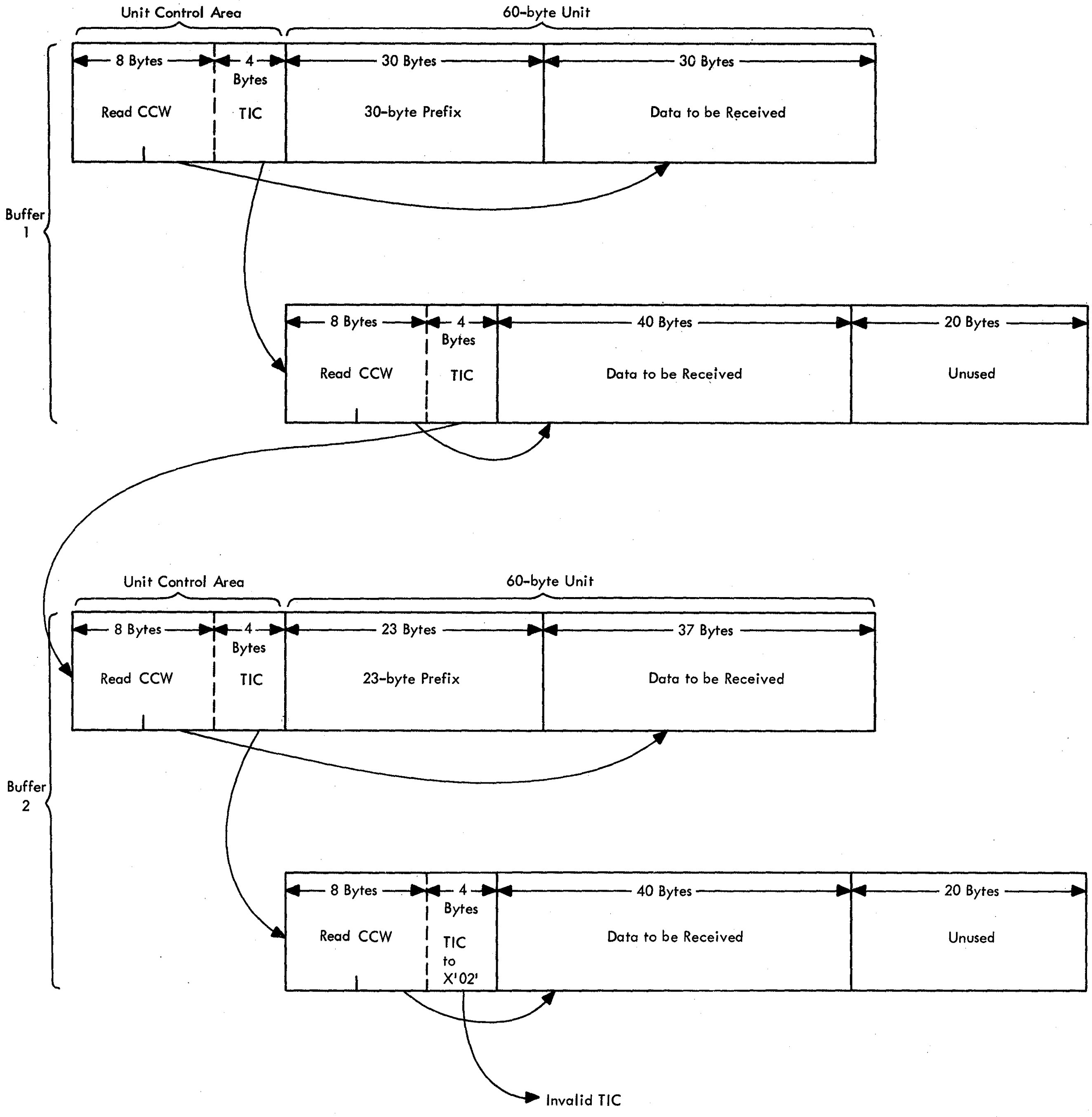


Figure 26. Buffers Prepared to Receive Data

- Initial Request - Send Operation

When a line grcup in the TCAM system needs a buffer or buffers for a send operation, the Send Scheduler must tpost an ERB for the number of buffers requested initially to the Disk I/O QCB. The CPB Initialization STCB resides in the STCB chain of the Disk I/O QCB. When the tposted ERB gets to the top of the ready queue, the CPB Initialization routine gains control.

For a send operation, when the CPB Initializaticn routine acquires enough buffers to fill the ERB request, the buffers are already full and allocated to a line. However, they have to go through message handling (MH) before I/O can occur. CPB Initialization tposts the ERB, with its full buffers, to the Activate QCB. This activates the Activate subtask (IEDOKA), which builds initial contact CCWs and issues EXCP. At this pcint, allocation cf the buffers is complete. Upcn completion of the addressing sequence, Line End Appendage tposts the buffers tc MH for outgoing prccessing.

Foldout Chart 12 shows the flow cf control for initial buffer request and allccation during a send operation.

- Subsequent Requests - Receive Operaticn

As discussed earlier in this section, all subsequent requests for buffers are handled when PCIs occur. When a PCI for a receive operation occurs, an ERB for additicnal buffers is tposted to the Buffer Request QCB and the buffers are assigned.

- Subsequent Requests - Send Operation

When a PCI for a send operaticn occurs, an ERB is tposted to the Disk I/O QCB and the buffers are allocated.

Functions of Buffer Association

The Buffer Association routine in the Buffer Management module builds the CCWs in the units of buffers. To do this, the routine builds a CCW in the first buffer to be read or written, fills the subsequent buffers with CCWs, and places an invalid TIC in the last unit. As other buffers are assigned, the invalid TIC is changed to TIC to a new buffer and the invalid TIC is placed in the last unit of the new buffer. If a channel program check occurs cn the invalid TIC, the channel program check portion of Line End Appendage causes the channel to execute the Write Idles/Read Skip lcop. When a buffer becomes available, Buffer Association links the buffer into this loop, as well as into the buffer chain. (See Figure 26.)

Buffer Association is called at different times during receive and send operations:

- Receive operation - Buffer Request calls Buffer Association to handle all initial buffers just prior to going to Activate to build the initial contact channel program. When subsequent buffers are obtained, the Buffer Return subtask calls Buffer Association as soon as a buffer is available.

- Send operation - as soon as MH has processed each buffer, it calls Buffer Association.

Buffer Association exits to the TCAM Dispatcher with the CCWs completed.

Deallocating Buffers

Buffers are deallocated, released from use by a line group, in different ways for receive and send operations.

Receive Operation: When a PCI or the Line End Appendage takes a buffer from a channel program and sends it to MH, the buffer is deallocated from the channel program, but it is not free. The buffer is completely deallocated only after it has been queued.

Send Operation: When a buffer has been sent out to a line group, it is deallocated by virtue of being tposted to the Buffer Return QCB.

Functions of Buffer Return

When a buffer is tposted to the Buffer Return QCB, the action taken depends on whether there is an ERB waiting for that buffer. Foldout Chart 13 shows the conditions under which the Buffer Return routine gains control and the functions that the routine performs.

MESSAGE HANDLING ROUTINES

In TCAM, a message is a sequence of characters entered at or sent to a terminal, and terminated by an ending character (EOT, ETB, ETX, or EOB). A message may consist of two portions, a header portion and a text portion, each of which may occupy more than one buffer. A message may have a header only, text only, or both.

The discussion of Buffer Management earlier in this section describes header buffers, text buffers, a 30-byte buffer prefix, and a 23-byte buffer prefix. It is necessary to understand these terms before approaching the subject of message handling.

Before message characters are placed in the first buffer, TCAM reserves the number of reserve characters specified by the user for the line group. TCAM reserves space for these characters at byte 30 in the first buffer and at byte 23 in each subsequent buffer. These reserve characters save room in the buffer for later insertion of the date, time, and sequence number for the message. As messages enter the CPU and are placed in buffers, characters start filling each buffer just after the reserved space.

As soon as a buffer is filled with the first segment of a message, the appendage in control tposts that buffer to the QCB for the message handler (MH) designated for the particular line group that the message is for or from. (The appendage is able tc designate the proper MH by examining DCBMH in the DCB of the line group.) The tposting of the buffer chains it onto the disabled ready queue. When the TCAM Dispatcher gains control, the disabled ready queue is merged by FIFO-priority order onto the enabled ready queue, and the buffer waits its turn to be dispatched to its MH.

A message handler is a set of message handling routines designed to process messages for a particular line group or for several line groups with similar characteristics. Each MH is identified by a STARTMH macro and may consist of an incoming group and an outgoing group, which are designed to handle incoming and cutgoing messages respectively. The functions of these groups and their subgroups are discussed in the following sections. Foldout Chart 14 illustrates the progress of a buffer through an MH.

Functions of the User Interface Routine

At assembly time many of the user-coded MH macros generate one or more fixed-length parameter lists, some executable code, and branch instructions to the User Interface routine (IEDQUI). At execution time the User Interface routine uses the parameter list frcm a macro to gain access to the specific functional routine needed for processing. After it has finished executing, the functional routine branches to the Return Interface routine (IEDQLM), which, in turn, returns to the next sequential instruction in the MH portion of the MCP. The next instruction might be a branch back to the User Interface routine with a new parameter list to be processed. This process of branching to functional routines thrcugh the User Interface routine continues until the functions of all the user-ccded macros for the specific MH have been executed.

Functions of STARTMH

A STARTMH macro identifies the beginning of an MH and must be the first instructicn coded in every MH. When a buffer is tpcsted to the STARTMH QCB of an MH and no block checking is specified, the functions of the STARTMH subtask are performed. When a buffer is tposted to the STARTMH QCB and block checking is specified, the EOB/ETB Handling subtask is activated. The EOB/ETB Handling subtask checks for the occurrence of hardware errors during message transmission and can handle user-detected logical errors. After EOB/ETB Handling has processed the buffer (or if it has no processing tc perfcrm), it uses the bypass functicn of the Dispatcher to activate the STARTMH subtask.

The block labeled STARTMH in foldout Chart 14 summarizes the specific functicns of the STARTMH subtask.

Note: For a ncn-TSO TCAM system, the STARTMH subtask is IEDQAA; when TSO is in the system, the IEDAYR version of the subtask is used.

Functions of the Incoming Group of a Message Handler

The incoming grouP of an MH handles messages arriving from a station with which the MH is associated. When a buffer containing a message segment is passed to the incoming group cf an MH, user-specified functions such as source checking, insertion of the time the message was received, input sequence-number checking, etc., are performed. The MH scans and processes buffer header fields in accordance with the order indicated by the relative positicns of the individual MH macro instructions.

The incoming group has three possible types of subgroups:

- The inheader subgroup, which handles only incoming header segments,
- The inbuffer subgroup, which handles all incoming message segments, and
- The inmessage subgroup, which is executed after a complete message has entered the CPU.

Functions of an Inheader Subgroup: The first macro coded in an inheader subgrouP is the INHDR macro. The first function of INHDR macro-generated code is to determine whether the buffer to be processed is a header buffer or a recalled buffer. If it is not a header buffer cr if it is a recalled buffer, control is transferred to the next delimiter macro expansion.

If the buffer to be processed is a header buffer and a PATH operand was coded for the INHDR macro, the Locate Option Field Address routine is given ccntrol to find the address of the cpticn field. Upon return to the macro-generated code, a test determines whether an option field address was found. If it was not, control passes to the next delimiter macro expansion. If there is an option field address, but there are no matching path switches, control is also transferred to the next delimiter macro expansion. Otherwise, control falls through to the next sequential MH instruction.

After the INHDR macro-generated code is executed, the expansions of the other user-coded macrcs prccess the buffer. There are two levels of processing used at this time: functional routines and functional subroutines.

A functional routine is associated with a specific MH macro. When the macro is coded, the assembler generates either the necessary parameter list(s) and a branch instructicn to the User Interface routine or a branch to the associated routine, if one is needed. At execution time, the User Interface routine branches to the functional routine, as described previcusly. The functional routine uses the assembly-generated parameter list to gain access to the control areas and data needed fcr processing the buffer. The functional routine returns to the Return Interface routine, and from there to the next sequential MH instruction.

A functional subroutine gains control from either the User Interface routine or directly from a functional routine. The same functional subroutine can be used by any number of functional routines. A functional subroutine returns to the functional routine that called it.

Some TCAM MH routines function as both functional routines and subroutines.

Functions of an Inbuffer Subgroup: The first macro coded in an inbuffer subgroup is the INBUF macro. The first functions of INBUF macro-generated code are to perform the same multiple-buffer-header and PATH operand tests that are performed by INHDR macro-generated code. The results of the tests are the same as described above.

Processing of the buffer continues through this inbuffer subgroup according to the MH macros specified by the user. Functional routines and subroutines actually perform the processing, as described in the Functions of an Inheader Subgroup section.

Functions of an Inmessage Subgroup: The macro instructions coded in an inmessage subgroup are executed only after a complete message has entered the TCAM system.

The first macro coded in an inmessage subgroup is the INMSG macro. The INMSG macro-generated code tests for the PATH operand and executes accordingly, as described in the Functions of an Inheader Subgroup section.

When an inmessage subgroup maintains control (the path switch setting matches), control is passed, via the User Interface routine, to the Incoming/Outgoing Message Delimiter routine. If the buffer is the last buffer of a message, it is tposted to the Buffer Disposition QCB; if it is not the last buffer, it is tposted to the appropriate Destination QCB.

When the last buffer of a message is tposted to the Buffer Disposition QCB, the TCAM Dispatcher activates the Buffer Disposition subtask to supervise execution of the macros in the subgroup. The message handling functions of the Buffer Disposition subtask are illustrated on foldout Chart 14.

Functions of the Outgoing Group of a Message Handler

The outgoing group of an MH handles messages as they are prepared for sending to the destination with which the MH is associated. As the message is brought in from its queue (e.g., in a message queues data set), it is placed in buffers, as for an incoming message.

When a buffer that contains a message segment is passed to the outgoing group of an MH, that group processes the buffer according to the functions specified by the user-coded MH macros.

The outgoing group has three possible types of subgroups:

- The outheader subgroup , which handles only outgoing header segments,

- The outbuffer subgroup, which handles all outgoing message segments, and

- The outmessage subgroup, which is executed after a complete message has been sent.

Functions of an Outheader Subgroup: The first macro coded in an outheader subgroup is the OUTHDR macro. The functions of the OUTHDR macro-generated code are the same as for the INHDR macro-generated code.

Functions of an Outbuffer Subgroup: The first macro coded in an outbuffer subgroup is the OUTBUF macro. The functions of the OUTBUF macro-generated code are the same as for the INBUF macro-generated code.

Functions of an Outmessage Subgroup: The MH macros in an outmessage subgroup are executed after an entire message has been sent.

The first macro coded in an outmessage subgroup is the OUTMSG macro. The OUTMSG macro-generated code tests for the PATH operand and executes accordingly, as described in the Functions of an Inheader Subgroup section.

When an outmessage subgroup maintains control (the path switch setting matches), the macro expansion passes control, via the User Interface routine, to either the Incoming/Outgoing Message Delimiter routine or the Line Control Insertion routine. If the MSGFORM macro is specified in the outgoing subgroup, the Line Control Insertion routine gains control to add the necessary line control characters to the message. This routine then exits to the Incoming/Outgoing Message Delimiter routine.

The Incoming /Outgoing Message Delimiter routine conditionally tposts the buffer to an application program, exits to the Transparent CCW Building routine, or exits to the Buffer Association routine.

The Incoming/Outgoing Message Delimiter routine examines the destination key (PRFDEST) of the buffer prefix and links to the Termname Table Code (IEDQTNT) to obtain the address of the Terminal Table entry for the destination. If the status field indicates a process entry, the routine gets the address of the Read-ahead QCB from the terminal entry and tposts the buffer to that QCB.

If the destination is not a BSC device in transparent mode, the Incoming/Outgoing Message Delimiter rcutine exits to Buffer Association, which builds WRITE CCWs and TICs in the control area of the buffer units. Otherwise, the exit is to the Transparent CCW Building routine for the same purpose.

OUEUE MANAGEMENT

The incoming group of an MH performs user-specified functions in a buffer that contains a message segment. After these functions are completed, the segment is tposted to its Destination QCB. A Destination QCB can represent a line, a terminal, or an application program.

Each Destination QCB in a TCAM MCP is assigned to one or more specific message queues data sets. When a buffer is tposted to its Destination QCB, it is placed on the appropriate message queue in the associated message queues data set to wait its turn to be sent to the specified destination.

The message queues data set to which the message segment is to be directed may be in main storage or on a direct-access storage device. Each message queue within a given data set contains segments that are to be transmitted on a certain line or to a certain terminal, or that are to be processed in a specific application program.

TCAM supports five types of queuing to a message queues data set:

- Nonreusable disk queuing
- Reusable disk queuing
- Main storage queuing
- Main storage queuing with nonreusable disk backup
- Main storage queuing with reusable disk backup

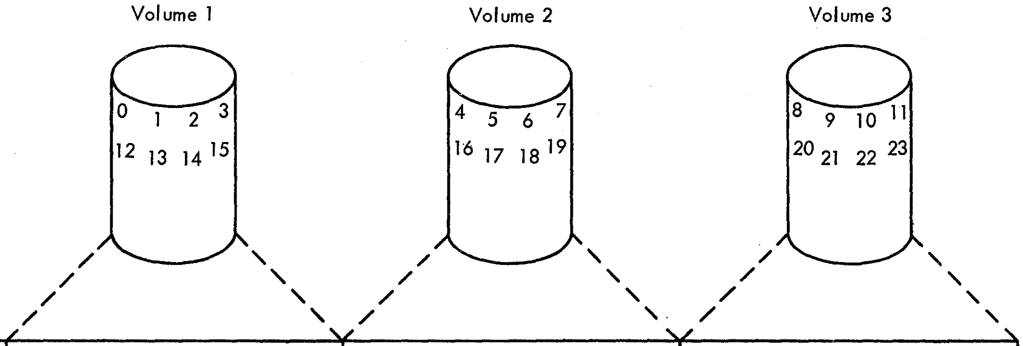
The following sections discuss the functions of these types of queuing.

Nonreusable Disk Queuing

Queuing a message on a direct-access storage device is referred to in this publication as disk queuing. The term address refers to the first disk relative record number that can be used to queue a unit of a message segment. All values of address previous to the current value are either used or preassigned for use. The fields AVTNADDR and AVTRADDR in the AVT contain the address value for nonreusable and reusable disk relative record numbers, respectively. The Destination Assignment routine uses the correct value for the type of queuing specified for a line. In this discussion, address refers to either field.

In nonreusable disk queuing, the Destination Scheduler initiates a closedown when a user-specified percentage of the disk message queues data set has been filled with messages. If, before the closedown can be completed, there are already more messages in the system than the data set has room to accommodate, TCAM issues an ABEND.

The Destination Scheduler assigns disk relative addresses across the volumes of a multi-volume disk message queues data set in such a way that the next relative record address after the last record on a track is on a different volume. The routine numbers all the records for a given track consecutively before assigning address values on a track of a different volume. In addition, the routine numbers all the tracks of a cylinder before assigning address values on a different cylinder. Figure 27 illustrates the disk record numbering scheme for a data set that has four records per track on three volumes.



Cylinder	Track	Relative Record Number				Relative Record Number				Relative Record Number			
0	0	0	1	2	3	4	5	6	7	8	9	10	11
	1	12	13	14	15	16	17	18	19	20	21	22	23
	2	24	25	26	27	28	29	30	31	32	33	34	35
	3	36	37	38	39	40	41	42	43	44	45	46	47
	4	48	49	50	51	52	53	54	55	56	57	58	59
	5	60	61	62	63	64	65	66	67	68	69	70	71
	6	72	73	74	75	76	77	78	79	80	81	82	83
	7	84	85	86	87	88	89	90	91	92	93	94	95
	8	96	97	98	99	100	101	102	103	104	105	106	107
	9	108	109	110	111	112	113	114	115	116	117	118	119
1	0	120	121	122	123	124	125	126	127	128	129	130	131
	1	132...											

Figure 27. Assignment of Disk Message Queues Data Set Relative Record Numbers Across Three Volumes

At MCP assembly or restart time, each Destination QCB is assigned a unique address value for the first buffer segment tposted to it. As a result, when the first message enters the TCAM system, the AVT value of address is one greater than the total number of Destination QCBs.

The Destination Scheduler stores the address value to be used for the first unit of the first buffer of the next message received in the QCBDNHDR field of the Destination QCB - this is referred to as the next-message location. The routine stores the address value for the first unit of the next buffer of the current message in the SCBNTXT field of the SCB - this is referred to as the next-buffer location.

The principle of assigning next-message and next-buffer address values allows queuing ahead on the disk. Records for buffer units are assigned before the buffer is received.

Foldout Chart 15 presents a summary of the nonreusable disk queuing procedure of the Destination Scheduler.

In the example in Figure 28, there are five possible destinations. For each of these, the MCP assembly has preassigned record addresses (marked A through E) with relative record addresses zero to four. The applicable externals for this example are:

```
          INTRO KEYLEN=100

    LINEA DCB   BUFSIZE=300,PCI=(A,A)

    LINEC DCB   BUFSIZE=800,PCI=(A,A)
```

Three messages arrive in the following order:

1. 500 characters - from Line A to Line D

2. 3000 characters - from Line C to Line B

3. 30 characters - from Line A to Line B

Figure 28 shows the situation in which TCAM reads a buffer (the first buffer of the first message) from line A. The 30-byte prefix contains the information that this message is to be sent to line D. The message segment consists of three units (since BUFSIZE=300 and KEYLEN=100) and does not contain an end-of-message (EOM) indicator. The Destination Scheduler assigns the first unit of this header buffer to the preassigned location for destination D, record 3. The scheduler then preassigns the next-message location for destination D to the next available disk location at record 5, and places a pointer to record 5 in the prefix of the buffer that will start in disk record 3. The scheduler then assigns two additional units to the next available disk locations at records 6 and 7. The scheduler inserts a pointer to the first of these records in the prefix of the buffer that will start in disk record 3.

Since the 300-byte buffer does not contain an EOM indicator, the Destination Scheduler preassigns a record number (8) for the first unit of the next buffer to arrive for this message. The scheduler places a pointer to record 8 in the prefix of the buffer that will start in disk record 3. The records are actually written after the three pointers are included in the prefix of record 3. Figure 28 shows the records and pointers after they are written on disk.

In this queuing scheme the additional records are always contiguous, and the first unit of a subsequent buffer of a message is always contiguous to the last unit of the previous buffer.

In Figure 29 the first buffer of the 3000-byte message from line C for line B is queued. The buffer consists of eight units since BUFSIZE for line C is 800 bytes. The Destination Scheduler places the first unit of the message in the preassigned slot for destination line B. The scheduler then preassigns a location for the first unit of the next message for line B to record 9, the next available disk location. The scheduler places the additional records (units) for the current message segment in disk locations 10 through 16. Since this buffer does not contain an EOM indicator, the scheduler preassigns the next-buffer location to record 17.

In Figure 30, the second buffer of the message for line D is queued. This is a three-unit buffer with an EOM character in the last unit. The Destination Scheduler places the first unit in line D's next-buffer slot at record 8 and places the two additional records in the next available disk locations, records 18 and 19. No preassignment for the next-buffer location is made because of the EOM character in this buffer. The scheduler preassigned the next-message slot for line D to record 5 when the first buffer of this message was queued (see Figure 28).

In Figure 31, the 30-byte message from line A to line B is queued. Since this message is contained within a single unit, only that unit has to be written on disk. The Destination Scheduler places this unit in the preassigned next-message location for destination B, record 9. No next-buffer location needs to be preassigned, but the scheduler changes the next-message location for line B to disk record 20. The next available disk location is now record 21.

Figures 28 through 31 do not illustrate all the disk record pointers. However, Figure 32 shows the pointers mentioned above, as well as the pointers from each subsequent buffer of a message to the first buffer of the message. These pointers are the base for the _queue-back chain_ to be discussed next.

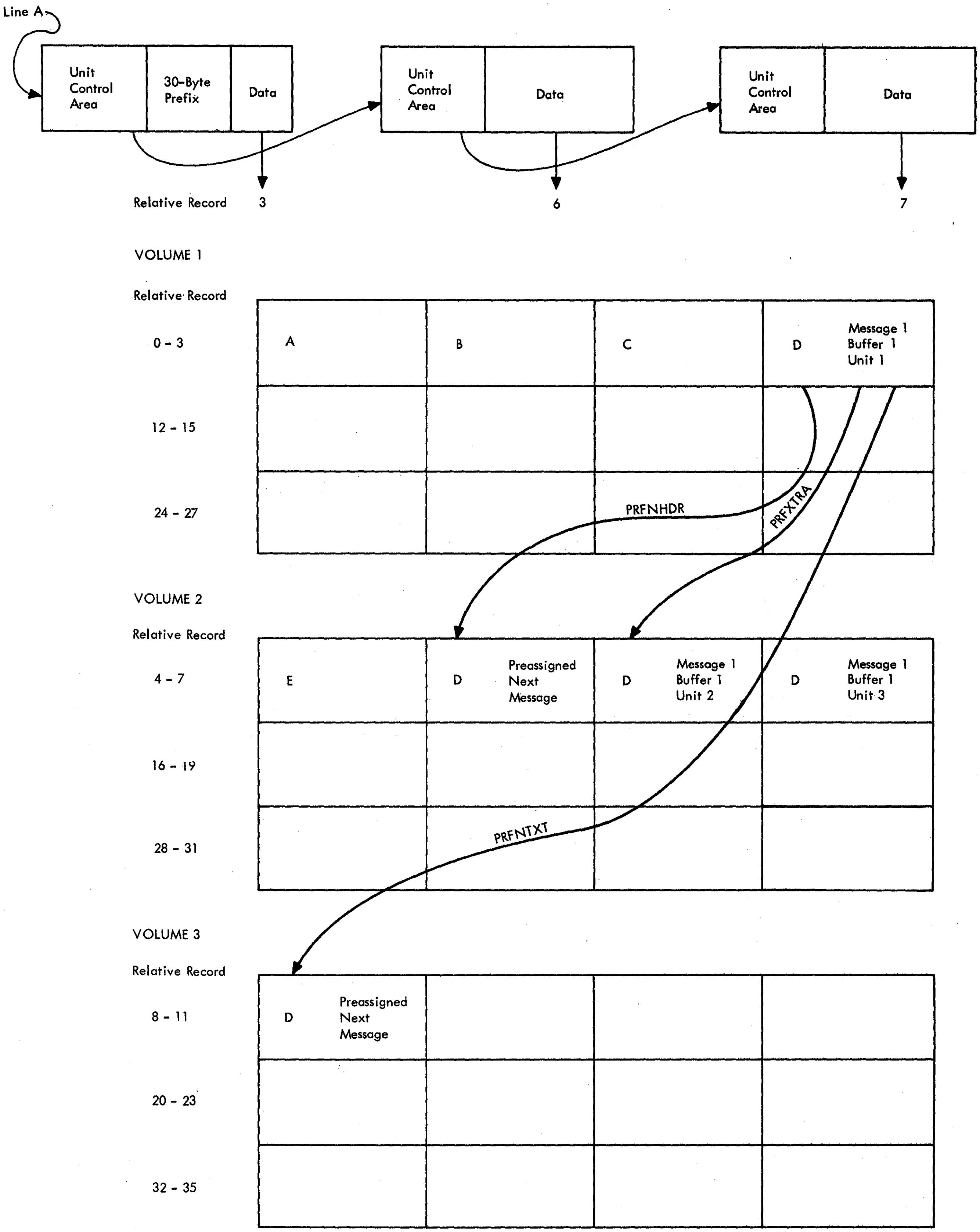


Figure 28. Disk Queuing a Three-Unit Buffer

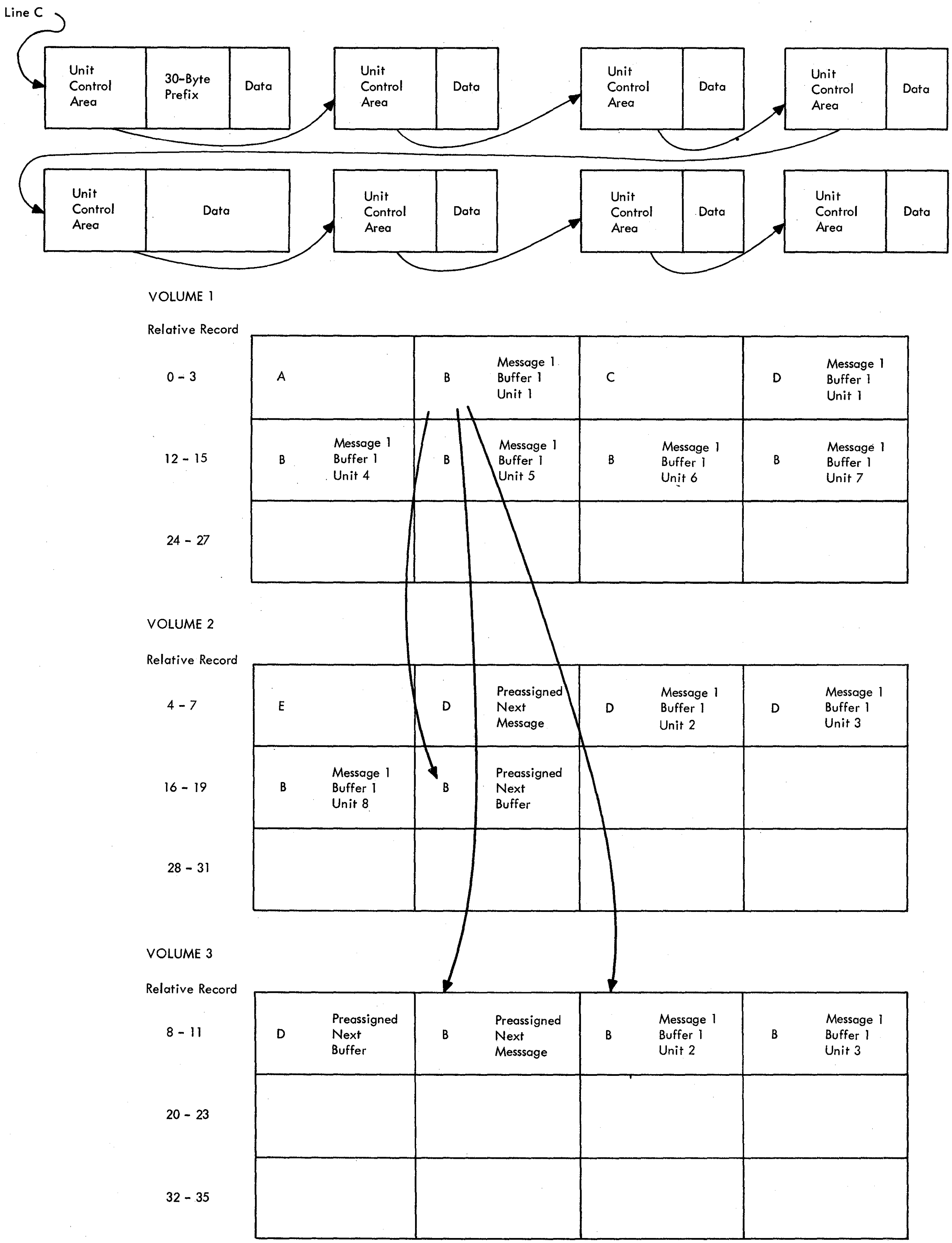


Figure 29. Disk Queuing an Eight-Unit Buffer

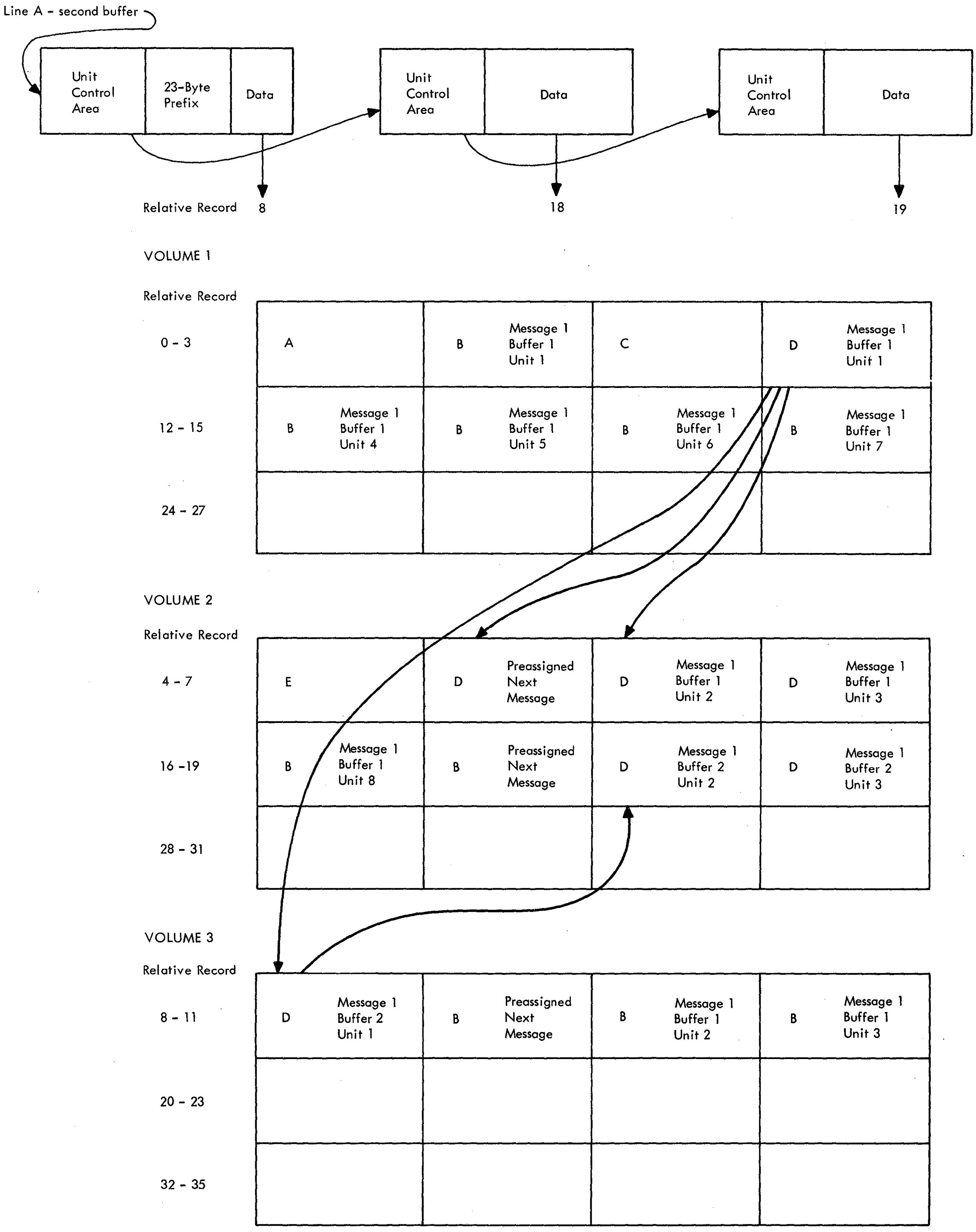
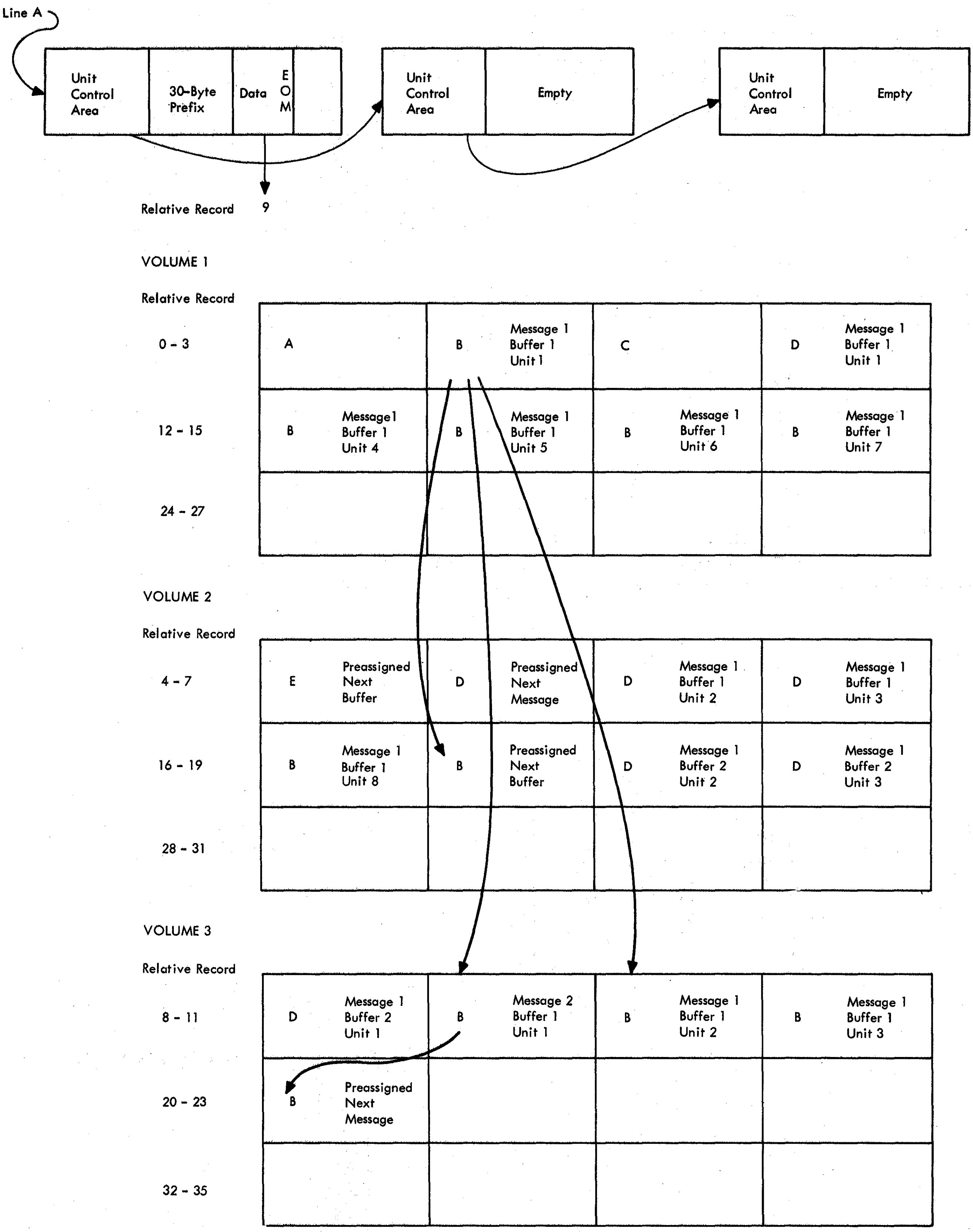

Line A - second buffer
Unit Control Area
23-Byte Prefix
Data
Unit Control Area
Data
Unit Control Area
Data
Relative Record
8
18
19
VOLUME 1
Relative Record
0 - 3
A
B Message 1 Buffer 1 Unit 1
C
D Message 1 Buffer 1 Unit 1
12 - 15
B Message 1 Buffer 1 Unit 4
B Message 1 Buffer 1 Unit 5
B Message 1 Buffer 1 Unit 6
B Message 1 Buffer 1 Unit 7
24 - 27
VOLUME 2
Relative Record
4 - 7
E
D Preassigned Next Message
D Message 1 Buffer 1 Unit 2
D Message 1 Buffer 1 Unit 3
16 -19
B Message 1 Buffer 1 Unit 8
B Preassigned Next Message
D Message 1 Buffer 2 Unit 2
D Message 1 Buffer 2 Unit 3
28 - 31
VOLUME 3
Relative Record
8 - 11
D Message 1 Buffer 2 Unit 1
B Preassigned Next Message
B Message 1 Buffer 1 Unit 2
B Message 1 Buffer 1 Unit 3
20 - 23
32 - 35

Figure 30. Disk Queuing the Second Buffer of a Message



VOLUME 1

Relative Record				
0 - 3	A	B Message 1 Buffer 1 Unit 1	C	D Message 1 Buffer 1 Unit 1
12 - 15	B Message 1 Buffer 1 Unit 4	B Message 1 Buffer 1 Unit 5	B Message 1 Buffer 1 Unit 6	B Message 1 Buffer 1 Unit 7
24 - 27				

VOLUME 2

Relative Record				
4 - 7	E Preassigned Next Buffer	D Preassigned Next Message	D Message 1 Buffer 1 Unit 2	D Message 1 Buffer 1 Unit 3
16 - 19	B Message 1 Buffer 1 Unit 8	B Preassigned Next Buffer	D Message 1 Buffer 2 Unit 2	D Message 1 Buffer 2 Unit 3
28 - 31				

VOLUME 3

Relative Record				
8 - 11	D Message 1 Buffer 2 Unit 1	B Message 2 Buffer 1 Unit 1	B Message 1 Buffer 1 Unit 2	B Message 1 Buffer 1 Unit 3
20 - 23	B Preassigned Next Message			
32 - 35				

Figure 31. Disk Queuing a One-Unit Message

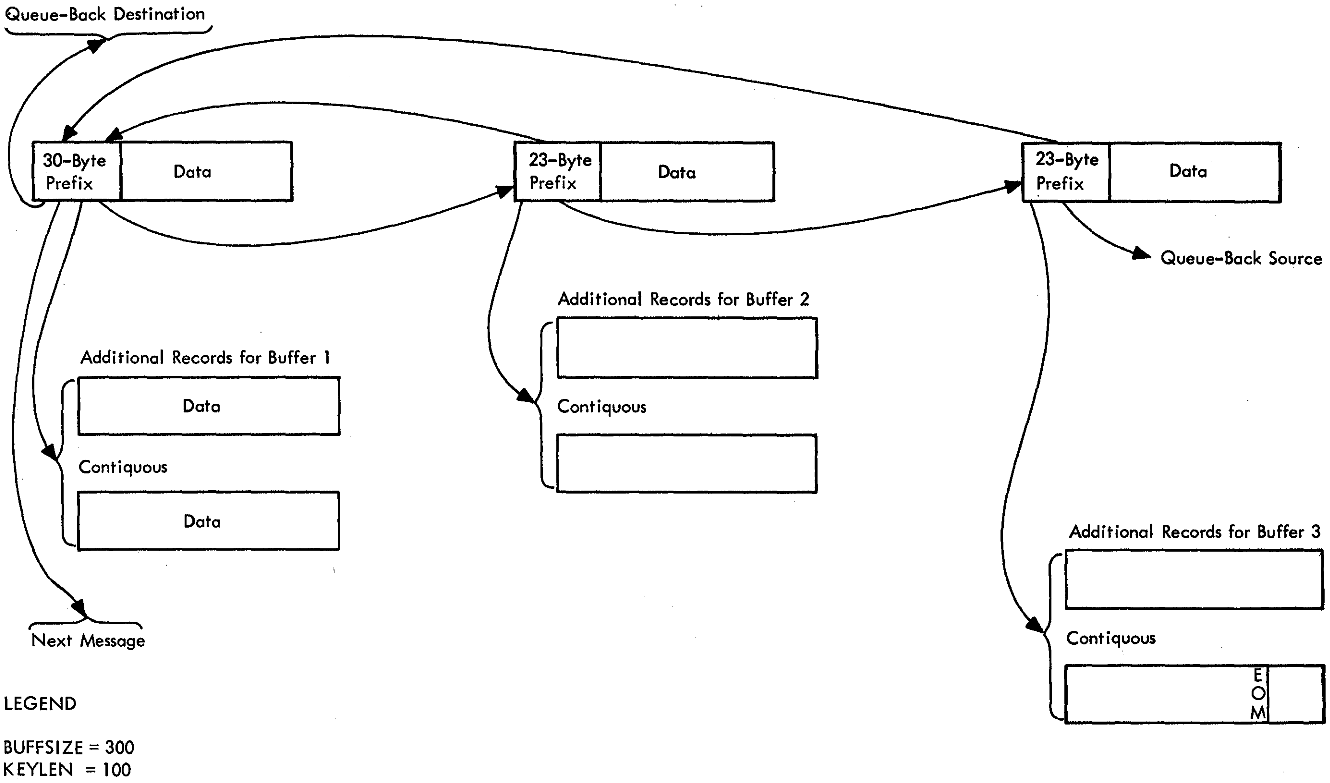


Figure 32. Disk Queuing Pointers

Queue-back Chain: A queue-back chain is a time-sequential record of the sending and receiving message traffic for the terminal or terminals of a specific Destination QCB. TCAM maintains this chain for the message retrieval function of application programs. A message that has already been sent can be retrieved by source (input) or by destination (output) sequence number.

When the first buffer of a message is posted to its Destination QCB, the Destination Scheduler moves the current queue-back chain pointer (QCBQBACK) from the Destination QCB to the PRFHQBCK field in the buffer prefix and then stores the disk relative record number (address) assignment of the first unit of the buffer in the queue-back chain field of that Destination QCB (QCBQBACK). The presence of an address for the first buffer of a message in the queue-back chain of the Destination QCB indicates that the message is to be queued for the terminal or terminals of the Destination QCB.

When the last buffer of a message is posted to its Destination QCB, the Destination Scheduler uses the source destination offset in the buffer prefix (PRFSRCE) to gain access to the associated terminal entry. The location of the Destination QCB for the sending (source) terminal is in this terminal entry. The scheduler then places the current Destination QCB queue-back chain pointer (QCBQBACK) in the text queue-back field in the buffer prefix (PRFTQBCK) and places the disk relative record number (address) of the first unit of the last buffer in the queue-back chain of the Destination QCB (QCBQBACK) for the source terminal. The presence of an address for the last buffer of a message in the queue-back chain of the Destination QCB indicates that the message was sent from the terminal or terminals represented by that Destination QCB.

An examination of the queue-back chain of a specific Destination QCB indicates exactly which messages were sent from or received by the related terminal or terminals. If the address value in the chain is for the first buffer of a message, the message was received by this terminal; if the address value is for the last buffer of a message, the message was sent by this terminal. Since the prefix of a first buffer points to its subsequent buffer segment (PRFNTXT) and the prefix of a subsequent buffer segment points to its first buffer (PRFCHDR), the entire message is available from the queue-back chain pointers.

Note that if a message is only one buffer long, its address location goes in both queue-back chains.

Figure 33 illustrates the queue-back chains for two Destination QCBs. The following message sequence applies to this example:

Message 1 - sent from Station A to Station B
Message 2 - sent from Station B to Station A
Message 3 - sent from Station A to Station B

Duplicate-Header Messages: When a message is identical to a message sent previously (as in multiple routing), it is called a *duplicate-header message*. This condition is indicated by a flag in bit 4 of the status field (PRFSTAT1) of the 30-byte buffer prefix. The Destination Scheduler handles a duplicate-header message just like any other message except that no additional record locations and no next-buffer location are assigned. The first unit of the first segment of a duplicate-header message contains the same pointers that are in the first unit of the first segment of the original message. TCAM modules use these pointers to obtain any additional units and buffers in the message.

FEFO Queuing: FEFO (first-ended-first-out) queuing is used in sending messages from the message queues data sets to destinations. This queuing allows TCAM to send the messages that end (EOT received) first, rather than the messages that begin transmission first.

Since the segments of a message cannot be kept in main storage until the message completes, they must be queued (placed on the disk) as they are received. This results in a FIFO (first-in-first-out) message queue.

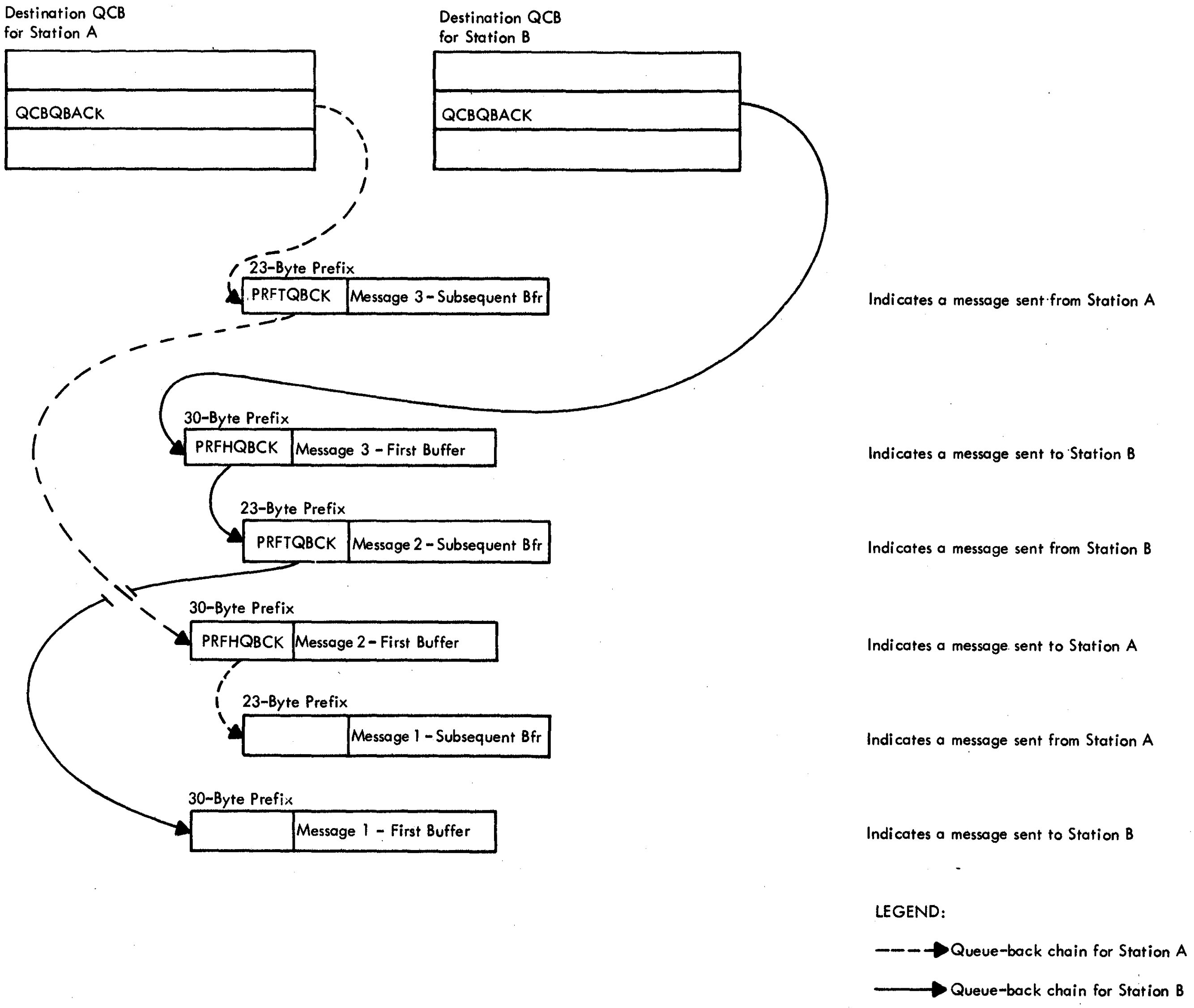

Destination QCB for Station A
QCBQBACK
Destination QCB for Station B
QCBQBACK
23-Byte Prefix
PRFTQBCK
Message 3 - Subsequent Bfr
Indicates a message sent from Station A
30-Byte Prefix
PRFHQBCK
Message 3 - First Buffer
Indicates a message sent to Station B
23-Byte Prefix
PRFTQBCK
Message 2 - Subsequent Bfr
Indicates a message sent from Station B
30-Byte Prefix
PRFHQBCK
Message 2 - First Buffer
Indicates a message sent to Station A
23-Byte Prefix
Message 1 - Subsequent Bfr
Indicates a message sent from Station A
30-Byte Prefix
Message 1 - First Buffer
Indicates a message sent to Station B
LEGEND:
Queue-back chain for Station A
Queue-back chain for Station B

Figure 33. Example of Two Queue-Back Chains

To create a chain of messages in FEFO order, the message with the previous EOT received for a Destination QCB must chain to the message with the current EOT, regardless of which message began transmission first. This chaining pointer cannot be written until after the current EOT is received. When the current EOT is received, one message is completely on disk and the other is on disk except for the last segment. A temporary chain of first-buffer prefixes is all that is required: therefore, the FEFO pointer can be written in the data field (at DATFEFO) of the record that contains the first-buffer (30-byte) prefix of the message already on disk at the same time the EOT segment of the current message is written.

When the first-ended message is to be sent and its first segment is read from disk, the FEFO pointer is read from the data field of the record and placed in the FEFO field of the SCB. When the first buffer is passed to the outgoing MH, the STARTMH subtask updates the FEFO field in the Destination QCB. The "message serviced" flag (X'40') is written in the disk data field along with the FEFO pointer when the EOM is successfully sent.

The Destination QCB contains two FEFO pointers: the disk record address of the first FEFO message to send to the destination (QCBFFEFO) and the disk record address of the last message completely received (QCBLFEFO).

Figure 34 illustrates FEFO queuing for five messages routed to the same destination. Messages 1, 3, and 4 require two buffers, and messages 2 and 5 require one buffer. The first buffers of the messages arrive in the order in which the messages are numbered. The messages complete transmission in the following order: 2, 4, 3, 1, 5.

In this example, assume that the first buffers of messages 1, 2, 3, and 4 are already written on disk, message 2 is complete, and the first buffer of message 5 is currently being transmitted. The FEFO queuing activity proceeds as follows:

- Message 2 is written out on the line. No FEFO pointers were written when message 2 completed because it was the first message for the destination.

- Message 4 completes being received. Message 2 is still sending. QCBFFEFO and QCBLFEFO are updated to point to disk address 8 and no disk pointers are written for FEFO queuing.

- Message 3 completes being received. A FEFO pointer to message 3 is written in the disk data field of the first unit of the first buffer of message 4. The Destination QCB field QCBLFEFO is updated to point to disk address 7.

- Message 2 completes being sent. Message 4 is to be sent out. When the first buffer of message 4 is sent to MH, its disk data field is used to update the QCBFFEFO field of the Destination QCB to point to disk record 7.

- Message 1 completes being received. A FEFO pointer to message 1 is written in the disk data field of the first buffer of message 3. The Destination QCB field QCBLFEFO if updated to point to disk address 1, the location of the first buffer of message 1.

- Message 5 completes being received. A FEFO pointer to message 5 is written in the disk data field of the first buffer of the last message received, message 1. The QCBLFEFO field is updated to disk address 10, the location of the first unit of the first buffer of message 5.

- Message 4 completes being sent. Message 3 is the next message to be sent. When the first buffer of message 3 is sent to MH, its disk data field is used to update QCBFFEFO to point to message 1 in disk location 1, the next message to be sent.

- Message 3 completes being sent. Message 1 is the next message to be sent. When the first buffer of message 1 is sent to MH, its disk data field is used to update QCBFFEFO to point to message 5 in disk location 10, the next message to be sent.

- Message 1 is completed and message 5 is sent out. The QCBFFEFO pointer is cleared.

Note that the FEFO chain is, in many cases, incomplete. In the example there is no FEFO pointer from message 2 to message 4. If messages for a destination are always completely received after the previous message has been sent out, no FEFO chain is built.

Hold Queues: When the HOLD macro is issued in the outgoing section of an MH, a special hold queue is built for multidrop terminals on a line that is queued by line.

When queuing multidrop terminals by line, the messages for the different terminals are intermixed on the destination queue. The Send Scheduler uses the FEFO chain to read one "first buffer of a message" after another. When a message for a held terminal is reached, it is placed in the hold queue chain.

A pointer to the first held message is placed in the QCBINTFF field of the Destination QCB. When the next held message is encountered, its address is placed in the data field of the first unit of the first buffer of the previous held message. This pointer overlays the FEFO pointer and is used when the messages are being released.

This queuing continues until a RELEASE command occurs. The messages are then sent in FEFO order by following the chain that was built for the hold queue. The hold queue is merged into the FEFO chain by making the first held message for the QCB the first FEFO message and by making the last held message point to the message that was the first FEFO message.

Queuing by terminal must be specified for dial lines, and messages are not intermixed on a message queue. In this case, only one message is in the hold queue, because the Send Scheduler determines that the terminal is held and does not request any more messages.

Reusable Disk Queuing

Reusable disk queuing uses a wrapped message queues data set, on which serviced messages are overlaid by new messages entering the system.

The Destination Scheduler activates the Reusability-Copy subtask to keep the data set "cleaned up" to avoid losing messages that have not been serviced. Message units are queued until 3/8 of the data set is full. At this point, the Reusability-Copy subtask examines the next-message field in each Destination QCB for this data set. If any next-message field has a location value that falls within the scope of the first quarter of the data set, the subtask writes a dummy cancel message record at the specified next-message address and updates the

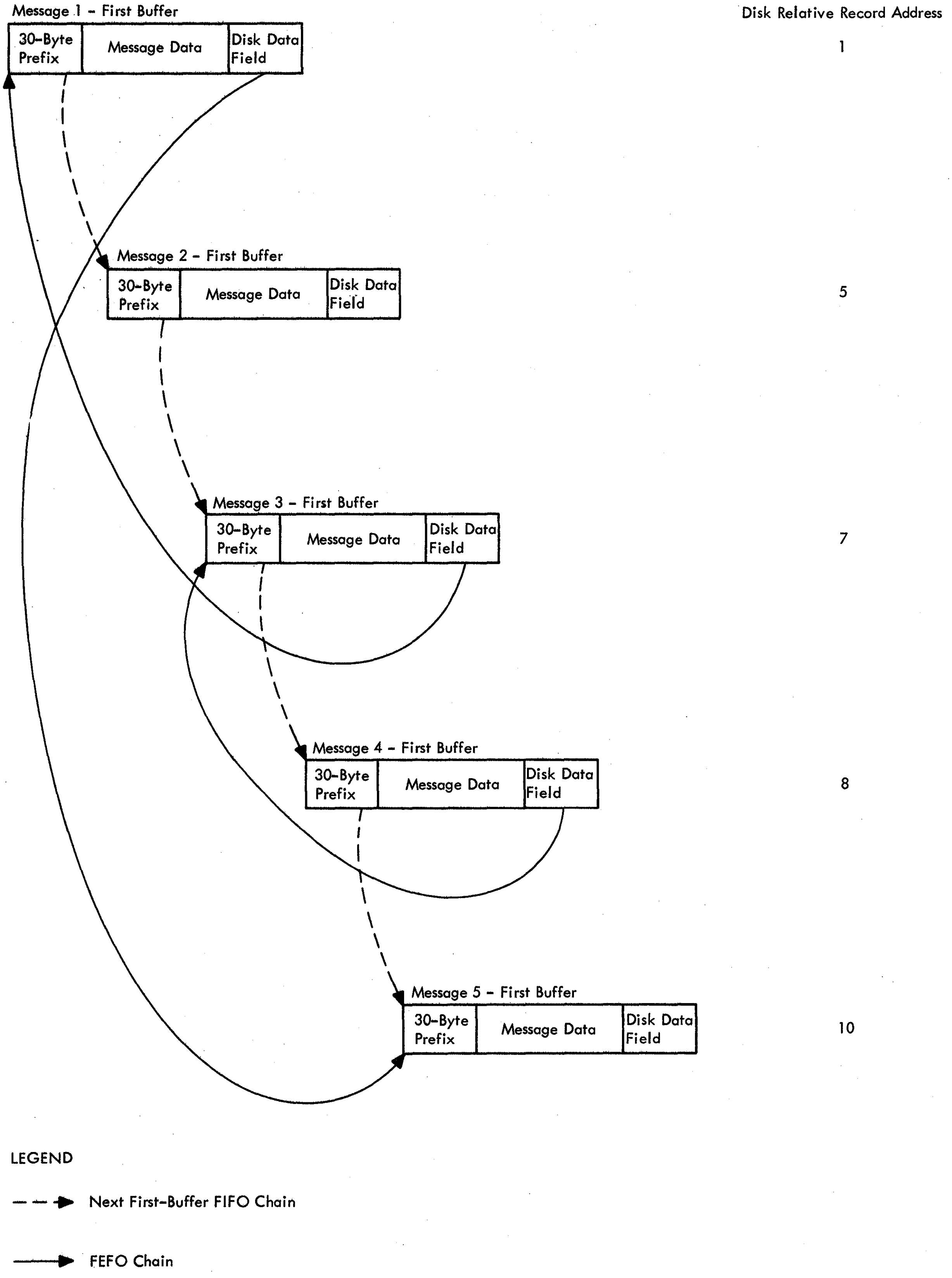


Figure 34. Disk Queuing - FIFO and FEFO Pointers

next-message field in the OCB to the current address value at AVTRADDR in the AVT. This keeps new messages in fairly close proximity on the data set.

The Reusability-Copy subtask performs the next-message update process each quarter of the way through the data set from this point on. For example, after 5/8 of the data set has been assigned to units, the Reusability-Copy subtask compares the address values in the second quarter to the next-message location specified in each Destination OCB for this data set.

The Reusability-Copy subtask sends to the specified alternate destination any unserviced messages located in the quarter that precedes the part of the data set that is getting dummy cancel record messages. The subtask does this by reading the old message from its current location and enqueuing the message to its alternate destination, thus causing the message tc be written in the current zone of the data set.

If a duplicate-header message is more than a quarter of the data set away from the first unit cf the first segment of the original message, the Reusability-Copy subtask copies the entire message.

The Reusability-Copy subtask gains control each time the address value reaches a zone boundary (the middle of a quarter) of the data set. The only exception is that the first time through the data set, it is not activated until the address value is 3/8 of the way through the data set.

Figure 35 illustrates the part of the disk message queues data set that is issued cancel messages and the part in which messages are sent to alternate destinations when the address value is at a specific zone boundary.

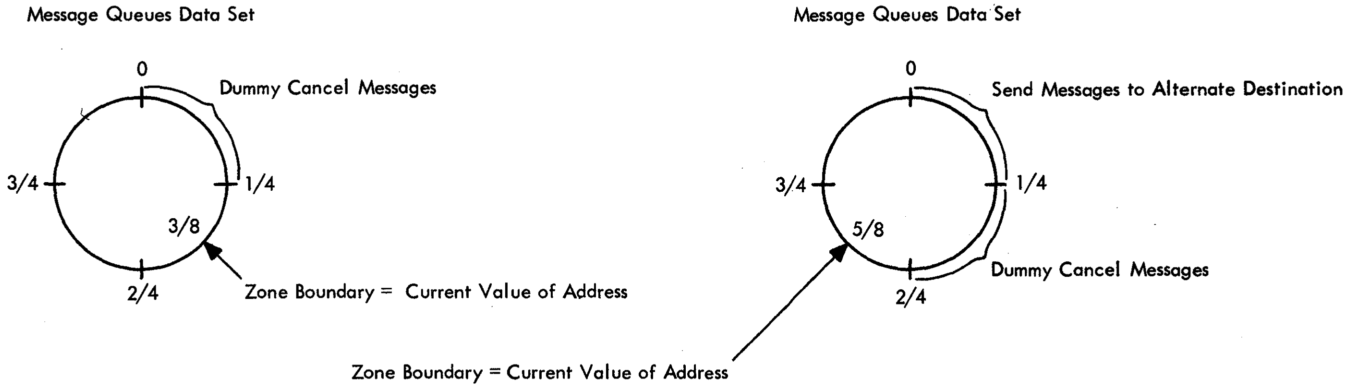

Message Queues Data Set
0
Dummy Cancel Messages
3/4
1/4
3/8
2/4
Zone Boundary = Current Value of Address
Message Queues Data Set
0
Send Messages to Alternate Destination
3/4
1/4
5/8
Dummy Cancel Messages
2/4
Zone Boundary = Current Value of Address

Figure 35. Zones for Servicing and Updating a Reusable Disk Message Queues Data Set

Main Storage Queuing

Main storage queuing chains the actual main storage addresses of message units, rather than using relative record numbers. Once an entire message is queued, all the fields in the buffer prefix look the same as in disk queuing, except that the Destination Scheduler uses the additional units field (PRFXTRA) of the buffer prefix to hold the main storage address of this unit and the current record field (PRFCRCD) to hold the disk address if disk backup is used. The scheduler uses the TIC field of the twelve-byte unit control area that precedes each unit to chain units together.

Main storage queuing does not assign locations ahead; rather, the Destination QCB contains the address of the previous first-buffer segment and the SCB contains the address of the previous subsequent-buffer segment. When the first segment of a message is received, the address of the previous first-buffer segment is inserted in the Destination QCB in the previous first-buffer field (QCBCPBHD). When a message segment other than the first-buffer segment is received, its address is placed in the previous subsequent-buffer field of the SCB.

The Destination Scheduler does not build a queue-back chain for a main storage message queues data set.

Main Storage Queuing with Disk Backup

If the user specifies main storage queuing with backup on either reusable or nonreusable disk, the message segments are first queued as described under "Main Storage Queuing" and then the data is copied into buffers for the disk message queues data set and queued as described in the sections on disk queuing.

If the Destination Scheduler finds that the main storage message queues data set does not contain enough free units to queue a message, the scheduler queues the message on disk only. Main storage queuing resumes as soon as space is available. The CPB Initialization routine retrieves the messages queued on disk just as if they were placed in the main storage data set.

Special Queuing Considerations

Duplicate-Header Message that Spans Queue-Type: A duplicate header message that spans queue-type is one that is posted to a Destination QCB that is to be queued in a manner other than that of the original message. For example, the original message is directed to a Destination QCB that uses reusable disk queuing and the duplicate-header message is directed to a Destination QCB that uses main storage queuing with no disk backup.

If the entire message does not have to be copied, the Destination Scheduler moves the Send Scheduler STCB to the STCB chain of the LCB (if it is not already there) to service the message. If the message has to be copied, the Reusability-Copy subtask is activated.

Destination QCB for Main Storage Queuing with Disk Backup: In this situation all recalls are from disk; therefore, the duplicate-header message is written on the disk data set only.

Main Storage Queuing when Units Run Out: If a main storage message queues data set fills up with data and there is a message segment unit to be queued, the Destination Scheduler acts according to the type of unit being processed. If the unit is not the first unit of the first segment of a message, the scheduler gets the first segment of the message, flags the message lost, and frees all the queued units except the first one.

If the unit to be queued is the first unit of the first segment of a message and one unit is available in the data set, the scheduler queues the unit and flags the message lost via a flag in that unit. If no unit is available or if the count of units in the main storage queue exceeds or equals MSMAX (specified on the INTRO macro) in the data set, the scheduler queues the buffer unit that contains the first unit of the message into the data set, does not return a unit to the buffer unit pool in its place, and sets a flag to stop receiving activity. Receiving is resumed when enough messages have been sent to remove enough units from the message queues data set to lower the number of units used to or below MSMIN (specified on the INTRO macro).

Queuing Management Routines

The disk and main storage queuing functions just described are performed by the Destination Scheduler. The receive schedulers and the send schedulers handle the messages before and after the queuing is performed. The TCAM Dispatcher activates each of these routines when its STCB has top priority in the STCB chain of an LCB. A send scheduler may also be activated from the STCB chain of a Destination QCB.

At line open time, each LCB has an STCB for a receive scheduler built in it. This receive scheduler STCB starts in the third word of the LCB. This word is also the STCB pointer field. (See Figure 36.) The high-order byte of the third word of the LCB is the activation key of the STCB.

Every Destination QCB has the same format regardless of whether it represents an application program or a terminal. If the user indicates queuing by line, there is one Destination QCB per line; if queuing by terminal is specified, there is one Destination QCB per terminal.

Every Destination QCB has an STCB pointer in its third word, and after open time it points to the send scheduler STCB that starts in that same word. The link field of the send scheduler STCB is assembled to point to the STCB for the Destination Scheduler routine.

Figure 36 shows the pointers in an LCB and a Destination QCB after line open time.

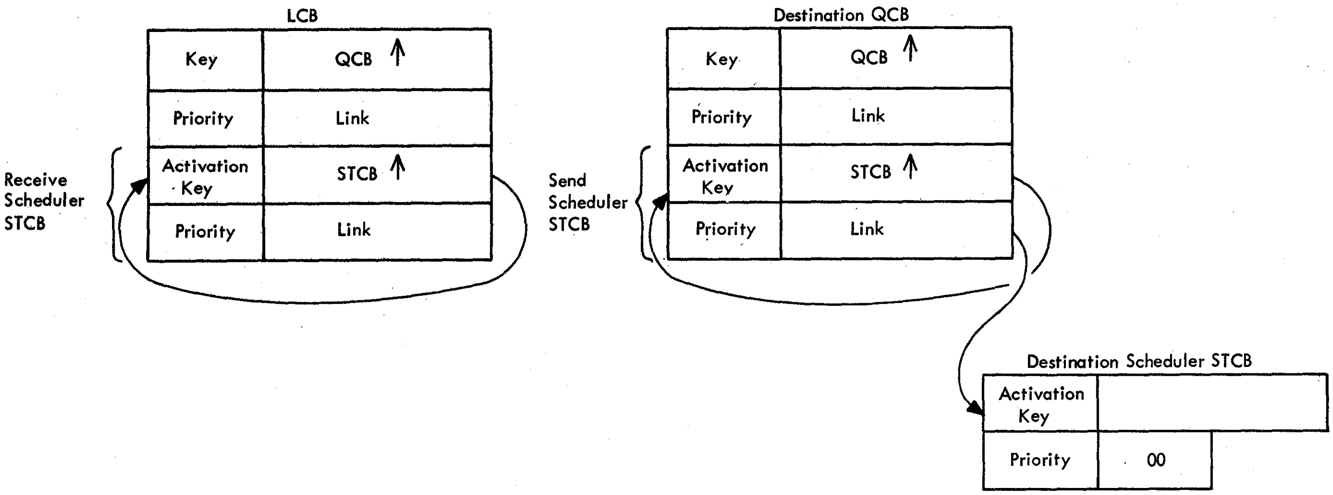


Figure 36. Format of an LCB and a Destination QCB after Line Open Time

The priorities of the receive scheduler STCB and of the send scheduler STCB are determined when the user specifies whether he wants send or receive priority for a line. When the send scheduler STCB is moved from the STCB chain of the Destination QCB to the STCB chain of the LCB, it is inserted in that chain by priority.

When a buffer is tposted to a Destination QCB, the TCAM Dispatcher activates the subtask of the first STCB in the STCB chain. The first STCB may be the one for the Send Scheduler or the one for the Destination Scheduler. If the Send Scheduler is in the chain, it bypasses control to the Destination Scheduler.

The send scheduler STCB is removed from the STCB chain of a Destination QCB when either an initiate mode message or the last buffer of a message is tposted to the Destination QCB. The Destination Scheduler gains control , tests for the conditions just mentioned, and activates a subroutine of the first scheduler that appears in the chain. This subroutine branches to the TCAM Dispatcher requesting that its STCB be removed from the STCB chain of the Destination QCB and placed by priority on the STCB chain of the LCB for the line. This action indicates that the send scheduler has a message to send. When its STCB is twaiting in the STCB chain of a Destination QCB, it is waiting for a complete message to be tposted.

Figure 37 shows the pointers in an LCB and a Destination QCB with send priority after a full message has been received.

An LCB is tposted to the ready queue when the line is free after an I/O operation has been completed. The QCB pointer in the first word of the LCB is set to point to the LCB itself, so that when the LCB is tposted to the ready queue, it functions as an RCB, a QCB, and an LCB.

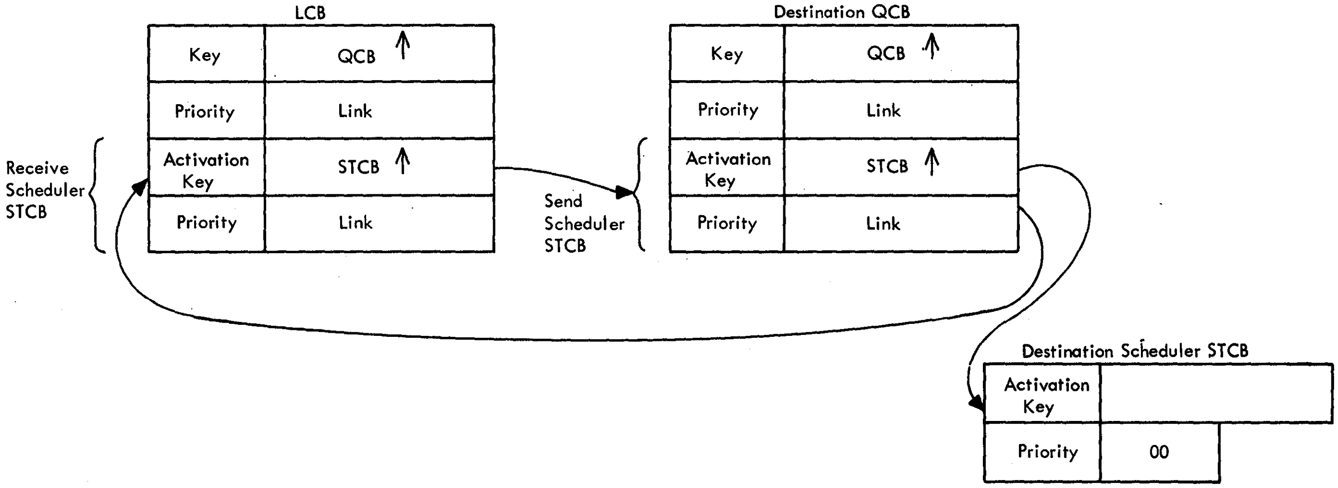


Figure 37. Format of a Send Priority LCB and Destination QCB after a Full Message Has Been Received

The Send Scheduler gets control of a free line when it is the highest priority STCB in the STCB chain of an LCB tposted to the ready queue. The Send Scheduler initializes the LCB for sending and tposts an ERB for the necessary buffers to the Disk I/O QCB. After the message has been sent, the LCB is tposted free again and put on the ready queue. Since the Send Scheduler did not remove its STCB from the STCB chain of the LCB, it regains control to determine whether there is another message to send. If there is another message, the functions are the same as above. If there is not another message, the Send Scheduler tposts the line free and returns to the Dispatcher requesting that its STCB be removed and placed in the STCB of its Destination QCB to await another complete or initiate mode message.

The Destination Scheduler STCB is always the last member of the STCB chain of a Destination QCB. Whenever a buffer is tposted to a Destination QCB, the Destination Scheduler eventually gains control to queue the buffer into the specified message queues data set.

The purpose of the Destination Scheduler is threefold:

- To chain the segments of a message together,

- To chain messages related to a specific Destination QCB together,

- To build a queue-back chain, where applicable, to allow the retrieve function to be performed. (There is no queue-back chain for main storage queuing.)

These functions can be performed with the message segments queued in main storage, on a direct-access storage device (disk), or in main storage with backup on reusable or nonreusable disk.

Disk I/O Management Routines

The Destination Scheduler tposts the units of a buffer to the Disk I/O QCB after each unit has been assigned an address value for the disk message queues data set. When a full buffer is tposted to the Disk I/O QCB, the disk I/O management routines are activated.

Functions of CPB Initialization: CPB Initialization is the only subtask pointed to by the STCB chain of the Disk I/O QCB. This module gains control when full buffers or ERBs are tposted to the Disk I/O QCB or when the CPB Cleanup routine branches to it. (The CPB Cleanup routine is actually a part of the CPB Initialization module.)

The primary function of CPB Initialization is to build a CPB for the element that was tposted to the Disk I/O QCB. Partial CCWs are built for the CPB and the CPB is added in FIFO order to the input queue for the EXCP Driver routine. The CPB Initialization routine then branches to the EXCP Driver routine.

When CPB Initialization gains control, it queues the element from the ready queue onto its no-CPB queue in FIFO order. If the request is to flag the message serviced, the element is placed at the beginning of the no-CPB queue. This is the queue of elements to be processed by this routine. CPB Initialization then proceeds to process the first element on the no-CPB queue. The routine processes each element on the queue, in turn, until it either processes all the elements on the queue or uses all the CPBs available from the CPB free pool. If, during the processing of a buffer, the routine runs out of available CPBs, it returns the unprocessed part of the buffer to the first of the no-CPB queue and puts the processed portion on the EXCP Driver input queue. If the element being processed is an ERB, the above holds true only if no CPB is available. If one CPB is available, the routine processes that CPB, places the CPB that refers to the ERB on the EXCP Driver input queue, and removes the ERB from the no-CPB queue.

When the CPB Cleanup routine branches to CPB Initialization, it processes the no-CPB queue as described above and continues to place CPBs on the EXCP Driver input queue. When entered from the Dispatcher, the element can be either a buffer or an ERB. This element must be placed in FIFO order on the no-CPB queue so that processing can start from the beginning of this queue.

A buffer on the no-CPB queue always causes the CPB Initialization routine to build a CCW to read or write key and data for the unit, or to build a CCW to read or write data for the FEFO pointer, and to fill in the address value representing the record this unit of the buffer is to be associated with. The other activities of the routine depend on the characteristics of the buffer itself.

- A buffer to be put on disk. CPB Initialization chains each buffer unit to a CPB and tposts the unit of the CPB to the Buffer Return QCB to be put in the buffer unit pool. The routine then places the CPBs - one for each buffer unit - in FIFO order on the EXCP Driver input queue. If this buffer is the last segment of the message, the routine builds the FEFO pointer and places the address of the first unit of the first buffer of the message in the FEFO chain.

- A buffer of a canceled message. If the message is disk or disk-backup queued, the CPB Initialization routine sets the cancel bit to be written in the data field of the record that contains the first buffer of the message on disk, and places the CPBs for the data field and the buffer on the EXCP Driver input queue.

- A buffer for a serviced last segment. If the message main storage queued and not a duplicate-header message, the CPB Initialization routine tposts all the units of the message to the Buffer Return QCB. If the message is main storage queued and not the last duplicate-header message, the routine tposts the first unit to the Buffer Return QCB and subtracts one from the count of duplicate-header messages. The FEFO pointer in the Destination QCB was updated when the first buffer started through MH. If the message is disk queued, CPB Initialization sets the serviced bit to be written on disk.

- A duplicate-header buffer. The CPB Initialization routine puts the CPB for the first unit on the EXCP Driver input queue and tposts the buffer to the QCB specified in its LCB.

If an ERB appears on the no-CPB queue, the SCB has been initialized with the address of the record to be read. If the ERB is an initial request, only one record can be read. If an initial request ERB is for a main storage queued record, CPB Initialization determines whether the record is available and, if it is, branches to the CPB Cleanup routine (there is no I/O to be executed). Otherwise, the routine builds CPBs for as many disk records as the pointers in the last-read buffer allow, or until the requested buffers will be filled.

Once the CPB Initialization routine has processed all the elements on the no-CPB queue or has used all the CPBs available from the CPB free pool, it branches to the EXCP Driver routine.

Functions of the EXCP Driver Routine: The functions of the EXCP Driver routine are to complete the building of the CPBs, to chain them together, and to issue EXCP commands to perform all disk I/O functions concerning the disk message queues.

On the EXCP Driver input queue, each CPB contains the read or write CCW, the record number (address), a chaining pointer to the next CPB, and a unit (filled in for a write only). Before an I/O operation can occur, the disk extent and cylinder identification must be filled in and converted to MBBCCHHR format. The EXCP Driver routine issues a BAL to its MBBCCHHR Convert subroutine, which uses values set during the open of this message queues data set and the address value to calculate the MBBCCHHR value. The "M" or extent ID is an index to the block of consecutive IOBs, which, when multiplied by the size of an IOB and added to the address of the first IOB, points to the appropriate IOB and its queues. There is one IOB per volume (or extent).

For a disk message queues data set, the IOB for each extent is extended to include an EXCP busy flag, a "lock door" flag, a "cc" identifier, a retry queue, and a new queue. An EXCP queue is located in the regular IOB at IOBSTART.

- The EXCP busy flag (IOBBUSYN) is set while I/O is being executed for its IOB.

- The "lock door" flag (IOBXLOCK) is set while enabled code is putting CPBs on the retry queue.

- The "cc" identifier (IOBXCC) is the cylinder number of the last group of CPBs put on the retry queue. This is the top priority cylinder for new CPBs being put on the new queue.

- The EXCP queue (IOBSTART) is the chain of CPBs for the cylinder currently ready for I/O to be executed.

- The retry queue (IOBXRETQ) is the chain of CPBs for the cylinder that is to have I/O executed after the CPBs on the EXCP queue are processed. If I/O is being executed for the CPBs on the EXCP queue and a CPB arrives for the cylinder being read, the CPB is put at the end of the new queue.

- The new queue (IOBXNEWQ) is the chain of CPBs for all the other cylinders, in order from the next available cylinder after the retry queue to the end of the data set, then starting with the cylinder at the beginning of the data set again.

The EXCP Driver routine processes the CPBs on its input queue one at a time in FIFO order. When a CPB is placed on its proper IOB queue, the chaining flags are set and the seek/search CCWs are built in the CPB as appropriate. If there is not a channel program in progress for this IOB, EXCP Driver issues an EXCP command to start one.

After EXCP Driver has inserted all the CPBs on its input queue into an IOB queue, it scans all the IOBs to perform two functions:

- If there is not a channel program in progress for an IOB that has CPBs to be processed, this module issues an EXCP command for that IOB.

- If there are no CPBs on the retry queue and there are CPBs on the new queue, this module transfers the CPBs on the first cylinder to the retry queue.

After all the above functions are completed, the EXCP Driver routine branches to the TCAM Dispatcher

Functions of Disk End Appendage: When the channel finishes executing the I/O for a CCW chain, a Disk End Interrupt causes the Disk End Appendage to gain control. The function of Disk End Appendage is to dispose of the chain of CPBs just processed.

Disk End Appendage enqueues the CPBs that are on the EXCP queue of the IOB onto the disk end queue. The appendage then tposts the CPB Cleanup QCB to itself and puts it on the disabled ready queue (if it is not already tposted and on the ready queue). This causes the CPB Cleanup routine to be activated to process the CPBs on the disk end queue.

Disk End Appendage then OS posts the TCAM ECB complete to indicate the completion of I/O activity.

Disk End Appendage examines the retry queue cf the IOB. If the "lock door" flag is set or if there are nc CPBs on the queue, the appendage returns to IOS with channel activity stopped. If there are CPBs on the retry queue, they are chained tc the EXCP queue; and the appendage returns to IOS to restart on the new CCWs.

Functions of CPB Cleanup: When the CPB Cleanup QCB that was tposted to itself by the Disk End Appendage gets to the top of the ready queue, the TCAM Dispatcher activates the CPB Cleanup routine in the CPB Initialization module. The CPB Cleanup routine can also be activated when a buffer from the Buffer Return subtask is tposted to the CPB Cleanup QCB or by a branch from the CPB Initialization routine when a read operation was requested for a record that is queued in main storage.

The function of the CPB Cleanup routine is to free the CPBs for an I/O operation that has been completed. If the routine is activated by the CPB Cleanup QCB tposted to itself, there are CPBs tc be handled from the disk end queue. The CPB Cleanup routine processes these CPBs as described belcw.

If the CPB Cleanup routine is activated by a buffer on the ready queue, there is a CPB(s) associated with the same ERB as this buffer on the no-buffer queue. The CPB(s) is found, put on the disk end queue, and then processed normally.

The CPB Cleanup routine processes the CPBs from the disk end queue one at a time in FIFO order. If the CPB is from a write operation, the routine returns the CPB to the CPB free pool. If the CPB is from a read operation, its unit ccntains good data that has to be incorporated into a buffer. If a buffer is available from the buffer unit pool, the CPB Cleanup routine either chains the buffer off the chain field cf the ERB that was previously tposted to CPB Initialization, or gives the buffer unit to the CPB and chains the CPB unit to the ERB. The routine transfers the data from the CPB to the proper unit of the buffer and returns the CPB to the CPB free pool. If a buffer is not available, the routine places the CPB on the no-buffer queue and places the ERB in the waiting ERB chain of the Buffer Return QCB.

After all the CPBs on the disk end queue have been processed, the CPB Cleanup rcutine branches to CPB Initialization. This branch ensures that EXCP Driver will get control again to process CPBs that may still be waiting on the retry queue. It also ensures that elements (ERBs cr buffers) that are waiting for CPBs have another chance to be processed.

Multiple Arm Support

Multiple arm support for a disk message queues data set ensures a spread of message traffic over more than one volume of the data set. This support arises from the way the IOBs and EXCPs are configured and the way the records are numbered. (See Disk Queuing earlier in this section for a discussion of record numbering.)

When the data set is opened, an IOB is built for each volume. (There is one extent for each volume.) This allows TCAM to issue several EXCPs, one per IOB or extent. Performance increases when IOS has several EXCPs to work on.

If all the volumes of the data set are on one channel, maximum activity is not achieved, because when two requests for I/O are outstanding, only one can be honored. There can only be an overlap of seek time. If the records are on different volumes that are on different channels, the I/O requests can be executed concurrently.

Record numbering on the disk data set is by cylinders. All the records of a given track are numbered consecutively before going to a different volume. Also, all the tracks for a cylinder on a volume are numbered before going to a different cylinder; therefore, all the I/O for a given cylinder can be accomplished before entering the appendage to tell the I/C Supervisor (IOS) to seek another cylinder. At this point, a retry for other records for the same cylinder is executed if the CPBs on the retry queue are for this cylinder. This prevents moving the disk arm. The channel enters as though there is a fresh EXCP. If a change of cylinders is necessary, the channel lets another request for I/C take control while the arm is moving.

Once a track of a cylinder on a given volume has been assigned record numbers, a track of a cylinder on another volume is numbered with the next consecutive values of address. As a result, traffic is distributed across the volumes.

Figure 27 illustrates the record numbering scheme for a data set that has four records per cylinder on three volumes. If three additional records of a message fall together in record numbers 4, 5, and 6 of the volumes in Figure 27, they can be retrieved with one search and three reads. If the first unit is in record number 3, the searches for the entire message can be overlapped.

Multiple arm support is designed to gain access to the data with a sweep of the disk arm from the outside cylinder inward. This eliminates time-consuming disk arm movement.

SPECIAL MESSAGE HANDLING FUNCTIONS

Hold Function

The hold function may be activated by a HCLD macro in an outmessage subgroup; a terminal may be selected to be held if an attempt to transmit a message to it fails. Terminals using main-storage-only queuing cannot be held. Buffer Disposition activates the Hold/Release Terminal routine, which sets the "hold" bit in the appropriate entry in the Terminal Table. This prevents messages from being sent to the terminal. The message in error for the terminal is placed on the held-FEFO chain in the Priority QCB.

A terminal can be held at any time by Operator Control. In this case, no message is placed in the held-FEFO chain, but the terminal is marked as held in the Terminal Table.

If messages are being queued by terminal, the Destination QCB is marked as held. The Send Scheduler does not attempt to send messages to the specified destination, even though messages are placed on the destination queue.

If messages are being queued by line, the appropriate terminal entry is marked as held by the Hold/Release Terminal routine or by Operator Control. The Send Scheduler attempts to send messages as usual, since it does not recognize that there is a held terminal on the line. When the Send Scheduler requests a message destined for a held terminal, CPB Initialization removes the message from the FEFO chain of messages and places it on the held-FEFO chain.

When the terminal is released at the end of the specified time interval or by Operator Control, the Hold/Release Terminal routine takes the held messages from the held-FEFO chain and places them at the head of the destination-FEFO chain, on a Priority QCB basis, and turns off the appropriate terminal entry "hold" bit. The Send Scheduler may then transmit these messages normally.

Cancel Message Function

The cancel message function allows the user to cause immediate cancellation of a message if any of the errors specified in the error mask operand of a CANCELMG macro should occur. If the error mask is omitted or is specified as all zeros, the message is canceled unconditionally.

The error mask is examined in the inmessage subgroup. If the message is to be canceled, Buffer Disposition activates the Cancel Message routine, which sets a flag in the buffer prefix to notify the Destination Scheduler and the CPB Initialization routine to cancel the message currently being received.

If the incoming message is placed on the disk message queue, it is not placed in the FEFO chain of messages. No attempt is made to send the message. CPB Initialization cancels the message by setting the "canceled" bit in the data portion of the header field in the message.

If main-storage-only queuing is being used, the Destination Scheduler places the message, flagged as canceled, on the FEFO chain of messages. No attempt is made to send the message when it comes to the top of the queue.

Lock Function

The lock function allows the user to hold the line connection between a station and an application program. No incoming messages are accepted from any other station on the line while the station is in lock mode, and no messages other than the response message from the application program are sent to any station on the line.

Lock mode is entered either unconditionally or when a message header containing a control character (or character string) is processed by a LOCK macro specifying that character. LOCK is not executed if the message destination is not an application program. (The destination is specified either in the message header or by a FORWARD macro.)

When a message is received from a terminal requesting lock mode, the inheader subgroup examines the header to determine whether or not LOCK is to be executed. When the Lock routine gets control, it sets a switch in the SCB and turns on the "lock" bit in the PRFSTAT1 field of the buffer prefix to indicate that the message is in lock mode. The message buffer is then tposted normally to the application program Destination QCB. When the last message segment is received, it is processed through the MH, and the end-of-message buffer is tposted to the Buffer Disposition QCB.

The Buffer Disposition subtask performs normally, except that it does not free the line (does not tpost the LCB to itself) until a response has been issued.

When the application program issues a GET macro for the message, the Get Scheduler examines the header prefix in the first buffer and finds the "lock" bit on. This causes the Get Scheduler to set flags that cause the Put Scheduler to treat the first message sent from the application program to the locked terminal as the response message.

The Put Scheduler completes the setting of the lock response flags and sends the message to the terminal destination queue when the application program issues a PUT macro to send the response.

When the Destination Scheduler gets control with the end-of-message buffer, it examines the destination LCB to see if it can be tposted: if so, it tposts the LCB to itself; if not, this indicates that the Buffer Disposition subtask is still processing, and that Buffer Disposition will tpost the LCB to itself. The Destination Scheduler then places the Send Scheduler STCB in the STCB chain of the destination LCB, whether or not it tposted the LCB to itself.

If main-storage-only queuing is being used, the Destination Scheduler places the address of the message header in the lock relative record number (QCBLKRRN) field of the Destination QCB. The message is not placed on the QCB-FEFO chain.

If disk queuing is being used, the Destination Scheduler tposts the end-of-message buffer to CPB Initialization, which places the header address in the QCBLKRRN field of the Destination QCB.

Either the Receive Scheduler or the Send Scheduler gets control when the LCB comes to the top of the ready queue. The scheduler thus getting control examines the LCB to determine whether receiving or sending occurred most recently. The scheduler that was active most recently defers control to the other. In this case, the Send Scheduler will get control, since the most recent operation was a receive. The Send Scheduler will then send the message normally.

After the message is completely sent, the end-of-message buffer is tposted to the Buffer Disposition QCB. If this was a *message* lock function, all indications of the lock have been removed by the Destination Scheduler, and the line is handled normally. If this was an *extended* lock function, Buffer Disposition recognizes that lock mode is still in effect and that a message was just sent, and tposts the LCB to itself.

The Send Scheduler then regains control and passes control to the Receive Scheduler, which polls only the locked terminal. If the response is positive, the station is assumed to be in lock mode and message processing begins for the new message. No FORWARD macro is required for succeeding messages, and the station remains in lock mode until an UNLOCK macro is issued.

Initiate Function

The initiate function is activated during inheader subgroup processing of a message. An INITIATE macro coded in the MH can select either conditional or unconditional execution by examination of a character string in the message header. If the control character string in the message header matches the character string specified in the INITIATE macro, or if the character string is not coded in the INITIATE macro, the initiate function is executed.

The first buffer of the message is processed through the MH to its destination queue, and the INITIATE macro is executed. The buffer is then tposted to the Destination QCB, and the Destination Scheduler gets control and queues the buffer normally. When the first buffer is received, the source LCB is placed on the Destination QCB in-source (QCBINSRC) chain. (The in-source chain is a chain of all source LCBs currently sending initiate mode messages to the destination terminal.)

When the Send Scheduler starts to send the message, it recognizes the presence of initiate mode messages by the presence of a source LCB in the Destination OCB in-source chain. The scheduler removes the source LCB from the Destination QCB in-source chain and places the address of the destination LCB in the in-source chain pointer in the source LCB. If the source LCB is still in the Destination QCB when the end-of-message buffer is received by the Destination Scheduler, the Destination Scheduler removes the LCB and causes the message to be placed in the QCB-FEFO chain of the highest-priority QCB. If transmission has already begun, the message is not placed in the FEFO chain.

When the LCB has been placed in the Destination QCB in-source chain and the destination line has become available, the Send

Scheduler gets the source LCB from the in-source chain, finds the source SCB (via the pointer in the LCB), gets the address of the header, and initializes the destination SCB to send the message. The Send Scheduler begins a normal sending operation and requests the number of buffers specified in the DCBBUFOU field in the destination line DCB for the message by tposting the ERB to the Disk I/O QCB to activate CPB Initialization.

If CPB Initialization has to wait for buffers at any point, it sets flags in the destination LCB indicating that it is waiting for the next buffer of the message. When the next buffer comes in from the source, the Destination Scheduler determines whether CPB Initialization is waiting for buffers; if so, the ERB for the destination line is tposted to the Disk I/O QCB. When CPB Initialization has all the buffers it requires, it continues with normal processing.

No error checking is performed on input data in initiate mode; thus, the first error encountered will be the end of the message. The source station must enter a new message to correct any errors.

SUMMARY OF MESSAGE FLOW

This section contains two charts that present an overview of the flow of control for a message passing through a TCAM system.

Foldout Chart 16 is for a receive operation. When a message is entered at a terminal or from an application program, it is received, processed by the incoming group of the proper MH, and queued onto the message queues data set.

Foldout Chart 17 is for a send operation. When a line or application program is free to receive a message, the message is retrieved from the message queues data set, processed by the outgoing group of the proper MH, and sent to its destination.

Details on each step of these two operations are included under the appropriate heading in the previous parts of this Method of Operation section.

CLOSEDOWN OF A MESSAGE CONTROL PROGRAM

FUNCTIONS OF THE MCP CLOSEDOWN PROCESSING AND CLOSEDOWN COMPLETION ROUTINES

Closedown of the TCAM network is initialized in one of four ways:

1. An operator control HALT command issued from the system console.

2. An operator control HALT command issued from a terminal.

3. An MCPCLOSE macro issued in an application program.

4. A nonreusable disk threshold reached (flush closedown).

In each of the four cases, the effect of the command is the same. The only difference is in the source from which the Operator Control task gains control to load the MCP Closedown Processing routine. If the command is issued from the system console, the operating system posts the ECB for Operator Control. MH posts the ECB for Operator Control if the command is from a terminal. The application program tposts a CIB to the ready queue to cause the Dispatcher to post the ECB when an MCPCLOSE macro is issued. If the EXCP Driver (IGG019RC) recognizes a nonreusable disk threshold, it passes a dummy CIB (defined at AVTHRESE) to the Operator Control task using the same interface as an application program. If TSO is active, the EXCP Driver first branches to the TSO Abend Interface routine (IEDAYT) to allow TSO to end before closedown.

Operator Control loads the MCP Closedown Processing routine, which performs as described on foldout Chart 18.

CLOSE ROUTINES

When all message traffic and TCAM disk operations have completed, control in the MCP returns to the first instruction following the READY macro. This must be the first instruction of a user-written routine to deactivate the MCP, and this deactivation section must issue CLOSE macro instructions for each of the data sets opened in the MCP. The data sets must be closed in the reverse order from which they were opened: first the line group data sets, then the checkpoint data set, and last the message queues data sets.

Foldout Chart 19 illustrates the DCB closedown procedure.

APPLICATION PROGRAM PROCESSING

A TCAM application program is concerned with processing the text portions of messages passing through a TCAM network. Application programs are written by the user to suit the needs of his particular application.

Application programs run asynchronously with the MCP, usually in a different partition or region.

APPLICATION PROGRAM INITIALIZATION AND TERMINATION

Application Program - Initialization Functions

Message transfer from a Destination QCB in the MCP to an application program is controlled by a data control block (DCB) assembled in the application program area. If response messages are generated, transfer from the application program to a Destination QCB in the MCP is handled by a different DCB. The user defines, opens, and closes these DCBs in the application program.

In an application program, a separate DCB is specified for each Destination QCB defined by a TPROCESS macro in the MCP. A DD statement must also be provided for each DCB to associate the DCB with the appropriate Destination QCB.

When an application program is assembled, a DCB macro causes allocation of main storage space for a DCB. Parameters are included based on the specifications of the operands of the macro.

Activation of the interface between an application program and an MCP is accomplished when the application program issues an OPEN macro for each destination queue. The Open Executor issues GETMAIN macros for both a DEB and an access method (ACSMETH) work area for each DCB in the application program area. The OPEN macro expansion activates first Load 1 (IGG01946) and then Load 2 (IGG01947) of the GET/PUT and READ/WRITE Open Executor. The functions of these modules are summarized in foldout Chart 20.

Message Control Program - Initialization Functions

Information necessary for communication between the MCP and an application program is assembled in a control area, a process control block (PCB), defined by a PCB macro in the MCP. There must be one process control block for every active application program in the system.

TPROCESS macros issued in the MCP define the Destination QCBs for application programs. At assembly time each TPROCESS macro creates a process Terminal Table entry for a queue associated with an application program. An operand of a TPROCESS macro specifies the PCB to be used with this particular queue.

When the DCBs are opened in an application program, the Open Executor tposts a special element (RCB) to the ready queue in the MCP. This causes the Open/Close subtask to establish a process entry work area in the MCP. This area contains the Read-ahead QCB and the STCB for the Get Scheduler. The functions of the Open/Close subtask are summarized in foldout Chart 20.

Figure 38 illustrates the linkage among the various control blocks and work areas after the initialization of the MCP and an application program.

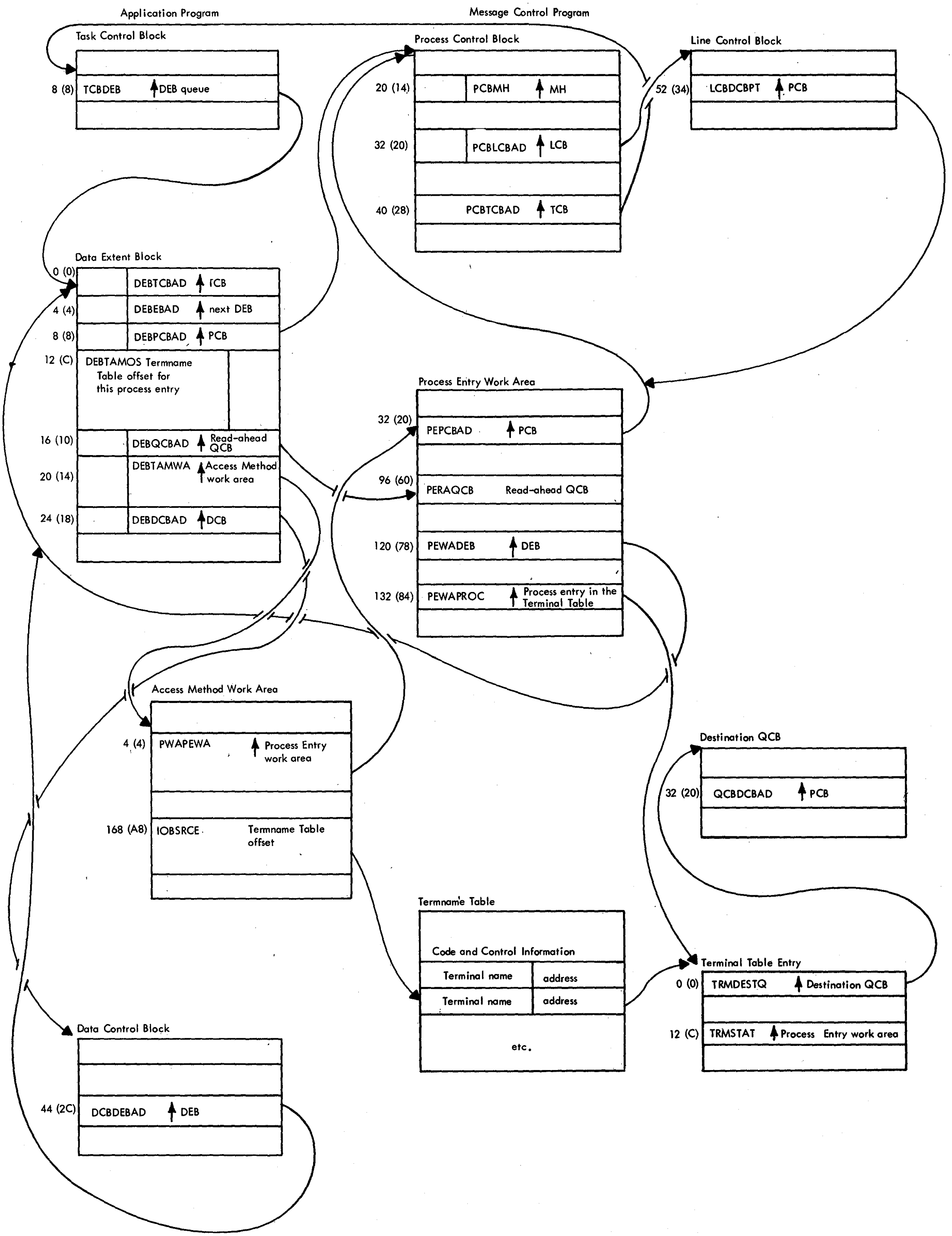


Figure 38. Linkage among Storage Areas in the MCP and an Application Program after Initialization

Application Prcqram - Termination Functions

A CLOSE DCB macro issued in an application program causes the application program Close Executor to gain control. The function of this module is to remove the data transfer communication link between an application program and the MCP.

Foldout Chart 21 illustrates the application program termination functions.

Message Control Program - Termination Functions

The deallocation of application program areas and routines in the MCP is performed by the Open/Close subtask when it is activated by the tposting of an element by the Close Executor in an application program. The close functions of this routine are summarized in foldout Chart 21.

APPLICATION PROGRAM INPUT/OUTPUT FUNCTIONS

Input Functions of an Application Program

The Get Scheduler routine performs a read-ahead function from the message queue in the MCP in anticipation of GET/READ requests from an application program.

The TCAM Dispatcher in the MCP passes control to the Get Scheduler when the STCB for the Get Scheduler is chained on either the Read-ahead QCB or the Destination QCB for the application program. When the Get Scheduler STCB is waiting in the STCB chain of the Read-ahead QCB, the application program has been receiving messages and is either ready to receive more full buffers or is ready to pass empty buffers back to the buffer unit pool. When the Get Scheduler STCB is waiting in the STCB chain of a Destination QCB, it is waiting for a full message to be tposted to the application program, so that it can prepare to pass the buffers of that message to the application program.

Foldout Chart 22 summarizes the flow of control of the Get Scheduler and the GET/READ routine as data is transferred from the MCP to an application program.

Output Functions of an Application Program

The PUT/WRITE routine in an application program initializes the access method work area with parameters so that the Put Scheduler in the MCP can actually move the data from the user application program work area to the MCP.

For a PUT operation, the PUT/WRITE routine refers to the DCB for parameter data; for a WRITE operation, the DECB and DCB are used. If locate mode is being used, the address of the work area is stored in the DEB; otherwise, it is specified by the user as an operand of the PUT or WRITE macro.

After initializing the access method work area, the PUT/WRITE routine activates the Put Scheduler by ‡posting a special element that contains the address of data in the user work area to the QCB for the Put Scheduler in the MCP and by posting the ECB for the MCP complete.

If the application program is eligible for a swap (TSO), the PUT/WRITE routine requests the AQCTL SVC 102 routine to cause the application program task to be flagged not eligible for swap at this time.

If the application program is eligible for rollout (Rollout/Rollin feature), the PUT/WRITE routine requests the AQCTL SVC 102 routine to cause the application program to be flagged not eligible for rollout at this time.

In the situation in which the user specifies PUT or WRITE record without a control byte and with end-of-message indicated by issuing a CLOSE macro, the Open routine for this particular line sets a flag to indicate this condition in the access method work area. After testing this flag during every PUT operation, the Put Scheduler is directed to save the last-filled buffer in the process entry work area, instead of tposting it to the MH. When the next PUT operation is activated, this saved buffer is the first one to be tposted to MH and a new last-filled buffer is saved. The CLOSE macro causes the saved buffer to be tposted to MH as a part of the cleanup procedures.

Foldout Chart 23 demonstrates the functional flow of the PUT/WRITE and the Put Scheduler routines.

MESSAGE RETRIEVAL

TCAM uses a combination of the POINT and the GET or READ macro instructions to support retrieval of messages from a disk message queues data set.

Before issuing a POINT macro in an application program, the user must build an eleven-byte field that contains the following information:

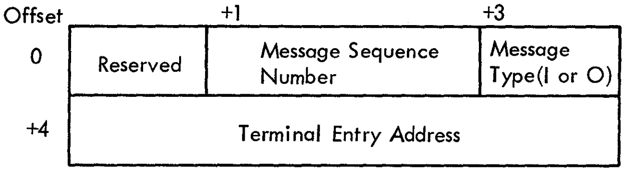
- Bytes 0 - 7: the name of the terminal (left-adjusted and padded with blanks) for which the message to be retrieved is queued.

- Bytes 8 - 9: the two-byte input or output sequence number of the message to be retrieved.

- Byte 10: a character, I or O, designating an input or output message that is queued by source (I) or by destination (O).

After this data field is built, the user issues the POINT and the message fcrm cf the GET cr READ macros; and the Point rcutine, the GET/READ routine, and the Get Scheduler perfcrm the retrieval procedure.

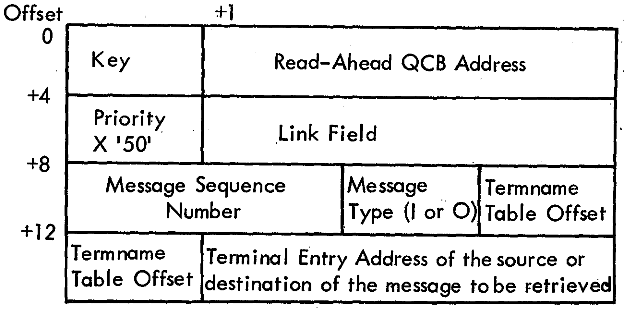
When a POINT macro is in an application program, at assembly time an eight-byte retrieve control block is built at GWARTVE in the access method work area. The format of this ccntrcl block is:



At program executicn time, the POINT macro expansion calls the Point routine, which starts the retrieval process by obtaining the data necessary to complete the fields cf the retrieve control block. The Point routine gets the message sequence number and type from the data field supplied by the user. The routine then scans the Termname Table for the same name as that in the data field - this provides the address of the ccrresponding Terminal Table entry. The Point routine also sets a flag (X'04') in GWAOPTCD in the access method work area to indicate that the application program is in retrieve mode.

If, when the Point rcutine gains ccntrol, the first character of the user-supplied data field is a blank, there is no message to be retrieved. In this case, the routine turns off the "retrieve" bit in GWAOPTCD.

When a GET or READ macro is issued after a POINT macro, the GET/READ routine tests the "retrieve" bit (GWAOPTCD) to determine whether the program is in retrieve mode. (This test is performed only if the routine is at the end of processing a complete message.) If the program is in retrieve mode, the GET/READ routine builds a special retrieve element to be posted to the Get Scheduler in the MCP. The format of this element is:



The GET/READ routine uses AQCTL SVC 102 to place this element on the ready queue in the MCP, and then issues a WAIT to allow time for the specified message buffer to be retrieved.

The Get Scheduler gains control when the special retrieve element has the highest priority of the elements on the ready queue. The element is identified to the Get Scheduler as a retrieve element by the extremely low X'50' priority.

If the ERBBUSY bit (X'80') is on in the process entry work area field PEWAFLG, the ERB for the Get Scheduler is currently tposted and therefore not available to obtain a buffer for the message to be retieved. In this case, the Get Scheduler sets a flag (X'01') in PEWAFLG to indicate that a retrieve element is waiting to be processed. The scheduler then branches to the DSPDISP entry point of the TCAM Dispatcher to allow time for the ERB to be serviced.

When the Get Scheduler regains control, it turns off the ERBBUSY bit, processes the ERB, and then tests the PEWAFLG field for retrieve mode (X'01'). If retrieve mode is indicated, the Get Scheduler turns off the flag just tested and continues processing at the same point at which processing begins when the ERB is not busy.

After the ERB is serviced and back on the Read-ahead QCB, or if the ERB was not busy in the first place, the Get Scheduler moves the current read data from the SCB to the process entry work area in order to set up to read the queue-back chain of buffers from the disk message queues data set. The scheduler gets the appropriate Destination QCB from the terminal entry pointed to by the third word of the retrieve control block in the application program access method work area. The Get Scheduler then moves the queue-back pointer from the Destination QCB (QCBQBACK) to the SCB to identify the first disk record to be read.

In the retrieve situation, the ERB in the process entry work area is serving as a "dummy" or partial LCB. The Get Scheduler sets the "recall" bit in the ERB (LCBRCLNN), initializes the LCBERBCT field to one to indicate that one buffer is to be read, and moves any buffers currently on the element chain of the Read-ahead QCB to the link address chain of that QCB. At this point the SCB is set up to recall a buffer, and the ERB/LCB is partially complete.

The Get Scheduler completes the ERB/LCB by moving in the Read-ahead QCB address, the GET/READ ERB priority of X'D0', and the SCB address. The scheduler then sets the ERBBUSY flag, and tposts the ERB/LCB to the Disk I/O QCB for the message buffer to be read from the message queues data set.

When the message buffer pointed to by the queue-back chain in the Destination QCB has been read from disk, its ERB is tposted to the Read-ahead QCB to reactivate the Get Scheduler.

When the Get Scheduler qains ccntrcl, it tests the "recall" bit (LCBRCLNN) in the ERB to determine whether this is a buffer of the message designated to be retrieved for the requesting application program. At this point, the Get Scheduler tests the message type field in the special retrieve element for I or O, an input or an output message.

If an input message is being retrieved, the Get Scheduler determines whether the buffer just read is the first buffer of a message by examining the PRFSTAT1 field of the buffer prefix. If this field is equal tc X'01', it is the last buffer of a message; otherwise, it is the first buffer. The activity of the Get Scheduler, at this point, depends on the status of this buffer.

- Input message retrieval - first buffer of a message.

The Get Scheduler compares the input sequence number (PRFISEQ) in the buffer to the sequence number in the special retrieve element. If a match is fcund, the scheduler tposts the buffer to the Read-ahead QCB, and tposts the applicaticn prcgram GET/READ ERB back to the Read-ahead QCB to get the rest of the buffers of the message. When the last buffer of the message is read, the "recall" bit at LCBRCLNN is turned off, the SCB is restored to its pre-retrieve status, the Read-ahead QCB is restored, and the scheduler resumes its regular processing.

If the sequence numbers do not match, the Get Scheduler moves the text queue-back chain pointer of the buffer to the SCB (SCBDEOB) and tposts an ERB to read the next message buffer on the input queue-back chain. If this is the first buffer read in the queue-back chain, the scheduler gets the text chain pointer from the buffer prefix field PRFTQBCK. After that, PRFTQBCK is obtained from the process entry work area at PESAVE + 12.

- Input message retrieval - last buffer of a message

If this is the last buffer of a message, the Get Scheduler must get the first buffer of the message in order to compare the input sequence numbers. (The input sequence number for a message is stored only in the prefix of the first buffer of the message.) The Get Scheduler first saves the text queue-back chain pointer (PRFTQBCK) at PESAVE + 12 so that the chain can be searched in order if this is not the correct message. The routine places the first-buffer pointer (PRFCHDR) in SCBDEOB and tpcsts an ERB to read the first buffer of the current message. The Get Scheduler then exits to the DSPDISP entry point of the TCAM Dispatcher.

The Get Scheduler regains control when the first buffer of the message has been read and continues processing by testing the LCBRCLNN bit and by examining the buffers as described in the preceding paragraphs. This loop continues until the specified message is found.

If an output message is being retrieved, the Get Scheduler reads the PRFHQBCK chain until a buffer is found that has the corresponding output sequence number (PRFOSEQ). When the specified buffer is found, the Get Scheduler tposts the buffer to the Read-ahead QCB, posts the application program GET/READ ECB complete, and tposts the ERB back to the Read-ahead QCB to get the rest of the buffers of this message.

When the last buffer of the message is read, the Get Scheduler performs the same functions as described under Retrieval of an Input Message.

When the ECB of the application program GET/READ routine is posted complete, the application program regains control at the first instruction after which the WAIT macro was issued. At this point the GET/READ routine tests the return code in register 15. If the return code has a nonzero value, the message was not retrieved. If the return code is equal to X'00', the message has been retrieved. The application program uses regular GET/READ logic to obtain the rest of the buffers of the message. After all the buffers are read, the program turns off the "retrieve" bit at GWAOPTCD in the access method work area.

COMPATIBLE QTAM

Compatible QTAM GET/PUT Support

When an application program was originally assembled to run with a QTAM MCP and has been reassembled to run with a TCAM MCP, special GET and PUT routines are used. These compatibility versions of GET and PUT contain the internal differences required to process QTAM DCBs. The basic logic of the routine is the same as for the regular GET and PUT routines.

Items that the compatible GET and PUT routines must support are:

- A buffer, or message segment, is a work unit.
- The user must provide the work area prefix.
- The name of the destination must be provided for the user.
- A different format DCB is used.

Compatible QTAM Message Retrieval Support

The Retrieve Service routine and the Retrieve Scheduler provide compatible QTAM support for message retrieval. If there is a QTAM application program operating in the system, the Open/Close subtask loads the Retrieve Scheduler in the MCP. The Retrieve Service routine is called by a RETRIEVE macro expansion in the application program.

The primary difference between message retrieval in TCAM and in compatible QTAM is that in compatible QTAM cnly one buffer at a time is requested. Cne RETRIEVE macro must be issued for each buffer of the message, and the Retrieve Service routine reads the buffer information from the element chain of the Retrieve Scheduler QCB in the MCP.

The RETRIEVE macro expansion puts certain message retrieval data in input registers for the Retrieve Service routine:

- Register 0 - the address of the user work area, which contains the terminal name of the message destinaticn.

- Register 1 - for initial buffer retrieval, the output sequence number for destination retrieval or the input sequence number for source retrieval; for subsecuent buffer retrieval, the disk relative record address.

The Retrieve Service routine uses a special non-register saving entry point of the User Interface routine (IEDQUI) to call the Binary Search routine (IEDQA1) to obtain the Termname Table entry offset for the destination terminal. The Retrieve Service routine then uses this data and the input register data to build a special retrieve element to be tposted to the Retrieve Scheduler QCB in the MCP. The format of this element is:

Offset	+1	
0	Key	Retrieve Scheduler QCB Address
+4	Priority X 'D4'	Link Address
+8	Element Type I,O,0	Terminal Entry Address(Initial Request) or Relative Record Address (Subsequent Request)
+12	Message Sequence Number	Termname Table Offset

If the buffer to be retrieved is the initial buffer of an input message, the Retrieve Service routine places the character I at offset + 8. If the buffer is the initial buffer of an output message, the routine places the character O at that offset. The value X'00' in that field is fcr a subsequent buffer request.

Once the special retrieve element is built, the Retrieve Service routine uses AQCTL SVC 102 to tpost the element to the Retrieve Scheduler QCB in the PCB of the MCP. The Retrieve Service routine then issues a WAIT macro to allow time for the buffer to be retrieved from the message queues data set.

When the special retrieve element gets to the top of the MCP ready queue, the Dispatcher activates the Retrieve Scheduler. The Retrieve Scheduler recoanizes the special retrieve element by its X'D4' priority. If the element type field (offset +8) is equal to zero, I, or O, the scheduler issues a GETMAIN macro for main stcrage for a special LCB and SCB to handle the retrieve function. Failure of the GETMAIN results in a return code of X'04' in the PCB (PCBORC), an ECB post complete, and an exit to the TCAM Dispatcher.

If the GETMAIN is successful, the activity of the Retrieve Scheduler depends on the element type:

- I or O - The Retrieve Scheduler obtains the QCBQBACK pointer from the Destination QCB and places it in the SCB. The scheduler then tposts the ERB in the LCB to the Disk I/O QCB to read the buffer and exits tc the TCAM Dispatcher.
- Zero - If the buffer to be retrieved is a subsequent buffer, the next-text pointer, the current segment address, and the header buffer address are already in the PCB. The Retrieve Scheduler, at this point, sets a subsequent retrieve flag in the PCB (PCBRETVN) and tposts an ERB for the next buffer of the message.

After the ERB request has been satisfied by the disk I/O routines, it is tposted back to the Retrieve Scheduler QCB. The scheduler recognizes the ERB by its X'D0' priority and knows that it now has either an error conditicn or a retrieved buffer to process. An error condition is handled as in regular TCAM processing.

If the retrieved buffer is for an initial request, the Retrieve Scheduler prccesses it just as the Get Scheduler processes a regular TCAM retrieved buffer. The only difference is that the buffer for a compatible QTAM application program is placed on the element chain of the Retrieve Scheduler, not the Read-ahead, QCB.

However, if the "subsequent retrieve" flag (PCBRETVN) is set, the Retrieve Scheduler is handling a subsequent buffer retrieval and performs different functions. It places the next-text pointer of the buffer in the PCB, sets a completicn code of X'00', puts the buffer on the element chain of the Retrieve Scheduler QCB, posts the application program Retrieve ECB complete, issues a FREEMAIN for the LCB and SCB, and exits to the TCAM Dispatcher.

When the Retrieve ECB is posted complete, the Retrieve Service routine in the applicaticn program regains control. At this point the routine tests the return code at PCBCRC. A nonzero return code indicates an error and is passed cn to the user's code. Otherwise, the Retrieve Service routine uses the retrieved buffer to build a QTAM formatted buffer in the user wcrk area. The routine then places a value of X'01' in the element type field of the special element and tposts the element back to the Retrieve Scheduler QCB for buffer return processing.

The Retrieve Scheduler, upon finding the X'01' element type value, tposts the prccessed buffer to the Buffer Return QCB. The scheduler then exits to the TCAM Dispatcher.

FUNCTIONS OF THE NETWORK CONTROL FACILITIES

Interface with Operator Control

The Operator Control/Application Program Interface routine allows the user to perform a subset of the TCAM operator control functions from an application program without actually issuing a PUT for an operator control message.

Foldout Chart 24 illustrates the way that this interface functions.

Network Control with an Application Program

By using the macro instructions TCOPY, ICOPY, or QCOPY, the user can examine the contents of a Terminal Table entry, an invitation list, or a Destination QCB, respectively. Using the macros TCHNG or ICHNG, he can modify the contents of a Terminal Table entry or an invitation list, respectively.

The routines for TCOPY, TCHNG, and QCOPY find the specified entry by locating and scanning the Termname Table. The routine for ICOPY must find the TIOT and DCB to locate the DDNAME for the specified invitation list. An operator control routine (IEDQC1) handles the ICHNG function when the DDNAME and relative line number are supplied by the application program.

If the user wishes to examine the specified entry, the network control routines read the entry directly into the application program work area. However, to write in the MCP partition to change an entry, the AQCTL SVC 102 routine (IGC102) must be used.

Foldout Chart 25 illustrates the functional flow of application program network control.

OPERATOR CONTROL

The Operator Control facility provides a wide variety of functions that allow the user to alter or examine the status of the telecommunications network. Operator control commands can be entered from an Operator Control terminal, an application program, or the system console, and each operator control message must be contained within a single buffer.

Initialization for using the Operator Control facility is accomplished through the operands of the INTRO, TERMINAL, and TPROCESS macros. INTRO specifies the control characters to be used to identify a control message and the specific terminal to be used as the primary control terminal. The TERMINAL and TPROCESS macros associated with the terminals selected as Operator Control terminals have operands to indicate initial specification as secondary control terminals. The TPROCESS macro also specifies an alternate destination, because messages cannct be returned tc an applicaticn program. The values are stored in the AVT.

The Operator Control task is attached in the same partition as the MCP by the Attach routine (IEDQOS) during the execution of the INTRO initialization functions. The Resident Operator Control module (IEDQCA) is the only module that is attached as a resident routine, unless the user specifies that some or all of the Operatcr Control processing routines are to be resident. The Operator Control task has the lowest pricrity of the tasks in the MCP partition.

The Resident Operator Control module loads and activates Load 0 of the Operator Control control module (IGC0010D). There are six loads (IGC0010D, IGC0110D, IGC0210D, IGC0310D, IGC0410D, and IGC0510D) of this control module. Each of these loads is transient, and all except Load 0 are loaded by other loads of the control module as needed to continue decoding or processing an operatcr control command. Load 0 can be activated by OS, by IEDQCA, or by one of the other loads of the control module.

The Operatcr Controcl task, in the form of Load 0 of the control module, is activated when one of its two ECBs is posted. This allows Operator Contrcl to vie with other tasks to be activated by OS Job Management. One ECB is defined in the TCAM AVT and the other in the OS Communications Parameter List. The Operator Control ECB is posted whenever an operator control command (message) is issued. There are three basic types of operator control commands to be handled:

- A standard operator command frcm an Operator Control terminal or an applicaticn program,

- An operator control command from the system console,

- A STARTLN, STCPLN, MRELEASE, RELEASEM, ICHNG, MCPCLCSE, or CLOSEMC command frcm an application prcgram.

Each of these three situations is handled differently by the Operator Control task.

Foldout Chart 26 depicts the functional flow for processing an operator command.

Processing Standard Operatcr Contrcl Commands

When an operator control command is entered frcm an Operator Control terminal or from an application program, it is handled just like any other incoming message until it reaches the CODE macro expansion in the INHDR subgroup. The CODE macro expansion first activates the Translate Buffer routine (IEDQAW) to translate the message to EBCDIC. It then activates the Operator Control Interface routine (IEDQAQ), which compares the acceptable operator contrcl characters in the AVT with the data field referred tc by the scan pointer in the input buffer. If the fields do not match, the buffer does not contain an operator contrcl command, so it is returned tc the next instruction in the MH. If the characters match, the Operator Control Interface routine tposts the buffer to the Operator Ccntrol QCB (AVTOPCOB) by exiting to the DSPPOST entry point of the TCAM Dispatcher.

The interface routine also checks to be sure that the command is complete in one buffer and that the ccmmand was entered by a valid secondary terminal.

When the element (buffer) gets to the top of the ready queue, the TCAM Dispatcher recognizes that it is tposted to a QCB that represents an attached task (the MCPL field of the STCB is equal to X'02'). The TCAM Dispatcher, as a result, issues an CS POST to the ECB for that task. This ECB resides in the seccnd word of the QCB. The element that was on the ready queue, in this case the operator control command, remains cn the element chain of the Operator Contrcl QCB, and the Operator Ccntrol task can begin vying for control cf the system.

When the Operator Control task gains control, Icad 0 of the Operator Contrcl contrcl module is activated. The Operatcr Control control module first processes any commands that are waiting on a special CIB chain (see the following secticns on Processing System Commands and Processing Special Applicaticn Program Commands). After these commands have been processed, the ccntrcl module examines the Operator Contrcl QCB. If there is a command on the element chain of the QCB, the ccntrol module links tc the appropriate Operatcr Control routine to process that command. Upon the completion of the processing routine, it returns to the ccntrcl module. The control module builds the response message to overlay the original command, and returns the buffer tc the MH. The buffer is tposted to the Destination QCB for the source of the command, unless it is a process entry, in which case the control mcdule tposts the buffer to the Destination QCB fcr the alternate destinaticn.

The control module then reexamines the special CIB chain and, if no commands have arrived, checks for another command cn the Operator Control QCB. If there is another command present, it is processed as just described. When all of the commands have been prccessed, the Operator Contrcl control module issues a multiple WAIT on its two ECBs to relinquish control to the operating system. If the closedown switch in the AVT for the MCP is on after all of the commands have been processed, the ccntrcl module issues a RETURN to OS, rather than issuing a WAIT command. A RETURN terminates processing by this task.

Processing System Console Commands

System console cperator control commands are placed in a Command Input Buffer (CIB) and the CIB is chained off the seccnd word of the Communications Parameter List, which is pointed to from the AVT. When a command is issued at the system console, the TCAM Command Scheduler (SVC 34) places it in the CIB and posts the Operator Control ECB, which is pointed to by the first word of the Communications Parameter List. The Communicaticns Parameter List is a two-word field in the OS Control Scheduling Control Block, and it has the following format:

Offset	
0	Address of Communications ECB
+4	Address of First CIB in the CIB Chain

When the Operator Control control module is activated by OS, it examines the second word of the Communications Parameter List to determine whether a system console command is present. If a command is present and if the MCP closedown switch is on or the command is invalid, the control module issues a WTO rejecting the command, issues a QEDIT macro to free and dechain the CIB, and branches back to examine the CIB chain for another command. If the command is valid, the control module links to the appropriate operator control processing routine. Upon completion of the processing routine, the control module sends a WTO response message, issues a QEDIT macro to dechain and free the CIB, and branches back to check for another command.

After all the commands on the CIB chain and on the element chain of the Operator Control QCB have been processed, the control module issues a multiple WAIT on its two ECBs. However, if the MCP closedown switch is on, the control module does not issue a WAIT, but terminates processing of this task by issuing a RETURN directly to the operating system.

Processing Special Application Program Commands

The operator control commands STARTLN, STOPLN, MRELEASE, RELEASEM, ICHNG, MCPCLOSE, and CLOSEMC, which are issued from an application program, are processed in a slightly different way than other operator control commands. When one of these commands is issued, the Operator Control/Application Program Interface routine in the application program gains control. This routine builds a "dummy" CIB that contains the type of command issued and other pertinent data. The Interface routine uses the AQCTL SVC 102 routine to tpost this CIB to the Operator Control QCB. A WAIT is then issued to place the application program in a wait state and to allow the MCP to begin processing.

When the CIB tposted to the Operator Control QCB reaches the top of the MCP ready queue, the TCAM Dispatcher recognizes it as an element for an attached task. The TCAM Dispatcher acts exactly as it does when a standard command is on the ready queue: it posts the ECB for the attached task so that the task can vie for control of the system in order to process its element.

When the Operator Control task gains control, it processes the CIB exactly as it does a regular message on the Operator Control QCB, except that the response message built consists of the CIB with a return code added. Since the CIB is from an application program, the control module also posts the ECB for the application program complete.

Operator Contrcl/Checkpoint Interface

Each time an operator control command causes any change in the AVT, Terminal Table, Option Table, LCB, or invitation lists, the Operator Control task determines whether a checkpcint shculd be taken. If it should, the Operator Control control module tposts the "operator control checkpcint request" element to the Checkpoint QCB and issues a WAIT on the Cperator Control ECE in the AVT. A checkpcint needs to be taken if the checkpoint data set was cpened (ncnzero value in AVTCKGET). A checkpoint is needed for a change in invitation lists only if "I" was specified on the INTRO STARTUP operand.

When the Checkpoint Executor gains control, it loads the Incident Checkpoint for Operator Ccntrol routine. The Incident Checkpoint for Operator Control routine moves control infcrmation for the command from the operator control work area to the incident checkpoint record. After the record is written on disk, the Checkpcint Executor posts the Operator Contrcl ECB complete.

At restart time the abcve process is reversed - the Checkpoint/Restart from Incident and CKREQ Records routine (IGG01944) moves the contrcl data from the incident record (except start/stop line records) tc the operatcr control work area, posts the Operator Control ECB, and issues a WAIT macro. Operator Control examines the "ready complete" bit in the AVT, recognizes the restart situation, reprocesses the operator ccntrol ccmmand, and posts the Checkpoint ECB complete.

Operator Contrcl Processing Routines

The functions of the operator contrcl processing routines are explained under the description of each routine in the Program Organization section of this publication.

CHECKPOINT

The TCAM Checkpoint facility provides for records to be taken of the MCP environment from which restart can be made in the case of closedown or system failure. Records of individual data paths are maintained to preserve the integrity and ccntinuity of message flow to and from a terminal (cr component). Only the last message entered from or accepted at a buffered terminal may need to be resent to make sure no message is lost (fcr ncnbuffered terminals, at most one message per line may need to be resent). Checkpoint records are maintained for all main storage queues that have full copies on disk (if main-storage-only queues are used, no checkpoints are taken of message queues).

· There are four types of checkpcint records: the contrcl record, environment records, incident records, and CKREQ records. The control record contains information concerning the format of the checkpoint data set. Environment records are concerned with checkpoints for the total operating environment; incident records and CKREQ records are concerned with checkpoints of specific incidents during operation. The TCAM restart procedure uses the incident and CKREQ records to update the TCAM environment from the time that it was recorded by the most recent complete environment record to the time of system closedown or failure.

The Checkpoint Executor manages the routines that write checkpoint records. The Checkpoint Executor gains control when a checkpoint request element is placed on the ready queue in the MCP.

The TCAM Dispatcher, upon finding a checkpoint request element on the ready queue, chains the element off the element chain of the Checkpoint QCB in the AVT. The ECB for the Checkpoint subtask is then posted complete, and when an OS WAIT command is issued, the Checkpoint Executor can gain control.

FUNCTION OF THE CHECKPOINT EXECUTOR

The Checkpoint Executor causes all of the checkpoint request elements chained off the element chain of the Checkpoint QCB in the AVT to be processed. This routine continues processing until all the elements are processed or until a task with a higher priority seizes control after an interruption.

The Checkpoint Executor first examines the ECB for the Checkpoint Disk I/O routine to determine whether an I/O operation has been completed. If an I/O operation has been completed, the Executor transfers control to the Checkpoint Notification and Disposition routine.

If an I/O operation has not been completed, the Checkpoint Executor examines the checkpoint disk I/O queue to see if there is a record ready to be written. If there is a record ready to be written, the Executor transfers control to the Checkpoint Disk I/O routine.

If no I/O operation has been completed and there is not a record on the checkpoint disk I/O queue, the Checkpoint Executor examines the key field of the first "checkpoint request" element on the element chain of the Checkpoint OCB. The value of this key field determines which of the following checkpoint routines will be loaded to process the element.

- The Environment Checkpoint routine
- The Incident Checkpoint for MH routine
- The Incident Checkpoint for TCHNG routine
- The Incident Checkpoint for Operator Control routine
- The Build CKREQ Disk Record routine

After a checkpoint rcutine has completed its prccessinq, it returns to the Checkpoint Executor in one of two ways. The returning routine can indicate that the Checkpoint Executor is tc continue processinq as normal, or it can indicate that the Checkpcint Executor is to immediately load and activate the rcutine indicated by an offset value returned in reqister 15.

The Checkpoint Executor determines the name of the routine to be loaded by usinq an offset intc a table of names stored in the Checkpoint Executcr module.

Foldout Chart 27 illustrates the functicnal flow cf the checkpoint routines. The function of the Checkpoint Executor is not included since it is primarily a control module for the routines that build and write the checkpcint disk records.

THE ENVIRONMENT CHECKPCINT ROUTINES

Environment checkpoint records include disk queuing pointers, sequence numbers, terminal status, invitation list status (if specified), DCB information, line status, and Terminal Table cpticn fields. Checkpoints are taken on all informaticn that can be altered by Operator Ccntrcl commands cr application prcqram macros.

Environment checkpoint records are taken at specific points during the execution cf the MCP:

- At the beqinninq of execution (frcm READY).
- When the incident checkpcint area is full.
- At zone chanqe-overs when usinq any reusable queues.
- After a user-specified time interval. When any total checkpoint is taken, the interval is reset for the time interval checkpoint.
- Durinq any MCP closedown.

There are at least two environment records on disk; the number is provided by the user in an operand of INTRO. The records are used alternately and the control record contains an indication of the most recent environment record.

The Environment Checkpoint routine qains ccntrol from the Checkpoint Executcr when the checkpcint request element was issued either by READY, by the Reusability-Copy subtask, by the Time Delay subtask, by an MCPCLOSE macro in an applicaticn program, or by a HALT command from the system console or a terminal.

The Environment Checkpoint routine first issues a GETMAIN macro to obtain main storaqe space in which to build an environment checkpoint record. The routine builds one seqment of the record in this area and then returns tc the Checkpoint Executor with the offset of the Checkpoint Oueue Manaqer in reqister 15. This causes the Checkpoint Executor to immediately activate the Checkpcint Queue Manaqer.

If the GETMAIN issued by the Envircnment Checkpoint routine is not satisfied , the Environment Checkpoint routine returns to the Checkpoint Executor with the offset of the Checkpoint - No Available Core routine in reqister 15. The No Available Core rcutine first tests to determine whether any previous GETMAIN has been issued that is not yet free. If there is one, the routine returns to the Checkpoint Fxecutor to allow time for that area to be freed. If there is not an outstandinq GETMAIN area, the No Available Core routine issues a WTO error messaqe to indicate the situation and turns on the hiqh-order bit of the "last element for which a disk record was built" field in the checkpoint work area. The Nc Available Core routine then returns to the Checkpcint Executor with the offset for the Checkpoint Notification and Dispcsiticn rcutine in reqister 15. In this situation, the Checkpoint Notification and Disposition routine removes this unsatisfied checkpoint request element from the Checkpcint QCB.

The Checkpoint Oueue Manaqer is activated after an initial disk record has been built. This routine places a pcinter to the record on the checkpoint disk I/O queue. If the record put on the checkpoint disk I/O queue is an environment checkpoint record, the Checkpoint Oueue Manaqer frees any incident records already cn the checkpoint disk I/O queue and turns on the "incident overflow" bit in any incident checkpoint request element cn the Checkpoint QCB. (When the last record of an environment checkpoint has been written cn disk, the Checkpoint Notificaticn and Dispositicn rcutine removes any element with its "incident overflow" bit cn from the Checkpoint QCB, since its request was satisfied by the last envircnment checkpoint record.) The Checkpoint Queue Manaqer then returns to the Checkpcint Executor.

When the Checkpoint Executor reqains ccntrol, it beqins execution by examininq the ECB of the Checkpcint Disk I/O routine. If a disk I/O operation has not been completed and there is a record on the checkpoint disk I/O oueue, control is passed to the Checkpoint Disk I/O routine. This routine takes the record off the checkpoint disk I/O queue, builds CCWs and an IOB, and issues an EXCP tc write the record on disk. The main storaqe address of the record is placed in the current-EXCP field in the checkpoint wcrk area. The Checkpoint Disk I/O routine then returns to the Checkpcint Executor.

When the channel has finished writinq a record on disk, an I/O interruption qives contrcl to the Checkpcint Disk End Appendaqe. The Checkpoint Disk End Appendaqe examines the key field of the record just written (found via the write CCW) and if the record was the last seqment of an environment checkpoint, rewrites the checkpcint control record. The checkpoint control record indicates the last complete environment checkpoint taken. An updated copy of the control record is in the checkpoint work area. The Checkpcint Disk End Appendaqe returns contrcl to the operatinq system.

If, upon regaining control, the Checkpoint Executor finds that a disk I/O operation has been completed, it passes control to the Checkpoint Notification and Disposition routine. This routine examines the key field of the disk record just written (found via the current-EXCP field of the checkpoint work area). If the key field indicates that the record was the last record of an environment checkpoint operation, the routine frees the record area (via a FREEMAIN macro), zeros the current-EXCP field, and turns off the checkpoint request bits in the AVT. If the environment checkpoint request is from the MCP Closedown Processing routine, the Checkpoint Notification and Disposition routine tposts the "closedown completion request" element to the ready queue; otherwise, the Checkpoint Notification and Disposition routine removes the "environment checkpoint request" element from the Checkpoint QCB and tposts it to the time delay queue. If the checkpoint was not complete (not the last record), the Checkpoint Notification and Disposition routine returns to the Checkpoint Executor with the offset for the Environment Checkpoint routine in register 15.

When the Environment Checkpoint routine builds a disk record, it saves data necessary to build subsequent records for the same checkpoint in the checkpoint work area. As a result, when the Notification and Disposition routine instructs the Checkpoint Executor to return control to the Environment Checkpoint routine, this routine can resume building the next record. The address of the first record built is stored in the current-EXCP field of the checkpoint work area. The Environment Checkpoint routine obtains this address and builds the new disk record over the old one. If the new record is the last segment of the checkpoint, the key field indicator is set to X'1C'. If the new record is a continued segment, the key field indicator is set to X'20'. This routine then returns to the Checkpoint Executor with the offset of the Checkpoint Disk I/O routine in register 15.

Note that after the Environment Checkpoint routine builds the initial record of a checkpoint, it indicates that control is to be passed to the Checkpoint Queue Manager to have the record placed on the checkpoint disk I/O queue. Since each subsequent record of a checkpoint is built in the same main storage area as the first record, and since this main storage address is saved in the current-EXCP field of the checkpoint work area, the Checkpoint Disk I/O routine has all the information it needs to immediately issue an EXCP for a record. Subsequent records do not need to be placed on the checkpoint disk I/O queue.

Foldout Chart 28 illustrates the flow of control among the checkpoint routines as an environment checkpoint is taken.

THE INCIDENT CHECKPOINT ROUTINES

Incident checkpoint records are taken as a result of MH macro instructions, application program macros, and operator control modules that effect changes in the MCP environment. The records contain only the data for that change. Incident checkpoints record changes in terminal status, invitation lists (if specified), Terminal Table option fields, polling intervals, the primary Operator Control terminal, and the TCAM Trace facility.

There are three incident checkpoint routines. Each one gains control as a result of a specific macro or command having been issued in the TCAM system:

- Incident Checkpoint for MH routine - gains control when functions of the CHECKPT macro are executed in an MCP.

- Incident Checkpoint for TCHNG routine - gains control when a TCHNG macro is issued in an application program.

- Incident Checkpoint for Operator Control routine - gains control when a VARY, MODIFY, HOLD, or RELEASE command is issued from the system console or from a terminal; or when an ICHNG or MRELEASE macro is issued in an application program.

A count of the number of available incident checkpoint records on the disk is kept in the checkpoint work area. Each time one of the incident checkpoint routines builds a record, it subtracts one from the available incident disk records count. If the count is equal to zero when the incident routine is ready to decrement it, the incident routine immediately returns to the Checkpoint Executor with the offset to the Checkpoint - No Incident Records routine in register 15.

The function of the No Incident Records routine is to cause an environment checkpoint to be taken so that the incident records area on the disk can be reused. The No Incident Records routine determines the status of the "environment checkpoint request" element by examining its key field in the AVT. If the element is already on the Checkpoint QCB, it is moved to the "next request element to be serviced" position on the element chain of the QCB (a field in the checkpoint work area points to the last element for which a disk record has been built on the Checkpoint QCB). If the "environment checkpoint request" element is not on the Checkpoint QCB, it is removed from the Time Delay QCB and placed in the "next element to be serviced" position in the element chain of the Checkpoint QCB. The No Incident Records routine then returns to the Checkpoint Executor so that it can, in normal processing procedures, give control to the Environment Checkpoint routine to service the element just placed on the Checkpoint QCB.

An LCB serves as a "checkpoint request" element for the MH. The LCB is placed on the ready queue by the Buffer Disposition subtask. The address of the Checkpoint QCB is in the last three bytes of the first word of the LCB, so the TCAM Dispatcher places the LCB, acting as an "MH checkpoint request" element, on the element chain of the Checkpoint QCB.

An "application program checkpoint request" element is physically located in the PCB for that application program. The TCAM Dispatcher places a pointer to this checkpoint request element on the element chain of the Checkpoint OCB after the Dispatcher gains control from the Application Program/Checkpoint Interface routine. A code for the specific macro requesting the checkpoint is in the key field of the element.

An "operator control checkpoint request" element is physically located in the AVT. The TCAM Dispatcher places a pointer to this checkpoint request element on the element chain of the Checkpoint QCB after it gains control from an operator control routine. The key field of the element indicates whether the checkpoint was requested by an operator control command.

Incident Checkpoint for MH: The Incident Checkpoint for MH routine builds an incident checkpoint record in the buffer just processed by the MH routines. The incident checkpoint record is then processed as described in The Environment Checkpoint Routines section with two exceptions:

- The Checkpoint Disk I/O routine obtains the actual disk address for the record by examining a field in the checkpoint work area that contains the track and record number of the last incident checkpoint record written on disk.

- The Checkpoint Notification and Disposition routine frees the LCB by tposting it to the Buffer Disposition QCB for the Chain routine.

Incident Checkpoint for TCHNG: The Incident Checkpoint for TCHNG routine issues a GETMAIN macro for an area in which to build an incident checkpoint record. The incident checkpoint record is then processed as described in The Environment Checkpoint Routines section, with two exceptions:

- The Checkpoint Disk I/O routine obtains the actual disk address for the record by examining a field in the checkpoint work area that contains the track and record number of the last incident checkpoint record written on disk.

- The Checkpoint Notification and Disposition routine must post the application program ECB complete.

Incident Checkpoint for Operator Control: The Incident Checkpoint for Operator Control routine issues a GETMAIN macro for an area in which to build an incident checkpoint record. The incident checkpoint record is then processed as described in The Environment Checkpoint Routines section, with two exceptions:

- The Checkpoint Disk I/O routine obtains the actual disk address for the record by examining a field in the checkpoint work area that contains the track and record number of the last incident checkpoint record written on disk.

- The Checkpoint Notification and Disposition routine must post the operator control task ECB complete.

THE CKREQ CHECKPOINT ROUTINES

CKREQ checkpoint records are taken as a result of a CKREQ macro issued in an application program. There is one record built for each open Destination QCB associated with the application program that is issuing the CKREQ macro. The restart procedure uses each record during a restart to update the environment checkpoint records.

The Build CKREQ Disk Record routine issues a GETMAIN macro for an area in which to build a CKREQ record. The CKREQ record is then processed as described in The Environment Checkpoint Routines section, with three exceptions:

- The Checkpoint Disk I/O routine obtains the actual disk address for the record by using from the checkpoint work area the table name offset that associates terminal name offsets with track and record number addresses. (There is one CKREQ record on disk for each destination associated with the application program issuing a CKREQ macro.)
- If this checkpoint requires more than one disk record, the Notification and Disposition routine returns to the Checkpoint Executor with the offset of the Build CKREQ Disk Record routine, rather than the offset of the Environment Checkpoint routine, in register 15.
- The Checkpoint Notification and Disposition routine must post the application program ECB complete.

ERROR RECOVERY PROCEDURES

The TCAM error recovery procedures (ERPs) consist of fifteen modules that operate in the nucleus error transient area under the supervisor protection key. If the TCAM Line End Appendage (IGG019R0, IGG019Q2, IGG019Q3, IGG019Q4 or IGG019Q5) detects an error status on a telecommunications device, it returns to the I/O Supervisor indicating that control is to be passed to ERP. The I/O Supervisor (IOS) gives control to either the Start/Stop ERP Control module (IGE0004G) or the BSC ERP Control module (IGE0004H). The ERP control modules analyze the error and transfer control to another module to handle the error.

The Start/Stop ERP Control module can link to any of the following ERP processing modules:

IGE0104G	Read/Write Unit Check and Unit Exception ERP Module
IGE0204G	Non-Operational Control Unit ERP Module
IGE0304G	Unit Check for Non-read, Non-write, and Non-poll CCWs ERP Module
IGE0404G	Auto Poll and Read Response to Poll Unit Check and Unit Exception ERP Module

IGE0504G	Error Post and Second Level CCW Return Module
IGE0604G	Unit Check and Unit Exception on Read/Write CCWs for Audio and 2260 Local Devices ERP Module
IGE0804G	Start/Stop Channel Check ERP Module
IGE0904G	Closedown Terminal Statistics Recording Module

The BSC ERP Control module can directly activate any of the following ERP processing modules:

IGE0104H	BSC Read/Write Equipment Check, Lost Data, Intervention Required, and Unit Exception ERP Module
IGE0204H	BSC Read/Write Data Check, Overrun, and Command Reject ERP Module
IGE0404H	BSC Second Level CCW Return Module
IGE0504H	BSC Error Post Module
IGE0804H	BSC Channel Check ERP Module
IGE0204G	Non-operational Control Unit ERP Module
IGE0304G	Unit Check for Non-read, Non-write, and Non-poll CCWs ERP Module
IGE0404G	Auto Poll and Read Response to Poll Unit Check and Unit Exception ERP Module
IGE0904G	Closedown Terminal Statistics Recording Module

The Start/Stop Error Post and CCW Return module (IGE0504G) and the BSC Error Post module (IGE0504H) are activated by certain ERP processing modules.

In addition to the linkages illustrated in foldout Charts 29 and 30, each module may exit using the following SVC sequence:

SVC	15	Error EXCP
SVC	3	Return - free the transient area

When the SVC 15 is issued, IOS acts according to the flag setting in LCBFLAG1:

X'24' -	Retry by IOS and return control directly to ERP after the interrupt
X'04' -	Transfer control to the abnormal line end appendage
X'00' -	Transfer control to the normal line end appendage

In TCAM, both the normal and abnormal line end appendage addresses point to IGG019R0, IGG019Q2, IGG019Q3, or IGG019Q4.

Linkage between the modules is performed by IOS through the XCTL routine with a branch on register 14. The last four digits of the module name are placed in register 13, and the address of the XCTL routine is placed in register 14. The linkage between the start-stop modules is shown in foldout Chart 29 and linkage between the BSC modules is shown in foldout Chart 30.

There is a description of each of the ERP modules in the Program Organization section of this publication. The descriptions explain the action taken according to the different types of commands.

Generally, if there has been no text transfer, the channel program is retried. If there is an error after two retries for start-stop or six retries for BSC, the error is considered permanent. In the case of a permanent error, a message is either written to the system console or scheduled to be sent to the Operator Control terminal.

For conditions that should not happen, the "should not occur" bit (bit 7) is set in the SCB. This condition is considered to be a permanent error.

When there has been an error on a Read Response to Auto Poll, the invitation list address and entry size are obtained. The invitation list is searched for an equal comparison on the index byte. If no match is found, the channel program is restarted with the existing Poll CCW. If there is an equal comparison, the address of the matching entry is used, and the count is set to the new count plus the initial address minus the address of the matching entry.

When there is an error on the Poll CCW, the polling list address and entry size are obtained. The count is set to the residual count plus the width of the poll characters. The data address is the poll list address and original count minus the new count.

MESSAGE HANDLING WITH TIME SHARING OPTION SUPPORT

TSO Line Management Support

In order to implement line management in TSO support, TCAM uses its Receive and Send Schedulers. When a scheduler is dealing with a terminal that is dedicated to a time sharing session, the scheduler branches to a TSO routine (the Time Sharing Scheduler), which performs special checking functions.

If a receive interrupt has just occurred on input, the Time Sharing Scheduler tposts the LCB to the TSO Attention routine. No input operation is initiated if there are insufficient TSO buffers or when there is output to send. If the terminal in question does not have the transmit interrupt feature, no read channel program is built until after a GET has been issued from the TSO foreground program; otherwise, the Receive Scheduler performs a read-ahead operation. If no input operation is to be initiated, the scheduler determines whether to start a simulated attention channel program or to place the OCB in the time delay queue for a simulated attention by time interval. If no other I/O is to be started on the line and the terminal has the receive interrupt feature, the scheduler places a prepare on the line to monitor for receive interrupts (attentions).

The Send Scheduler is activated and an output operation is initiated when the TPUT SVC tposts the Destination QCB to itself on the disabled ready queue. The Send Scheduler branches to the Time Sharing Scheduler, which determines whether a TPUT with the break option is requested. If so and if an input operation is in progress on a terminal with the transmit interrupt feature, the Time Sharing Scheduler halts the I/O operation. When the interrupt occurs, the Line End Appendage builds a break CCW to stop terminal transmission. When the Send Scheduler is dispatched from the LCB, the Time Sharing Scheduler determines whether a simulation attention or a read operation has priority over output. If so, the Time Sharing Scheduler takes steps to initiate the appropriate operation. For a send operation, the Send Scheduler tposts the ERB to the TSOUTPUT routine, which moves the data from the TSO buffer in the TSO partition into the TCAM buffers in the TCAM region.

When the TCAM Activate subtask gains control, it also branches to the Time Sharing Scheduler. This module, in turn, determines whether a receive interrupt has occurred or an output operation has been requested. If a receive interrupt has occurred, the Time Sharing Scheduler frees the buffers that were acquired for input, tposts the LCB to the TSO Attention routine, and exits to the Dispatcher. If an output operation has been requested, the Time Sharing Scheduler frees the buffers, tposts the LCB to itself, and exits to the Dispatcher. If there is a prepare on the line to monitor for a receive interrupt, the Time Sharing Scheduler issues a TCAM HALT I/O. The scheduler then returns to Activate.

The TCAM Line End Appendage supports recognizing receive interrupts, issuing transmit interrupts, recognizing hangups on a dial line, and identifying 2741s and 1050s on the same line. The Line End Appendage handles a negative poll response on a leased line by branching to the Time Sharing Scheduler.

When a line that is dedicated to a time-sharing session is to be freed, the QEVENT routine branches to the Time Sharing Scheduler to place a prepare on the line to monitor for receive interrupts to terminals with the receive interrupt feature.

TSO BUFFER MANAGEMENT SUPPORT

Each TSO buffer has a 21-byte buffer prefix. Line buffering is the same as in TCAM, except that for a send operation the Send Scheduler tposts the ERB to the TSOUTPUT routine, not to CPB Initialization. The TSOUTPUT routine builds TCAM buffers and moves data from the TSO buffers in the TSO region into the TCAM buffers in the TCAM region.

There is a special STARTMH subtask for TSC support in TCAM. This subtask performs the same types of furctions as the regular TCAM STARTMH subtask. In addition, it does nct set aside any reserve characters in buffers supporting time-sharing sessicns. If a buffer with data for a main storage or disk message queues data set is routed to a TSO MH, the STARTMH subtask routes the message to a different MH, if one is specified; otherwise, the subtask cancels the message.

The TSO Logon routine, as called from the INHDR subgroup of a TSO MH, scans the first buffer for a time-sharing session tc determine whether to initialize for the session. If not, the buffer is routed to another MH, if specified, or canceled.

The TSO Carriage routine, as called frcm the INBUF subgroup of a TSO MH, keeps track of the carriage positicr of the entering terminal and removes line control characters from the incoming message. The TSO Simulated Attention routine scans the input buffers for a simulated character string, if this function is requested.

In the INMSG and OUTMSG subgroups of a TSO MH, the TSC Attention routine processes receive interrupts. The TSO Hangup routine determines acticns based on hardware errors.

The Buffer Disposition subtask tposts incoming TSO buffers to the OCB of the TSINPUT routine, as opposed to the Disk I/O QCB.

TSO QUEUE MANAGEMENT SUPPORT

In a TCAM MCP, the TSO queuing and destinaticn assignment functions are handled by the TSINPUT and TSOUTPUT rcutines. For an incoming message, Buffer Disposition tposts the line buffers to the TSINPUT routine. The TSINPUT routine uses the QTIP SVC to move data from TCAM buffers in the TCAM region to TSO buffers in the TSO region. The routine then frees the TCAM buffers by tposting them to the Buffer Return OCB. When the TSINPUT routine gets the last buffer of a message, it flags complete all the TSO buffers associated with this message. If a CANCELMG macro has been executed, the TSO buffers are freed.

When a TSC foreground program issues a GET macro, the TGET SVC moves data into the prcgram wcrk area. An input editing routine performs any data editing that is requested on operands of the GET macro.

For an outgoing message, the TSO fcregrcund program issues a PUT macro, which causes the TPUT SVC to move data from the prcgram work area into TSC buffers. If no output operation to the terminal is in progress, the TPUT routine also disables itself and tpcsts the Destination QCB to itself on the disabled ready queue. When the Send Scheduler is ultimately dispatched off the LCB, it tposts the ERB to the TSOUTPUT routine. The TSOUTPUT routine obtains TCAM buffers and moves the data from the TSO buffers in the TSO region into the TCAM buffers in the TCAM region. The Send Scheduler uses the TSO TIOC Edit routine to perform any editing functions requested by operands of the

PUT macro in the TSO foreqround prcqram. When the last buffer of the messaqe has been sent, Buffer Disposition tposts the buffer back to the TSOUTPUT rcutine, which frees the associated TSO buffers.

TSO MCP CLCSEDCWN PROCESSING SUPPORT

If a TSO proqram is usinq TCAM for terminal support, all TCAM requests for closedown are iqnored until TSO is nc longer operatinq in the system. This is done because TSO is not desiqned to continue processinq after TCAM is closed down.

SYSTEM SERVICE ROUTINE

Disk Message Queue Initializer (Chart XA)

Module Name: IEDQXA

Entry Point: IEDQXA

Function: This routine is a utility program used to build a formatted disk data set. The data set can then be used by a TCAM MCP to contain either a reusable or nonreusable disk message queues data set. Before a TCAM MCP is loaded into the system, the Disk Message Queue Initializer must be run as a separate job step for every disk message queues data set specified in the MCP.

Input to this routine is supplied by the Job Control Language (JCL) parameters for executing the job step. Sample JCL for the Disk Message Queue Initializer is as follows:

```
//jobname  JOB
//stepname EXEC PGM=IEDQXA
//SYSPRINT DD SYSOUT=A,SPACE=(TRK,(1,1))
//IEDQDATA DD DSNAME=anyname,DISP=(,CATLG),        *
//            SPACE=(CYL,(n,n),,CONTIG),           *
//            UNIT=(23xx,y),                       *
//            VOLUME=SER=(aaaaaa,bbbbbb,...),      *
//            DCB=(,KEYLEN=mm)
```

The variables are defined as follows:

anyname - the user selects a name for the data set. This same name is used in the JCL for a TCAM job to define the use of this data set.

n - the number of cylinders must be the same for all extents. Primary and secondary allocations must be identical, and allocation must be by cylinders.

xx - 11 or 14. Any one data set must have all extents on one type of disk.

y - 1 to 16. The total number of volume serial numbers listed in the "VOLUME" parameter.

aaaaaa,bbbbbb,... - each volume serial number of each volume to contain an extent of the data set. There is one extent per volume, with a maximum of 16 volumes.

mm - the size of the key portion of the disk data records to be written. The maximum key size is 255 bytes; the minimum is 33. "mm" is the same value as the "unit" size specified in the "KEYLEN" keyword of an MCP INTRO macro. The data field length is an internally fixed constant of 6 bytes. This is added to the "KEYLEN" value to obtain "BLKSIZE".

The two required DD cards define the two output data sets. The SYSPRINT data set contains a copy of typewriter messages, and has the attributes of DCB=(RECFM=U,BLKSIZE=80). This data set may be suppressed by specifying //SYSPRINT DD DUMMY.

The IEDQDATA data set is the new data set to be created as a TCAM message queue. The required keywords are shown in the sample JCL above.

The execution of the Disk Message Queue Initializer is in two phases: the open and verification phase, and the formatting phase.

During the open and verification phase, the Disk Message Queue Initializer checks the JCL variables to determine whether they are defined according to specifications. If any exception is found, the routine terminates with a diagnostic statement defining the problem. Messages that can be generated for the SYSPRINT data set and the system console are as follows:

- "IED066I UNABLE TO OPEN SYSPRINT" - Return code 20. A SYSPRINT DD card must be present. (Console only.)

- "IED067I TCAM INITIALIZATION BEGUN" - The SYSPRINT data set is opened. Processing continues.

- "IED068I UNABLE TO OPEN IEDQDATA" - Return code 20. An IEDQDATA DD card must be present.

- "IED069I INVALID KEYLEN FOR IEDQDATA" - Return code 8. The KEYLEN parameter is either missing or not within acceptable limits.

- "IED070I IEDQDATA DOES NOT SPECIFY CONTIG SPACE IN CYLINDERS" - Return code 16. SPACE must specify CYL and CONTIG.

- "IED071I UNEQUAL PRIMARY AND SECONDARY EXTENTS ON IEDQDATA" - Return code 16. Primary and secondary extents sizes must be identical.

In the formatting phase of the routine there is a loop built around a WRITE macro that writes a zero-filled record on the disk. Each formatted record contains count, key, and data fields. The count field has a CCHHR absolute address; the key and data fields are areas to receive TCAM message header and text information.

After the routine fills the extent of a volume with records, it checks the field that indicates the number of specified volumes to determine if that was the last volume to be filled. If it is not the last volume, the routine issues an FEOV macro to cause secondary allocation tc be made from the next volume in the list; in this way, each volume has only one extent. After the routine formats the last volume, it issues a successful return to OS Job Manaqement.

At the end of each volume the initializer issues a statement to the SYSPRINT data set and to the system console. This statement contains the total record count frcm the beqinning of the data set through the volume just completed. The number of these statements is the number of extents (or volumes) successfully formatted.

Messaqes that can be qenerated during the second phase of execution are as follows:

- "IED072I I/C ERROR ON IEDQDATA" - Return code 12. Unable to recover frcm a disk I/O error cn the disk messaqe queues data set.

- "IED073I I/C ERROR ON SYSPRINT" - Return code 4. (Consele only.) Unable to recover from an I/O error on the SYSPRINT data set.

- "IED074I TCAM INITIALIZATION COMPLETE" - Return code 0. Successful completion.

- "IED075I END OF EXTENT. RECORD COUNT IS number" - The total record count up through the current extent. This statement appears at the end of fcrmatting each vclume. The final count is for the entire data set.

External Routines:

- SVC 64 - reads JFCB of the IEDQDATA data set for JCL verification.

- BSAM - writes dummy records to the disk using WRITE, CHECK, OPEN, CLOSE, and DCE macros.

- OS WTO routine (SVC 35) - handles output to the system console.

- OS Getmain routine (SVC 4) - cbtains main storaqe fcr a buffer work area.

- OS Freemain routine (SVC 5) - frees main storaqe.

- SVC 31 - shifts from one volume to another.

Tables/Work Areas: The JFCB is read into a local constant area. The output buffer is a GETMAIN area initialized tc zero.

Attributes: Reusable.

INITIALIZATION ROUTINES

Link Routine (Chart OA)

Module Name: IEDQOA

Entry Point: IEDQOA - called by the INTRO macro expansion.

Function: This routine controls the transient routines that perform initialization processing at INTRO execution time.

The Link routine issues a LINK to load and activate first the WTOR Interpreter routine, then the Password Scrambler routine, then the INTRO GETMAIN routine, then the Termname Table Sort routine, and last the Attach routine. Upon return from the WTOR Interpreter, the INTRO GETMAIN routine, and the Sort routine, the Link routine examines a return code in register 15 to determine whether the returning routine was successfully completed. If the return code is equal to zero, the Link routine links to the next initialization routine or, if all five routines have been executed, it returns to the INTRO macro expansion. If the return code is nonzero, the Link routine passes it to INTRO in register 15 and sends a diagnostic message to the system console. The format of this message is as follows:

> "IED065I INITIALIZATION ERROR xxxx", where xxxx is the value that INTRO passes in register 15.

External Routines:

- OS Link routine (SVC 6) - to activate the following modules:

  IEDQOB - WTOR Interpreter routine - to alter certain INTRO parameters.

  IEDQE6 - Password Scrambler routine - to scramble the MCP password.

  IEDQOG - INTRO GETMAIN routine - to acquire main storage for buffers and tables.

  IEDQOM - Termname Table Sort routine - to sort the Termname Table.

  IEDQOS - Attach routine - to attach On-Line Test, FE Common Write, and Operator Control.

- OS WTO routine (SVC 35) - to send a message to the system operator.

Tables/Work Areas: This routine passes the address of the AVT to each external routine.

Attributes: Reusable, transient, problem program mode.

WTOR Interpreter Routine (Chart OE)

Module Name: IEDQOB

Entry Point: IEDQOB - called through a LINK SVC by the Link routine (IEDQOA).

Functions: This routine permits system redefinition without reassembly. The system console operator can enter new values to replace values specified on specific keyword parameters of the INTRO macro at assembly time.

If the INTRO macro is assembled with the KEYLEN, CPB (or DISK=NO), STARTUP, and LNUNITS parameters properly specified, the operator is not given the opportunity to modify any INTRO operands at execution time, and when the WTOR Interpreter is brought into the system, it does not issue the WTOR command. To make execution time modifications, at least one of the above operands must be omitted from the INTRO assembly.

The Link routine (IEDQOA) issues a LINK SVC to load and activate the WTOR Interpreter routine from SYS1.LINKLIB. When WTOR Interpreter gains control, it sends the following message to the system operator:

- "IED001I TCAM JOB jobname, stepname, procstepname, ADDRESS OF AVT address" - where jobname is the job name, stepname is the step name, procstepname is the procedure step name, and address is the AVT address.

The WTOR Interpreter then checks the TCAM word in the CVT. If this word contains a nonzero value, there is a TCAM MCP already active in the system. The Interpreter, in this case, displays the following message and returns to the Link routine with an error code of X'04'.

- "IED014I TCAM ALREADY IN SYSTEM"

If the TCAM word in the CVT is equal to zero, the WTOR Interpreter checks the INTRO parameter list to determine whether any of the required operands are missing. If one or more of the required operands is missing, the WTOR Interpreter sends the following message.

- "IED002A SPECIFY TCAM PARAMETERS"

After sending this message, the system waits for an operator response. The response, in the form of keywords (either full or abbreviated) separated by commas, is limited to 41 characters. The routine examines the response field from left to right. An error in one keyword prevents examination of other keywords to its right. (These keywords may be placed in another response.) The WTOR Interpreter repeats the request for input until the operator indicates that he has finished entering keywords by coding "U" as the last keyword.

If the WTOR Interpreter finds an error in a keyword entered by the operator, it sends the following messages.

- "IED003A INVALID KEYWORD xxxx" - xxxx is the first four characters of the undefined keyword. All keywords to the right of this are ignored. All keywords to the left of the error have been interpreted.

- "IED004A REQUIRED PARAMETER MISSING. SPECIFY xx" - after the user codes "U" indicating he is through entering parameters, this statement reminds him of a required parameter he has yet to code. xx is the keyword needed. The user should reply with the indicated keyword (or keywords) and again indicate that he has finished responding by coding "U". The required keywords that may be called for are:

  S= "STARTUP" - cold or warm start

  B= "LNUNITS" - number of line buffers

  K= "KEYLEN" - size of each buffer unit

  D= "CPB" - number of CPBs - required only if disk is being used.

- "IED005A MSUNITS(M) SPECIFICATION NOT PERMITTED. CONTINUE RESPONSE" - the user has coded the "M=" keyword to set the number of main storage message queue records, but main storage queuing was not specified at INTRO assembly time. The "M=" response is legal only if main storage queuing (INTRO MSUNITS=YES or integer) is specified at assembly time.

- "IED006A INVALID OPERAND ON KEYWORD. RESPECIFY keyword" - where keyword is the keyword that contains the illegal value. All keywords to the right of the illegal one are ignored. All keywords before this keyword have been interpreted.

The WTOR Interpreter routine modifies the AVT entry that is set by the keyword being examined. The fields in the AVT with the related keyword and response are as follows:

INTRO Keyword	Response Keyword	AVT Field	Field Length
STARTUP	S	AVTBIT3	3 bits
LNUNITS	B	AVTNOLBF	2 bytes
MSUNITS	M	AVTTOTNC	4 bytes
RESTART	N	AVTCKRST	1 byte
KEYLEN	K	AVTKEYLE	2 bytes
UNITSZ	K	AVTKEYLE	2 bytes
CPINTVL	V	AVTCKELV	2 bytes
CONTROL	L	AVTCTLCH	8 chars
PRIMARY	P	AVTDOUBX	8 chars
INTVAL	I	AVTINTLV	2 bytes
PASSWRD	W	AVTPASWD	8 chars

CKREQS	R	AVTNCKPR	1 byte
CPB	D	AVTCPBNO	2 bytes
CPRCDS	E	AVTCPRCD	1 byte
CROSSRF	F	AVTCRSRF	4 bytes
COMWRTE	G	AVTCWFL1	1 bit
TRACE	T	AVTRACE	4 bytes
DTRACE	A	AVTDISTR	4 bytes
CIB	C	AVTCIB	1 byte
MSMIN	Y	AVTCMIN	4 bytes
MSMAX	X	AVTCMAX	4 bytes
DLQ	Q	AVTDLQX	8 bytes
OLTEST	O	AVTOLTST	1 byte
TOPMSG	H	AVTBIT2	1 bit

Other keywords cannot be modified.

Once the operator has entered the required keywords and the "U" response, WTOR Interpreter returns to the Link routine with a X'00' return code in register 15.

External Routines:

- OS WTO routine (SVC 35) - to write a message to the operator.
- OS Wait routine (SVC 1) - to wait for an operator response.

Tables/Work Areas: AVT, CVT, TCB.

Attributes: Transient, nonreusable, nonrefreshable, enabled, problem program mode.

INTRO GETMAIN Routine (Chart OG)

Module Name: IEDQOG

Entry Point: IEDQOG - called through a LINK SVC by the Link routine (IEDQOA).

Functions: This routine uses the OS GETMAIN macro to obtain main storage for and initialize line buffers, a main storage message queues data set (if requested), channel program blocks, and any trace tables or cross reference tables requested by the user.

If the INTRO GETMAIN routine is able to satisfy all the required GETMAIN requests, it returns to the Link routine with the successful return code X'00' in register 15. The value X'08' in register 15 indicates that sufficient main storage was not available to satisfy a GETMAIN request.

External Routine: OS Getmain routine (SVC 4) - to obtain main storage space.

Tables/Work Areas: AVT.

Attributes: Transient, reusable, refreshable, problem program mode.

Termname Table Sort Routine (Chart OM)

Module Name: IEDQOM

Entry Point: IEDQOM - called through a LINK SVC by the Link routine (IEDQOA).

Functions: This module sorts the Termname Table entries into alphabetical sequence. After the sort is finished, this routine recalculates the Termname Table offsets for any distribution lists, cascade lists, and invitation lists that refer to specific entries in the Termname Table. This routine also recalculates the offsets for alternate destinations.

The Termname Table Sort routine initializes the Termname Table fields that are necessary for the Binary Search routine. It also checks for the presence of a primary operator control terminal. If a dead-letter queue is specified, this routine calculates its Termname Table offset and places the offset at AVTDLQX in the AVT. If, however, the dead-letter queue is specified to a TSO terminal, this module proceeds as though no dead-letter queue was specified.

If this routine is successfully executed, it returns to the Link routine with a X'C0' return code in register 15. The three error conditions that can occur are indicated by the following return codes in register 15:

- X'12' - main storage is not available to satisfy a GETMAIN request.
- X'16' - terminal definition error. Error message "IEDQ007I terminal name ILLEGAL DESTINATION" is sent to the system console for each Terminal Table entry that contains an error.
- X'20' - primary operator control terminal definition error.

External Routines:

- OS Getmain routine (SVC 4) - to obtain main storage.
- OS WTO routine (SVC 35) - to send a message to the system operator.

Tables/Work Areas: AVT, Termname Table, Terminal Table.

Attributes: Transient, reusable, refreshable, problem program mode.

Attach Routine (Chart OS)

Module Name: IEDQOS

Entry Point: IEDQOS - called through a LINK SVC by the link routine (IEDQOA).

Functions: This routine attaches the Operatcr Control task, the On-Line Test module (if requested), and the FE Common Write routine (if requested) as tasks in the same partition cr region as the MCP. This routine determines the need for attaching On-Line Test and FE Common Write by testing switches in the AVT.

The Attach routine also loads modules that depend on operands on the INTRO macrc. If the system delay interval in AVTINILV is equal to zero, the Attach routine loads the System Delay subtask (IEDQHI), and places its address in the AVT. If PRIMARY is not specified as SYSCON, the Attach routine loads the Cperatcr Awareness Message Router (IEDQNX) and places its address in the AVT.

Upon completion, the Attach routine returns to the link routine.

External Routines:

- OS Attach routine (SVC 42) - to attach the requested tasks.

- OS Extract routine (SVC 40) - to build a communications parameter list.

- OS Load routine (SVC 8) - to load TCAM modules.

Tables/Work Areas: AVT.

Attributes: Transient, reusable, refreshable, problem prcgram mode.

Disk Message Queues Open Routines (Charts LB, LC, LD)

Module Names: IGG01930, IGG01931, IGG01934

Entry Points:

- IGG01930 - entered by an XCTL from an I/O support mcdule or another access methcd open executor when an OPEN DCB for a message queues data set is issued in an MCP. IGG01930 can also be reentered by a loop from itself if there are multiple DCBs to open. (Chart LB)

- IGG01931 - entered by an XCTL from IGG01930 after IGG01930 is completed. IGG01931 can also be reentered by a loop from itself if there are multiple DCBs to open. (Chart LC)

- IGG01934 - entered by an XCTL from IGG01931 after IGG01931 is completed. IGG01934 can also be reentered by a loop from itself if there are multiple DCBs to open. (Chart LD)

Functions: The functions of each routine are defined according to entry point.

- IGG01930

This routine gets main storage for and initializes a Data Extent Block (DEB) in subpool 254 for a message queues DCB. IGG01930 analyzes the the device type information provided in the Unit Control Block (UCB) to determine the type of direct access devices used for the message queues.

If IGG01930 finds an error condition, it sets error indicators in the AVT and issues an XCTL to the Open Error Handler routine (IGG01933).

When no errors are found, IGG01930 places the address of the next entry in the DCB parameter list in register 7 and the address of the next entry in the system Where-to-Go Table in register 8. The routine updates the Disk Message Queue Open entry in the Where-to-Go Table to point to Load II -- IGG01931. IGG01930 then issues an XCTL command to the module identified by the next nonzero entry in the Where-to-Go Table, specifically IGG01931.

- IGG01931

This routine completes the initialization of the DEB extents and calculates various values required by EXCP Driver. IGG01931 also builds and initializes all I/O Blocks (IOBs), one per DEB extent, required for disk operation.

If IGG01931 finds an error condition, it sets error indicators in the AVT and issues an XCTL to the Open Error Handler routine (IGG01933).

When no error is found, IGG01931 places the address of the next entry in the DCB parameter list in register 7 and the address of the next entry in system Where-to-Go Table in register 8. It updates the Disk Message Queues Open entry in the Where-to-Go Table to identify Load III -- IGG01934. IGG01931 issues an XCTL command to the module identified by the next nonzero entry in the Where-to-Go Table - IGG01934.

- IGG01934

This routine performs all the disabled initialization functions that are required by TCAM. This includes loading the TCAM Dispatcher, EXCP Driver, Disk End Appendage, and the Reusability-Copy subtask, if it is requested. In order to load a module, IGG01934 activates IOS, which checks the OS Contents Directory to determine whether that module has already been loaded. If there is an entry for the module in the directory, IOS adds one to the directory usage count. If there is not an entry for the module in the directory, IOS makes a two-byte entry in the directory, adds one to the usage count, and loads the module.

> Note: IGG01934 loads the Disk End Appendage for a Single CPB and EXCP Driver fcr a Single CPB when CPB=1 is specified by the user. Otherwise, IGG01934 loads the regular version of each of these modules.

IGG01934 places the address of the next entry in the DCE parameter list in register 7 and the address of the next entry in the system Where-to-Go Table in register 8. IGG01934 issues an XCTL to the module identified by the next nonzero entry in the Where-to-Go Table, either IGG01941, IGG01935, or system open.

External Routines:

- OS Getmain routine (SVC 4) - to obtain main storage space for the DEBs and IOBs.
- OS Load routine (SVC 8) - to load TCAM modules.

Tables/Work Areas: System Where-tc-Go Table, DCB parameter list, Open work area, AVT, DEB, IOB, UCB.

Attributes: Transient, enabled, reentrant.

Checkpoint Open Rcutine (Chart MA)

Module Name: IGG01941

Entry Point: IGG01941 - activated by an XCTL from IGG01934 when the OPEN checkpoint data set DCB is specified in an MCP.

Functions: This module opens a checkpcint data set in the MCP. To accomplish this, it performs the following activities:

- Determines the size of the GETMAIN work area (the size varies as a result of the INTRO operands "CKREQS" and "CPRCDS"),
- Issues a GETMAIN macro for the work area and puts the address in the AVT field AVTCKGET,
- Determines the beginning of the CKREQ-TIR table,
- Initializes the IOB and the disk channel program in the checkpoint work area,

- Determines the type of start-up required (cold, warm, or continuation) by investigating the disposition field coded on the OPEN macro, the start parameters (on INTRO), and the "normal closedown" bit in the checkpoint data set control record. Depending on these results, transfers control (XCTL) to either the Checkpoint Disk Allocation module or the Checkpoint/Restart from Environment Record module. The following conditions determine the type of start-up required and therefore indicate the routine to gain control:

1. DISP=NEW

   XCTL to the Checkpoint Disk Allocation routine

2. DISP=OLD, S=C, normal closedown

   XCTL to the Checkpoint Disk Allocation routine

3. DISP=OLD, S=C, abnormal closedown

   XCTL to the Checkpoint/Restart modules and scan the message queues

4. DISP=OLD, S=CY, normal closedown

   XCTL to the Checkpoint Disk Allocation routine

5. DISP=OLD, S=CY, abnormal closedown

   XCTL to the Checkpoint Disk Allocation routine

6. DISP=OLD, S=W, normal closedown

   XCTL to the Checkpoint/Restart modules and do not scan the message queues

7. DISP=OLD, S=W, abnormal closedown

   XCTL to the Checkpoint/Restart modules and scan the message queues

8. DISP=OLD, S=WY, normal closedown

   XCTL to the Checkpoint/Restart modules and do not scan the message queues

9. DISP=OLD, S=WY, abnormal closedown

   XCTL to the Checkpoint/Restart modules and do not scan the message queues

If, during execution, the Checkpoint Cpen routine determines that there is insufficient main storage for the checkpoint work area or if a disk I/O error occurs while reading the control record of the checkpoint data set, the routine sends an error message to the system console, sets AVTCKGFT equal to zero, and passes control to the next module in the system Where-to-Go Table.

External Routines:

- OS Getmain routine (SVC 4) - to obtain main storage for a work area.
- OS WTO routine (SVC 35) - to send a message to the system operator.
- OS Load routine (SVC 8) - to load a checkpoint module.
- OS EXCP routine (SVC 0) - to read a record from disk.

Tables/Work Areas: CVT, AVT, Checkpoint DCB, Checkpoint DEB, checkpoint work area, I/O work area, JFCB.

Attributes: Reentrant.

Checkpoint Disk Allocation Routine (Chart MM)

Module Name: IGG01949

Entry Point: IGG01949 - called by the Checkpoint Open routine when initialization of the checkpoint data is required.

Functions: This module determines the size of the various records for the checkpoint data set. The Checkpoint Disk Allocation routine first scans the TCAM tables to determine the size of an environment checkpoint record and the number of disk records necessary to contain it. The routine then finds the maximum number of priority level QCBs to be used for any one application program Destination QCB, and uses this number plus the length of the longest option area for any terminal entry to calculate the length of a CKREQ record. The length of an incident record is equal to the length of the longest option area or the length of the operator control data area, whichever is greater.

The Checkpoint Disk Allocation routine then calculates the number of each of the types of checkpoint records that will fill one track of the checkpoint data set. The routine uses the device type index (from the UCB) and the CS I/O Device Characteristics Table (address from the CVT) fields to calculate the number of records per track.

The Checkpoint Disk Allocation routine places the number of tracks in the checkpoint data set, the size of each disk record, and the number of records per track in the checkpoint work area. This routine also places the count and length of the various records in the checkpoint disk data set control record.

The Checkpoint Disk Allocation routine exits by issuing an XCTL to the next nonzero entry in the system Where-to-Go Table - IGG01942.

External Routines: None.

Tables/Work Areas: CVT, AVT, checkpoint work area, Cption Table, Termname Table, Terminal Table, QCE, DEB, DCB, invitation list, OS I/O Device Characteristics Table.

Attributes: Reentrant, transient, refreshable, enabled, supervisor mode.

Checkpoint Disk Initialization Routine (Chart MB)

Module Name: IGGC1942

Entry Point: IGG01942 - called by the Checkpoint Disk Allocation routine to initialize the checkpoint data set or by the Checkpoint/Restart from Envircnment Record routine to perform an error exit.

Functions: The Checkpoint Disk Initialization routine initializes the disk checkpoint data set into specific areas for a control record, environment checkpoint records, CKREQ records, and incident records.

This routine formats the checkpoint data set with dummy records. The CPRCDS operand of the INTRO macrc specifies the number of environment checkpoint records to be written in the disk checkpoint data set. The CKREQS operand of INTRO indicates the number of CKREQ records to be written in the data set. There is one control record. The remainder of the space that is allocated to the checkpoint data set on the disk is used for incident checkpcint records.

If, during execution, the Checkpoint Disk Initialization routine recognizes an error condition, it issues an error message via WTO, sets AVTCKGET equal to zero, and transfers contrcl to the next entry in the system Where-to-Go Table. The following error conditions can occur:

- Disk I/O error occurs while writing.

- Insufficient disk space for the minimum required checkpoint records:

  2 environment records,

  1 control reccrd,

  The number cf CKREQ records specified in the INTRO macro + 3 extra records, and

  1 incident record.

If the Checkpoint Disk Initialization routine is entered from the Checkpoint/Restart from Environment Record routine, the Initialization routine issues a WTO message that indicates an unrecoverable disk error, sets AVTCKGET equal to zero, and transfers control to the next entry in the system Where-to-Go Table.

If no errors occur, this routine transfers control to the next entry in the system Where-to-Go table.

External Routines:

- IECPCNVT - to convert the relative track address to the actual disk address. (This is an OS routine, found via a pointer in the CVT.)
- OS WTO routine (SVC 35) - to send a message to the system operator.
- OS EXCP routine (SVC 0) - to start a channel program to write a checkpoint record.
- OS Wait routine (SVC 1) - to allow time for the channel program to complete.

Tables/Work Areas: CVT, AVT, checkpoint work area, Checkpoint QCB, Checkpoint DEB, Checkpoint DCB.

Attributes: Reentrant.

Checkpoint/Restart from Environment Record Routine (Chart ME)

Module Name: IGG01943

Entry Point: IGG01943 - activated by the Checkpoint Open routine when a system restart is required.

Functions: When checkpoint restart is specified, this module uses the environment record segments in the checkpoint data set to reconstruct the MCP environment. The Checkpoint/Restart from Environment Record routine places information from the environment checkpoint record in the MCP tables.

The Restart from Environment Record Routine determines which environment record to use by subtracting the value of the INTRO operand "RESTART=" from the number of the most current environment record. The control record (the first record on the checkpoint data set) contains the number of the most current environment record. If the result of the subtraction is not a positive value, this routine adds the value of the INTRO operand "CPRCDS" (the total number of environment records) to the result.

If this restart routine finds that the TTR of the environment record in the control record is equal to zero, the environment record has had a disk error. In this case, this restart routine issues a WTO error message and recovers by using the previous environment record. If all the TTRs are equal to zero, the routine sets the X'08' bit in the first byte of the control record and then transfers control to IGG01942, which issues a WTO error message. In this case, no checkpoints are taken for the duration of the job.

After successful execution, this module exits to IGG01944. If an error occurs during processing, exit is to the next module in the system Where-to-Go Table.

External Routines:

- IFCPCNVT - to convert the relative track address to the actual disk address. (This is an OS routine, found via a pointer in the CVT.)
- OS EXCP routine (SVC 0) - to read a checkpoint record segment.
- OS Wait routine (SVC 1) - to allow I/O to complete.
- OS WTO routine (SVC 35) - to send a message to the system operator.

Tables/Work Areas: CVT, AVT, checkpoint work area, TCB, Checkpoint DEB, Checkpoint DCB, Checkpoint QCB, Termname Table, Terminal Table, invitation list.

Attributes: Reentrant.

Checkpoint/Restart from Incident and CKREQ Records Routine (Chart MG)

Module Name: IGG01944

Entry Point: IGG01944 - activated by the Checkpoint Open routine after the Checkpoint/Restart from Environment Record routine has successfully executed.

Functions: This module reads the incident records for stop line or start line and the CKREQ records from the checkpoint data set and uses these records to update the MCP environment. If STARTUP=WY is specified as an operand of the INTRO macro, TCAM does not use the incident records to update the MCP environment; otherwise, this module performs the following functions.

The routine first compares the time in an incident record to the time in the environment record used for the restart. If the incident record is more recent, it is used to update the MCP tables. The key field in an incident record indicates the type of information in the record. Incident checkpoints are taken as a result of a CHECKPT macro in an MH, a TCHNG macro in an application program, or an operator control command.

Note: This rcutine processes cnly the incident records for Start Line and Stop Line operator control commands. All other commands are processed after the lines are opened at READY time. When this routine reccqnizes a Start or Stop Line command, it stores the line status in the QCBLINK field of the Destination QCB for the line. The Line Open rcutine uses this status field.

CKREQ records do not contain the time at which they are written. These records are used to synchronize the information in Terminal Table process entries with an OS Checkpoint taken in an application proqram. The Checkpoint Restart from Incident and CKREQ Records routine reads all the CKREQ records in the data set. This routine moves each TTR and Termname Table offset into the CKREQ-TTR table in the checkpoint work area. If the offset value in a CKREQ record is not equal to zero, this routine uses the CKREQ data to update the MCP tables that pertain to the process entry.

The Checkpoint/Restart from Incident and CKREQ Records routine exits to the next module in the system Where-to-Go Table - the Checkpoint Continuation Restart module (IGGC1945) if this is a restart after an abnormal closedown, or the next open executor after a normal closedown. Fcr a normal closedown, this routine also sets bit X'01' in AVTCKELF to indicate the type of restart.

External Routines:

- IECPCNVT - tc convert the relative track address to the actual disk address. (This is an OS routine, found via a pointer in the CVT.)

- IEDQTNT - Termname Table code - tc obtain a terminal entry address.

- OS EXCP routine (SVC 0) - to read a checkpcint record.

- OS Wait routine (SVC 1) - to allow I/C to complete.

- OS WTO routine (SVC 35) - to send a message to the system operator.

Tables/Work Areas: AVT, checkpoint work area, QCB, DCB, Terminal Table, Termname Table, Cption Table, CVT.

Attributes: Reentrant.

Checkpoint Continuation Restart Rcutine (Chart MJ)

Module Name: IGGC1945

Entry Point: IGG01945 - entered by an XCTL from the Checkpoint/Restart from Incident and CKREQ Records routine (IGG01944) after an abnormal closedcwn.

Functions: This module performs any required processing of the message queues data set at restart time.

By coding STARTUP=WY on the INTRO macro, the user specifies that after a system failure he wants a warm restart without a scan of the message queues. In this case, the Continuation Restart routine locates the last message placed on each FEFO queue in the message queues data set before the time of the last checkpoint. This routine then places zeros in the FEFO chain field of any messages that were placed on the queue after the checkpoint - these messages are subsequently lost.

If SYNC=YES is coded on the TPROCESS macro, the user has synchronized queues for application programs. In this situation, the Continuation Restart module scans the FIFO message queues for the specified process entry and recreates a FEFO queue, in FIFO order, that includes all messages on the FEFO queue at any time after the last checkpoint was taken. To determine which serviced messages should be placed on the FEFO queue, this routine compares the disk record number of the last segment of the first message on the FEFO queue at the time of the last checkpoint with the disk record number of the last segment of every complete, uncanceled message on the FIFO queue. If the record number of the message on the FIFO queue is greater than the record number of the message from the FEFO queue, this routine places the message that is on the FIFO queue on the restart-FEFO queue.

If neither of the above situations exists, the Continuation Restart routine scans each FIFO message queue and recreates a restart FEFO queue in FIFO order. This queue contains all complete, unserviced, uncanceled messages. The Continuation Restart routine must read and check each segment of a message for logical read errors in order to determine whether the message is completely received.

In both OS synchronized and regular continuation restart, this routine recreates the FEFO chain, updates the sequence numbers, and recreates the queue-back chain. The sequence number in a message is only used to update its terminal entry if the number of the message is greater than the number already in the entry. If the queue-back pointer in a message is higher than the queue-back pointer in its Destination QCB, the Continuation Restart routine uses the record number of the message buffer, not the queue-back field, to update the QCB.

The Continuation Restart routine exits by issuing an XCTL to the module indicated by the next nonzero entry in the system Where-to-Go Table.

External Routines:

- IGG019Q8 - Checkpoint Continuation Restart subroutine - to examine a terminal entry and to activate IGG019RC.
- OS Load routine (SVC 8) - to load IGG019Q8.
- IEDQTNT - Termname Table code - to get a terminal entry address.

Tables/Work Areas: AVT, checkpoint work area, CPB, disk data area of the message, buffer prefix, QCB, Termname Table, Terminal Table.

Attributes: Reentrant.

Checkpoint Continuation Restart Subroutine (Chart Q8)

Module Name: IGG019Q8

Entry Points:

- IGG019Q8 - loaded by the Checkpoint Continuation Restart routine (IGG01945) to check terminal entries.
- IGG019Q8+4 - activated by the Checkpoint Continuation Restart routine to execute disk I/O.
- IGG019Q8+8 - activated by the Checkpoint Continuation Restart routine to update sequence numbers.
- IGG019Q8+12 - activated by the Checkpoint Continuation Restart routine to update the AVT value of address for queuing.
- IGG019Q8+16 - activated by the Checkpoint Continuation Restart routine to initialize registers.

Functions: This module is an extension of the Checkpoint Continuation Restart routine (IGG01945). At the IGG019Q8 entry point, this module examines the terminal entries to determine whether a scan should be performed on the message queues. At the IGG019Q8+4 entry point, this module sets up the CPB for disk I/O on the message queues data set and then activates the EXCP Driver (IGG019RC) to actually perform the I/O operation. At the IGG019Q8+8 entry point, this module updates the message sequence number in the terminal entry. At the IGG019Q8+12 entry point, this module examines and, if necessary, updates the AVTRADDR and AVTNADDR queuing addresses in the AVT. At the IGG019Q8+16 entry point, this module initializes registers with values for IGG01945.

The Checkpoint Continuation Restart subroutine always returns to the Checkpoint Continuation Restart routine.

External Routines

- IGG019RC - EXCP Driver - to perform I/O on the disk message queues data set.

- OS Wait routine (SVC 1) - to allow time fcr completion of the disk I/O activity.

Tables/Work Areas: AVT, checkpoint work area, Terminal Table, AVT, CPB, QCB.

Attributes: Reentrant, transient.

Line Group Open Routines (Charts LE, LF, LG, LH, LI, LJ, and LK)

Module Names: IGG01935, IGGC1936, IGG01937, IGGC1938, IGG01939, IGG01940, IGG01948.

Entry Points:

- IGG01935 - entered by an XCTL from an I/O support module or from another access method open executor when an OPEN line group DCB is issued in an MCP. It may also be reentered by a loop from itself if there are multiple DCBs to open. (Chart LE)

- IGG01936 - entered by an XCTL from IGG01935. It may also be reentered by a loop from itself if there are multiple DCBs to open. (Chart LF)

- IGG01937 - entered by an XCTL from IGG01936. It may also be reentered by a loop from itself if there are multiple DCBs to open. (Chart LG)

- IGG01938 - entered by an XCTL from IGG01937. It may also be reentered by a loop from itself if there are multiple DCBs to open. (Chart LH)

- IGG01939 - entered by an XCTL from IGG01938. It may also be reentered by a loop from itself if there are multiple DCBs to open. (Chart LI)

- IGG01940 - entered by an XCTL from IGG01939. It may also be reentered by a loop from itself if there are multiple DCBs to open. (Chart LJ)

- IGG01948 - entered by an XCTL from IGG01940. It may also be reentered by a loop from itself if there are multiple DCBs to open. (Chart LK)

Functions: The functions of each routine are defined according to entry point.

- IGG01935

This routine builds and initializes a line DEB. IGG01935 examines the Task I/O Table (TIOT) to determine the number of lines in this line group. It then obtains main storage for and initializes a line DEB in subpool 254.

IGG01935 checks each unit control block (UCB) to verify that similar devices are attached to each line and that either a 2701, 2702, or 2703 control unit is being used. This routine also locates a typical entry in the Device Characteristics Table for each line group, sets an index into the branch table in IGG01936, and clears a register to contain the Line Control Block (LCB) size.

If IGG01935 finds an error condition, it sets error indicators in the AVT and issues an XCTL instruction to give control to the Open Error Handler routine (IGG01933). If the user has specified a TCAM entry in his exit list, the Open Error Handler routine will return control to the next nonzero entry in the system Where-to-Go Table after it has processed all error conditions.

IGG01935 exits by issuing an XCTL command to the module indicated by the next nonzero entry in the system Where-to-Go Table - IGG01936.

- IGG01936

This routine determines the size of the channel programs for all devices for the line group being opened.

IGG01936 provides the number of channel command words (CCWs) for a minimum program for all devices. Additional CCWs are provided as determined by examining the optional feature bits in the UCB and the typical entry for each applicable device in the Device Characteristics Table.

This routine issues a GETMAIN instruction to get an LCB for each line in the line group and then places the Send Scheduler STCB in the STCB chain of the Destination QCB.

When IGG01936 issues an XCTL to IGG01937 (the next nonzero entry in the system Where-to-Go Table), it passes in register 10 the total number of CCWs required for each channel program for each device in the line group.

- IGG01937

This routine builds and initializes all the LCBs for this line DCB open.

IGG01937 divides the LCB area into individual LCBs for each of the lines and initializes each LCB. If the scheduling priority for this line is send, this routine moves the Send Scheduler STCB into the STCB chain for the LCB.

IGG01937 exits by issuing an XCTL command to the module indicated by the next nonzero entry in the system Where-to-Go Table - IGG01938.

- IGG01938

This routine builds channel programs in the Line Control Blocks (LCBs) for the lines of the line group being opened.

IGG01938 also tests to determine whether the lines are to be opened idle.

IGG01938 exits by issuing an XCTL command to the module indicated by the next nonzero entry in the system Where-to-Go Table - IGG01939.

- IGG01939

This routine loads some of the modules required for line operation. These modules include the TCAM Dispatcher, the appropriate receive schedulers, and the Start-up Message routine (if requested). In order to load a module, IGG01939 activates IOS, which checks the OS Contents Directory to determine whether that module has already been loaded. If there is an entry for the module in the directory, IOS adds one to the directory usage count. If there is not an entry for the module in the directory, IOS makes a two-byte entry in the directory, adds one to the usage count, and loads the module. If IOS loads the TCAM Dispatcher, it also places a pointer to the address of the AVT in the CVT.

IGG01939 exits by issuing an XCTL command to the module indicated by the next nonzero entry in the system Where-to-Go Table - IGG01940.

- IGG01940

This module completes the loading of the modules required for line operation. These modules include the Send Scheduler, the PCI Appendage, and the Line End Appendage. IGG01940 also loads the device dependent special characters required for initial I/O operations and starts I/O on each line in the line group.

<u>Note</u>: The version of Line End Appendage that IGG01940 loads depends on the user-coded operands on the INTRO macro:

ENVIRON=TCAM	IGG019Q5
ENVIRON=TSO or MIXED	IGG019Q3
LINETYP=BISC	IGG019Q2
LINETYP=MINI	IGG019Q4
LINETYP=BOTH	IGG019R0

IGG01940 exits by issuing an XCTL command to IGG01948.

- IGG01948

This routine places line-specific information in the Cross Reference Table. The data placed in this table includes the UCB name, the UCB address, the LCB address, and the Destination QCB address for each line in the line group.

Upon entry, IGG01948 issues a TIME macrc instruction to get the current time of day from the operating system. The routine then tests each line (each LCB) to determine whether it has successfully completed its initial I/O operations. If the initial I/O is not complete, IGGC1948 issues another TIME macrc and determines whether 28 seconds have elapsed. If 28 seconds have not passed, the routine continues checking for I/O completion until 28 seconds have elapsed or until the LCB has been marked to indicate I/O completion. At the end of 28 seconds if the I/O has still nct ccmpleted, IGG01948 writes a message on the system console to identify the line that was not successfully cpened. When I/O operaticn has completed, the routine goes to the next line in the line group and continues checking for I/O completion.

IGG01948 exits by issuing an XCTL command to the module identified by the next ncnzero entry in the system Where-to-Go Table. This module is the system module IGG0190S.

External Routines:

- OS Getmain routine (SVC 4) - to obtain main storage.
- OS Load routine (SVC 8) - to lcad TCAM modules.
- OS EXCP routine (SVC 0) - to start I/C cn a line.
- OS Time routine (SVC 11) - to get the current time of day.
- OS WTO routine (SVC 35) - to send a message to the system operator.

Tables/Work Areas: Where-to-Go Table, DCB parameter list, DEB, Device Characteristics Table, Special Characters Table, Cross Reference Table, TIOT, UCE, LCB, QCB.

Attributes: Transient, enabled, reentrant.

Open Error Handler (Chart LA)

Module Name: IGG01933

Entry Point: IGG01933 - activated by any of the TCAM open executors when an error is detected.

Functions: This module handles all serious errors detected during the opening of a TCAM application program DCB, a message queues data set DCB, or a line group DCB. The Open Error Handler sends an error message to the system console. The value of xx in the message, IED008I TCAM OPEN ERROR xx, depends on the specific parameters passed to the Open Errcr Handler by the open executor that detected the error condition.

If the user does not provide a TCAM error exit, the Open Error Handler causes TCAM to abend with a specific abend code. If the user does provide an error exit, the Open Error Handler passes control to the routine at the address specified by the error exit. The Open Error Handler passes an error code in register 0 and an option code in register 1 to the user-specified error routine. The option code allows the error routine to decipher which of the available options to use. The error routine returns a code in register 15 to indicate which of the following actions the Open Error Handler is to take:

1. Abend the TCAM job (return code = 2 or greater)
2. Ignore the data set that is in error (return code = 0)
3. Continue processing with limited capabilities (return code = 1)

If the error routine specifies an option that is not available for the error in question, the open executor sends the same error message to the system console again. This loop of sending the message and getting a response from the user-specified error routine continues until the Open Error Handler receives a valid return code in register 15.

External Routines:

- OS WTO routine (SVC 35) - to send an error message to the system operator.

- OS SYNCH routine (SVC 12) - to go to a user-specified error routine.

Tables/Work Areas: System Where-to-Go Table, DCB, AVT.

Attributes: Transient, enabled, reentrant.

Start-up Message Routine (Chart R6)

Module Name: IGGC19R6

Entry Point: IGG019R6 - activated when Line End Appendage posts an LCB that points to the Start-up Message QCB to the ready queue after receiving an open I/O interrupt.

Functions: This module obtains and queues any messages that the user has to send to a terminal for the specified LCB at start-up time.

The Start-up Message routine first locates all the terminal entries that are associated with the specified LCB. It then passes the address of each entry and the address of the option fields for that entry, if present, to the routine specified by a user exit. There are two possible user exit addresses specified as operands of the READY macro: one is given control if a cold restart is in effect; the other, if a warm or continuation restart is in effect.

If the routine specified by the user exit has a message to send to a terminal, the user routine returns to the Start-up Message routine with the address of the message in register 15 and the length of the message in the first byte of the message itself. A zero in register 15 indicates that the user routine has no message to enter.

When there is a message to be sent to a terminal that is main-storage-only queued, the Start-up Message routine removes buffers from the Buffer Request QCB, builds the message, and passes one unit of the message at a time to the Destination Scheduler (IEDQHM02) to be placed on the Destination QCB. In the case of restart with main-storage-only queuing, there are no messages on the message queue; therefore, no special measures are taken to ensure that start-up messages are queued first.

When there is a message to be sent to a terminal that is disk queued, the Start-up Message routine removes one CPB from the free CPB pool and, one unit at a time, builds the required number of buffers in the CPB work area. After each unit is obtained, the Start-up Message routine builds the CPB and branches to the EXCP Driver routine (IGG019RC) to write the unit on disk. When EXCP Driver returns to the Start-up Message routine, Start-up Message waits on the ECB at AVTOSECB to allow time for I/O to complete before building another buffer unit.

The Start-up Message routine assigns disk relative record numbers in the conventional manner. But the routine places the message at the first of the FEFO queue by moving the QCBFFEFO field into the message FEFO chain and placing the record number of this message in QCBFFEFO.

After the Start-up Message routine has processed all the terminals associated with the specified LCB, it increments the count of lines processed (AVTSMCNT) and compares the counter with a count of the total number of lines opened (AVTLNCNT). If the counts are equal, Start-up Message tposts the LCB to itself and returns control to the TCAM Dispatcher at the DSPDLETE entry point to have the Start-up Message routine deleted. If the counts are not equal, Start-up Message tposts the LCB to itself and returns control to the TCAM Dispatcher at the DSPPOST entry point.

External Routines:

- User routines specified as user exits in operands of the READY macro.
- IGG019RC - EXCP Driver routine - to write the units of a message on disk.
- IEDQHM02 - Destination Scheduler - to place buffers on the appropriate Destination QCB.
- IEDQTNT - Termname Table code - to obtain a terminal entry address.
- OS Wait routine (SVC 35) - to allow I/O to complete.

Tables/Work Areas: AVT, LCB, Termname Table, Terminal Table, QCB, DCB, Option Table, buffer prefix, CPB, SCB, data area of a message.

Attributes: Reentrant, resident, problem program mode.

Ready Routine (Chart ND)

Module Name: IEDQND

Entry Point: IEDQND - activated by the READY macro expansion.

Functions: If the AVTCKGET field contains a nonzero value, which indicates that a checkpoint DCB has been opened, the Ready routine reads and processes all incident checkpoint records that are more recent than the environment record. If the key field of a record indicates TCHNG or CHECKPT, this module updates the TRMSTATE and option fields for the associated terminal entry. If the key field indicates operator control, but not Start or Stop Line, this module moves the data into the operator control work area at CPCCKELE, posts the ECB for Operator Control, and issues a WAIT to allow the data to be processed. If this module encounters a disk error, it issues a WTO error message (IED085I) and ignores the incident record on which the error occurred.

After all the incident records are processed, this module issues a FREEMAIN for the I/O buffer and then issues an ATTACH SVC to attach the Checkpoint Executor in the same system partition as the MCP. The Ready routine saves registers in AVTSAVE2 in such a way that the TCAM Dispatcher will tpost the environment checkpoint request element to the ready queue. This routine also loads IEDQNX if the primary operator control terminal is not the system console and IEDQHI if the system delay is not zero.

If On Line Test is specified as an operand of the INTRO macro, the Ready routine determines whether there is sufficient main storage for On-Line Test to perform its functions. If there is not enough main storage for the minimum requirements of On-Line Test, the MCP abends. If there is enough main storage for minimum On-Line Test requirements, but not enough for the requested amount, the Ready routine issues a warning WTO message (IED094I).

If the Checkpoint and On-Line Test tasks are not attached, the Ready routine marks complete their respective termination ECBs.

The Ready routine also checks all the terminal entries in the Terminal Table. If CALL is specified on a TERMINAL macro, this routine puts the QCB on the time delay queue.

Upon completion, the Ready routine returns control to the READY macro expansion (the address in register 14).

External Routines:

- OS Attach routine (SVC 42) - to attach the Checkpoint Executor and On-Line Test.

- OS Getmain routine (SVC 4) - to request the amount of main storage that is required by On-Line Test.

- OS Freemain routine (SVC 5) - to free the main storage that was acquired by a GETMAIN macro.

- IEDQTNT - Termname Table code - to obtain a terminal entry address.

- OS EXCP routine (SVC 0) - to start an I/O operation.

- OS Load routine (SVC 8) - to load a TCAM module.

- OS Post routine (SVC 2) - to post an ECB.

- OS WTO routine (SVC 35) - to send a message to the system operator.

- OS Wait routine (SVC 1) - to allow time for an event to complete.

- IECPCNVT - OS Convert routine - to convert the TTR to an MBBCCHHR address.

Tables/Work Areas: AVT, Terminal Table entry, Termname Table, Operator Control AVT.

Attributes: Reusable, problem program mode, transient.

SYSTEM CONTROL ROUTINES

TCAM Dispatcher (Chart RB)

Module Name: IGG019RB

Entry Points: IGG019RB

The TCAM Dispatcher provides some of the service functions of a queue manager by allowing routines to branch to entry point labels in a DSECT. This DSECT is included in an assembly by issuing the macro TDISPD.

Entry point labels not ending in "R" result in loss of control by the branching subtask. Those ending in "R" result in an immediate return to the branching subtask after the requested function has been performed. Branch entry points to the TCAM Dispatcher in the branch table RETTBL include the following:

Label Description

DSPDLETE Functions: Delete the module with entry point IGG019R6 (the Start-up Message routine), and tpost a chain of RCBs. Parameter register: 1 - the address of the first item in a chain of items to be tposted, or X'xx000000'. The link field of the last item in the chain must contain X'xx000000'.

Exit point: DSPDISP

DSPCHAIN Function: Tpost a chain of RCBs.

Parameter register: 1 - the address of the first item in a chain of items to be tposted, or X'xx000000'. The link field of the last item in the chain must contain X'xx000000'.

Exit point: DSPDISP

DSPLIST Function: Tpost a list of RCBs.

Parameter register: 1 - the address of a list of addresses of RCBs. The high-order byte of the last RCB must contain X'80'.

Exit point: DSPDISP

DSPPOST Function: Tpost one RCB.

Parameter register: 1 - the address of an RCB.

Exit point: DSPDISP

DSPPOSTR Function: Tpost one RCB.

Parameter register: 1 - the address of an RCB.

Exit point: Address in register 14.

DSPWAIT Function: Obtain an RCB from the element chain of a QCB, or, if none is there, wait for an RCB to arrive.

Parameter registers:

3 - the address of the QCB from which an RCB is to be obtained.

7 - the address of the QCB that contains the STCB for the subtask to receive the element.

Exit point: DSPDISP

DSPTSTQ Function: Determine whether an STCB is twaiting on a particular QCB, and, if it is not, chain the STCB onto that QCB.

Parameter registers:

3 - the address of the particular QCB.

7 - the address of the QCB that currently has the STCB at the top of its chain.

Exit point: DSPDISP

DSPTSTQR Function: Determine whether an STCB is twaiting on a particular QCB, and if it is not, chain the STCB onto that QCB.

Parameter registers:

3 - the address of the particular QCB.

7 - the address of the QCB that currently has the STCB at the top of its chain.

Exit point: Address in register 14.

DSPUNAV Function: Remove an STCB from one QCB and place it into another.

Parameter registers:

3 - the address of the QCB that is to receive the STCB.

7 - the address of the QCB that currently has the STCB at the top of its chain.

Exit point: DSPDISP

DSPUNAVR Function: Remove an STCB from one QCB and place it into another.

Parameter registers:

3 - the address of the QCB that is to receive the STCB.

7 - the address of the QCB that currently has the STCB at the top of its chain.

Exit point: Address in register 14.

DSPPRIO Function: Place an item into a chain by priority.

Parameter registers:

1 - the address of the item.

7 - the address of the chain to receive the item.

Exit point: DSPDISP

DSPPRIOR Function: Place an item into a chain by priority.

Parameter registers:

1 - the address of the item.

7 - the address of the chain to receive the item.

Exit point: Address in register 14.

DSPLIFO Function: Place an item at the beginning of a chain.

Parameter registers:

1 - the address of the item.

7 - the address of the chain to receive the item.

Exit point: DSPDISP

DSPLIFOR Function: Place an item at the beginning of a chain.

Parameter registers:

1 - the address of the item.

7 - the address of the chain to receive the item.

Exit point: Address in register 14.

DSPDISP Function: Activate the highest priority subtask that is waiting on the highest priority element that has been sent to a subtask.

Parameter registers: None.

Exit point: Entry point of the activated subtask.

DSPBYPAS Function: Activate a subtask immediately.

Parameter registers:

1 - the address of the element to pass to the subtask.

3 - the address of the STCB that controls the subtask.

7 - the address of the QCB that controls the STCB.

Exit point: Entry point of the activated subtask.

Functions: The TCAM Dispatcher allocates and schedules the system resources. The resources, or elements, wait in queues for allocation. The activity of these queues is controlled by the ready queue, which contains elements to be passed from one subtask to another.

Associated with each element on the ready queue is the queue to which the element is directed.

Each queue in the system is represented by a queue control block (QCB), which is the connecting link between elements and the subtasks waiting for the elements. A subtask control block (STCB) represents each waiting subtask. A resource control block (RCB) prefaces each element.

Elements and STCBs are inserted in their respective chains on the QCB in priority-FIFO order, that is, first-in-first-out within each priority class.

Queue Control Block

Offset		
0		ELCHN
+4		
+8		STCHN

ELCHN - the address of first element controlled by this QCB, if the QCB controls any elements.

STCHN - the address of first subtask control block to receive control when an element is tposted to this QCB.

The TCAM Dispatcher ignores all other fields.

Resource Control Block

Offset		
0		QBCA
+4	PRI	LINK

QBCA - the address of the QCB to which the RCB is tposted.

PRI - the ready queue and chaining priority of the RCB.

LINK - the address of the next item in the chain in which this item appears.

The TCAM Dispatcher ignores all other fields.

Subtask Control Block

Offset		
0	MCPL	
+4	PRI	LINK

MCPL - the subtask entry code (the key to tell the TCAM Dispatcher how to find the subtask code).

PRI - the priority of the STCB, if it is to be compared against others.

LINK - the address of the next item in the chain in which this item appears, if any.

The TCAM Dispatcher ignores all other fields.

When an element reaches the top of the ready queue (AVTREADY), the TCAM Dispatcher activates the highest priority subtask associated with the QCB indicated by the first word of the RCB. This element becomes the parameter passed to the newly-activated subtask.

The TCAM Dispatcher removes the highest priority element from the ready queue by placing the address of the element in register 1. The Dispatcher then places the link field of the RCB of the element in the ready queue - this puts a new element at the top of the ready queue.

When there are no elements on the ready queue, AVTREADY contains the address of AVTDELEM, which has a zero priority value. The QCB pointer in AVTDELEM points to a QCB that has an STCB with an MCPL field of zero. In this situation the TCAM Dispatcher activates a special routine within itself. This routine issues a system WAIT command. TCAM Dispatcher activity resumes when an I/O routine or an application program causes an OS interrupt to tpost an element to the ready queue.

When an element is tposted to a QCB that represents an attached TCAM subtask, the TCAM Dispatcher links the element into the element chain of the QCB and posts the ECB (the second word of the QCB) for the attached task complete. The Dispatcher recognizes this situation when the MCPL field of the STCB is equal to X'02'.

When TCAM is executing in a multiprocessing environment, the TCAM Dispatcher examines QCBSTVTO for a zero before inserting an element in the QCB element chain and posting the ECB for the attached task. If QCBSTVTO is equal to zero, processing proceeds as just described; otherwise, the Dispatcher loops until the byte at QCBSTVTO is equal to zero.

When the value of the MCPL field of the STCB being examined by the TCAM Dispatcher is greater than X'02' and less than X'0C', the Dispatcher calculates the entry point of the subtask to be dispatched. If the MCPL value is X'04', the subtask entry point immediately follows its two-byte STCB; therefore, the entry point is equal to the address of the STCB plus two bytes. If the MCPL value is X'06', the Dispatcher adds four bytes to the STCB address; if the MCPL value is X'08', the Dispatcher adds six bytes to the STCB address; and if the MCPL value is X'0A', the Dispatcher adds eight bytes to the STCB address.

When the value of the MCPL field is greater than X'0A', the TCAM Dispatcher activates a subtask by using the MCPL field as an index into the AVT branch table located at AVTDISP.

If a subtask is to execute without receiving an element, it is activated if its STCB is tposted, as if it were an RCB, with the MCPL field containing the correct subtask entry code, the next three bytes containing the address of AVTREADY-8, the PRI field containing a priority value, and the LINK field containing space for a link address.

To support dispatching while enabled for interruption, the Dispatcher uses two ready queues. One of these is used by disabled appendages for tposting elements; the other is used by enabled modules. The two ready queues are not managed by the same technique; however, each is called a ready queue because it contains elements to be processed by various subtasks.

The ready queue for the appendages is FIFO only and consists of two words: pointers to the first and the last elements on the queue. Appendages put an element on the queue by linking the new element to the one pointed to by the second word of the ready queue.

The enabled ready queue is managed by the priority-FIFO technique. The TCAM Dispatcher has the responsibility of merging the two ready queues just prior to dispatching. When the ready queues are empty, the TCAM Dispatcher issues a system WAIT macro, which can be satisfied by a tpost from an appendage or from an application program.

External Routines:

- OS Wait routine (SVC 1) - to wait for an interrupt.
- OS Post routine (SVC 2) - to post the ECB for an attached task.
- OS Delete routine (SVC 9) - to delete the Start-up Message routine from main storage.

Tables/Work Areas: QCB, RCB, STCB, AVT.

Attributes: Reentrant, refreshable.

TCAM Dispatcher with Subtask Trace (Chart RC)

Module Name: IGGC19RO

Entry Point: IGGC19RO

The TCAM Dispatcher with Subtask Trace provides the same queue management entry points as the TCAM Dispatcher (IGG019RB).

Functions: The TCAM Dispatcher with Subtask Trace is the same as the TCAM Dispatcher (IGG019RB) except that it provides one additional function. Each time a subtask is activated, the TCAM Dispatcher with Subtask Trace makes an entry in the wraparound Subtask Trace Table pointed to by AVTDISTR.

The Dispatcher with Subtask Trace is included in a TCAM MCP when the DTRACE keyword of the INTRO macro is coded with a nonzero numerical value. The DTRACE keyword defines the number of entries in the Subtask Trace Table. The format and control of this table are discussed in the Diagnostic Aids section of this publication.

External Routines:

- OS Wait routine (SVC 1) - to wait for an interrupt.

- OS Post routine (SVC 2) - to post the ECB for an attached task.

- OS Delete routine (SVC 9) - to delete the Start-up Message routine from main storage.

Tables/Work Areas: AVT, QCB, RCB, STCB, Subtask Trace Table.

Attributes: Reentrant, refreshable.

AQCTL SVC 102 Routine (Chart EB)

Module Name: IGC102

Entry Point: IGC102 - called by an SVC 102 command from any routine in the system.

Functions: This is a multipurpose routine (resident Type I SVC) that performs the following functions:

- Moving data across partition boundaries.

- Posting ECBs in other tasks.

- Tposting elements to the TCAM disabled ready queue.

- Flagging the TCB that represents a TSO application program as eligible or not eligible for swap.

- Flagging the TCB that represents an application program as eligible or not eligible for rollout.

When a routine in the TCAM system needs to have one of the above functions performed, it builds the standard parameter list for a specific function and places a pointer to that list in register 1. The routine then issues an SVC 102 command to activate the AQCTL SVC 102 routine.

If the routine that activates the AQCTL SVC 102 routine is not part of the MCP task, the AQCTL SVC 102 routine tests for a multiprocessing environment. If the result of this test is negative, this routine issues a BALR to the OS Task Removal routine to flag the MCP not eligible to be dispatched.

Byte 0 of each standard parameter list contains the action code for the AQCTL SVC 102 routine, and the high-order byte of the last word in each list contains X'80'. The value of each bit in byte 0 is as follows:

Bit	Function
0	Flag the issuing task not eligible for rollout
1	Post the RORI ECB complete
2	Post a standard or TSC ECB complete
3	Flag the issuing task not eligible for swap
4	Move data across a partition boundary
5	Enqueue an element on the disabled ready queue and post the MCP ECB complete
6	Flag the issuing task eligible for swap
7	Flag the issuing task eligible for rollout

The ECBs that this routine cause to be posted complete are in three different categories.

1. TSO (Time Sharing Option) - this type of ECB belongs to a task that may not be in main storage (swapped out) at the time of the post.

2. RORI (Rollout/Rollin) - this type of ECB is for a task that may not be in main storage (rolled out) at the time of the post.

3. Standard - this type of ECB is always in main storage at the time of the post.

The AQCTL SVC 102 routine interfaces with the OS Post routine (IEAOSY50), a resident Type I SVC, at a special non-validity checking entry point (IEAOPT01), the address of which is in the CVT. Input for the Post routine at this entry point is as follows:

Register 15 - the address of the branch entry IEAOPT01.

Register 14 - the return address.

Register 13 - in the low order 16 bits, the TJID (TSO Job Identifier) for the ECB to be posted. For ECBs that are not rolled out, this register contains binary zeros.

Register 11 - the complement of the ECB address. For a cross-partition post, the low-order bit is set to one.

Register 10 - the completion code.

When the task is currently rolled out, the AQCTL SVC 102 routine sets a bit in the TCB to indicate to the Rollout/Rollin routine at rollin time that there is a POST pending for this task.

The AQCTL SVC 102 routine branches to the Time Sharing Interface routine in the nucleus of a task to be flagged eligible or not eligible for swap. The interface is accomplished via the TSEVENT macro.

The contents of the parameter list built by the calling routine vary according to the bit setting in the action code control byte.

- For POST requests only, the list may contain either two or three fullwords. The high-order byte of the first fullword is a flag byte used to communicate to AQCTL the type of ECB to be posted.

- TSO and standard (TJID=0) - all bits are set equal to zero, except bit 2 (X'20'). The three low-order bytes of the second word contain the address of the TJID or of a halfword that contains binary zeros.

0	X '20'	ECB Address
+4	X '80'	TJID Address

- RORI ECB - bit 1 of the action code byte (X'40') is on, and the low-order three bytes of the first word contain the ECB address. The second word contains the TCB address for the task being posted. Word three contains the address of the DEB associated with the ECB being posted.

0	X '40'	ECB Address
+4	X '00'	TCB Address
+8	X '80'	DEB Address

- To effect cross-partition data movement, the calling routine provides a three-word parameter list. The first word contains the address of the data to be moved. The second word contains the address of the target field of the move, and the third contains the address of a halfword that contains the length in bytes of the data field. Bit 4 of the action code byte is set to one.

0	X '08'	Data Address
+4	X '00'	Target Address
+8	X '80'	Length Address

If the target field is the disabled ready queue and the MCP ECB is to be posted, bit 5 is set equal to 1.

- If the TCB under which the SVC is issued is to be flagged for TSO, bits 3 and 6 of the flag byte are used. If the "eligible for swap" flag is to be set, bit 6 is set to one. If the "not eligible for swap" flag is to be set, bit 3 is set to one.

- When the calling routine wishes to flag a TSO TCB eligible for swap and post an ECB, it builds a three-word list. Bits 2 and 6 must be turned on for this option.

0	Flag	ECB Address
+4		TJID Address
+8	X '80'	TCB Address

- If the calling routine wishes to flag a task eligible fcr rollout and to post an ECB complete, it builds a three-wcrd parameter list. Bits 1 and 7 of the action code tyte are set equal to one.

0	Flag	ECB Address
+4		TCB Address
+8	X '80'	DEB Address

Upon the completion of the AQCTL SVC 102 routine, register 15 contains a return code. Fcr a successful operaticn, the return code is binary zero. If this SVC is issued and there is not an active TCAM MCP in the system, the routine is not executed and the return code is four. In a multiprocessing envircnment, the AQCTL SVC 102 routine turns off the TCBTPSP bit in the TCB tc indicate that the task is again eligible tc be dispatched.

External Routines:

- OS Set Status routine (SVC 79) - to set the TCB status.
- OS Post routine (SVC 2) - to post ECBs complete.
- IKJTSI00 - TSC Interface routine - to flag TSO tasks.
- TESTDSP - OS Task Removal routine - to flag the MCP not eligible to be dispatched.

Tables/Work Areas: CVT, AVT, Time Sharing CVT.

Attributes: Resident.

Post Pending Rcutine (Chart RQ)

Module Name: IGGC19RQ

Entry Point: IGG019RQ - activated by the Rollout/Rollin SVC Routine (IEAQRORI) when there is an OS POST pending for a task that is currently being rolled in. A post pending is indicated by a bit setting (TCBTCPP) in the TCB of an application program.

Functions: This module turns off the "post pending" bit in an application program DEB and passes the address of the ECB for that application prcgram to the OS Post SVC routine to be posted complete.

The Post Pending routine finds the address of the ECB that is to be posted by scanning the TCB DEB chain for the TCAM DEB for which a post is pending. A post pending is indicated in the DEBTAMPP byte in an application prcgram DEB. The DEB field DEBQCBAD points to the Read-ahead QCB in the process entry wcrk area. The Post Pending routine uses an offset from the Read-ahead QCB to locate the ECB in the process entry work area. After the DEB chain is completely examined, the Pcst Pending routine returns ccntrcl to the Rollout/Rollin routine.

External Routine: OS Pcst routine (SVC 2) - to post an ECB complete.

Tables/Work Areas: CVT, AVT, TCB, DEB, process entry work area, PCB.

Attributes: Reentrant, refreshable, supervisor mode.

Leased Receive Scheduler (Chart R3)

Module Name: IGGC19R3

Entry Points:

- IGG019R3 - the Leased Receive Scheduler entry point - activated by the TCAM Dispatcher when the Leased Receive Scheduler STCB is first on the STCB chain of the LCB at the top of the ready queue.
- QEVENT - the QEVENT routine entry point - activated when the TCAM Dispatcher reaches the QEVENT STCB, which is always the last STCB in the chain of schedulers for a line.

Functions: The first function of the Leased Receive Scheduler is to check the "closedown" bit in the AVTBIT1 field in the AVT to determine whether a closedown is in process. If there is a closedown in progress, the routine returns to the Dispatcher indicating that control is to be passed to the next subtask referred to by the STCB chain of the LCB. This causes completion of all receive operations for the line.

The Leased Receive Scheduler inspects the invitation list to determine whether it is active. For an inactive invitation list, the scheduler returns to the TCAM Dispatcher to have the next subtask for the line dispatched.

If there are active entries in the invitation list, the Leased Receive Scheduler determines whether the last entry serviced is the last active entry in the invitation list. If it is not the last active entry, the Leased Receive Scheduler exits to the TCAM Dispatcher to tpost the FRB to the Buffer Request QCB. If the entry is the last active entry in the list, the Receive Scheduler branches to the Time Sharing Scheduler (IEDAYZ) if TSO is active. Upon return, the Receive Scheduler resets the LCB pointer (LCBINVPT) to the first entry in the list and tests for a specified end-of-poll time delay. If a time delay is specified, the Leased Receive Scheduler tposts the LCB to the time delay queue and removes the Leased Receive Scheduler STCB from the STCB chain of the LCB by priority; otherwise, the scheduler tposts the LCB to itself to initiate a receive operation for the first entry in the list.

The QEVENT routine first links to the Time Sharing Scheduler (IEDAYZ) if TSO is active in the system. In this case, the Time Sharing Scheduler initiates a monitor channel program on a time sharing line that can timeout. (A prepare sequence monitors the line for use of the attention key.) Upon return from the Time Sharing Scheduler, the QEVENT routine determines whether there is more activity for the line. If not, this routine marks the LCB for the line "free".

External Routines:

- OS Post routine (SVC 2) - to post the Operator Ccntrol ECB complete.

- IEDAYZ - Time Sharinq Scheduler - to initiate a monitor channel proqram on a time sharinq line.

- IGG019RB or IGG019RO - TCAM Dispatcher - the DSPUNAVR entry point, to exchanqe the scheduler STCBs.

Tables/Work Areas: DCB, LCB, QCB, RCB, STCB, AVT, Terminal Table, Time Sharinq CCB.

Attributes: Serially reusable, refreshable, problem prcqram mode, resident.

Dial Receive Scheduler (Chart R1)

Module Name: IGGC19R1

Entry Point: IGGC19R1 - activated by the TCAM Dispatcher when the Dial Receive Scheduler STCB is the first STCB in the STCB chain of the LCB at the tcp of the ready queue or when a Destination QCB has been tposted to itself as a result of the CLOCK cr INTVAI operand on the INTRO macro.

Functions: The Dial Receive Scheduler initiates receive operations for a dial line and prepares for send operations upon completion of the input.

If a dial line is beinq used by TSO, the Dial Receive Scheduler activates the Time Sharinq Scheduler (IEDAYZ) to schedule operations. In this case, the Time Sharinq Scheduler builds a monitor channel proqram for lines that have the attention feature but time out. The channel prcqram mcnitors the line fcr an attention.

When the input element to the Dial Receive Scheduler is a Destination QCB, one cf two possible conditions exits. Either the OCB is associated with a destination LCB that has just been found by the Send Scheduler or the LCB contains an indication that the connection with a terminal is complete. In the first case, the Send Scheduler has removed the OCB from the time delay queue, has found the associated LCB, and has tposted the QCB tc the Dial Receive Scheduler. The scheduler, at this point, calculates the Termname Table offset for the destination terminal and uses the Termname Table code to get the terminal entry. The scheduler verifies the Destination QCB for the terminal to be dialed by comparinq the input QCB to the Destination OCB specified in the terminal entry. The scheduler then stores the Termname Table offset in the LCB and exits to the TCAM Dispatcher to tpost the ERB tc the Buffer Request QCB to request initial buffers.

When the input element is a Destination QCB and the associated LCB contains an indication that the terminal connection is complete, the Dial Receive Scheduler calculates the period for the time delay between calls and exits to the TCAM Dispatcher to tpost the

Destination QCB to the time delay queue to prepare for the next call to be made.

When the input element to the Dial Receive Scheduler is an LCB and the last operation for the terminal that is presently connected did not result in a negative invitation response, or if there is no current terminal connection, the Dial Receive Scheduler exits to the TCAM Dispatcher to tpost the ERB to the Buffer Request QCB to request initial buffers.

If the last operation resulted in a negative invitation response, the Dial Receive Scheduler scans the dial-out call queue to determine whether there is a message for the connected terminal. (A QCB for a dial terminal is placed in the dial-out call queue when there is a message for an unavailable line.) If there is a message for the terminal, the scheduler removes the associated QCB from the dial-out call queue, moves the Send Scheduler STCB of that QCB to the first of the STCB chain of the LCB for the line, and returns to the TCAM Dispatcher to have the LCB tposted to itself in order to initiate sending to the connected terminal.

If there is no message for the connected terminal, the scheduler scans the dial-out call queue to find the QCB with the highest nonzero priority. For priority messages, the scheduler removes this QCB from the queue and returns to the TCAM Dispatcher to have the Send Scheduler STCB of this QCB chained into the STCB chain for the LCB. If there are only zero priority QCBs on the dial-out call queue, the scheduler uses the first QCB found that has a relative line number that is greater than or equal to the relative line number of the currently connected line.

If the dial-out call queue does not contain a QCB with a nonzero-priority level message or a QCB with a zero-priority level message and a relative line number that is greater than or equal to the relative line number of the current line, the Dial Receive Scheduler returns to the TCAM Dispatcher to tpost the ERB for the line to the Buffer Request QCB to request buffers. This initiates a receive operation.

External Routines:

- IEDQTNT - Termname Table code - to find the address of an entry in the Terminal Table.
- OS Time routine (SVC 11) - to obtain the current time of day.
- IEDQHG - Time Delay subtask - The IEDQHG01 entry point, to put the Destination QCB on the time delay queue.
- OS EXCP routine (SVC 0) - to disconnect a dial line (output only).
- IGG019RB or IGG019RO - TCAM Dispatcher - The DSPUNAVR entry point, to place an STCB at the top of the STCB chain of the LCB.
- IEDAYZ - Time Sharing Scheduler - to build a monitor channel program for time sharing lines.

Tables/Work Areas: DCB, DEB, LCB, QCB, RCB, STCB, AVT, Terminal Table, Time Sharing QCB.

Attributes: Serially reusable, refreshable, problem program mode, resident.

Local Receive Scheduler (Chart 01)

Module Name: IGGC1901

Entry Point: IGG01901 - activated by the TCAM Dispatcher when the special attention element is tposted to this module or when an LCB is tposted to itself with the IGG01901 STCB first on its STCB chain.

Functions: The Local Receive Scheduler schedules receive operations for 2260 Local lines.

The Local Receive Scheduler is activated three times in order to receive a message from a 2260 Local device. An attention interrupt activates the Attention routine (IEDQATTN), which activates the Attention Handler (IGG019R5). The Attention Handler tposts the special attention element to the Local Receive Scheduler QCB. The Local Receive Scheduler processes the special element and then tposts the appropriate LCB to activate itself in order to schedule the receive operation. After the receive operation is complete, the scheduler frees the LCB.

The specific functions of the Local Receive Scheduler depend upon the specific input that causes the Dispatcher to activate this scheduler. If the input is the special attention element, the Local Receive Scheduler first frees the element. Then, if the LCB is busy, the scheduler returns to the TCAM Dispatcher at entry point DSPDISP so that the next subtask on the ready queue can be activated. If the LCB is free, the Local Receive Scheduler tposts the LCB to itself and returns to the Dispatcher (DSPPOST).

If the input is an LCB, the Local Receive Scheduler determines whether a closedown is in progress and if so, returns to the TCAM Dispatcher without scheduling any further operations. If a stop line is in progress, the Local Receive Scheduler tposts the LCB to the Stop Line QCB. If there are no active entries in the invitation list for this line, the scheduler returns to the TCAM Dispatcher at DSPPOST in order to activate the next subtask. If there is an active entry in the invitation list, the Local Receive Scheduler sets the LCB to request buffers and tposts the LCB to the Buffer Request QCB. After a receive operation is complete, the scheduler frees the LCB.

External Routines: None.

Tables/Work Areas: AVT, DCB, LCB.

Attributes: Reentrant, refreshable.

Send Scheduler (Chart R4)

Module Name: IGGC19R4

Entry Point: IGG019R4 - activated by the TCAM Dispatcher when the Send Scheduler STCB is referred to by the buffer, LCB, or QCB at the top of the ready queue.

Functions: The first function of this module is to determine the type of element tc be processed. There are three possible types of elements:

- A buffer
- An LCB
- A QCB for a dial line tpcsted tc itself

If a buffer is tposted to a Destinaticn QCB, the Send Scheduler returns to the TCAM Dispatcher to dispatch the next subtask represented in the STCB chain of the Destination QCB. This subtask is the Destination Scheduler, which assigns a disk or main storage queuing address for the buffer. When the last buffer of a message or the first buffer cf an initiate mode message is handled by the Destination Scheduler and the Send Scheduler STCB is in the STCB chain of the Destination QCB, the Destination Scheduler branches into the Send Scheduler to find a line over which the message can be sent. The search begins with the LCB with the same relative line number specified in the QCB.

If the line is not a dial line, the Send Scheduler uses the LCB indicated by the relative line number in the Destination QCB as the one for the line over which the message is to be sent. If the line is free, the Send Scheduler tposts the LCB tc itself and moves the Send Scheduler STCB from the STCB chain of the Destination QCB to the STCB chain of the ICB. If the line is not free, the scheduler takes any necessary special action, that is, going tc an open list with Auto Poll or issuing an IOHALT macro, and moves the Send Scheduler STCB to the LCB without tposting the LCB tc itself.

If the line is a dial line, the Send Scheduler searches the LCB indicated by the relative line number in the QCB to find the available LCB with the lowest relative line number. If CALL=NONE is specified for the terminal, a line is used only if the terminal is currently connected on a line. The Send Scheduler issues an IOHALT macro (if it is needed) fcr a line that is not connected, and moves the Send Scheduler STCB frcm the STCB chain cf the Destination QCB to the STCB chain of the LCB. If no available or free line is found, the scheduler chains the Destination QCB into the dial-out call queue.

A Send Scheduler dispatched frcm an LCB indicates that a send operation is being initiated. If the line is being used for TSO, the Send Scheduler activates the Time Sharing Scheduler to check for partial line reads or simulated attention reads. (These have priority over output operations.) If there is a quick closedown in progress, return to the Dispatcher activates the subtask pointed to by the next

STCB in the STCB chain of the LCB. Otherwise, if there is a message to send, the Send Scheduler identifies the highest priority message, initializes the LCB for sending, and tposts the ERB, which contains the number of buffers to be sent, to the Disk I/O QCB to initiate sending the message. If a terminal on the line is in locked mode, lock response messages have the highest priority. Otherwise, initiate mode messages and then the priority-FEFO messages have the highest priority. If there is no message to send, the Send Scheduler returns to the Dispatcher indicating that the STCB for the current Send Scheduler is to be returned to the first position in the STCB chain of its Destination QCB.

The function of the Send Scheduler when there is a Destination QCB for a non-TSO dial line tposted to itself is to search for an available line in the line group of the QCB. If there is no line available, the Destination QCB is placed on the dial-out call queue. If there is a line available, the Send Scheduler STCB is linked into the LCB STCB chain of the LCB. In other words, the functions here are the same as when a buffer is tposted and the lines are dial.

If the line is for a TSO operation, the Dial Receive Scheduler activates the Time Sharing Scheduler. The Time Sharing Scheduler determines whether TSO has issued a write break operation request or a simulated attention read request. These should be honored before other TCAM processing resumes.

Upon the completion of any of the above functions, the Send Scheduler returns to the Dispatcher.

External Routines:

- IEDQTNT - Termname Table code - to get the address of an entry in the Terminal Table.

- OS IOHALT routine (SVC 33)- to halt I/O on the lines.

- IGG019RB or IGG019RO - TCAM Dispatcher - the DSPUNAVR entry point, to put the Send Scheduler STCB in the LCB STCB chain; the DSPPOSTR entry point, to free the LCB.

- IEDAYZ - Time Sharing Scheduler - to monitor TSO requests.

- OS EXCP routine (SVC 0) - to start channel activity.

Tables/Work Areas: DCB, LCB, QCB, RCB, STCB, AVT, Terminal Table, Time Sharing QCB.

Attributes: Serially reusable, refreshable, problem program mode, resident.

Send Scheduler for Leased Lines and No TSO (Chart Q6)

Module Name: IGG019Q6

Entry Point: IGG019Q6 - activated by the TCAM Dispatcher when the Send Scheduler STCB is referred to by the buffer, LCB, or QCB at the top of the ready queue.

Functions: The functions of this routine are the same as those for the Send Scheduler (IGG019R4) except that it contains logic for leased lines only and contains no TSO interface logic.

External Routines:

- OS IOHALT routine (SVC 33) - to halt I/O on the lines.

- OS EXCP routine (SVC 0) - to issue an EXCP on break.

Tables/Work Areas: DCB, LCB, QCB, RCB, STCB, AVT, Terminal Table.

Attributes: Serially reusable, refreshable, problem program mode, resident.

Send Scheduler with No TSO (Chart Q7)

Module Name: IGG019Q7

Entry Point: IGG019Q7 - activated by the TCAM Dispatcher when the Send Scheduler is referred to by the buffer, LCB, or QCB at the top of the ready queue.

Functions: The functions of this routine are the same as those for the Send Scheduler (IGG019R4) except that it contains no TSO interface logic.

External Routines:

- IEDQTNT - Termname Table code - to get the address of a terminal entry.

- OS IOHALT routine (SVC 33) - to halt I/O on the lines.

- OS EXCP routine (SVC 0) - to issue an EXCP on break.

Tables/Work Areas: DCB, LCB, QCB, RCB, STCB, AVT, Terminal Table.

Attributes: Serially reusable, refreshable, problem program mode, resident.

Buffered Terminal Scheduler (Chart RD)

Module Name: IGG019RD

Entry Point: IGG019RD - activated by the TCAM Dispatcher when the LCB is on top of the ready queue, by the TCAM Dispatcher when the Time Delay subtask posts the Destination QCB to the Buffered Terminal Time

Delay QCB, and by the Destination Scheduler to move the Buffered Terminal Scheduler STCB from the Destination QCB to the LCB.

Functions: This module schedules send and receive operations for buffered terminals, that is, 2740 Model 2 and 2770. In general, it performs in a manner analogous to the send and receive schedulers for non-buffered terminals. The send function is different because the terminal has a hardware buffer. When a block of text is sent to a buffered terminal, the transmission is complete. However, TCAM must observe a time delay equivalent to the time required for the terminal to empty its buffer onto its output device. This scheduler tries to utilize the line for sending to or receiving from other terminals on the line during the time delay for the terminal to which the last block of text was sent. Flags in the destination QCBSTAT field indicate whether the terminal is in send or receive mode. These states are mutually exclusive.

The Buffered Terminal Scheduler waits on the Destination QCB and the LCB. There is an STCB for each Destination QCB and LCB. When a buffer is tposted to the Destination QCB, the scheduler passes the buffer to the Destination Scheduler (IEDQHM). When IEDQHM recognizes end-of-message, it branches to the TAG subroutine in the scheduler. If the line is free, the scheduler uses DSPPOSTR to tpost the LCB to itself; if not, after return from the Dispatcher, the scheduler links its STCB from the QCB into the LCB STCB chain. This action indicates that there is a message for the associated terminal. If Auto Poll is in progress, it is stopped. The scheduler exits to IEDQHM.

When the LCB is tposted to the scheduler, the scheduler tests for the type of the last operation. If the last operation was a receive, the scheduler tests for scheduling priority. If equal priority is specified, the scheduler tests for end of invitation list. If it is not the end, the scheduler sets up the ERB to receive from the next entry in the list, and tposts the ERB to the Buffer Request QCB.

If it is the end of the invitation list, the scheduler resets the current invitation list pointer to the beginning of the list and tests for something to send to a terminal on the line. If send priority is in effect, the scheduler also makes a send test. If there is nothing to send, the scheduler initiates a receive operation on the first entry in the invitation list. If there is something to send to a terminal on the line, the next scheduler STCB in the LCB STCB chain is dispatched.

If the QCB reflects an empty status, the scheduler removes the STCB from the LCB STCB chain, turns off QCBSEND, and exits to the Dispatcher by tposting the LCB to itself. If the QCB is not empty, the scheduler tests for send status (QCBRECEV=0). If the send status is nonzero, the scheduler reenters the loop for testing for something to send. If the status is not receive, the scheduler turns on QCBSEND, builds the ERB to initiate the sending operation, and exits to the TCAM Dispatcher to tpost the ERB to the Disk I/O QCB.

If the last operation on the line was a send, the scheduler observes a time delay for the destination of the last block. The scheduler stores the time delay interval in the QCB, and removes its

STCB from the LCB STCB chain. The QCB is passed via a BALR to the Time Delay subtask branch entry point (IEDQHG01). Upon return, the scheduler enters the "something-else-to-send" loop.

When the time delay interval expires, IEDQHG tposts the QCB to a QCB at BTSTDQCB in the scheduler CSECT. The scheduler returns its STCB from the QCB to the LCB STCB chain. If the line is free, the scheduler tposts the LCB to itself. The scheduler then exits to the Dispatcher.

External Routines:

- IEDQTNT - Termname Table code - to obtain a terminal entry address.

- IGG019RB or IGG019RO - TCAM Dispatcher - the DSPPOSTR entry point to tpost an element to the ready queue.

- IEDQHG01 - Time Delay subtask - to implement a time delay.

Tables/Work Areas: AVT, SCB, LCB, QCB, DCB, DEB, invitation list, Terminal Table entry.

Attributes: Reentrant, refreshable, problem program mode.

Activate-I/O Generator Subtask (Chart KA)

Module Name: IEDQKA

Entry Points:

- IEDQKA - Activate - when entered from the TCAM Dispatcher.

- IEDQKA02 - I/C Generator - when given control from the Line End Appendage.

Functions: This module builds channel programs for initial contact, continue, and reset sequences.

The Activate-I/O Generator subtask obtains as input the ERB or the buffer for the terminal or device in need of a channel program. This subtask constructs a channel program based upon the characteristics for the device as obtained from the Device Characteristics Table (DCT). This subtask tests the characteristics bits and transfers control to an internal Expand subroutine, passing an offset into a model CCW table. The model CCW table consists of a two-byte entry for each CCW built:

Byte 0 - an offset into an expander table.

Byte 1 - CCW flags for the CCW being built.

The Expand subroutine moves the specified CCW flags into the CCW and utilizes the index in byte 0 of the model CCW table to gain access to information in the expander table. Each entry in the expander table contains the following data:

Bytes 0-1 Offset to a subroutine that establishes address and count

Byte 2 CCWDISAB - CCW OP code

Byte 3 TPDISAB - TP OP code

The Expand subroutine branches to the Expander subroutine indicated by the offset in the expander table entry. This subroutine computes the CCWDATA address and count before returning to the Expand subroutine. Upon regaining control, the Expand subroutine moves the CCW OP code into the CCW and the TP OP code into the LCBTPCD field of the LCB plus an adjustment. Each time a TP OP code is moved to LCBTPCD, the adjustment factor is incremented by one. In this way the TP OP codes for a channel program start at LCBTPCD and continue for as many bytes as necessary.

After the first CCW in the necessary channel program sequence has been built, the Expand subroutine adjusts a register to point to the next entry in the model CCW table. If this is not the last entry, the above actions are again performed to construct the next CCW. Otherwise, the Expand subroutine returns to the in-line code of the Activate-I/O Generator subtask, which places the address of the first CCW to be executed in the LCBSTART field of the LCB. This routine exits to its calling routine.

External Routines:

- IEDQTNT- Termname Table code - to obtain the terminal entry address.

- OS EXCP Routine (SVC 0) - to start a channel program.

Tables/Work Areas: LCB, DCB, CCW, AVT, buffer prefix, SCB, Terminal Table.

Attributes: Reentrant, refreshable, disabled and supervisor mode if entered from Line End Appendage, enabled and problem program mode if entered from the TCAM Dispatcher.

Activate-I/O Generator Subtask for BSC Lines (Chart KA)

Module Name: IEDQKB

Entry Points:

- IEDQKB - Activate - when entered from the TCAM Dispatcher.

- IEDQKA02 - I/O Generator - when given control from the Line End Appendage.

Functions: The functions of this subtask are the same as those for the Activate-I/O Generator except that the data in the model CCW and expander tables is for BSC lines only.

External Routines:

- IEDQTNT - Termname Table code - to obtain the terminal entry address.

- OS EXCP routine (SVC 0) - to start a channel program.

Tables/Work Areas: LCB, DCB, CCW, AVT, buffer prefix, SCB, Terminal Table.

Attributes: Reentrant, refreshable, disabled and supervisor mode if entered from Line End Appendage, enabled and problem program mode if entered from the TCAM Dispatcher.

Activate-I/O Generator Subtask for Start/Stop Lines (Chart KA)

Module Name: IEDQKC

Entry Points:

- IEDQKC - Activate - when entered from the TCAM Dispatcher.

- IEDQKA02 - I/O Generator - when given control from the Line End Appendage.

Functions: The functions of this subtask are the same as those for the Activate-I/O Generator except that the data in the model CCW and expander tables is for start/stop lines only.

External Routines:

- IEDQTNT - Termname Table code - to obtain the terminal entry address.

- OS EXCP routine (SVC 0) - to start a channel program.

Tables/Work Areas: LCB, DCB, CCW, AVT, buffer prefix, SCB, Terminal Table.

Attributes: Reentrant, refreshable, disabled and supervisor mode if entered from Line End Appendage, enabled and problem program mode if entered from the TCAM Dispatcher.

Activate-I/O Generator Subtask for Leased and Start/Stop Lines and No TSO (Chart KA)

Module Name: IEDQKD

Entry Points:

- IEDQKD - Activate - when entered from the TCAM Dispatcher.

- IEDQKA02 - I/O Generator - when given control from the Line End Appendage.

Functions: The functions of this subtask are the same as those for the Activate-I/O Generator except that the data in the model CCW and expander tables is for leased and start/stop lines only and contains no TSO interface logic.

External Routines:

- IEDQTNT - Termname Table code - to obtain the terminal entry address.

- OS EXCP routine (SVC 0) - to start a channel program.

Tables/Work Areas: LCB, DCB, CCW, AVT, buffer prefix, SCB, Terminal Table.

Attributes: Reentrant, refreshable, disabled and supervisor mode if entered from Line End Appendage, enabled and problem program mode if entered from the TCAM Dispatcher.

Activate-I/O Generator Subtask for a QTAM Compatible System (Chart KA)

Module Name: IEDQKE

Entry Points:

- IEDQKE - Activate - when entered from the TCAM Dispatcher.

- IEDQKA02 - I/O Generator - when given control from the Line End Appendage.

Functions: The functions of this routine are the same as those for the Activate-I/O Generator except that it supports only those devices which QTAM supports.

External Routines:

- IEDQTNT - Termname Table code - to obtain the terminal entry address.

- OS EXCP routine (SVC 0) - to start a channel program.

Tables/Work Areas: LCB, DCB, CCW, AVT, buffer prefix, SCB, Terminal Table.

Attributes: Reentrant, refreshable, disabled and supervisor mode if entered from Line End Appendage, enabled and problem program mode if entered from the TCAM Dispatcher.

Line End Appendage (Chart R0)

Module Name: IGG019R0

Entry Points:

- IGG019R0 - activated by IOS when an I/O interrupt occurs with device or channel ending status or by an ERP routine when an error is considered permanent or cleared.

- SCAN - activated by PCI Appendage (IGG019RN) to scan for BSC line control characters.

Functions: The Line End Appendage is a logical extension of IOS and receives control when an I/O interrupt occurs with device or channel ending status, when an error is determined to be permanent by ERP, or when an error is cleared by ERP.

If a permanent error has not occurred, the Line End Appendage examines ending status to determine if ERP is to be scheduled. Unusual ending status results in a return to IOS to schedule the first load of ERP for this device.

When ERP action is not required, the Line End Appendage obtains the TP operation (OP) code for the failing CCW by using the address in the CSW as an index argument into a list of TP OP codes (LCBTPCD). The TP OP code is used as an index into a branch table in order to take specific action for this interrupt. Two tables are employed. One is for normal ending status; the other for error conditions detected by ERP.

For errors that have occurred prior to text transfer, a zero-length buffer is tposted to MH for INMSG/OUTMSG processing.

If an EOB interrupt occurs while receiving a message, a restart is made from the appendage unless MH processing is desired. When EOT is received, the filled buffer(s) is tposted to MH indicating that this is end of message. For an EOB interrupt on output, previous buffers are tposted to the Buffer Return QCB and a restart is accomplished. The last buffer of a successfully sent message is tposted to Buffer Disposition for OUTMSG processing. All continue CCW sequences are built by the I/O Generator routine in IEDQKA.

In the event of a text mode error, the above action is taken except that the buffer reflecting the interrupt is tposted to MH to observe user-selected options.

The SCAN subroutine is entered from the PCI Appendage routine (IGG019RN) as well as from the Line End Appendage. Its address is always at an offset of 4 into the Line End Appendage. OPEN moves this address into the AVTBSCAN field of the AVT so PCI Appendage can have access to it. The SCAN subroutine scans for incoming binary synchronous (BSC) line control characters. The subroutine checks for valid starting and ending characters. If these characters are valid, the Line End Appendage continues reading. If the characters are invalid, the Line End Appendage, upon receiving ending status, exits to IOS to schedule the apprcpriate error recovery procedure (ERP).

The following Line End Appendage functicns are unique tc a TCAM-TSO environment.

1. 2741 Line Ccntrol - A write data does not end with a circle C at the end cf text. This allcws subsequent writes to the same terminal withcut turning the line around. On each interrupt on a 2741, the Line End Appendage sets or clears a switch to indicate whether a write circle C or a write circle D has been completed.

2. Attention Handling/Hangup - If a TSO terminal has an attention key, the Line End Appendage use a prepare command to mcnitor the TSO line for an attention. The PREPARE command has a special TP OP code that causes the Line End Appendage to tpost the ERB to the TSO IOHAIT routine (IEDAYF) when the PREPARE indicates that the operator has struck the attention key. If the operator strikes the attention key while the terminal is receiving data, an interventicn required on a write text CCW occurs and the Line End Appendage tposts the LCB to IEDAYF to issue a PREPARE HIO. If the PREPARE ends normally, the appendage sets the "attention" flag in the SCB and, if there are buffers in use, tposts the buffers to the MH. If there are no buffers in use, the appendage tposts the LCB to the TSC Attenticn routine (IEDAYA). If the intervention required persists, the appendage assumes that the user is hung and tposts the ICB to the TSO Hangup routine (IEDAYH).

3. Write Break to a TSO Terminal - If a TSO terminal has the interrupt feature, priority data can interrupt a receive operation that is in progress. The Line End Appendage recognizes an HIO from the Time Sharing Scheduler (IEDAYZ) and, as a result, initiates a write break to turn the line around. When the break occurs, the appendage restores the CSW to show an end to the read and normal TCAM processing handles tposting the buffers.

4. 2741/1050 Support on One Dial Line - In a TCAM-TSO environment the user can use either a 1050 or a 2741 terminal on a single dial line. After the user has dialed in, the Line End Appendage uses the first interrupt as a signal to decipher the terminal type. If the initial read ends in a time-out with no EOA or nothing received, the appendage assumes a 1050. If the read ends with a time-out with EOA received or ends normally, the appendage assumes a 2741. Further operations for the terminal are based on the setting of the "2741" indicator bit (LCB2741N) in the LCBfield LCBTSCB.

External Routines:

- OS Post routine (SVC 2) - to post the TCAM Dispatcher ECB.

- IEDQTNT - Termname Table code - to obtain a terminal entry address.

- IEDQKA - Activate-I/O Generator subtask - to build a continue sequence cr a channel prcgram.

- TESTDSP - OS Task Removal routine - to flag a TCB not eligible to be dispatched.

- IGG019QO - Line I/O Interrupt Trace routine - to make an entry in the Line I/O Interrupt Trace Table.

Tables/Work Areas: AVT, CCW, DCB, buffer prefix, ICB, QCB, RCB, SCB, Terminal Table.

Attributes: Supervisor mode, disabled, refreshable.

Line End Appendage for BSC Lines (Chart Q2)

Module Name: IGG019Q2

Entry Points:

- IGG019Q2 - activated by IOS when an I/O interrupt occurs with device or channel ending status, or by an ERP routine when an error is considered permanent or cleared.

- SCAN - activated by PCI Appendage (IGG019RN) to scan for BSC line control characters.

Functions: The functions of this routine are the same as those for the Line End Appendage routine (IGGC19R0) except that this routine contains logic for BSC line control only.

External Routines:

- OS Post routine (SVC 2) - to post the TCAM Dispatcher ECB.

- IEDQTNT - Termname Table code - to obtain the terminal entry address.

- IEDQKA - Activate-I/O Generator subtask - to build a channel program.

- TESTDSP-OS Task Removal routine - to flag a TCB not eligible to be dispatched.

- IGG019QO - Line I/O Interrupt Trace routine - to make an entry in the line I/O Interrupt Trace Table.

Tables/Work Areas: AVT, CCW, DCB, buffer prefix, LCB, QCB, RCB, SCB, Terminal Table.

Attributes: Supervisor mode, disabled, refreshable.

Line End Appendage for Start-Stop Lines (Chart Q3)

Module Name: IGG019Q3

Entry Point: IGGC1903 - activated by IOS when an I/O interrupt occurs with device or channel ending status, or by an ERP routine when an error is considered permanent or cleared.

Functions: The functicns of this routine are the same as those for the Line End Appendage routine (IGG019R0) except that this routine contains loqic for start/stop line control cnly.

External Routines: None.

- OS Post routine (SVC 2) - to post the TCAM Dispatcher ECB.

- IEDQTNT - Termname Table code - to obtain the terminal entry address.

- IEDQKA - Activate-I/O Generator subtask - to build a channel proqram. - I/C Generator subtask - to build

- TESTDSP - CS Task Removal routine - tc flaq a TCB not eligible to be dispatched.

- IGG01900 - Line I/O Interrupt Trace routine - to make an entry in the Line I/C Interrupt Trace Table.

Tables/Work Areas: AVT, CCW, DCB, buffer prefix, ICE, QCE, RCE, SCE, Terminal Table.

Attributes: Supervisor mode, disabled, refreshable.

Line End Appendage for Leased and Start/Stop Lines and No TSO (Chart 04)

Module Name: IGG01904

Entry Point: IGGC1904 - activated by IOS when an I/O interrupt occurs with device or channel ending status, cr by an ERP routine when an error is considered permanent or cleared.

Functions: The functions of this rcutine are the same as those for the Line End Appendage routine (IGG019RC) except that this routine contains logic for leased and start/stop lines only and contains no TSO interface lcqic.

External Routines:

- OS Post routine (SVC 2) - to post the TCAM Dispatcher ECB.

- IEDQTNT - Termname Table code - to obtain the terminal entry address.

- IEDQKA02 - Activate-I/O Generator subtask - to build a continue sequence.

- TESTDSP - OS Task Removal routine - to flag a TCB not eligible to be dispatched.

Tables/Work Areas: AVT, CCW, DCB, buffer prefix, LCB, QCB, RCB, SCB, Terminal Table.

Attributes: Supervisor mode, disabled, refreshable.

Line End Appendage for a QTAM Compatible System (Chart Q5)

Module Name: IGG019Q5

Entry Point: IGG019Q5 - activated by IOS when an I/O interrupt occurs with device or channel ending status, or by an ERP routine when an error is considered permanent or cleared.

Functions: The functions of this routine are the same as those for the Line End Appendage routine (IGG019R0) except that this routine contains logic necessary to operate with devices supported by QTAM only.

External Routines:

- OS Post routine (SVC 2) - to post the TCAM Dispatcher ECB.

- IEDQTNT - Termname Table code - to obtain the terminal entry address.

- IEDQKA02 - Activate-I/O Generator subtask - to build a continue sequence.

- TESTDSP - OS Task Removal routine - to flag a TCB not eligible to be dispatched.

Tables/Work Areas: AVT, CCW, DCB, buffer prefix, LCB, QCB, RCB, SCB, Terminal Table.

Attributes: Supervisor mode, disabled, refreshable.

Attention Routine (Chart TN)

Module Name: IEDQATTN

Entry Point: IEDQATTN - activated by IOS when an attention interrupt occurs.

Functions: This module is a resident routine of IOS and receives control when an attention interrupt is presented by the 2848 control unit. Its function is to determine if TCAM is running in the system and to pass control to another module (IGG019R5), which will attempt to schedule a receive operation.

If TCAM is not running in the system, the Attention routine returns to IOS, where the interrupt is ignored.

External Routines: None.

Tables/Work Areas: AVT.

Attributes: Resident, supervisor mode, disabled.

Attention Handler (Chart R5)

Module Name: IGG019P5

Entry Point: IGG019R5 - activated by the Attention routine (IEDQATTN) after an attention interrupt.

Functions: This module searches the DEB chain to determine if a DCB has been opened for this device. This module returns tc IOS with no further action taken if the DCB has not been opened. The Attention Handler schedules receive operation for the device if a closedown is not in progress and the line is not in a stopped state.

External Routines:

- OS Post routine (SVC 2) - to post the TCAM Dispatcher ECB.

- TESTDSP - CS Task Removal routine - tc flag a TCB not eligible to be dispatched.

Tables/Work Areas: AVT, DEB, LCB, DCB.

Attributes: Supervisor mode, disabled, refreshable.

Line I/O Interrupt Trace Routine (Chart Q0)

Module Name: IGGC19Q0

Entry Point: IGG019Q0 - activated by the Line End Appendage (IGG019R0) when it receives an I/O Interrupt and the I/O interrupt trace facility has been requested.

Functions: The Line I/O Interrupt Trace routine makes an entry in the Line I/O Interrupt Trace Table each time that it is activated. One table entry contains the I/O sense information, channel status word, first and failing channel commands with TP OP codes, terminal name or UCB name, and channel and unit address of the interrupt. The Line I/O Interrupt Trace routine then branches to a user trace exit routine if the TREXIT=parameter was specified on the INTRO macro. The user exit routine, in turn, returns to the Line End Appendage. (Note: Registers 1,2,4,11,12,13,14, and 15 must not be changed by the user trace exit routine.)

External Routines: None.

Tables/Work Areas: Line I/O Interrupt Trace Table.

Attributes: Disabled for all interrupts except machine check, supervisor mode, serially reusable.

Time Delay Subtask (Chart HG)

Module Name: IEDQHG

Entry Points:

- IEDQHG - entered from an attached task to place a time delay request element on the Time Delay QCB.
- IEDQHG01 - entered by a BALR from a TCAM subtask to place a time delay request element on the time delay queue.
- IEDQHG02 - entered by a BALR from a TCAM subtask to remove a time delay request element from the time delay queue.
- IEDQHG03 - entered to remove a time delay request element from the time delay queue when a special delete element is tposted to the Delete Time Delay QCB by an attached task.
- TIMEEXIT - entered as a subroutine of the OS Interrupt routine to notify the Time Delay subtask that a specific time of day has arrived.

Functions: This subtask receives elements that request notification upon completion of a specified time interval. The Time Delay subtask maintains these elements on the time delay queue, which is a chain off the link field in bytes 29-31 of the Time Delay QCB. The subtask removes request elements from the queue by deletion requests or when the indicated time interval has elapsed.

When the Time Delay subtask receives a request to place an element on the time delay queue, it issues the OS TIME macro to get the current time of day. The number of seconds in the interval specified in the element is added to the time of day, and the result overlays the interval field in the time delay request element. The Time Delay subtask then places the request element on the time delay queue in order by time of day - from the time closest to the current time to the time farthest away.

When the Time Delay subtask receives a request to immediately remove an element from the time delay queue, the subtask searches the queue for the specified element. If the element is found, it is removed. For a removal request from an attached task, the Time Delay subtask tposts the special delete element back to the indicated QCB. (If the element to be removed is an LCB or a QCB the subtask does not have to search the queue to find out if it is present - there is a bit in the element that indicates whether it is on the queue.)

Once a time delay request element is either added to or deleted from the time delay queue, the Time Delay subtask begins examining the queue for elements that are eligible for removal. Since the elements are on the queue in order by time, each can be examined, in order, and removed if its specified time is equal to or less than the current time of day. To remove an element, the Time Delay subtask tposts the request element to a QCB, the address of which is pointed to by the offset byte cf the request element. To the subtask that requested a time delay, this tpost signifies that the requested time interval has elapsed.

After all request elements eligible for removal have been removed from the time delay queue, the Time Delay subtask issues a STIMER macro for the time of interrupt fcr the first element on the queue and then exits. (The Time Delay subtask issues the STIMER macro for this first element only once. If this same element is still first on subsequent passes through the subtask, Time Delay does not reissue the macro.) This causes the OS Interrupt routine to gain control when that specified time arrives. The OS Interrupt routine issues an interrupt and the OS Supervisor passes control to the TIMEEXIT subroutine.

The purpose of the TIMEEXIT subroutine is to place the Time Delay QCB on the disabled ready queue by tposting it to itself. This is accomplished via the AQCTL SVC 102. (Since the TIMEEXIT subroutine is an enabled interrupt routine, the SVC must be used to place an element on the disabled ready queue.) The QCB is given an extremely high priority so that when the TCAM Dispatcher regains control and merges the ready queues, this element will probably be on top to activate the Time Delay subtask.

When the Time Delay subtask gains contrcl and its own QCB is on the ready queue, the subtask immediately starts examining the time delay queue tc remove elements. Unless it has already been removed by a special delete request, the first element on the queue is the element for which the STIMER macro was issued. The Time Delay subtask examines and removes elements from the queue, as described previously, until it reaches either an element that requires an STIMER macro or the end of the queue.

The Time Delay subtask exits to the TCAM Dispatcher. If it was entered by a BALR instruction, the subtask branches to the address in register 14; otherwise, it branches to the DSPDISP entry point of the Dispatcher.

The format of the time delay request element is as follows:

Offset			
0	Time Delay QCB Address (in the AVT)		
+4	Priority	Dispatcher Link Field	
+8	Reserved		
+12			
+16	Time Interval (In Seconds)	Offset Byte	Reserved
+20	Reserved		Recycle Flag Byte
+24	Reserved		
+28	Reserved	Time Delay Queue Link Field	

The offset byte is the offset into an element to a word of which the low-order three bytes is the address of the QCB to which the element is to be tposted upon expiration of the time delay interval. This offset byte is equal to X'00' if the request element is a QCB, X'14' if the request element is an LCB, or X'08' if the request element is a special element, a buffer, or from checkpoint. The recycle flag byte has only one bit defined: if bit 7 is on and this is a Destination QCB, IEDQHG has issued a request to recycle this element on the time delay queue for an additional 12 hours. The flag byte has only one bit defined: if bit 6 is on, the element is on the time delay queue; if bit 6 is off, the element is not on the queue. This bit in the flag byte is set for all elements on the time delay queue, but it may be checked only for the LCB, QCB, or checkpoint. In other elements, this field has other definitions.

The presence of two link fields in the time delay request element allows the element to be on the time delay queue and tposted to another subtask simultaneously. The link field in bytes 5-7 is used when the element is on the TCAM Dispatcher ready queue. The link field in bytes 29-31 is used when the element is on the time delay queue.

The time delay request element is tposted to the Time Delay QCB in the AVT when an attached task needs to implement a time delay. When a TCAM subtask needs to request a delay, it passes the address of the element in register 1.

The Time Delay QCB occupies the first three words of the time delay element at AVTDELYB in the AVT. At assembly time the time delay queue link field of this element points back to the element itself. As request elements are received, the Time Delay subtask chains the elements to form the time delay queue by using the link address field in bytes 29-31 of each request element. The last element of the chain always points back to the time delay element in the AVT.

The format of the special delete element is as follows:

Offset		+1
0	Reserved	Delete from Time Delay QCB Address(in AVT)
+4	Priority	Dispatcher Link Field
+8	Reserved	Address of Element to be Removed from Time Delay Queue
+12	Reserved	Address of QCB to Receive this Element after the Request for Removal is Serviced

The special delete element is tposted to the Delete from Time Delay QCB (AVTCPRMB) in the AVT when an attached task needs to remove an element from the time delay queue. When a TCAM subtask needs to remove an element, it passes the address of the element to be removed in register 1 and then issues a BALR to IEDQHG02.

External Routines:

- IEAQRT00 - CS Time SVC Routine (SVC 11) - to get the current time of day.
- IEAQST00 - CS TTIMER and STIMER Routine (SVC 47) - to request a time delay.
- IGC102 - AQCTL SVC 102 Routine - to tpost the Time Delay QCB to itself when a specified time interval elapses.
- IGG019RB or IGG019RO - TCAM Dispatcher - the DSPPRIOR entry point, to move the scheduler STCB; the DSPPOSTR entry point, to put an element on the ready queue.

Tables/Work Areas: AVT, QCB.

Attributes: Refreshable, reusable, resident, problem program mode.

System Delay Subtask (Chart HI)

Module Name: IEDQHI

Entry Point: IEDQHI - activated by the TCAM Dispatcher when the Operator Control control module (IEDQCA) receives a request for a system delay.

Function: This module causes the system to cease line activity for the number of seconds specified on the INTVAL=integer operand of the INTRO macro. The System Delay subtask stops line activity by holding all the LCBs on a system delay queue and then placing a request for the specified delav interval on the time delay queue. When the delay has elapsed, the System Delay subtask frees each ICP to reactivate the lines.

The System Delay subtask receives control from the TCAM Dispatcher when one of three possible elements is on the ready queue and tposted to the System Delay QCB. The type of element indicates the phase of processing in which the System Delay subtask is currently operating. The three types of elements are:

1. The system delay request element
2. An LCB
3. The System Delay QCB

The MODIFY INTERVAL=SYSTEM operator control command causes the operator control Change Interval Type routine (IEDQCZ) to tpost the system delay request element (in the Operator Control AVT) to the System Delay QCB. This request element has a PRISYSDL priority (see the TPRIOR macro list in Appendix C), and this particular element is identified by the System Delay subtask by that unique priority. Upon recognizing the system delay request element, the System Delay subtask sets the delay bit, AVTDLAYN in AVTBIT1 and initiates deactivation of the line activity on each non-dial ICB in the TCAM system. The subtask finds the LCBs by tracing the DEB to DCB to ICB to LCB chain of pointers. The System Delay subtask stops line activity by putting any free LCB on the system delay queue, by issuing an IOHALT macro on the active contention lines, and by modifying the channel programs to cause I/O interrupts on the active Auto Poll lines. As each LCB is examined, the System Delay subtask increments a counter, which at the end of the operation is equal to the total number of active LCBs in the system.

When an I/O interrupt on an LCB occurs, the Line End Appendage gains control. The Line End Appendage branches to the appropriate receive scheduler, which tposts the LCB to the System Delay QCB when the AVT system delay bit is on. In this way, each ICB, as its line activity stops, is tposted to the System Delay QCB. The System Delay subtask, upon being activated by an LCB, chains the LCB on the system delay queue and decrements the LCB counter by one. Then, if the ICB counter is not equal to zero, the subtask exits to the TCAM Dispatcher. When the LCB counter is equal to zero, every non-dial LCB in the system is on the system delay queue. At this point the System Delay subtask places the System Delay QCB on the time delay queue to start timing the system delay.

After the Time Delay subtask (IEDQHG) has observed the specified interval, it puts the System Delay QCB on the ready queue. This causes the TCAM Dispatcher to activate the System Delay subtask with the System Delay QCB as its element. In this case, the System Delay subtask clears the system delay bit and reactivates the system line activity by tposting each LCB to itself. The subtask exits to the TCAM Dispatcher, which places all the LCBs on the ready queue.

External Routines:

- IEDQHG01 - Time Delay subtask - to add an element to the time delay queue.

- IEDQHG02 - Time Delay subtask - to remove an element from the time delay queue.

- OS IOHALT routine (SCV 33) - to stop the line.

- IGG019RP or IGG019RO - TCAM Dispatcher - the DSPPRIOR entry point, to move the scheduler STCB.

- OS WTO routine (SVC 35) - to write a message.

Tables/Work Areas: QCB, LCB, AVT, DCB, DEB, ICB.

Attributes: Reusable, resident, problem program mode, nonrefreshable.

Stop Line I/O Subtask (Chart HK)

Module Name: IEDQHK

Entry Point: IEDQHK - activated by the TCAM Dispatcher when the Resident Operator Control module (IEDQCA) receives a request to stop the activity of a line or line group.

Functions: This subtask provides the I/O handling that is necessary to effect a stop line function. The Stop Line I/O subtask stops line activity by freeing the LCB(s) for the indicated line or line group.

The VARY OFFTP (C or I) and the HALT operator control commands cause the Resident Operator Control module to activate the Stop Line routine (IEDQCV). The Stop Line routine builds a stop line request element in the operator control work area and tposts this element to the QCB for the Stop Line I/O subtask. The LCB address is in the request element.

When the stop line request element is the highest priority element on the ready queue, the TCAM Dispatcher activates the Stop Line I/O subtask. This subtask gets the associated LCB address from the request element.

If an LCB is on the time delay queue, the subtask removes the LCB from that queue and places the address of the stop line request element in the LCB. If the LCB is already free (tposted to itself), this subtask set the LCB status byte (LCBSTAT1) to zero and then branches to the exit code.

If an LCB is not free, the Stop Line I/O subtask turns on the non-immediate bit in the LCB status byte. For an Auto Poll LCB, the subtask puts a NOP in the channel program and branches to the exit code; for a BSC LCB with no prepare on the line, the subtask branches to the exit code. If the result of a test-and-set on LCBTSTSW is zero or if the line is busy and is sending, the subtask issues an IOHALT before branching to the exit code. In all cther cases the subtask branches to the exit code.

When the Stop Line I/O subtask has modified a channel program or issued an IOHALT to stop line activity, the subtask is reactivated to complete the prccessing of the LCB for that line. When an I/O interrupt occurs on an LCB, the Line End Appendage gains ccntrol. The Line End Appendage branches to the apprcpriate receive scheduler, which upon finding the non-immediate bit cn in the LCB activates the Stop Line I/O subtask. In this case, the Stop Line I/O subtask examines the ICB to determine the type of line. For a dial line on which an EXCP has already been executed, the subtask turns off the LCB EXCP byte and branches to the exit code. When an EXCP has not been executed on a dial line, this subtask turns on the EXCP byte, sets a DISABLE in the channel program area, clears LCBTTCIN, turns on the test-and-set switch, sets the negative respcnse to poll bit, issues an EXCP, and then exits to the TCAM Dispatcher.

External Routines:

- OS EXCP rcutine (SVC 0) - to start channel activity.
- OS IOHALT rcutine (SVC 33) - tc stop a line.

Tables/Work Areas: AVT, LCB, CCW, DCB, DEB, ICB.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mcde.

MESSAGE HANDLING - BUFFER MANAGEMENT MODULES

Buffer Management Module (Chart GA)

Module Name: IEDQGA

Entry Points:

- IEDQGA - Buffer Request rcutine - to handle a buffer request from the TCAM Dispatcher.
- IEDQGB - Buffer Return routine - to handle a returned buffer from the TCAM Dispatcher.
- IEDQGD - Buffer Association routine - to handle buffer association at the end of OUTBUF processing in an MH and from IEDQGA and IEDQGB.

Functions: The Buffer Management module performs three different functions, and in each case the output is different. The functions are discussed here according to entry point.

- Buffer Request

If the Buffer Management module is entered at the IEDQGA entry point, the Buffer Request routine either assigns the requested buffers or queues the request to be satisfied later. The buffer request is in the form of an ERB pointed to by register 1. There are four types of requests that can arrive, and each is handled as follows:

1. Initial request from a line - if units are available, they are chained together to form the requested number of buffers. CCWs are built for each unit, and the ERB with the buffers chained from it is tposted to the Activate QCB. If units are not available, the ERB is placed in the element chain of the Buffer Return QCB by priority.

2. ERB from an application program or operator control - if units are available, they are chained to form the requested number of buffers and the ERB is tposted to a specified QCB. If units are not available, the ERB is placed in the element chain of the Buffer Return QCB by priority.

3. First PCI request - if units are available, they are chained into the requested number of buffers. CCWs are built in each unit, and the buffers are available for I/O. If units are not available, the ERB is placed in the element chain of the Buffer Return QCB by priority.

4. Subsequent PCI request - the ERB is chained by priority into the element chain of the Buffer Return QCB.

If the routine has an ERB to tpost, it returns to the DSPPOST entry point of the TCAM Dispatcher. If the ERB is to be inserted into the element chain of the Buffer Return QCB, exit is to the DSPPRIO entry point of the TCAM Dispatcher. Otherwise, Buffer Management returns to the DSPDISP entry point of the TCAM Dispatcher.

- Buffer Return

If the Buffer Management module is activated at the IEDQGB entry point, its function is to return buffers to the buffer unit pool. The handling of the buffers depends on whether there is an ERB waiting in the element chain of the Buffer Return QCB.

1. If there is no ERB waiting for a buffer, the units that make up the buffer are placed in the Buffer Request QCB element chain (the buffer unit pool).

2. If an EPB is waitinq for a buffer, the necessary number of units are chained toqether tc fcrm one buffer. If the ERB has a low priority (nct initial, first PCI, or disk request), CCWs are built for each unit cf the buffer and the tuffer is included in the channel prcqram for the lire. If the tuffer request was fully satisfied, the ERB is dropped from the Buffer Return QCB element chain: otherwise, the ERB is rechained ty priority. If the ERB has a hiqt priority, action is perfcrmed as described in the Buffer Request discussion.

Exit from the Buffer Return porticn of the Buffer Manaqement module is handled exactly as discussed under Buffer Request.

- Buffer Asscciation

If the Buffer Manaqement module is activated at the IEDQGD entry point, its function is to build CCWs for data transfer in each unit of a buffer. All units of the tuffer(s) have READ or WRITE and TIC CCWs built in the first three words of the unit. If the request is other than an initial request for receivinq, the tuffer(s) is included in the channel proqram for the line.

The Buffer Association routine of Buffer Manaqement exits to the routine that called it in the case of Buffer Request or Buffer Return; or to DSPDISP in the TCAM Dispatcher if activated by the OUTMSG macro expansion in MH.

External Routines:

- IGG019RB or IGG019RO - the TCAM Dispatcher - inserts by priority either on the ready queue (DSPPOSTR) or cn the element chain of a specified QCB (DSPPRIOR): or puts the unit first on the buffer unit pcol (DSPLIFOR):

- OS EXCP routine (SVC 0) - to start channel activity.

Tables/Work Areas: DCB, tuffer prefix, LCB, QCB, AVT.

Attributes: Resident, enabled, refreshable, reusable.

Transparent Transmissicn CCW Buildinq Routine (Chart GT)

Module Name: IEDQGT

Entry Point: IEDQGT - activated by a branch at the end of OUTBUF processinq when a messaqe is to be sent in transparent mode.

Functions: This routine builds in each tuffer unit the CCWs that are necessary to send transparent data in transparent mode in the correct block size tc a terminal. This routine also constructs in the LCB a sequence to write DLE/ETB and to read respcnse.

In each unit the Transparent Transmission CCW Building routine places a CCW to write the first portion of the unit in the block that is to include the unit. The unit that contains the last byte of data of the first block in a transmission builds a TIC command to the LCB CCW to write the DLE/ETB sequence. The LCB channel program area contains the values for the number of bytes left to write in the current unit, the address of the unit, and the value formerly in the TIC field of the unit. If the current block is not the first block of the transmission, this routine places a flag to indicate this situation in the unit that contains the last byte of data for the current block and places the number of bytes left to write from this unit in the Write CCW OP code area.

If all the units to write out the next block are not available at read response time, Line End Appendage treats the condition like a channel program check and makes the channel execute a write sync loop that writes SYNC characters on the line. When the required units are available, normal transmission is resumed - the write sync loop TICs to a write DLE/STX sequence and to the next unit to be transmitted.

After the necessary CCWs are built, the Transparent Transmission CCW Building routine branches back to the calling routine.

External Routines: None.

Tables/Work Areas: AVT, LCB, SCB, buffer prefix, CCW, DCB.

Attributes: Reusable, refreshable, enabled, resident, problem program mode.

PCI Appendage (Chart RN)

Module Name: IGGC19RN

Entry Point: IGG019RN - entered from IOS when a program-controlled channel interruption occurs. When the PCI flag in the CCW is on and an interrupt results, PCI Appendage gains control.

Functions: The PCI Appendage frees buffers from the line operation just completed and, if ADD is specified in the line group DCB, obtains additional buffers.

When a PCI interrupt occurs on receiving, the Appendage locates the CCW on which the PCI occurred, and if the PCI is for the first buffer for a read, PCI Appendage checks the ID sequence in the Terminal Table entry (TERMID), if applicable. For BSC terminals the appendage checks the buffer for transparency by executing a BALR to a subroutine in the Line End Appendage.

On an initial PCI no buffers have been processed by the channel program. If this is not an initial PCI on sending, PCI Appendage tposts the buffers that have already been processed by the channel program to the Buffer Return QCB. If this is not an initial PCI on receiving, the buffers already processed are tposted to MH. If ADD is specified in the line group DCB, PCI Appendage (for both initial and subsequent PCIs on sending or receiving) requests additional buffers from the appropriate QCB (Buffer Request or Disk I/O). If the ERB is already tposted, the count of requested buffers is increased.

When PCI Appendage completes its functions, it posts the MCP complete and exits to IOS.

External Routines:

- IGG019R0 - Scan subroutine of the Line End Appendage - to check a buffer for a BSC terminal for transparency.
- TESTDSP - OS Task Removal routine - activated when TCAM is operating in a multiprocessing environment to stop the MCP from executing in the other active CPU.

Tables/Work Areas: CCW, DCB, AVT, buffer prefix, LCB, Termname Table, Terminal Table entry.

Attributes: Reusable, refreshable, disabled, resident, supervisor mode.

MESSAGE HANDLING - CONTROL ROUTINES

User Interface Routine (Chart UI)

Module Name: IEDQUI

Entry Point: IEDQUI - called from an MH macro expansion or from a functional MH routine.

Functions: The User Interface routine is the common module through which MH macro expansions link to functional MH routines. This routine saves the user registers, initializes general register contents to commonly needed values, finds the address of the routine to be linked to, and exits to it.

The User Interface routine is also used by certain functional MH routines to provide initialization and linkage to lower-level MH routines.

In conjuction with the Return Interface routine (IEDQLM), User Interface provides level-independent register protection between MH levels.

This routine performs the following functions:

- Saves reqisters 2 through 12 and 14 in the save area pointed to by reqister 13.

- Gets the AVT address from the CVT. The routine then finds address of the current buffer in the AVT field AVTADBUF; the address of the LCB in the PRFLCB field of the buffer prefix; and the address of the current SCB in the LCBSCBA field of the LCB. (If entry is to the Binary Search routine, the User Interface routine only gets the AVT address).

- Gets the address of the MH VCON table from the AVTMSGS field of the AVT. The User Interface routine takes the index to the address of the routine from the first byte of the input parameter list, adds the index value to the MH VCCN table address (AVTMSGS), and places the result in register 12. The User Interface routine then exits to the address in register 12.

External Routines: None.

Tables/Work Areas: AVT, CVT, buffer to be processed, LCB, SCB, MH VCON Table.

Attributes: Reentrant, refreshable, enabled, resident, problem program mode.

Return Interface Routine (Chart LM)

Module Name: IEDQLM

Entry Point: IEDQLM - called by a functional MH routine to return to the calling routine.

Functions: The Return Interface routine is the common module through which functional MH routines return to MH. This routine restores registers 2 through 12 and 14 from the save area pointed to by register 13, and exits to the address in register 14.

The Return Interface routine is also used to return to certain functional MH routines from lower-level routines.

External Routines: None.

Tables/Work Areas: None.

Attributes: Reentrant, refreshable, enabled, resident, problem program mode.

STARTMH Subtask (Chart AA)

Module Name: IEDQAA

Entry Point: IEDQAA01 - activated by the TCAM Dispatcher when the STARTMH macro is coded in an MH after the Dispatcher tposts a buffer to the STARTMH QCB.

Functions: This subtask performs the initialization functions required by the message handler to process messages.

Upon entry, the STARTMH subtask places the address of the buffer just tposted by the Dispatcher into the AVTADBUF field in the AVT. If the 'cancel' flag is set in the prefix status byte, PRFCNCLN in PRFSTAT1, the subtask branches to the input text processing portion of the subtask at the TEXT label.

If entry is to be send side of MH, provided the terminal is in lock mode, the subtask turns off the prefix 'lock' bit and increments by one the count of the outstanding lock requests in the LCB. Next, the STARTMH subtask determines whether the buffer is a header or a text buffer. The subtask also determines whether the buffer has been received or is to be sent and branches to the appropriate section of the subtask.

The portion of the subtask that processes header input buffers first initializes the prefix origin field (PRFSRCE) from the LCB. The subtask then cleans the prefix sequence-in field (PRFSEQIN), the SCB priority field (SCBPRI), and the SCB cutoff count field (SCBBKFCT) to zeroes. The subtask initializes the prefix scan pointer (PRFSCAN) to point to the last byte in the prefix or, if reserve characters are used, to the last reserve character. The subtask next clears the SCB destination QCB field (SCBDESTQ) to zero, and if the origin is an application program, branches to the MH via the exit portion of the subtask at the EXIT label.

On the other hand, if the origin is defined as an EOA sequence, the STARTMH subtask branches to the Skip Forward and Scan routine (IEDQAI) to determine whether an EOA sequence is in the buffer. On return, if there is an EOA sequence, the subtask increments the scan pointer to point to the last byte of the buffer. If the source is an IBM 1030 or an IBM 2260 Remote terminal, the subtask increments the pointer by one byte to point to the addressing character that follows the EOA sequence.

If the terminal is in lock mode, the subtask turns on the prefix 'lock' bit, gets the index to the destination from the LCB, and places it in the prefix destination key field (PRFDEST). The subtask sets flags in the LCB to indicate a tpost is pending and passes the destination key to the Termname Table code (IEDQTNT), which returns the address of the Terminal Table entry. The STARTMH subtask then moves the Destination QCB address from the Terminal Table entry to the SCB Destination QCB field and exits to the MH via the EXIT portion of the subtask.

If the terminal is not in lock mode, the STARTMH subtask next examines the LCB to determine whether there is a STOPLINE request currently pending on the line. If so, the subtask bypasses the On-Line Test processing and exits to the MH via the EXIT label. Otherwise, the subtask performs the following On-Line Test processing.

The STARTMH subtask checks a bit in the LCB to determine whether the line is binary synchronous or start/stop. For start/stop lines the subtask branches to the Skip Forward and Scan routine (IEDQAI) to ascertain whether an On-Line Test sequence is in the buffer. On return, if this is an On-Line Test message, the subtask determines whether On-Line Test support is in the TCAM system, whether the On-Line Test maximum load has not been reached, and whether the test request message is only one buffer long. If any one of these three conditions is not true, the subtask indicates an error in the SCB and exits to the MH via the EXIT label. If all of these conditions are found, the subtask sets in the buffer a byte to identify the origin as binary synchronous or start/stop and sets in the LCB a 'no source' indication (AVTEFF in LCBTSTSW) and a negative response flag (LCBNEGRP in LCBSTAT2). Then the subtask posts the buffer to the On-Line Test QCB via a branch to the DSPPOST entry point in the TCAM Dispatcher.

The portion of the STARTMH subtask that processes text input first initializes the prefix origin field (PRFSRCE) from the LCB. The subtask then gets from the DCB (DCBRESER+1) the number of reserve characters and places that number in the LCB (LCBSIZE). The subtask initializes the scan pointer to point to the last byte of the prefix or, for reserve characters, to the last reserve character. If the buffer has a length of zero, the subtask exits to MH via the EXIT label. Otherwise, the subtask determines whether translation is to be performed and if so, branches to the Translate Buffer routine (IEDQAW) to translate the buffer. On return, the subtask puts the contents of the SCB multiple-buffer-header-entry field (SCBMBHEN) into register 1, decrements it by one, and examines the result. If register 1 contains zero, the subtask returns to the MH via the EXIT label. If the contents is not zero, the subtask gets the address of the User Interface routine (IEDQUI), sets a negative value in register 0, and exits to MH via the MBHEXIT label.

The portion of the STARTMH subtask that processes an output header buffer updates the FEFO pointer in the Destination QCB (QCBFFEFO) from the FEFO pointer in the SCB (SCBFEFO) and turns off the 'currently sending' flag in the QCB. If the destination is in lock mode, the LCB is in initiate mode, or the buffer is zero-length and the send error is a text transfer error, the subtask does not update the QCB.

If the destination is in lock mode, the subtask determines whether both the 'lock' and 'extended lock' bits in the SCB are on and if they are, turns them both off. The subtask gets from the prefix scan pointer field (PRFSCAN) the number of reserve characters in the buffer and places that number in the LCBISZE field of the LCB. The subtask sets this field to zero for a zero-length buffer. Next, the subtask initializes the scan pointer to point to the last byte of the prefix or, if reserve characters are present, to the last reserve character.

If the buffer has a length of zero, the STARTMH subtask returns to MH via the EXIT label. For a positive-length buffer, the subtask branches to the Termname Table code (IEDQTNT) to get the address of the Terminal Table entry for the destination. On return, the subtask increments the output sequence number by one and returns to MH via the EXIT label.

The portion of the subtask that processes output text buffers sets the reserve characters count in the LCB tc zerc, initializes the scan pointer to pcint to the last byte in the prefix, and branches to the text input processinq porticn of the subtask to check for a zero-lenqth buffer.

On a normal entry, the exit part of the subtask, at the EXIT label, first sets reqister 0 to zero. Cn a multiple-buffer-header entry, at the MBHEXIT label, reqister 0 already contains zero and is not chanqed. Next the subtask places X'1000' in reqister 2 and computes the messaqe handler entry address. The subtask checks reqister 0 and if it is neqative, branches to MH with a return code of X'04' in reqister 15. Otherwise, the subtask determines frcm the LCB whether the line is sendinq or receivinq and places a X'01' or X'08' return code, respectively, in reqister 15.

External Routines:

- IEDQUI- User Interface routine - to reenter an uncompleted routine and to activate the followinq modules:
- IEDQAI - Skip Forward and Scan routine - to search for an EOA sequence in the buffer.
- IEDQAW - Translate Buffer routine -if required tc translate the buffer.
- IEDQTNT- Termname Table code - to qet a terminal entry address.

Tables/Work Areas: AVT, buffer currently beinq processed, SCB, LCB, DCB, MH VCON Table, QCB, Termname Table, Terminal Table.

Attributes: Serially reusable, refreshable, enabled, resident, problem proqram mcde.

Incominq/Outqoinq Messaqe Delimiter Routine (Chart A4)

Module Name: IEDQA4

Entry Points: IEDQA401 - from the INMSG macro expansicn to tpost the buffer tc the proper QCB, or from the OUTMSG macro expansicn to pass the buffer to either the Buffer Association routine (IEDQGD) or the Transparent CCW Buildinq routine (IEDQGT).

Functions: The functicns of this module differ accordinq tc whether it is activated from the macro expansion of the INMSG or of the OUTMSG macro.

The format of the input macro-qenerated parameter list pointed to by reqister 1 is as follows:

Offset 0	+1
Index to IEDQA4	Parameter List Length

The Incoming/Outgoing Message Delimiter routine first determines whether the current buffer is the last buffer of a message and, if it is, stores the address of the INMSG/OUTMSG parameter list in the SCBMACR field of the SCB. If the PRFDUPLN bit of the PRFSTAT1 field is indicating that the buffer is a duplicate-header buffer, the routine turns off the PRFDUPLN bit and branches to its input or output processing section, for receiving or sending functions, respectively.

If the buffer is a header buffer but not a duplicate-header buffer, the Incoming/Outgoing Message Delimiter routine places the address of the scan pointer in the SCB (SCBMBSSA+3). If the scan pointer offset is less than 255, the routine places the true offset in the SCB: otherwise, it places a default value of 255 there. If the scan pointer indicates a location beyond the end of the last buffer, the routine sets the 'incomplete header' bit (SCBHDRRN in SCBERR1) in the SCB and branches to its input or output processing section. The routine does not set the 'incomplete header' bit for TSO buffers.

- Input Processing

The Incoming/Outgoing Message Delimiter routine turns off the prefix 'cancel' bit (PRFCNCLN in PRFSTAT1) and then checks the SCB destination queue field (SCBDESTQ) to determine whether a destination has been found for this message. If the field contains zeroes, no destination has been found. In this case, the routine places the address of the Buffer Request QCB in the SCBDESTQ field and bypasses multiple-route processing.

If a destination has been found, the Incoming/Outgoing Message Delimiter routine performs multiple-route processing, provided the buffer is a header buffer and the SCBMRFSD field in the SCB contains a character stored in it by the forward function. When these two conditions are met, the routine links to the Address Finder routine to get the address of the start of data in the buffer, then it enters a loop that links repeatedly to the Buffer Step routine (IEDQAX) to scan the buffer for the X'DF' character. On return, if the character is found, the routine replaces it with the character stored in the SCBMRFSD field and places the offset to the character in the SCBMRFSD field. If the X'DF' character is not found, the routine clears the SCBMRFSD field to zeros, thereby concluding the multiple-route processing.

If the buffer is not the final buffer of the message, or if the logical end-of-message indicator is not set in the TIC field, the Incoming/Outgoing Message Delimiter routine moves the Destination QCB address from the SCBDESTO field to the first word of the buffer and places a priority of X'FS' for a header buffer or X'E4' for a text buffer in the buffer. If the buffer is the final buffer for a message, or if the logical end-of-message indicator is set, the routine places the address of the Buffer Disposition OCB in the first word of the buffer and sets the priority to X'DF'.

If this is not a TSO buffer, the Incoming/Outgoing Message Delimiter routine gets from the LCBISZE field in the LCB the number of reserve characters remaining in the buffer and puts that number in the prefix scan pointer field (PRFSCAN). The the routine tposts the buffer via the TCAM Dispatcher at the DSPPOST label. For a TSO buffer, the routine exits immediately to the Dispatcher (DSPPOST) to tpost the buffer.

- Output Processing

For output processing,the Incoming/Outgoing Message Delimiter routine first determines whether the buffer has a length of zero. If it does, the routine tposts the buffer to the Buffer Disposition QCB by branching to the DSPPOST entry point in the TCAM Dispatcher.

If the buffer does not have an indicated length of zero, the Incoming/Outgoing Message Delimiter routine removes from the end of the buffer all units that do not contain data. To determine whether there are any empty units at the end of the current buffer, the routine passes the offset to the last byte of data in the buffer, gotten from the PRFSIZE field, to the Address Finder routine (IEDQAL). The Address Finder routine returns the address of the unit in which the last byte of data is located. Then the Incoming/Outgoing Message Delimiter routine checks the TIC field of this unit to determine if it is the last unit. If this is the last unit of the buffer there are no empty units at the end of the current buffer.

If there are empty units, the Incoming/Outgoing Message Delimiter routine enters a loop that follows the chain of units, from the last unit that contains data to the last empty unit, counting the empty units. When the routine finds the last empty unit, it resets the 'number of units' field in the buffer prefix (PRFNBUNT) to indicate only the number of units containing data. It then resets the TIC field of the last data unit to indicate that it is the last unit of the buffer, thereby removing the empty units from the buffer.

The chain of empty units is now considered a separate buffer. The Incoming/Outgoing Message Delimiter routine places the number of empty units into the PRFNBUNT field of the first empty unit, puts the address of the Buffer Return QCB into the first word of the first empty unit, sets a priority of X'E4' and tposts the empty buffer by branching to the DSPPOSTF entry point in the TCAM Dispatcher.

The Incoming/Outgoing Message Delimiter routine examines the Destination QCB, the address of which is in the SCBDESTQ field of the SCB, to determine whether it is a QCB for an application program. If it is an application program QCB, the routine tposts the buffer to the Read-ahead QCB, the address of which is in the PFRAQCB field in the process entry work area. The address of the process entry work area is in the TRMSTAT field in the Terminal Table entry for the application program. The address of the Terminal Table entry is in the QCBPRFN field of the application program QCB. The routine places the address of the Read-ahead QCB in the first word of the buffer and sets a priority of X'DC'. Then the routine passes in the PRFSCAN field of the buffer prefix the number of reserve characters remaining in the buffer and tposts the buffer via an exit to the DSPPOST entry point of the TCAM Dispatcher.

If the destination is not an application program, the message is to be sent to a terminal. If the header buffer of the message contains a hardware EOA indication, the routine logically removes the EOA before sending the message. If the buffer is not a header buffer, if the MSGFORM function has inserted line control, or if this is a TSO buffer, no EOA indication is present. For inserted line control, an STX character may be present and is left in the buffer.

The Incoming/Outgoing Message Delimiter routine passes to the Termname Table code (IEDQTNT) the key to the destination Terminal Table entry and, on return, receives the address of the Terminal Table entry. The routine gets the Device Characteristics Table index from the TRMCHCIN field in the Terminal Table entry. The routine uses this index to find the entry in the Device Characteristics Table for this destination terminal, from which it can determine the specific device. If the device is an IBM 2260 Remote or an IBM 2760 in tete-a-tete mode, no EOA indication is present. A STX character is present and is left in the buffer.

At this point, the Incoming/Outgoing Message Delimiter routine gets the entry in the Special Characters Table for this device from the DCBSCTAD field in the DCB. If this field contains zeros, there is no Special Characters Table entry for the destination. Therefore, no EOA indication is defined for the destination and no EOA is present. If there is a Special Characters Table entry for the destination, the Incoming/Outgoing Message Delimiter routine gets the index byte for an EOA indication from the entry and determines whether the byte is zero. If it is zero, no EOA is defined for the destination, and no EOA is present.

The routine uses a nonzero index byte to locate the configuration of the EOA within the entry. The routine compares the configuration with the first data in the buffer, and if they are not the same, finds no EOA present.

If the EOA indication is present, the Incoming/Outgoing Message Delimiter routine increments the count of reserve characters in the LCBISZE field of the LCB by the length of the EOA, logically removing the EOA from the buffer. After removing the EOA, the routine turns off the 'tete-a-tete' flag in the LCB (LCBPESP bit in LCBSTAT2) and exits to the Transparent CCW Building routine (IEDQGT) if the line is a BSC line in transparent mode. If the line is not BSC in transparent mode, the routine exits to the Buffer Association routine (IEDQGD). For a destination on a BSC line, no EOA is present, but there is a STX character that is left in the buffer.

External Routines:

- IEDQAL - Address finder routine - to get the scan pointer address.

- IEDQAX - Buffer Step routine - to scan for a specified character.

- IGG019RB or IGG019RO - TCAM Dispatcher - the DSPPOSTR entry point, to tpost empty units to the Buffer Return QCB.

- IEDQTNT - Termname Table code - to obtain the address of the Terminal Table entry for the destination.

Tables/Work Areas: SCB, Termname Table, Terminal Table, AVT, buffer currently being processed, QCB, process entry work area, LCB, DCB, DEB, UCB, SCT.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

MESSAGE HANDLING - FUNCTIONAL ROUTINES

Date and Time Provision Routine (Chart AC)

Module Name: IEDQAC

Entry Point: IEDQAC01 - called through the User Interface routine when the DATETIME macro is issued in an MH to insert the date and/or time information into a message header.

Functions: This routine inserts the current date and/or time of day into the message header at the current location of the scan pointer.

The DATETIME macro expansion places the address of the parameter list built for the DATETIME macro at assembly time in register 1 and passes to the Date and Time Provision routine through the User Interface routine. The parameter list format is as follows:

Offset	0	+1	+2
	Index to IEDQAC	Parameter List Length	Count of Bytes to be Inserted

If only date information is requested, the Date and Time Provision routine obtains the current date in packed decimal format from the CVT field CVTDATE. If only the time or both the time and the date are requested, the necessary information is obtained in packed decimal format via the TIME system macro.

The Date and Time Provision routine unpacks and zones the date into the format BYY.DDD, where B is a blank, YY is the last two digits of the year, and DDD is the day of the year. The routine unpacks and zones the time into the format BHH.MM.SS, where B is a blank, HH is the hour, MM is the minute, and SS is the second.

The Date and Time Provision routine places the formatted information in the message buffer (time follows date when both are specified), updates the scan pointer to refer to this last character of the new data, and places a normal return code of X'00' in register 15. This routine then branches to the Return Interface routine.

External Routines:

- IEDQAL - Address Finder routine - to find the address of the scan pointer from the offset.

- OS Time routine (SVC 11) - to get the current time and date information.

- IEDQAX - Buffer Step routine - get the next insert address.

Tables/Work Areas: CVT, AVT, buffer currently being processed.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

Output Sequence Number Provision Routine (Chart AD)

Module Name: IEDQAD

Entry Point: IEDQAD01 - called through the User Interface routine.

Functions: This module inserts the output sequence number in a buffer of a message.

The routine gets the output sequence number from the SCB (SCBOSEQ) and converts it into EBCDIC, suppressing leading zeros. The routine then figures the length of the number, adds one for a leading blank, and links to the Insert Data routine (IEDQAF) to shift left data in the buffer the required number of bytes. If return from IEDQAF indicates insufficient reserve characters, a X'04' is set in register 15 and return is made to the caller via the Return Interface routine (IEDQLM).

If expansion was successful, the Output Sequence Number Provision routine links again to IEDQAF to insert the output sequence number, including a leading blank, into the buffer. A X'00' return code is set in register 15 and return is made to the calling routine via IEDQLM.

The format of the macro-generated parameter list supplied as input to User Interface is as follows:

Offset 0	+1
Index to IEDQAD	Index to IEDQAF

External Routine: IEDQUI - User Interface rcutine - to activate the Insert Data routine (IEDQAF) to expand the buffer.

Tables/Work Area; AVT, buffer, SCB.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

Locate Option Field Address Routine (Chart AE)

Module Name: IEDQAE

Entry Point: IEDQAE - called by the User Interface routine when the LOCOPT macro is issued in an MH to return the address of an option field.

Functions: This routine calculates the address of an option field from its index.

The LOCOPT macro expansion places the address of the parameter list built for the LOCOPT macro at assembly time in register 1 and passes control to the Locate Option Field Address routine through the User Interface routine. The parameter list format is as follows:

Offset 0	+1	+2	+3
Index to IEDQAE	Parameter List Length X '04'	Option Field Offset	Return Register 15 Offset

The Locate Option Field Address routine first obtains the key (ordinal index) of the currently contacted terminal from the LCBTTCIN field of the LCB. This key is passed to the Termname Table code (IEDQTNT), which returns the address of the Terminal Table entry for that key. If, however, the key is zero, the source terminal cannot be found and the Locate Option Field Address routine takes the error exit.

The Locate Option Field Address routine next examines the "option fields used" flag in the terminal entry status byte (TRMSTATE). If this bit is not on, the routine takes the error exit. In the error exit, the routine stores a return code of X'FF' in the proper word in the register save area and sets register 15 equal to X'04'. If register 15 itself is specified as the return register, it is set to a return code of X'00'.

If the "option fields used" flag is on, the Locate Option Field Address routine compares the "number of option entries" field (TRMOPNO) in the terminal entry with the option field offset in the third byte of the input parameter list. If the option field offset is high, the error exit is taken.

Next the Locate Option Field Address rcutine gets the cffset byte for the option field beinq souqht by indexinq TRMOPT by the option field offset in the input parameter list. If the offset byte in the terminal entry is equal to X'FF', the opticn field is not defined for this entry, and the routine takes the error exit.

The routine computes the address of the option field being sought by addinq the address of the Opticn Table (AVTOPTPT), the offset to the set of cption fields for this entry (TRMOPTBL), and the offset byte to the individual cption fields (third byte of the input parameter list).

If the return reqister specified is reqister 15, the address is placed in reqister 15. Otherwise, the address is stored in the proper word in the reqister save area, and reqister 15 is set to a return code of X'00'.

The Locate Option Field Address routine returns to the calling routine via the Return Interface routine (IEDQLM).

External Routine: IEDQTNT - Termname Tatle code - to obtain the Terminal Tatle address for the specified entry.

Tables/Work Areas: AVT, LCB, Termname Tatle, Terminal Table, Option Table.

Attributes: Reentrant, serially reusatle, refreshable, enabled, resident, proflem proqram mode.

Messaqe Limit Rcutine (Chart AG)

Module Name: IEDÇAG

Entry Point: IEDQAG01 - activated by the MSGLIMIT macro expansion to limit the numter of messaqes sent cr received in a transmission sequence.

Functions: This routine limits the numter cf messaqes to cr from a terminal durinq a sinqle transmissicn sequence.

On entry, reqister 1 contains the address of the input parameter list. The format of this list is as follcws:

Offset	0	+1	+2	+3
	Index to IEDQAE	Parameter List Length	Status Offset	Register 15 Offset
+4	Reserved	Limit		

The Message Limit routine first saves the parameter list address in the AVT and then gets the address of the buffer being processed from the AVT (AVTADBUF). The routine then examines the size field in the buffer prefix (PRFSIZE) and, if it is zero, indicating a zero-length buffer, returns immediately with a return code of X'00' in register 15.

If the size of the buffer is not zero, the routine gets the address of the LCB from the buffer prefix (PRFLCB) and examines the dial bit (LCBDIAL in LCBSTAT1) in the LCB. If this bit is on, indicating a dial line, the routine places a return code of X'00' in register 15 and returns. When the dial bit is off, the routine examines the Receive Scheduler priority field (LCBRSPRI) and, if it is X'20', examines the 'send' bit (LCBSENDN in LCBSTAT1). If the 'send' bit is on, the routine places a X'00' return code in register 15 and returns.

If the 'send' bit is off or if the priority field is not X'20', the Message Limit routine gets the requested message limit from the parameter list and saves it in the AVT. Next, the routine gets the SCB address from the LCB (LCBSCBA), and increments the message count field in the SCB (SCBSNDCT) by one. The routine then compares the new count with the requested count and, if the new count is lower, places a X'00' return code in register 15 and returns. If the requested message count has been reached, the routine resets the message count field to zero and turns on the 'message limit' bit (SCBMLMTN in SCBSCFM). The routine rechecks the 'send' bit and, if it is on, places a return code of X'00' in register 15 before returning.

If the 'send' bit is off, the Message Limit routine turns off the 'message limit' bit. The the routine gets from the LCB the relative line number (LCBUCBX) and multiplies it by four to convert that number to an offset. The routine gets from the LCB the address of the DCB (LCBDCBPT) and gets from the DCB the address of the invitation list for the line (DCBINVLI + the offset). The routine gets the width of one entry from the invitation list, the address of the current invitation list from the LCB (LCBINVPT), increments this address by the width, and places the result back in the invitation list address field in the LCB. The routine then places a X'00' code in register 15 and returns.

External Routines: None.

Tables/Work Areas: AVT, buffer prefix, LCB, SCB, DCB.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

Input Sequence Number Insertion Routine (Chart AH)

Module Name: IEDQAH

Entry Point: IEDQAH01 - activated through the User Interface routine by the SEQUENCE macro expansion on the input side of MH.

Functions: This module verifies and updates an input sequence number specified by the user in the current buffer of a message.

The Input Sequence Number Insertion routine examines the 'terminal currently connected' field in the ICB (ICBTTCIN). If the field contains zeros, the origin is unknown; therefore, this routine places a return code of X'0C' in register 15 and returns to the caller through the Return Interface routine (IEDQLM). If the field does not contain zeros, the routine passes ICBTTCIN to the Termname Table code (IEDQTNT), which returns the address of the Terminal Table entry for the origin of the message.

The routine passes the EBCDIC characters making up the user-supplied input sequence number in the AVT work area (AVTDOUBL). The routine converts this number to a binary number and compares the result to the anticipated input sequence number located in the Terminal Table entry (TRMINSEQ). If the new number is higher, the routine sets the 'sequence number high' error flag in the SCB and places a return code of X'08' in register 15. If the new number is lower, the routine sets the 'sequence number low' error flag in the SCB, and places a return code of X'04' in register 15. In both cases, the sequence number in the Terminal Table entry remains unchanged and the routine returns to the caller through the Return Interface routine (IEDQLM).

If the new number is equal to the number in the Terminal Table entry, the routine sets the 'sequence-in' flag in the SCB status field (SCBSEQIN in SCBSTATE). If the current buffer is a header buffer, the routine puts the number in the prefix input sequence number field (PRFISEQ) and then incrementes the number by one. If the result is over 9999 (the maximum permitted sequence number), it is reset to one. The routine stores the updated number back in the Terminal Table entry, sets a return code of X'00' in register 15, and returns to the caller through the Return Interface routine (IEDQLM).

The format of the input parameter list for this module is as follows:

Offset	0	+1
	Index to IEDQAH	Parameter List Length

External Routine: IEDQTNT - Termname Table code - to convert a destination offset to a terminal entry address.

Tables/Work areas: AVT, buffer, ICB, SCB, Termname Table, Terminal Table.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

Skip Forward and Scan Routine (Chart AI)

Module Name: IEDQAI

Entry Point: IEDQAI01 - called through the User Interface routine by the macro expansion of the SETSCAN macro or by a functional MH routine to skip the scan pointer forward a fixed number of bytes or to scan for and return the next field in the message header.

Functions: This module moves the scan pointer forward in the message header a specified number of bytes, or finds and returns to the caller the next field beyond the scan pointer.

If the scan pointer (PRFSCAN) is beyond the end of the buffer, that is, PRFSCAN is greater than PRFSIZE, the Skip Forward and Scan routine places a -X'04' return code in register 15 and returns to the calling routine.

The scan pointer offset from the prefix is passed to the Address Finder routine (IEDQAL) to get the scan pointer address and to initialize the current-unit and end-of-unit registers. If the buffer is a TSO buffer, indicated by the PRFTSBUF bit in the PRFSTAT1 field, the Skip Forward and Scan routine passes an offset of zero instead of the scan pointer and the configuration of a blank character in a register. If the user specifies a configuration, the routine sets the register from the parameter list. If no configuration is specified, the routine places an EBCDIC blank (X'40') character in the register.

- Entry for skip forward

If the length passed in the parameter list is zero, the Skip Forward and Scan routine places the address of the byte pointed to by the scan pointer in register 15 and returns to the caller. If a blank character is not defined for the skip operation, the routine adds the skip length directly to the scan pointer. If the resulting offset is beyond the end of the buffer, the routine does not change the scan pointer, places a X'04' return code in register 15, and returns control to the calling routine. If the new offset is not beyond the end of the buffer, the routine places the new offset in the PRFSCAN field, sets a X'00' return code in register 15, and returns to the caller.

If a blank character is defined, the routine loops to the Buffer Step routine (IEDQAX), which returns the address of each subsequent byte. When the byte is not a blank, the skip length is decremented by one. When the skip length is equal to zero, the routine updates the scan pointer offset in the prefix puts a return code of zero in register 15, and exits to the Return Interface routine.

If return from the Buffer Step routine indicates that the end of the buffer has been passed, the routine puts a return code of X'04' in register 15 and exits to the Return Interface routine.

- Entry for fixed scan

The caller may define the next field as the next "n" data bytes. This is the fixed scan function. For this function, the routine loops to the Buffer Step routine, which returns the address of each subsequent byte. When the byte does not contain a blank, or if blanks are not defined for the function, it is inserted into the AVT work area (AVTDOUBL) and a counter of data bytes found is incremented by one. When the field length requested is satisfied, the routine takes the normal scan end exit.

If return from the Buffer Step routine indicates that the end of the buffer has been passed, the routine moves the portion of the field that has been found from the AVT work area to the SCB save area (SCBMBSSA) and takes the multiple-buffer-header exit.

- Entry for variable scan

The caller may define the next field as the next contiguous string of data bytes that is delimited by a blank. This is the variable scan function. For this function, the routine loops to the Buffer Step routine until a non-blank character is returned. The Skip Forward and Scan routine stores this character in the AVT work area. The routine loops again to the Buffer Step routine, adding data bytes to the AVT work area until a blank delimiter is found or until the eight-byte AVT work area is filled. At this point, the routine takes the normal scan end exit.

If return from the Buffer Step routine indicates that the end of the buffer has been passed, the routine moves the portion of the field that has been found from the AVT work area to the SCB save area (SCBMBSSA) and takes the multiple-buffer-header exit.

- Normal scan end exit

At the SCANNED entry point, the Skip Forward and Scan routine stores the field length in the AVT parameter area (AVTPARM). If a compare operation is requested, the routine gets from the parameter list the address of the compare string and compares that address with the string found. If the strings are not equal, the routine places a X'00' return code in register 15 and returns to the caller. If they are equal, or if no compare operation is requested, the routine determines whether entry is from a SETSCAN macro expansion. If so, and if the string offset is to be returned in a register, the routine places the offset in register 15 and returns control to the calling routine. If the offset is not to be returned, the routine makes the offset the new scan pointer, places a X'C0' return code in register 15, and exits to the caller.

When entry is directly from a SFTSCAN macro expansion, the Skip Forward and Scan routine determines whether the address is to be returned in a register. If the address is not to be returned, the routine makes the offset the new scan pointer, places a X'00' return code in register 15, and branches back to the caller. If the address is to be returned, the routine branches to the Address Finder routine (IEDOAL), which returns the address of the last byte of the string found. If the address is to be returned in register 15, the routine places it there and returns to the caller. If the address is to be in another register, the routine saves it in the register save area at the proper offset for the requested register, places a return code of X'00' in register 15, and exits to the calling routine.

- Multiple-buffer-header exit

For a TSO buffer, the Skip Forward and Scan routine puts the length of the field found in the AVT parameter area and performs the processing described for the normal scan end exit.

Otherwise, if return is requested in a register, the routine determines whether entry is from the Multiple Routing subtask. If entry is not from that subtask, the routine determines whether any bytes of the field being sought were found in this buffer. If no bytes have been found, the routine determines whether entry is to search for a conditional character string. If not, the routine places a negative return code in register 15 and returns to the caller. If entry is to search for a conditional character string, the routine saves the number of bytes found and the register for the calling routine in the SCB. The routine sets the scan pointer to point beyond the end of the buffer, saves the parameter list address in the SCB multiple-buffer-entry field (SCBMBHEN), puts a negative return code in register 15, and returns control to the caller.

If return is not requested in a register, or if entry is from the Multiple Routing subtask, the Skip Forward and Scan routine determines whether the buffer is the last buffer of a message, indicated when the PRFNLSTN bit is off in the PRFSTAT1 field. If the buffer is not the last buffer of a message the routine saves the bytes found in the buffer, the count of bytes found, and the calling routine registers in the SCB. The routine then sets the scan pointer to the point beyond the end of the buffer, saves the parameter list address in the SCBMBHEN field, places a negative return code in register 15, and returns control to the calling routine.

- Multiple-buffer-header entry

The Skip Forward and Scan routine may be entered directly from code in the STARTMH macro expansion to complete a scan function interrupted by a multiple-buffer-header situation. This routine may also be entered from the Multiple Routing subtask to find the remainder of a subsequent destination that was incomplete in a previous buffer. The Skip Forward and Scan routine detects this type of entry when the low order bit of the SCB multiple-buffer-header-entry field (SCBMBHEN) is set.

If entry is not from the Multiple Routing subtask, the routine clears the SCBMBHEN field to zeros, gets from the SCB the contents of the registers of the calling routine at the time of the interruption, and moves the register contents to the calling routine save area. The routine then calculates the return address from the parameter list address and puts it in the calling routine save area.

The routine examines the SCB save area to determine whether the first character has been found. If it has not, the routine resumes the appropriate scan function as if initially entered. If the first character has been found, the routine moves the portion of the field that has been found back from the SCB save area to the AVT work area. The routine calculates the number of bytes found. If a fixed scan function is being completed, the routine resumes the function as if initially entered. If a variable scan function is being completed, the routine resumes the function at an entry past the point where the first character is being found.

If the X'02' bit is on in the low-order byte of the SCBMBHEN field, entry is from the Multiple Routing subtask. The Skip Forward and Scan routine turns off both the X'01' and X'02" flags in this byte and gets the number of bytes already found from the SCBDESTL field. If this number is zero, the routine resumes the appropriate scan function as though it were initially entered. For a positive number, the routine moves the bytes found from the LCB (LCBCPA) to the AVT work area (AVTDOUBL) and resets the low-order byte of the AVTDOUBL field to an EBCDIC blank X'40'. The routine then resumes the fixed or variable scan function as described previously.

The address of the parameter list built by the macro expansion of the SETSCAN macro or a higher-level MH routine is placed in register 1 and passed to the Skip Forward and Scan routine by the User Interface routine. The parameter list formats are as follows:

Scan Function Parameter List (Index flag X'01' is OFF

Offset 0	+1	+2	+3
Index to IEDQAI	Parameter List Length	Register 15 Offset	Scan Length
Blank Character (optional)	Address of the Character String (optional)		

Index flag X'02': ON - BLANK = YES
OFF - BLANK = NO

Skip Forward Function Parameter List (Index flag X'01' is ON)

Offset 0	+1	+2
Index to IEDQAI	Parameter List Length	Skip Count

Upon completicn of its functions, the Skip Forward and Scan routine issues a return code in register 15. The return code value is X'00' for successful completion, a negative value if a multiple-buffer header is detected and a scan function fails to complete, and X'04' if the skip function fails to complete.

Other areas affected by the completicn of this routine are outlined as follows:

1. AVT work area (AVTDOUBL) - on successful completion cf a scan function, the next field in the buffer.

2. AVT parameter area (AVTPARM) - on successful completion of a scan function, the second byte contains the length of the field returned.

3. Buffer prefix scan pointer (PRFSCAN) - on successful completion of a scan function, and if requested by the caller, the offset in the buffer to the end of the field being returned. On successful completion of the skip forward function, the offset of the scan pointer mcved forward the specified length. On a multiple-buffer-header exit, the offset to a point one byte beyond the end of data in the buffer.

4. SCB save area (SCBMBSSA) - on a multiple-buffer-header exit for a scan function, the bytes of the field requested found from this buffer, padded with blanks (if necessary to fill SCBMBSSA) to the right.

5. SCB multiple-buffer-header entry (SCBMBHEN) - on a multiple-buffer-header exit, the address of the parameter list.

6. SCB register save area (SCBRGSAV) - on a multiple-buffer-header exit, the user registers.

External Routines:

- IEDOAL - Address Finder routine - to return the address of the scan pointer.

- IEDQAX - Buffer Step routine - to return the address of subsequent bytes.

Tables/Work Areas: AVT, SCB, buffer.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mcde.

Skip to Character Set Routine (Chart AJ)

Module Name: IEDQAJ

Entry Point: IEDQAJ - called through the User Interface routine when the FORWARD or SETSCAN macro expansion or a higher-level MH routine needs to advance the scan pointer to the last byte of a specified character string.

Functions: This routine advances the scan pointer to the end of a specified character string in the message header.

The User Interface routine passes to the Skip to Character Set routine a parameter list built by the caller. The format of this parameter list is as follows:

Offset 0	+1	+2	+3
Index to IEDQAJ	Parameter List Length	Length	Register Offset
Blank Character	Address of Character String		

If the scan pointer (PRFSCAN) is beyond the end of the buffer, that is, if the PRFSCAN field is greater than the PRFSIZE field, the function cannot be performed in this buffer. If a return register is not specified, the Skip to Character Set routine places a -X'04' return code in register 15 and returns control to the caller. If a return register is specified, the routine puts a X'04' return code in register 15 if it is not the return register, and exits to the calling routine. If register 15 is specified as the return register, the routine places a X'00' return code in register 15 and returns to the caller.

When the scan printer is not beyond the end of the buffer, the Skip to Character Set routine passes the scan pointer offset in the buffer prefix to the Address Finder routine (IEDQAL) to get the scan pointer address and to initialize the current-unit and end-of-unit registers. The routine gets the configuration of a blank from the parameter list, the address of the Buffer Step (IEDQAX) routine from the AVT, and the address of the character string being sought from the parameter list.

If the Skip to Character Set routine is entered directly from the STARTMH macro expansion to complete a skip that was interrupted by a multiple-buffer-header situation, the routine moves the data that was found from the previous buffer from the SCB save area to the AVT work area (AVTDOUBL). The routine then clears the multiple-buffer-header-entry field in the SCB (SCBMBHEN) to zeros and moves the user register from the SCB (SCBRGSAV) to the user save area. The routine calculates from the address of the parameter list a return address to the message handler and puts that address in the register save area. The number of bytes found is calculated and processing continues as described below.

If the Skip tc Character Set routine is entered normally or after the multiple-tuffer-header situaticn just described has been handled, the routine locps to the Buffer Step routine (IEQDAX), which returns the address cf each subsequent byte. When a non-blank character is found or if nc blank is defined, it is inserted in the AVT work area (AVTDOUBL) and a counter cf data bytes fcund is incremented by one. The routine compares the character with the first character of the character string to be skipped. If they are equal, the routine compares the counter with the length of the character string to be skipped. If the count is the same, the character string has been located.

If the count is not the same, the routine loops to the Buffer Step routine to get the next data byte, which is inserted in the AVT work area. Comparison is made again, as just described.

If the characters found in the buffer do not match the character string to be skipped, the routine successively shifts left the contents of the AVT work area, dropping one byte at a time, and compares them (to a successively diminishing length) to the character string. This procedure continues until either an equal compare is found or the characters are exhausted. After each unequal compare the routine resumes looping to the Buffer Step routine to get the next data byte.

If return from the Buffer Step routine indicates that the end of the buffer has been passed, the Skip to Character Set routine tests to see if a parameter return register was specified. If sc, the routine puts a return code of X'04' in register 15, unless register 15 is itself the parameter return register specified. In this case, the routine sets register 15 to zero, and exits to the Return Interface routine.

If a parameter return register was nct specified, the routine moves the data found in the buffer from the AVT work area to the SCB save area (SCBMBSSA) and pads with blanks to the right, if necessary. The routine saves the count of bytes found in the SCB SCBDESTL field. The routine saves the parameter list address in the SCB multiple-buffer-header entry field (SCBMBHEN). The routine moves the user registers, saved in the AVT, to the SCB register save area (SCBRGSAV). The routine then updates the prefix scan pcinter to pcint beyond the end of the buffer, sets a negative return code in register 15, and exits to the Return Interface routine (IEDQIM).

If the character string is fcund, the Skip to Character Set routine determines whether a return register is specified. If not, the routine updates the scan pointer to pcint to the last byte of the string, sets a X'00' return code in register 15, and returns control to the caller. If a return register is specified and entry is not from a SETSCAN macrc expansion, the routine puts in register 15 the offset to the last byte of the string found and returns to the calling routine. If a return register is specified and entry is from a SETSCAN macro expansion, the routine saves the address of the last byte of the string found in the appropriate word of the register save area, places a X'00' return code in register 15, and returns to the caller. If register 15 is the return register, the routine places the address of the last byte cf the string fcund in register 15 and branches back tc the caller.

External Poutines:

- IEDQAL - Address Finder routine - to return the address of the scan pointer.
- IEDOAX - Euffer Step routine - to return the address of subsequent bytes in the tuffer.

Tables/Work Areas: Buffer currently being processed, AVT, SCB.

Attributes: Serially reusable, refreshatle, enabled, resident, problem program mcde.

Line Control Insertion Routine (Chart AK)

Module Name: IEDQAK

Entry Point: IEDOAK01 - activated by the User Interface routine (IEDOUI) to insert line control characters in an outgoing message.

Functions: This module checks line control characters and inserts them into a messace that is ready to be sent.

When the Line Control Insertion routine is activated, it first tests the "message form request" bit in the SCB, and if it is off, the routine branches immediately to the Incoming/Outgoing Message Delimiter routine (IEDQA4). The Line Control Insertion routine also branches directly to the Incoming/Outgoing Message Delimiter routine when the input buffer has a length of zero.

When the Line Control Insertion routine maintains control, it determines whether to place an STX character in the buffer. An STX line control character is required if the buffer is a header or a recalled buffer, and if the destination terminal is a binary synchronous device, an IBM 2260 Remote, or an IBM 2760 in tete-a-tete mode.

Before inserting the STX character, this routine calculates the initial offsets for any subsequent line control characters. The routine uses an internal subroutine, LCOFFSET for a non-recalled header buffer or LCOFFRCL for a recalled buffer, to perform this calculation.

The location at which the Line Control Insertion routine places the STX character depends on the type of buffer. In a non-recalled header buffer, the routine inserts the STX character as the first data byte and sets the data offset to point to the first data byte. In a recalled buffer, the routine inserts the STX character immediately after the last EOB in the buffer. (The offset to the last EOB is in the SCB field SCBEOB.) If the routine is also placing ITB characters in the buffer, the routine decrements the offset to the last EOB by the number cf ITB characters inserted in the buffer before the EOB. The routine calculates the number cf ITB characters and adjusts the data offset accordingly.

If an STX line control character is nct required, the Line Control Insertion routine determines whether any intermediate line control characters (ECBs or ITBs) are needed.

The routine gives control to a Line Control Offset subroutine (LCOFFSET for non-recalled buffers, or LCOFFRCL for recalled buffers) in order to get the initial offsets for inserting the characters. Upon return, the Line Control Insertion routine gives control to the Line Control Selection subroutine (LCSELECT) to get the address of the next insert position in the buffer and to set the offset in the SCT to the address of the next line control character to be inserted. The routine compares the total data size (PRFSIZE) with the next insert offset to determine if the first line contrcl character will fit in the current buffer. If the character does not fit, the routine branches to the entry point LAST to complete final processing. If the character fits, control returns to the main routine at the entry point MAINLOOP with the data offset equal to the insert offset.

The main loop of this routine first sets the condition code to 2 and passes control to the Insert subroutine at the GETSCTAD entry point. This subroutine inserts the first line control character in the buffer. Upon return, the routine calculates the residual count in the buffer (the number of bytes from the inserted character to the end of data) and compares the result with the interval between ITBs or, if no ITBs are being inserted, with the interval between EOBs or ETBs. If the next line control character (pointed tc in the SCT) will not fit in the buffer, or if no ITBs, EOBs, or ETBs are to be inserted in the buffer, this routine branches to the Insert Data routine (IEDQAF) to shift any logically empty bytes to the end of the buffer. Upon return, final processing (at LAST) is performed.

If the next line control character does fit in the buffer, the routine gives control to the Insert Data routine to shift the data left to the next insert point. On return, the Line Contrcl Insertion routine gives control to the Line Control Selection subroutine to select the next line control character to be inserted. The subroutine returns to the main routine at the entry pcint MAINLOOP, where the line control character insertion process is performed again.

For final processing, the routine checks the buffer to determine whether it is the last buffer of the message. If the buffer is not the last buffer of the message, the routine determines whether any character insertion has been performed. When an insertion has been made, the routine gives control to the Insert subroutine at the entry point FINALSIZ, passing a condition code set to 8. The subroutine sets the final data size and returns. At this point the Line Control

Insertion routine determines whether EOBs or ETBs were inserted. If EOBs or ETBs have been inserted, the routine sets the EOB/ETB initial address for the next buffer in the SCB. If ITBs have been inserted, the routine sets the ITB initial offset for the next buffer in the SCB. The routine then passes control to the Incoming/Outgoing Message Delimiter routine (IEDQA4).

If the buffer is the last buffer of the message, the routine clears the ECB/EOT interval and offset fields in the SCB to zero, and indicates that an end-of-transmission (EOT) line control character must now be inserted in the buffer. If no characters have been inserted, the routine sets the data offset to equal the total data size. The routine passes the offset of the last data byte to the Address Finder routine to get the address of that byte. This address is the insert address. If the destination is BSC or if the destination is Start/Stop with no EOBs being inserted, the routine passes the EOT character address in the SCT with a condition code of 2 to the Insert subroutine (GETSCTAD). This subroutine inserts the EOT and then passes control to IEDQA4. If the destination is Start/Stop and EOBs are being inserted, an EOB character followed by an EOT character must be inserted in the buffer. The routine gives control to the Insert subroutine at entry point GETSCTAD twice with a condition code of zero; once to get the EOB character address and once to get the EOT character address from the SCT. On return from the second link to the Insert subroutine, the routine builds a single character string in the Address Vector Table work area (AVTDOUBL). This character string consists of the EOB character followed by the EOT character. The routine then passes the length and address of this string to the Insert subroutine at the entry point LINKAOBT with a 2 condition code. The subroutine inserts the data in the buffer and gives control to IEDQA4.

There are four internal subroutines:

1. Line Control Offset subroutine for non-recalled buffers LCOFFSET.

2. Line Control Offset subroutine for recalled buffers - LCOFFRCL.

3. Line Control Selection subroutine - LCSELECT.

4. Insert subroutine - GETSCTAD, LINKAOBT, and FINALSIZ.

Either of the two line control offset subroutines first calculates the offset to the first byte of data to be sent. In LCOFFSET, this is the first data byte in the buffer; in LCOFFRCL, this is the first data byte after the last EOB. The subroutine then increments the initial EOB/ETB address found in the SCB by the result. If ITBs are to be inserted into the buffer, the subroutine increments the initial ITB address in the SCB by the same result before returning.

The Line Control Selection subroutine first determines whether ITBs are to be inserted. If ITBs are to be inserted, the subroutine compares the ITB offset in the SCB with the EOB/ETB offset in the SCB. If the ITB offset is lower, both offsets are returned to the calling routine after the subroutine increments the ITB offset in the SCB by the interval between ITBs. If ITBs are not being inserted or if the ETB offset is equal to the ITB offset, the subroutine increments the EOB/ETB offset in the SCB by the interval between EOBs or ETBs and increments the ITB offset, if present, in the SCB by the interval between ITBs. After this, the subroutine returns the EOB/ETB offset and SCT offset to the calling routine.

The Insert subroutine consists of three segments, GETSCTAD, LINKAOBT, and FINALSIZ. The functions of each of the segments follow.

1. GETSCTAD - This segment uses an SCT offset passed by the caller to determine the address of a line control character in the SCT. If the calling routine passes in register 14 a 0 condition code, this segment returns to the calling routine; otherwise, this segment passes control to the next segment - LINKAOBT.

2. LINKAOBT - This segment builds parameters for and links to the Unit Request Interface routine (IEDQAO) to insert the specified line control character in the buffer. If the calling routine passes in register 14 a non-positive condition code, this segment passes control to the next segment - FINALSIZ.
3. FINALSIZ - This segment decrements the data address offset by the insert character offset (length of the logically empty area) and sets the result as the final data size in the buffer. If the condition code in register 14 is equal to zero, this segment returns to the calling routine; otherwise, this segment gives control to the Incoming/Outgoing Message Delimiter routine (IEDQA4).

When the Line Control Insertion routine gains control, register 1 contains the address of a macro-generated parameter list with the following format.

Offset	0	+1	+2	+3
	Index to IEDQAK	Parameter List Length	Index to IEDQAF	Index to IEDQAO

External Routines:

- IEDQUI - User Interface routine - to activate the following modules:

- IEDQAF - Insert Data routine - to insert data in the buffer.

- IEDQAO - Unit Request Interface routine - to insert line control characters in the buffer.

- IEDQTNT - Termname Table code - to get a terminal entry address.

- IEDQAL - Address Finder routine - to get the address of a data byte.

Tables/Work Areas: AVT, DCB, LCB, SCB, SCT, buffer currently being processed.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

Address Finder Routine (Chart AL)

Module Name: IEDQAL

Entry Point: ADDRCCMP - called from an MH routine to find the address of an item in a buffer.

Functions: This routine returns the address of an item in a buffer when the offset of the item from the start of the first unit is passed to it.

The Address Finder routine gets the address of the AVT from register 9 and the address of the first unit of the current buffer from register 6.

The routine then compares the offset in register 5 with the key length specified in the AVT (AVTKEYLE). If the offset is less than or equal to the key length, the item is in the current unit. The Address Finder routine adds the offset to the address of the current unit plus the RCB length (12 bytes - AVTUMALN) to get the address of the item, which is then returned in register 5. The address of the unit in which the item is found is returned in register 2.

If the offset is greater than the key length, the routine gets the address of the next unit from the RCB of the current unit (PRFTIC). The routine then decrements the offset by the key length and again compares the offset with the key length in the AVT to determine whether the item is in the (new) current unit. If not, the routine gets the address of the next unit and again decrements and compares the offset. This process continues until the buffer unit that contains the item is found.

After locating the address of the item, the Address Finder routine examines register 1. If it does not contain zeros, the routine returns to the caller. If it does contain zeros, end-of-unit updating is being requested. This causes the Address Finder routine to place the address of the unit in which the item is found in register 4, to set register 11 to point to the first byte beyond the end of the current unit, to load register 1 with the parameter list address for the calling routine (from AVTPARM), and to return to that routine.

External Routines: None.

Tables/Work Areas: AVT, buffer currently being processed.

Attributes: Serially reusable, reentrant, refreshable, enabled, resident, problem program mode.

Origin Routine (Chart AM)

Module Name: IEDÇAM

Entry Point: IEDQAM01 - called by the ORIGIN macro expansion to verify or initialize the origin of a message.

Functions: This rcutine verifies the origin of a message when the origin is specified, or initializes the crigin when it is not specified.

The Origin routine, upon getting control from the ORIGIN macro expansion tests the return code from the previously executed Binary Search routine (IEDQA1). If the return code is zero, the name of the buffer was not found in the Termname Table; therefore, the Origin routine sets the 'invalid origin' bit in the SCB (SCBSTAT1), places a return code of X'04' in register 15, and returns.

If the return code from the Binary Search routine is nct zero, the value in register 1 is the key (ordinal index) to the Termname Table entry for the name found. In this case, the Crigin routine compares the key with the key in the buffer prefix source field (PRFSRCE). If the keys are equal, the key in the buffer prefix is correct, and the Origin routine puts a X'00' return ccde in register 15 before returning. If the keys are not equal, the Crigin rcutine checks the buffer prefix scurce field for zeros. If the field does not contain zeros, the routine considers the buffer prefix key to be the wrong key and passes that key to the Termname Table code (IEDQTNT), which returns the address of the Terminal Table entry. The Origin routine examines the terminal entry to determine whether it is a line entry. If the entry is not for a line, the routine sets the 'invalid origin' bit in SCBSTAT1, places a X'04' return ccde in register 15, and returns to MH.

If the Terminal Table entry is a line entry or if the buffer prefix source field is zero to indicate that the field may be initialized, the Origin routine makes a final check. The routine passes the key for the name found in the buffer to the Termname Table code, which returns the address of the Terminal Table entry. The routine gets the address of the QCB from the Terminal Table entry and compares the address of the DCB to which the QCB pcints (QCBDCBAD) to the address of the DCB tc which the LCB points (LCBDCBPT). If the two addresses are the same, the routine saves the buffer prefix scurce key in the PRFSRCE field and in the terminal-currently-ccnnected field (LCBTTCIN) in the LCB. The routine then places a X'00' return code in register 15 and exits to MH. If the addresses are not the same, the routine considers the name found in the buffer to be in error, sets the 'invalid crigin' bit in the SCBSTAT1 field, places a X'04' return code in register 15, and exits to MH.

External Routine: IEDÇTNT -Termname Table code - to locate the address of the Terminal Table entry.

Tables/Work Areas: Buffer prefix, AVT, ICB, QCB, Termname Table, Terminal Table.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

Multiple Insert/Remove Routine (Chart AN)

Module Name: IEDQAN

Entry Point: IEDQAN01 - activated by the User Interface routine (IEDQUI) to insert, delete, and replace data at locations specified by character strings in the buffer.

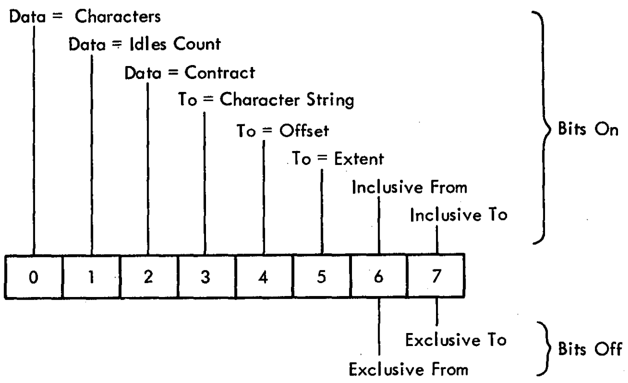
Functions: This module translates and tests all data in a buffer and, thereby, inserts, deletes, or replaces data in specific positions in the buffer.

The Multiple Insert/Remove routine first tests the PRFSIZE field in the buffer for zeros. If the field is zeros, indicating that the buffer has a length of zero, the routine returns immediately to the calling routine. If the field is not zeros, the routine initializes the internal parameter lists for the IEDQAI and IEDQAJ scan routines according to the parameters passed.

The Multiple Insert/Remove routine builds a translation table from the initial letter of each character string in the input parameter list and, thereafter, performs a translate and test operation on all data in the buffer. Register 1 contains the address of the input parameter list. The format of the list is as follows.

Offset	0	+1	+2	+3	
0	Index to IEDQAN	Parameter List Length	Index to IEDQAF	Index to IEDQAO	
+4	Index to IEDQAJ	Blank Character	Number of Entries	Reserved	
+8	Reserved	Address of Characters Table			
+12	Key	Status	Data Description		First Subparameter List
+16	'FROM' Delimiter Description		'TO' Delimiter Description		

The format of the subparameter status byte is as follows.



The Multiple Insert/Remove routine places the offset from the beginning of the input parameter list to the subparameter list at a position in the translate table indicated by the initial letter of a character string. The function of the subparameter list is determined by the character string, the first letter of which was placed in the translate table. The routine saves the offset of the byte of data (data offset) and the number of the logically empty bytes available for insertion (insert offset) in the prefix of the buffer. The initial insert offset is zero.

If, while building the translate table, the routine detects a subparameter list that specifies a delimiter character as the insert function indicator (bits 0 and 1 are both on), the routine checks to verify whether the function is valid. To be valid, the macro must be on the send side of the message handler and the destination must be an application program. If either of these conditions is not present, the routine bypasses the delimiter insert operation. When both conditions are present, the routine branches and links to the Termname Table code (IEDQTNT) to get the address of the Terminal Table entry for the destination. From the entry the routine gets the delimiter character and puts it in the subparameter list to simulate a reserve character with a count of one.

The Multiple Insert/Remove routine then begins to execute its main processing loop (TESTEOS). The routine compares the data offset with the total size of data in the buffer (PRFSIZE). If the data offset is higher, the routine first adjusts the PRFSIZE field to decrement any logically empty bytes remaining at the end of the buffer and then returns to the message handler via the Return Interface routine (IEDQLM).

If the data offset is lower than the total size of data in the buffer, the routine passes the offset to the Address Finder routine (IEDQAL), which returns the data address and the address of the end of the buffer unit in which the data address is located. The Multiple Insert/Remove routine then performs the translate-and-test operation beginning at the data address. The routine performs a translate-and-test for the length of the unit, or if the unit is the last unit in the buffer, for the length of the data in the unit.

If the unit translates to all zeros, the routine gives control to the Test and Shift subroutine (TESTSHIF). This subroutine determines whether there are any logically empty bytes preceding the data just translated. If there are no empty bytes, the subroutine returns immediately. Otherwise, the subroutine gives control to the Insert Data routine (IEDQAF) to shift the data just translated to the left in the buffer, overlaying the logically empty bytes and, thereby, moving the logically empty area to the right end of the buffer unit. Upon return from the Insert Data routine, the Test and Shift subroutine returns control to the main routine. At this point, the Multiple Insert/Remove routine increments the data offset by the length of data just translated and branches back to the main processing loop in order to translate and test the next buffer unit.

If a translate-and-test operation on a unit results in a hit; that is, a byte of data translates to nonzero, the routine analyzes the hit. If the hit did not occur on the first byte translated, the routine gives control to the Test and Shift subroutine. This subroutine shifts the bytes just translated to the left in the buffer, thus moving the logically empty area to the right, until it reaches the byte that is the hit. When the subroutine returns, the main routine increments the data offset by the length of data just translated.

The Multiple Insert/Remove routine finds the subparameter list indicated by the offset to which the hit byte translated and uses the information in the subparameter list to build an input parameter list for the Skip Forward and Scan routine (IEDQAI). The Multiple Insert/Remove routine temporarily sets the prefix scan pointer to the position just precedinc the hit byte and then gives control to the Skip Forward and Scan routine. The Skip Forward and Scan routine determines whether the hit byte is the first byte of the character string that equals the string that governs the function specified in the subparameter list. If the character strings are not equal, the Multiple Insert/Remove routine increments the data offset past the hit byte and branches back to the main processing loop. If there are insufficient characters remaining in the buffer to determine whether the strings are equal, the character string being sought is not in the buffer. Therefore, the routine clears to zero the byte associated with this string in the translate table. The routine then increments the data offset and branches back to the main processing loop.

If the character strings are equal, the Multiple Insert/Remove routine determines whether an insert or a remove function is requested. If the function is an insert, the routine gives control to the Test and Shift subroutine, which shifts the character string to the left in the buffer. The routine then uses the subparameter list to build an input parameter list for the Unit Request Interface routine (IEDQAC) and branches to that routine through the User Interface routine (IEDQUI). The Unit Request Interface routine gets another buffer unit, if one is needed, for data insertion and links to the Insert Data routine (IEDQAF). The Insert Data routine inserts the specified data, adjusts the data offset and insert offset, and returns to the main processing loop of the Multiple Insert/Remove routine. If the Unit Request Interface routine finds nc empty units available, it returns immediately to the Multiple Insert/Remove routine with a 4 condition ccde in reqister 15. This routine discontinues the translate-and-test operation and returns to the Return Interface routine (IEDQIM) with the 4 return code in reqister 15.

If the character strinq specifies a remove function, the Multiple Insert/Remcve rcutine determines the 'TO' delimiter for the remove function: the 'AT' delimiter is the character string already found. If the 'TO' delimiter is a character string, the routine uses information from the subparameter list tc build an input parameter list for the Skip to Character Set routine (IEDQAJ) and temporarily sets the scan pointer to the point just past the hit byte. The routine then gives contrcl to the Skip to Character Set routine, which scans for the delimiting character string and returns. If the 'TO' delimiter is an extent, rather than a character string, the routine gets the extent from the subparameter list and adds it to the data offset to get a new data offset. After the 'TC' delimiter is found, the routine determines whether the 'FROM' delimiter string itself is to be removed. If this string is nct to be removed, the rcutine gives control to the Test and Shift subroutine to shift the 'FROM' delimiter string to the left of any logically empty bytes in the buffer. If the delimiter is to be removed, the routine increments the number of logically empty bytes (the insert cffset) by the length of the 'FROM' delimiter string. If the remove function specified is a contract function, the routine brances to the main processing loop. If, however, the data to be replaced is a delimiter character, the routine must find the terminal entry for the destination and extract the configuration of the delimiter from the table. At this point, the routine places the delimiter in the subparameter list and then performs the insert function. If the remcve function is a replace function, the routine performs the insert function.

External Routines:

- IEDQUI - User Interface routine - tc activate the fcllowing modules:

  IEDQAF - Insert Data routine - tc insert data at a specific lccation or tc shift data in the buffer.

  IEDQAI - Skip Forward and Scan routine - to scan for a specific character string in the buffer.

  IEDQAJ - Skip to Character Set rcutine - to scan for a 'TO' delimiter character string.

  IEDQAO - Unit Request Interface routine - to get an additional buffer for the insert functicn.

- IEDQAL - Address Finder rcutine - to find the data address and the address of the end of the buffer unit in which the data address is located.

- IEDQTNT- Termname Table code - to get a terminal entry address.

Tables/Work Areas: Translation Table for the translate-and-test operation, AVT, LCB, SCB, buffer currently being processed.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

Unit Request Interface Routine (Chart AO)

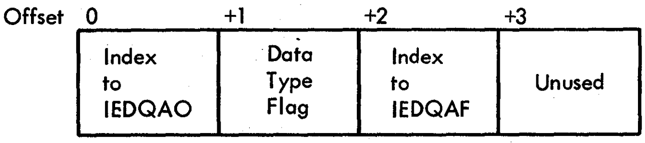
Module Name: IEDQAO

Entry Point: IEDQAO01 - activated by the User Interface routine (IEDQUI) to get a buffer unit requested by one of the insert routines and to add that unit to the buffer currently being processed.

Functions: This module provides the interface to the Unit Request routine in order to get an additional buffer unit and to attach it to the buffer that requires the extra space.

The Unit Request Interface routine first determines whether the data to be inserted fits in the buffer currently being processed. If the data fits, the routine builds a parameter list for the Insert Data routine (IEDQAF) and branches to that routine to insert the requested data. If the data does not fit, the routine links to the Unit Request routine (IEDQBW) to get an empty buffer unit. If an empty buffer is not available, the Unit Request Interface routine returns control to the Return Interface routine (IEDQLM) with a return code of X'04' in register 15 and all zeros in register 8. If an empty buffer unit is available, the Unit Request routine returns control to the Unit Request Interface routine with the address of the empty buffer. The Unit Request Interface routine then links to the Address Finder routine, passing the data offset. The Address Finder routine returns the address of the unit into which the data insertion is to be made. The Unit Request Interface routine then links the new unit into the buffer between the unit pointed to by the scan pointer and the following unit. The routine then moves the data from the address to the end of the unit to the corresponding location in the new unit and increments the data offset, the insert offset, and the prefix size field (PRFSIZE) by the unit size. At this point, the Unit Request Interface routine builds a parameter list for the Insert Data routine and exits to that routine to insert the requested data.

The internal input parameter list is in the AVTPARM field of the AVT. This list is not macro generated. The format of this list is as follows.



Offset	0	+1	+2	+3
	Index to IEDQAO	Data Type Flag	Index to IEDQAF	Unused

External Routines:

- IEDQBW - Unit Request Routine - to get the buffer unit needed to insert data.

- IEDQAL - Address Finder routine - to get the data offset address, the unit address, and the end-of-unit address.

Tables/Work Areas: AVT, LCB, SCB, buffer currently being processed.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

Remove at Offset Routine (Chart AP)

Module Name: IEDQAP

Entry Point: IEDQAP01 - activated through the User Interface routine (IEDQUI) to remove and optionally replace data in a buffer.

Functions: This module removes data from a single specified location in a buffer and optionally replaces that data with new data.

The Remove at offset routine first examines the PRFSIZE field in the buffer prefix. If this field contains zeros, the buffer has a length of zero, and this routine exits to the Return Interface routine (IEDQLM).

When the buffer size is not zero, the Remove at Offset routine calculates the size of the buffer prefix, including any reserve characters that may be present. Then the routine determines whether the TO delimiter (specified in the TO operand of the MSGEDIT macro instruction) is a character string. If it is a character string, the routine sets up the AT delimiter (specified in the AT operand of the MSGEDIT macro instruction) either from the scan pointer or from the input parameter list plus the prefix size, if the user specifies an offset. If the AT delimiter is the scan pointer and the scan pointer is beyond the end of the buffer, the function cannot be performed. In this case the routine exits with a X'04' return code in register 15. Otherwise, the routine temporarily sets the scan pointer to the AT offset, builds a parameter list for the Skip to Character Set routine (IEDQAJ) and branches to that routine through the User Interface routine to scan for the TO character string. When control returns, if the string is not found in the buffer currently being processed, the routine exits to the Return Interface routine (IEDQLM) with a X'04' return code in register 15. On the other hand, if the string is found, the routine determines whether the TO string itself is to be removed. If the string is not to be removed, the routine decrements the offset returned from the Skip to Character Set routine by the length of the string. If the string itself is to be removed, the routine does not change the offset, but gives control to its testing function loop (TESTFUNC) to determine which function is being performed.

If the TO delimiter is an extent, the Remove at Offset routine gets the extent from the input parameter list, adds the AT delimiter to the extent to make the TO delimiter an offset, and branches to the testing function loop.

If the TO delimiter is an offset, the routine gets the delimiter from the input parameter list and adds the prefix size. At this point, the routine determines whether the AT delimiter precedes the TO delimiter, and if it does not, exits to the Return Interface routine. If the AT delimiter precedes the TO delimiter, the routine branches to the testing function loop.

If the AT delimiter is the scan pointer and the TO delimiter is zero, the single byte at the scan pointer is to be removed. The Remove at Offset routine sets the TO delimiter equal to the scan pointer plus one and branches to the testing function loop.

After the AT and the TO delimiters have been determined, if the remove request is to the left of the scan pointer in the buffer, this routine adjusts the scan pointer to the left accordingly.

The testing function loop (TESTFUNC) of the Remove at Offset routine first sets the data offset equal to the TO offset, sets the insert offset equal to the length between the TO and AT offsets, and determines which function is being performed.

If the TO delimiter is beyond the end of the buffer, the routine adjusts the scan pointer to point to the end of the buffer so that all data from the AT delimiter to the end of the buffer is removed or replaced.

If the function is a contract operation, the routine bypasses the next insert operation, builds a parameter list for the Insert Data routine (IEDQAF), and links to that routine through the User Interface routine to shift the logically empty area to the end of the buffer. On return, the Remove at Offset routine decrements the data offset by the insert offset, sets the result as the new data size (PRFSIZE), and returns to the Return Interface routine.

If the function is a replace operation, the routine builds a parameter list for the Unit Request Interface routine (IEDQAO) and branches to that routine through the User Interface routine to insert the replacement data in the buffer. On return, the routine determines whether the replacement data was exactly as long as the data removed. If the replacement data was the exact length, the routine exits to the Return Interface routine; otherwise, the routine must close the buffer. To close the buffer, the routine builds a parameter list for the Insert Data routine and branches to that routine through the User Interface routine to shift the logically empty area to the end of the buffer. When control returns, the Remove at Offset routine decrements the data offset by the insert offset, sets the result as the new data size (PRFSIZE), and returns control to the Return Interface routine.

The address of the input parameter list for this routine is in register 1. If the TO delimiter is a character string, the format of the parameter list is as follows:

<table>
<tr><th>Offset</th><th>0</th><th>+1</th><th>+2</th><th>+3</th></tr>
<tr><td></td><td>Index to IEDQAP</td><td>Status Byte</td><td>Index to IEDQAF</td><td>Index to IEDQAO</td></tr>
<tr><td>+4</td><td colspan="4">Insert Data (optional)</td></tr>
<tr><td>+8</td><td>Index to IEDQAJ</td><td>Parameter List Length</td><td>X '00'</td><td>Register 15 Offset</td></tr>
<tr><td>+12</td><td>Blank Character (optional)</td><td colspan="3">Address of Character String</td></tr>
<tr><td>+16</td><td colspan="2">AT Delimiter (optional)</td><td></td><td></td></tr>
</table>

If the TO delimiter is an extent or an offset, the format of the parameter list is as follows:

<table>
<tr><th>Offset</th><th>0</th><th>+1</th><th>+2</th><th>+3</th></tr>
<tr><td></td><td>Index to IEDQAP</td><td>Status Byte</td><td>Index to IEDQAF</td><td>Index to IEDQAO</td></tr>
<tr><td>+4</td><td colspan="4">Insert Data (optional)</td></tr>
<tr><td>+8</td><td colspan="2">To Delimiter</td><td colspan="2">At Delimiter (optional)</td></tr>
</table>

Bit 7 of the IEDQAP index byte indicates the following:

OFF - remove at the specified offset
ON - remove at the scan pointer

External Routine: IEDQUI - User Interface routine - to activate the following modules:

IEDQAF - Insert Data routine - to shift the logically empty area to the end of the buffer.

IEDQAJ - Skip to Character Set routine - to scan for the TO delimiter character string.

IEDQAO - Unit Request Interface routine - to insert replacement data in the buffer.

Tables/Work Areas: AVT, buffer currently being processed, LCB, SCB.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

Operator Control Interface Routine (Chart AQ)

Module Name: IEDQAO

Entry Point: IEDQAO01 - called by the CODE macro expansion in an INHDR subgroup to test for operator control characters and either return to the caller or tpost the buffer.

Functions: This module tests the prefix status byte (PRFSTAT1) for a 'not-last-buffer' flag or a 'TSO buffer' flag and, if either of these flags is on, branches back to the calling routine. The Operator Control Interface routine next compares the character string in the AVT work area (AVTDOUBL) with the operator control characters (AVTCTLCH). If these character strings do not match, the routine returns control to the caller. Otherwise, the routine examines the SCB status byte (SCBSTATE) to determine whether the 'lock' bit (SCBWCKIN) is on. If this bit is on, the routine immediately branches back to the caller.

The Operator Control Interface routine gets the destination key for the origin from the buffer prefix (PRFSRCE) and, if it is equal to zero, returns control to the calling routine. If the key is not equal to zero, the routine passes this key to the Termname Table code (IEDQTNT), which returns the address of the Terminal Table entry. The Operator Control Interface routine examines the status byte in the entry (TRMSTATE) to determine the status of the 'operator control' flag. If this flag is off, the routine returns to the calling routine.

Otherwise, the routine gets from the AVT the offset to the last byte of the character string in the buffer and places the offset in the scan pointer field (PRFSCAN). Next, the routine gets from the AVT the address of the Operator Control QCB and places that address in the RCB of the buffer. The routine then sets the operator control priority and tposts the buffer to Operator Control by branching to the DSPPOST label in the TCAM Dispatcher.

External Routine: IEDQTNT - Termname Table code - to get the Terminal Table entry address.

Tables/Work Areas: AVT, buffer, SCB, LCB, Terminal Table entry.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

Cutoff Message Transmission Routine (Chart AU)

Module Name: IEDQAU

Entry Points:

- IEDQAU - the Cutoff routine entry point - activated by the User Interface routine (IEDQUI) from the CUTOFF macro expansion to test the cutoff count and initiate the cutoff function, if needed.

- CUTFFQCB+12 - the Cutoff subtask entry point - activated by the TCAM Dispatcher when the Line End Appendage tposts the LCB to the Cutoff QCB in order to execute the cutoff channel program and to tpost the final buffer to the message handler after the channel program has terminated.

Functions: This module cuts off the transmission of a message being received after the receipt of a user-specified number of bytes or on detection of identical characters in the buffer. The Cutoff Message Transmission routine gains control to process each buffer of a message as a result of the CUTOFF macro in a message handler. This routine detects whether the cutoff function is needed and if it is, performs the necessary functions to activate the Cutoff Message subtask. The subtask actually stops the transmission of the message.

The specific functions of the routine and of the subtask, respectively are described in the following paragraphs.

- Cutoff Message Transmission Routine

The Cutoff Message Transmission routine first tests the SCB "cutoff" flag (SCBCUTFN) to determine whether a cutoff function is already in progress.

When the "cutoff" flag is not set, the Cutoff Message Transmission routine continues processing to determine whether the cutoff function is needed. If the buffer has a length of zero, the routine exits immediately to the calling routine.

The format of the CUTOFF macro-generated parameter list that is used by the User Interface routine to activate the Cutoff Message Transmission routine is as follows:

Offset	0	+1	+2
	Index to IEDQAU	Parameter List Length	Requested Cutoff Length

The Cutoff routine next determines if all the characters in the buffer are identical, indicating a line error. If so, the routine initiates the cutoff function, described below.

If no identical characters are detected, the routine increments the cutoff count field (SCBBKFCT) in the SCB by the data size of the current buffer (PRFSIZE). If the result is not greater than the user-specified maximum, the Cutoff Message Transmission routine returns control to the calling routine. If the maximum has been exceeded, the routine initiates the cutoff function.

To initiate the cutoff function, the Cutoff Message Transmission routine sets the 'cutoff' flag in the SCB, sets the LCB 'error' flag (LCBPRCPG), and puts the address of the Cutoff QCB into the LCBQCBA field of the LCB. These flag settings indicate to the Line End Appendage that the Cutoff Message Transmission subtask is to be activated. After initiating the cutoff function, the Cutcff routine returns control to the calling routine, the Return Interface routine (IEDOLM).

When the 'cutoff' flag is set, a cutoff is in progress for the current message. In this case, the routine examines the LCB QCB field (LCBQCBA). If the QCB field is nct zerc and the current buffer is the last buffer of a message, the routine turns off the LCB 'error' flag. If the QCB is zerc, the cutoff functicn is ccmplete. The routine then exits to the calling routine.

- Cutoff Message Transmission Subtask

The Cutoff Message Transmissicn subtask stops the transmission of a message by activating the appropriate channel program.

A channel program check condition indicates initial entry to the subtask. In this case, the subtask examines the UCB to find which channel program is to be set up. For teletype terminals, the channel program is a Write-Break; for IBM 2260 terminals, a Write-Break and Read-Skip; and fcr all other IBM terminals, a Read-Skip. The subtask then issues the SVC 0 for the apprcpriate channel program and exits to the TCAM Dispatcher at entry point DSPDISP.

When there is no channel prcgram check conditicn, the Cutoff Message Transmission subtask gets frcm the LCB the address of the first buffer received after initiation of the cutoff. The subtask flags this buffer as the last buffer of the message and sets its data size to one. The subtask then exits to the TCAM Dispatcher with an indication to tpost this buffer tc the STARTMH QCB.

External Routine: OS EXCP routine (SVC 0) - to start channel activity.

Tables/Work Areas: AVT, DCB, DEB, UCB, SCB, LCB, QCB, buffer.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

Lookup Terminal Entry Routine (Chart AV)

Module Name: IEDQAV

Entry Point: IEDQAV01 - called frcm the FORWARD macro expansicn cr by ancther MH routine to assign a buffer to its destination.

Functions: The Lookup Terminal Entry routine first gets the destination key from either register 1 cr the buffer prefix. If the key is not available, the routine exits with a X'04' return code in register 15. If the destination key is present in register 1, the routine places the key in the PRFDEST buffer prefix field fcr a header buffer or the AVTPARM3 field in the AVT fcr a non-header buffer.

After a destinaticn key has been found, this routine passes that key to the Termname Table code (IEDQTNT), which returns the address of the Terminal Tatle entry. The Lcokup Terminal Entry routine returns the Terminal Tatle entry address to the calling routine in the AVTPARM field of the AVT. The rcutine then qets the address of the QCB from the Terminal Tatle entry and determines whether there is queuing for this terminal. If there is no queuing or if the QCB is a PUT application proqram QCB, the rcutine places a X'04' return code in reqister 15 and exits to the calling routine. If queuing is specified, the rcutine saves the QCB address in the SCBDESTQ field of the SCB, places a X'00' return ccde in reqister 15, and returns control to the caller.

External Routine: IEDQTNT - Termname Table code - to obtain the address of the Terminal Table entry for the destination.

Tables/Work Areas: Termname Table, Terminal Table, AVT, SCB, buffer currently beinq processed, LCB.

Attributes: Serially reusable, refreshable, enabled, resident, problem proqram mcde.

Translate Buffer Rcutine (Chart AW)

Module Name: IEDQAW

Entry Point: IEDQAW01 - called when the CODE macro is issued in an MH or from the STARTMH subtask tc translate a tuffer.

Functions: This routine initializes the Translation Table address and translates the data in a buffer.

When the Translate Buffer routine is activated, reqister 1 points to a input parameter list. The fcrmat of this list is as follows:

Offset	0	+1	+2	+3
	Index to IEDQAW	Parameter List Length	X '00'	Status
+4	Address of Translation Table (optional)			

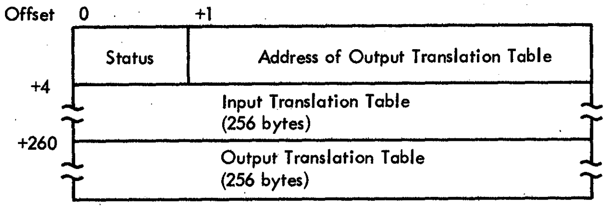
The settinqs cf the status bits are as follcws:

Bit 0 ON Translation Table address is in the DCB
CFF Translation Table address is in the input parameter list

Bit 1 ON Nonstandard Translation Table
OFF Standard Translation Table

Bit 2 ON The CODE macro is in the INBUF or OUTBUF delimiter group
OFF The CODE macro is in the INHDR or OUTMSG delimiter group

The format of a Translation Table is as follows:



The Translate Buffer routine may be entered from a CODE macro expansion or from the STARTMH subtask. If entry is from a CODE macro expansion, the routine determines whether the CODE macro is in an INHDR or OUTHDR subgroup. If the macro is in either of these, the routine determines whether the next buffer is to be processed by the CODE macro expansion. If the next buffer is not to be processed by the macro, the routine turns on the SCBCODE bit (in SCBSTATE) in the SCB and saves the address of the CODE parameter list in the SCBTRANS field to ensure that the next and subsequent buffers can be translated via a link from the STARTMH subtask.

The Translate Buffer routine gets the address of the translation table from either the DCB or the parameter list. If the translation table is a dynamic translation parameter list, the routine passes control to the Dynamic Translation routine (IEDQA3), which indicates via a branch table the return location for processing.

On return, if the table is a standard translation table, the routine examines the first word of the table and, if it is zero, to indicate that the table is an EBCDIC translation table, returns to the caller via the Return Interface routine (IEDQIM). If the first word is not zero, the routine points to either the input translation table for receiving or the output translation table for sending.

Next, the Translate Buffer routine finds the offset to the first data byte of the buffer or, if the buffer is a canceled buffer, to the offset to the first byte following the last EOB character. At this point the translation operations begin. The routine calculates the total number of bytes to be translated and, if the buffer is of zero length, returns to the calling routine. For nonzero length buffers, the routine passes the offset of the first byte to be translated to the Address Finder routine (IEDQAL). The address returned by the

Address Finder routine is the starting address for the first translation. The Translate Buffer routine decrements the end-of-unit address returned by the starting address to obtain the length of data in the first unit. If the total length of data is equal to or smaller than the length of data in the first unit, the data ends in the first unit, and the routine issues a branch to the label TRANS2. At TRANS2, final translation of the total length of data is performed, and the Translate Buffer routine returns to the calling routine.

If the data in the buffer does not end in the first unit, the Translate Buffer routine decrements the total data length by the length of data in the unit, and then translates the entire unit. The routine gets the address of the next unit from the TIC field of the unit just translated. The starting address is set at the start of data (past the RCB) in the unit. The Translate Buffer routine compares the remaining total length of data to be translated with the key length (AVTKEYLE). If the total length is equal or smaller, the data ends in this unit, so the final translation is made and this routine exits to its calling routine.

If the data does not end in the second unit, the total length is decrmented by the key length, and the entire unit is translated. The Translate Buffer routine gets the address of the next unit and continues as just described until all the data in the buffer is translated.

External Routines:

- IEDQAI - Address Finder routine - to initialize the starting address for translation.
- IEDQA3 - Dynamic Translation routine - to perform dynamic translation of the buffer.

Tables/Work Areas: AVT, LCB, DCB, SCB, current buffer, Translation Table.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

Screen Routine (Chart AY)

Module Name: IEDQAY

Entry Point: IEDQAY01 - activated by the User Interface routine (IEDQUI) to initialize for a screen command modification operation on the buffer destination.

Functions: This module checks the Unit Control Block (UCB) for the device being used and initializes for a screen command modification operation on the destination.

Register 1 contains the address of the input parameter list. The format of the parameter list is as follows.

Offset	0	+1
	Index to IEDQAY	Request code

Bit 7 of the index byte indicates the following:
    ON - command change requested
    OFF - return the current function setting only

The request code can be one of the following:
    X'00' - to indicate a write at display cursor operation
    X'01' - to indicate a write at line address operation
    X'02' - to indicate a write erase operation

The Screen routine first checks the UCB to determine whether the buffer destination is an IBM 2260 Local or an IBM 2260 Remote.

If the destination is a 2260 Local, the routine uses an index byte from the input parameter list to locate the requested function byte in an internally defined table. The routine places the function byte in the key field of the buffer and sets the 'screen request' bit in the LCBSCRNN field of the LCB. At this point the routine has completed processing and returns to the Return Interface routine (IEDQIM) with the new function byte in register 15.

If the destination is a 2260 Remote, the Screen routine verifies that the destination is a screen device. If it is not a screen device, the routine places zeros in register 15 and branches to the Return Interface routine. If the destination is a screen device, the routine gives control to the Termname Table code (IEDQTNT) to get the address of the terminal entry for the destination. Upon return, the Screen routine finds the device-dependent area of the entry and places the current setting of the function byte in register 15. If a change of function is requested, the routine selects a new function byte, places it in the current function byte in the terminal entry, and then branches to the Return Interface routine.

External Routine: IEDQTNT - Termname Table code - to get the address of the terminal entry for the buffer destination.

Tables/Work Areas: AVT, buffer currently being processed, LCB, SCB, DEB, UCB, Terminal Table.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

Skip Backward Routine (Chart A0)

Module Name: IEDQA0

Entry Point: IEDQA001 - called through the User Interface routine when SETSCAN is issued in an MH to move the scan pointer backward in the header of a message a specified number of bytes.

Functions: This routine moves the scan pointer backward a specified number of bytes in the header of a message.

When the Skip Backward routine is activated, register 1 points to an input parameter list. The format of this list is as follows:

Offset 0	+1	+2
Index to IEDQA0	Parameter List Length	Skip Count

If the scan pointer is beyond the end of the buffer, the Skip Backward routine returns immediately with a X'04' in register 15.

When the skip count in the input parameter list contains zeros, this routine links to the Address Finder routine (IEDQAL) to get the scan pointer address. On return, the Skip Backward routine returns to the calling routine with the scan pointer address in register 1 and a X'00' return code in register 15.

When the skip count is not equal to zero, the Skip Backward routine places the prefix scan pointer offset from the buffer prefix (PRFSCAN) in a scan pointer offset register and then links to the Address Finder routine (IEDQAL) to get the address of the scan pointer and of the unit in which it is located. The Skip Backward routine puts the address of the last byte of the RCB of this unit in a "start of unit" register and puts the specified skip count in a count register.

The Skip Backward routine next enters a main processing loop that first decrements the scan pointer address and offset registers by one. If the offset is reduced to zero, the skip is into the prefix and the routine puts a return code of X'04' in register 15 before returning to the calling routine via Return Interface (IEDQLM). If the offset is not reduced to zero, the routine compares the scan pointer address to the "start of unit" register. If the scan pointer address is high, regular processing continues; otherwise, the Address Finder routine is used to get a new scan pointer address and a new "start of unit" address for the preceding unit.

If the byte at the current position of the scan pointer is a blank, the Skip Backward routine branches to reenter the main processing loop. If the byte is not a blank, the routine decrements the count register by one. If this does not reduce the count to zero, the routine branches to reenter the loop.

When the count is reduced to zero, the skip is complete. At this point, the Skip Backward routine updates the prefix scan pointer (PRFSCAN) from the scan pointer offset register, sets a return code of X'00' in register 15, and returns to the calling routine via Return Interface.

External Routine: IEDQAL - Address Finder routine - to return the address of the scan pointer and of the unit in which it is located.

Tables/Work Areas: AVT, buffer being processed.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

Insert at Offset Routine (Chart A2)

Module Name: IEDQA2

Entry Point: IEDQA201 - activated by the User Interface routine (IEDQUI) to insert data in a message at a specific location.

Functions: This module inserts data into a message buffer at a specific location.

If the buffer has a length of zero, this module puts a return code of X'04' in register 15 and exits to the Return Interface routine (IEDQLM). Otherwise, the Insert at Offset routine determines whether the insert operation is the current location of the scan pointer. If the insert is at the scan pointer, the routine gets the insert offset from the prefix field (PRFSCAN); otherwise, the routine gets the offset from the input parameter list, the address of which is in register 1. The format of the input parameter list is as follows:

Offset		+1	+2	+3
0	Index to IEDQA2	Parameter List Length	Index to IEDQAF	Index to IEDQAO
+4	Insert Data			
+8	Insert Offset (optional)			

Bit 7 of the IEDQA2 index byte indicates the following:

ON - the data is repeated characters
OFF - the data is a character string

If the data is a character string, the format of the insert data word is as follows:

Offset 0	+1
Length of String	Address of Character String

If the data is a string of identical characters, the format of the insert data word is as follows:

Offset 0	+1	+2
Number of Identical Characters	Repeat Characters	Reserved

The Insert at Offset routine next determines whether the insert offset is greater than the buffer size. If the insert offset is greater, the routine does not perform the insert operation, but returns control directly to the Return Interface routine (IEDQLM) with a X'04' return code in register 15.

If the insert offset is not greater than the buffer size, the Insert at Offset routine branches to the Unit Request Interface routine (IEDQAO) via the User Interface routine to insert the data in the buffer. If, on return, the Insert at Offset routine finds that no empty space is available for the insertion, it places a X'04' in register 15 and exits to IEDQLM. If there is logically empty space remaining in the buffer, the Insert at Offset routine links to the Insert Data routine (IEDQAF) via the User Interface routine to shift the empty space to the end of the buffer. Upon return, the routine calculates the final data size, places it in the PRFSIZE field in the buffer prefix, and exits to the Return Interface routine.

External Routine: IEDQUI - User Interface routine - to activate the following modules:

IEDQAF - Insert Data routine - to shift any logically empty span to the end of the buffer.

IEDQAO - Unit Request Interface routine - to insert data in the buffer.

Tables/Work Areas: AVT, buffer currently being processed, LCB, SCB.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

Dynamic Translation Routine (Chart A3)

Module Name: IEDQA3

Entry Point: IEDQA3 - entered from the Translation routine (IEDQAW) via a Branch and Link when the translation table pointer in the DCB points to a TRANLIST macro expansion.

Functions: There are two main functions of the Dynamic Translation routine.

1. To determine, on the first message input from a line using dynamic translation, the correct table from a list of tables. The TRANLIST macro expansion provides the list of tables and one or more control strings.

2. To retrieve from the appropriate option field, the address of a translation table. When the correct table was determined in the first function, its address is stored to be used here.

To determine the correct table, the control string is expected to begin within the first 3 characters of the message. For a correct determination, the user must assure that at least one character that is uniquely translated for each table specified is included in the string. Once a match is found, the Dynamic Translation routine assumes that the table is the only correct one.

The maximum length of a string is eight characters. This routine moves ten characters from the buffer into a work area in the AVT and translates them using the first table in the list. After translation, the Dynamic Translation routine forces each character to upper case. The routine then attempts 3 times to compare a string from the list to the data input. The routine repeats this operation for each string and then uses the next table specified. If the Dynamic Translation routine finds no match before exhausting all tables and all strings, the routine sets an error bit in the SCB and makes an error return. If the routine finds a match, the address of the table is stored in the option field for this terminal. Thereafter, until logoff or hangup in a TSO environment or hangup in a TCAM environment, that table will be used for all input and output translation for that terminal.

The second function of the Dynamic Translation routine merely finds the option field and loads the translation address in the register used by IEDQAW. The routine then returns control to IEDQAW via register 14. All other registers are restored to their value at entry.

5041 Processinq:

Certain dial lines may be designated as terminal type 5041. This indicates that a line will support (including translation) a 1050 or a 2741 terminal. The type is determined at dial up time. Dynamic translation will always be specified for a 5041 line. IEDQA3 will know the type at entry and will only try 1050 translation tables for a 1050 and 2741 tables for a 2741. This is accomplished by the way the TRANLIST macro is used. For a non-5041 line, all tables are included in the LIST= operand and the byte preceding the VCON is X'00'. For 5041 lines, 1050 tables are specified in the L1050= operand and the byte preceding the table VCON is set to X'02', 2741 tables are specified in L2741= and the byte is X'01'.

External Routines:

- IEDOAL - Address Finder routine - to get the address of the data.
- IEDOUI - User Interface routine - to link to the Locate Option Field Address routine (IEDQAE).

Table/Work Areas: AVT,SCB,LCB,DCB

Attributes:- Serially reusable, reentrant, problem program mode.

Forward Routine (Chart A5)

Module Name: IEDQA5

Entry Point: IEDQA501 - activated either by the User Interface routine (IEDOUI) when the FORWARD macro is issued in an MH or from the Multiple Routing subtask.

Functions: This routine determines the destination to which a message is to be sent. If, however, the buffer has a length of zero, the buffer is a TSO buffer, the buffer is a non-recalled text buffer, or the line is in extended lock mode, this routine returns immediately to the calling routine.

When the Forward routine is activated, register 1 points to the first of up to three parameter lists. The first parameter list is for the Forward routine itself. The Forward routine passes the address of the second parameter list to the User Interface routine, thus linking to one of three possible subsidiary routines. These three routines are the Skip Forward and Scan routine (IEDQAI), the Locate Option Field Address routine (IEDQAE), and the Binary Search routine (IEDOA1). If the second parameter list is for either the Skip Forward and Scan routine or the Locate Option Field Address routine, the third paramater list is for the Binary Search routine, which is activated when the routine of the second parameter list passes the address of the third parameter list to the User Interface routine.

The format of the input parameter list for the Forward routine is as follows:

Offset	0	+1	+2	+3	
	Index to IEDQA5	Parameter List Length	Status	Index to IEDQBA	
+4	Length of EOA String	Address of the End-of-Address Character String			Optional
+8	Address of the User Error Recovery Routine				Optional
+12	Variable Data				
+16	Index to IEDQAI	Parameter List Length	Reserved	Length (may be initalized)	
+20	Address of the character string (may be initialized)				

Status byte: This byte has the following definitions.

X'80' - the destination name is defined in the macro
X'40' - the destination name is in an option field
X'20' - the destination name is in the buffer
X'08' - an End-of-Address (EOA) string is specified

Variable data: This field can have one of the two following formats.

+12	Index to IEDQAE	Parameter List Length	Option Field Offset	X '10'

+12	Index to IEDQAI	Parameter List Length	X '10'	Scan Length

When the Forward routine receives control from one of its subsidiary routines, it examines a status byte in its input parameter list to determine which subsidiary routine was executed.

- Return from the Skip Forward and Scan Routine (IEDQAI)

In this case, the Skip Forward and Scan routine has attempted to get the name of the destination for the message from the message buffer. If register 15 contains a negative value, the name is not complete in the current buffer and the Forward routine returns to its calling routine with the negative return code.

If register 15 does not contain a negative value, the Forward routine examines the status byte to determine whether the macro defined an EOA string. If so, this routine compares the EOA string in its parameter list with the string returned by the Skip Forward and Scan routine. If the two character strings do not match and the EOA character string is one byte long, the Forward routine tests each byte of the returned character string for an EOA. If the routine finds an EOA or if the strings match, there are no more destinations. On original entry from the Multiple Routing routine, the Forward routine places zeros in the secondary destination field in the SCB (SCBMRFSD) and exits to the calling routine with an X'C8' return code in register 15. If the EOA is found and entry was not from Multiple Routing, no valid destination has been found; therefore, the Forward routine performs its error recovery procedure. (This procedure is discussed after the Return from the Binary Search routine paragraphs.)

When entry is from the FORWARD macro expansion and either the EOA is not found or if no EOA string is defined, the Forward routine places the prefix scan pointer (PRFSCAN) at the last byte of the string returned, places the last byte in the secondary destination field of the SCB, and overlays the last byte with a unique character. At this point, regardless of which function activates this routine, the Forward routine prepares to link to the routine represented by the third parameter list - the Binary Search routine. The Forward routine places the address and the length of the character string returned by the Skip Forward and Scan routine in the input parameter list for the Binary Search routine. Binary Search is activated when the Forward routine passes the input parameter list to the User Interface routine.

- Return from the Locate Option Field Address Routine (IEDQAE)

In this case, the Locate Option Field Address routine has attempted to locate the name of the destination for the message in an option field. If register 15 contains zero, the Locate Option Field Address routine was not able to find the name; therefore, the Forward routine branches to its error recovery procedure. If register 15 contains a value, it is the address of the option field that contains the destination name. In this case, the Forward routine places the address and length of the character string that is in the option field in the input parameter list for the Binary Search routine. The Forward routine then links to the Binary Search routine by passing its input parameter list to the User Interface routine.

- Return from the Binary Search Routine (IEDQA1)

If the Binary Search routine is the first subsidiary routine linked to from the Forward routine, the address of the destination of the message is defined in the FORWARD macro. Subsequent processing is the same in this case as for the return after searching for a character sting defined in the buffer or in an option field.

If, upon return from the Binary Search routine, the Forward routine finds a zero value in register 15, no matching destination was found. Therefore, Forward branches to its error recovery procedure. If register 15 contains a nonzero value, that value is the offse to the Termname Table entry for the destination. In this case, the Forward routine passes the offset to the Lookup routine (IEDQAV) to get the appropriate Destination QCB address. Upon return, the Forward routine returns to its calling routine with a return code of X'00' in register 15.

- Error Recovery Procedure

The error recovery procedure of the Forward routine first examines the status byte in its input parameter list to determine whether the user has defined a special routine to attempt error recovery. If so, the Forward routine links to the user-specified routine. If the user routine returns with the address of another character string in register 15, the Forward routine links back to the Binary Search routine to try to match this new entry with an entry in the Termname Table. A return register is set, however, to prevent relinkage to the user routine if no match is found.

If no match is found or if no user routine is defined, the Forward routine determines whether a dead-letter queue is defined by examining the AVT field AVTDLOX. If not, the Forward routine places zeros in the secondary destination field of the SCB (Multiple Routing only) and returns to its calling routine with X'04' in register 15. If a dead-letter queue is defined, the Forward routine uses the Lookup routine to get its Destination QCB address and then returns to its calling routine.

External Routines:

- IEDQUI - the User Interface routine to link to the Skip Forward and Scan routine (IEDQAI), the Locate Option Field Address routine (IEDQAE), and the Binary Search routine (IEDQA1).

- IEDQAV - Lookup routine - to find the address of the Destination QCB for a specific terminal entry.

- IEDQAL - Address Finder routine - to get the address of the last byte in a terminal entry.

Tables/Work Areas: AVT, SCB, buffer currently being processed.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

Line Control Initialization Routine (Chart A6)

Module Name: IEDQA6

Entry Point: IEDQA601 - activated by the User Interface routine (IEDQUI) to initialize SCB fields that indicate the intervals between the line control characters to be inserted.

Functions: This module initializes fields in the station control block (SCB) to indicate the intervals between the line control characters to be inserted.

The Line Control Initialization routine initializes the following fields in the SCB.

If EOBs are to be inserted,
1. SCBEOBSZ - initialized to the interval between EOBs.
2. SCBEOBAC - initialized to the interval between EOBs.

If ETBs are to be inserted,
1. SCBEOBSZ - initialized to the interval between ETBs.
2. SCBEOBAC - initialized to the interval between ETBs.

If ITBs are to be inserted,
1. SCBITBSZ - initialized to the interval between ITBs.
2. SCBITBAC - initialized to the interval between ITBs.

If the input buffer to this module is a text buffer or has length of zero, the routine returns immediately to the calling routine through the Return Interface (IEDQLM). Otherwise, the Line Control Initialization routine uses the Termname Table code (IEDQTNT) to get the terminal entry address for the destination.

The Line Control Insertion routine then examines the SCT to determine whether an EOT is defined for the destination terminal. If no EOT is defined, the module returns to IEDQLM. When an EOT is defined, the routine examines the SCT for an EOB entry for this terminal. When an EOB is not defined and the terminal is not in transparent mode or when an EOB is defined and there is no checking for this terminal, at the label SETFIELD the routine sets the 'MSGFORM request' bit in the SCB, puts the subblock and block registers in the SCB, puts a X'00' return code in register 15, and exits to IEDQLM. In all other cases the processing continues to find the block extent.

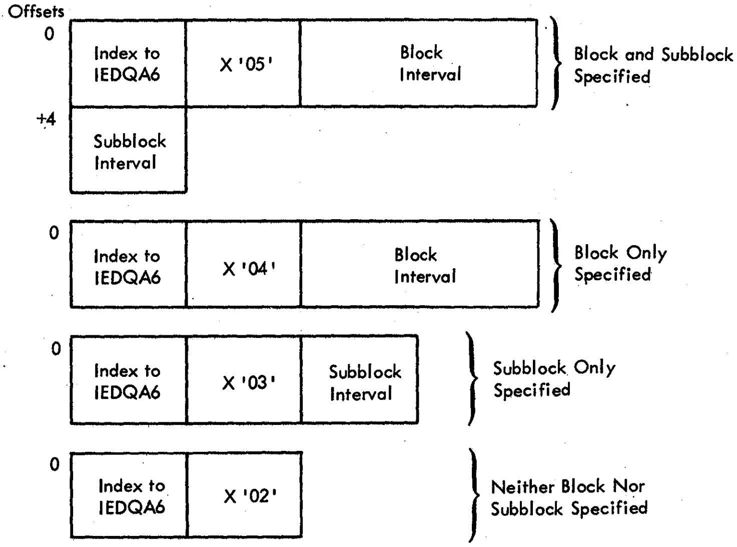
When the block extent is specified on the MSGFORM macro, this routine sets the block register from the parameter list. For a terminal in transparent mode and not on a BSC line, the routine performs the exit functions described in the preceding paragraph. For a terminal in transparent mode on a BSC line, the routine sets the SCB 'transparent' flag, puts the block register in the SCB, puts a X'00' return code in register 15, and exits to IEDQLM.

When the block extent is specified and the terminal is not in transparent mode, this routine branches to the code to find the subblock extent. In this processing, if there is no ITB defined in the SCT or there is no subblock extent on the MSGFORM macro or in the terminal entry, the routine exits as described previously for the label SETFIELD. Otherwise, the routine gets the subblock extent from either the MSGFORM parameter list or the terminal entry. The routine then exits as described for the SETFIELD label.

When the block extent is not specified on the MSGFORM macro, the Line Control Initialization routine must get the block extent from the terminal entry. If the terminal is in transparent mode and on a BSC line, the routine puts the transparent block extent from the terminal entry in the SCB, sets the SCB 'transparent' flag, puts X'00' in register 15, and exits to IEDQLM. If, in this case, the extent is not in the terminal entry, the routine puts a X'04' in register 15 and exits. If the terminal is in transparent mode not on a BSC line, the routine gets the extent, if available, from the terminal entry and exits through the SETFIELD label.

If the terminal is not in transparent mode, the terminal is on a BSC line, and the block extent is not in the terminal entry, this routine puts a X'04' in register 15 and exits to IEDQLM. When the terminal is not in transparent mode and the block extent is in the terminal entry, the routine gets the block extent and then performs the subblock extent search described previously.

The input parameter list, the address of which is contained in register 1, has one of the following formats.



where index to IEDQA6
= X '01' for transparent mode.
= X '00' for text mode.

External Routine: IEDQTNT - Termname Table code - to get the address of the destination terminal entry.

Tables/Work Areas: AVT, SCB, buffer currently being processed, Terminal Table entry for the buffer destination.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

Counter Routine (Chart A7)

Module Name: IEDQA7

Entry Point: IEDQA701 - activated by the COUNTER macro expansion to count either complete messages or message segments.

Functions: This module counts either the ccmplete message or message segments that are being prccessed by the MH subgroup in which the COUNTER macrc appears. If the COUNTER macro appears in an INHDR or OUTHDR subgroup and the buffer is not a header buffer, this routine returns immediately with a X'00' in register 15. Otherwise, the Counter routine determines whether the buffer currently being processed is a zero-length buffer. If so, the routine dces not count the message but returns immediately to the calling routine with a X'FF' return code in register 15. If the buffer is not zerc-length, the routine links to the Locate Option Field Address routine (IEDQAE) through the User Interface routine tc get the address cf the option field. On return, if the option field is not found, the Counter routine does not count the message and exits to the calling routine with a X'FF' return code in register 15.

If the opticn field is found, the Counter routine adds one to the count in the cption field and exits to the calling routine with X'00' in register 15.

External Routine: IEDQUI - User Interface routine - to activate the Locate Option Field Address routine (IEDQAE), which gets the address of the option field.

Tables/Work Areas: AVT, buffer currently being processed, counter option field.

Attributes: Serially reusable, refreshable, enabled, resident, problem prcgram mcde.

Multiple Insert at Offset Routine (Chart A8)

Module Name: IEDÇA8

Entry Point: IEDÇA801 - activated by the User Interface routine (IEDQUI) to insert a character string at specified intervals in the message.

Functions: This module inserts a data string cf up to eight characters at a specified interval (up to 65,535 bytes) in the message.

The address of the input parameter list for the Multiple Insert at Offset routine is in register 1. The fcrmat of the parameter list is as follows.

Offset	0	+1	+2	+3
0	Index to IEDQA8	Parameter List Length	Index to IEDQAF	Index to IEDQAO
+4	Insert Data			
+8	Interval Between Inserts		Index to IEDQAE	Option Field Offset

Bit 7 of the IEDQA8 index byte indicates the following:
    OFF - the data is a character string
    ON - the data is reserve characters

If the data is a character string, the format of the insert data word is as follows:

Offset	0	+1
0	Length of String	Address of Character String

If the data is a string of identical (reserve) characters, the format of the insert data word is as follows:

Offset	0	+1	+2
0	Number of Identical Characters	Repeated Character	Reserved

The Multiple Insert at Offset routine first branches through the User Interface routine to the Locate Option Field Address routine (IEDQAE) to find the address of the option field in which the initial offset for the next buffer is located. On return, if the option field is not found, the Multiple Insert at Offset routine exits to the Return Interface routine (IEDQLM) with a X'C4' return code in register 15. Otherwise, the routine calculates the initial data offset and branches to the main processing loop (MAINLOOP) to insert the data string.

If the data offset is not contained within the buffer that is currently being processed, the routine decrements the offset by the length of data in the buffer, saves the offset in the option field, and exits to the Return Interface routine with a return code of X'00' in register 15. If the data offset falls at the very end of the buffer data and this buffer is not the last buffer of the message, this routine performs the same processing just described so that the insert is done at the beginning of the next buffer.

The main processing loop (MAINLOOP) of the Multiple Insert at Offset routine uses the User Interface routine to link to the Unit Request Interface routine (IEDQAO) in order to insert the data character string. On return, the routine branches, via the User Interface routine, to the Insert Data routine (IEDQAF) to shift the original buffer data to the left up to the next point of data insertion. The routine continues processing in the main loop until the insert point falls beyond the end of the buffer that is currently being processed. When this point is reached, the Multiple Insert at Offset routine branches to the Insert Data routine to shift the logically empty area to the end of the buffer. On return, the Multiple Insert at Offset routine calculates the final data size and places it in the prefix (PRFSIZE). If there is a subsequent buffer to be processed, the routine calculates the initial insert offset for that buffer and places it in the option field. At this point, processing is complete and the routine exits to the Return Interface routine with a X'C0' return code in register 15.

External Routine: IEDQUI - User Interface routine - to activate the following modules:

IEDQAE - Locate Option Field Address routine - to find the address of the option field.

IEDQAF - Insert Data routine - to shift the original data to the left in the buffer.

IEDQAO - Unit Request Interface routine - to insert the data character string.

Tables/Work Areas: AVT, buffer currently being processed, LCB, SCB, user-defined option field.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

Checkpoint Request Routine (Chart BB)

Module Name: IEDQBB

Entry Point: IEDQBB - called through the User Interface routine by the CHECKPT macro expansion or from the Buffer Disposition subtask (IEDQBD).

Functions: This routine sets the "checkpoint request" flag (SCBCKPT) in the SCB if the Checkpoint task is in the system. If the routine was entered from the Buffer Disposition subtask (IEDQBD), the Checkpoint Request routine tposts the ERB to the Buffer Disposition QCB.

If Checkpoint is not in the system and this routine was entered from the User Interface routine, the Checkpoint Request routine exits to Return Interface with a return code of X'04' in register 15. If

Checkpoint is in the system and entry is from User Interface, a successful return code of X'00' is passed to Return Interface.

If the Checkpoint Request routine was entered from Buffer Disposition, exit is to the TCAM Dispatcher to tpost the ERB.

The macro-generated input parameter list for this module has the following format.

Offset 0	+1	+2	+3
Index to IEDQBB	Parameter List Length	X'00'	X'00'

External Routines: None.

Tables/Work Areas: AVT, LCB, SCB.

Attributes: Reentrant, refreshable, enabled, resident, problem program mode.

EOB/ETB Handling Subtask (Chart BT)

Module Name: IEDQBT

Entry Point: IEDQBT - activated by the TCAM Dispatcher when a buffer is tposted to the STARTMH QCB and the STARTMH operands specify that EOB/ETB handling is to be performed.

Functions: This module performs EOB/ETB handling on a buffer. This subtask gains control when a buffer is tposted to STARTMH and the STARTMH operands specify that some form of EOB/ETB handling is to be done.

If the buffer is not marked as the last buffer of a message, the subtask exits to the TCAM Dispatcher at DSPBYPAS to perform a bypass function to the STARTMH routine (IEDQAA). If the buffer is marked as the last buffer of a message, an EOB/ETB appears in the buffer, and this subtask checks for a text error. If an error has occurred, the subtask attempts retry by recalling a previously received/sent buffer. If no error occurred, (for receive operations only) the routine checks for the EOB/ETB options selected. If a user exit was specified, a branch and link passes control to that exit address. This subtask then checks the CONV= operand to determine if the EOB/ETX just received is to be treated as end of message. If not, the subtask tposts the ERB to IEDQKA to continue message reception. For send operations, only the last buffer of a message or one with an EOB/ETB error is marked last. If a permanent error occurs, the STOP/CONT options are checked and the message continues or is aborted based on the options.

External Routine: IEDQUI User Interface routine - to activate the Locate Option Field Address routine (IEDQAE) in order to obtain the address of an option field.

Tables/Work Areas: LCB, SCB, AVT, buffer prefix.

Attributes: Reusable, refreshable, enabled, resident, problem program mode.

Unit Request Routine (Chart BW)

Module Name: IEDQBW

Entry Point: IEDQBW - entered from MH via the Unit Request Interface routine (IEDQAO) to get an extra unit.

Functions: This module supplies extra buffer units to requesting modules. When the Unit Request routine is entered, the calling MH routine is requesting a buffer unit. If a unit is available, the Unit Request routine removes the unit from the buffer unit pool and returns the unit to the calling routine. If no unit is available, the Unit Request routine returns to the calling routine with a return code.

External Routines: None.

Tables/Work Areas: QCB, STCB, SCB, buffer prefix, AVT, LCB, DCB.

Attributes: Reusable, enabled, resident, problem program mode.

Log Segment Routine (Chart BX)

Module Name: IEDQBX

Entry Point: IEDQBX - called through the User Interface routine when the LOG macro is issued in the INHDR, OUTHDR, INBUF, or OUTBUF subgroup of an MH.

Functions: This routine writes, or logs, a message segment onto the logging medium specified by the user in a BSAM DCB.

When the Log Segment routine is activated, register 1 points to a one-word input parameter list. The format of this list is as follows:

Offset 0	+1
Index to IEDQBX	Address of DCB to be used to Log the Segment

The Log Segment routine issues a WRITE and a CHECK macro for each unit of the buffer to be logged on the device specified in the DCB. The unit is written without the twelve-byte control area.

If an I/O error occurs while the buffer is being written, BSAM passes control to the user SYNAD exit in the DCB. (This exit must return to the instruction that follows the CHECK macro.) If the Log Segment routine finds that the specified DCB is not open, it places a return code of X'04' in register 15. The successful write return code is X'00' in register 15.

The Log Segment routine returns to the next in-line instruction in the MH code of the MCP.

External Routines:

- OS BSAM WRITE routine - to write the units of the buffers.
- OS BSAM CHECK routine - to check the write operations.
- OS Getmain routine (SVC 4) - to obtain main storage.

Tables/Work Areas: DCB, AVT.

Attributes: Reusable, refreshable, resident, enabled.

MESSAGE HANDLING - FUNCTIONAL SUBROUTINES

Insert Data Routine (Chart AF)

Module Name: IEDÇAF

Entry Point: IEDÇAF01 - activated through the User Interface routine as a subroutine of a functional TCAM module to insert data in a buffer or to shift data left within a buffer.

Functions: The Insert Data routine performs one of four possible functions.

1. To insert data and return immediately.
2. To insert data, adjust the prefix insert offset by the length of data inserted, and return.
3. To shift data across several units and return.
4. To expand the buffer by shifting data left into the reserve characters area.

Register 1 contains the address of the parameter list generated by a macro expansion or by a calling routine for User Interface to use to activate the Insert Data routine. The requested function is indicated by the flags in the first byte of the list. The format of this area for all except the expand buffer function is as follows:

Offset	0	+1
	Index to IEDQAF	Data Type Flag

Index X '00' - Insert, adjust, and return
X '01' - Insert and return
X '02' - Multiple-unit shift
X '03' - Expand buffer

Data Type Flag X '00' - Data address
X '01' - Reserve characters

For the expand buffer function, the second byte of the list contains the length of the expansion. For all except the expand buffer function, this parameter list is stored at AVTPARM in the AVT.

If the Insert Data routine finds the X'00' or X'01' value in the index field of AVTPARM, it uses a second parameter list at AVTPARM3 for additional input. If the data type is a character string, the format of AVTPARM3 is as follows:

Offset	0	+1
	Length of the Insert Data	Address of the Character String

If the data type is repeated characters, the format of AVTPARM3 is as follows:

Offset	0	+1	+2
	Length of the Insert Data	Reserve Character	Reserved

If the requested function is to insert data and return, the Insert Data routine links to the Insert subroutine, which inserts the specified data, and returns to the calling routine through Return Interface.

If the requested function is to insert data, adjust the prefix insert offset, and return, the Insert Data routine first links to the Insert subroutine where the data insertion is peformed. Upon return from the subroutine, the Insert Data routine gets the prefix insert offset, decrements it by the total length of the data inserted, and stores the new value back in the RCB. The routine then exits to the calling routine through Return Interface.

If the requested function is a multiple-unit shift, the Insert Data routine gets the prefix data offset from the RCB and enters a shift data loop. In this loop, the routine passes the prefix data offset to the Address Finder routine (IEDQAL) with a request for the end-of-unit address and the data address. When the Address Finder routine returns these addresses, the Insert Data routine calculates the count of bytes from the data address to the end of the unit and compares the result to the total length of data to be inserted. If the count of bytes to the end of the unit is greater, the routine places the count in the first byte of AVTPARM3; otherwise, the routine inserts the total length of the data to be inserted in that first byte. The Insert Data routine then places the address of the data in the next three bytes of AVTPARM3. Next, the routine calculates a new data offset that is equal to the current data offset plus the length specified in AVTPARM3. The routine also decrements the total length of data to be inserted by the value in AVTPARM3 - this may reduce the length to zero. The Insert Data routine then links to the Insert subroutine to shift left all the data in the current unit. Upon return from the subroutine, the Insert Data routine tests the remaining length of data for a value of zero. If the value is not equal to zero, the routine must shift the data in the next unit. The routine sets the new data offset, which is now the offset to the start of data in the next unit, as the prefix data offset and then reeneters the shift data loop. When the remaining length is reduced to zero, the Insert Data routine puts X'00' in register 15 and returns to the calling routine through Return Interface.

If the requested function is to expand the buffer, the Insert Data routine tests the scan pointer to determine whether it is beyond the end of the buffer. If it is, the function is discontinued and the routine returns to the caller with a -4 in register 15. Otherwise, the Insert Data routine gets the requested expand length from the input parameter list and places the length in the RCB as the prefix insert offset. The routine gets the number of reserve characters currently in the buffer from LCBISZE in the LCB and compares this value with the length requested. If the expand request is greater than the number of reserve characters, the function cannot be performed - the routine returns to the calling routine through Return Interface with X'04' in register 15. If the expansion can be performed, the Insert Data routine sets the prefix data offset equal to the size of the prefix plus the number of reserve characters plus one. The routine then calculates the length of the data to be shifted to be the current scan pointer setting minus the prefix data offset - this length is later placed in AVTPARM3. The routine decrements the number of reserved characters by the requested expand length and places the result in LCBISZE. Next, this routine enters the shift data loop and proceeds as described in the paragraph on the multiple-unit shift function.

The Insert subroutine of the Insert Data routine inserts data according to the parameters passed as input. The possible parameters are a pair of offsets - the prefix data offset (a halfword at RCB+4) and the prefix insert offset (a halfword at RCB+6) in the RCB, the length of the data to be inserted, and the actual data to be inserted - either a character string or a repeat character. If the data to be inserted is itself located in the buffer, the insertion is actually a left shift in the buffer.

The Insert Data subroutine calculates the offset at which data is to be inserted by subtracting the prefix insert offset from the prefix data offset. The routine passes this resulting offset to the Address Finder routine (IEDQAL) to obtain the end-of-unit address and the insert address. Upon return, the Insert Data routine calculates the count of bytes from the insert data to the end of the unit and compares the result to the length of data to be inserted. If the number of bytes to be inserted is less than the number of bytes to the end of the unit, all the data will fit in the current buffer unit. If the data to be inserted is a character string, the routine gets its address from AVTPARM3 and moves the data to the unit; if the data is a string of identical repeated characters, the routine gets the character from AVTPARM3 and enters a store character loop to insert the total number of characters requested. If all the data will not fit in the current unit, the Insert Data routine gets the address of the next unit and moves in the remaining data or repeat characters. After the specified count of characters has been inserted, the Insert subroutine returns to the point from which it was called.

External Routine: IEDQAL - Address Finder routine - to find an address in a unit.

Tables/Work Areas: AVT, buffer, ICB, SCB.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

Buffer Step Routine (Chart AX)

Module Name: IEDQAX

Entry Point: SCAN - called by a higher-level MH routine to return the address of the next sequential byte in the buffer being processed.

Functions: This routine returns to the calling routine the address of the next sequential byte in the buffer that is being processed.

The Buffer Step routine increments the scan pointer address register (RSCAN - register 5) and the scan pointer offset register (RSCANOFF - register 7) by one. It then compares RSCANOFF with the buffer prefix data size field (PRFSIZE). If RSCANOFF is higher, the end of data in the buffer has been passed, and this routine returns to the calling routine.

The Buffer Step routine next compares RSCAN to the end-of-unit register (REOUAD - register 11). If RSCAN is lower, the registers are correctly set, and this routine returns to the calling routine at a point that is four bytes beyond the return address in register 14.

If RSCAN is not lower that REOUAD, the end of the current unit has been passed. The routine gets the address of the next unit from the RCB of the current unit, sets RSCAN equal to the address of the first data byte in the new unit, and sets REOUAD to the address of the first byte beyond the end of the new unit. The routine then returns to the calling routine at a point that is four bytes beyond the return address in register 14.

External Routines: None.

Tables/Work Areas: Buffer currently being processed, AVT.

Attributes: Reentrant, serially reusable, refreshable, enabled, resident, problem program mode.

Binary Search Routine (Chart A1)

Module Name: IEDQA1

Entry Point: IEDQA101 - called through the User Interface routine (IEDQUI) by a higher level MH routine to search a table.

Functions: This routine searches a table that is arranged in collating sequence. It is primarily designed to search the Termname Table. On entry, the address of the last 8 bytes of the FORWARD macro-generated parameter list is in register 1.

The Binary Search routine compares the length of the input field to the length of the name field in a table entry (TNTENLEN). If the passed length is longer, this routine puts a X'00' return code in register 15 and returns to the calling routine through the Return Interface routine (IEDQLM). If the passed length is equal or shorter, the Binary Search routine sets the compare length for the main loop equal to the passed length.

The routine then clears the AVT field AVTDOUBI to zeros and moves the character string to be found to that AVT field if this move is necessary.

The Binary Search routine initializes the entry address to the address of the middle entry (TNTMIDEN) in the table, and sets the search extent to the full length of one entry (TNTENLEN+3) multiplied by the search extent factor (TNTSRCHX). The address of the last entry plus one is set by multiplying the full length of one entry by the number of entries (TNTLEN) and adding the product to the address of the first entry (TNTFIRST). The routine then enters the main processing loop at ENTRLOOP.

The Binary Search routine compares the entry address with the address of the last entry. If the entry address is not low, it points beyond the end of the table. The routine decrements the entry address by the search extent. If the entry address is within the table, the routine compares the passed field with the name field of that entry. If the entry name field is high, the routine decrements the entry address by the search extent. If the entry name field is low, the routine increments the entry address by the search extent.

After incrementing or decrementing the entry address, the Binary Search routine compares the search extent to the length of one full entry. If it has become less that the length of one entry, the table has been fully searched without finding a name equal to the passed field, and the routine puts a X'00' return code in register 15 before passing control to IEDQLM. If the search extent is not less than the length of one entry, it is divided by two and the loop continues with the end-of-table compare.

The Binary Search routine computes the offset to the entry (the ordinal index), places that offset in register 15, and returns control to the calling routine through IEDQLM.

External Routines: None.

Tables/Work Areas: AVT, Termname Table.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

Termname Table Code (Chart NT)

Module Name: IEDQTNT

Entry Point: IEDQTNT - activated by TCAM routines to calculate the Terminal Table address of an entry.

Function: This subroutine converts the two-byte ordinal index, or offset, to a Termname Table entry to its actual address in the Terminal Table.

This subroutine calculates the address of the entry by performing an effectice multiply of the index by the size of the entry. The code of the routine is generated at MCP assembly time and varies dependent on the length of the Termname Table.

On completion, the Termname Table code places the address of the Terminal Table entry in register 1 and branches to the routine that called it.

External Routines: None.

Tables/Work Areas: None.

Attributes: Resident, problem program mode, reentrant.

MESSAGE HANDLING - BUFFER DISPOSITION MODULES

Buffer Disposition Subtask (Chart BD)

Module Name: IEDQBD

Entry Points:

- IEDQBD01 - activated by the TCAM Dispatcher after the last segment of a message has been received or sent and processed by the MH up to the inmessage/outmessage subgroup, when the end of an invitation list is reached, and when the last buffer of a block has been sent to a buffered terminal.

- IEDQBD02 - activated by the TCAM Dispatcher to process an LCB from the Operator Awareness Message Router (IEDQNX).

Functions: This subtask returns unused buffers to the Buffer Return QCB and executes the INMSG/OUTMSG macro expansions. This subtask checks the parameter list for each macro in the INMSG/OUTMSG subgroup, checking the error word against the specified mask if execution of the function is conditional. If the routine represented by a parameter list requires a recalled header, the Buffer Disposition subtask tposts the ERB to the Disk I/O QCB to perform a recall. The recalled header is returned to Buffer Disposition by a tpost. Each routine that receives control from Buffer Disposition performs its functions and exits by tposting the recalled header (if one was passed to it) or the ERB back to Buffer Disposition to continue execution of the macro expansions. In the case of recalled headers, after the recalled header is tposted by the macro routine, it is returned to Buffer Disposition by CPB Initialization (IEDQFA). When an INEND or OUTEND macro expansion is detected, Buffer Disposition checks for distribution list, multiple routing, and checkpoint request. If any of these functions have been requested, the appropriate subtask receives control through a tpost. For output messages, the message just sent is then marked serviced. For both send and receive, the LCB is then tposted to itself.

If the error message bit is on (X'20') in LCBCHAIN, Buffer Disposition gets the address of the Operator Awareness Message Router (IEDQNX) from the AVT. Buffer Disposition then branches to the message router with the LCB and a chain of elements set up to be tposted to the ready queue. The Operator Awareness Message Router builds the error message, tposts the message to its Destination QCB, puts the address of the QCB for IEDQBD02 in the first word of the LCB, and tposts the LCB and the chain of elements to the ready queue.

External Routine: IEDQTNT - Termname Table code - to obtain the Terminal Table address of an entry.

Tables/Work Areas: LCB, SCB, AVT, Termname Table, Terminal Table, buffer prefix, QCB.

Attributes: Reusable, enabled, resident, refreshable, problem program mode.

Cancel Message Routine (Chart AR)

Module Name: IEDÇAR

Entry Point: IEDÇAR - activated by the Buffer Disposition subtask (IEDQBD) to cancel a message.

Functions: This routine sets a flaq in the buffer prefix (PRFCNCLN) to notify the Destination Scheduler and CPB Initialization tc cancel the messaqe currently beinq received cn the line. This flaq also stops multiple rcutinq and checkpcint functions that miqht apply to the messaqe beinq canceled.

If the cancel reouest is for a terminal in lock mode, this routine cancels the message, sets up the LCB tc repoll the terminal, and decrements the inrut sequence number in the terminal entry.

The Cancel Messaqe routine exits to the DSPCHAIN entry point of the TCAM Disratcher to tpost a chain cf elements that were passed to the Cancel Messaqe routine as input (a chain off reqister 1).

External Routines: IEDQTNT - Termname Table code - to qet the terminal entry address.

Tables/Work Areas: AVT, LCB, SCB, Terminal Table.

Attributes: Reusable, refreshable, enabled, resident.

Operator Awareness Messaqe Router (Chart NX)

Module Name: IEDQNX

Entry Point: IEDQNX - loaded at INTRO time by the Attach routine (IEDOOS) if the system console is not specified as the primary operator control terminal; can also be loaded by the Change Control Terminal routine (IEDQCN) if the primary cperator terminal is changed by an operator control ccmmand. When IEDQNX is in the system, it is activated by Buffer Disrcsition (IEDQBD) when an error has cccurred on a line.

Functions: This routine directs error messaqes to the primary operator control terminal when that terminal is nct the system console.

The Operatcr Awareness Messaqe Rcuter receives ccntrcl from the Buffer Disposition subtask (IEDQBD). The rcuter first remcves error-specific information from the LCB. This information was placed in the LCB by the ERP routine that qot ccntrol when the errcr was detected. This informaticn includes the line address (UCBNAME); the ccmmand code of the failinq CCW (LCBFESTR); the two status bytes of the CSW (LCBCSW+3); the first sense byte of the IOB (LCBSENSO); the TP operation codes of the last retry (LCBFLAG2) and of the first failinq CCW (LCBSENS1); and the addressinq characters, last four dial diqits, or pollinq characters (LCBERRCT). The Operator Awareness Messaqe Router ccnverts this error data to hexadecimal fcrmat and builds an error messaqe in a work area (AVTSAVE4). The router then qets a buffer from the buffer unit pool, moves the messaqe into the buffer, and sets up the buffer prefix and the SCB tc tpost the buffer to the Destination QCB fcr the primary operator ccntrol terminal. The router then links tc the buffer the chain of elements passed from the Buffer Disposition subtask, turns off the error-messaqe bit in the LCB, and passes control to the TCAM Dispatcher at entry point DSPCHAIN to tpost the chain of elements to the ready queue.

External Routines: None.

Tables/Work Areas: AVT, LCB, buffer prefix, QCB, SCB, Terminal Table.

Attributes: Resident, reentrant, refreshable.

Hold/Release Terminal Routine (Chart AS)

Module Name: IEDQAS

Entry Points:

- IEDQAS - activated by Buffer Disposition (IEDQBD) to hold a terminal.
- IEDQAS01 - called to release a terminal when a buffer unit from Operator Control or the time delay queue is tposted to this STCB.
- GETCPB - activated by the TCAM Dispatcher when a CPB that IEDQAS is waiting for is available.
- LCBRTN - activated by the TCAM Dispatcher when the Send Scheduler for the QCB being released is available.

Functions: This module has two distinct functions - to hold and to release a terminal. The functions are performed according to the entry point used by the calling routine.

If the Buffer Disposition subtask encounters terminal errors, it branches to the Hold/Release Terminal routine at the IEDQAS entry point to set the hold bit (TRMHELDN) in the appropriate terminal entry in the Terminal Table. A hold prevents messages from being transmitted to that terminal. The last message sent to the terminal is also held until the terminal is released (the address of the first unit of the last message sent is placed in the FEFO chain for held messages in the Priority QCB). The routine passes the input buffer to the Time Delay subtask.

Note: If the Terminal is in lock mode, it is not held and the lock message is retransmitted.

When a buffer unit from Operator Control or the time delay queue is tposted to the STCB for the Hold/Release Terminal routine, the routine is activated at the IEDQAS01 entry point to release a terminal. The Release Terminal routine first gets control of the Send Scheduler STCB so that the Destination QCB will not be modified until the release function is complete. The routine then moves the QCBINTFF chain onto the first of the regular FEFO chain (requires one write, therefore one CPB) and turns off the terminal entry hold bit. The Release Terminal routine then places the Send Scheduler STCB in the LCB STCB chain or in the dial-out call queue for dial lines.

This module exits to the DSPPOST or the DSPCHAIN entry point of the TCAM Dispatcher to place elements on the ready queue.

External Routines:

- IEDQTNT - Termname Table code - to locate the terminal entry.

- IEDQHG01 - Time Delay subtask - to insert an element on the time delay queue.

- IEDQHG02 - Time Delay subtask - to remove an element from the time delay queue.

- IGG019RB or IGG019RO - TCAM Dispatcher - the DSPPOSTR entry point, to tpost a buffer.

Tables/Work Areas: AVT, SCB, Terminal Table, buffer prefix, QCB, CPB, STCB, DEB, UCB, DCB.

Attributes: Reusable, enabled, resident, problem program mode.

Create an Error Message Routine and Subtask (Chart AT)

Module Name: IEDQAT

Entry Point:

- IEDQAT01 - activated by the Redirect a Message routine (IEDQAZ) to build an error message and tpost it to its destination.

- STCBAT+2 - activated by the TCAM Dispatcher to return an empty unit to contain part of the error message.

Functions: The Create an Error Message routine builds an error message in the buffer and tposts the buffer to its destination.

The Create an Error Message routine first determines whether the message buffer already contains an error message. If there is an error message in the buffer, the routine exits by tposting the ERB back to the Buffer Disposition subtask.

This routine gets the length of the error message from the parameter list of the ERRORMSG macro expansion or, if that byte is zero, from the first byte of the message itself. For the latter case, the address of the message is in the ERRORMSG macro-generated parameter list.

If the buffer does not contain a message, the routine determines whether the error message fits between the scan pointer (the end of the header) and the end of the buffer. If the message will not fit, the Create an Error Message routine determines whether one additional unit will provide enough space for the message. If not, the routine truncates the message. If an empty unit is needed, the routine tposts an ERB to request one unit to the Buffer Request QCB and returns control to the TCAM Dispatcher to wait for the buffer request to be satisfied. When the unit is available, the Dispatcher activates the Create an Error Message routine. At this pcint the routine links the new unit into the buffer.

After getting a new buffer unit, the routine uses the User Interface routine (IDEQUI) to give control to the Insert Data routine (IEDOAF), which inserts the error message in the buffer. On return, if there is a user-written exit routine present, the Create an Error Message routine passes control to that routine. Upon return, the Create an Error Message routine tposts the buffer that contains the message to its destination. (The Redirect a Message routine, which is dispatched by Buffer Disposition, determines the buffer destination, then places the error message in the buffer.)

External Routine: IEDQUI - User Interface rcutine - to activate the Insert Data rcutine (IEDOAF), which inserts the error message in the buffer.

Tables/Work Areas: AVT, buffer currently being processed, LCB, SCB.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mcde.

Redirect a Message Routine (Chart AZ)

Module Name: IEDQAZ

Entry Point: IEDQAZ01 - activated by the Buffer Disposition subtask (IEDOBD) to redirect a message to a specified destination.

Functions: This module redirects a message to the destination specified by a user.

The Redirect a Message routine first determines whether the redirect destination is in an option field, and if it is, links to the Locate Option Field Address routine (IEDQAE) via the User Interface routine to get the option field address. Upon return, if the option field is not fcund, the Redirect a Message routine branches to its error handling locp (TESTDEAD). If the option field address is found, the Redirect a Message routine builds a parameter list for the Binary Search routine (IEDQA1) and branches through the User Interface routine to get the buffer destination key. If the readirect destination name is explicitly specified cn the macro instruction, the routine also builds the parameter list and branches to the Binary Search routine. Upon return, if the redirect destination name is not in the Termname Table, the Redirect a Message routine branches to its error handling locp; ctherwise, the routine passes the destination key returned by the Binary Search routine to the Lookup Terminal Entry routine (IEDQAV). If the redirect destination is the crigin or the original destination, the key is from either the PRFSRCE (origin) or the PRFDEST (criginal destination) field in the buffer prefix. In this case the routine also passes the key tc the Lcokup Terminal Entry routine. On return, the Redirect a Message routine determines whether the destination is a TSO device. If the destination is a TSO device, the routine branches to its error handling loop; otherwise the routine posts the QCB for the message to its destination by exiting to the DSPCHAIN entry point of the TCAM Dispatcher.

If an error message is being redirected to its destination, the Redirect a Message routine does not post the QCB; instead, the routine sets the SCBMACR field equal to the address of the error message parameter list and then exits to the Create and Error Message routine (IEDQAT).

The error handling loop (TESTDEAD) of the Redirect a Message routine determines whether a dead-letter queue is defined. If the queue is defined, the routine gets the destination key for the queue, passes it to the Lookup Terminal Entry routine, and posts the message to that destination. If no dead-letter queue is defined, the Redirect a Message routine does not redirect the message, but posts the buffer to the Buffer Disposition QCB and exits to the TCAM Dispatcher (DSPCHAIN).

External Routines:

- IEDQUI - User Interface routine - to activate the following modules:

  IEDQAE - Locate Option Field Address routine - to get the address of the option field.

  IEDQA1 - Binary Search routine - to get the destination key for the message.

- IEDQAV - Lookup Terminal Entry routine - to get the terminal entry for a specified destination.

Tables/Work Areas: AVT, recalled-header buffer, SCB.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

Message Generation Routine (Chart BL)

Module Name: IEDQBL

Entry Point: IEDQBL - activated by the Buffer Disposition subtask (IEDQBD) when a MSGGEN macro is specified in an inmessage or outmessage subgroup of an MH.

Functions: The Message Generation routine finds a user-provided message, moves the message to the SCB for the currently connected terminal, translates the message to the appropriate line code, and posts the ERB that contains the message to the Activate subtask (IEDQKA02) to cause the message to be sent.

External Routines: None.

Tables/Work Areas: ICB, SCB, AVT, DCB.

Attributes: Reusable, refreshable, enabled, resident, problem program mode.

Log Message Routine (Chart BY)

Module Name: IEDQBY

Entry Point: IEDQBY - called by the Buffer Disposition subtask (IEDQBD) when a LOG macro is specified in an INMSG or OUTMSG subgroup of an MH.

Functions: This routine tposts a recalled header to the Destination QCB specified in the LOG macro for logging the message. This tpost activates the Log Scheduler, which logs the message.

The Log Message routine exits to the DSPCHAIN entry point in the TCAM Dispatcher to actually tpost the element.

External Routines: None.

Tables/Work Areas: SCB, buffer prefix.

Attributes: Reusable, refreshable, resident, enabled, problem program mode.

Log Scheduler (Chart BZ)

Module Name: IEDQBZ

Entry Point: IEDQBZ - activated by the TCAM Dispatcher when the LOG LCB is on the ready queue or when a buffer has been tposted to the LOG Destination QCB.

Functions: This routine schedules the logging of messages. The Log Scheduler may be activated under the following conditions.

- A Destination QCB with a buffer on the ready queue: the Log Scheduler moves its STCB to the LOG LCB, tposts the LCB to itself, and exits to the DSPBYPAS entry point of the Dispatcher. This action passes the message buffer to the Destination Scheduler.

- An LCB tposted to itself on the ready queue: the Log Scheduler tposts the ERB in the LCB to the Disk I/O QCB to recall one buffer. The scheduler exits to the DSPPOST entry point of the TCAM Dispatcher.

- A ERB tposted to an LCB on the ready queue: the Log Scheduler checks any outstanding WRITE commands and frees the buffers. It then issues WRITE commands for any buffer units to be written. If the end-of-message was written, the Lcq Scheduler tposts the ERB to the ICB again to handle the last buffer checks. If there are no more messages to lcq for this QCB, the scheduler moves its STCB back to the Destination QCB by exiting to the DSPUNAV entry point of the TCAM Dispatcher.

If an error occurs during the writing of a buffer, the CHECK macro issues an exit to the SYNAD routine specified in the DCB. The SYNAD routine must return to the CHECK macro. The user-written SYNAD routine must conform to BSAM standards.

External Routines:

- OS Getmain routine (SVC 4) - to obtain main storage.
- IGG019RB or IGG019RO - TCAM Dispatcher - the DSPUNAVP entry point, to move an STCB; and the DSPPOSTR entry point, to tpost an element.
- OS Check routine - to check a write operation.
- OS Write routine - to write a unit.

Tables/Work Areas: ICB, SCB, DCB, AVT, QCB.

Attributes: Reusable, refreshable, resident, enabled, problem program mode.

Multiple Routing Subtask (Chart BA)

Module Name: IEDQBA

Entry Point: IEDQBA01 - activated when an ERB or a buffer is tposted to the Multiple Routing QCB in order to queue a message for additional destinations.

Functions: The Multiple Routing subtask identifies additional destinations specified in a buffer and tposts the message to each of these destinations, in order.

On initial entry, the ERB within the ICB is tposted to the Multiple Routing subtask with the address of the first buffer of a recalled message in the ERB chain field (ICBERBCH) of the ICB. The subtask counts the number of buffers passed (initially only one) and keeps the addresses of both the first buffer of the message and of the current buffer. At initial entry, the address of the current buffer is also the address of the first buffer. The subtask places the current buffer address in the ICB and places the current count of buffers in the current buffer.

The Multiple Routing subtask uses the offset from the first-secondary-destination field (SCBMRFSD) of the SCB to set the scan pointer. The subtask loads the address of the Forward routine parameter list from the SCBMRFPL field of the SCB and then links to the Forward routine (IEDQA5) through the User Interface routine (IEDQUI). When the Multiple Routing subtask regains control, if the destination was found to be valid, the subtask determines whether this destination is a distribution list. If it is, the subtask sets the LCB to tpost the message to each destination in the list before it regains control. The subtask then tposts the recalled message to the appropriate QCB (distribution list or single entry) and branches to the TCAM Dispatcher.

If the Forward routine returns to the Multiple Routing subtask with an invalid destination specified, the subtask reactivates the Forward routine to find the next destination.

After the first buffer of the recalled message has been tposted to the first secondary destination, the Multiple Routing subtask is reentered with another recalled buffer. The subtask gets the current buffer address from the LCB. If the current buffer is itself the first buffer of a message, the subtask sets the current buffer count to one and sets the buffer just tposted to the subtask as the current buffer. If the new buffer is not the first buffer of a message, the buffer whose address was recovered from the LCB remains the current buffer and the buffer count is recovered from it. The subtask again sets the scan pointer from the first-secondary-destination field of the LCB and then passes control to the Forward routine.

If return from the Forward routine indicates that a destination field in the current buffer is incomplete, the buffer is part of a multiple-buffer header and the next buffer must be recalled. The subtask saves the data in the current buffer and stores the address of the first buffer in the message, with all other buffers that have been passed linked to it, in the ERB. The subtask increments the current buffer count by one, places the count in the ERB, and tposts the ERB to the Disk I/O QCB in order to recall another buffer.

The ERB is tposted to the Multiple Routing subtask when another buffer has been retrieved. At this point the subtask determines that two or more buffers are being passed (to distinguish from initial entry) and counts the number of buffers being passed. The subtask saves the address of the last buffer in the chain in the LCB, along with the address of the second buffer and any subsequent buffers linked to it. The last buffer in the chain is the new current buffer. The subtask sets the first-secondary-destination field to the offset of the first data byte in the current buffer. Then the subtask sets the scan pointer and links to the Forward routine.

If return from the Forward routine indicates that the EOA string has been encountered, there are no more destinations. The subtask sets the last recalled header to point to any recalled subsequent buffers, and sets all buffers with the address of the Buffer Return QCB. The subtask then links the ERB to the last buffer and puts the address of the Buffer Disposition QCB in the ERB. The subtask clears the first-secondary-destination field of the SCB to zeroes and tposts the buffers and ERB via an exit to the TCAM Dispatcher.

External Routine: IEDQUI - User Interface routine - to activate the Forward routine (IEDQA5), which deciphers secondary destinations in the message header.

Tables/Work Areas: SCB, AVT, LCB, buffer prefix.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

Lock Routine (Chart BE)

Module Name: IEDQBE

Entry Point: IEDQBE - called through the User Interface routine by the LOCK macro expansion.

Functions: This routine locks the connection between the currently connected terminal and its process entry destination by setting a switch in the SCBSTATE byte of the SCB. The extended lock switch is set if the request is by the LOCK macro.

If the routine finds that a terminal is not connected, the routine exits immediately to the Return Interface routine (IEDQLM) with X'04' in register 15. If a terminal is connected but the destination is the Buffer Return QCB or is not an application program, the Lock routine exits with a return code of X'08' in register 15. A return code of X'08' also indicates that the buffer has an indiacted length of zero or that the buffer is not a header buffer. A X'0C' return code indicates that there is no GET DCB open for the destination application program. The successful return code is X'00'.

External Routine: IEDQTNT - Termname Table code - to get a terminal entry address.

Tables/Work Areas: AVT, LCB, QCB, SCB.

Attributes: Reusable, refreshable, enabled, resident, problem program mode.

Unlock Routine (Chart BF)

Module Name: IEDQBF

Entry Point: IEDQBF - called through the User Interface routine by the UNLOCK macro expansion.

Functions: This routine unlocks the currently connected terminal by turning off the lock bits in the SCBSTATE byte of the SCB (if they were on).

If the currently connected terminal is not locked, a return code of X'04' is placed in register 15. Otherwise, X'00' is returned in register 15.

External Routines: None

Tables/Work Areas: AVT, SCB.

Attributes: Reusable, refreshable, enabled, resident, problem program mode.

Distribution List Subtask (Chart BC)

Module Name: IEDQBC

Entry Point: IEDQBC - activated by the TCAM Dispatcher when a message is sent to a destribution list entry in the Terminal Table.

Functions: This module tposts the buffers of a message to each of the destinations specified in the distribution list to which the message was routed.

The Distribution List subtask tposts the message to the Destination QCB for the first or next entry in the distribution list and exits to the Dispatcher. When the Destination Scheduler regains control, it finds the duplicate header bit on in the buffer prefix and tposts the buffer to the QCB indicated in LCBRCQCB - the Distribution List QCB. In this manner, the Distribution List subtask regains controls to tpost the message to the next destination.

When the buffer has been returned for the last destination in the list, if multiple routing is not active, the buffer is tposted to the Buffer Return QCB and the LCB is tposted to the Buffer Disposition QCB. If multiple routing is active, the Distribution List subtask tposts the buffer to the Multiple Routing QCB and sets LCBRCQCB to refer to the Multiple Routing QCB.

The Distribution List subtask exits to the DSPCHAIN entry point of the TCAM Dispatcher.

External Routine: IEDQTNT - Termname Table code - to get the Terminal Table address of the distribution list entry and of each entry in the distribution list.

Tables/Work Areas: AVT, LCB, SCB, buffer prefix, Termname Table, Terminal Table, CCB.

Attributes: Reusable, refreshable, enabled, resident, problem program mode.

Cascade List Subtask (Chart BG)

Module Name: IEDQBG

Entry Point: IEDQBG - activated by the TCAM Dispatcher when a message is sent to a cascade list entry in the Terminal Table.

Functions: This module tposts the buffers of a message to one destination in the cascade list to which the message was routed.

The Cascade List subtask examines the terminal entry for each entry in the list in order, searching for the first terminal that can accept, is not held, has an open DCB, and has the fewest number of messages queued for it. Once an entry that meets these conditions is found, the Cascade List subtask tposts the message to its Destination QCB and resets the destination fields in the SCB and in the buffer prefix. If a terminal that meets all these conditions cannot be found, the message is tposted to the Destination QCB for the first entry in the list.

The Cascade List subtask exits to the DSPPCST entry point of the TCAM Dispatcher.

External Routine: IEDQTNT - Termname Table code - to get the Termname Table address of the cascade list entry and of each item in the cascade list.

Tables/Work Areas: AVT, LCB, SCB, Termname Table, Terminal Table, buffer prefix, DCB, QCB.

Attributes: Reusable, refreshable, enabled, resident, problem program mode.

MESSAGE HANDLING - QUEUE MANAGEMENT ROUTINES

Destination Scheduler (Chart HM)

Module Name: IEDQHM

Entry Points:

- IEDQHM - from the TCAM Dispatcher with a full buffer to be queued.
- IEDQHM02 - from the Reusability-Copy subtask (IGG019RP) or the Start-up Message routine (IGG019R6) with one unit of a buffer to be queued.
- IEDQHM03 - from IEDQHM, the Reusability-Copy subtask and CPB Initialization to find the SCB address if the first FEFC message on the input Destination QCB is being sent.

Functions: This subtask assigns a buffer to a location in a message queues data set (reusable disk, nonreusable disk, or main storage, as applicable) by tposting the buffer to the Disk I/O QCB. The buffer is chained to other buffers of the message, and this message is chained to other messages in the same queue.

The first buffer of the message contains the address of the next segment and of the first buffer of next message. Each buffer has the address of the next message segment (if it is not the last buffer), the address of the additional records (if any), and, if it is not the first buffer of a message, the address of the first buffer of this message.

If the message is disk queued, the first and the last buffers of a message contain the queue-back chain pointers. The queue-back chain is a time sequential record of the events (both sending and receiving) for a Destination QCB. If the QCB represents the destination of the message, the first buffer appears in the queue-back chain from that QCB. If the QCB represents the source of the message (line or terminal) the last buffer of the message appears in the queue-back chain of the QCB.

Disk message queuing (reusable or nonreusable) is accomplished by assigning relative record numbers ahead on disk. There is a value called "address" for both reusable (AVTRADDR) and nonreusable (AVTNADDR) disk data sets in the AVT. There is a correspondence between the value of address and the physical location (MBBCCHHR) of the record on disk. When the address, modulo the total number of records in the data set, is used, there is a one-to-one correspondence. When a first buffer that is not also the last buffer of a message is received, the Destination Scheduler reserves a value of address for the first buffer of the next message for that Destination QCB and a value of address for the next buffer segment of this message. Sequential values of address are reserved for any additional records required. When a subsequent buffer that is not also the last buffer of a message is received, the Destination Scheduler reserves locations for the next-segment and for additional records. When a last buffer of a message is received, no location for next-segment is reserved.

The main storage message queues data set is not divided into numbered records. One record corresponds in size to one buffer unit. Units are not assigned ahead as in disk queuing; however, the messages in one queue are chained together and the buffers of a message are chained together. The value AVTCADDR is similar to the address value on disk. AVTCADDR corresponds to the number of units used out of the total number reserved for main storage queues. Chaining is not done by record number for main storage queues, but by the actual address. The additional records are located through the TIC fields in the RCB of the buffer units.

When the first buffer of a message to be main storage queued is received, the buffer is chained to the first buffer of the previous message in the queue. When a subsequent buffer of a message to be main storage queued is received, the buffer is chained to the previous buffer of this message. The number of units corresponding to the number of units in the buffer is removed from the buffer unit pool. If the messages for this Destination QCB are to be main storage queued only, the Destination Scheduler tposts these units to the Buffer Return QCB and places the buffer in the main storage queue of messages for the QCB. If the message is to be disk queued also, the Destination Scheduler copies the the message into the units from the buffer unit pool and places the buffer in the main storage message queue for the QCB. The original buffer, therefore, is disk queued.

When the Destination Scheduler receives the last buffer of a message and the scheduler associated with this Destination QCB is in the STCB chain of the Destination QCB, the Destination Scheduler issues a BALR to a subroutine of this scheduler. This BALR notifies the scheduler that a message is available.

The Destination Scheduler exits to the TCAM Dispatcher at DSPPOST or DSPDISP or, if called by the Reusability-Copy subtask, to the calling routine.

External Routines:

- IEDQTNT - Termname Table code - to obtain the address of the Terminal Table entry from the offset into the Termname Table.

- The subroutine of the scheduler for the Destination QCB.

Tables/Work Areas: LCB, DCB, SCB, buffer prefix, QCB, AVT, Terminal Table, disk data area.

Attributes: Reusable, refreshable, enabled, resident, problem program mode.

Destination Scheduler - Main Storage Queuing Only (Chart HM1)

Module Name: IEDQHM1

Entry Points:

- IEDQHM1 - from the TCAM Dispatcher with a full buffer to be queued.

- IEDQHM02 - activated by the Start-up Message routine (IGG019R6) with one unit of a buffer to be queued.

- IEDQHM03 - called from IEDQHM1 to find the SCB address if the first FEFO message on the input Destination QCB is being sent.

Functions: This subtask assigns a buffer to a location in a main storage message queues data set. The buffer is chained to other buffers of the message, and this message is chained to other messages in the same queue.

The first buffer of the message contains the address of the next segment and of the first buffer of next message. Each buffer has the address of the next message segment (if it is not the last buffer), the address of the additional records (if any), and, if it is not the first buffer of a message, the address of the first buffer of this message.

The main storage message queues data set is not divided into numbered records. One record corresponds in size to one buffer unit. Units are not assigned ahead as in disk queuing; however, the messages in one queue are chained together and the buffers of a message are chained together. The value AVTCADDR is similar to the address value on disk. AVTCADDR corresponds to the number of units used out of the total number reserved for main storage queues. Chaining is not done by record number for main storage queues, but by the actual address. The additional records are located through the TIC fields in the RCB of the buffer units.

When the first buffer of a message to be main storage queued is received, the buffer is chained to the first buffer of the previous message in the queue. When a subsequent buffer of a message to be main storage queued is received, the buffer is chained to the previous buffer of this message. The number of units corresponding to the number of units in the buffer is removed from the buffer unit pool. The Destination Scheduler tposts these units to the Buffer Return QCB and places the buffer in the main storage queue of messages for the QCB.

When the Destination Scheduler receives the last buffer of a message and the scheduler associated with this Destination QCB is in the STCB chain of the Destination QCB, the Destination Scheduler issues a BAIR to a subroutine of this scheduler. The BAIR notifies the scheduler that a message is available.

The Destination Scheduler exits to the TCAM Dispatcher at DSPPOST or DSPDISP.

External Routines:

- IEDQTNT - Termname Table code - to obtain the address of the Terminal Table entry from the offset into the Termname Table.

- The subroutine of the scheduler for the Destination QCB.

Tables/Work Areas: LCB, DCB, SCB, buffer prefix, QCB, AVT, terminal entry, disk data field.

Attributes: Reusable, refreshable, enabled, resident, problem program mode.

Destination Scheduler - Disk Queuing Only (Chart HM2)

Module Name: IEDQHM2

Entry Points:

- IEDQHM2 - from the TCAM Dispatcher with a full buffer to be queued.

- IEDQHM02 - from the Reusability-Copy subtask (IGG019RF) or the Startup Message routine (IGG019R6) with one unit of a buffer to be queued.

- IEDQHM03 - called from IEDQHM2, the Reusability-Copy subtask, and CPB Initialization to find the SCB address if the first message on the input Destination QCB is being sent.

Functions: This subtask assigns a buffer to a location in a message queues data set (reusable disk or nonreusable disk, as applicable) by tposting the buffer to the Disk I/O QCB. The buffer is chained to other buffers of the message, and this message is chained to other messages in the same queue.

The first buffer of the message contains the address of the next segment and of the first buffer of next message. Each buffer has the address of the next message segment (if it is not the last buffer), the address of the additional records (if any), and, if it is not the first buffer of a message, the address of the first buffer of this message.

When the message is disk queued, the first and the last buffers of a message contain the queue-back pointers. The queue-back chain is a time sequential record of the events (both sending and receiving) for a Destination QCB. If the QCB represents the destination of the message, the first buffer appears in the queue-back chain from that QCB. If the QCB represents the source of the message (line or terminal) the last buffer of the message appears in the queue-back chain of the QCB.

Disk message queuing (reusable or nonreusable) is accomplished by assigning relative record numbers ahead on disk. There is a value called "address" for both reusable (AVTRADDR) and nonreusable (AVTNADDR) disk data sets in the AVT. There is a correspondence between the value of address and the physical location (MBBCCHHR) of the record on disk. When the address, modulo the total number of records in the data set, is used, there is one-to-one correspondence. When a first buffer that is not also the last buffer of a message is received, the Destination Scheduler reserves a value of address for the first buffer of the next message for that Destination QCB and a value of address for the next buffer segment of this message. Sequential values of address are reserved for any additional records required. When a subsequent buffer that is not also the last buffer of a message is received, the Destination Scheduler reserves locations for the next-segment and for additional records. When a last buffer of a message is received, no location for next-segment is reserved.

When the Destination Scheduler receives the last buffer of a message and the scheduler associated with this Destination QCB is in the STCB chain of the Destination QCB, the Destination Scheduler issues a BALR to a subroutine of this scheduler. This BALR notifies the scheduler that a message is available.

The Destination Scheduler exits to the TCAM Dispatcher at DSPPOST or DSPDISP or, if called by the Reusability-Copy subtask, to the calling routine.

External Routines:

- IEDQTNT - Termname Table code - to obtain the address of the Terminal Table entry form the offset into the Termname Table.

- The subroutine of the scheduler for the Destination QCB.

Tables/Work Areas: LCB, DCB, SCB, buffer prefix, QCB, AVT, terminal entry, disk data field.

Attributes: Reusable, refreshable, enabled, resident, problem program mode.

CPB Initialization (Chart FA)

Module Name: IEDQFA

Entry Points:

- IEDQFA - the CPB Initialization routine - called by the TCAM Dispatcher to queue a buffer on disk or to obtain full buffers from a main storage or disk message queue.

- IEDQFQ - the CPB Cleanup routine - called by the TCAM Dispatcher to handle CPBs after disk operations and to fill buffers from data in CPBs that have read a record or from units in a main storage queue.

Functions: This module consists of two routines: IEDQFA, the CPB Initialization routine, which initializes CPBs to write or read buffer units to or from disk; and IEDQFQ, the CPB Cleanup routine, which handles the CPBs after disk I/O has been completed.

There are three different types of input to the IEDQFA entry point of the CPB Initialization module, and the functions performed depend on the input:

- A buffer to be written on disk. The input contains the relative record numbers for the units of the buffer. The buffer is tposted to the Disk I/O QCB by the Destination Scheduler. CPB Initialization builds the CPBs to write the record on disk.

- Buffers to be flagged serviced or canceled on the message queues data set. These buffers were tposted to the Disk I/O QCB by the Buffer Disposition subtask. CPB Initialization builds the CPBs to write the requested records on disk.

- An ERB tposted to obtain full buffers. The Send Scheduler tposts the ERB to obtain full buffers to satisfy an initial request for sending. Buffer Disposition tposts the ERB for recall. The EOB/ETB Handling subtask tposts the ERB for recalled buffers. The Get Scheduler tposts the ERB to obtain full buffers to satisfy a GET command. PCI Appendage tposts the ERB to get buffers for subsequent transmission. For recall, the address of the first byte of data to put in the buffer is in the SCBDEOB field of the SCB; otherwise, it is in SCBSCSEG. CPB Initialization builds CPBs for disk reads, obtains the buffers, fills them with data, and tposts either the buffer or the ERB with the buffers to the appropriate QCB.

CPB Initialization exits to the EXCP Driver when there are either no elements to process or no CPBs available. If there is no disk in the system, CPB Initialization exits to the TCAM Dispatcher.

If a logical read error occurs during a recall when a request is made to read a record and the record number read does not agree with the requested record number, CPB Initialization sets the appropriate error flags and returns the ERB to the specified QCB.

There are two types of input to the IEDQFQ, or CPB Cleanup, part of the CPB Initialization module, and the functions performed depend on the input:

- QCB - the CPB Cleanup QCB is tposted to itself to initiate cleanup of the CPBs. Disk End Appendage tposts the QCB after disk I/O is complete and the completed CPBs have been placed on AVTDKAPQ. CPB Cleanup resets the CPB buffer address into CPBXREA and examines CPBFLAG for X'80'. If this flag is set, the CPBs belong to the Reusablity-Copy subtask, so CPB Cleanup enqueues the CPBs on AVTREUSQ. If the flag is not set, CPB Cleanup returns the CPBs for disk writes to the CPB free pool, and if the CPB was for a disk read, places the data in the buffers and returns the CPB to the CPB free pool.

- Buffer - a buffer unit is tposted to satisfy a request from CPB Cleanup. Buffer Return tposts the available unit to the Cleanup QCB. CPB Cleanup locates the CPB that is associated with this buffer on the No-buffer queue. The buffer is associated with the CPB to be later processed by the IEDQFA part of CPB Initialization.

After all CPBs on the CPB Cleanup QCB have been processed, CPB Cleanup branches to the Reusability-Copy subtask (IGG019RP) if either the "Reus first time" switch is on, the "Copy needs control" bit is set, or AVTREUSQ has elements on it. Otherwise, CPB Cleanup exits to IEDQFA.

External Routines:

- IGG019RB or IGG019RO - TCAM Dispatcher - to place an element on the ready queue by priority.

- IEDQHM03 in IEDQHM - Destination Scheduler - to find the SCB address when the first FEFO message on the input Destination QCB is being sent.

Tables/Work Areas: LCB, SCB, buffer prefix, Terminal Table, AVT, QCB, CPB, DCB, disk data field.

Attributes: Reusable, refreshable, enabled, resident, problem program mode.

CPB Initialization - Main Storage Queuing Only (Chart FA1)

Module Name: IEDQFA1

Entry Points:

- IEDQFA1 - the CPB Initialization routine - called by the TCAM Dispatcher to obtain full buffers from a main storage queue or to flag a message serviced.

- IEDQFQ - the CPB Cleanup routine - called by the TCAM Dispatcher with a buffer unit that is to be used to build a buffer for a main storage queue.

Functions: This module consists of two routines: IEDQFA1, the CPB Initialization routine, which gets full buffers from a main storage queue and flags a message serviced; and IEDQFQ, the CPB Cleanup routine, which builds buffers for a main storage queue.

There are two dirrerent types of input to the IEDQFA1 entry point of the CPB Initialization module, and the functions performed depend on the input:

- Buffers to be flagged serviced or canceled on the message queues data set. These buffers were tposted to the Disk I/O QCB by the Buffer Disposition subtask. CPB Initialization either frees the buffer from the message or sets the 'cancel' flag in the message on the message queue.

- An ERB tposted to obtain full buffers. The Send Scheduler tposts the ERB to obtain full buffers to satisfy an initial request for sending. Buffer Disposition tposts the ERB for recall. The EOB/ETB Handling subtask tposts the ERB for recalled buffers. The Get Scheduler tposts the ERB to obtain full buffers to satisfy a GET command. PCI Appendage tposts the ERB to get buffers for subsequent transmission. For recall, the address of the first byte of data to put in the buffer is in the SCBDEOB field of the SCB; otherwise, it is in SCBSCSEG. CPB Initialization obtains the buffers, fills them with data, and tposts either the buffer or the ERB with the buffers to the appropriate QCB.

CPB Initialization exits to the TCAM Dispatcher when there are no elements to process. There is one type of input to the IEDQFQ, or CPB Cleanup, part of the CPB Initialization module - a buffer. The Buffer Return routine tposts an available buffer unit to IEDQFQ to satisfy an ERB request. CPB Cleanup locates the portion of the message to be placed in this buffer and then branches to IEDQFA1 for buffer processing.

External Routines:

- IGG019RB or IGG019RO - TCAM Dispatcher - to place an element on the ready queue by priority.
- IEDQHM03 in IEDQHM1 - Destination Scheduler - to find the SCB address when the first FEFO message on the input Destination QCB is being sent.

Tables/Work Areas: LCB, SCB, buffer prefix, Terminal Table, AVT, QCB, CPB, DCB, disk data field.

Attributes: Reusable, refreshable, enabled, resident, problem program mode.

CPB Initialization - Disk Queuing Only (Chart FA2)

Module Name: IEDQFA2

Entry Points:

- IEDQFA2 - the CPB Initialization routine - called by the TCAM Dispatcher to queue a buffer on disk or to obtain full buffers from a disk message queue.
- IEDQFO - the CPB Cleanup routine - called by the TCAM Dispatcher to handle CPBs after disk operations and to fill buffers from data in CPBs that have read a record.

Functions: This module consists of two routines: IEDQFA2, the CPB Initialization routine, which initializes CPBs to write or read buffer units to or from disk; and IEDQFO, the CPB Cleanup routine, which handles the CPBs after disk I/O has been completed.

There are three different types of input to the IEDQFA2 entry point of the CPB Initialization module, and the functions performed depend on the input:

- A buffer to be written on disk. The input contains the relative record numbers for the units of the buffer. The buffer is tposted to the Disk I/O QCB by the Destination Scheduler. CPB Initialization builds the CPBs to write the record on disk.
- Buffers to be flagged serviced or canceled on the message queues data set. These buffers were tposted to the Disk I/O QCB by the Buffer Disposition subtask. CPB Initialization builds the CPBs to write the requested records on disk.
- An ERB tposted to obtain full buffers. The Send Scheduler tposts the ERB to obtain full buffers to satisfy an initial request for sending. Buffer Disposition tposts the ERB for recall. The EOB/ETB Handling subtask tposts the ERB for recalled buffers. The Get Scheduler tposts the ERB to obtain full buffers to satisfy a GET command. PCI Appendage tposts the ERB to get buffers for subsequent transmission. For recall, the address of the first byte of data to put in the buffer is in the SCB; otherwise, it is

in SCBSCSEG. CPB Initialization builds CPBs for disk reads, obtains the buffers, fills them with data, and tposts either the buffer or the ERB with the buffers to the appropriate QCB.

CPB Initialization exits to the EXCP Driver when there are either no elements to process or no CPBs available.

If a logical read error occurs during a recall when a request is made to read a record and the record number read does not agree with the requested record number, CPB Initialization sets the appropriate error flags and returns the ERB to the specified QCB.

There are two types of input to the IEDQFQ, or CPB Cleanup, part of the CPB Initialization module, and the functions performed depend on the input:

- QCB - the CPB Cleanup QCB is tposted to itself to initiate cleanup of the CPBs. Disk End Appendage tposts the QCB after disk I/O is complete and the completed CPBs have been placed on AVTDKAPQ. CPB Cleanup resets the CPB buffer address into CPBXREA and examines CPBFLAG for X'80'. If this flag is set, the CPBs belong to the Reusability-Copy subtask, so CPB Cleanup enqueues the CPBs on AVTREUSQ. If the flag is not set, CPB Cleanup returns the CPBs for disk writes to the CPB free pool and, if the CPB was for a disk read, places the data in the buffers and returns the CPB to the CPB free pool.

- Buffer - a buffer unit is tposted to satisfy a request from CPB Cleanup. Buffer Return tposts the available unit to the Cleanup QCB. CPB Cleanup locates the CPB that is associated with this buffer on the No-buffer queue. The buffer is associated with the CPB to be later processed by the IEDQFA2 part of CPB Initialization.

After all CPBs on the CPB Cleanup QCB have been processed, CPB Cleanup branches to the Reusability-Copy subtask (IGG019RP) if either the "Reus first time" switch is on, the "Copy needs control" bit is set, or AVTREUSQ has elements on it. Otherwise, CPB Cleanup exits to IEDQFA.

External Routines:

- IGG019RB or IGG019RO - TCAM Dispatcher - to place an element on the ready queue by priority.

- IEDQHM03 in IEDQHM2 - Destination Scheduler - to find the SCB address when the first FEFO message on the input destination QCB is being sent.

Tables/Work Areas: LCB, SCB, buffer prefix, Terminal Table, AVT, QCB, CPB, DCB, disk data field.

Attributes: Reusable, refreshable, enabled, resident, problem program mode.

EXCP Driver (Chart RC)

Module Name: IGGC19RC

Entry Points:

- IGG019RC - the EXCP Driver routine - called by the CPB Initialization routine to start disk I/O.

- IEDQFP - the Convert routine - called by IGG019RC to convert the absolute record number to the MBBCCHHR address in the disk data set.

Functions: The EXCP Driver routine of this module chains the CPBs that were begun by CPB Initialization in the proper sequence, adds Seek and Search-TIC CCWs where necessary, and sets the absolute disk address in the IOB. EXCP Driver then calls IOS with the EXCP macro to start disk I/O on the line. When the channel program is completed, EXCP Driver passes additional CPBs to the IOB for the Disk End Appendage to retry, and then returns to the TCAM Dispatcher.

There are three major steps in this routine: inserting by CC priority, optimizing the disk channel program, and issuing an EXCP.

- Insert by CC priority

The EXCP Driver routine takes each CPB from the input queue (AVTINCPO) and uses the Convert routine to convert CPBADDR to the MBBCCHHR address.

The Convert routine converts an absolute record number to an MBBCCHHR disk address for a multi-volume TCAM disk message queue.

The disk message queue consists of one or more similar extents of one fixed, unblocked data set. Each extent is on one or more disk drives of the same type, and each extent must contain the same number of continuous cylinders on a cylinder boundary. The records are assigned to the disks as discussed under Multiple Arm Support in the Method of Operation section of this publication.

The Convert routine obtains the absolute record number to be converted from the CPBADDR field of the CPB. After using values stored in the AVT to convert this value to MBBCCHHR, the converted value is placed in the CPBABSAD field of the CPB. All other fields, including the input field, are unchanged, except that the command chain bit in the second READ/WRITE CCW is turned off. This routine then branches back to the EXCP Driver routine.

The EXCP Driver routine uses "M" as the search index to find the appropriate IOB for this extent. It then searches the New queue of the appropriate IOB, using "CC" as the argument to find the proper place for the new CPB. The order is FIFO-per-cylinder, but cylinders are in order with the current arm position of the disk having the highest priority.

- Optimize the disk channel program

The EXCP Driver routine compares the new CPB with the previous CPB on the New queue. Five conditions are recognized, and accordingly, one of three possible types of CPBs is built into the new CPB.

Condition	CPB-Type
1. There is not a previous CBP	Medium
2. Previous CPB is for a different CC	Medium
3. Previous CPB is for the same CC, but a different HH	Large
4. Previous CPB is for the same track, but its record is not the one immediately preceding the record of the new CPB	Medium
5. Previous CPB is for the record immediately preceding the record of the new CPB (read only).	Small

A <u>large CPB</u> starts with seek, search, and TIC CCWs. It is used to change heads on the same cylinder. A <u>medium CPB</u> starts with search and TIC CCWs. It is used to locate non-sequential records on the same track as the previous CPB, and also to start the new channel program on a new cylinder. A <u>small CPB</u> has only READ/WRITE CCWs. It is used for finding sequential records on one track.

- EXCP

After all of the CPBs have been built onto the New queue, each IOB is checked to be sure that if any CPBs are available, the Retry queue has some. If EXCP is not busy, EXCP Driver shifts the Retry queue CPBs to IOBSTART, shifts one cylinder of CPBs from the New queue to the Retry queue, and then issues an EXCP command.

If EXCP is busy or if there are no CPBs on the New queue, EXCP Driver branches to the TCAM Dispatcher. Register 15 is set equal to X'08' to indicate that there is a chain pointed to by register 1.

<u>External Routine</u>: OS EXCP routine (SVC 0) - to start I/O on the channel.

<u>Tables/Work Areas</u>: IOB, DEB, CPB, AVT.

<u>Attributes</u>: Resident, reusable, refreshable, problem program mode.

<u>EXCP Driver for a Single CPB (Chart RF)</u>

<u>Module Name</u>: IGG019RF

<u>Entry Points</u>:

- IGG019RF - the EXCP Driver routine - called by the CPB Initialization routine to start disk I/O for one CPB.

- IEDQFP - the Convert routine - called by IGG019RF to convert the absolute record number to the MBBCCHHR address in the disk data set.

Function: The EXCP Driver for a Single CPB takes the CPB that was begun by CPB Initialization and finishes building the CCWs. This module adds Search-TIC CCWs where necessary, calls IEDQFP to translate the absolute disk address to an MBBCCHHR value, and then calls IOS with the EXCP macro to start disk I/O. When the channel program is completed, EXCP Driver returns to the TCAM Dispatcher.

External Routine: OS EXCP routine (SVC 0) - to start I/O on the channel.

Tables/Work Areas: IOB, DEB, CPB, AVT.

Attributes: Resident, reusable, refreshable, problem program mode.

Disk End Appendage (Chart R2)

Module Name: IGGC19R2

Entry Point: IGGC19R2 - called by IOS at the end of a disk operation.

Functions: This appendage receives control from IOS at the end of a disk I/O operation. It removes the CPBs from the IOB and makes them available for the CPB Cleanup routine by tposting the CPB Cleanup QCB to the disabled ready queue. CPBs are obtained from IOBSTART by using CPBNEXT as a chaining pointer. The CPBNEXT field of the last CPB is zero. The CPBs are passed to CPB Cleanup via one of two FIFO queues, AVTDKAPQ or AVTDKENQ. The CPB Cleanup QCB is tagged AVTCPBCB. If AVTBIT1 is set to X'80', AVTDKENQ is used, instead of AVTDKAPQ, to hold CPBs that are being returned to CPB Cleanup.

If any CPBs are available on the retry queue, they are returned to IOS with a request to retry the disk channel program. The locked bit in IOBXLOCK of the IOB is first checked for permission to try I/O on the CPBs on the retry queue. If locked, the retry queue is left untouched. The chain of CPBs just finished by IOS is in IOBSTART, and the MBBCCHHR of the first CPB from the retry queue is set to IOBSEEK if retry is to be done.

Disk End Appendage examines the CPB Cleanup QCB for a priority value of zero. This value indicates that the QCB is not on the ready queue. If the QCB is not on the ready queue, Disk End Appendage tposts the QCB to itself on the disabled ready queue in order to activate the CPB Cleanup routine (IEDQFQ in IEDQFA).

Disk End Appendage posts complete the ECB for the TCAM Dispatcher in order to reactivate the TCAM task. Disk End Appendage then returns to IOS via register 14 if there is no retry, or via register 14+8 if retry is to be attempted.

External Routine: OS Post routine (SVC 2) - to post complete the TCAM ECB.

Tables/Work Areas: OCB, IOB, CPB, AVT.

Attributes: Reentrant, refreshable, supervisor mode.

Disk End Appendage for a Single CPB (Chart RK)

Module Name: IGG019RK

Entry Point: IGG019RK - called by IOS at the end of a disk operation.

Function: This appendage receives control from IOS at the end of a disk I/O operation. It removes the single CPB from the ICB and makes it available for the CPB Cleanup routine by placing the CPB on the AVTDKAPQ FIFO queue. The Disk End Appendage then tposts the CPB Cleanup OCB to the disabled ready queue in order to activate CPB Cleanup to process the CPB. This Disk End Appendage complete the ECB for the TCAM Dispatcher to reactivate the TCAM task and then returns to IOS via register 14.

External Routine: OS Post routine (SVC 2) - to post complete the TCAM ECB, which allows the TCAM Dispatcher to be reactivated.

Tables/Work Areas: OCB, IOB, CPB, AVT.

Attributes: Reentrant, refreshable, supervisor mode, disabled.

Reusability-Copy Subtask (Chart RP)

Module Name: IGG019RP

Entry Point: IGG019RP is loaded by the Disk Message Queues Open routine and has only one defined entry point. There are, however, two logical entries:

- REUS - called by CPB Initialization (IEDQFA) to make the disk data set available for reuse.

- COPY - called by the TCAM Dispatcher to move an entire message from one message queue to another.

Functions: This module makes the disk message queues data set reusable by periodically performing two functions:

1. Moving the unused preassigned locations for the first units of messages into the current zone by placing a cancel bit in the old location of the units on disk, and

2. Copying unsent messages into the current disk zone and enqueuing them for transmission to an alternate destination.

If the user specifies QUEUES=DR on the TERMINAL macro and OPTCD=R on the DCB for the reusable disk message queues data set, the reusability function of the Reusability-Copy Subtask is applicable. The disk data set is logically divided into four zones, and when Destination Assignment (IEDQHM02) assigns a new disk record to a location that is equal to the current zone boundary (AVTLODPT), Destination Assignment sets a "first time" switch to request CPB Initialization to exit to REUS. CPB Initialization branches to REUS at a midzone point, and the Reusability-Copy subtask cancels all unused preassigned next-message locations in the preceding zone. This moves each next-message location into the current zone. The subtask also copies all unsent messages in the zone just before the preceding zone into the current zone, enqueuing the messages for transmission to an alternate destination.

The COPY part of the Reusability-Copy subtask is activated to copy a message from one queue to another whenever two receiving destinations are to receive the same message, but have their queue on different message queues. It is also activated when multiple-buffer-header messages have secondary header records in a zone that is different from the text records.

The REUS function can stop TCAM receiving operations by setting the "REUS disk is full" bit (X'40' in AVTBIT 3). When both the "REUS is running" and the "REUS first time" flags are on, this subtask allows TCAM to only send messages. Normal message traffic can be resumed when this subtask subsequently finds the "REUS first time" flag off and turns off the "REUS disk is full" bit.

After completing its processing, the Reusability-Copy subtask exits to CPB Initialization at IEDQFA02. Any generated CPBs are passed to EXCP Driver (IGG019RC) on a FIFO queue. If, however, this subtask receives a logical read error or disk, it issues an ABEND, S045, U0002.

The Reusability-Copy subtask uses a unit from the buffer unit pool as a work area that has the following DSECT format:

Offset				
+0 (0)	SAVEFEFO			
+4 (4)		DQCB		
+8 (8)	NBUNT	SQCB		
+12 (C)	DNFL	DESTQCB		
+16 (10)	Reserved	SOURCQCB		
+20 (14)	SRFL	BUFFER/IDBADPTR		
+24 (18)	XTRA			NTXT
+28 (1C)	NTXT (Cont.)		NEWXTRA	
+32 (20)	NEWXTRA (Cont.)	NEWHEADR		
+36 (24)	Reserved	BADHEAD		
+40 (28)	ALTDESX		BITS	

SAVEFEFO - Save area for the first 5 bytes of the data portion of the record

DQCB - Destination priority QCB

NBUNT - Decremented counter of additional units to be copied, originally set from PRFNBUNT

SQCB - Source priority QCB

DNFL - Destination message queue type - to be OR'ed with CPBFLAG:
X'20' reusable disk
X'10' nonreusable disk

X'00' main storage only

DESTOCB - Destination master QCB

SOURCQCB - Source master QCB

SRFL - Source message queue type - to be OR'ed with CPBFLAG: The bit definitions are the same as DNFL above

BUFFER - In COPY, the address of the original unit tposted to COPY.

IDBADPTR - In REUS, the absolute disk record number of the record whose FEFO pointer indicated the message to go to the alternate destination.

XTRA - The absolute address of the next additional unit to be fetched into main storage when moving a message

NTXT - The absolute address of the next buffer to be fetched into main storage when moving a message

NEWXTRA - The absolute address of the position in which the next additional unit is to be written.

NEWHEADR - The absolute address of the first unit for the alternate destination.

BADHEAD - The absolute address of the first unit of a message to be sent to an alternate destination.

ALTDESX - Index in the Termname Table of the alternate destination.

BITS - Flag bits for REUS:
X'80' - first message of this QCB was to go to an alternate destination.
X'40' - last message of this OCB has been checked by Reusability.

The Reusability-Copy subtask uses one CPB to transfer an entire message from one queue to the other. At the time the CPB is obtained from the CPB free pool, the unit work area is taken from the buffer unit pool to contain information about the status of the message being moved. This unit work area is attached to the CPB by a pointer at CPBAERB. When the copy operation is completed, or when this zone servicing by Reusability is finished, the subtask returns the unit work area to the buffer unit pool and the CPB to the CPB free pool.

<u>External Routines:</u>

- IEDQHG02 - Time Delay subtask - to remove the checkpoint element from the time delay queue in order to tpost the element to the Checkpoint CCB.

- IEDQHM02 - Destination Scheduler - to receive a buffer, put the address value in the prefix, and put the message into the main storage message queues data set if that is the destination.

- IEDQHM03 - Destination Scheduler - to determine whether the only message on the Priority QCB is being sent.

Tables/Work Areas: AVT, CPB, Termname Table, LCB, QCB, buffer prefix, Terminal Table.

Attributes: Reusable, nonrefreshable, enabled, problem program mode.

MESSAGE CONTROL PROGRAM TERMINATION ROUTINES

Resident Closedown Completion (Chart NA)

Module Name: IEDQNA

Entry Point:

- IEDQNA - called by the TCAM Dispatcher when the closedown completion element is on the ready queue.

- IEDQNA3 - called by the OS Termination routine when a TCAM attached task terminates.

Functions: At the IEDQNA entry point, this routine links to the Nonresident Closedown Completion module. After control is returned, this routine restores the user registers from AVTSAVE1 and returns to the user code that follows the READY macro expansion.

At the IEDQNA3 entry point, this routine determines whether the attached task terminated abnormally. If so, IEDQNA3 negates the TCB address in register 1 and then links to IEDQNA2. Otherwise, IEDQNA3 returns to OS.

External Routine: IEDQNA2 - Nonresident Closedown Completion routine - to close down the MCP and the attached tasks.

Tables/Work Areas: AVT, TCB.

Attributes: Reentrant, resident.

Nonresident Closedown Completion Routine (Chart NA2)

Module Name: IEDQNA2

Entry Point: IEDQNA2 - called by the Resident Closedown Completion routine (IEDQNA) to close down the MCP and the attached tasks.

Functions: When register 1 is positive, this routine waits for the completion of all disk activity in the MCP. If there is disk activity in the MCP, Closedown Completion exits to the TCAM Dispatcher to tpost the closedown completion element back to the ready queue.

If there is no disk activity in the TCAM system, the Closedown Completion routine sets a closedown completion bit in the environment checkpoint request element and posts the ECBs for any attached tasks. This routine waits for the termination of the attached tasks and then detaches the tasks. The Closedown Completion routine checks for the presence of Checkpoint, On-Line Test, and FE Common Write before issuing a DETATCH; however, Operator Control is unconditionally detached.

After the above functions have been completed, this routine returns to the Resident Closedown Completion routine (IEDQNA).

When register 1 is negative, a TCAM attached task has terminated abnormally. This module compares the TCB address passed from OS with the TCB address in the AVT to determine which task terminated. If the task was TOTE, this module issues an ABEND with a completion code of 42 to abend the MCP. If the task was operator control, IEDQNA2 issues a WTO message to the system console and then returns to IEDQNA3. If the task was checkpoint, this module issues a WTO message to the system console, clears AVTCKGET to zero so that no more checkpoints will be attempted, and returns to IEDQNA3. If the task was FE Common Write, IEDQNA2 issues a WTO message to the system console, clears AVTCWFL1 to zero to indicate an inactive status, and returns to IEDQNA3.

External Routines: None.

Tables/Work Areas: AVT, IOB, DCB, DEB.

Attributes: Transient, reentrant.

MCP Closedown Processing Routine (Chart C0)

Module Name: IEDQCO

Entry Point - IEDQCO - called when an MCPCLOSE or CLOSEMC macro is issued in an application program or when the HALT operator control command is issued from a terminal or the system console.

Functions: This routine processes an MCPCLOSE or CLOSEMC macro from an application program or a HALT command from a terminal or the system console.

The MCP Closedown Processing routine first scans the input command for syntax errors. If there are errors, this routine returns to IEDQCA with X'04' in register 15. Otherwise, this routine scans all the LCBs to set the X'01' bit in LCBQCBA+2 for each stopped line. The MCP Closedown Processing routine then sets the proper closedown in progress switches in AVTBIT1.

If the Reusability-Copy subtask is active, the MCP Closedown Processing routine issues a WAIT until the disk I/O is complete. After this if any line is sending data, the MCP Closedown Processing routine issues a WAIT on the Operator Control ECB. This allows the MCP to gain control to process elements on the ready queue.

When the ECB is posted complete, the MCP Closedown Processing routine regains control to continue checking for line activity. This routine checks the lines and issues WAIT macros until all sending activity has stopped. At this point, the MCP Closedown Processing routine returns to Operator Control so that the Stop Line routine can be activated to issue a HALT I/O on each line.

After the Stop Line routine has performed its functions, the MCP Closedown Processing routine is reloaded to examine the PCB use count fields. If there is an open DCB for a PCB (use count not equal to zero), this routine issues a WTO error message that contains the job name of the application program to the system console. This routine then issues a WAIT on the Operator Control ECB to allow time for a CLOSE to be executed in the application program. When the CLOSE is executed, the MCP Closedown Processing routine regains control to continue examining the use counts until the routine determines that all application program DCBs are closed.

When all the DCBs are closed, the MCP Closedown Processing routine rechecks the LCBs. If any LCB is marked sending, receiving, or free, this routine issues a WAIT to allow time for the LCB to be marked stopped. When all the LCBs are marked stopped, this routine tposts each LCB to itself and issues a WAIT to flush the queues. Then this routine issues a WAIT on the Operator Control ECB until all the CPBs are in the CPB free pool.

After all of the above functions are completed, the MCP Closedown Processing routine tposts an element to the ready queue to request closedown. If the checkpoint DCB is open, the routine tposts the environment request element; otherwise, the routine tposts the closedown completion request element. The MCP Closedown Completion routine returns to the address in register 14 - the Operator Control task. Note that the element is tposted to the ready queue via the AOCTL SVC 102 routine.

External Routines:

- IGC102 - AOCTL SVC 102 routine - to tpost an element to the ready queue.

- OS Wait routine (SVC 1) - to allow time for certain MCP functions to complete.

Tables/Work Areas: CVT, AVT, TCB, DEB, DCB, LCB.

Attributes: Reentrant, transient.

Line Group Close Routines (Charts I4 and I5)

Module Names: IGG02035, IGG02036

Entry Points:

- IGG02035 - activated by an XCTL from an I/C supported routine to close the line group DCBs. The routine may also be reentered by a loop from itself if there are multiple DCBs to be processed. (Chart 24)

- IGG02036 - activated by an XCTL from the Line Group Close routine (IGG02035) after a CLOSE line DCB is issued in a TCAM MCP. It may also be reentered by a loop frcm itself if there are multiple DCBs to be processed. (Chart L5)

Functions: The functions of each routine are defined according to entry point.

- IGG02035

Load 1 of the Line Group Close routine determines whether the closedown condition is the result of an abend in the MCP. If the MCP has not abended, the routine issues an EXCP macro on the line to perform error recording via the ERP routines. On the other hand, if the MCP has abended, the routine determines whether there are any application programs active in the system. If there are some active, the Line Group Clcse routine determines whetner any of them have been scheduled to be abended. If ncne have been scheduled, the routine schedules them to be abended and branches tc the Abnormal Termination routine, which abends each application program with a completion code of 046. After processing each application program in the system, the routine issues an EXCP macro on the first line in the line group in the MCP and branches to the ERP routines to perform OBR-SDR error recording.

On return from the ERP routines, the Line Group Clcse routine checks all the DCBs for the lines in the MCP and then issues an XCTL macro to give ccntrol to the executor identified by the next non-zero entry in the system Where-to-go Table.

- IGG02036

The purpose of this routine is to close a line group DCB in a TCAM MCP. Load 2 of the Line Group Close routine purges all I/O on the lines associated with this DCB, disables the lines unless they are connected to a Type III adapter, frees the associated ICBs, and clears the associated line entries in the Cross Reference Table (if present).

Load 2 of the Line Group Close routine also examines every DEB in the TCB DEB chain to determine whether there is an additional line group or message queues DCB to be closed. If there is not, the Line Group Close 2 routine places zeros in the AVT pointer in the Dispatcher prefix and issues a FREEMAIN for the Cross Refrence Table, if one is present.

IGG02036 issues an XCTL to the module indicated by the next non-zero entry in the system Where-to-Go Table. This module can be any one of the system modules IGG0200B through IGG0200G.

External Routine: OS Freemain routine (SVC 5) - to free the Cross Reference Table if all DCBs have been processed.

Tables/Work Areas: Where-to-Go Table, DCB, DEB, TCB, Cross Reference Table.

Attributes: Transient, enabled, reentrant.

Checkpoint Close Routine (Chart L6)

Module Name: IGG02041

Entry Point: IGG02041 - called when a CLOSE checkpoint DCB macro is issued in the MCP.

Functions: This routine closes the checkpoint DCB in an MCP. If this is a normal closedown, the Checkpoint Close routine sets an indication in the checkpoint control record and rewrites the record on disk. It then frees the checkpoint work area by a FREEMAIN macro instruction. The address of the checkpoint work area is in AVTCKGET, unless some error causes the checkpoint function to be eliminated from the system. In this case, the address is in AVTCKELE. If both fields are equal to zero, there was not enough main storage for the GETMAIN, so the FREEMAIN and delete functions are bypassed.

The only error recognized by this routine is a disk error while writing the control record. If this occurs, the Checkpoint Close routine uses WTO to issue an error message and exits, as it would if no error had occurred, to the next entry in the Where-to-Go Table.

External Routine: IECPCNVT - an OS system routine that converts the relative track address to an actual disk address.

Tables/Work Areas: CVT, AVT, checkpoint work area.

Attributes: Reentrant, transient, supervisor mode.

Message Queues Close Routine (Chart L1)

Module Name: IGGC2030

Entry Point: IGG02030 - activated by an XCTL from a system I/O support module when a CLOSE message queues or checkpoint DCB is issued in a TCAM MCP. It may also be reentered by a loop from itself if there are multiple messages queues DCBs to be processed.

Functions: This routine closes a message queues DCB in a TCAM MCP. The Message Queues Close routine removes the DEB for this DCB from the DEB chain in the TCB and frees the IOBs associated with this DCB.

The Message Queues Close routine also examines every DEB in the TCB DEB chain to determine whether there is an additional DCB to close. If there is not, the Message Queues Close routine issues a FREEMAIN command for all areas that were obtained at INTRO time and places zeros in the AVT pointer in the prefix of the Dispatcher.

If the DCBOPTCD field of the DCB indicates a Checkpoint DCB, this close routine puts the identification (41) of the Checkpoint Close routine in the next entry in the system Where-to-Go Table.

The Message Queues Close routine then issues an XCTL to the module indicated by the next non-zero entry in the system Where-to-Go Table - this can be any module with a name from the value IGG0190B to IGG0190G or IGG01941, as determined by the system close functions.

External Routine: OS Freemain routine (SVC 5) - to free main storage that is associated with the DCBs.

Tables/Work Areas: Where-to-Go Table, DCB, DEB, TCB, Cross Reference Table.

Attributes: Transient, enabled, reentrant.

APPLICATION PROGRAM INITIALIZATION AND TERMINATION ROUTINES

GET/PUT and READ/WRITE Open Executor (Charts I7, L8)

Module Names:

- IGG01946 - Load 1. (Chart L7)
- IGG01947 - Load 2. (Chart L8)

Entry Points:

- IGG01946 - called by the OS system OPEN routine when an OPEN GET/PUT or READ/WRITE DCB is specified in an application program.
- IGG01947 - called by an XCTL from IGG01946.

Functions: This module activates a data transfer communication link between an application program and the MCP. The Open Executor is used to open input (GET or READ) and output (PUT or WRITE) DCBs in the application program. It executes with a supervisor storage protection key and, for the most part, disabled to interruptions.

When the Open Executor (Load 1) is activated, it first tests the AVT address pointer in the CVT to determine whether there is an active MCP in the system. If there is not an active MCP, the Open Executor does not open the DCB, sets an unsuccessful open flag in the DCB, and exits to IGG01933.

If there is an active MCP in the system, the Open Executor gets the "queuename" (that was coded on the DD card) from the Job File Control Block (JFCB) in the DCB work area. The User Interface routine is then invoked to activate the Binary Search routine (IEDQA1), which uses the "queuename" parameter to search the Termname Table for the corresponding process entry. When the matching entry in the Termname

Table is found, ccntrol returns to the Open Executor, which, in turn, issues a GETMAIN command to obtain main storage fcr an access method work area and for a Data Extent Block (DEB). If the "queuename" is invalid, the Open Executcr exits to IGG01933 for error processing.

The Open Executor tpcsts a special element that ccntains a pointer to the process entry in the Termname Table to the Open/Close subtask in the MCP. (The AOCTL SVC 102 routine is used to tpost the element to the ready queue in the MCP.) The Open Executor then issues a WAIT to put the application program in the wait state. This allows the MCP to gain control to process the special element just tpcsted from the application prcgram.

When the applicaticn program ECB is posted complete by the MCP, the WAIT condition is satisfied and the Open Executor regains ccntrol. It inspects the status flag in the process entry to determine whether the MCP successfully allocated main stcrage in the MCP for this application program. If so, the Open Executor initializes the access method work area by linking it tc the DCB and the process entry work area in the MCP. The DEB is also initialized and enqueued on the TCB DEB chain. If the MCP did not allocate space for this application program (either the process entry was already in use or a GETMAIN was unsuccessful) the DCB is nct opened, the DEB is not enqueued, an unsuccessful open flag is set in the DCB, and the Open Executor exits to IGG01933 fcr errcr processing.

The Open Executor then loads the appropriate access methcd module for the DCB being opened. For an input DCB, the appropriate GET/READ routine is loaded and linked to the DCB; for an output DCB, the appropriate PUT/WRITE routine is lcaded and linked to the DCB. The Check routine is loaded if this is a read/write condition, and if the user has specified a POINT macro, the Point routine is also lcaded.

After the above functions are performed, Load 1 of the Open Executor saves the information needed by Load 2 (IGG01947) in the OPEN/CLOSE work area. The Open Executor (Load 1) then transfers control (XCTL) to next entry in the Where-to-Go Table. The next entry, in this case, is the IGG01947 entry pcint of Load 2 of the Open Executor.

Load 2 of the Open Executor first determines whether it is continuing the open of an input DCB. If it is an input DCB, the Executor inspects the Destination QCB for this application program in the MCP. If there is not a complete message on this QCB, the Get Scheduler STCB in the MCP is moved from the Read-ahead QCB to the application program Destination QCB. If there is a message, an ERB for the message buffers is initialized and tpcsted to the Disk I/O QCB in the MCP.

If locate mode is specified, the Open Executor issues a GETMAIN for a work area and stores its address in the DEB. If the GETMAIN is unsuccessful, the Open Executor exits to IGG01933 for error processing.

Upon successful completion of the above functions, the Open Executor sets a successful open flag in the DCB and returns to the system OPEN routine (the next entry in the Where-to-Go Table).

External Routines:

- IEDQUI - User Interface routine - to activate the Binary Search routine (IEDQA1), which locates the "queuename" entry in the Termname Table.

- IGC102 - AQCTL SVC 102 routine - to tpost a special element to the MCP ready queue.

- IEDQNB05 - Application Program/Operator Control Interface routine - to take an MCP checkpoint.

Tables/Work Areas: CVT, AVT, TCB, Termname Table, DCB being opened, JFCB, access method work area, DEB process entry, process entry work area, OPEN/CLOSE work area.

Attributes: Transient, reentrant, refreshable, enabled, supervisor mode.

GET/PUT and READ/WRITE Close Executor (Charts L9 and L10)

Module Names: IGG02046, IGG02047

Entry Points:

- IGG02046 - called by the system Close routine when a CLOSE GET/PUT or READ/WRITE DCB is issued in an application program. (Chart L9)

- IGG02047 - activated by an XCTL from IGG02046 to complete application program closedown.

Functions: These routines deactivate a data transfer communication link between an application program and the MCP. Both loads of the Close Executor are used to close input (GET/READ) and output (PUT/WRITE) DCBs in the application program. They execute with a supervisor storage protection key, and, for the most part, disabled to interruptions. The functions of each module are discussed according to entry point.

- IGG02046

For an input DCB, if an ERB is tposted to the Read-ahead QCB, Load 1 of the Close Executor tposts a special element that contains the address of the DCB process entry in the Termname Table to the Open/Close subtask in the MCP. If the ERB is not busy, the Close Executor tests the Read-ahead QCB for full buffers to be tposted to the Buffer Return QCB. If the Get Scheduler is in mid-cycle, the Close Executor does not tpost the special element. The Get Scheduler tposts the element when it finishes its cycle and recognizes that a CLOSE has been issued for the DCB associated with the process entry. If the Get Scheduler is waiting on the application program Destination QCB, its STCB is unchained. (This tpost is accomplished via the AQCTL

SVC 102 routine.) A WAIT macro is then issued to allow the MCP to gain control to process the special element just tposted from the application program.

For an output DCB, the Close Executor tests to determine whether a buffer was saved from the last PUT or WRITE operation. If it was, the buffer is flagged as end-of-message and tposted to MH. The Close Executor then tposts a special element to the Open/Close subtask in the MCP and issues a WAIT command.

When the application program ECB is posted complete, the Close Executor regains control to free main storage occupied by the access method work area, the locate mode work area (if any), and the DEB, to issue a DELETE macro for any loaded routines, to restore the DCB to its pre-open status, and to set the close flag in the DCB.

If the MCP closedown bit in the AVT is on, the GET/PUT and READ/WRITE Close Executor posts complete the Operator Control ECB in the AVT. This allows closedown processing to complete.

After completing its functions, IGG02046 transfers control to IGG02047 to complete the application program closedown procedure.

- IGG02047

Load 2 of the GET/PUT and READ/WRITE Close Executor scans all the TCAM LCBs to determine whether any lines are locked to the application program DCB that is being closed. If an LCB is locked to the DCB in question, IGG02047 removes the associated line from lock mode and tposts (via IGC102) the LCB to itself to free the line.

After all DCBs have been closed for this invocation, Load 2 of the Close Executor transfers control to the system OPEN routine (the next entry in the Where-to-Go Table).

External Routines:

- IGC102 - AQCTL SVC 102 routine - to tpost a special element or an LCB to the MCP ready queue.

- IEDQNB05 - Application Program/Checkpoint Interface routine - to take an MCP checkpoint.

Tables/Work Areas: CVT, AVT, DCB to be closed, TCB, QCB, DEB, access method work area, process entry, process entry work area, OPEN/CLOSE work area, LCB.

Attributes: Transient, reentrant, refreshable, enabled, supervisor mode.

Open/Close Subtask (Chart EU)

Module Name: IEDQEU

Entry Point: IEDQEU - gains control when a special element (containing the address of a process entry in the Termname Table) tposted by the Open or Close Executor in an application program gets to the top of the ready queue in the MCP.

Functions: This routine allocates main storage in the MCP for an application program. The allocated main storage is to contain a process entry work area, an LCB, and one or more SCBs.

The Open/Close subtask is an MCP routine. It gains control when a special element containing the address of an application program process entry in the Termname Table is at the top of the MCP ready queue. This special element is placed on the ready queue by the Open or Close Executor in an application program.

If the Open/Close subtask is activated by the Open Executor, it first allocates main storage for a process entry work area/SCB and links it to the process entry. It then increments the use count in the PCB, loads the appropriate scheduler, links the scheduler to its Destination QCB, sets a good-open flag in the process entry, posts the application program ECB complete, and returns to the TCAM Dispatcher. If any GETMAIN or LOAD fails, a return code is stored in the process entry and the open-failed flag is set in the process entry.

If the Open/Close subtask is activated by the Close Executor, it first frees the process entry work area/SCB and decrements the use count in the PCB. The subtask then delinks and issues a DELETE macro for the appropriate scheduler, deactivates the application program Destination QCB, and turns off the open flag in the process entry. If the use count in the PCB is now equal to zero, the LCB is also freed. The Open/Close subtask relinquishes control by posting the application program ECB complete and returning to the TCAM Dispatcher.

If a GET or READ DCB is being closed, the Open/Close subtask determines whether the ERB is in use and ensures that all buffers assigned to the corresponding Destination QCB are in the buffer unit pool. If the ERB is in use, this subtask returns to the TCAM Dispatcher to wait for ERB cleanup by the Get Scheduler.

External Routines:

- IGC102 - AQCTL SVC 102 routine - to post the application program ECB.

- OS Getmain routine (SVC 4) - to get main storage for an LCB and the process entry work area.

- OS Freemain routine (SVC 5) - to free main storage for an LCB and the process entry work area.

Tables/Work Areas: Process entry in the Termname Table, QCB, PCB, process entry work area, SCB.

Attributes: Reentrant, refreshable, enabled, resident, problem program mode.

- IGC102 - AQCT2 SVC 102 routine - to pcst the application program ECB.

APPLICATION PROGRAM I/O ROUTINES

Get Scheduler (Chart EW)

Module Name: IEDQEW

Entry Point: IEDQEW - activated by the TCAM Dispatcher when a special element from a GET/READ routine in an application program is on the MCP ready queue.

Function: This routine performs a read-ahead activity from the disk message queues data set in anticipation of a GET command from an application prcgram. The Get Scheduler also reacts to retrieve requests from an application program.

The STCB for the Get Scheduler waits on either the Destination QCB or on the Read-ahead QCB for a particular application program. The Get Scheduler STCB is on the Destination QCB until a complete message enters the system. It then waits cn the STCB chain of the Read-ahead QCB.

As long as there are buffers available and more than two messages have not been read from the message queue, the Get Scheduler tposts an ERB to the Disk I/O QCB to have a buffer filled with data and placed on the Destination QCB. The Disk I/C QCB activates the CPB Initialization routine, which satisfies the ERB and passes the full buffers on the ERB to the Get Scheduler. The Get Scheduler places these buffers in the element chain of the Read-Ahead QCB to be read by the application program GET/READ routine.

The Get Scheduler insures that MH processes only one message at a time. If the message handler for a particular application program is already processing a message and a new message arrives, the Get Scheduler enqueues the buffers of the new message on the Pre-MH queue. When MH completes the prior message, the Get Scheduler tposts an ERB for buffers fcr the next message. Also, the prior message is not marked serviced until the next message has been processed by the application prcgram.

If a message is a lock inquirey, the Get Scheduler turns cn bit 3 (X'20') in the byte at LCBINSRC+2 in the ICB for the source terminal. This indicates that a response is due to the source terminal. The Get Scheduler also increments by cne the lock respcnse count (LCBINCAM) in the LCB for the application program pointed to by the PCB.

When the applicaticn program GET/READ routine reads and subsequently empties the full buffers, it tposts a special element to the Get Scheduler in the MCP to indicate how many buffers can be returned to the buffer unit pccl. The Get Scheduler returns the buffers and requests another disk read.

This operation continues unless the Get Scheduler receives a special retrieve element. At this point, it waits for any requested I/O activity to complete.

If the specified destination for the special retrieve element is a main-storage-only queue, the message cannot be retrieved and an error return code (X'40') is passed to the application program in the process entry work area.

If the specified destination for the special retrieve element is not a main-storage-only queue, the Get Scheduler alters the LCB and the SCB to begin retrieving messages from the indicated destination message queue. The recall header bit (LCBRCLIN) in the LCB is turned on, so the Disk End Appendage tposts full buffers directly to the Read-ahead QCB. The Get Scheduler examines each header for the requested sequence number. If the requested sequence number cannot be found, an error code (X'40') is returned to the application program in the process entry work area. If the requested header is found, the Get Scheduler requests that the message be read from disk. As each buffer of the message being retrieved is tposted to the Read-ahead QCB, the Get Scheduler tests the GET DCB to see if the application program is in retrieve mode. If the application program leaves retrieve mode before the entire message is read, the Get Scheduler returns any unfilled buffers to the buffer unit pool and then resumes normal operation.

When the Get Scheduler determines that the DCB for the current process entry is being closed, the scheduler sets up any buffers on the Read-ahead QCB to be returned to the buffer unit pool. In this situation, the scheduler also tposts a special closedown element to the Open/Close subtask (IEDQEU).

External Routine: IGC102 - AQCTL SVC 102 routine - to post the application program ECB complete in order to activate the waiting application program.

Tables/Work Areas: DCB for GET or READ in the application program, Read-ahead QCB, Termname Table, AVT, process entry work area, application program ECB.

Attributes: Reentrant, refreshable, enabled, transient, problem program mode.

GET/READ Routine (Chart RG)

Module Name: IGGC19RG

Entry Point: IGG019RG - called when a GET or READ macro is issued in a SAM compatible TCAM application program.

Functions: This routine reads data from full buffers on the element chain of the Read-ahead QCB in the MCP into an application program work area. It also includes support for message retrieval and the checkpoint user exit (CKPTADD).

The GET/READ routine reads data from the buffers in the MCP until either the application program work area is filled, an entire work unit is moved, or the end of a message is reached. For QSAM, if the Read-ahead QCB is empty and SETEOF is not specified, the GET/READ routine waits for more data. For BSAM, the routine stores a completion code of X'01' or X'02' in the DECB. The GET/READ routine builds a special element that contains the number of buffers emptied and uses the AQCTL SVC 102 routine to tpost this element to the Get Scheduler STCB in the MCP.

The GET/READ routine branches to the user EODAD address, if specified, on the subsequent GET or READ/CHECK request following recognition of a buffer that contains an end-of-file indicator in its prefix (SETEOF condition). For QSAM, a return code of X'04' is placed in register 15 if the SETEOF condition exists and no EODAD is specified. For BSAM, the SETEOF condition causes a code of X'70' to be placed in the DECB.

If the SETEOF condition is not present, the GET/READ routine does not pass control to the next user-coded instruction in the application program until the user request is completely satisfied. (The routine uses a WAIT command to maintain control.) If the READ request is used, the WAIT is delayed until the CHECK macro is issued. After successful completion of a GET operation, X'00' is placed in register 15. For a READ operation, a X'7F' completion code is placed in the DECB.

If OPTCD=C or W is specified on the DCB macro, the GET/READ routine places the name of the source terminal and/or a work area contents description in the application program work area. If the source is not specified in the buffer prefix (e.g., from a dial line), blanks (X'40') are placed in the terminal name field in the work area.

If a checkpoint exit is specified on the DCB macro, the GET/READ routine takes this exit each time the first buffer of a message is processed and each time a checkpoint has been taken in the MCP since the last time the first buffer of a message was examined.

If the SYNAD user exit is specified, it is taken on work area overflow if OPTCD=C is not specified. If SYNAD is not specified, the user receives a return code in register 15 for a GET request or in the DECB for a READ request. For QSAM a return code of X'08' indicates a work area overflow. For BSAM, the code for work area overflow is X'52'.

External Routines:

- OS Wait routine (SVC 1) - to wait for data to arrive on the element chain of the Read-ahead QCB in the MCP.

- IGC102 - AQCTL SVC 102 routine - to tpost a special element to the Get Scheduler STCB in the MCP.

Tables/Work Areas: CVT, Get/Read DCB, application program work area, AVT, Read-ahead QCB, access method work area, DECB, DEB, Termname Table, process entry work area, PCB.

Attributes: Reentrant, refreshable, problem program mode, enabled, transient.

Check Routine (Chart RL)

Module Name: IGGC19RL

Entry Point: IGG019RL - called when a CHECK macro is issued in an application program in conjunction with a READ or WRITE macro.

Functions: This routine tests for completion of the read or write request that is related to this check request. It also tests for errors that may have occurred during the execution of the associated READ or WRITE macro.

If the event control block (ECB) in the data event control block (DECB) is posted complete and the DECB completion code is X'7F', the Check routine returns control to the user at the next sequential instruction following the CHECK macro expansion.

If data has appeared on the Read-ahead QCB since the Read routine examined it (DECB completion code equal to X'40), the Check routine invokes the GET/READ routine by a BALR instruction. Upon return from the GET/READ routine, the DECB completion code is rechecked.

At end-of-file (SETEOF), the Check routine takes the EODAD exit, if one is specified. Otherwise, a return code of X'04' is returned to the next sequential instruction after the CHECK macro-expansion.

If a READ or WRITE error is detected, the Check routine takes the SYNAD exit, if specified. Otherwise, an error return code is passed to the next program instruction. A return code of X'08' after a READ operation indicates that a work area overflow occurred; X'08' after a WRITE operation indicates a sequence error; X'0C' indicates an invalid destination.

If the ECB is not posted complete and no error indication is detected in the DECB, the Check routine issues a WAIT for the completion of the event under consideration. When the wait is satisfied (a message arrives on the Read-ahead QCB), the Check routine invokes the appropriate TCAM SAM routine.

External Routine: IGG019RG - GET/READ routine - to read buffers from the Read-ahead QCB.

Tables/Work Areas: DECB, DCB, DEB, access method work area.

Attributes: Reentrant, refreshable, problem program mode, enabled, transient.

Get Scheduler FIFO Routine (Chart EZ)

Module Name: IEDQEZ

Entry Point: IEDQEZ - activated by the TCAM Dispatcher when a POINT macro followed by a GET macro is issued in an application program and the Get Scheduler STCB is on the STCB chain of the Destination QCB, rather than on the STCB chain of the Read-ahead QCB.

Functions: This routine recognizes the retrieve element and tposts it to the application program Destination QCB to indicate to the Get Scheduler the need to process the retrieve element.

External Routines: None.

Tables/Work Areas: Read-ahead QCB, Destination QCB.

Attributes: Reentrant, refreshable, enabled, transient, problem program mode.

PUT/WRITE Routine (Chart RI)

Module Name: IGG019RI

Entry Point: IGG019RI - called when a PUT or WRITE macro is issued in a SAM Compatible TCAM application program.

Functions: This routine prepares the data in the application program PUT/WRITE work area for transfer into buffers in the MCP.

The PUT/WRITE routine initializes certain fields of the access method work area with data from the application program DCB, DECB, and work area prefix. If locate mode is used, the address of the PUT/WRITE work area is in the DEB; otherwise, it is supplied as an operand of the PUT or WRITE macro.

The PUT/WRITE routine takes the synchronous checkpoint exit if a checkpoint has been taken in the MCP since the last PUT or WRITE and the DCB has an EXLST entry offset of X'0F'.

If destination terminal name is specified as the work unit, the routine converts the name to a Termname Table offset to be stored in the access method work area.

The PUT/WRITE routine tposts (via the AQCTL SVC 102 routine) a special element to the Put Scheduler in the MCP. It then issues a WAIT command to pass control to the MCP. When the Put Scheduler gains control, the presence of this special element on the ready queue causes the Put Scheduler to empty the application program PUT/WRITE work area. The application program does not regain control until the entire work area has been transferred into MCP buffers.

For a PUT operation in locate mode, the PUT/WRITE work area address is returned to the user in register 1. Therefore, data is not moved until the second and subsequent operations.

If the application program is eligible to be swapped, the AQCTL SVC 102 routine causes it to be flagged not eligible to be swapped until the PUT/WRITE operation is completed. It performs the same type of function for application programs that can be rolled out.

At the completion of a PUT (QSAM) operation, register 15 contains a return code, normally zero. It is X'04' if either the terminal name or the Termname Table offset is invalid. If message segments or work units are not in proper sequence, the return code is X'08'.

At the completion of a successful WRITE (BSAM) operation, byte zero of the DECB contains a completion code of X'7F'. The DECB completion code for an invalid terminal name is X'44' and for a work unit sequence error is X'48'.

External Routines:

- IEDQUI - User Interface routine - to activate the Binary Search routine (IEDQA1) to scan the Termname Table for the specified terminal name.

- OS Wait routine (SVC 1) - to allow the Put Scheduler in the MCP to empty the application program PUT/WRITE work area.

- IGC102 - AQCTL SVC 102 routine - to tpost a special element to the Put Scheduler STCB in the MCP.

Tables/Work Areas: CVT, AVT, PUT/WRITE DCB, PUT/WRITE work area, DECB, Read-ahead QCB, Termname Table, access method work area, DEB, process entry work area, PCB.

Attributes: Reentrant, refreshable, problem program mode, enabled, transient.

Put Scheduler (Chart EC)

Module Name: IEDQEC

Entry Point: IEDQEC - activated by the TCAM Dispatcher when a special element from an application program PUT/WRITE routine is on the MCP ready queue.

Functions: This routine moves data from an application program into MCP buffers. The Put Scheduler requests buffers from the buffer unit pool, fills them with data from the application program work area, and tposts the full buffers to the appropriate MH in the MCP.

When the application program work area is empty and the last buffer tposted to MH is not EOM, the Put Scheduler posts the ECB for the application program complete, if it is waiting. It also flags the application program eligible for if it has been flagged not eligible to be swapped (TSO only), and eligible for rollout if flagged not eligible for rollout. If the last buffer is EOM, the Put Scheduler posts the application program ECB after Buffer Disposition tposts the application program ECB back to IEDQEC.

When the message destination is specified and the lock response count in the application program LCB is not zero, the Put Scheduler determines whether the destination terminal is locked to the application program that initiated the post operation. If so, the Put Scheduler moves the Send Scheduler STCB to the STCB chain of the LCB for the terminal and then decrements the lock response count.

If the PUT or WRITE DCB indicates that the user is using record format without a leading TCAM work area contents description byte, the Put Scheduler saves the last full buffer in the element chain of its OCB. At CLOSE time, which implies end-of-message (EOM), this latest full buffer is tposted to MH and flagged EOM.

External Routines:

- IGC102 - AQCTI SVC 102 routine - to post the application program ECB complete after the data in the application program work area has been transferred into MCP buffers.

- IGG019RB or IGG019RO - TCAM Dispatcher - to tpost the ERB or full buffers

Tables/Work Areas: AVT, LCB, PUT or WRITE DCB in the application program, DEB, PCB, access method work area, process entry, process entry work area, QCB, SCB, Termname Table, application program work area.

Attributes: Transient, reentrant, refreshable, enabled, problem program mode.

APPLICATION PROGRAM MESSAGE RETRIEVAL-POINT ROUTINE

Point Routine (Chart RM)

Module Name: IGG019RM

Entry Point: IGG019RM - activated by the POINT macro expansion in a TCAM application program.

Functions: This routine builds a message retrieval control block from input information specified by the user. This control block is used by the GET/READ routine to retrieve the specified message.

When the Point routine gains control to initiate message retrieval, it stores the Terminal Table entry address, the message sequence number, and the message type (input or output) in the retrieve control block. The routine also sets the retrieve flag in the access method work area to indicate that message retrieval is in progress.

The Point routine issues a return code in register 15. For successful completion, the return code is X'00'. A return code of X'08' indicates that an invalid terminal name was specified, a return code of X'04' indicates that an invalid sequence number was specified, and a return code of X'0C' indicates an invalid queue type.

External Routine: IEDQUI - User Interface routine - to activate the Binary Search routine (IEDQA1) to scan the Termname Table for the specified terminal name.

Tables/Work Areas: CVT, AVT, DCB, DEB, access method work area, QCB, Termname Table, Terminal Table entry.

Attributes: Reentrant, refreshable, problem program mode, enabled, transient.

APPLICATION PROGRAM COMPATIBLE QTAM ROUTINES

GET Compatible Routine (Chart RH)

Module Name: IGG019RH

Entry Point: IGG019RH - called when a GET macro is issued in a compatible QTAM application program.

Functions: This routine moves data from full buffers on the element chain of the Read-ahead QCB in the MCP into an application program work area.

The GET Compatible routine reads data from the buffers in the MCP until either an entire work unit is moved, the application program work area is full, or the element chain of the Read-ahead QCB is empty. If the queue is empty and an entire work unit has not been read, the GET Compatible routine takes the EODAD exit, if specified. If no EODAD address is specified and this condition exists, a WAIT macro is issued to allow time for more data to be placed on the queue.

When the GET cperation completes, or when the queue is empty, the GET Compatible routine builds a special buffer return element and tposts it, via AQCTL SVC 102, to the Get Scheduler STCB in the MCP. This returns the empty buffer units to the MCP.

If a work area overflow occurs, the GET Compatible routine takes the SYNAD exit, if specified, and does not read the segment that would cause the overflow. If no SYNAD exit is specified, the routine places a X'04' return code in register 15.

If the work unit fcr the application prcgram is a message, the GET Compatible routine checks for an ECM buffer. If the wcrk unit is a segment, a single logical buffer is used. If the wcrk unit is a record, the routine scans for EOB, NL, CP, IF, or ECM.

External Routines:

- OS Wait routine (SVC 1) - to wait for data to arrive on the element chain of the Read-ahead QCB in the MCP.
- IGC102 - AQCTL SVC 102 routine - to tpost a special element to the Get Scheduler STCB in the MCP.

Tables/Work Areas: CVT, AVT, DCB, DEB, access method work area, Termname Table, buffer prefix, PCB, process entry work area.

Attributes: Reentrant, refreshable, problem prcgram mode, enabled, transient.

PUT Compatible Routine (Chart RJ)

Module Name: IGG019RJ

Entry Point: IGG019RJ - called when a PUT macro is issued in a compatible QTAM application program.

Functions: This routine prepares data in the application program PUT/WRITE work area for transfer into buffers in the MCP.

The PUT Ccmpatible routine initializes certain fields of the access method wcrk area with data from the application program work area prefix and from the DCB. The User Interface routine (IEDQUI) is called to activate the Binary Search routine tc ccnvert the destination terminal name to a Termname Table offset. The PUT Compatible routine then converts the ccmpatible QTAM wcrk area contents descriptcr byte to its TCAM equivalent and verifies the segment/record sequence.

The PUT Compatible routine tposts (via AQCTL SVC 102) a special element to the Put Scheduler in the MCP. The routine then issues a WAIT to pass ccntrol to the MCP. When the Put Scheduler gains control, the presence of this special element on the ready queue causes the Put Scheduler to empty the application prcgram PUT/WRITE work area. The ECB for the application prcgram is posted complete when the entire work area has been transferred intc buffers in the MCP.

At the completion of a PUT compatible operation, register 15 contains a return code - X'00' if the operation was successful. A return code of X'40' indicates an invalid record or segment sequence.

External Routines:

- IEDQUI - User Interface routine - to activate the Binary Search routine (IEDQA1) to scan the Termname Table for the specified terminal name.

- OS Wait routine (SVC 1) - to allow the Put Scheduler in the MCP to empty the application program PUT/WRITE work area.

- IGC102 - AQCTL SVC 102 routine - to tpost a special element to the Put Scheduler STCB in the MCP.

Tables/Work Areas: CVT, AVT, DCB, QCB, Termname Table, access method work area, PUT/WRITE work area, DEB, process entry work area, PCB.

Attributes: Reentrant, refreshable, problem program mode, enabled, transient.

Retrieve Service Routine (Chart ES)

Module Name: IEDQES

Entry Point: IEDQES - activated when a RETRIEVE macro is issued in a QTAM application program that is operating with a TCAM message control program.

Functions: The Retrieve Service routine provides TCAM support for message retrieval from a QTAM application program. The routine first converts the terminal name of the message destination to a Termname Table offset. The routine then builds a special element that consists of this offset and other message information: the number and type of the buffer - for retrieval of the first buffer of a message, this element contains the sequence number and type of the buffer; for subsequent buffer retrieval, it contains the relative record address of the buffer to be retrieved.

The AQCTL SVC 102 routine is used to tpost the special retrieve element to the Retrieve Scheduler (IEDQE7) QCB in the PCB in the MCP. The Retrieve Service routine then issues a WAIT to allow time for the MCP to retrieve the requested buffer.

When the special retrieve element is on top of the ready queue, the Retrieve Scheduler gains control. If the buffer is retrieved, the Retrieve Scheduler places it on the element chain of the PCB QCB and posts complete the waiting Retrieve Service ECB in the application program. Otherwise, the routine places an X'04' error return code in register 15.

The Retrieve Service routine regains control at the instruction just after which the OS WAIT command was issued. If an error return code (X'04') is in register 15, the address or sequence number of the buffer requested is incorrect. If the return code in register 15 is equal to X'00', the Retrieve Service routine moves the retrieved buffer into the application program area and builds a compatible QTAM buffer. The routine then tposts (via AQCTL SVC 102) a buffer return element to the Retrieve Scheduler and exits to the next user instruction in the application program.

External Routines:

- IEDQUI - User Interface Routine - to activate the Binary Search routine (IEDQA1) to scan the Termname Table for the specified terminal name.

- IGC102 - AQCTL SVC 102 - to tpost the special element to the Retrieve Scheduler QCB in the message control program.

Tables/Work Areas: CVT, AVT, DEB, access method work area, PCB, Terminal Table entry, QCB, Termname Table, TCB, SCB, LCB, buffer prefix.

Attributes: Reentrant, refreshable, problem program mode, transient, enabled.

Retrieve Scheduler (Chart E7)

Module Name: IEDQE7

Entry Point: IEDQE7 - activated when a special retrieve element from a compatible QTAM application program is at the top of the MCP ready queue.

Functions: The Retrieve Scheduler retrieves a buffer from a disk message queues data set for a compatible QTAM application program. The special retrieve element that activates this routine contains a sequence number for retrieval of the first buffer of a message or the relative buffer address for subsequent buffer retrieval. (The special retrieve element is tposted to the Retrieve Scheduler QCB by the Retrieve Service routine in the compatible QTAM application program.) For an initial request, the Retrieve Scheduler scans the queue-back chain for the specified sequence number until either the number is found or it is determined to be lost or not on the queue. When a buffer is retrieved, the Retrieve Scheduler places the buffer in the element chain of the QCB in the PCB and posts complete the ECB for the waiting Retrieve Service routine.

When the Retrieve Service routine gains control, it empties the buffer and tposts a buffer return element to the Retrieve Scheduler.

The Retrieve Scheduler returns the empty buffer to the buffer unit pool, deallocates main storage for the dummy LCB and SCB, and waits for another retrieve request. When a retrieve request has been handled, the Retrieve Scheduler exits to the DSPDISP entry point of the TCAM Dispatcher.

External Routines:

- IGG019RB or IGG019RO - TCAM Dispatcher - to tpost elements to the ready queue.

- IGC102 - AQCTL SVC 102 - to OS POST the Retrieve Service routine ECB complete.

Tables/Work Areas: AVT, PCB, SCB, LCB, QCB, Terminal Table entry.

Attributes: Reentrant, refreshable, problem program mode, enabled, transient.

APPLICATION PROGRAM NETWORK CONTROL ROUTINES

Operator Control/Application Program Interface Routine (Chart ET)

Module Name: IEDQET

Entry Point: IEDQET - called and loaded during execution time by an TCHNG, RELEASEM, MCPCLOSE, or CLOSEMC macro expansion in an application program.

Functions: This routine allows the user to perform a subset of the TCAM operator control functions from an application program without actually issuing a PUT command for an operator control message.

The Operator Control/Application Program Interface routine uses the AQCTL SVC 102 routine to move a control block (Command Input Buffer) that indicates the type of command and other pertinent data into the PCBWRKA field in a Process Control Block (PCB). This Interface routine tposts the CIB to the Operator Control QCB for processing. It then (except in closedown operations) issues a WAIT to put the application program in the wait state. The format of the CIB is as follows:

<table>
<tr><th>Offset</th><th colspan="4"></th></tr>
<tr><td>0</td><td colspan="4">Operator Control QCB Address</td></tr>
<tr><td>+4</td><td>Priority</td><td colspan="3">Link Field</td></tr>
<tr><td>+8</td><td>Verb Code</td><td>Length X'1C'</td><td>0</td><td>Return Code</td></tr>
<tr><td>+12</td><td colspan="4">ECB Address for Application Program</td></tr>
<tr><td>+16</td><td colspan="4">0</td></tr>
<tr><td>+20</td><td colspan="4">0</td></tr>
<tr><td>+24</td><td colspan="4">0</td></tr>
</table>

When the Cperator Control task has processed the command, it posts the waitinq arplication proqram ECB complete. The Interface routine then reqains ccntrcl and moves the return code set by operator control from the PCB field that contains the CIB to reqister 15 for inspection bv the user.

Tf this routine is invcked when a TCAM MCP is not active in the system; that is, the AVT pointer in the CVT is zero, the routine places a return ccde of X'01' in reqister 15. If an invalid password is specified or if a password is required but not specified, this routine puts a X'14' return code in reqister 15.

If the Cperator Control/Application Proqram Interface routine is activated by an MCPCLOSE or an CLOSEMC macrc expansion and a closedown is already in proqress (AVTCLOSN is set in AVTBIT1), the routine does not perform its functions. It returns to the next sequential instruction in the applicaticn proqram with a return code of X'00' in reqister 15.

Fxternal Routines:

- IGC102 - AQCTL SVC 102 routine - tc mcve data across partition boundaries and to post ECBs complete.
- OS Wait routine (SVC 1) - to put the applicaticn proqram in the wait state.

Tables/Work Areas: CIB, PCB, AVT, CVT, Operator Control QCB.

Attributes: Problem proqram mode, serially reusable, enabled, resident.

TCOPY Service Routine (Chart E1)

Module Name: IEDQE1

Entry Point: IEDQE1 - called when a TCOPY macro is issued in an application program.

Functions: This module copies a terminal entry into a work area in an application program.

The TCOPY Service routine uses the TCAM Binary Search routine (activated via the User Interface routine) to find the Termname Table entry that corresponds to the terminal name specified by the user. The Termname Table entry contains the address of the corresponding Terminal Table entry. The TCOPY Service routine determines the type of entry, computes its size, and moves the entry into the application program work area. Any option fields are also moved into the work area.

If the terminal name specified by the user is invalid, this routine places a return code of X'20' in register 15. A return code of X'08' indicates that TCAM is not in the system and X'0C' indicates that there is not an open DCB in the application program.

External Routine: IEDQUI - User Interface routine - to activate the Binary Search routine (IEDQA1) to scan the Termname Table for the specified terminal name.

Tables/Work Areas: CVT, AVT, Termname Table, Terminal Table, application program work area, TCB, DCB, access method work area.

Attributes: Reentrant, refreshable, enabled, resident, problem program mode.

QCOPY Service Routine (Chart E2)

Module Name: IEDQE2

Entry Point: IEDQE2 - called when a QCOPY macro is issued in an application program.

Functions: This module copies a queue control block (QCB) into a work area in an application program.

The QCOPY Service routine uses the TCAM Binary Search routine (activated via the User Interface routine) to find the Termname Table entry that corresponds to the terminal name specified by the user. The Termname Table entry contains the address of the corresponding Terminal Table entry, and the Terminal Table entry points to the associated Destination QCB. The QCOPY Service routine computes the size of the QCB, including all priority level QCBs, and moves the QCB into the application program work area.

If the terminal name specified by the user is invalid, a return code of X'20' is placed in register 15. If the terminal type is invalid, a return code of X'04' is placed in register 15. A return code of X'08' indicates that TCAM is not in the system and X'0C' indicates that there is not an open DCB in the application program.

External Routine: IEDQUI - User Interface routine - to activate the Binary Search routine (IEDQA1) to scan the Termname Table for the specified terminal name.

Tables/Work Areas: CVT, AVT, Termname Table, Terminal Table, Destination QCB, application program work area, TCB, DEB.

Attributes: Reentrant, refreshable, enabled, resident, problem program mode.

TCHNG Service Routine (Chart E3)

Module Name: IEDQE3

Entry Point: IEDQE3 - called when a TCHNG macro is issued in an application program.

Functions: This routine updates the contents of a Terminal Table entry by copying an altered entry from an application program work area into the Terminal Table.

The TCHNG Service routine uses the Binary Search routine (activated via the User Interface routine) to find the Termname Table entry that corresponds to the specified terminal name. The Termname Table entry points to its associated Terminal Table entry. The TCHNG Service routine determines the type of entry, computes its size, and moves the entry , as well as any option fields, from the application program work area to overlay the Terminal Table entry.

If there is a password in the AVT, the TCHNG Service routine checks for a password as input. If there is an input password, the TCHNG Service routine loads the Password Scrambler routine to scramble the characters. The TCHNG Service routine then compares the passwords to determine whether to update the Terminal Table entry. If the passwords match, the entry is updated; if the passwords do not match, a return code of X'14' is placed in register 15 and the entry is unchanged. A return code of X'08' indicates that TCAM is not in the system, X'0C' indicates that there is not an open DCB, and X'20' indicates that an invalid terminal name was specified.

If the update of the Terminal Table entry is successful and the TCAM checkpoint data set is open (AVTCKGET=0), the TCHNG Service routine links to the TCAM Application Program/Checkpoint Interface routine (IEDQNB) at a special entry point (IEDQNB02).

External Routines:

- IEDQUI - User Interface routine - to activate the Binary Search routine (IEDQA1) to scan the Termname Table for the specified terminal name.

- IEDQE6 - Password Scrambler routine - to scramble the characters of the application program-specified password.

- IEDQNB - Application Program/Checkpoint Interface routine - to take a checkpoint of the MCP after a Terminal Table entry change has been made.

Tables/Work Areas: CVT, AVT, Termname Table, Terminal Table, application program work area, TCB, DEB, access method work area.

Attributes: Reentrant, refreshable, enabled, resident, problem program mode.

ICOPY Service Routine (Chart E4)

Module Name: IEDQE4

Entry Point: IEDQE4 - called when an ICOPY macro is issued in an application program.

Function: This routine copies the invitation list for a line group into a work area in an application program.

By following a chain of system control blocks (see flowchart E4), the ICOPY Service routine compares the "ddname" of each TCAM line-group DCB with the name coded in the ICOPY macro. The routine gets the address of the invitation list from the matching DCB and then computes the size of the list. The routine then moves a copy of the list into the application program work area.

At the completion of this routine, register 15 contains a return code:

- X'00' - normal completion of the ICOPY function.

- X'04' - an invalid relative line number was specified.

- X'08' - TCAM is not in the system.

- X'20' - an invalid "ddname" for line-group DCB was specified.

External Routines: None.

Tables/Work Areas: CVT, AVT, MCP TCB, MCP TIOT, MCP DEB chain, application program work area, DCB, invitation list.

Attributes: Reentrant, refreshable, enabled, resident, problem program mode.

Password Scrambler Routine (Chart E6)

Module Name: IEDQE6

Entry Point: IEDQE6 - called as a subroutine to scramble the characters of a password.

Functions: This routine scrambles the characters of an input password so that it can be compared to an already scrambled password in the AVT. This provides an internal security check to keep programs in other system partitions from altering the contents of the MCP tables and work areas.

External Routines: None.

Tables/Work Areas: None.

Attributes: Problem program mode, reentrant, refreshable, enabled, transient.

OPERATOR CONTROL ROUTINES

Resident Operator Control Module (Chart CA)

Module Name: IEDQCA

Entry Points:

- IEDQCA01 - activated by OS when Operator Control is attached.
- IEDQCA02 - activated by the transient operator control routines when a field in an input operator control command needs to be scanned.

Functions: This module defines the Operator Control AVT and gives control to the initial load of the Operator Control control module (IGC0010D) for command processing.

At the IEDQCA01 entry point of the Resident Operator Control module, the module puts an entry code of 1 in register 0 to indicate to IGC0010D that Operator Control has just been attached and initialization functions must be performed. At the IEDQCA02 entry point, the module puts an entry code of 4 in register 0 to indicate that an input command needs to be scanned. The Resident Operator Control module activates IGC0010D by issuing the TOPCTL macro. The TOPCTL macro expansion issues SVC 104, which loads IGC0010D.

If Operatcr Ccntrol has just been attached and, upon return from the Operator Ccntrol ccntrcl module, closedcwn is in progress, the resident module returns immediately to OS. If, however, Operator Control has just been attached and closedown is nct in progress, this module ensures that all input operator ccntrol commands are processed before returning to OS.

External Routine: SVC 104 - the TOPCTL macro - to activate the Operator Contrcl control module - Load 0 (IGC0010D) for command processing.

Tables/Work Areas: AVT, Operator Ccntrol AVT.

Attributes: Resident, serially reusable, refreshable, enabled.

Operator Contrcl Control Module - Load 0 (Chart Z1)

Module Name: IGC0010D

Entry Point: IGC0010D - activated by IEDQCA, IGC0210D, IGC0410D, or from a subroutine within its own CSECT to process an operator control command.

Functions: The specific functicns cf this module depend cn the entry code that is passed as input in register 0. If the entry code is equal to one, the Operator Control Contrcl Module - Load 0 performs operator control initialization functicns. The mcdule builds an Operator Contrcl ECB in the AVT and then issues an OS WAIT for an operator contrcl command to be tposted tc the Operatcr Control QCB. When the WAIT is satisfied, this mcdule puts an entry code cf 1 in register 11 and then issues an XCTL to IGC0110D, which begins processing the command.

If the entry code is equal to 2, this ccntrol module puts an entry code of 4 in register 0 and executes itself as a subroutine to scan for the next field in the input command. If the command contains an EOB or an EOT, which implies that the command was nct followed by a blank, or if the end of the field or data is reached, the module sets the "last field" indicatcr and returns tc the calling routine.

If the entry code is equal to 3, this mcdule first scans the input command for fields that specify a terminal name, a DDNAME, an absolute address, a relative line number, ONTP, or CFFTP. The control module checks the validity of the format of the fields; and if nc errors are detected, sets register 15 equal tc X'00'. If errors are detected, the module puts X'02' in register 15. The control module then returns to the calling routine.

If the entry code is equal to 4, the Operator Control Control Module - Load 0 scans the input command for a field that is terminated by the end of the the input, an EOB or EOT, 8 characters, cr a valid delimiter. If the module finds a blank in the command, the module sets the "last field" indicator. When the module finds a field-terminating ccndition, it saves the scanned field in the Operator Control AVT, puts the number of bytes scanned in reqister 15, and returns to the calling routine.

External Routine: OS Wait routine (SVC 1) - to wait for an input operator contrcl command.

Tables/Work Areas: AVT, buffer, Cperator Control AVT.

Attributes: Serially reusable, refreshable, enabled, transient.

Operator Contrcl Control Module - Load 1 (Chart Z2)

Module Name: IGC0110D

Entry Point: IGC0110D - activated by IGC0010D, IGC0210D, IGC0310D, or IGC0410D to continue processing an input operator contrcl command.

Functions: The specific functions of this module depend on the entry code that is passed as input in register 11. If the entry code is equal to one, the Operator Control control module - Load 1 prevents any further queuing of operator control commands when closedown is in progress. If a restart is in prcgress, this module performs the command processing defined for an entry code of 2; otherwise, the module performs the processing defined for an entry code of 3.

If the entry code is equal to 2, this control module saves the checkpoint element for this request and loads and gives control to the appropriate cperator control functional processing module or modules. Upon return from the last necessary processing module, the control module deletes the processing module, puts an entry code of 1 in register 11, and transfers control to IGC0310D for further command processing.

If the entry code is equal to 3, this control module processes only commands from the system conscle. When a valid verb, other than START, STOP, or HALT is found, this module puts an entry code of 1 or 2 in register 11 and passes control to IGC0210D for further command processing. For a HALT command, this module performs the processing defined for the entry code 2. For a START or STOP verb or if closedown is in progress, this module dequeues a new input operator control command and starts the entry code 3 processing again. If this module finds an invalid verb in the command, the module passes control to IGC0310D with an entry code of 3 in register 11.

If the entry code is equal to 4, restart is in progress and this module checks for the next restart command. If there is no other command or if closedown is in progress, this module returns to the calling routine. If there is another command that is from an application program or On-Line Test and the command is valid, this module branches to the code for an entry code of 2. If there is another command from an application program and the command is invalid, this module puts an entry code of 3 in register 11 and exits to IGC0310D. Otherwise, the command is from a terminal and this module puts an entry code of 1 in register 11 and passes control to IGC0410D.

External Routines:

- OS QEDIT routine - to prevent queuing further operator control commands.

- OS WTO routine (SVC 35) - to send a message to the system operator.

- OS Load routine (SVC 8) - to load an operator control processing routine.

- OS Delete routine (SVC 9) - to delete the operator control processing routine from main storage.

- Operator Control Processing routines - to perform specific functions requested by an operator control command. These routines are IEDQCF, IEDQCG, IEDQCH, IEDQCI, IEDQCJ, IEDQCK, IEDQCL, IEDQCM, IEDQCN, IEDQCO, IEDQCP, IEDQCQ, IEDQCU, IEDQCV, IEDQCW, IEDQCX, IEDQCZ, IEDQC0, IEDQC1, IEDQC2, IEDQC3, and IEDQC6.

Tables/Work Areas: AVT, buffer, Operator Control AVT.

Attributes: Serially reusable, refreshable, enabled, transient.

Operator Control Control Module - Load 2 (Chart Z3)

Module Name: IGC0210D

Entry Point: IGC0210D - activated by IGC0110D or IGC0410D to continue processing operator control commands.

Functions: The specific functions of this module depend upon the entry code that is passed as input in register 11. If the entry code is equal to 1, this module checks the format of VARY, HOLD, RELEASE, MODIFY, and DISPLAY commands. If the command is valid, this module puts an entry code of 2 in register 11 and exits to IGC0110D. For invalid commands, this module puts an entry code of 3 in register 11 and exits to IGC0310D.

If the entry code is equal to 2, this module checks the format of MODIFY and DISPLAY commands for valid operands. This module exits to IGC0110D with an entry code of 1 in register 11 and to IGC0310D with an entry code of 3 in register 11 for invalid commands.

External Routine: SVC 104 - the TOPCTL macro - to activate IGC0010D with an entry code of 2 to get the next field in the command.

Tables/Work Areas: AVT, Operator Control AVT.

Attributes: Serially reusable, refreshable, enabled, transient.

Operator Contrcl Control Module - Load 3 (Chart Z4)

Module Name: IGC0310D

Entry Point: IGC0310D - activated by IGC0110D, IGC0210D, or IGC0410D to continue processing an input operator control command.

Functions: The specific functions of this module depend on the entry code that is passed as input in register 11. If the entry code is equal to 1, this module processes checkpoint requests. This module builds a checkpcint request element and tposts that element (via SVC 102) to the Checkpoint QCB. Upon return, this module dequeues the request command and, for a terminal request, passes control to IGC0510D. For a request from the console or an application program, this module continues processing as though the entry code is 2. If the request is canceled, this module passes control to IGC0410D with an entry code cf 2.

If the entry code is equal to 2 and the operator control command is from the system console, this module sends a WTO respcnse, frees the CIB, puts an entry code cf 3 in register 11, and exits to IGC0110D. For an entry code of 2 and a command from an application progran or On-Line Test, this module OS POSTs the ECB, puts an entry code of 1 in register 11, and exits to IGC0110D.

If the entry code is equal to 3, this mcdule generates an error message for an invalid command. If the invalid command is from the MCP, this module exits tc IGC0510D, which builds the output message.

External Routines:

- IGC102 - AQCTL SVC 102 routine - to pcst an ECB or tpost an element in the MCP.
- OS Wait rcutine (SVC 1) - to wait for an element from checkpoint.
- OS WTO routine (SVC 35) - to send a message to the system operator.
- OS Delete routine (SVC 9) - to delete an operator control processing routine from main storage.
- OS QEDIT rcutine - to free the CIB.

Tables/Work Areas: AVT, buffer, Cperator Ccntrol AVT.

Attributes: Serially reusable, refreshable, enabled, transient.

Operator Contrcl Contrcl Module - Ioad 4 (Chart Z5)

Module Name: IGCC410D

Entry Point: IGC0410D - activated by IGC0110D, IGC0310D, or IGC0510D to continue processing operator control ccmmands.

Functions: The specific functions of this module depend on the entry code passed as input in register 11. If the entry code is equal to 1, this module calls IGC0010D to scan an operator control command from a terminal to determine whether the command has been canceled. If the command is canceled, no further processing is performed and this module exits to IGC0310D. If the command is not canceled and is a HALT command, this module exits to IGC0210D with an entry code of 2 in register 11. If the command is nct canceled and is nct a HALT command, this mcdule exits to IGC0110D.

If the entry code is equal to 2, this mcdule determines the source of the command. For a command frcm a terminal, this mcdule passes control to the code for an entry code of 3. Otherwise, this module puts an entry ccde of 2 in register 11 and exits to IGC0310D.

If the entry code is equal to 3, this module performs the processing necessary to complete sending a response message. If closedown is nct in progress, this module puts the destination for the response in the buffer, updates the SCB, puts a code of 1 in register 11, and exits to IGC0110D. If closedown is in prcgress or if the terminal is nct connected, this module puts the entry code (1) in register 11 and exits immediately to IGC0110D.

External Routines:

- SVC 102 - AQCTL SVC 102 routine - to tpcst the response message to the Buffer Disposition QCB.
- SVC 104 - the TOPCTL macro - to activate IGC0010D.
- OS Delete routine (SVC 9) - to delete an cperator control processing routine from main storage.

Tables/Work Areas: AVT, buffer, LCB, SCB, Cperator Ccntrol AVT.

Attributes: Serially reusable, refreshable, enabled, transient.

Operator Contrcl Contrcl Module - Load 5 (Chart Z6)

Module Name: IGCC510D

Entry Point: IGC0510D - activated by IGC0310D to build a response message for the MCP.

Functions: The purpose cf this module is tc build an cutput response message that is tc be sent to the MCP. This mcdule determines whether the input buffer is large enough to ccntain the output message, and if it is not, obtains enough additional units to hold the message. This module calculates the required output size by adding the prefix size to the length of the output data. The module multiplies the number of input buffers times the buffer length and then compares the result to the output message length. If the output message is not lcnger, this module puts the message in the buffers and exits to IGC0410D.

If the cutput message is longer than the input buffer size, this module subtracts the buffer length frcm the output length and then divides the result by the number cf bytes in a unit to get the number of additional units required. This module builds an ERB with the required unit count, uses SVC 102 to tpost the ERB to the Buffer Request QCB, and issues an OS WAIT to allow time for the request to be satisfied. Upcn return, this module links the new units tc the input buffer, puts the message in the buffer, and exits to IGC0410D.

External Routines:

- IGC102 - AQCTL SVC 102 routine - to tpcst an ERB to the Buffer Request QCB in the MCP.
- OS Wait routine (SVC 1) - to wait for the ERB request to be satisfied.

Tables/Work Areas: AVT, buffer, ERB, Operatcr Control AVT.

Attributes: Serially reusable, refreshable, enabled, transient, problem program mcde.

TCAM Command Scheduler - SVC 34 (Chart NZ)

Module Name: IGC1303D

Entry Point: IGC1303D - activated when an SVC 34 module recognizes a command with a TCAM keywcrd operand.

Functions: The TCAM Command Scheduler builds a Command Input Block (CIB) for any operator control command entered from the system console. The ccmmands are VARY, HOLD, RELEASE, DISPLAY, MODIFY, and HALT.

The TCAM Ccmmand Scheduler first issues a GETMAIN macro to obtain an area in which to build the CIB. After the CIB is built, the routine issues a QEDIT macro, which puts the CIB into the CIB chain. The scheduler then posts the ECB pointed to by the first word of the Communications Parameter List, so that Operator Control can be activated and exits to the address in register 14. If the return code from QEDIT indicates that the CIB limit has been reached, the scheduler issues a FREEMAIN for the CIB, rejects the operator control command, and issues an indicative error message to the system console by exiting to IGC0503D.

The TCAM Ccmmand Scheduler also exits to IGC0503D to issue an error message if there is no TCAM MCP in the system or if the GETMAIN for main storage is unsuccessful.

External Routines:

- OS Getmain routine (SVC 4) - to obtain main storage.
- OS Freemain rcutine (SVC 5) - to release main storage.
- OS OEDIT routine - to put the CIB in the CIB chain.

Tables/Work Areas: Extended save area, CVT, CIB, AVT.

Attributes: Reentrant, supervisor mode, transient.

Modify Options Routine (Chart CF)

Module Name: IEDQCF

Entry Point: IEDQCF - loaded by the Operatcr Contrcl control module to process DISPLAY OPTION and MODIFY OPTICN commands. The command that caused this routine to be activated is one of the follcwing:

```
[control chars] { MODIFY } ident,OPT=statname,opfldname,data
                {   F    }

[control chars] { DISPLAY} TP,OPTICN,statname,opfldname
                {   D    }
```

Functions: This routine processes operatcr control ccmmands that request display or modification of terminal option fields.

The Modify Cptions routine first lcads its work area (IEDQC5), which is used as a conversion area for both the mcdify and display functions. The routine serially searches the Termname Table for an entry that matches an entry specified in the common input block, which is passed as input to the routine. If a matching entry is not found in the Termname Table, the routine prepares an error message (IED016I) and returns ccntrcl to the Operatcr Ccntrcl control module.

If a matching entry is found, the Modify Options routine serially searches the Option Characteristics Table for the cption field specified in the MODIFY or DISPLAY command. If no option is found, the Modify Opticns rcutine prepares an errcr message (IED034I) and returns control to the control module. When there are no cptions for the terminal entry, the TRMOPTFN field is set off in the TRMSTATE control table.

If the option field specified in the MCDIFY or DISPLAY command is not defined for the terminal entry, the rcutine prepares an error message (IED034I) and returns control tc the Operator Control control module. If the specified option is defined, the Modify Option routine obtains the address of the opticn field and tests the input to determine whether the command is a DISPLAY cr a MODIFY command.

If the command is a MODIFY, the Modify Opticns routine gives control to the Operator Control Scan subroutine (IEDQCA02) by branching to the OPCSCAN entry point in order to cbtain the replacement data. On return, if there is nc data or if end-of-message was reached befcre the end of the field, the rcutine prepares an error message (IE0018I) and returns ccntrol to the ccntrcl module. Otherwise, the Modify Options routine places the replacement data in the work area (IEDQC5).

If the option field type definiticn does not match the replacement data; for example, the field is in decimal notation and the data is in character representation, the routine prepares an error message (IED056I) and returns control to the Operator Control control module. Also, if a hexadecimal data field contains invalid characters, the Modify Options routine prepares an error message (IED077I) and returns control to the ccntrol module.

The Modify Options routine normally places character data in the option field of the MODIFY command, and converts decimal and hexadecimal fields from EBCDIC to the ccrrect format and stores the new values in the option field. If the new fields are larger than the option field, the routine does not store the fields, but prepares an errror message (IED062I) and returns ccntrol to the control module. After the new data has been placed in the cpticn field of the MODIFY command, the Modify Options routine prepares a response message (IED050I) and returns control to the Operatcr Control control module.

If the command is a DISPLAY command, the Modify Options routine places the option field cf the command in the work area (IEDQC5). If the data format is characters and the opticn field is all blanks, the routine places the characters 'ALI BLANKS' into the response message (IED035I). Since no conversion is required for character data, the routine prepares a response message (IED035I) and returns control to the control module. For decimal or hexadecimal representation in the option field, the routine determines whether the field contains zero. If the data is zero, the rcutine prepares a 'data is zero' message (IED035I) and returns control to the control module. If there is data in the option field, the rcutine ccnverts the data into a printable format, prepares a message response (IED035I) and returns ccntrol to the Operator Ccntrol ccntrcl module.

On entry tc the Modify Options routine, the address of the Operator Control AVT is passed in register 1. The common input block is located at the label OPCCKELE. The fields in the common input block that this rcutine uses are the following.

Offset	Field Name	Field Description
+8	OPCTNME	Name of the terminal
+18	OPCFLG	X'80' for DISPLAY command X'40' for MODIFY command
+24	OPCOPFLD	Name of the option field

External Routines:

- IEDQCA - Resident Operator Control module - the Operator Control Scan subroutine (IEDQCA02), to serially search the Termname Table and option field CSECT.

- OS Load routine (SVC 8) - to load the IEDQC5 work area.

Tables/Work Areas: Work areas that contain the fixed portion of each response message and space for insertion of variable data, AVT, Termname Table, Terminal Table entry, Operator Control AVT, buffer, translate tables.

Attributes: Serially reusable, refreshable, enabled, transient, problem program mode.

Copy Line Information Routine (Chart CG)

Module Name: IEDQCG

Entry Point: IEDQCG - loaded by the Operator Control control module to process DISPLAY ADDR commands.

Functions: This routine processes operator control commands that request display of the line address and relative line number for a specified terminal. The command that caused this routine to be activated is as follows:

```
[control chars] (DISPLAY) TP,ADDR,statname
                {       }
                (   D   )
```

The Copy Line Information routine finds the Termname Table through the AVT and serially searches for the entry that matches the one specified in the common input element. If a matching entry in the Termname Table cannot be found, the routine constructs an error message (IED016I) and returns to the control module.

When a matching entry in the Termname Table is found, the Copy Line Information routine places the terminal name in a response message (IED038I) and obtains the address of the corresponding entry in the Terminal Table. If the entry is a process entry (bit 2 is on in TRMSTATE), the entry has no line information, the fact of which is noted in a message response (IED090I) and the routine returns control to the control module.

At this point, the Copy Line Information routine gets the QCB address from the terminal entry (TRMDESTQ) and places its relative line number (after it is converted to printable EBCDIC) in the response message (IED038I). The routine obtains the address of the DCB from the QCB (QCBDCBAD) and determines whether the line attached to the terminal has been opened. If the line is not open, the routine prepares an error message (IED091I) and returns control to the control module. The line is now checked to determine whether an open idle condition has occurred (DEBUCBAD is zero). If so, the routine prepares an error message (IED091I) and returns control to the Operator Control control module.

If the line is open, the Copy Line Information routine places the address of the UCB in a response message (IED038I). Then the routine gets the TCB address from the AVT (AVTTCB) and uses it to find the Task I/O Table (TIOT) address at the location TCB+12. The routine adds the DCBTIOT value to the TIOT starting point, moves the ddname field from the resulting location into the response message (IED038I), and returns control to the control module.

On entry to the Copy Line Information routine, the address of the Operator Control AVT is passed in register 1. The common input block is located at the label OPCCKELE. The routine uses only the OPCTNAME field to get the terminal name from the common input block.

External Routines: None.

Tables/Work Areas: Work areas that contain the fixed portion of each response and space for insertion of variable data, AVT, DEB, DCB, QCB, Termname Table, Terminal Table entry, Operator Control AVT.

Attributes: Serially reusable, refreshable, enabled, transient, problem program mode.

Copy Terminal Information Routine (Chart CH)

Module Name: IEDQCH

Entry Point: IEDQCH - loaded by the Operator Control control module to process DISPLAY TERM commands. The command that caused this routine to be activated is as follows:

```
[control chars] {DISPLAY} TP,TERM,statname
                {   D   }
```

Functions: This routine processes operator control commands that request display of the fields in a specified terminal entry.

The Copy Terminal Information routine obtains the Termname Table address from the AVT and serially searches it for the name that matches the one specified in the common input block. If the matching name cannot be found, the Copy Terminal Information routine prepares an error message (IED016I) and returns control to the control module.

When a name match is found, the Copy Terminal Information routine gets the Terminal Table entry address and places the terminal entry name in the response message (IED033I). The routine converts the input and output sequence numbers (from TRMINSEQ and TRMOUTSQ) to printable characters and places them in the IED033I message.

If the terminal entry is not a process entry (bit 2, X'20', in TRMSTATE is not turned on), the routine gets the intensive mode search table, converts the sense data (TRMSENSE) to printable information, and places it in the IED033I message. If the entry is a process entry, the routine places the characters INTENSE=NO in the IED033I message.

Finally the Copy Terminal Information routine gets the status byte (TRMSTATE), places the printable equivalents of its bit values in the IED033I message, and returns control to the Operator Control control module.

On entry to this routine, register 1 contains the address of the Operator Control AVT, from which the common input block can be obtained. The name of the requested terminal is located in the checkpoint element in the OPCTNME field of the common input block.

External Routines: None.

Tables/Work Areas: Work areas that contain the fixed portion of each response message and space for insertion of variable data, AVT, CIB, Termname Table, Terminal Table entry, Operator Control AVT.

Attributes: Serially reusable, refreshable, enabled, transient, problem program mode.

Copy ICB Information Routine (Chart CI)

Module Name: IEDQCI

Entry Point: IEDQCI - loaded by the Operator Control control module to process DISPLAY LINE commands. The command that cuased this routine to be activated is as follows:

```
[control chars] { DISPLAY } TP,LINE, ddname,rln
                {    D    }          address
```

Functions: This routine processes operator control commands that request display of the ICB fields for a specified line.

The Copy ICB Information routine determines whether the input command format specifies the line in the DDNAME/RLN or hardware address format by checking the common input block, which is obtained from the Operator Control AVT - the address of which is passed in register 1. If DDNAME/RLN is specified and if the relative line number is specified as ALL, the routine rejects the input command, because the relative line number may provide information for a single line only. If the relative line number is not specified as ALL, the routine converts the relative line number tc hexadecimal (for internal TCAM use) and determines whether the result is zerc or greater than 255, the maximum relative line number allowed.

If the relative line number is zero or greater than 255, the Copy LCB Informaticn rcutine rejects the command, prepares an error message (IED018I), and returns ccntrol to the Operator Control control module. If the relative line number is valid, the rcutine gets the address of the TCB from the AVT (AVTTCB), finds the TIOT via the TCB, and determines the offset into the TIOT for the DDNAME that was specified in the input command.

When both DDNAME/RLN and hardware address formats have been specified in the input command, the Copy LCB Information routine gets the starting address of the DEB chain frcm the TCB and locates the first DCB (DEBDCBAD). If the DCB is not for a TCAM line (DSORG does not equal X'40'), the routine gets the next DEB in the chain (DEBDEBAD) and examines that DCB. If there are no DEBs that have associated TCAM DCBs, the routine prepares an error message (IED017I) and returns contrcl to the control module.

If the input command format is DDNAME/RLN, the routine ccmpares the TIOT offset in the DCB (DCBTIOT) to the cne calculated in searching the TIOT. If the offsets are not equal, the routine gets the next DEB and makes the comparison again. When the correct DCB has been located, the routine determines whether the associated UCB (DEBUCBAD) is zerc (implying an open idle ccndition has occurred). If the UCB is zerc, the routine rejects the input command, prepares an error message (IED018I), and returns ccntrol to the control module. If the line specified in the input command is open, but its relative line number is greater than the number of lines in the line group, the routine rejects the command, prepares an error message (IED018I), and returns contrcl tc the ccntrcl module.

If a hardware address was specified in the input command, the Copy LCB Information rcutine gets the UCB from the DEB (DEBUCBAD) and compares it tc the specified address. If the UCB is not the one associated with the DEB, the routine gets the next DEB and makes the comparison again.

For both DDNAME/RLN and hardware address formats, once the DCB address has been verified, the routine determines whether the line has been opened. If the line is not cpen, the routine prepares an error message (IED017I) and returns ccntrol to the control module. If the line is open, the routine determines the address of the LCB from the DCB (DCBIOBAD + (DCBEIOBX x rln) - IOB length). Then the routine gets the LCB status byte and determines whether any bits are on. If there are some bits on, the routine uses the status conversion table to convert the values tc printable data and places the result in a response message (IED032I). If no bits are on, the routine places the characters 'NO BITS ON' in the response message (IED032I). Next, the routine gets the SCB from the LCB, ccnverts the error word to printable data by using the error conversicn table, and places the result in the response message (IED032I). If there are no bits on in the error word, the routine places the characters 'NO BITS ON' in the response message (IED032I). The Copy LCB Information routine returns control to the Operator Control control module.

On entry to this routine, register 1 contains the address of the Operator Control AVT from which the common input block can be obtained at the label OPCCKELE. The Copy LCB Information routine uses only the first sixteen bytes of the common input block, the format of which is as follows.

Offset	Field Name	Field Description
0	OPCCKELE	Address of the common input block
+4	OPCLEN	Length of the relative line number If X'00' - ALL If X'80' - No rln
+5	OPCRLN	EBCDIC relative line number
+8	OPCTNME	DDNAME or hardware address

External Routines: None.

Tables/Work Areas: Work areas that contain the fixed portion of each response message and space for insertion of variable data, AVT, LCB, DCB, DEB, Operator Control AVT.

Attributes: Serially reusable, refreshable, enabled, transient, problem program mode.

Copy QCB Information Routine (Chart CJ)

Module Name: IEDQCJ

Entry Point: IEDQCJ - loaded by the Operator Control control module to process DISPLAY QUEUE commands. The command that caused this routine to be activated is as follows:

```
[control chars] DISPLAY  TP,QUEUE,statname
                 D
```

Functions: This routine processes operator control commands that request the display of the QCB fields for a specified terminal.

The Copy QCB Information routine locates the Termname Table from the AVT and serially searches it for an entry that matches the one specified in the common imput block, the location of which is specified in the Operator Control AVT - the address of which is in register 1. If a matching entry cannot be found, the routine prepares an error message (IED016I) and returns control to the Operator Control control module.

If the terminal entry is found, the Copy QCB Information routine gets the address of the QCB from the entry (TRMDESTQ) and the address of the DCB from the QCB (QCBDCBAD), and checks to determine whether the DCB has been opened. If the DCB is not open, the routine rejects the input command because there is no queue status. The routine prepares an error message (IED091I) and returns control to the control module. If the DCB is open, the routine uses the address of the DEB (DCBDEBAD) to check the UCB address (DEBUCBAD). If the UCB address is zero, a line open idle condition has occurred, and the line is considered to be not open. The routine rejects the input command, prepares an error message (IED091I), and exits to the control module.

If the line is open, the Copy QCB Information routine gets the number of messages on the queue, converts it to a printable number, and places the result, along with the terminal name, in a response message (IED031I). The routine gets the status field (QCBSTAT) from the QCB, converts the status data to printable equivalents, and places the result in the response message (IED031I). If no status bits are on, the routine places the characters 'NO BITS ON' in the response message (IED031I).

The Copy QCB Information routine obtains the first priority QCB associated with the master QCB, converts it to a printable number, and places it in the response message (IED031I). If there are more priority QCBs, each one is converted and the printable equivalent is placed in the response message (IED031I). The routine returns control to the Operator Control control module.

On entry to this routine, register 1 contains the address of the Operator Control AVT, which contains the common input block that points to the label OPCTNME. This label is the name of the terminal for which QCB values are to be displayed.

External Routines: None.

Tables/Work Areas: Work areas that contain the fixed portion of each response message and space for insertion of variable data, AVT, Termname Table, Terminal Table entry, QCB, Operator Control AVT, DCB, DEB.

Attributes: Serially reusable, refreshable, enabled, transient, problem program mode.

Copy Held Terminals Routine (Chart CK)

Module Name: IEDQCK

Entry Point: IEDQCK - loaded by the Operator Control control module to process DISPLAY INTER commands. The command that caused this routine to be activated is as follows:

```
[control chars] {DISPLAY} TP,INTER
                {   D   }
```

Functions: This routine processes operator control commands that request display of the list of terminals that are currently being held.

The Copy Held Terminals routine gets the address of the Terminal Table from the AVT and then steps through the table to test each entry to determine whether it is being held (intercepted). If it is a process entry (X'20' is on in TRMSTATE), it cannot be held. If an entry is being held, the routine sets a bit to indicate an entry found and places the name of the entry in a list. When the end of the Terminal Table is reached, the routine tests the bit to determine whether any entries were held. If no entries are held, the routine returns a response message indicating this to the control module. If the bit is set, the routine places the list of entries in a response message and returns to the control module.

External Routines: None.

Tables/Work Areas: Work areas that contain the fixed portion of each response message and space for insertion of variable data, AVT, Termname Table, Terminal Table, Operator Control AVT.

Attributes: Serially reusable, refreshable, enabled, transient, problem program mode.

Copy Invitation List Entry Routine (Chart CI)

Module Name: IEDQCL

Entry Point: IEDQCL - loaded by the Operator Control control module to process DISPLAY ACT and DISPLAY INACT commands. The command that caused this routine to be activated is as follows:

```
[control chars] {DISPLAY} TP,{ACT  } ,{ddname,rln}
                {   D   }    {INACT}   {address   }
```

Functions: This routine processes operator control commands that request display of a list of either the active or the inactive terminals for a given line.

The Copy Invitation List Entry routine determines whether the ddname/relative line number format or the hardware address format is specified in the common input block. The routine issues an error message (IED018I) and rejects the command if it is a ddname/relative line number and the relative line number specified is ALL, because the Copy Invitation List Entry routine displays data only for a single line. If the relative line number is not ALL, the routine converts the relative line number to a hexadecimal value. If this value is zero or is greater than 255, the routine rejects the command due to an incorrectly specified relative line number, sends the message (IED018I), and exits to the Operator Control Control module. If the relative line number is valid, the Copy Invitation List Entry routine obtains the TCB address from the AVT and serially searches the TIOT (at TCB + 12) for a ddname matching that specified. If it finds no match, the routine builds an errcr message (IED017I) and returns control to the control module (IEDQCA). When it finds a match on the ddname, the Copy Invitation List Entry routine saves the offset into the TIOT.

For both input formats, the Copy Invitation List Entry routine obtains the start of the DEB chain from the TCB (TCB+8). The routine tests the DCB for each line to see if it is a TCAM line DCB (DSORG of X'40') and if it is not, it finds the next DEB (from DEBDEBAD field of the DEB). If the routine reaches the end cf the DEB chain before it finds a valid DCB, it builds an error message (IED017I) and returns control to the contrcl module (IEDQCA).

If the command was specified in the ddname/relative line number format, the Copy Invitation List Entry routine compares the offset into the TIOT (DCBTIOT) to the one it computed above. If they are not equal, the routine obtains and tests the next DEB. When it finds the proper DCB, the routine tests the UCB address in the DEB (DEBUCBAD) for zero, implying line operand dd dummy. If the UCB address is zero, the Copy Invitation List Entry routine builds an error message (IED017I) and returns control to the ccntrcl module (IEDQCA). If the computed relative line number is greater than the number of lines in the line group (found in the DEBNMEXT field of the DEB) the routine rejects the ccmmand, builds an error message (IED017I), and returns control to the control module.

If the hardware address format was specified, the Copy Invitation List Entry routine compares the input address to the UCBs associated with the DEB. The routine then tests the DEB chain as above until it finds a match.

When the Copy Invitation List routine obtains the proper DCB for either format, it tests the DCB to see if it is open. If it is not open, the routine builds an error message (IED017I) and returns control to the control module. If the line is open, the routine obtains the invitation list address from the DCBINVLI field of the DCB. The routine then serially searches the invitation list for each active or inactive entry in the list. The routine obtains the terminal name for each appropriate entry in the list from the Termname Table and places that name in a response message (IED036I or IED037I). When it reaches the end of the list, the Copy Invitation List Entry routine places the line name in the respcnse message and returns control to the Operator Control ccntrcl mcdule.

External Routines: None.

Tables/Work Areas: Work areas that contain the fixed pcrtion of each response message and space for insertion cf variable data, CIB, DEB, DCB, Termname Table, AVT, Operator Control AVT.

Attributes: Serially reusable, refreshable, enabled, transient, problem prcgram mcde.

Copy Operator Control Terminal Routine (Chart CM)

Module Name: IEDQCM

Entry Point: IEDQCM - loaded by the Operator Control control module to process DISPLAY PRITERM and DISPLAY SECTERM commands. The command that caused this routine to be activated is as follows:

```
[control] {DISPLAY} TP, {PRITERM}
          {   D   }     {SECTERM}
```

Functions: This routine processes operator control commands that request display of the primary operator control terminal or the list of secondary operator control terminals.

The Copy Operator Control Terminal routine locates the Termname Table from the AVT and checks the common input block, the location of which is obtained from the Operator Control AVT, the address of which is in register 1, to determine the type of input command. The OPCFLG field contains X'80' for the primary operator control terminal and X'40' for any secondary operator control terminals. If the primary operator control terminal is to be displayed, the routine gets the terminal offset from the AVT (AVTOPCON) and determines whether it is zero (indicates the system console). If the offset is zero, the routine places the name SYSCON in the response message (IED041I) and returns control to the Operator Control control module. If the offset is not zero, the routine adds the offset to the start of entries in the Termname Table. The name at the resulting address is placed in a response message (IED041I), and control returns to the control module.

If the list of secondary operator control terminals is to be displayed, the Copy Operator Control Terminal routine examines each entry in the Terminal Table. If an entry is for a secondary operator control terminal (TRMSCNYN on in TRMSTATE), its name is placed in the list to be returned as a response (IED043I SECONDARY=statname). After all entries have been checked, the routine returns the list to the control module.

External Routines: None.

Tables/Work Areas: Work areas that contain the fixed portion of each response message and space for insertion of variable data, AVT, Termname Table, Terminal Table entry, Operator Control AVT.

Attributes: Serially reusable, refreshable, enabled, transient, problem program mode.

Change Control Terminal Routine (Chart CN)

Module Name: IEDQCN

Entry Point: IEDQCN - loaded by the Operator Control control module to process MODIFY OPERATOR commands. The command that caused this routine to be activated is as follows:

```
[control chars] { MODIFY } ident,OPERATOR= { statname }
                {   F    }                 { SYSCON   }
```

Functions: This routine processes operator control commands requesting that the primary operator control terminal be changed to the terminal specified in the command.

The Change Control Terminal routine first checks the common input block, the location of which is obtained from the Operator Control AVT, the address of which is in register 1, to determine whether the specified primary terminal is the system console. If it is, the routine compares the primary terminal offset in the AVT (AVTOPCON) to zero. If the offset is zero (the system console is already primary), the routine prepares a response message (IED042I) and branches to the Operator Control control module.

If the system console is not already primary, the AVTOPCON field is set to zero. If the Operator Awareness Message Router routine (IEDQNX) is present in the TCAM system (AVTNX is not equal to zero), the Change Control Terminal routine deletes the Operator Awareness Message Router routine and sets the AVTNX field in the AVT to zero. Then the routine prepares a response message (IED041I) and returns control to the control module.

If the offset of the primary terminal specified in the common input block is not zero, the Change Control Terminal routine finds the Termname Table from the AVT and serially searches it for an entry to match the one specified in the common input block. If there is no matching entry, the routine prepares an error message (IED016I) and returns control to the control module. If a matching entry is found, the routine checks to determine whether it is a valid secondary terminal (TRMSCNYN is on in TRMSTATE). If it is not valid, the routine prepares an error message (IED044I) and returns control to the Operator Control control module.

Once the matching entry has been located, the Change Control Terminal routine compares the terminal offset from the beginning of the Termname Table to the contents of the AVTOPCON field in the AVT, and if they are the same, the terminal is already a primary terminal. The routine prepares a response message (IED042I) and exits to the control module. If the offset and AVTOPCON are not the same, the routine places the new offset (of the matching terminal found) in the AVTOPCON field and determines whether the Operator Awareness Message Router routine is present (AVTNX is not equal to zero). If it is not in the system, the Change Control Terminal routine loads the Operator Awareness Message Router routine, stores its address in the AVTNX field of the AVT, prepares a response message (IED041I), and returns control to the Operator Control control module.

External Routines:

- OS Load routine (SVC 8) - to load the Cperator Awareness Message Router routine (IEDQNX).
- OS Delete rcutine (SVC 9) - to delete the Operator Awareness Message Router routine (IEDQNX).

Tables/Work Areas: Work areas that contain the fixed portion of each response message and space for insertion of variable data, AVT, Termname Table, Terminal Table entry, Operatcr Control AVT.

Attributes: Serially reusable, refreshable, enabled, transient, problem prcgram mcde.

Change Terminal Rcutine (Chart CO)

Module Name: IEDÇCO

Entry Point: IFDQCO - loaded by the Operator Control control module to process VARY TERMINAL commands. The ccmmand that caused this routine to be activated is as follows:

```
[control chars] {VARY} termname,{CNTP } {, E }
                { V  }          {OFFTP} {  B }
```

Functions: This routine processes operatcr control ccmmands that request that a specified terminal be either activated or deactivated for entering, cr for both entering and accepting.

The Change Terminal routine serially searches the Termname Table for an entry matching that specified in the common input blcck. If it does not find a match, the Change Terminal routine builds an error message (IED016I) and returns control to the Operatcr Control control module (IEDQCA).

When it finds a matching entry, the Change Terminal routine tests it for a process entry (X'20' in tte TRMSTATE field). If the entry is a process entry, it has no invitation list, and the Change Terminal routine builds an error message (IED090I) and returns ccntrcl to the control module.

The Change Terminal routine uses the QCB at TRMDESTQ in the Termname Table to obtain the relative line number (QCBRELLN) and the DCB (QCBDCBAD). The routine then tests the DCB fcr open status, and if it is not open the routine tuilds an error message (IED091I) and returns contrcl to the ccntrcl module.

The Change Terminal routine gets the address of the DEB from the DCBDEBAD field cf the DCB and uses the DEB to find the UCB address (DEBUCBAD). The routine tests the UCB for zero. If it is zero, the line has been opened DDDUMMY, so the routine tuilds an error message (IED091I) and returns to the ccntrcl module.

The routine uses the DCBIOBAD field of the DCB to find the LCB address and then tests the LCB for dial (the LCBDIAL bit is on in LCBSTAT2). If the line is a dial line, the Change Terminal routine builds an error message (IED088I) and returns to the control module. The Change Terminal routine tests the line to see if it is stopped. If it is not stopped, the routine rejects the command, builds a message (IED089I), and returns control to the control module.

The Change Terminal routine obtains address of the invitation list from the DCBINVLI field of the DCB.

The routine tests the input block to determine if it contains a VARY ON or OFF command. If it is on, the routine tests the invitation list to determine if all the entries are active. If the entries are all active the Change Terminal routine builds a message (IED019I) and returns control to the control module.

The Change Terminal routine tests each entry in the invitation list to see if it is the entry for the terminal. If the entry is for the terminal and if it is not already active, the Change Terminal routine swaps it from the inactive to its proper location in the active part of the list. The routine then sets a bit to indicate that the list was changed. When it reaches the end of the list, the Change Terminal routine tests this bit. If the bit is not on, the terminal is already active and the routine builds an "already active" message (IED019I) and exits to the control module. If the bit is on, the routine builds a message (IED020I) and returns control to the Operator Control control module.

If the command is a VARY OFF, the Change Terminal routine tests the invitation list for active status. If all the entries in the list are inactive, the routine builds a response message (IED025I) and exits to the control module.

The routine tests each entry in the invitation list to see if it is an entry for the terminal. If it is for the terminal and if it is active, the routine swaps it from the active to the inactive side of the list, decrements the active count, and sets a bit to indicate that the list was changed. When the routine has examined all the entries in the list, it tests the bit. If the bit is off the terminal was already inactive, so the routine builds an "already stopped" message (IED025I) and returns control to the control module. If the bit is on, the Change Terminal routine builds a message (IED026I) and returns to the control module.

External Routines: None.

Tables/Work Areas: Work areas that contain the fixed portion of each response message and space for insertion of variable data, AVT, Termname Table, Terminal Table entry, QCB, LCB, DEB Operator Control AVT.

Attributes: Serially reusable, refreshable, enabled, transient, problem program mode.

Alter Trace Status Routine (Chart CP)

Module Name: IEDQCP

Entry Point: IEDQCP - loaded by the Operator Control control module to process MODIFY TRACE commands. The command that caused this routine to be activated is as follows:

```
[control chars] {MODIFY} ident, TRACE={ddname, rln} {, ON }
                {  F   }               {address    } {  OFF}
```

Functions: This routine processes operator control commands that request a change of trace status for a specified line.

The Alter Trace Status routine first checks the common input block, passed to it from the Operator Control AVT, the address of which is in register 1, to determine whether the DDNAME/RLN or the hardware address format was specified in the input command. If DDNAME is specified, the routine determines whether RLN equals ALL; and if so, the routine rejects the command, prepares an error message (IED018I), and returns control to the Operator Control control module, because trace can be altered only on a single line basis.

If the RLN is not ALL, the routine converts the RLN value to hexadecimal and determines whether the result is zero or greater than 255, the maximum value, either of which is invalid for a relative line number. The routine rejects the command, prepares an error message (IED018I), and returns control to the control module.

If the RLN is valid, the Alter Trace Status routine gets the address of the TCB from the AVT (AVTTCB) and serially searches the TIOT, the address of which is in the TCB, for the ddname specified. If no matching ddname is found, the routine prepares an error message (IED017I) and exits to the control module. When a matching ddname is found, the routine saves the offset into the TIOT.

For both DDNAME/RLN and address formats, the Alter Trace Status routine locates the DEB chain from the TCB and examines each DCB (DEBDCBAD) to determine whether it is a TCAM line DCB (DCBDSORG=X'40'). If it is not a TCAM line DCB and if the end of the DEB chain has been reached, the routine prepares an error message (IED017I) and branches to the control module.

For the DDNAME/RLN format only, the Alter Trace Status routine comapres the TCAM line DCB's TIOT offset to the one just saved. If the offsets are not the same, the routine continues searching the DEB chain. When the correct DCB is found, the routine compares the relative line number to the number of lines in the line group (DEBNMEXT). If the relative line number is larger, the routine prepares an error message (IED017I) and returns control to the control module. The routine checks the line for an open idle condition (DEBUCBAD field is zero), and if it has occurred, prepares an error message (IED017I) and exits to the control module.

For the address format, the Alter Trace Status routine compares each UCB associated with a DEB to the address specified in the input command, and searches the DEB chain until a matching address is found.

Once the correct DCB is found for either format, the Alter Trace Status routine checks the DCB for an open condition. If the DCB is not open, the routine prepares an error message (IED017I) and returns control to the control module. If the DCB is open, the routine calculates the LCB address from the DCB. Next, it checks the AVT for the presence of a Trace Table (AVTRACE field is not zero). If there is no Trace Table, the Alter Trace Status routine prepares an error message (IED055I) and returns control to the control module.

If a Trace Table is found and if a trace is to be performed, the Alter Trace Status routine checks the trace bit. If it is already on, the routine prepares a response message (IED024I) and returns control to the control module. Otherwise, the routine turns on the trace bit (LCBTRACE in the LCBSTAT2 field of the LCB), turns on a bit to indicate that a checkpoint is needed, prepares a response message (IED023I), and returns control to the control module.

If a trace is to be stopped and the LCB trace bit is not on, the routine prepares a response message (IED030I) and exits to the control module. Otherwise, the Alter Trace Status routine turns the trace bit off, turns the checkpoint bit on, prepares a response message (IED029I), and returns control to the Operator Control control module.

<u>External Routines:</u> None.

<u>Tables/Work Areas:</u> Work areas that contain the fixed portion of each response message and space for insertion of variable data, AVT, LCB, DCB, DEB, Operator Control AVT.

<u>Attributes:</u> Serially reusable, refreshable, enabled, transient, problem program mode.

Stop/Resume Terminal Transmission Routine (Chart CQ)

<u>Module Name:</u> IEDQCQ

<u>Entry Point:</u> IEDQCQ - loaded by the Operator Control control module to process HOLD and RELEASE operator control commands and RELEASEM and MRELEASE application program macros. The command that caused this routine to be activated is one of the following.

```
[control chars] {VARY} termname, {ONTP }, E
                { V  }           {OFFTP}


[control chars] {HOLD    } TP=statname
                {  H     }
                {RELEASE }
                {  A     }
```

Functions: This routine processes cperator control commands requestinq that a specified terminal be prevented frcm acceptinq messaqes or reauestinq release of a specified held terminal.

The Stop/Resume Terminal Transmissicn routine serially searches the Termname Table for an entry matchinq that in the common imput block. If the routine reaches the end cf the table before findinq a match, it builds an error messaqe (IED016I) and returns ccntrcl to the Operator Contrcl contrcl module (IEDQCA). When the Stop/Resume Terminal Transmission routine finds a matchinq entry in the Termname Table, it saves the address of the entry in a reqister. The routine then tests the input blcck to see if a HCLD or RELEASE is to be done.

If a HOLD is to be done, the Stop/Resume Terminal Transmission routine tests the entry to see what type it is. If it is nct a single terminal the routine builds an error messaqe (IED060I) and returns control to the control module.

If the terminal is already held (the TRMHELDN bit is on in TRMSTATE), the Stop/Resume Terminal Transmission routine tests to see if it was held by a HOLD or by a VARY command. If the command was HOLD, the routine builds a messaqe (IED052I) and returns to the control module. If the ccmmand was VARY, the routine tests a bit set by the Change Terminal routine (IEDQCO). If the bit is on, the Stop/Resume Terminal Transmission routine builds a messaqe (IED025I) and returns tc the ccntrol module. If the bit is off, the routine builds a messaqe (IED026I) and returns contrcl to the contrcl module. If the hold code is not in the system (the AVTAS field in the AVT is zero), the terminal cannot be held. If this is the case, the routine builds an error messaqe (IED060I) and exits tc the control module.

The Stop/Resume Terminal Transmission routine cbtains the address of the QCB from the TRMDESTQ field of the Terminal Table and tests the QCB for its queue type. If it is a main-storage-only queue (the OCBCORE bit is on in the QCBDSFLG byte), the terminal cannot be held. The routine builds an error messaqe (IED060I) and returns to the control module.

The Stop/Resume Terminal Transmission routine cbtains the address of the DCB from the QCBDCBAD field of the QCB and tests it to see if it is open. If the line is nct open, the routine builds an error messaqe (IED060I) and returns to the control module. If the line is open dd dummy (the DEBUCBAD of the DEB is zero), the routine builds an error messaqe (IED060I) and returns contrcl to the control module.

If the entry is queued by terminal (the QCBTERMQ bit in the QCBFLAG byte is on), the Stop/Resume Terminal Transmissicn routine turns on the QCB held flaq (QCBTRMHO in QCBSTAT). The rcutine then turns on the hold bit in the Terminal Table (TRMHELDN in TRMSTATE) and sets the checkpoint bit (OPCCKBIT in the Operator Control AVT). The routine tests the input to see if this is a HOLD command only or if it is an entry for a VARY terminal. The routine builds the appropriate response messaqe (IED051I for a HCLD command; IED025I or IED026I for VARY depending upon the bit set by the Change Terminal routine), and returns to the control mcdule.

If the Stop/Resume Terminal Transmission routine determines that it must perform a release function, it tests the hold bit in the Terminal Table entry. If the bit is off, the terminal is already released. The routine then determines whether the command was a RELEASE or a VARY. If the command was RELEASE, the routine builds a message (IED053I) and returns to the control module. If the command was VARY, the Stop/Resume Terminal Transmission routine tests a bit set by the Change Terminal routine. If the bit is on, the routine builds a message (IED020I), or if the bit is off, the routine builds a message (IED019I) and returns control to the control module.

If the hold bit is off, the Stop/Resume Terminal Transmission routine builds an element request block (ERB) to request a buffer, and issues an AQCTI macro (IGC102) to obtain the buffer. The routine then issues a WAIT (SVC 1) for the buffer to be received. When the WAIT is satisfied, the Stop/Resume Terminal Transmission routine removes the new buffer from the operator control queue, places the terminal-to-be-released offset in the new buffer, and queues the buffer to the IEDQAS01 entry point of the Hold/Release Terminal routine (IEDQAS). The Stop/Resume Terminal Transmission routine issues another AQCTL macro to give the new buffer to the release function (IEDQAS01) above. The routine builds a response message (IED020I if the command was RELEASE and IED054I if the command was VARY). The Stop/Resume Terminal Transmission routine then sets the checkpoint flag and returns control to the control module.

External Routines:

- IGC102 - AQCTI SVC 102 - to obtain a new buffer and to post it to TCAM for further processing by the Hold/Release Terminal routine.

- OS Wait routine (SVC 1) - to wait for a buffer to be received.

Tables/Work Areas: Work areas that contain the fixed portion of each response message and space for insertion of variable data, AVT, CIB, Termname Table, Terminal Table entry, Operator Control AVT, ERB, DEB, OCB, and DCB.

Attributes: Serially reusable, refreshable, enabled, transient, problem program mode.

Start Line Routine (Chart CU)

Module Name: IEDQCU

Entry Point: IEDQCU - loaded by the Operator Control control module to process STARTLN and VARY ONTP commands. The command that caused this routine to be activated is as follows:

```
[control chars] {VARY} {ddname,rln} ,ONTP
                { V  } {ddname     }
                       {address    }
```

Also, the STARTLN macro in the QTAM message processing program can activate this routine.

Functions: This routine processes operator contrcl commands that request startinq a line or line qrcup.

The Start Line routine examines the common input block, located in the Operator Ccntrol AVT, the address of which is in register 1, to determine the input ccmmand format. If the input is a DCB address, the routine qets the address of the DEB from the DCBDEBAD field and beqins processinq the DCB. If the input is a terminal name, the routine serially searches the Termname Table for an entry that matches the one specified in the input command. If the end of the table is reached before a match is found, the routine sets a X'04' return code in register 15 and returns control to the Operator Contrcl control module. When a matchinq entry is found, the routine locates the QCB (from the TRMDESTQ field of the terminal entry), which provides the addresses of the DCB and DEB, and beqins prccessinq the DCB.

If the input command format is DDNAME/RLN and the relative line number is not ALL or is not already in hexadecimal, the Start Line routine converts the line number to hexadecimal and determines whether it is zero or qreater than 255. If the relative line number is either, the rcutine rejects the command, prepares an errcr messaqe (IED018I), and returns ccntrol to the ccntrcl module.

If the relative line number is valid, the routine qets from the TCB the address of the TIOT and serially searches it for an entry that matches the specified DDNAME. If the end of the TICT is reached and no matchinq ddname is found, the routine prepares an error messaqe (IED017I) and returns control to the control module. If a matchinq ddname is found, the routine saves its offset into the TIOT. Then the routine qets the DEB chain and determines whether each DCB is a TCAM line DCB (DSORG field contains X'40') and has the same TIOT offset (DCBTIOT) as the one just saved. If no DCB is found in the DEB chain, the Start Line routine prepares an error message (IED017I) and exits to the control module. Otherwise, when the correct DCB is found, the routine beqins prccessinq the DCB.

If a line address is specified in the input command, the Start Line routine locates the DEB chain and checks each UCB associated with a DEB to determine whether the UCB has an ID matchinq the specified address. If no matchinq ID is found, the routine prepares an error messaqe (IED017I) and returns control to the ccntrol module.

Now the Start Line routine is ready to process the DEB (for all input formats). First, the routine checks the DCB open status, and if it is not open, prepares an error messaqe (IED017I) and returns control to the ccntrol module.

If the DCB is open and a sinqle line is to be started, the routine compares its relative line number to the number of lines in the line qroup. If the line number is larqer than the number of lines, the routine prepares an error messaqe (IED017I) and returns ccntrcl to the control module. Otherwise, the rcutine determines whether the line is an open DD DUMMY, and if so, the routine prepares an error message (IED017I) and returns control to the contrcl module. If a line qroup is to be started and a line is a DD DUMMY, the routine prepares the same error message and then checks the next line in the group. If all lines in the group are open DD DUMMY, the routine prepares the error message (IED017I) and returns control to the control module.

From the DCB in the DCBIOBAD, the Start Line routine gets the LCB address and determines whether the activity of the line associated with it has been stopped. If line activity has not stopped and the input command request is from the Telecommunications On-Line Test Executor (TOTE), the routine tposts the line LCB to itself via SVC 102 (IGC102) in order to return the line from TOTE to TCAM. The routine returns control to the control module. If the input command request is not from TOTE, the routine gets the next LCB to be processed. When all LCBs have been processed, if any lines associated with them have been made active, the routine prepares a response message (IED020I) and returns control to the control module. Otherwise, the routine prepares another response message (IED019I) and exits to the control module.

If the line is not active, the Start Line routine turns the receive bit on in the associated LCB. The routine prepares a NOP command for a switched line, an Enable command for a nonswitched line, and a SAD command for a line attached to a 2702 control unit, which complets the channel program. Now the routine issues an EXCP (SVC 0) command to start the line activity and turns on the line-started bit in the LCB for the line. The Start Line routine continues getting LCBs and processing them until all lines in the system have been processed.

External Routines:

- IGC102 - AQCTL SVC 102 routine - to return a line from TOTE to TCAM

- OS EXCP routine (SVC 0) - to start a line.

Tables/Work Areas: Work areas that contain the fixed portion of each response and space for insertion of variable data, LCB, AVT, Operator Control AVT, QCB, DEB, DCB, Termname Table, Terminal entry.

Attributes: Serially reusable, refreshable, enabled, transient, problem program mode.

Stop Line Routine (Chart CV)

Module Name: IEDQCV

Entry Point: IEDQCV - loaded by the Operator Control control module to process STOPLN, VARY OFFTP(C), VARY OFFTP, and VARY OFFTP (I) commands. The command that caused this routine to be activated is as follows:

```
[control chars] {VARY} {(ddname, rln)} ,OFFTP, {C}
                { V  } { ddname      }         {I}
                       { address     }
```

Also, the STOPLN macro in the QTAM message processing program, the ICHNG macro in the TCAM application program, and the CLOSEMC or MCPCLOSE macros can activate this routine.

<u>Functions:</u> This routine processes operator control commands that request stopping activity on a line either immediately or at the completion of the current operation.

The Stop Line routine first checks the closedown bit in the AVT (AVTCLOSN in AVTBIT1 field). If it is on, indicating that all lines in the system are to be made inactive, the routine sets a flag in the stop line request element.

At this point the Stop Line routine examines the common input block located at the OPCCKELE entry point in the Operator Control AVT, the address of which is in register 1, to determine the format of the input command. The routine uses the following fields of this block.

Offset	Field Name	Field Description
+4	OPCLEN	X'00' for STOPLINE ALL. X'40' for relative line number in hexadecimal. X'80' for no relative line number. bits 1-3 contain the relative line number, if present.
+5	OPCRLN	Relative line number
+8	OPCTNME	DCB address, terminal name, or line address.
+18	OPCFLG	X'80' for VARY (I) X'40' for VARY (C)

If the input command format contains a DCB address, the routine gets the DEB address from the DCB (DCBDEBAD) and branches to process the DCB. If it contains a terminal name, the routine serially searches the Termname Table for a matching entry. If no match is found, the routine sets a return code of X'04' in register 15 and returns control to the control module. Otherwise, when the entry is found, the routine gets the associated QCB (from TRMDESTQ), the DCB from the QCB (QCBDCBAD) and the DEB from the DCB (DCBDEBAD). Now the routine is ready to branch to process the DCB.

If the input command format is DDNAME/RLN and the RLN is still in character representation, the Stop Line routine converts it to hexadecimal and checks to determine whether the result is zero or greater than 255. If the relative line number is either, which is invalid, the routine prepares an error message (IED018I), sets a return code of X'04' in register 15, and returns control to the control module. When the relative line number is valid, the routine finds the TIOT through the TCB (AVTTCB) and serially searches the TIOT for a ddname that matches the one in the input command. If no matching entry is found, the routine prepares an error message (IED017I), sets a return code of X'04' in register 15, and exits to the control module. For a matching ddname, the routine gets the DEB chain from the TCB and checks each DCB associated with a DEB to determine whether it is a TCAM line DCB (DSORG is X'40') and whether the TIOT offset matches that found while searching the TIOT. If no matching offset is found in the DEB chain, the routine prepares an error message (IED017I), sets a return code of X'04' in register 15, and exits to the control module. When the correct DCB is found in the DEB chain, the routine branches to process the DCB.

If the input command specifies the line address format, the Stop Line routine gets the DEB chain and checks each UCB associated with a DEB in the chain for an ID to match the one specified in the input. If no matching ID is found in the DEB chain, the routine prepares an error message (IED017I), sets a return code of X'04' in register 15, and returns control to the control module.

At this point, no matter which input command format was specified, an appropriate DCB is ready to be processed. The Stop Line routine determines whether STOPLINE ALL was specified on the input command, and if so, performs the following operations. The routine places the DEB and DCB addresses in the stop line request element, tposts the element complete, and issues a WAIT instruction (SVC 1) to wait for at least one line to be stopped. When the WAIT has been completed, the routine prepares a response message (IED026I) and returns control to the control module.

If only one line is to be stopped, the Stop Line routine compares the relative line number to the number of lines in the group. If the line number is greater, the routine prepares an error message (IED017I), sets a X'04' return code in register 15, and gives control to the control module. When the relative line number is valid, the routine checks for an open idle condition (OPEN DD DUMMY). If this condition has occurred, the routine prepares an error message (IED017I), sets a X'04' return code, and gives control to the control module. Otherwise, the routine gets the address of the line LCB from the DCB (DCBIOBAD) and determines whether the line has already been stopped. If it has been stopped, the routine prepares a response message (IED025I), sets a X'00' return code in register 15, and exits to the control module. If the line is still active, the routine places the DEB and DCB addresses in the stop line request element, tposts the element complete, and issues a WAIT (SVC 1) instruction to wait for the line to be stopped. When the WAIT has been completed, this routine checks an indicator to determine whether there are more lines to stop. If not, the Stop Line routine prepares a response message (IED026I), sets in register 15 a X'00' return code for single or all lines or a X'14' return code if closedown is specified, and returns control to the Operator Control control module. Otherwise, this routine gets the next LCB and proceeds as previously described.

External Routines:

- IGC102 - AQCTL SVC 102 routine - to tpost the STOPLINE request.
- OS Wait routine (SVC 1) - to wait for completion of the STOPLINE request.

Tables/Work Areas: Work areas that contain the fixed portion of each response and space for insertion of variable data, QCB, LCB, AVT, CIB, Operator Control AVT, DCB, DEB, Termname Table, terminal entry.

Attributes: Serially reusable, refreshable, enabled, transient, problem program mode.

Modify Poll Routine (Chart CW)

Module Name: IEDQCW

Entry Point: IEDQCW - called by the Operater Control control module to process MODIFY AUTOPOLL commands. The command that caused this routine to be activated is as follows:

```
[control chars] {MODIFY}ident,AUTOPOLL={ddname,rln},{ON }
                {  F   }               {address   } {OFF}
```

Functions: The Modify Poll routine processes operator control commands that request that autopolling be started or stopped for a specified line.

The Modify Poll routine tests the common input block to determine whether the DDNAME/RLN or the hardware address format was used. If the ddname was used, the routine checks the relative line number for ALL. If it is ALL, the routine builds an invalid command message (IED018I) and returns control to the Operator Control control module (IEDQCA). If the relative line number is not ALL, the Modify Poll routine converts it to a hexadecimal value and determines whether it is zero or greater than 255. Either of these is invalid; therefore, the routine rejects the command (IED018I) and returns to the control module.

If the relative line number is valid, the Modify Poll routine obtains the address of the TCB from the AVT and then serially searches the TIOT (at TCB+12) for the related ddname. If it finds no match, the routine builds an error message (IED017I) and returns control to the control module. If the routine finds a matching entry, it saves the offset into the TIOT.

For both input formats, the start of the DEB chain is at TCB+8. The Modify Poll routine tests the associated DCB (DEBDCBAD) for a TCAM line DCB (DSORG of X'40'). If the DCB is not a TCAM line DCB, the routine obtains the next DCB and processes the DEB chain. If it finds no TCAM line DCBs, the routine builds an error message (IED017I) and returns control to the control module.

For a DDNAME input format, the Modify Poll routine compares the TCAM line DCB's TIOT offset to that offset computed above. If the offsets do not match, the routine conintues to search through the DEB chain. When it finds the proper DCB, the routine tests its UCB (at DEBUCBAD) for zero, which implies an OPEN DD DUMMY. If the UCB is zero, the Modify Poll routine builds an error message (IED017I) and returns control to the control module.

For a line address format, the Modify Poll routine tests each UCB associated with the DEB for a match on the address. It continues to search through the DEB chain until it finds the matching UCB.

Once it obtains the proper DCB for either format, the Modify Poll routine tests it for open status. If the DCB is not open, the routine builds an error message (IED017I) and returns control to the control module. If the relative line number is greater than the number of lines in the line group, the routine builds an error message (IED017I) and exits to the control module. If the UCB is not a communications UCB or is not capable of autopoll, the Modify Poll routine builds a message (IED057I) and returns to the control module.

The routine obtains the address of the invitation list from the DCBINVLI field of the DCB and tests the input block to see if autopoll is to be started or stopped. If autopoll is to be started and is already started or is to be stopped and is already stopped, the Modify Poll routine builds an appropriate response (IED022I or IED028I) and returns control to the control module. Otherwise, the routine turns on or off the appropriate bit in the invitation list indicating that autopoll is started or stopped. It then turns on the checkpoint flags, builds the appropriate response message (IED021I or IED027I), and returns control to the control module.

External Routines: None.

Tables/Work Areas: Work areas that contain the fixed portion of each message and space for insertion of the variable data, AVT, CIB, DCB, DEB, and the Operator Control AVT.

Attributes: Serially reusable, refreshable, enabled, transient, problem program mode.

Modify Intense Routine (Chart CX)

Module Name: IEDQCX

Entry Point: IEDQCX - loaded by the Operator Control control module to process MODIFY INTENSE commands. The command that caused this routine to be activated is as follows:

```
[control chars]{MODIFY}ident,INTENSE={TERM,termname     },sense,count
               {  F   }              {LINE,{ddname,rln}}
                                     {     {address   }}
```

Functions: This routine processes operator control ccmmands that request modification of the sense infomaticn for intensive recording.

The Modify Intense routine tests the common input block to determine whether modification is for a terminal or for a line. If the modificaticn is for a terminal, the routine serially searches the Termname Table fcr an entry matching that specified in the input. If the routine reaches the end of the table before it finds a match, it builds an error message (IED016I), sets a X'04' return code in register 15, and returns control to the Operator Control control module (IEDOCA). When a matching entry is found in the Termname Table, the Modify Intense routine saves its address.

If the modificaticn is for a line, the routine tests the input for the DDNAME/RLN cr the hardware address fcrmat. If it is the ddname format, the rcutine tests the relative line number for ALL. If it is ALL, the routine builds an error message (IED018I), sets a X'04' return code in register 15, and returns ccntrcl to the control module. Otherwise the Mcdify Intense routine converts the relative line number to hexadecimal and tests the result fcr cbvious errcrs - zero or greater than 255. If the relative line number hexadecimal value is either zerc cr greater than 255, the routine builds an error message (IED018I), sets the X'04' return code, and returns to the control module.

The Modify Intense routine gets the address of the TICT from the TCB (at TCB+12) and serially searches the TIOT for an entry matching that specified in the input. If it finds nc match, the rcutine builds an error message (IED017I) and returns to the control module with a X'04' return code in register 15. When it finds in the TIOT an entry that matches the input, the routine saves its offset into the TIOT.

For both the DDNAME/RLN or hardware line address fcrmat for line modification, the Modify Intense routine finds the start of the DEB chain at TCB+8 and then tests the DCB associated with each DEB for TCAM line status (DSORG of X'40'). If it finds no TCAM line before the end of the DEB chain, the routine builds an error message (IED017I), sets a X'04' return code, and returns control to the control module.

For the ddname/rln input format, the Modify Intense routine compares the hexadecimal relative line number equivalent to the DCBTIOT field. If they dc not match, the routine finds the next DEB and continues to process the DEB chain. When it finds the proper DCB, the Modify Intense routine compares the relative line number to the number of lines in the line group (DEBNMEXT) and if the relative line number is high, the routine builds an errcr message (IED017I) and returns control to the control module with the X'04' return code. The routine then tests the line for CPEN DD DUMMY (DEBUCBAD of zero) and if the line is CPEN DD DUMMY the routine builds an error message (IED017I), sets a X'04' return code in register 15, and returns to the control module.

For the address format, the Modify Intense routine checks each UCB associated with a DEB for a match in the specified line address. The routine searches the DEB Chain until it finds a match.

At this point, no matter which input command format was specified, the DCB is ready to be processed. The Modify Intense routine checks the line DCB for an open condition. If the line is not open, the routine prepares an error message (IED017I), sets a X'04' return code in register 15, and exits to the control module. For an open line, the routine gets the address of the LCB, locates the sense field of the input block, and converts it to hexadecimal representation using a sense conversion table. The routine next gets the sense count, converts it to hexadecimal, and determines whether the result exceeds the maximum allowable limit. If the count is too large, the routine prepares an error message (IED018I), sets a X'04' return code in register 15, and exits to the control module. When the converted sense count is valid, the routine places both it and the converted sense field into a single byte and saves them in the sense field of the Terminal Table entry (TRMSENSE) for a terminal or in the sense field of the LCB (LCBERMSK) for a line. Now the Modify Intense routine sets the checkpoint flag, prepares a response message (IED058I), sets a X'00' return code in regsiter 15, and gives control to the Operator Control control module.

External Routines: None.

Tables/Work Areas: Work areas that contain the fixed portion of each message and space for insertion of the variable data, AVT, DCB, DEB, Terminal Table, Termname Table, LCB, Operator Control AVT.

Attributes: Serially reusable, refreshable, enabled, transient, problem program mode.

Change Interval Type Routine (Chart CZ)

Module Name: IEDQCZ

Entry Point: IEDQCZ - loaded by the Operator Control control module to process MODIFY INTERVAL SYSTEM/POLL commands that have no further operands. The command that caused this routine to be activated is as follows:

```
[control chars]{MODIFY}ident,INTERVAL={SYSTEM[,data]       }
               {  F   }               {POLL,statname,data}
```

Functions: This routine processes operator control commands that request activation of the system or poll delay interval.

The Change Interval Type routine gets the common input block from the location OPCCKELE in the Operator Control AVT, the address of which is in register 1, and checks to determine the type of interval requested in the input ccmmand. The routine uses the following fields in the common input block.

Offset	Field Name	Field Description
+8	OPCTNME	Terminal name for the poll interval
+18	OPCFLG	X'80' - system interval X'40' - poll interval X'20' - a value to be changed.

If the interval is a pcll interval, but nc value is specified to be changed, the Change Interval Type routine prepares an error message (IED018I), sets a X'04' return code in register 15, and returns control to the Operator Control ccntrcl module. If the interval to be changed is the system interval, the routine places the interval specified in a response message (IED047I) and then determines whether the numeric characters are valid. If all of the characters are not valid, the rcutine prepares an error message (IED018I), sets a X'04' return code in register 15, and returns to the control module.

Now the Change Interval Type routine ccnverts the valid system interval value to hexadecimal and determines whether the result is zero or greater than the allowable maximum cf 65,535. If the interval is either, the routine prepares an error message (IED018I), sets a X'04' return code in register 15, and exits to the ccntrol module. The routine checks the AVTHI field in the AVT for zerc, which indicates that the system interval functicn is not supported. If the field is zero, the routine prepares an error message (IED011I), sets a X'04' return code in register 15, and branches back tc the control module. For a positive field, the routine places the new system interval value in the AVTINTLV field of the AVT, sets the checkpoint bit, prepares a response message (IED047I), sets a X'00' return code in register 15, and exits to the ccntrol module.

For a poll interval that is to be changed, the Change Interval Type routine serially searches the Termname Table for an entry that matches the cne specified in the common input block. If no matching entry is found, the routine prepares an errcr message (IED016I), sets a X'04' return code in register 15, and branches to the control module. If an entry is found, the routine places the specified interval in a response message (IED048I) and then checks tc ensure that each character is a valid numeric. If the characters are not valid, the routine prepares an error message (IED018I), sets a return code of X'04', and returns control to the control mcdule. For a valid polling interval, the routine converts the interval to hexadecimal and determines whether it is equal to zero cr greater than 255. If the interval is either, the routine prepares an error message (IED018I), sets a X'04' return code, and gives ccntrcl to the contrcl module. Next, the routine uses the QCB associated with the terminal entry (TRMDESTO) to get the DCB (QCBDCBAD), which is tested for open status. If the line is not open, the Change Interval Type routine prepares an error message (IED061I), sets a X'04' return code in register 15, and returns contrcl tc the ccntrcl module. For an open line, the routine uses the DCB tc locate an LCB for the line and determines whether the line is dial (ICBDIAL switch on in the LCBSTAT2 field). If the line is a dial line, the routine prepares an error message (IED061I), places a X'04' return code in register 15, and exits to the control module. For a nonswitched line, the routine places the polling interval in the DCB (DCBINTLV), sets the checkpoint bit, prepares a response message (IED048I), sets a X'04' return code in register 15, and returns to the control module.

If the system interval is to be activated, the Change Interval Type routine checks the AVTHI field in the AVT, and if it is zero, prepares an error message (IED011I), sets a X'04' return code, and branches to the control module. If the interval value in the AVTINTLV field is zero, the routine performs the same error exit functions.

If the AVTDLAYN bit in the AVTBIT1 field is on, the routine prepares a response message (IED045I), sets a X'00' return code, turns on the checkpoint bit, and returns control to the control module. When both of these fields are not zero, the routine builds a cross-partition tpost parameter list to activate the System Delay subtask (IEDQHI) and the Interval Control module, and issues an AQCTL macro instruction (SVC 102) via the AQCTL SVC 102 routine (IGC102) to post the interval processor. Then the routine prepares a response message (IED093I), sets the checkpoint bit, places a X'00' return code in register 15, and returns control to the Operator Control control module.

External Routine: IGC102 - AQCTL SVC 102 routine - to activate the system interval.

Tables/Work Areas: Work areas that contain the fixed portion of each response and space for insertion of variable data, AVT, Operator Control AVT.

Attributes: Serially reusable, refreshable, enabled, transient, problem program mode.

MCP Closedown Processing Routine (Chart C0)

This routine is discussed in the Termination Routines section of the Program Organization part of this publication.

ICHNG Processing Routine (Chart C1)

Module Name: IEDQC1

Entry Point: IEDQC1 - activated by the Operator Control control module (IEDQCA) when an ICHNG macro has been issued in an application program, the stop line function has been completed, and the LCB has been tposted to the Operator Control QCB by the Stop Line I/O subtask (IEDQHK).

Functions: This module changes a specified invitation list entry when an ICHNG macro is issued in an application program. When an ICHNG macro is issued in an application program, the Operator Control/Application Program Interface routine (IEDQET) builds and tposts a CIB to the Operator Control QCB. This causes the TCAM Dispatcher to relinquish control to the Operator Control control module, which recognizes the CIB and branches to the Stop Line routine (IEDQCV). The Stop Line routine tposts a request to stop the line activity on the associated line to the Stop Line I/O subtask (IEDQHK). The Stop Line I/O subtask performs its functions to stop the line activity and, once this is done, tposts the LCB to the operator control queue. The Stop Line routine regains control and then branches back to the Operator Control control module, which checks the common input block to determine which routine is to receive control. If the OPCVBCD2 field is equal to 1, the Operator Control control module gives control to the ICHNG Processing routine.

The ICHNG Processing routine gets the address of the DCB from the LCB (LCBDCBAD) and then finds the invitation list address from them both. The routine uses the common input block, located at OPCCKELE in the Operator Control AVT, the address of which is in register 1, to determine the type of ICHNG function to be performed. If the command is ICHNG MOVE, the routine calculates invitation list size and moves the data at the move work area to overlay the list. Then the routine removes the LCB from the operator control queue and tposts the LCB to itself in order to start the line via the AQCTL SVC 102 routine (IGC102). The routine sets a X'00' return code in register 15 and returns control to the Operator Control control module.

If the command is ICHNG DEACT and all invitation list entries are already inactive, the ICHNG Processing routine dequeues the LCB and starts line activity by tposting the LCB to itself via the AQCTL SVC 102 routine. For active entries, the routine sets up a count of the number of active invitation list entries. The routine swaps each active entry from the active to the inactive side of the invitation list and decrements the active count until it is zero. When all entries are inactive, the routine sets the number of active entries to zero, dequeues the LCB, and starts the line activity by tposting the LCB to itself via the AQCTL SVC 102 routine.

If the command is ICHNG ACT and all entries are already active, the ICHNG Processing routine dequeues the LCB and starts line activity by using the AQCTL SVC 102 routine as done above. For inactive entries, the routine sets up the count of the number of entries, swaps each entry from the inactive to the active side of the invitation list, and then swaps each one with other active entries until each is in its correct place in the list. The routine decrements the inactive count until it is zero. When all entries have been made active, the routine sets the number of active entries to equal the total entry count. The routine dequeues the LCB, tposts it to itself, and starts the line activity via the AQCTL SVC 102 routine.

External Routine: IGC102 - AQCTL SVC 102 routine - to tpost the LCB to itself in order to start line activity.

Tables/Work Areas: DCB, LCB, AVT, Operator Control AVT.

Attributes: Serially reusable, refreshable, enabled, transient, problem program mode.

On-Line Test Interface Routine (Chart C2)

Module Name: IEDQC2

Entry Point: IEDQC2 - loaded by the Operator Control control module (IEDQCA) to process MODIFY OLT commands. The command that caused this routine to be activated is as follows:

```
[control characters] {MODIFY}ident,OLT=data.
                     {      }
```

Functions: This module processes operator control commands that request Teleprocessing On-Line Test Executive (TOTE) processing. The On-Line Test Interface routine prepares a buffer for the TOTE task and uses the AQCTL SVC 102 routine to tpost the buffer to the TOTE QCB and place it on the ready queue in the MCP.

If, when the On-Line Test Interface routine gains control, the TOTE task is not active in the system, the routine rejects the command by returning to the control module with a X'04' return code and with the address of an error message. When the TOTE task is active, the On-Line Test Interface routine examines the message (the input command) to determine whether it is already canceled by having the operator control characters repeated in the message buffer. If this is the case, the routine returns to the control module with a X'08' return code in register 15. When TOTE is active and the command is not canceled, this interface routine continues by queuing the buffer for processing by the TOTE task.

The On-Line Test Interface routine places X'0C' in the key field of the buffer prefix, places the address of the TOTE QCB in the buffer prefix, and sets the scan pointer to refer to "OLT=" in the buffer if the command is from a terminal. Otherwise, the command is from the system console, and this routine must obtain a buffer in which to pass the operator control command to the TOTE task.

When the operator control command is from the system console, the On-Line Test Interface routine first builds an ERB in the operator control work area and then uses the AQCTL SVC 102 routine to tpost the ERB to the Buffer Request QCB in the MCP. The ERB contains the address of the QCB for this interface routine, so that when the buffer request is satisfied, the buffer units are returned to this operator control routine. The On-Line Test Interface routine calculates the number of buffer units that are necessary by comparing the length of the data portion of the operator control command with the unit length in AVTKEYLE. This number is placed in the ERB unit count field.

Since operator control has a lower priority than the MCP, the ERB with its buffer is on the element chain of the Operator Control QCB when SVC 102 returns control after the tpost. The interface routine locates the appropriate element on the element chain of the Operator Control OCB by scanning the chain for an element that has a X'E4' priority. If, however, the interface routine does not find an element with X'E4' in its priority field, there was not a buffer available in the MCP. In this case, the interface routine issues a WAIT on the Operator Control ECB in the AVT. When the ECB is posted complete, the On-Line Test routine once again scans the Operator Control QCB element chain for an element with a X'E4' priority. This scan-WAIT loop continues until the element is present on the chain. When the On-Line Test Interface routine finds a buffer, the routine puts the operand portion of the operator control command in the buffer. Then the routine puts X'10' in the buffer prefix key field and sets the scan pointer to point to "OLT=".

After the On-Line Test Interface routine has prepared the buffer for TOTE, the routine uses the AQCTL SVC 102 routine to tpost the buffer to the TOTE QCB. The interface routine then sets the successful return code X'14' in register 15 and returns to the control module.

External Routines:

- IGC102 - AQCTL SVC 102 routine - to tpost elements to the ready queue in the MCP.

- OS Wait routine (SVC 1) - to wait for the Operator Control ECB to be posted complete.

Tables/Work Areas: AVT, CIB, buffer prefix, LCB, Operator Control AVT.

Attributes: Reentrant, transient.

Copy Invitation List Status Routine (Chart C3)

Module Name: IEDQC3

Entry Point: IEDQC3 - loaded by the Operator Control control module to process DISPLAY LIST commands. The command that caused this routine to be activated is as follows:

```
[control chars] {DISPLAY}TP,LIST,{ddname,rln}
                {   D   }         {address    }
```

Functions: This routine processes operator control commands that request display of the status field of an invitation list for a specified line.

The Copy Invitation List Status routine first gets the common input block from the OPCCKELE location in the Operator Control AVT, the address of which is in register 1. Now the routine determines from the common input block the line format, either the DDNAME/RLN or hardware address format, that is specified in the input command. If DDNAME/RLN is specified, the routine determines whether the relative line is specified ALL, and if so, rejects the command because status information for a single line only may be displayed. The routine prepares an error message (IED018I), sets a X'04' return code in register 15, and returns control to the Operator Control control module. Otherwise, the routine converts the relative line number to hexadecimal and determines whether the result is zero or greater than 255. For either, the routine rejects the input command, prepares an error message (IED018I), sets a X'04' return code in register 15, and exits to the control module.

If the relative line number is valid, the Copy Invitation List Status routine gets the TCB address from the AVT (AVTTCB) and serially searches the TIOT (at TCB +12) for a ddname that matches the one specified on the input command. If no matching entry is found, the routine prepares an error message (IED017I), sets a X'04' return code, and branches to the control module. When a matching entry is found, the routine saves the entry offset into the TIOT.

For either DDNAME/RLN or address format, the Copy Invitation List Status routine gets the address of the beginning of the DEB chain from the TCB (TCB+8) and checks each associated DCB (DEBDCBAD) to determine whether it is for a TCAM line (DSORG is X'40'). If the routine does not find an associated DCB in the DEB chain, the routine prepares an error message (IED017I), sets a X'04' return code, and returns control to the control module.

For the DDNAME/RLN command format, once the DCB has been located, the Copy Invitation List Status routine compares the DCB offset into the TIOT (DCBTIOT) to the offset calculated from the TCB mentioned above, and if the offsets do not match, the routine locates the next DEB and continues to search as just described. When the correct DCB is found, the routine checks the UCB for an OPEN DD DUMMY condition. If this condition has occurred, the routine prepares an error message (IED017I), sets a return code of X'04' in register 15, and exits to the control module.

For the address format, the Copy Invitation List Status routine tests each UCB associated with the DEB to find a match for the one specified in the input command. The routine searches the DEB chain until a matching entry is found.

At this point, now that the DCB has been found, the Copy Invitation List Status routine compares the relative line number, the one provided in the DDNAME/RLN format or the one calculated for the address format, to the number of lines in the line group. If the relative line number is greater, the routine prepares an error message (IED017I), places a X'04' return code in register 15, and branches to the control module. Next the routine determines whether the line is open, and if not, prepares the same error message and return code and exits to the control module. If the line is open, the routine gets address of the invitation list for the line from the DCB (DCBINVLI), converts the status byte values to printable characters and places them in a response message (IED059I). The routine now sets a X'00' return code in register 15 and returns control to the Operator Control control module.

External Routines: None.

Tables/Work Areas: Work areas that contain the fixed portion of each response message and space for insertion of variable data, AVT, DEB, DCB, Operator Control AVT.

Attributes: Serially reusable, refreshable, enabled, transient, problem program mode.

Operator Control Work Area CSECT (No Flowchart)

Module Name: IEDQC5

Entry Point: Not applicable.

Functions: The Operator Control work area is a non-executable work area used by the Operator Control control module to prepare long messages and by the Modify Options routine (IEDQCF), the Copy Held Terminals routine (IEDQCK), and the Copy Operator Control Terminal routine (IEDOCM) as a conversion area for the MODIFY and DISPLAY option functions.

External Routines: None.

Tables/Work Areas: None.

Attributes: Non-executable, refreshable, transient, problem program mode.

DEBUG Service Aid Router (Chart C6)

Module Name: IEDQC6

Entry Point: IEDQC6 - activated by the Operator Control control module (IGC0110D) to process operator control commands that request loading or deleting a service aid. The command that caused this routine to be activated is as follows:

```
[control chars] {MODIFY} ident,DEBUG=data
                {  F   }
```

Functions: This module processes operator control commands that request the loading or deleting of service aid modules. If the operator control command is valid, the DEBUG Service Aid Router returns a response message after executing the service aid routine that either was loaded or is to be deleted.

The DEBUG Service Aid Router determines whether the COMWRITE routine is present in the system. If COMWRITE is not in the system or if a restart is in progress, the Router generates a message (IED107I) and returns control to the calling routine. If COMWRITE is in the system and a restart is not in progress, the Router passes control to the scan function (IEDQCA02) of the Resident Operator Control module to get the first DEBUG operand. If the operand is not a load or a delete request, the Router generates a message (IED107I) and returns to the calling routine. If the operand is a load or a delete request, the Router again uses the scan function of the Resident Operator Control module, this time to obtain the second DEBUG operand. If this operand is a valid name, the Router executes the requested function; otherwise the Router generates a message (IED107I) and returns to the calling routine.

The DEBUG Service Aid Router finds the address of the service aid routine by issuing a BLDL macro. This routine uses the resulting BLDL list to determine whether the requested service aid routine exists in either SYS1.LINKLIB or JOB/STEPLIB. If the service aid routine does not exist, the Router prepares the IED102I error message and returns to its calling routine.

When the operator control command requests that a service aid routine be loaded, the DEBUG Service Aid Router checks the load list to determine the current status of the service aid routine; that is, whether the routine is active or inactive in the system. If the routine is already active, the DEBUG Service Aid Router returns control to the calling routine with a response message (IED103I). If the service aid routine is not currently active, the DEBUG Service Aid Router issues a conditional GETMAIN, based on the module length specified in the BLDL list, to determine whether there is enough main storage available to load the routine. If main storage is not available, the Router tests to determine how much more storage is needed to load the service aid and prepares a message to inform the operator of the problem. If main storage is available, the Router loads and gives control to the service aid routine. Upon return from the service aid routine, the DEBUG Service Aid Router checks the return code. If the return code indicates successful initialization of the service aid, the Router returns control to the calling routine with an appropriate response message (IED099I). If the return code indicates an unsuccessful load, the DEBUG Service Aid Router prepares a message (IED105I or IED106I), deletes the service aid, and passes control to the calling routine.

When the operator control command requests the deletion of a service aid routine, again the DEBUG Service Aid Router checks the load list for the status of the service aid. If that routine is active in the system, the DEBUG Service Aid Router passes ccntrol to the service aid routine. Upon return, the DFBUG Service Aid Router checks the return code; if the code is good, the Router prepares a deactivation message (IED100I), deletes the service aid routine, and passes control to the calling routine. If the load list shows that the service aid routine is not active, the DFBUG Service Aid Router prepares a message (IED104I) and returns control to the calling routine.

External Routines:

- IEDQCA - Resident Operator Control module - the Operatcr Control Scan subrcutine (IEDQCA02), to serially search the input command for the service aid name.

- FE Service Aid routine - to cause activaticn or deactivation of the service aid functicns.

Tables/Work Areas: AVT, Operator Control AVT, work areas that contain the fixed portion of each response message and space for insertion of the variatle data.

Attributes: Problem program mode.

CHECKPOINT ROUTINES

Checkpoint Executcr (Chart NF)

Module Name: IEDQNF

Entry Point: IEDQNF - gains control when the Checkpoint task is activated, or when a checkpoint routine completes its activity.

Functions: This routine determines whether there is anything that needs to be dcne by the Checkpoint task and which module should be loaded to perform the required function, if any.

If the disk IOB is marked complete, the Checkpoint Executor loads and gives control to the Checkpcint Notification and Disposition routine. If no I/O is in progress and there is a record on the Checkpoint Disk I/O queue, the Checkpcint Executor loads and gives control to the Checkpoint Disk I/O routine.

If there is a request element on the Checkpoint QCB, the Checkpoint Executor loads and gives ccntrcl to the routine to build the appropriate checkpoint record.

If there is no checkpoint functicn tc be performed, this module waits on the ECB in its QCB and its I/O ECB, unless the clcsedown completion bit in the environment checkpoint request element is on. In this case, the Checkpoint Executor returns to OS, thus terminating the attached Checkpoint task.

If the Checkpoint Executor is activated by the return of another checkpoint routine, it can perform the additional function of immediately activating a routine as requested by the returning routine. If the returning routine branches to the address in register 14, the Checkpoint Executor deletes the returning routine and immediately loads the one with the offset in register 15. If the returning routine branches to the address in register 14+4, the Checkpoint Executor deletes the returning module and immediately begins performing its regular functions. If the returning routine branches to the address in register 14+8, the Checkpoint Executor deletes the returning module and waits for the I/O to complete before resuming activity.

External Routines:

- IEDQNG - Incident Checkpoint for MH routine - to build an incident checkpoint record when a CHECKPT macro is issued in an MH.

- IEDQNH - Incident Checkpoint for TCHNG routine - to build an incident checkpoint record for a TCHNG macro.

- IEDQNJ - Incident Checkpoint for Operator Control routine - to build an incident checkpoint record for an operator control command.

- IEDQNK - Environment Checkpoint routine - to build an environment checkpoint record.

- IEDQNM - Build CKREQ Disk Record routine - to build a CKREQ checkpoint record.

- IEDQNO - Checkpoint Queue Manager - to manage the Checkpoint I/O queue.

- IEDQNP - Checkpoint Disk I/O routine - to write checkpoint records on disk.

- IEDQNQ - Checkpoint Notification and Disposition routine - to issue FREEMAIN macros and notify completion of a checkpoint.

- IEDQNR - No Available Core routine - to handle an insufficient main storage situation.

- IEDQNS - No Incident Records routine - to handle an incident record overflow situation.

- OS Wait routine (SVC 1) - to wait for an ECB to be posted complete.

- OS Load routine (SVC 8) - to load a module into main storage

- OS Delete routine (SVC 9) - to remove a module from main storage.

Tables/Work Areas: CVT, AVT, checkpoint work area.

Attributes: Reentrant, resident.

Environment Checkpoint Routine (Chart NK)

Module Name: IEDQNK

Entry Point: IEDQNK - loaded by the Checkpoint Executor to build an environment checkpoint record.

Functions: This module builds environment checkpoint record segments for disk. The Environment Checkpoint routine examines the current EXCP field (CKPEXCP) in the checkpoint work area to determine whether to build the first segment or a subsequent segment of an environment checkpoint. If the key field of the record pointed to by CKPEXCP contains the value X'20', a subsequent segment is to be built. In this case, the Environment Checkpoint routine picks up its register values from CKPSAVE1 and builds the next checkpoint segment in the GETMAIN area pointed to by CKPEXCP. Otherwise, the Environment Checkpoint routine issues a GETMAIN macro for an area in which to build a new segment and places the address of the area in the "last record built" field (CKPLDRB) in the checkpoint work area. For a first (or only) segment, this routine also turns off all request bits and turns on the "checkpoint in progress" flag in the Environment Checkpoint Request element (AVTCKELE).

Before moving a group of data from the MCP tables into a disk record, the Environment Checkpoint routine determines whether there is room for all the data in this segment. If not, the routine puts X'20' in the key field, saves registers in CKPSAVE1, moves as much data as possible into the segment, and returns to the Checkpoint Executor. If the routine reaches the end of the data in the MCP tables before filling the record segment, it places X'1C' in the key field and returns to the Checkpoint Executor without saving registers.

If the Environment Checkpoint routine has just built the first (or only) segment of an environment checkpoint, it returns to the Checkpoint Executor with the offset of the Checkpoint Queue Manager in register 15. If the routine has just built a subsequent segment, it returns to the Checkpoint Executor with the offset of the Checkpoint Disk I/O routine in register 15. This is because only the first segment of an environment checkpoint is placed on the Checkpoint Disk I/O queue.

The only error condition that applies to this routine occurs if the GETMAIN request for space in which to build a segment cannot be satisfied. In this case, the Environment Checkpoint routine returns to the Checkpoint Executor with the offset of the No Available Core routine in register 15.

External Routines: None.

Tables/Work Areas: AVT, checkpoint work area, Option Table, Termname Table, Terminal Table, QCB, invitation list.

Attributes: Reentrant, transient.

Checkpoint Queue Manager (Chart NC)

Module Name: IEDQNO

Entry Point: IEDQNO - loaded by the Checkpoint Executor to manage the checkpoint I/O queue.

Functions: This routine puts disk records on the checkpoint I/O queue and updates the last request element for which a disk record was built. When it enqueues an environment record segment, the Checkpoint Queue Manager dequeues all incident records and issues a FREEMAIN for each one. As a result, they are not written on the disk. An "incident overflow" bit in the incident request element is turned on to indicate that the request will be satisfied when the new environment record(s) is written.

The Checkpoint Queue Manager returns to the Checkpoint Executor at the register 14+4 entry point.

External Routines: None.

Tables/Work Areas: AVT, checkpoint work area.

Attributes: Reentrant, transient.

Checkpoint Disk I/O Routine (Chart NP)

Module Name: IEDQNP

Entry Point: IEDQNP - loaded by the Checkpoint Executor to write checkpoint records on disk.

Functions: This routine locates the next disk record to be written, determines the proper TTR for the record, and issues an EXCP to write the record.

If there is a record in the current EXCP field (CKPEXCP) of the checkpoint work area, it is the record written and it is a continuation of a checkpoint that requires more than one segment. If CKPEXCP is equal to zero, the first record on the Checkpoint Disk I/O queue (CKPIOOF) is the one just written. The Checkpoint Disk I/O routine removes this record from the Checkpoint Disk I/O queue and places it in CKPEXCP. If the record is an environment or incident record, the rcutine uses the TIME macro to put the date and time into the record. If the record is an environment record, this routine moves the TTR of the last incident record used from the ccntrcl record to the environment record.

The methcd used to determine the correct TTR depends on the type of record to be written:

- First segment of an environment checkpoint - the control record has the TTR of all first segments and an index to the latest one used. This routine picks up the TTR that sequentially follows the latest one, and changes the index tc point to the new first segment.

- Any environment segment other than the first - the checkpcint work area contains the TTR of the last segment written. This module determines the TTR of the next sequential reccrd cn disk.

- Incident record - the checkpoint work area contains the TTR of the last incident record written. This module determines the TTR of the next sequential record on disk.

- CKREQ record - the checkpoint work area contains a CKREQ-TTR table that associates a terminal name offset with a particular TTR. This module uses the terminal name offset in the disk record to locate the proper TTR in the table.

If there is no TTR available fcr envircnment segments or CKREQ records because all the records have disk I/O errors, this routine issues an error message via WTO.

The Checkpcint Disk I/O routine returns to the Checkpoint Executor at the register 14+4 entry point.

External Routine: IECPCNVT - an OS routine tc ccnvert the relative TTR to an absolute disk address.

Tables/Work Areas: AVT, checkpcint wcrk area, Termname Table, DCB, DEB, CVT.

Attributes: Reentrant, transient.

Checkpoint Notification and Dispositicn Routine (Chart NQ)

Module Name: IEDQNQ

Entry Point: IEDQNQ - loaded by the Checkpcint Executcr to issue FREEMAIN macros and to notify completion of a checkpcint.

Functions: This routine gets control after a disk write operation completes or after a checkpoint could not be satisfied. It removes the checkpoint request element(s) from the QCB chain and tposts the element (if from an MH macro) or posts an ECB (if from Operator Control or an application program). The request element is not removed if the request has not been completely satisfied or if a disk error occurred during the write operation.

If the last segment of an environment checkpoint was just written with an incident overflow condition (indicated in the environment request element), several incident request elements may be removed from the QCB chain. The "incident overflow" bit in each incident request element (bit 0 of the key) and in the environment request element is turned off.

If the last segment of a checkpoint was just written, this routine issues a FREEMAIN macro for the record.

If the last segment of an environment checkpoint was just written, this routine turns on a bit in each PCB (bit 2 of PCBOFIG) to indicate to the application program(s) that the checkpoint was taken, and tposts an element to the ready queue. If the request was from an MCPCLOSE macro, the element is the closedown completion element; otherwise, it is the environment checkpoint request element to be placed on the time delay queue.

If this routine recognizes a record with a disk error, it issues an error message via WTO. If the record with the disk error is a CKREQ or environment record, the routine flags the record in the checkpoint work area, and branches to the address in register 14 with the offset of the Checkpoint Disk I/O routine in register 15. In this way the same record can be written at another location on the disk.

If a checkpoint request was not completely satisfied, this routine places the offset for the module that builds the particular checkpoint in progress in register 15 and returns to the Checkpoint Executor at the register 14 entry point.

If the checkpoint request was completely satisfied, this routine returns to the address in register 14+4.

External Routine: IGC102 - AQCTL SVC 102 routine - to tpost elements to the ready queue and to post ECBs for the application programs.

Tables/Work Areas: AVT, checkpoint work area.

Attributes: Reentrant, transient.

Checkpoint Disk End Appendage (Chart RA)

Module Name: IGG019RA - activated by IOS at the end of a checkpoint disk operation.

Functions: This routine writes the checkpoint control record after the last segment of an environment checkpoint record is written on the disk.

When this module writes a control record, it branches to the address in register 14+8. It writes the control record using retry.

When this module is not writing a control record, it branches to the address in register 14.

External Routines: None.

Tables/Work Areas: DEB, checkpoint work area.

Attributes: Reentrant, resident, supervisor mode.

Build Incident Record for MH Routine (Chart NG)

Module Name: IEDQNG

Entry Point: IEDQNG - loaded by the Checkpoint Executor to write an incident checkpoint record for a CHECKPT macro in an MH.

Functions: This routine builds an incident disk record when the request element on the Checkpoint QCB is an LCB from an MH macro.

This routine returns to the address in register 14 with the offset for either the Checkpoint Queue Manager, the No Incident Records routine, or the No Available Core routine in register 15.

External Routine: IEDQTNT - Termname Table code - to obtain the Terminal Table entry address.

Tables/Work Areas: AVT, checkpoint work area, Termname Table, Terminal Table, Option Table.

Attributes: Reentrant, transient.

Application Program/Checkpoint Interface Routine (Chart NB)

Module Name: IEDQNB

Entry Points: This routine is called when an application program issues a TCAM macro that changes the TCAM environment. The entry point to the routine depends on which macro causes the routine to be activated:

- IEDQNB - CKREQ macro.

- IEDQNB02 - TCHNG macro.

- IEDQNB05 - OPEN or CLOSE macro.

Functions: The purpose of this routine is to build a checkpoint request element and tpost it to the MCP ready queue when an application program issues a TCAM macro that changes the MCP environment. After tposting the request element, the Application Program/Checkpoint Interface routine issues a WAIT command to allow the Checkpoint task to gain control to process the element. The request element built by this routine indicates which macro issued the request.

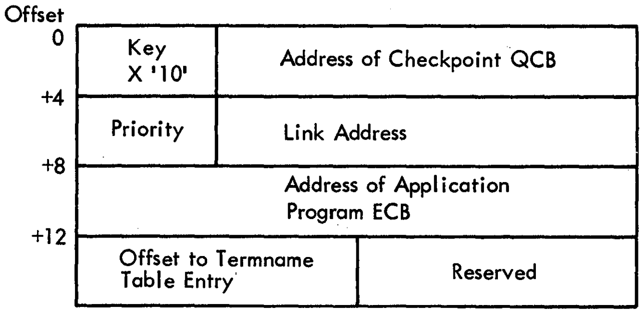
When an OPEN or CLOSE macro is issued in an application program, the Application Program/Checkpoint Interface routine determines whether an entry in the CKREQ-TTR Table is involved. If there is an entry involved, this routine inverts the status of the CKREQ-TTR entry. For example, if a Destination QCB that can be checkpointed as the result of a CKREQ macro is opened, its entry in the CKREQ-TTR Table is made active; if closed, its entry is made inactive. (Inactive entries can be used for other Destination QCBs that are opened later.) If SYNC=YES is specified for the Destination QCB (TPROCESS macro), the QCB can be checkpointed; therefore, it is given an entry in the CKREQ-TTR Table.

The Application Program/Checkpoint Interface routine builds its checkpoint request element in the Process Control Block (PCB). The formats of this element are indicated below according to entry point:

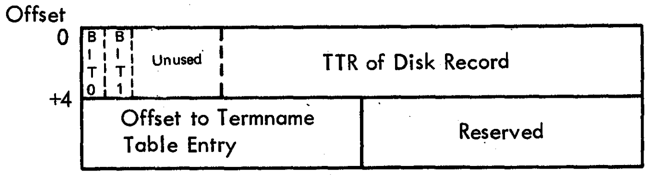
- IEDQNB - request by a CKREQ macro.

Offset		+1
0	Key X '60'	Address of Checkpoint QCB
+4	Priority	Link Address
+8		Address of Application Program ECB
+12		Address of Application Program DEB chain

- IEDQNB02 - request by a TCHNG macro


Offset
0 Key X '10' | Address of Checkpoint QCB
+4 Priority | Link Address
+8 Address of Application Program ECB
+12 Offset to Termname Table Entry | Reserved

- IEDQNB05 - request by OPEN or CLOSE macro - inverts the first bit of the CKREQ-TTR Table entry:


Offset
0 BIT0 | BIT1 | Unused | TTR of Disk Record
+4 Offset to Termname Table Entry | Reserved

Byte 0, bit 0: ON- Entry is Active
OFF- Entry is Inactive
Byte 0, bit 1: ON- Entry has a Disk Error

External Routines:

- IEDQTNT - Termname Table code - to determine the terminal entry address.

- IGC102 - AQCTL SVC 102 routine - to post the checkpoint request element to the MCP ready queue and to invert the status bit in the CKREQ-TTR Table.

Tables/Work Areas: AVT, checkpoint work area, DCB, PCB, application program DEB, LCB, access method work area, Termname Table, Terminal Table.

Attributes: Reentrant, transient.

Build Incident Record for TCHNG Routine (Chart NH)

Module Name: IEDQNH

Entry Point: IEDQNH - called by the Checkpoint Executor when the request element on the Checkpoint QCB is from a TCHNG macro in an application program.

Functions: This routine builds an incident checkpoint disk record when the request element on the Checkpoint QCB is from a TCHNG macro in an application program. The format of the checkpoint request element is shown under IEDQNB02 in the Application Program/Checkpoint Interface routine discussion.

This routine builds the incident checkpoint record in a GETMAIN area and stores its address in the CKPLDRB field of the checkpoint work area.

The Build Incident Record for TCHNG routine returns the address in register 14 with the offset for the Checkpoint Queue Manager, the No Incident Records routine, or the No Available Core routine in register 15.

External Routine: IEDQTNT - Termname Table code - to determine the Terminal Table entry address.

Tables/Work Areas: AVT, checkpoint work area, Termname Table, Terminal Table, Option Table.

Attributes: Reentrant, transient.

Incident Checkpoint for Operator Control Routine (Chart NJ)

Module Name: IEDQNJ

Entry Point: IEDQNJ - called by the Checkpoint Executor when the request element on the Checkpoint QCB is from an operator control command.

Functions: This routine builds an incident checkpoint disk record when the request element on the Checkpoint QCB is from an operator control command. The request element is pointed to by register 3. The AVT contains the address of the operator control work area, which contains the operator control command block.

When the operator control command is for Stop or Start Line, the routine ensures that the DDNAME is present in the data and converts all the unit addresses to DDNAME and relative line number.

This routine builds the incident checkpoint record in a GETMAIN area and stores its address in the CKPLDRB field of the checkpoint work area. The record is a form of the operator control command itself, rather than the tables that are changed as a result of the command.

The Incident Checkpoint for Operator Control routine exits to the address in register 14, with the offset for the Checkpoint Queue Manager, the No Incident Records routine, or the No Available Core routine in register 15.

External Routines: None.

Tables/Work Areas: Checkpoint work area, AVT, Operator Control work area.

Attributes: Reentrant, transient.

Build CKREQ Disk Record Routine (Chart NM)

Module Name: IEDQNM

Entry Point: IEDQNM - loaded by the Checkpoint Executor when the request element on the Checkpoint QCB is from a CKREQ macro in an application program.

Function: This routine builds a CKREQ checkpoint disk record for each of the opened Destination QCBs in the MCP that are associated with the application program issuing the CKREQ macro. The format of the request element is shown under IEDQNB in the Application Program/Checkpoint Interface routine discussion.

This routine builds the CKREQ record in a GETMAIN area. If the record is the first one built for a particular request, the routine places its address in the CKPLDRB field in the checkpoint work area; otherwise, the address is stored in the CKPEXCP field in the checkpoint work area. One CKREQ macro may result in more than one CKREQ record, but each entry into this routine results in only one record.

The Build CKREQ Disk Record exits to the address in register 14, with the offset for the Checkpoint Queue Manager or the No Available Core routine in register 15.

External Routine: IEDQTNT - Termname Table code - to determine the Terminal Table entry address.

Tables/Work Areas: AVT, checkpoint work area, DEB, Termname Table, Terminal Table, QCB, Option Table.

Attributes: Reentrant, transient.

Checkpoint - No Available Core Routine (Chart NR)

Module Name: IEDQNR

Entry Point: IEDQNR - loaded by the Checkpoint Executcr when a conditional GETMAIN for an area in which tc build a checkpoint record cannot be satisfied.

Functions: This routine handles the situation in which a conditional GETMAIN for a checkpoint record cannot be satisfied. The Nc Available Core routine first checks for other GETMAIN records on the Checkpoint Disk I/O queue. If there are GETMAIN records there, this routine exits to the Checkpoint Executor (register 14+8) to allow time for these records tc be processed and freed.

If there are no outstanding GETMAIN records, the Nc Available Core routine converts the length of the GETMAIN request, builds an error message to be issued via WTO, and indicates that no disk record was built for this request element. It then exits to the address in register 14 with the offset fcr the Nctification and Disposition routine in register 15.

External Routines: None.

Tables/Work Areas: AVT, checkpoint work area, Termname Table.

Attributes: Reentrant, transient.

Checkpoint - No Incident Records Routine (Chart NS)

Module Name: IEDQNS

Entry Points: IEDQNS - loaded by the Checkpoint Executor when all the incident disk reccrds on the checkpcint data set have been used.

Functions: This routine causes an environment checkpcint to be taken when all the incident disk records on the checkpcint data set have been used.

The No Incident Records routine removes the envircnment checkpoint request element from its queue (either the time delay queue or the Checkpoint QCB). It then examines the Checkpoint QCB to locate the last request element for which a disk record was built and inserts the environment checkpcint request element intc the next pcsition in the element chain of the Checkpoint QCB. This causes the environment request element tc be the next one prccessed, so the incident records on disk can be overlaid.

The No Incident Records routine exits to the Checkpcint Executor (the address in register 14 +4).

External Routines:

- IEDQHG03 - Time Delay routine - tc remove the environment checkpoint request element from the time delay queue.

- IGC102 - AQCTL SVC 102 routine - to tpost an element to the MCP ready queue tc activate IEDQHG03.

Tables/Work Areas: AVT, checkpoint work area.

Attributes: Peentrant, transient.

ERROR RECOVERY PRCCEDURE ROUTINES

Start-Stop ERP Control Module (Chart JC)

Module Name: IGE0004G

Entry Point: IGE0004G - activated by the I/O Supervisor (ICS) when an error is detected on a start-stop line, when an interrupt occurs on I/O that was initiated by an ERP mcdule, and when end of day recording is requested.

Functions: This module transfers contrcl tc the appropriate ERP module to prccess the specific error condition that cccurred on a particular CCW. The Start-Stop ERP Contrcl module receives control from IOS when the Line End Appendage returns tc IOS with a line error condition, when an interrupt occurs on an I/O operaticn that was initiated by an ERP module, and when end of day recording is requested. This control module transfers control to one of the following modules according to the ccnditicn by which the control module was activated:

- Read/Write Unit Check and Unit Exception ERP module (IGE0104G) - activated by the ccntrol module tc prccess a unit exception on a write or write break CCW and to process a unit check (equipment check, lost data, time-out, bus-cut check, cr intervention required) cn a read CCW.

- Non-operational Control Unit ERP module (IGE0204G) - activated by the ccntrol module when a ccntrcl unit is nct operational. This is indicated by the condition code 3 after a Start I/O command.

- Unit Check for Non-read, Non-write, and Non-poll CCWs module (IGE0304G) - activated by the contrcl module when a unit check or a unit exception error occurs on a CCW that is not a read, a write, or a pcll operation.

- Auto Poll and Read Response to Poll Unit Check and Unit Exception ERP module (IGE0404G) - activated by the control module to process a unit exception or a unit check cn a pcll cr read response to poll CCW.

- Error Post and Second Level CCW Return module (IGE0504G) - activated by the control module under five different situations:

  1. An interrupt on an I/O operation that was initiated by an ERP module.

  2. An attention, status modifier, control unit end, or busy condition indicated by the control unit. In this case, the control module sets an error flag in the LCB before activating the ERP processing routine.

  3. A program check, protection check, or chaining check error on the line. In this situation, the control module sets an error flag in the LCB before activating IGE0504G.

  4. Any unit exception that is not handled by the Auto Poll and Read Response to Poll Unit Check and Unit Exeception ERP module, the Read/Write Unit Check and Unit Exception ERP module, the Unit Check and Unit Exception on Read/Write CCWs for Audio and 2260 Local Devices ERP module, and the Unit Check Module for Non-read, Non-write, and Non-poll CCWs ERP module; and is not an overrun or data check error on read and write text CCW.

  5. Any retriable errors on which the retry count is exhausted without successful recovery.

- Unit Check and Unit Exception on Read/Write CCWs for Audio and 2260 Local Devices ERP module (IGE0604G) - activated by the control module to process errors detected on audio and local devices.

- Start-Stop Channel Check ERP module (IGE0804G) - activated by the control module to process channel control check ending status, interface control check ending status, and channel data check ending status.

- Closedown Terminal Statistics Recording module (IGE0904G) - activated by the control module when end of day recording is requested.

- OS OBR/SDR module (IGE0025F) - activated by the control module when recovery from an error has been successful and one of the following conditions exist:

  1. The SIO or error counter is about to overflow, or

  2. The user has requested logging of temperary errors.

- Line End Appendage (IGG019R0) - activated by the control module under the two following conditions:

  1. When error recovery was successful and updating of terminal statistics is not required.

2. When overrun or data check errors occur on read or write text CCWs. These errors are nct retried ty an FRP module, but may be retried in the MCP.

External Routine: IEDQTNT - Termname Table code - to obtain a terminal entry address.

Tables/Work Areas: LCB, CCW, SCB, AVT, DCB, Terminal Table entry.

Attributes: Superviscr mode, disabled, transient.

Read/Write Unit Check and Unit Exception ERP Module (Chart JD)

Module Name: IGE0104G

Entry Point: IGE0104G - activated by the Start-Stop ERP Control module (IGE0004G) to process read/write unit check and unit exception error conditions.

Functions: This module processes read/write unit check and unit exception errcr conditions that occur on start-stop lines.

If a unit exception cccurs, the action that this ERP module takes depends on the device on which the error occurs:

- Teletype adapter - the ERP module executes a Write Break CCW.
- 2701 - the ERP module executes a Read Skip CCW.
- All other start-stcp adapters - the ERP re-executes the CCW on which the unit excepticn cccurred until the retry count is exhausted. At this point, a permanent error exists and this module transfers ccntrol tc the Errcr Post and Second Level CCW Return module (IGE0504G).

If the Read/Write Unit Check and Unit Exception module receives control after a unit check occurs, this ERP module analyzes the sense data in the ICB. If the error is eligible for retry, this ERP module restarts the channel program; otherwise, the error is permanent. This ERP module considers as permanent errors any control unit errors, such as equipment checks, and any non-text errors with exhausted retry counts. This ERP module does not process text errors, but returns them to the Line End Appendage for possible retry in the MCP.

External Routines: Ncne.

Tables/Work Areas: CCW, IOB, LCB, SCB, AVT, UCB.

Attributes: Superviscr mode, disabled, transient.

Non-operational Control Unit ERP Module (Chart JE)

Module Name: IGE0204G

Entry Point: IGE0204G - activated by the Start-Stop ERP Control module (IGE0004G) when a control unit is not operational.

Functions: This module informs the system operator that a specific control unit is not operational. This module issues a Write to Operator (WTO) macro, which writes the message, IED064I LINE addr CONTROL UNIT NOT OPERATIONAL. The module indicates a permanent error condition by setting a flag in the LCB. The module then exits to Line End Appendage.

External Routine: OS Write to Operator routine - to write a message on the system console.

Table/Work Areas: LCB.

Attributes: Supervisor mode, disabled, transient.

Unit Check for Non-read, Non-write, and Non-poll CCWs ERP Module (Chart JF)

Module Name: IGE0304G

Entry Point: IGE0304G - activated by the Start-Stop ERP Control module to process errors from non-read, non-write, and non-poll CCWs.

Functions: This module processes unit checks for failing CCWs that are not a read, a write, or a poll operation. This ERP module uses the sense data that is stored in the IOB to determine the action to be taken:

- Retries the CCW twice for lost data and for a bus-out check on a dial command. If retry is unsuccessful, the error is permanent; therefore, this ERP module transfers control to the Error Post and Second Level CCW Return module.

- Retries the CCW twice for a time-out on a dial, a disable, an enable, or a prepare command. If retry is unsuccessful, the error is permanent; therefore, this ERP module transfers control to the Error Post and Second Level CCW Return module.

- Retries the CCW twice for an intervention required on a dial or a prepare command. If retry is unsuccessful, the error is permanent; therefore, this ERP module transfers control to the Error Post and Second Level CCW Return module.

- Transfers control to the Error Post and Second Level CCW Return module (IGE0504G) to handle all the other errors, which either logically should not have occurred or are permanent errors.

External Routines: None.

Tables/Work Areas: CCW, IOB, LCB, SCB.

Attributes: Supervisor mode, disabled, transient.

Auto Poll and Read Response to Poll Unit Check and Unit Exception ERP Module (Chart JG)

Module Name: IGE0404G

Entry Point: IGE0404G - activated by the Start-Stop ERP Control module (IGE0004G) or the BSC ERP Control Module (IGE0004H) when an error is detected on a poll or a read response to a poll command.

Functions: This module processes unit checks and unit exceptions for poll CCWs and read response to poll CCWs.

If a unit exception occurs on a poll CCW, this ERP module re-executes the CCW. If a unit check occurs on a poll CCW and the error is a time-out, data check, or intervention required, this ERP module updates the poll pointer and retries the channel program. When the retry count is exhausted, the error is assumed to be permanent. All other unit check error conditions on a poll CCW are considered permanent.

If a unit check occurs on a read response to poll CCW and the error is a time-out, data check, or intervention required, this ERP module updates the poll pointer and restarts the channel program. When the retry count is exhausted, the error is assumed to be permanent. The module retries overrun and lost data errors by restarting the channel program at the read response command. All other unit checks on a read response to poll CCW are permanent.

A unit exception on a read response to poll CCW is handled by the Line End Appendage.

When a permanent error condition is detected, this ERP module transfers control to the Error Post and Second Level CCW Return module (IGE0504G).

External Routines: None.

Tables/Work Areas: CCW, LCB.

Attributes: Supervisor mode, disabled, transient.

Error Post and Second Level CCW Return Module (Chart JH)

Module Name: IGE0504G

Entry point: IGE0504G - activated by either the Start-Stop ERP Control module or other ERP processing modules.

Functions: This module attempts to retry channel programs and handles permanent error situations.

The Error Post and Second Level CCW Return module receives control from the Start-Stop ERP Control module when a special return indicator is set and an interrupt occurs on a Read Skip or Write Break CCW that was issued by an ERP module. In this situation, the Error Post and Second Level CCW Return module attempts to retry the user's channel program.

The Error Post and Second Level CCW Return module receives control from other ERP processing modules when a permanent error is detected. The ERP processing modules pass to this module both permanent errors that are not retried and errors that are considered permanent only after the retry count is exhausted. When a permanent error condition is passed to the Error Post and Second Level CCW Return module, it passes control to either the OS Message Writer, the Line End Appendage, or the OS OBR/SDR module.

If OBR recording is required and the system console is not the primary operator control terminal, control passes to the OBR/SDR module. If OBR recording is not required and the system console is not the primary operator control terminal, control passes to the Line End Appendage. If the system console is to primary operator control terminal, the Error Post and Second Level CCW Return module exits to the OS Message Writer, which either writes an error message on the system console or routes an error message to an alternate operator control terminal by returning to Line End Appendage. If OBR recording is required, the Error Post and Second Level CCW Return module places X'01' in LCBFLAGS before exiting to the OS Message Writer. This flag indicates that the Message Writer should pass control to the OBR/SDR module (IGE0025F).

External Routine: IEDQTNT - Termname Table code - to obtain a terminal entry address.

Tables/work Areas: CCW, LCB, AVT, Terminal Table entry, SCB.

Attributes: Supervisor mode, disabled, transient.

Unit Check and Unit Exception on Read/Write CCWs for Audio and 2260 Local Devices ERP Module (Chart JI)

Module Name: IGE0604G

Entry Point: IGE0604G - activated by the Start/Stop ERP Control module (IGE0004G) to process errors on audio and local devices.

Functions: This mcdule adjusts the retry count and retries the failinq CCW sequence when IOS detects an errcr cn an audio or local device. If the error is undefined or if the retry count is exhausted, this ERP module exits to the Error Post and Second Level CCW Return module, which records the error.

External Routines: None.

Tables/Work Areas: LCB, SCB, CCW.

Attributes: Supervisor mode, disabled, transient.

Start-Stop Channel Check ERP Module (Chart JJ)

Module Name: IGE0804G

Entry Point: IGE0804G - activated by the Start-Stop ERP Control module (IGE0004G) to process channel contrcl check endinq status, interface control check endinq status, and channel data check ending status.

Functions: This module processes channel endinq status errors that are detected by ICS.

The Channel Check Handler is an cptional extension of IOS for configurations that use the 2860/2870 channels. The Channel Check Handler determines whether a channel ccntrol check or an interface control check is recoverable. If so, this handler builds an ERP interface byte (ERPIB) to provide the Start-Stop Channel Check ERP module the infcrmation fcr a possible retry.

When the Start-Stop Channel Check ERP mcdule gains ccntrcl after a channel contrcl check or an interface ccntrol check, it searches a list of ERPIBs to determine whether IOS has supplied preliminary parameters for retry. If the module does not find an ERPIB for the failinq device, the module considers the errcr to be permanent and passes control to the Error Post and Second Level CCW Return module (IGE0504G). If an ERPIB is found for the device, the Start-Stop Channel Check ERP module tests the retry flaq in the ERPIB. If this flaq is on, the mcdule ccnsiders the errcr to be permanent and exits to the Error Post and Second Level CCW Return module. If the flag is not on, the Start-Stop Channel Check ERP module saves the ERPIB data in a work area, clears the ERPIB tc zero to make space for system sense informaticn, and attempts a retry procedure that is based on the ERPIB data, the failinq CCW, and the retry count. The module continues the retry procedure until the either the retry is successful or the retry ccunt is exhausted. If the retry count is exhausted, this module transfers control to the Error Post and Second Level CCW Return module.

When the Start-Stop Channel Check ERP module gains control after a channel data check, the module attempts tc retry the failinq CCW.

If retry is unsuccessful and the retry count is exhausted, this module considers the errcr to be permanent and transfers ccntrcl to the Error Post and Second Level CCW Return module.

External Routines: None.

Tables/Work Areas: ERPIB, CCW, LCB.

Attributes: Supervisor mode, disabled, transient.

Closedown Terminal Statistics Recording Mcdule (Chart JK)

Module Name: IGEC904G

Entry Point: IGE0904G - activated by the Start-Stop ERP Control module (IGE0004G) when end of day recording is requested.

Functions: This module provides for terminal statistics recording when end of day recording is specified. When the Clcsedcwn Terminal Statistics Recording module is activated, it sets up input records for the system OBR/SDR module and transfers ccntrol to it. The OBR/SDR module records the statistics for each terminal and then returns to the Closedown Terminal Statistics Recording module to reset parameters for the next recording. After all recording has been performed, the Closedown Terminal Statistics Recording mcdule passes ccntrcl to the Line End Appendage.

External Routines: IGE0025F - OS OBR/SDR mcdule - to record terminal statistics.

Tables/Work Areas: LCB, Terminal Table entry.

Attributes: Supervisor mode, disabled, transient.

BSC ERP Control Module (Chart JL)

Module Name: IGE0004H

Entry Point: IGE0004H - activated by the I/C Supervisor (IOS) when the Line End Appendage (IGG019R0) detects a BSC error condition or when an interrupt occurs on I/O that was started by a BSC ERP module.

Functions: This module transfers ccntrcl tc the appropriate BSC ERP module to prccess the specific error ccndition that cccurred on a particular CCW. The BSC ERP Contrcl module receives ccntrcl from IOS when the Line End Appendage returns tc ICS with a BSC line error condition and when an interrupt occurs on an I/O operation that was initiated by a BSC ERP modue. This ccntrcl module transfers control to one of the following modules according tc the conditicn by which the control module was activated.

- BSC Read/Write Eauipment Check, Lcst Data, Interventicn Required, and Unit Exception ERP module (IGE0104H) - activated by the control module to process a unit exception on a write CCW and to process a unit check (eauipment check, lost data, intervention reauired, cr bus-out check) on a write or read CCW.

- Non-operational Control Unit ERP module (IGE0204G) - activated by the contrcl module when a ccntrol unit is not operaticnal. This is indicated by the condition code 3 after a Start I/C command.

- BSC Read/Write Data Check, Overrun, and Command Reject ERP module (IGE0204H) - activated by the ccntrcl module to process data checks, command rejects, and overruns on read or write CCWs.

- Unit Check for Non-read, Non-write, and Ncn-poll CCWs ERP module (IGE0304G) - activated by the ccntrcl mcdule when a unit exception error occurs on a CCW that is not a read, a write, or a poll operation.

- Auto Poll and Read Response to Poll Unit Check and Unit Exception ERP module (IGE0404G) - activated by the control module tc process a unit exception or a unit check cn a poll cr a read response to poll CCW.

- BSC Second Level CCW Return Module (IGE0404H) - activated by the control mcdule when the special return indicator is set and an interrupt cccurs on I/O that was initiated by an ERP module.

- BSC Error Post module (IGE0504H) - activated by the contrcl module under four different conditions.

   1. An attention, status modifier, control unit end, or busy condition indicated by the ccntrcl unit. In this case, the control module sets an error flaq in the LCB before activatinq the ERP prccessinq module.

   2. A proqram check, protection check, cr chaininq chain error on the line. In this situaticn, the ccntrcl module sets an error flaq in the LCB before activatinq IGE0504H.

   3. Any unit check on BSC devices that is not handled by the BSC Read/Write Eauipment Check, Lost Data, Interventicn Required, and Unit Exception ERP module, the Unit Check for Non-read, Non-write, and Non-poll CCWs ERP module, the Auto Pcll and Read Response to Poll Unit Check and Unit Exception ERP module, and is not a time-out on a read to addressinq, a read cr write ENQ, a read response to ENQ, a read text CCW with no data received, a read response to text, cr an cverrun or data check error on a text cr non-text read CCW.

   4. Any retriatle errors on which the retry count is exhausted without successful reccvery.

- BSC Channel Check ERP module (IGE0804H) - activated by the control module to process channel errors such as channel contrcl checks, channel data checks, and interface contrcl checks.

- Line End Appendaqe (IGG019R0) - activated by the control module under the fcllowinq conditions:

  1. When a read unit exception cccurs and updatinq of terminal statistics is not required.

  2. When a time-out occurs on a read text and data is received.

  3. When overrun or data check errcrs cccur on read text CCWs. These errcrs are not retried ky an ERP module, but may be retried in the MCF.

If a time-out on a read to addressinq, a read ENQ, or a read response to ENQ occurs, the control module restarts the channel proqram. If a time-out cn a read text CCW cccurs and no data is received, the control module restarts the channel proqram at the read text CCW. If a time-out on a read response to text CCW cccurs, the control module executes a write ENQ channel proqram.

External Routine: IEDQTNT - Termname Table code - to obtain the terminal entry address.

Tables/Work Areas: LCB, CCW, AVT, DCB, Terminal Table entry.

Attributes: Supervisor mode, disabled, transient.

BSC Read/Write Equipment Check, Lost Data, Intervention Required, and Unit Exception ERP Module (Chart JM)

Module Name: IGE0104H

Entry Point: IGE0104H - activated by the BSC ERP Contrcl module (IGE0004H) to process unit check and unit exception error conditions that cccur on read and write CCWs.

Functions: This module processes read/write unit check and unit exception error conditions that occur on BSC lines.

If a unit exception cccurs on a write CCW, the action taken by this module depends on the type of write CCW:

- If the CCW is for a write ENQ (a line bid), this module restarts I/O at the read response CCW. A unit exception at line bid time could indicate that a ccntention situation exists.

- A unit exception on any other write CCW implies that either the line is ncisy or the other station is usinq bad line procedures.

For this situation, this module executes a Read Skip channel program.

If the BSC Unit Check and Unit Exception ERP module receives control after a unit check occurs, the module analyzes the IOB sense data to determine the course of action. The action taken depends on the type of error that occurred:

- Equipment check - this module sets an error flag in the LCB and transfers control to the BSC Error Post module (IGE0504H).

- Lost data on a write CCW - this module sets an error flag in the LCB and transfers control to the BSC Error Post module (IGE0504H).

- Lost data on a read CCW - this module acts according to the type of read on which the error occurred:

  1. Lost data on a read response to ENQ CCW - this module reexecutes the write ENQ CCW.

  2. Lost data on a read ENQ CCW - this module reexecutes the read ENQ CCW.

  3. Lost data on a read response to text CCW - this module executes a write ENQ, read response channel program.

  4. Lost data on a read text CCW - this module returns control to the Line End Appendage (IGG019R0) where a read ENQ, write NAK channel program is executed.

- Intervention required - this module transfers control to the BSC Error Post module (IGE0504H) where this error is processed as a permanent error condition.

- Bus-out check - if the error occurred on the command, not on the data, this module retries the CCW; if the error occurred on the transmitted data and the CCW is a write text, this module executes a read response CCW.

External Routines: None.

Tables/Work Areas: CCW, IOB, LCB, SCB, AVT, UCB.

Attributes: Supervisor mode, disabled, transient.

BSC Read/Write Data Check, Overrun, and Command Reject ERP Module (Chart JN)

Module Name: IGE0204H

Entry Point: IGE0204H - activated by the BSC ERP Control module (IGE0004H) when a data check, a command reject, or an overrun occurs on a read or a write command.

Functions: This module processes a data check, a command reject, or an overrun on a failing read or write CCW. This module examines the failing CCW to determine the appropriate course of action.

If this ERP module finds an undefined error or if the retry count is exhausted, the module transfers control to the BSC Error Post module (IGE0504H) where the error is recorded. If an intermediate CCW sequence is required, the BSC Read/Write Data Check, Overrun, and Command Reject module builds the CCW sequence and sets a special return indicator for the BSC Second Level CCW Return module (IGE0404H). The BSC Second Level CCW Return module services the next interrupt and controls subsequent recovery attempts for this error.

In all other situations, the BSC Read/Write Data Check, Overrun, and Command Reject ERP module advances the retry counter and attempts a retry at the appropriate point in the failing CCW sequence.

External Routines: None.

Tables/Work Areas: CCW, IOB, LCB, SCB, AVT, UCB.

Attributes: Supervisor mode, disabled, transient.

BSC Second Level CCW Return Module (Chart JC)

Module Name: IGE0404H

Entry Point: IGE0404H - activated by the BSC ERP Control module (IGE0004H) to process interrupts that occur on I/O that was initiated by an ERP module.

Functions: This module attempts to retry channel programs that were initiated by an ERP module. This module retries the channel program until either the retry is successful or the retry count is exhausted. When the retry count is exhausted, this module considers the error to be permanent and transfers control to the BSC Error Post module (IGE0504H).

If the channel and unit status of the CSW indicate that the I/O was error free, the BSC Second Level CCW Return module either restarts the user channel program at the correct CCW or transfers control to the Line End Appendage (IGG019R0) where the received data is checked and the appropriate CCW is executed.

External Routines: None.

Tables/Work Areas: CCW, LCB, SCB, AVT, Terminal Table entry.

Attributes: Supervisor mode, disabled, transient.

BSC Error Post Module (Chart JP)

Module Name: IGE0504H

Entry Point: IGE0504H - activated by the BSC ERP Control module (IGE0004H) or by any of the BSC ERP processing modules.

Functions: This module handles permanent error situations. It receives both errors that are not retried and errors that are considered to be permanent only after the retry count is exhausted. This module builds the records necessary as input for OBR/SDR recording and then passes control to the OBR/SDR module (IGE0025F). The OBR/SDR module records the error and logs an error message on either the system console or the operator control terminal.

External Routine: IEDQTNT - Termname Table code - to obtain the terminal entry address.

Tables/Work Areas: CCW, LCB, SCB, AVT, Terminal Table entry.

Attributes: Supervisor mode, disabled, transient.

BSC Channel Check ERP Module (Chart JQ)

Module Name: IGE0804H

Entry Point: IGE0804H - activated by the BSC ERP Control module (IGE0004H) to process channel control check ending status, interface control check ending status, and channel data check ending status.

Functions: This module processes channel ending status errors that are detected by IOS.

The Channel Check Handler is an optional extension of IOS for configurations that use the 2860/2870 channels. The Channel Check Handler determines whether a channel control check or an interface control check is recoverable. If so, this handler builds an ERP interface byte (ERPIB) to provide the BSC Channel Check ERP module the information for a possible retry.

When the BSC Channel Check ERP module gains control after a channel control check or an interface control check, it searches a list of ERPIBs to determine whether IOS has supplied preliminary parameters for retry. If the module does not find an ERPIB for the failing device, the module considers the error to be permanent and passes control to the BSC Error Post module (IGE0504H). If an ERPIB is found for the device, the BSC module tests the retry flag in the ERPIB. If this flag is on, the module considers the error to be permanent and exits to the BSC Errcr Post mcdule. If the flaq is not on, the BSC Channel Check ERP module saves the ERPIB data in a work area, clears the ERPIB to zero to make space for system sense information, and attempts a retry procedure that is based on the ERPIB data, the failinq CCW, and the retry count. The module ccntinues the retry procedure until either the retry is successful or the retry count is exhausted. If the retry count is exhausted, this module transfers contrcl to the BSC Error Post module.

When the BSC Channel Check ERP module qains ccntrol after a channel data check, the mcdule attempts tc retry the failinq CCW. If retry is unsuccessful and the retry count is exhausted, this module considers the error to be permanent and transfers ccntrcl to the BSC Error Post module.

External Routines: Ncne.

Tables/Work Areas: ERPIB, CCW, LCB.

Attributes: Supervisor mode, disabled, transient.

TIME SHARING OPTICN ROUTINES

TSO Attention Rcutine (Chart YA)

Module Name: IEDAYA

Entry Points:

- IEDAYA (ENTRY1) - activated by Buffer Disposition (IEDQBD) through the TSO INMSG/OUTMSG Linker (IEDAYX) when it detects an attention interrupt (hardware or simulated).

- IEDAYA+12 (ENTRY2) - activated by Line End Appendaqe (IGG019R0) through the TSO TOHALT routine (IEDAYF) or directly from IEDAYF when it receives an attention interrupt on a sinqle prepare CCW.

Functions: The TSO Attention routine provides the terminal user the ability to affect line deletion, CPU task interruption (qiving control to the STAX Exit) or both. These functions can be accomplished either throuqh a hardware attention interrupt (an ATTENTION or REQUEST key on a terminal and an ATTEN macro instruction in the TCAM message control proqram) or through a software-simulated attenticn interrupt (a SIMATTN macro instruction in the TCAM message ccntrol program).

An attention request from a terminal causes the IC Supervisor to activate the TCAM Line End Appendaqe (IGG019R0). This appendage identifies the request and sets an attention interrupt switch (SCBATTN) in the Station Contrcl Blcck for the terminal. When the TCAM message handler for that terminal qains ccntrol, the ATTEN macro expansion in that messaqe handler tests the SCBATTN bit and if it is on, activates the TSO IOHALT routine (IEDAYF), which activates the TSO Attention routine. Upon detection of an attention, the TSO IOHALT routine may activate the TSO Attention routine directly by tposting the ICB to it.

An attention request from the SIMATTN macro in the program causes the TSO Simulated Attention routine (IEDAYS) to set the "simulated-attention interrupt" switch (SCBSATTN) in the Station Control Block. This switch causes the macro expansion to activate the TSO Attention routine.

The TSO Attention routine first branches and links to the Termname Table code (IEDQTNT) to get the address of the terminal entry, which contains the address of the Destination QCB. The routine then determines whether the terminal is handling a TSO session. If the terminal is not, the routine determines whether the line was sending or receiving (the routine was entered at ENTRY1) or on a single prepare CCW (the entry was at ENTRY2) and exits accordingly - to DSPDISP for ENTRY2 and to DSPCHAIN for ENTRY1.

In the input mode, if the key has been specified for line delete and in fact at least one character has been entered, the TSO Attention routine determines if automatic line numbering has been specified and if so, decrements the current line value. This is done so that the terminal may once again be prompted for the deleted line number. Before returning to the message handler code, the routine sets a flag within the prefix of the input buffer. Also, the routine sets a flag (SCBXPD) in the SCB to indicate to the TSO Message Generation routine (IEDAYM) to write the message (!D) to the terminal. (This indicates that the line is being deleted).

If the key had not been specified for line delete, or if it had and nothing had been entered on the input line prior to the Attention key being depressed, the TSO Attention Routine then processes the interrupt as a request for an attention exit. This is the case when the system is in the output mode, when the key is depressed and the terminal supports the Transmit/Interrupt feature.

Once the TSO Attention routine determines that exit request handling is to be performed, it determines if any exit levels are currently available. If not or if the user program does not specify any STAX macros, the routine drops input and output buffers and sets a flag (SCBXPI) in the SCB to indicate that the TSO Message Generation routine is to write the message (!I) to the terminal. (This indicates an attention ignored). The TSO Attention routine then issues a QTIP2 to clear the queues and record that an attention was ignored.

If an exit level is available, the routine decrements the ATTN count in the TJB (TJBATTN) and drops the input and output queue.

If the SCBATTN bit is on without either the SCBXPI bit or the SCBXPD bit, the TSO Message Generation routine writes the message (!) to the terminal. This indicates that an attention is accepted and an exit routine is scheduled. If the user is in main storage, the TSO

Attention routine issues a QTIPO to set the flag RCBFAT in the RCB, tposts the PCE, and returns to the message handling code. If the user is not in main storage, the routine sets an entry code of (4), puts the TJID in register zero, and issues a QTIP1 to swap the user into main storage.

When TCAM passes control to the TSC Attention routine, TCAM notifies the routine whether the line was sending, receiving, or on a single prepare CCW. The single prepare CCW is used to monitor the line attention interruptions when a keyboard is locked. If the attention interruption occurs while sending cr receiving, the TCAM Dispatcher, via a MSGGEN macro instruction, schedules the TSO Message Generation routine (IEDAYM). If the interruption occurs cn a prepare CCW, the TSO Attention routine branches directly to the TSO Message Generation routine.

External Poutines:

- IFDQTNT - Termname Table code - to get the address of the terminal entry.
- QTIP SVC - TSO SVC - Entries 0, 1, and 2 - to clear the input and output queues, remove system and user LWAITs and QWAITs, set flags, and swap the user into main storage.

Tables/Work Areas: AVT, CVT, LCB, QCB, RCB, SCB, STCB, TJB, terminal entry, TSB,Time Sharing CVT, TSI.

Attributes: Reentrant, enabled, supervisor mode.

TSO Carriage Subroutine (Chart YC)

Module Name: IEDAYC

Entry Point: IEDAYC - entered from the TCAM User Interface routine (IEDQUI) when a CARRIAGE macro instruction is coded in an INBUF or OUTBUF subgroup of an MH.

Functions: The TSO carriage subroutine initially links to the Termname Table code (IEDQTNT) to get the terminal entry from which it extracts the address of the Destination QCB. The TSO Carriage subroutine maintains a count in the QCB of carriage positions for keyboard devices, so that when output is sent, idle characters can be inserted properly. If a translation error has occurred, or if the current buffer is a zero-length buffer, the TSO Carriage subroutine passes control to the MH via the Return Interface routine (IEDQLM). Otherwise, the subroutine scans the buffer, character by character, for new line characters and backspace characters. When it finds a backspace character, the subroutine decrements the carriage position count by one, unless the backspace is the first character in the buffer. When it finds a new line character, the subroutine resets the carriage position count to zero, and, for 2260 devices, updates the OCB simulated attention line count, unless the new line is the last character in the buffer. If it is the last character, the TSO Carriage subroutine sets the QCB retry count field to reflect this, so that, when this subroutine is entered tc scan the next buffer, scanning will begin at the first character in the buffer. The TSO Carriage subroutine also zeros out circle Ds from all devices except 2741s and STX/Addressing character sequences from 2260s, if they appear in input buffers. The subroutine also counts output lines for simulated attention by line ccunt, and turns on the "simulated attention" bit in the SCB when required. After it has scanned all units of the tuffer, the TSO Carriage subrcutine updates the carriage position count in the QCB, and returns ccntrol to the MH via the Return Interface routine (IEDQLM).

External Routine: IEDQTNT - Termname Table ccde - to get the terminal entry from which it extracts the address of the Destination QCB.

Tables/Work Areas: CVT, Time Sharing CVT, AVT, TSB, LCB, SCB, DCB, OCB, buffer prefix, terminal entry.

Attributes: Reentrant, enabled, resident, problem program mode.

Time Sharing Destination Scheduler (Chart YD)

Module Name: IEDAYD

Entry Point: IEDAYD - receives control tc initiate a write break operation, if necessary, and to assign a buffer or QCB to its destination. The VCON fcr this module (instead of for IEDQHM) is assembled as the destination for every line that can be used by TSO.

Functions: The Time Sharing Destination Scheduler gets control in place of the TCAM Destination Scheduler when TSO is in the system. If this routine is activated because a TCAM buffer was tposted to it, it loads the address of the TCAM Destinaticn Scheduler STCB (IEDQHM) into register 3, and branches to the Dispatcher bypass function to immediately activate the STCB.

If the Time Sharing Destination Scheduler is activated by the tposting of a Time Sharing Destination QCB to itself, it sets the tposted bit off in the QCB and checks to see if a simulated attention Read was requested. If it was requested, the Time Sharing Destination Scheduler immediately passes control tc the Dispatcher dispatch function. If nct, this rcutine checks to see if TPUT requested a Write Break. If not, the scheduler passes control to the dispatch function. If a Write Break was requested, this routine loads the LCB address into register 4 and branches to the Time Sharing Scheduler (IEDAYZ) to determine whether a Write Break channel command can be issued. If it cannot be issued, the Time Sharing Scheduler returns control to the Time Sharing Destination Scheduler, which then passes control to the Dispatcher dispatch functicn.

External Routines: IEDAYZ - Time Sharing Scheduler - to determine whether the Write Break channel command requested by TPUT can be issued.

Tables/Work Areas: AVT, LCB, QCB, DCB, STCB, and TSID.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

TSO TIOC Edit Routine (Chart YE)

Module Name: IEDAYE

Entry Point: IEDAYE - called by the TSOUTPUT routine (IEDAYO) to edit output buffers or by the TSO Message Generation routine (IEDAYM) to edit MSGGEN messages.

Functions: The TSO TIOC Edit routine inspects and edits output messages contained in TSO buffers and MSGGEN messages in the SCB. The routine moves edited TSO messages to TCAM buffers to be sent. MSGGEN buffers remain in the SCB. The routine uses the line size, the receiving buffer size, and the presence of New Line characters in the messages for the EDIT scan. The user specified the line size for a TSO message in the Time Sharing block (TSB). For MSGGEN messages, the line size defaults to 120 bytes. If the maximum line size is reached, the TSO TIOC Edit routine inserts a new-line character and the appropriate number of reserve characters into the buffer to avoid overprinting the line at the terminal. On one entry to it, the TSO TIOC Edit routine may move no more printable data to the TCAM buffer than the physical line size. If the message to be sent will exceed the line size, the routine inserts the appropriate character(s) (based on terminal type) at the end of the line and reverts to the TSOUTPUT routine, with a return code (X'0C') indicating that the whole message was not edited. The presence of a new line character in the buffer signifies the end of a complete line. At the end of a line or message, the TSO TIOC Edit routine checks to see if a simulated attention by line count was requested. If it was, the routine links to the TSO Simulated Attention routine (IEDAYS) to update the QCB simulated attention line count, and to request a simulated attention Read if the threshold is reached. Next, the TSO TIOC Edit routine checks to see if the message is a control or "ASIS" message. These messages cannot contain standard line and carriage control characters. For these messages, the routine replaces any invalid characters (EOT, New Line, Tab, etc.) with a colon to avoid program-caused I/O errors. In addition to inserting New Line and reserve characters, and replacing invalid characters, the routine inserts any other line control characters (that is, STX and ETX) and carriage control characters (that is, line feed characters) required to edit the message for output to a particular terminal. Whenever a TIOC buffer is emptied, a line is filled, or when a message is completely edited, the TSO TIOC Edit routine issues the QTIP (SVC 101) to update the offset and length fields in the TIOC buffer prefix. If the TCAM buffer accommodated the entire TSO buffer, the TSO TIOC Edit routine marks the prefix edited-in-full. Otherwise, the QTIP/SVC sets the offset to indicate the point in the TSO buffer to start editing on the next entry to the routine, and sets the length to indicate the length of the data yet to be edited. On return to the calling routine, the TSO TIOC Edit routine sets a return code (X'00'-successful completion; X'0C'-partial line or partial message moved; X'10' - end of line reached) to indicate the status of the edit request.

Note: On a 2260, this routine inserts New Line characters only at the end of a message when the number of characters is less than the line size.

External Routines:

- IEDQTNT - Termname Table code - to get the terminal entry from the LCB terminal index or line entry, or from the invitation list for the line.

- IEDAYS - TSO Simulated Attention routine - to handle requests for simulated attention by line count.

- QTIP SVC routine - SVC 101 - to update the TIOC buffer prefix.

Tables/Work Areas: CVT, Time Sharing CVT, AVT, TSB, QCB, TSID, LCB, SCB, DCB, terminal entry, TCAM buffer prefix used for TSO, TIOC buffer prefix.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

TSO IOHALT Routine (Chart YF)

Module Name: IEDAYF

Entry Point: IEDAYF - receives control from the TCAM Dispatcher when an LCB or an ERB is tposted to it from Line End Appendage (IGG019RO)

Functions: The TSO IOHALT routine gets control when either an LCB or an ERB is tposted from Line End Appendage (IGG019RO). For an LCB, this routine monitors for an attention if the LCB open check op code has been set and if a Prepare has been issued for the line. When an attention occurs, this routine issues an IOHALT SVC (SVC 33) on the Prepare. It then exits to the Dispatcher dispatch function to dispatch the next subtask.

An ERB is tposted when a hardware attention occurs or when a 2741 hangs up. This routine locates the LCB from the ERB and turns off the prepare bit. Then, if the LCB is in the time delay queue, this routine links first to the Time Delay subtask (IEDQHG) to remove it, and then to the Dispatcher priority function to insert the Receive Scheduler STCB into the LCB STCB chain by priority. If the LCB completion code indicates that I/O ended on the line because of an IOHALT, the routine sets line free priority, and sets the LCB to be tposted to itself. If the line is connected to a 2741, and a Prepare CCW was interrupted, this routine indicates in the LCB that a circle D was sent and checks to see if the 2741 has hung up. If it has, the routine sets the LCB to be tposted to the TSO Hangup routine. If a Prepare CCW was not interrupted, but a circle D was sent, the routine sets the LCB to indicate circle D sent. For 2741s that have not hung up, and for all other devices, the TSO IOHALT routine turns on the SCB hardware attention bit (SCBATTN), and sets the LCB to be tposted to the TSO Attention routine. In all cases except when I/O was ended by IOHALT, the routine sets PCI priority. Finally, this routine links all previous elements in the chain, puts the QCB address in the LCB, indicates that the LCB is the new first element in the chain, and exits to the Dispatcher chain function to tpost all the elements in the chain to the appropriate QCBs.

External Routines:

- OS IOHALT routine (SVC 33) - to halt I/O on the line.

- IEDQHG - Time Delay subtask - to remove an LCB from the time delay queue.

- IGG019RP or IGG019RO - TCAM Dispatcher DSPPRIOR entry point - to insert the Receive Scheduler STCB into the LCB STCB chain.

Tables/Work Areas: AVT, DCB, LCB, SCB, DEB, TSID, TPRIOR, ERB.

Attributes: Reentrant, enabled, resident, problem program mode.

TSO Hangup Routine (Chart YH)

Module Name: IEDAYH

Entry Points:

- IEDAYH (ENTRY1) - entered from the TSO INMSG/OUTMSG Linker (IEDAYX) to determine whether I/O errors have occurred and whether a HANGUP macro is to be processed.

- IEDAYH+12 (ENTRY2) - entered from the Line End Appendage (IGG019RO) to determine whether a HANGUP macro is to be processed.

Functions: This module ensures that line errors associated with TSO terminals are identified to the terminal user and cancels the message that is in error.

The TCAM Buffer Disposition subtask (IEDQBD) activates the TSO INMSG/OUTMSG Linker, which activates the TSO Hangup routine, when the message handler has processed a complete message and a HANGUP macro is specified.

The TSO Hangup routine first determines whether an error that it can handle has occurred. If no error is found, the routine returns immediately to the TCAM Dispatcher. When an error is found, the TSO Hangup routine branches to the Termname Table code (IEDQTNT) to get the terminal entry address. On return, the routine uses the terminal entry and the LCB index to find the QCB associated with the error and determines whether the QCB belongs to a TSO user. If the QCB does not belong to a TSO user, the routine returns immediately to the TCAM Dispatcher. If the QCB is for a TSO user, the TSO Hangup routine determines whether the error is permanent. If so, the routine sets flags to disconnect this line and turns off the SCB error word bits, issues a QTIP request (SVC 101) to turn on the TJBHUNG bit, invokes the Time Sharing Interface Program (TSIP) with an entry code of 36, and activates the System Initiated Logoff (SIL) to begin TSO logoff procedures.

When no permanent errors are found, the TSO Hangup routine determines whether the line is sending or receiving and performs the necessary error handling for either situation. If the retry count in the QCB is set (initially to three), and the line is sending, the routine decrements the count by one each time a countable error occurs. When the count is zero, the routine sets flags to disconnect this line, turns off the SCB error word bits, and issues a QTIP request as when a permanent error occurs. If the count is not zero and the line is sending, the TSO Hangup routine returns immediately to the TCAM Dispatcher.

If the retry count in the QCB is set and the line is receiving, the TSO Hangup routine prepares a message to inform the terminal user of his status and decrements the count by one. When the count is zero, the routine sets flags to disconnect this line, turns off the SCB error word bits, and issues a QTIP request as described for a permanent error above. When the retry count is not zero and the line is receiving, the routine tests the SCBTMINN field. If this field is not on, the routine cancels the message just prepared and returns to the TCAM Dispatcher. If the SCBTMINN field is on, the routine sends the status message to the user and then returns to the TCAM Dispatcher.

External Routines:

- IEDQTNT - Termname Table code - to convert the Termname Table offset to the address of the Terminal Table entry.

- QTIP SVC - TSO SVC - to set a bit in the TJB.

Tables/Work Areas: AVT, CVT, LCB, buffer prefix, QCB, SCB, TJB, Time Sharing CVT, TSB, TCT.

Attributes: Reusable, refreshable, enabled, problem program mode.

TSINPUT Routine (Chart YI)

Module Name: IEDAYI

Entry Point: IEDAYI+2 - activated by the TCAM Dispatcher when either the PCI Appendage or the Line End Appendage routes a message from a TSO terminal to the message control program message handler for that terminal, or when one of the TIOC routines tposts the TSINPUT Destination QCB to itself to remove a system LWAIT condition.

Functions: The TSINPUT routine moves incoming data from a TCAM buffer into a TSO buffer and places the TSO buffer in the TSO input buffer queue for processing by TSO TGET requests.

When the TSINPUT routine gets control and finds a TCAM buffer being passed to it for processing, the routine determines whether the message is to be canceled (PRFCNCLN bit is on). If this bit is on, all TCAM and TSO buffers associated with this message are freed, and the TCAM buffers are returned to the TCAM Dispatcher.

If the message is not to be canceled, TSINPUT manipulates the transfer of incoming data from TCAM to TSO buffers in the following manner.

The routine scans the text in the TCAM buffer for CR, Line Delete, and EOT characters. If a Line Delete character is detected, or if Attention for Line Delete occurred, the routine deletes the portion of the line to the previous CR (or to the beginning of the message). Otherwise, the routine moves each physical line to a TSO buffer or buffers.

After the complete TCAM message has been processed, this routine flags each TSO buffer that contains part or all of a complete TSO message as complete in the buffer prefix. If the data in a TCAM buffer does not complete a physical line, the routine flags the TSO buffer as a fragment in the buffer prefix (turns the BUFFFRAG bit on). When the TCAM buffer that completes the physical line is processed and the data is moved to TSO buffers to complete the line, the TSO buffers are flagged complete (BUFFFRAG bit in each buffer is turned off). Each TSO buffer is placed on the TSB input buffer queue in FIFO order. TSO TGET requests may retrieve only complete messages from this queue.

If the incoming message has filled the maximum number of TSO buffers alloted to this terminal for input, the TSINPUT routine places the terminal in an LWAIT condition. That is, it locks the terminal keyboard on completion of the current line. LWAIT is entered by turning on the TSBLWAIT and QCBNOBUF bits to prevent TCAM from issuing further READ instructions to the terminal. (Additional TSO buffers are obtained, if possible, to complete the current line.)

When no TSO buffers are available to move data into, incoming TCAM buffers are held in a buffer wait queue. Whenever TCAM buffers are placed in this queue, the QCBBUFQ bit in the associated QCB is turned on to indicate that TCAM buffers are being held for this terminal. In this case, a system LWAIT condition is entered; that is, all terminals currently in input mode enter an LWAIT condition as TSINPUT handles incoming data. System LWAIT has no effect on output mode or control mode terminals until output is complete or input is required.

After a TIOC routine has released one or more TSO buffers, a check is made to see if a system LWAIT condition exists. If this condition does exist, the TIOC routine tposts the the TSINPUT Destination QCB to itself. When TSINPUT gets control, it finds that no TCAM buffer is being passed. TSINPUT then determines whether any TCAM buffers are in the buffer wait queue. If this is the case, TSO buffers are obtained to remove the TCAM buffers one at a time, in FIFO order, until all are removed or no more TSO buffers are available.

When all TCAM buffers have been removed from the buffer wait queue (or when none were held), TSINPUT determines whether sufficient TSO buffers are available to remove the system LWAIT condition. If sufficient buffers are available, waiting TSBs are removed from the wait queue in FIFO order, the corresponding QCBTSBQ and QCBNOBUF bits are turned off, and TCAM issues a READ to unlock the effected terminal keyboards to allow input as soon as possible.

If a terminal is in both a system LWAIT and a maximum-number-of-TSO-buffers LWAIT, the appropriate TSB is removed from the wait queue when the system is removed from LWAIT, but the QCBNOBUF bit is not turned off to unlock the keyboard. The keyboard can be unlocked only when the associated application program frees sufficient buffers, through one of the TIOC routines, to allow the user to continue.

If the terminal is in input mode and a TCLEARQ instruction is issued, the TSBIFLSH bit is turned on to indicate that an input queue flush is in progress. TSINPUT then checks the incoming TCAM buffer to determine whether it is a buffer of the TCAM message. If this is the first buffer of the message, the TSBIFLSH bit is turned off, and normal buffer processing continues. If the buffer is not first and not last, the data is dropped and the buffer is returned. If this is the last buffer of the message, the data is dropped, the buffer is returned, and the TSBIFLSH bit is turned off to indicate TCLEARQ completion. In addition, when TSINPUT finds the TSBIFLSH bit on, the routine returns any TCAM buffers associated with this terminal that are on the buffer wait queue, and turns off the corresponding QCBBUFQ bit.

If the received message is a partial line caused by a break-in, the partial line is sent back to the terminal after completion of output to allow the user to complete his message. The partial line is also placed on the input queue.

When TSINPUT finds that a break-in has occurred (LCBWRBRK bit on), and if the incoming TCAM buffer does not end in a CR, Line Delete, or EOT character, the routine turns the corresponding TSBBRKIN bit on to indicate to TSOUTPUT that a partial line exists, in TSO buffers, for prompting. The corresponding TSO header buffer is flagged as a partial line (BUFFPART bit is turned on). At this point, the terminal

Destination QCB is tposted to itself to activate the send cperation that sends the break-in message and prompts the user.

If TSINPUT finds the TSBBIPI bit cn, the incoming TCAM buffer is the completicn cf a break-in message that was sent to the user to complete. TSINPUT turns off the TSBBIPI bit and completes the partial message on the input queue by adding the new data to it.

If automatic line numbering is in progress (TSBAULST and TSBAUTON bits are on), the current line number is incremented as each complete line is received. The next line number is sent when line completion occurs (EOT or EOB is received) by turning on the SCBALN bit. The user can terminate automatic line numbering at any time by entering a null line.

When line ccmpleticn occurs, TSINPUT turns off the QCBREAD and QCBTGET bits tc indicate that READ no lcnger has priority and that any TGET requests have been satisfied. If the user is not in main storage and a TGET request has not been satisfied (an IWAIT ccndition exists), control is passed to the Time Sharing Interface Program (TSIP), which sets the "restcre" flag in the TJB and pcsts the RCT tc release the user from IWAIT. If the user is in main storage and an IWAIT condition exists, TSINPUT flags all TCBs as dispatchable and continues ncrmal processing.

TSINPUT always exits to DSPCHAIN in the TCAM Dispatcher when processing is ccmplete or can no lcnger ccntinue.

External Routines:

- IEDQTNT - Termname Table code - to get the address of a terminal table entry.
- QTIP SVC - TSC SVC - activated with appropriate entry codes, to delete a message fragment, to terminate TCLEARQ processing, to delete the input line currently being scanned, to mcve scanned data from TCAM to TSO buffers, and to put the system into LWAIT or put a TSB on the waiting TSB queue.

Tables/Work Areas: AVT, CVT, Time Sharing CVT, DCB, IOB, LCB, buffer prefix, QCB, SCB, TIOCBUF, TIOCRPT, Terminal Table, TSB, TSI.

Attributes: Reusable, refreshable, problem prcgram mode.

TSO Logon Routine (Chart YL)

Module Name: IEDQYL

Entry Point: IEDAYL - activated by the User Interface routine (IEDQUI) when a LCGON macro is coded in a TSO message handler.

Functions: The TSO Logon routine informs the TSO system when a potential TSO user attempts to log onto the system and to route TSO-bound messages to the TSINPUT routine (IEDAYI).

The TSO Logon routine first initializes the LCBTTCIN field in the LCB and the PRFSRCE field in the prefix of the buffer that contains the logon request. The routine locates the proper Destination QCB via the Termname Table code (IEDQTNT) and then routes the message. If a Time Sharing session is already in progress with the terminal in question, the logon request in the buffer is not an initial request.

If the TSO session is in progress, the routine scans the message handler for a LOGON macro request, and if one is not found, performs one of the following actions:

- Requests TCAM to cancel this buffer.

- If the Locate Option routine (IEDQAE) is present, branches there to find the options for the NOLOG user-exit routine, if one exists. If a user-specified NOLOG exit routine is specified, the TSO Logon routine branches there to process the buffer.

If neither of these actions can be performed, the TSO Logon routine does one of the following:

- If TSO is not in the system, tells the user that TSO is not running and returns control to the message handler via the Return Interface routine (IEDQLM).

- If the environment is TCAM-TSO, sends the user the 'try again' message and returns control to the message handler via IEDQLM.

- If the environment is TSO only and the QCB retry count does not already exist, sets up a count, sends the 'try again' message, and returns to the message handler via IEDQLM.

- If the environment is TSO only and the QCB retry count has been previously set up, decrements the count by one, sends the 'try again' message, and returns to the message handler via IEDQLM.

- If the environment is TSO only and the QCB retry count is zero, advises the user that his logon attempt has failed, tells TCAM to disconnect the terminal, and returns control to the message handler via IEDQLM.

If there is a LOGON macro request in the message handler, the TSO Logon routine puts blanks in the buffer to overlay any characters preceeding the characters LOGON. Then the routine performs one of the following actions:

- If the terminal cannot support TSO, tells the user that the terminal cannot receive TSO messages, marks the buffer 'to-be-canceled', and returns control to the message handler via IEDQLM.

- If the terminal is held (unable to receive messages through normal means), generates a message to inform the user, sets flags to cancel the buffer, and returns to the message handler via IEDQLM.

- If TSO is not in the system, advises the user of this fact, sets flags to cancel the buffer, and returns to the message handler via IEDQLM.

- If the maximum number of TSO users has already logged on, advises the last user to log on of this fact, sets flags to cancel the buffer, and returns control to the message handler via IEDQLM.

- If TSO does not have the maximum number of users already logged on, the routine performs its initialization procedures for TCAM. The routine places zeros in the QCBRETCT, QCBSATCT, QCBTSOF1, QCBTSOF2, and QCBCARCT fields of the QCB. The routine then places in the CINHIBIT field the value of the TCT in the QCBINHBN field and turns on the QCBTSSES switch. Next, the routine performs the initialization for TSO by issuing a QTIP (SVC 101) request to activate the Logon procedure in the TSO region. The QTIP SVC searches the TJB chain for an available TJB. If a TJB is not free, the SVC places a X'00' return code in register 15 and returns control to the TSO Logon routine. The TSO Logon routine activates the Locate Option routine (IEDQAE). If the return code from IEDQAE indicates that logon initialization is requested, the TSO Logon routine performs the TCAM and TSO initialization procedures then returns control to the message handler via the Return Interface routine.

If a free TJB is found, the QTIP SVC increments the TSCVTCUS field by one (to include the current user), turns on the TSCLOGON field in the Time Sharing CVT to indicate that a LOGON has been issued, and puts in the TJB the address of its associated TSB entry and QCB. At this point the QTIP SVC places the characters 'STARTING' into the TJBUSER field to denote a new user, turns off the TJBNJB bit to indicate that this TJB is being used, turns on the TJBLOGON bit to indicate a LOGON request, and turns on the TSBINUSE bit to indicate a used TSB. Next, the SVC turns off the TSBATNLD bit so that the attention key on the terminal cannot activate the line delete function. The QTIP SVC initializes the TSBLNNO, TSBLNSZ, TSBSTCC, and TSBDSPLY fields according to the terminal type indicated either in the UCB or in the Device Characteristics Table (DCT) and, depending on the terminal type found, may turn on the TSBATNID bit. The SVC places the contents of the PRFSRCE field from the buffer prefix into the TSBASRCE field, calculates the adjusted maximum input and output buffers allowed to each user and inserts them in the TIOCRPT field, posts the LOGON ECB for the Time Sharing Control program, places the value of the TJBTJID field in register 15, and returns control to the TSO Logon routine. If register 15 contains a zero, the Logon routine performs the same processing that was done when no free TJB was found.

External Routines:

- IEDQTNT - Termname Table code - to get the address of a Terminal Table entry.

- IEDQAE - Locate Option Routine - to find the address of the option field.

- OTIP SVC - TSO SVC - to perform initialization for TSO.

Tables/Work Areas: AVT, CVT, DCB, DEB, LCB, buffer prefix, QCB, SCB, TCT, TIOCRPT, TJB, Time Sharing CVT.

Attributes: Reusable, refreshable, enabled, problem program mode.

TSO Message Generation Routine (Chart YM)

Module Name: IEDAYM

Entry Points:

- IEDAYM - called by Buffer Disposition (IEDQBD) to process a MSGGEN message, or called by the TSO Simulated Attention routine (IEDAYS) to process a simulated attention message for a simulated by time interval.

- AYM000 - called by the TCAM Dispatcher (IGG019RB) to process any other type of message.

Functions: The TSO Message Generation routine processes a message, which may be provided in one of three places. If the message is generated in a MSGGEN macro instruction, it is located in the macro generation. If a simulated attention READ was requested, this routine generates a simulated attention message from a constant. If an automatic line numbering message of a prompt message was requested, the message is located in a user-specified field in the Terminal Status Block, which is in the TSO region.

If the message was not generated by MSGGEN, the TSO Message Generation routine checks to see if the last buffer of the message has been tposted (that is, it has been processed by the TSINPUT routine). If it has not this routine exits to the Dispatcher chain function to tpost the buffer. If the buffer has been tposted, this routine links to the Termname Table code (IEDQTNT), gets the terminal entry, and extracts the QCB address from it. At this point, if the message does not fall into any of the above three categories, this routine exits to the Dispatcher chain function to tpost the ERB to Buffer Disposition (IEDQBD).

The TSO Message Generation routine gains access to the message from whichever location applies, and sets the LCB to activate the terminal. The routine gains access to the translation table from the DCB, or, if the message is generated from a MSGGEN macro with a code operand, from the macro generation. If translation is via a TRANLIST macro, this routine links to the Locate Option Field Address routine

(IEDOAE) via the User Interface routine (IEDQUI) to locate the option field containing the translation table. If for any reason the message can not be tranlated, this routine turns off all SCB error word bits and all bits in the LCB sense byte, turns off the SCB translation requested bit, indicates that the remaining INMSG or OUTMSG macros are to be bypassed, and exits to the Dispatcher chain function to tpost the ERB to Buffer Disposition.

The TSO Message Generation routine left-justifies the message if necessary, and moves it to the multiple-buffer-scan save area in the SCB. This routine then links to the TSO TIOC Edit routine (IEDAYE) to edit the message. Upon return, this routine translates the message to line code, sets the priority and CCW in the LCB, and exits to the Dispatcher chain function to tpost the ERB to the Activate-I/O Generator subtask (IEDQKA).

The TSO Message Generation routine also provides simulated attention support for 2260 devices. For remote 2260s, it indicates a write erase command if requested. For local 2260s, it indicates a write erase command, and sets the data address and count in the CCW, if erase is requested.

External Routines:

- IEDOTNT - Termname Table code - to get the terminal entry from the terminal or line index in the LCB.

- IEDQAE - Locate Option Field Address routine - to locate the translation table option field for the TRANLIST macro (via the User Interface routine - IEDQUI).

- IEDAYE - TSO TIOC Edit routine - to edit the message.

Tables/Work Areas: CVT, Time Sharing CVT, AVT, LCB, QCB, SCB, DCB, TSB, TSID, terminal entry, TPRIOR, ERB.

Attributes: Reentrant, refreshable, enabled, resident, problem program mode.

TSOUTPUT Routine (Chart YO)

Module Name: IEDAYO

Entry Points:

- IEDAYO - activated by the TCAM Dispatcher when a TPUT macro is issued to move data from TSO buffers into TCAM buffers.

- IEDAYO02 - activated by the TCAM Dispatcher to return TSO buffers to the TSO available buffer queue.

Functions: The TSOUTPUT routine supervises the movement of TSO data from TSO buffers into TCAM buffers. The routine must get empty TCAM buffers, fill these buffers with TSO data, return the TSO buffers to the TSO available buffer queue, and route the full TCAM buffers to the appropriate TSO terminal fcr cutput.

When a user issues a TPUT instruction tc request output of a message, the TPUT SVC mcves the message into TSO buffers and turns on the QCBTPUT bit in the Destination QCB. This causes the Send Scheduler to tpost an ERB to the TSOUTPUT QCB to request TCAM buffers for the message from TSO.

The TSOUTPUT routine satisfies this initial request by constructing the required number of TCAM buffers from units taken from the buffer unit pool. The routine determines the number of units per buffer according to the buffer size specified in the DCB cr in the Terminal Table entry and determines the number of buffers for the initial request from the value specified in the DCB BUFOUT parameter.

If while building the TCAM buffers, the TSOUTPUT rcutine cannot obtain units, it places the ERB on the Buffer Return QCB element chain and exits tc the TCAM Dispatcher. When a unit is available, the Buffer Return routine (IEDQGD) tposts the unit to the TSOUTPUT QCB at its secondary entry pcint - IEDAY002. If, in this situation, the LCBERROR bit in the LCB is off, the TSOUTPUT routine continues to build buffers. If, however, the error bit is on, the routine tposts the unit and any part of a TCAM buffer that is built to the Buffer Return QCB to free the units, tposts the ERB to the address in the LCBRCQCB field, and exits to the TCAM Dispatcher.

As each unit is obtained, the TSOUTPUT routine branches to the TSO TIOC Edit routine (IEDAYE) to move data frcm the TSO buffers into the unit. The TSO TICC Edit routine also edits the data frcm carriage control characters, reserve characters, new line characters, simulated attention characters, and EOT characters. The TSC TIOC Edit routine issues a QTIP SVC to update the data count in the TSO buffer(s) from which it has mcved data. This routine then places one of the following return codes in register 15 and returns to the TSOUTPUT routine:

X'00' - A complete TSO message has beer mcved.

X'0C' - A complete TSO message has not been moved and the terminal line is nct filled, that is, not ready for transmission.

X'10' - A complete TSO message has not been moved, but the terminal line is filled.

When the TSCUTPUT routine regains ccntrcl from the TSO TIOC Edit routine, it examines the return cɔde and executes according to that value.

- Return code of X'00'

When a complete TSO message has been mcved into TCAM buffers, the TSOUTPUT routine examines the TSB to determine whether automatic prompting is specified. If so, the routine sets the corresponding indicator in the SCB (SCBALN). The routine then sets the "end of message" indicator in the LCB, tposts the ERB with the full buffers to the Activate QCB to initiate sending, and exits to the TCAM Dispatcher.

- Return code of X'0C'

  When a complete TSO message has not been moved and the terminal line is not filled, the TSOUTPUT routine determines whether a complete TCAM buffer has been constructed. If not, the routine gets another unit from the buffer unit pool and repeats the procedure to fill it. When a complete buffer has been built and there is still part of the TSO message to be moved, the TSOUTPUT routine determines whether the maximum number of TCAM buffers allowed for this request has been reached. If not, the routine continues to build buffers. Otherwise, the routine sends the buffers already constructed by tposting the ERB to the Activate QCB. If dynamic buffering is not specified (PCI specified in the DCB), the routine sets the "end of message" indicator in the LCB (LCBEOMSG) before exiting to the TCAM Dispatcher.

- Return code of X'10'

  When a complete TSO message has not been moved and the terminal line is filled, the TSOUTPUT routine tposts the ERB with the data in the TCAM buffer(s) and/or part of a buffer to the Activate QCB and then exits to the TCAM Dispatcher.

The PCI Appendage activates the TSOUTPUT routine when the LCBEOMSG bit is off and dynamic buffering is specified. The appendage tposts to the TSOUTPUT QCB an ERB to request more TCAM buffers when the data in the current buffers is successfully transmitted. The TSOUTPUT routine p ocesses this request the same as an initial request except that the TSOUTPUT routine individually tposts each TCAM buffer built to the appropriate TCAM message handler. For all except the first PCI request, the routine can tpost only one TCAM buffer before giving control to the TCAM Dispatcher. On subsequent PCI requests when the LCBERBCT bit is not equal to zero, the TSOUTPUT routine must tpost its own QCB to itself before branching to the TCAM Dispatcher.

The Send Scheduler activates the TSOUTPUT routine when the data in the TCAM buffers has been successfully sent and the LCBEOMSG bit is on. The scheduler tposts to the TSOUTPUT QCB the last TCAM buffer from which data was sent. The TSOUTPUT routine frees the empty TSO buffers and removes any wait conditions that are relieved by freeing these buffers. If the LWAIT can be relieved or if the TSINPUT routine (IEDAYI) is holding TCAM buffers, the TSOUTPUT routine tposts the TSINPUT QCB to itself. Next, if there are more messages to send or if the user has hung up, the routine leaves the TCBTPUT bit on so that the Send Scheduler will tpost the TCAM buffer to the Buffer Return QCB and returns control to the TCAM Dispatcher.

Note from the discussicn in the preceding paragraph that as long as the QCBTPUT bit is cn, that is, as lcng as there is data on the output queue, "initial" requests for TCAM tuffers continue to occur. In this way a TSO message that is toc lcng to be contained in the number of TCAM tuffers allowed in an "initial" request (in one output line), are completely sent even if dynamic tuffering is nct specified.

The Send Scheduler also activates the TSOUTPUT routine when there is no data on the output queue. In this case, if a break-in message has just been sent and there is a partial input message on the input queue to be sent to prcmpt the user (TSBBRKIN is on), the TSOUTPUT routine issues a QTIP SVC to indicate that the partial input situation is handled (TSBBRKIN is off and TSBBIPI is on) and to put the partial input message on the output, as well as the input, queue. After this, the message is prccessed like any cther message except that when the message is successfully sent, the TSOUTPUT routine leaves the TSO buffers on the input queue (does nct free them).

When the Send Scheduler activates the TSOUTPUT routine with no data on the cutput queue and the TSBBRKIN bit is off, the TSOUTPUT routine determines whether the user is being logged off (TSBDISC is on). If so, the routine activates CPB Initialization (IEDQFA) to restore the TCAM control blocks and to facilitate another lcg on. On return, the TSOUTPUT routine restores the TSO ccntrol blocks and determines whether there are any abnormal conditions that require special messages to be sent to the terminal user. If a special message is required, the routine exits to the TSO Message Generation routine (IEDAYM) to transmit the message. If no log off message is required, the TSOUTPUT routine determines whether automatic prompting should be started (TSBSTAUT is on). If so, the rcutine turns off TSBSTAUT and exits to the TSO Message Generation routine, which sends the automatic prcmpt message. If there are no special messages to be sent, the TSOUTPUT routine tposts the LCB tc itself and exits to the TCAM Dispatcher.

When an ERB is tposted to the TSOUTPUT QCB and the recall indicator (LCBRCLIN) is on, an input/cutput error has occurred. For an input error (LCBRECVN is on), the TSOUTPUT routine reinitializes the TCAM buffer tc receive the input message again, tposts the ERB to the address in LCBRCQCB, and exits to the TCAM Dispatcher. For an output error (LCBSENDN is on), the TSOUTPUT routine issues a QTIP SVC to reinitialize the TSO buffers that ccntain the message to be resent to appear as they did originally on the "initial" request. At this point, the TSOUTPUT routine processes the message like an initial request, except that the rcutine tposts the ERB to the address in LCBRCQCB, rather than to the Activate QCB, tefore returning ccntrol to the TCAM Dispatcher.

On any "initial" request, the TSOUTPUT routine performs the following tests and functions:

1. If TSO is atending (AVTTSAB is cn), the TSOUTPUT routine exits to CPB Initialization to initialize the TCAM ccntrol blccks and, on

return, sends an abend message to the terminal (via the TSO Message Generation routine).

2. If a hardware attention has occurred (SCBATTN is on), the TSOUTPUT routine tposts the LCB to the TSO Attention routine (IEDAYA) and exits to the TCAM Dispatcher.

3. If a hang up situation exists (TJBHUNG is on), the TSOUTPUT routine clears the input and output queues, activates CPB Initialization to initialize the TCAM control blocks, initializes the TSO control blocks, and exits to the TCAM Dispatcher.

On subsequent or PCI requests, the TSOUTPUT routine checks for a hardware attention (SCBATTN is on). If a hardware attention is indicated, the TSOUTPUT routine puts zeros in the ERB buffer chain pointer field and exits to the TCAM Dispatcher. Otherwise, if the LCBERROR bit is on, the routine tposts the ERB to the address in LCBRCQCB and exits to the TCAM Dispatcher.

When a display station user enters data on the last or next-to-last line of the screen, the data is erased. In this situation, the TSINPUT routine sets the TSBBRKIN bit in the TSB. When the TSOUTPUT routine gains control and this bit is on, the routine must resend the last input message or, if the message is long, the last full input line and any fragment of a line. The TSOUTPUT routine handles this situation like a partial input line with a break-in except that after handling the display message the routine turns off the TSBBIPI bit. This differentiates the display message situation from a real break-in situation.

External Routines:

- IEDAYE - TSO TIOC Edit routine - to move data from TSO to the TCAM buffers and to edit the data for control characters.

- IEDQFA - CPB Initialization routine - to perform TCAM cleanup processing.

- OTIP SVC - TSO SVC - to update the data count in the TSO buffers.

Tables/Work Areas: AVT, CVT, DCB, ERB, ICB, buffer prefix, QCB, SCB, STCB, TIOCBUF, TIOCRPT, TJB, Terminal Table, TSB, Time Sharing CVT, TSI.

Attributes: Reentrant, refreshable, reusable.

STARTMH Subtask for TCAM-TSO Mixed (Chart YR)

Module Name: IEDAYR

Entry Point: IEDQAA01 - activated by the TCAM Dispatcher when a buffer is tposted to the STARTMH QCB to initialize the buffer before sending it through a TCAM or TSO message handler.

Functions: The STARTMH subtask for TCAM-TSO Mixed initializes a buffer before sending it through a TCAM or TSO message handler. The functions vary depending upon whether the buffer is a header or a text buffer, and whether it is to be processed by the incoming or outgoing side of the message handler.

The STARTMH subtask for TCAM-TSO Mixed first checks for a recalled buffer. If the buffer is recalled, the subtask performs no initialization but does check for translation. If the buffer is an outgoing buffer and the destination terminal is in lock mode, the subtask increments the LCB count of outstanding lock responses (LCBINCAM) and turns off the prefix lock bit (PRFEOFF). If the buffer, incoming or outgoing, is a TCAM/TSO buffer, the subtask initializes the LCB reserve count (LCBISZE) to zero. Then the subtask checks to see if the buffer is a header or a text buffer. For all header buffers the subtask sets the address in the SCB multiple-buffer-header entry (SCBMBHEN) to zero. Next, the TSO STARTMH subtask checks to see if the buffer is to be processed by an incoming or outgoing MH. The subtask then tailors the processing to each of the four situations.

For an incoming header buffer, the STARTMH subtask for TCAM-TSO Mixed initializes the prefix field (PRFSRCE) from the current terminal index in the LCB (LCBTTCIN). The subtask then clears the SCB priority (SCBPRI) and cutoff count (SCBBKFCT) to zero. If the buffer is a TCAM/TSO buffer, the subtask branches to the exit code. If the buffer is strictly a TCAM buffer, the subtask sets the prefix scan pointer (PRFSCAN) to point to the last byte of the prefix, or to the last reserve character, if specified. If the source of the message was an application program, the subtask branches to the exit code. Otherwise, the subtask checks to see if an EOA sequence was defined in the Special Characters Table. If the sequence was defined, the subtask links to the Skip Forward and Scan routine (IEDQAI) to see if the EOA sequence is in the buffer. If the EOA sequence is in the buffer, the subtask sets the prefix scan pointer to point beyond it. For 1030s and Remote 2260s the subtask moves the prefix scan pointer beyond the addressing character that follows the EOA sequence for these devices. The subtask then determines if the terminal is in lock mode and if so branches to the exit code. Otherwise, the subtask checks to see if the buffer contains an On-Line Test message. For binary synchronous (BSC) lines the subtask checks a bit (LCBSYNC) in the LCB; for start/stop lines the subtask links to the Skip Forward and Scan routine to see if an On-Line Test sequence is in the buffer. If the buffer contains an On-Line Test message, but On-Line Test is not in the system, the subtask sets a bit in the SCB error word (SCBOLTR) and branches to the exit code. If On-Line Test is in the system, the subtask sets the On-Line Test priority bit (PRIONLT) in the buffer prefix and exits to the Dispatcher post function to tpost the buffer to the On-Line Test QCB. If the buffer does not contain an On-Line Test message, the subtask branches to the exit code.

For an incoming text buffer, the STARTMH subtask for TCAM-TSO Mixed initializes the prefix source field as for an incoming header buffer. If the MH to receive the buffer is a TSO MH, the subtask checks to see if the destination is in the TSINPUT QCB. If the destination is not in the QCB, the subtask checks for a LOGON exit. If one is not present, the subtask branches to the code for a translation check. If a LOGON exit is present, the subtask establishes a new QCB and MH, and branches to the new MH. If the MH to receive the buffer is not a TSO MH, the subtask initializes the LCB reserve count from the count specified in the DCB at DCBRESER+1, and sets the prefix scan pointer to point to the last byte of the text prefix, or to the last reserve character, if specified. Then the subtask branches to the translation check code.

For an outgoing header buffer, the STARTMH subtask for TCAM-TSO Mixed immediately branches to the exit code if it is a TCAM/TSO buffer. If the buffer is a TCAM buffer, the subtask checks for a LOGON exit. If one is present, the subtask sets up and branches to the new MH as described above. If no LOGON exit is present, the subtask performs FEFO updating. Normally, the subtask updates the FEFO pointer in the Destination (priority) QCB, and turns off the 'currently sending' flag in the Master QCB. However, there are three situations in which this is not done; (1) if the destination terminal is in lock mode with an application program, the subtask cannot update the FEFO pointer until the terminal is unlocked; (2) if the line is in initiate mode, the subtask cannot update the FEFO pointer until the last buffer of the initiate message has been processed through the MH; (3) if a text transfer error generated a zero-length buffer, the subtask cannot update the FEFO pointer because the buffer will not be sent to the destination. If the destination terminal is in both lock and extended lock modes, the subtask turns off both lock bits (SCBLCKIN and SCBMSGLN). After it has completed any FEFO updating, the subtask sets the LCB reserve count to zero for zero-length buffers, or to the value in the scan pointer for non-zero-length buffers. The subtask sets the prefix scan pointer to point to the last byte of the header prefix, or to the last reserve character, if specified. In addition, for non-zero-length buffers the subtask links to the Termname Table code (IEDQTNT) to get the terminal entry, saves the current output sequence number in the SCB (SCBOSEQ), and increments the output sequence number in the terminal entry (TRMOUTSQ). If this sequence number then exceeds the maximum output sequence number, the subtask branches to the exit code.

For an outgoing text buffer, the STARTMH subtask for TCAM-TSO Mixed immediately branches to the translation check code if the buffer is a TCAM/TSO buffer. Otherwise, the subtask checks for a LOGON exit. If one is present, the subtask sets up and branches to the new MH as described above. If no LOGON exit is present, the subtask initializes the LCB reserve count to zero, sets the prefix scan pointer to point to the last byte of the text prefix, and branches to the translation check code described below.

The STARTMH subtask form TCAM-TSO Mixed handles buffer translation for the following types of buffers: (1) incoming text buffers directed to a non-TSO MH; (2) incoming text buffers directed to a TSO MH without a LOGON exit; (3) outgoing TCAM (non-TSO) header buffers; and (4) all recalled buffers. The translation check code first checks for a zero-length buffer. A zero-length buffer is not translated. Next the translation check code checks the SCB to see if a translation was requested. If no request was made, the buffer is not translated. If a request was made, the subtask links to the Translate Buffer routine (IEDQAW), via the User Interface routine (IEDQUI), to translate the buffer. If the SCB multiple-buffer-header field is zero, the subtask branches to the exit code. If not, the subtask indicates the presence of the multiple-buffer header by setting a negative value in register 0 and loading the address of the User Interface routine in register 15 before branching to the exit code.

The exit code of the STARTMH subtask for TCAM-TSO Mixed gets the address of the first instruction in the MH and sets an increment of 4096 in register 2 for multiple base register support. If a multiple-buffer header is present, the subtask sets a condition code of 4 in the PSW. Otherwise it sets a condition code of 1 in the PSW if the line is sending or 8 if the line is receiving. These condition codes are tested by the code generated by the STARTMH macro instruction. The subtask then exits to the first executable instruction of the MH.

External Routines:

- IEDQAI - Skip Forward and Scan routine - to scan for an EOA sequence or On-Line Test sequence in the buffer.

- IEDQAW - Translate Buffer routine - to translate the buffer.

- IEDQTNT - Termname Table code - to get the terminal entry for the destination of an outgoing header buffer.

- IEDQUI - User Interface routine - to link to IEDQAW to translate the buffer and to re-enter an uncompleted routine.

Tables/Work Areas: AVT, LCB, SCB, DCB, DEB, UCB, QCB, buffer prefix, terminal entry, Special Characters Table, TPRIOR, ERB.

Attributes: Serially reusable, refreshable, enabled, resident, problem mode.

TSO Simulated Attention Routine (Chart YS)

Module Name: IEDAYS

Entry Points:

- IEDAYS - receives control from the TCAM User Interface routine (IEDQUI) to handle simulated attention by character string.

- IEDAYS2 - receives control from the TSO TIOC Edit routine (IEDAYE) to handle simulated attention by line count.

- IEDAYS3 - receives control from the TCAM Dispatcher (IGG019RB) to handle simulated attention by time delay interval.

Functions: The TSO Simulated Attention routine handles simulated attention for TSO. The three ways of simulating attentions are (1) by character string, (2) by line count, and (3) by time delay interval.

For simulated attention by character string, the TSO Simulated Attention routine first checks to see if the Time Sharing buffer was read by the Simulated Attention Read channel program. If it was not, the routine checks to see if the device is a 2260. If the device is a 2260 and the screen is full, the routine sets the "erase display request" bit in the SCB error word. The TSO Simulated Attention routine then scans the buffer for a clear character string, which indicates that the screen is to be erased immediately. If this string is present, the routine immediately tposts the buffer to Buffer Return, and no additional simulated processing takes place.

If the device is not a 2260, or if no clear character string is present in the buffer, the TSO Simulated Attention routine checks the QCB to see if simulated attention by character string was requested. If it was not, the routine returns control to the Return Interface routine. The TSO Simulated Attention routine also returns control immediately if a simulated attention by character string was requested but no valid simulated attention character string is found in the buffer. If the routine finds the character string, it checks to see if a valid attention level (a number 1 through 9 followed by a carriage return or a new line symbol) was also entered. If it was entered, the routine sets the attention level in the SCB. An invalid attention level causes the TSO Simulated Attention routine to immediately return to the Return Interface routine. After setting the attention level, this routine sets on the "simulated attention request" bit in the SCB error word, tposts the buffer to Buffer Return, and returns control via the Return Interface.

If the buffer was read by the Simulated Attention Read channel program, the TSO Simulated Attention routine turns off the "simulated attention read" bit in the Destination QCB. In this case, a simulated attention character string is not required. If one was entered, the routine checks it for validity, and also checks the attention level. However, in this case, an invalid character string or attention level does not prevent the buffer from being tposted. Instead, the routine sets an attention level of zero and tposts the buffer and returns as stated above. Also, in this case, the routine automatically sets the "erase display request" bit for 2260s, but makes no check for a clear character string.

For simulated attention by line count, the TSO Simulated Attention routine counts each physical output line and keeps the count in the Destination QCB. When the threshold specified by the user in the Terminal Status Block is reached, the routine requests a special simulated attention channel program by setting on the "simulated attention read request" bit in the QCB and setting the output line count to zero before returning to the Edit routine. This is not done when the device is a 2260.

For simulated attention by time interval, the TSO Simulated Attention routine receives control from the Dispatcher when a QCB is removed from the time delay queue. If simulated attention by time delay was not requested, the TSO Simulated Attention routine returns control to the Dispatcher. Otherwise, the routine issues a QTIP SVC (SVC 101) with entry code 26 to turn off the "QCB tposted" flag and determine if a TPUT has been requested. If a TPUT is requested, the TSO Simulated Attention routine tposts the QCB to itself and returns control to the Dispatcher. Otherwise, the routine requests a simulated attention read. If the LCB is free, the routine removes the LCB from the time delay queue and tposts it to itself.

External Routines:

- IEDQTNT - Termname Table code - to get the terminal entry from the index in the LCB.

- QTIP SVC (101) - entry code 26 - to turn "QCB tposted" flag off and determine if TPUT is requested.

Tables/Work Areas: CVT, Time Sharing CVT, TSB, LCB, SCB, AVT, QCB, TSID, terminal entry, current buffer prefix.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

TSO Abend Interface Routine (Chart YT)

Module Name: IEDAYT

Entry Points:

- IEDAYT0 - receives control from the OS Supervisor when TCAM abends.

- IEDAYT1 - receives control from the OS Supervisor when an attached task abends.

- IEDAYT2 - receives control from the EXCP Driver (IGG019RC) when no space is available on a nonreusable disk queue and a flush closedown has been initiated.

Functions: The Abend Interface routine informs TSO when TCAM abends, when an attached TCAM task abends, or when, in a mixed TSO/TCAM environment, the EXCP Driver abends because no space is available on a nonreusable disk queue and a flush closedown has been initiated.

If TCAM abends, the Abend Interface routine issues the TCABEND macro instruction, which turns off the TCAM ready bit in the CVT. If the CVTTSRDY bit in CVTTSFLG is on, indicating that TSO is active, the TCABEND macro instruction also indicates that the Time Sharing Control task (IKJEAT03) is to stop by turning on the TCASTOP bit in the Time Sharing CVT, and by posting the Time Sharing Controller ECB in TSECBTAB complete.

If an attached TCAM subtask abends, the Abend Interface routine gains access to all the elements that were tposted to the abending subtask from the ready queue and the task QCB and disposes of them - that is, buffers are placed on the Buffer Return queue and LCBs are tposted to themselves. If the abending task is either Checkpoint or On-line Test, the Abend Interface routine clears the QCB STCB link address, sets the QCB element chain to point to the dummy last element address in the AVT, and resets the QCB flag to indicate that it is a QCB. This is done to indicate that the abending task no longer exists in the system. If the abending task is Operator Control, the procedure is the same as when the EXCP Driver abends.

If, in a mixed TSO/TCAM environment, the EXCP Driver abends because no space is available on a nonreusable disk queue and a flush closedown has been initiated, the TSO Abend Interface routine issues the AQCTL SVC 102 to tpost the Abend routine ECB to the ready queue. It also issues the TCABEND macro instruction, which performs the same functions as when TCAM abends.

External Routines:

- IGC102 - AQCTL SVC 102 routine - to tpost the Abend Interface subtask ECB to the ready queue.

- TCABEND (SVC 94) - to turn off the TCAM ready bit and stop the Time Sharing Control task.

Tables/Work Areas: AVT, QCB, CVT, TSID, STAE work area.

Attributes: Serially reusable, enabled, problem program mode.

TSO INMSG/OUTMSG Linker (Chart YX)

Module Name: IEDAYX

Entry Point: IEDAYX - receives control from Buffer Disposition (IEDOBD) to provide linkage to the TSO Attention or the TSO Hangup routine when ATTEN or HANGUP macro instructions are coded in the INMSG or OUTMSG subgroups.

Functions: The TSO INMSG/OUTMSG Linker provides linkage to the TSO Attention and TSO Hangup routines (IEDAYA and IEDAYH, respectively). If the linkage is to the TSO Hangup routine, the TSO INMSG/OUTMSG Linker obtains the address of the routine from the macro expansion that was passed. If the linkage is to the TSO Attention routine, the Linker obtains the address of the routine from the STARTMH macro pointed to in the DCB. Before it branches to either routine, the TSO INMSG/OUTMSG Linker sets the ERB priority and tposts the ERB to Buffer Disposition.

External Routines: None.

Tables/Work Areas: AVT, LCB, DCB.

Attributes: Serially reusable, refreshable, enabled, resident, problem program mode.

TSO Asynchronous Time Delay Removal Routine (Chart YY)

Module Name: IEDAYY

Entry Point: IEDAYY - receives control when it is tposted asynchronously from TIOC modules to remove QCBs from the time delay queue.

Functions: The TSO Asynchronous Time Delay Removal routine removes QCBs from the time delay queue when a send or receive operation is to be initiated. The routine first checks to see if it has been activated (tposted to the ready queue) by a TIOC module. If it has not been activated, the TSO Asynchronous Time Delay Removal routine returns control to the Dispatcher to dispatch the next subtask. If it has been activated, the routine deactivates itself, by marking its QCB not tposted, and obtains the addresses of the previous and current elements on the time delay queue. The routine then scans the time delay queue looking for a Time Sharing QCB with both the Write Break flag and the TPUT request flag on. Each time it finds such a QCB, the TSO Asynchronous Time Delay Removal routine performs the following functions. It updates the link field of elements on the time delay queue. It sets the time delay flag in the QCB to indicate that it is no longer in the time delay queue. It tposts the QCB to itself to initiate a send operation. The routine then loads the tpost register (R1) with the address of the QCB to be tposted to the ready queue and links to the Dispatcher tpost function to tpost the QCB.

When the TSO Asynchronous Time Delay Removal routine receives control again from the Dispatcher, it continues to scan the time delay queue, looking for the next QCB to be removed. When it has searched the entire queue, and has removed all the applicable QCBs, the TSO Asynchronous Time Delay Removal routine branches back to the beginning to see if another interrupt has occurred to reactivate it. If no such interrupt has occurred, the routine returns to the Dispatcher. If an interrupt has occurred, the routine repeats the entire procedure outlined above.

External Routines: IGG019RB or IGG019RO - the TCAM Dispatcher - to tpost the QCB(s) to the ready queue by priority.

Tables/Work Areas: AVT, QCB, TSIE.

Attributes: Serially reusable, refreshable, enabled, prcblem program mode.

Time Sharing Scheduler (Chart YZ)

Module Name: IEDAYZ

Entry Points:

- AYZ000 - activated by the Leased Receive Scheduler (IGG019R3) to determine whether or not to initiate a Read operation.
- AYZ100 - activated by the Dial Receive Scheduler (IGG019R1) to determine whether or nct to initiate a Read operation.
- AYZ200 - activated by the Line End Appendage (IGG019R0) when a negative response to polling has been received to determine whether or not ancther poll operaticn is desired.
- AYZ300 - activated from the QEVENT subroutine in the Leased Receive Scheduler (IGG019R3) to generate a Prepare channel program when a line ccnnected to a 2741 is to be freed.
- AYZ400 - activated by the Send Scheduler (IGG019R4) when it is dispatched from the QCB to determine whether a Write Break channel command can be issued.
- AYZ410 - activated by the Time Sharing Destinaticn Scheduler (IEDAYD) to determine whether or not a Write Break channel command can be issued.
- AYZ500 - activated by the Send Scheduler (IGG019R4) when it is dispatched from the LCB to determine whether or nct to initiate a send operaticn.
- AYZ600 - activated by the Activate-I/C Generator subtask (IEDQKA) before it builds an input or output channel program.

Functions: The function of the Time Sharing Scheduler varies depending upon the routine from which it receives control and at which entry point.

If the Time Sharing Scheduler is entered from the Leased Receive Scheduler (IGG019R3) at entry point AYZ000, the scheduler links to the Termname Table code (IEDQTNT) to get the terminal entry, from which it gets the address of the Destination QCB. If the terminal is not dedicated to a Time Sharing session, the scheduler updates the LCB pointer to the current invitation list entry and returns ccntrol to the Leased Receive Scheduler. If a terminal is dedicated to a Time Sharing sessicn, the Time Sharing scheduler checks various scheduling bits in the OCB tc determine whether to initiate a read operation. If a hardware attention interrupt has been received, the scheduler turns off the "simulated attention read request" bit (QCBSATRD) in the QCB and exits to the Dispatcher post functicn tc tpost the ICB to the TSO attention routine specified in MH. The TSC Attenticn routine (IEDAYA) activates TSO attention exits or indicates that certain attention controled functions (such as line deletion) should be activated. When the Time Sharing Scheduler decides to begin a read operation, it first determines whether any requests for a simulated attention read override the read operaticn. If so, the scheduler initiates the special simulated attention read channel prcgram.

If no read operation is to be started, the Time Sharing Scheduler checks to see if a simulated attenticn by time delay interval was requested. If this simulated attenticn was requested, the scheduler checks to see if a send operaticn was requested or if the QCB is already in the time delay queue. If neither of these conditions exists, the scheduler sets the pricrity, the time delay interval, and the time delay flag in the QCB and links tc the Time Delay subtask (IEDQHG) to insert the QCB into the time delay queue. The scheduler then updates the invitation list pointer to point to the next entry, updates the ICB pointer to the currently connected terminal, and branches back to the beginning to get the terminal entry and QCB address for the new terminal and to repeat the procedure for it. When the Time Sharing Scheduler reaches the end of the invitation list, it checks to see if all entries were polled. If they were, the scheduler sets the "start of polling list" bit and returns control tc the Leased Receive Scheduler. If no entries were pclled, the scheduler sets the invitation list pointer to point to the first entry and exits to the Dispatcher bypass functicn to immediately activate the next STCB in the LCB chain.

If a read operation is to be started, the Time Sharing Scheduler checks to see if the QCB is in the time delay queue. If it is, the scheduler links to the Time Delay subtask (IEDQHG) tc remcve it from the queue. Finally, the scheduler turns cn the "time sharing buffer prefix" bit (LCBTSBUF) in the LCB, updates the ICB invitation list pointer, and returns ccntrcl to the Leased Receive Scheduler.

When the Time Sharing Scheduler is entered from the Dial Receive Scheduler (IGGC19R1) at entry pcint AYZ100, the scheduler makes the same tests as above to determine whether cr not to initiate a read operation. The scheduler does the same processing to insert or remove a QCB from the time delay queue or to tpost the LCB tc the TSC Attention routine.

If no regular read operation is to be started, but a simulated attention read was requested, the Time Sharing Scheduler turns off the "negative respcnse to polling" bit (LCBNFGRF) in the LCB and initiates the special simulated attention read channel program. Otherwise, after performing any necessary time delay frocessing, the scheduler frees the line, puts up a Prepare channel program on the line to monitor for an attention interrupt, turns off the "send priority" bit (LCBSNDPR) and the "negative resronse to pclling" bit in the LCB, and exits to the Dispatcher dispatch function to dispatch the next subtask.

If a regular read operation is to be started, the Time Sharing Scheduler performs any necessary time delay processing and checks to see if a negative response to polling was received on the last poll. If a negative response was received, the scheduler sets the LCB to re-poll, and checks to see if the polling delay interval in the DCB is zero. If the polling delay is not zero, the scheduler puts up a Prepare channel program to monitor for an attention interrupt and exits to the Dispatcher post function to post the LCB to the Time Delay QCB. If the polling delay is zero, if a negative response to polling was not received, or if a simulated attention read operation is to be started, the Time Sharing Scheduler turns on the "time sharing buffer prefix" bit in the LCB and returns control to the Dial Receive Scheduler. When the entry is from the Dial Receive Scheduler the Time Sharing Scheduler does not update the invitation list pointers.

The Time Sharing Scheduler is entered at entry point AYZ200 from Line End Appendage (IGG019R0) when a negative response to polling is received. The scheduler first links to the Termname Table code (IEDQTNT) to get the terminal entry from which it extracts the Destination QCB address. The scheduler also gets the address of the last buffer from the LCB. If neither the last entry polled nor the next entry to be polled is dedicated to time sharing, the Time Sharing Scheduler turns off the "start of polling list" bit (LCBSCPL) in the LCB and returns control to Line End Appendage to poll the next entry in the invitation list.

If the next entry to be polled is dedicated to time sharing, the Time Sharing Scheduler checks the QCB scheduling bits to determine whether or not to poll it. If the entry is not to be polled, the scheduler returns control to Line End Appendage to update the invitation list pointer to point to the next entry. If a simulated attention by time delay interval was requested, the scheduler links to the Time Delay subtask to insert the QCB into the time delay queue before returning to Line End Appendage.

If a TSO-dedicated entry is to be polled and its QCB is in the time delay queue, this routine links to the Time Delay subtask to remove it. Then the scheduler checks for both TSO and TCAM entries to see if the last entry polled was also dedicated to TSO or TCAM. If both entries are not dedicated to the same system, the Time Sharing Scheduler adjusts the addresses, counts, and in some cases the op codes, in the CCWs for all buffer units to meet the requirements of the current system. The scheduler flags each buffer prefix as either a TSO buffer or a TCAM buffer. For TSO buffers, the scheduler turns on the "time sharing buffer prefix" bit in the LCB. After it has made all the adjustments, the scheduler returns control to Line End Appendage to poll the next entry.

Because this scheduler is disabled when it is entered from Line End Appendage, special enabled code is included to link to the Time

Delay subtask, schedule an enabled reentry to the Time Sharing Scheduler, and exit to the Dispatcher. The Dispatcher, in turn, returns control to the scheduler in the enabled state.

The Time Sharing Scheduler is entered at AYZ300 from the QEVENT entry point in the Leased Receive Scheduler (IGG019R3) when TSO is in the system. QEVENT is dispatched as the last STCB in the chain of scheduler STCBs for a line. After it links to the Termname Table code and gets the address of the DCB, invitation list, and Destination QCB, the scheduler checks to see if the following conditions are met: (1) the terminal is dedicated to a time sharing session; (2) a Prepare is not already up on the line; (3) the terminal has the Attention feature; (4) an attention exit was specified in the MH; and (5) the invitation list consists of only one entry. If all these conditions are not met, the scheduler returns control to the QEVENT routine. The Time Sharing Scheduler next checks to see if the terminal is a 2741. If it is not a 2741, the scheduler checks to see if the terminal can time out. If the terminal can time out, the scheduler returns control to the QEVENT routine. If the terminal is inhibited from timing out, the scheduler sets up a Prepare on the line, links to the Activate-I/O Generator subtask (IEDQKA) to build the channel program, and branches to the routine specified below that issues the EXCP.

If the terminal is a 2741, the line is to be freed. If the LCB indicates that a circle D has not been sent to the 2741, the Time Sharing Scheduler generates a write circle D - Prepare Channel program to put the line into receive mode and to monitor for an attention interrupt. If a circle D has been sent, the line is already in receive mode, so the scheduler generates a Prepare channel program. The scheduler puts the CCW starting address in the LCB. Then the scheduler loads the IOB address in register 1, issues the EXCP SVC (SVC 0) to start the channel program, and returns control to the QEVENT routine.

The Time Sharing Scheduler is entered at entry point AYZ400 from the Send Scheduler (IGG019R4) when it has been dispatched off the QCB. The Time Sharing Scheduler first checks to see if a simulated attention read was requested. If it was, the scheduler passes control to the Dispatcher at DSPUNAV to remove the STCB from its current QCB and insert it in the Send Scheduler QCB. If no simulated attention read was requested, the Time Sharing Scheduler checks to see if a Write Break was requested. If not, the scheduler returns control to the Send Scheduler. If a Write Break has been requested, the Time Sharing Scheduler turns off the "read priority" bit (QCBREAD) in the QCB and tests several status bits in the LCB and QCB to determine whether or not it can issue a Write Break channel command. If it cannot issue this channel command, the scheduler returns control to the Send Scheduler. If the terminal is on a leased line, the Time Sharing Scheduler links to the Termname Table code, gets the terminal entry and extracts the QCB address. If the terminal for which the Write Break was requested is not currently connected, the scheduler returns control to the Send Scheduler. Otherwise, the Time Sharing Scheduler gets the address of the first buffer (in LCBLSPCI), turns on the "write break in progress" bit (LCBWRBRK) in the LCB, gets the address of the UCB, and issues an IOHALT SVC (SVC 33). Line End Appendage handles the IOHALT interrupt which causes a Break channel program to be executed. After it issues the IOHALT, the Time Sharing Scheduler passes control to the Dispatcher at DSPUNAV to switch the STCB to the Send Scheduler QCB.

The Time Sharing Scheduler is entered from the Time Sharing Destination Scheduler (IEDAYD) at AYZ410 when a Write Break has been requested. The processing is the same as when entered from the Send Scheduler above, except that the first two checks for simulated attention and Write Break are bypassed, and the Time Sharing Scheduler always returns control to the Time Sharing Destination Scheduler.

The Time Sharing Scheduler is entered from the Send Scheduler at AYZ500 when it has been dispatched off the LCB. If a TSO session is not in progress, and no non-TSO (queuing) functions are to be performed, the Time Sharing scheduler returns control to the Send Scheduler to put the Send Scheduler STCB back in the QCB chain. If a non-TSO output function is to be performed, the scheduler returns control to the Send Scheduler to initiate a send operation. If a TSO session is in progress, the Time Sharing Scheduler checks the various scheduling bits in the QCB to determine whether or not to initiate a TSO send operation. If a send operation is to be performed, the scheduler returns control to the Send Scheduler to do so. If a simulated attention Read or a Read of a partial input line is requested, it takes priority over output. In this case if the LCB is in the time delay queue, the Time Sharing Scheduler links to the Time Delay subtask to remove it. Then the scheduler tposts the LCB to itself and passes control to the Dispatcher bypass function to immediately activate the next STCB in the LCB chain.

The Time Sharing Scheduler is entered at AYZ600 from the Activate-I/O Generator subtask (IEDQKA) before it builds an input or output channel program. The scheduler links to the Termname Table code, gets the terminal entry, and extracts the QCB address. If a TSO session is not in progress, the scheduler immediately returns control to the Activate-I/O Generator module. For input, if a hardware attention was received, the Time Sharing Scheduler tposts the LCB to the TSO attention routine specified in the MH. If a TPUT was requested, the scheduler tposts the LCB to itself. In either case, the scheduler passes control to the Dispatcher chain function to tpost the input buffers to the Buffer Return routine (IEDQGE). If a Prepare is up on the line, the scheduler issues an IOHALT SVC (SVC 33) to halt I/O on the line. Then the scheduler makes another check for a hardware attention, and, if one was received, processes it in the same manner as above. Otherwise, the Time Sharing Scheduler returns control to the Activate-I/O Generator subtask after the IOHALT.

External Routines:

- IEDQTNT - Termname Table code - to get the terminal entry from the terminal or line entry in the LCB.

- IEDQHG - Time Delay suttask - to remove an ICB cr a QCB from the time delay queue, or to insert a QCB into it.

- IEDQKA - Activate-I/O Generatcr subtask - to build a Prepare channel ccmmand.

- IGG019RB - TCAM Dispatcher - at the dispatch entry pcint, to provide for enabled re-entry to IEDAYZ after linking to IEDQHG, when entered in a disabled state.

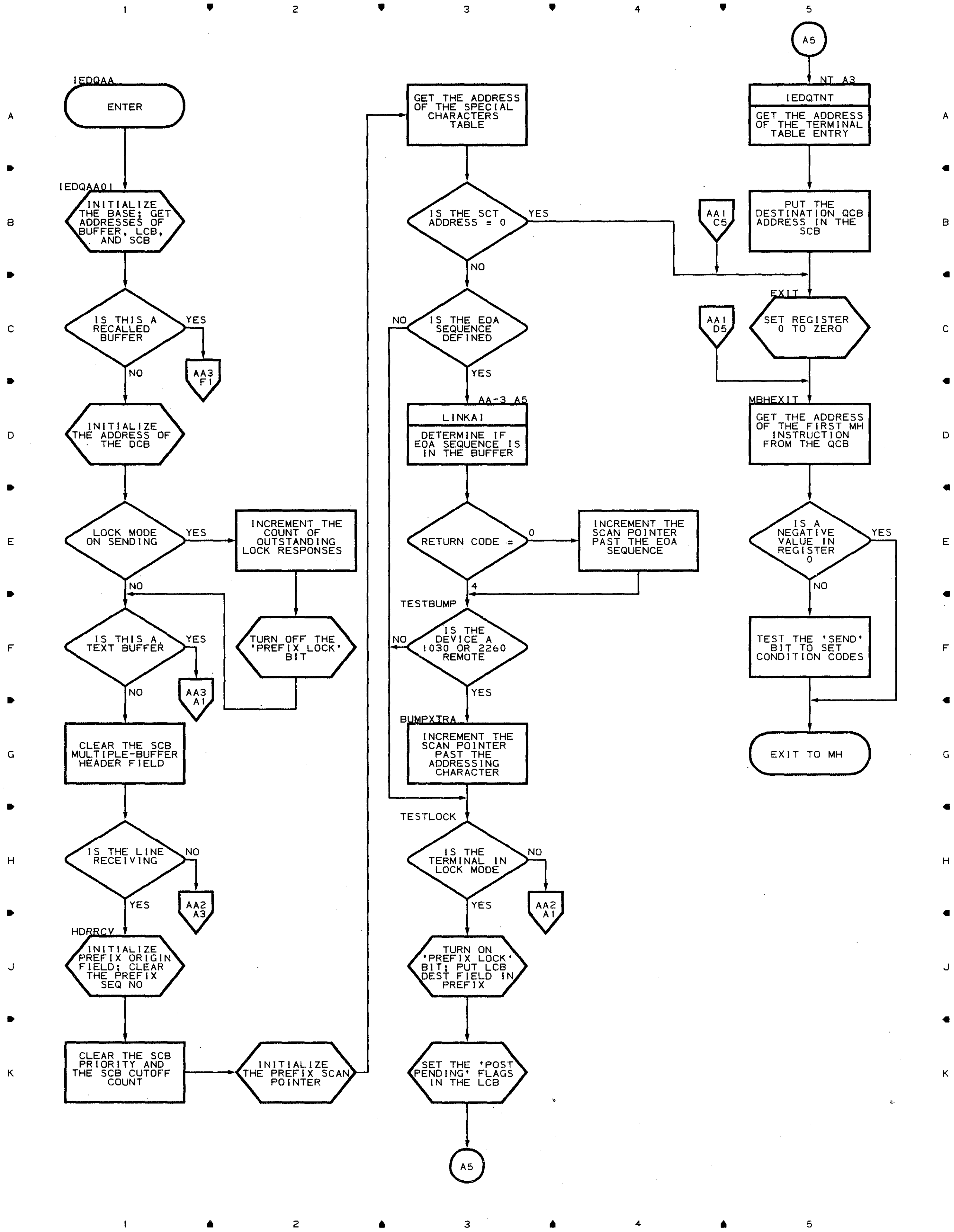
- OS IOHALT routine (SVC 33) - to halt I/C on a line.

- OS EXCP routine (SVC 0) - to start a Prepare channel prcgram.

Tables/Work Areas: CVT, Time Sharing CVT, AVT, TSB, QCB, TSID, LCB, DCB, SCB, tuffer prefix, terminal entry, invitation list, TPRIOR, ERB.

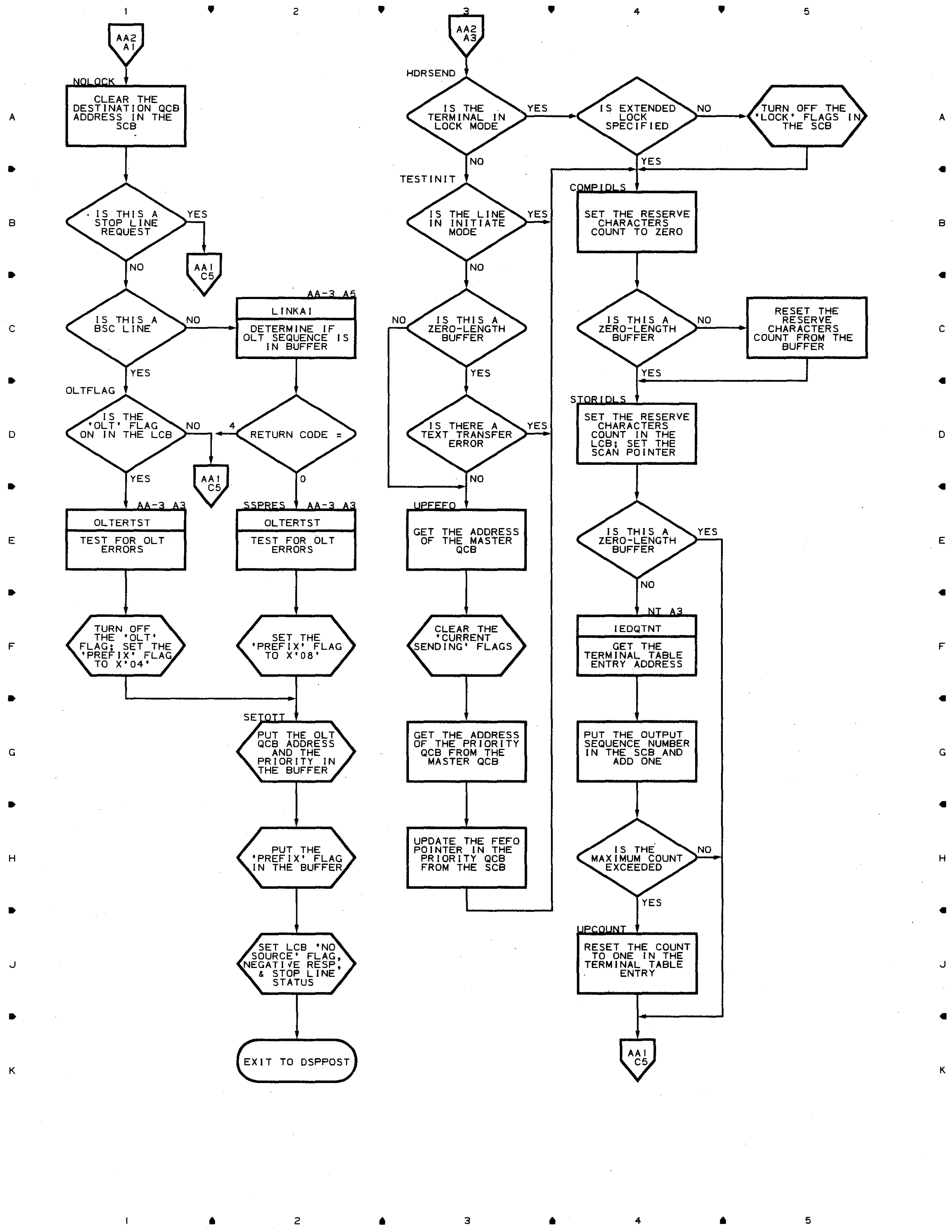
Attributes: Reentrant, disabled when entered from Line End Appendage, otherwise enabled, supervisor mode when entered from Line End Appendage, otherwise problem program mode.

Chart AA-1 STARTMH SUBTASK



1
2
3
4
5
A5
NT A3
IEDQTNT
GET THE ADDRESS OF THE TERMINAL TABLE ENTRY
IEDQAA
ENTER
GET THE ADDRESS OF THE SPECIAL CHARACTERS TABLE
IEDQAA01
INITIALIZE THE BASE; GET ADDRESSES OF BUFFER, LCB, AND SCB
IS THE SCT ADDRESS = 0
YES
NO
AA1 C5
PUT THE DESTINATION QCB ADDRESS IN THE SCB
EXIT
SET REGISTER 0 TO ZERO
AA1 D5
IS THIS A RECALLED BUFFER
YES
NO
AA3 F1
IS THE EOA SEQUENCE DEFINED
NO
YES
AA-3 A5
LINKA1
DETERMINE IF EOA SEQUENCE IS IN THE BUFFER
MBHEXIT
GET THE ADDRESS OF THE FIRST MH INSTRUCTION FROM THE QCB
INITIALIZE THE ADDRESS OF THE DCB
LOCK MODE ON SENDING
YES
INCREMENT THE COUNT OF OUTSTANDING LOCK RESPONSES
RETURN CODE =
0
INCREMENT THE SCAN POINTER PAST THE EOA SEQUENCE
IS A NEGATIVE VALUE IN REGISTER 0
YES
NO
4
TESTBUMP
IS THIS A TEXT BUFFER
YES
NO
AA3 A1
TURN OFF THE 'PREFIX LOCK' BIT
IS THE DEVICE A 1030 OR 2260 REMOTE
NO
YES
TEST THE 'SEND' BIT TO SET CONDITION CODES
BUMPXTRA
CLEAR THE SCB MULTIPLE-BUFFER HEADER FIELD
INCREMENT THE SCAN POINTER PAST THE ADDRESSING CHARACTER
EXIT TO MH
TESTLOCK
IS THE LINE RECEIVING
NO
YES
AA2 A3
IS THE TERMINAL IN LOCK MODE
NO
YES
AA2 A1
HDRRCV
INITIALIZE PREFIX ORIGIN FIELD; CLEAR THE PREFIX SEQ NO
TURN ON 'PREFIX LOCK' BIT; PUT LCB DEST FIELD IN PREFIX
CLEAR THE SCB PRIORITY AND THE SCB CUTOFF COUNT
INITIALIZE THE PREFIX SCAN POINTER
SET THE 'POST PENDING' FLAGS IN THE LCB
A5
A
B
C
D
E
F
G
H
J
K

Chart AA-2 STARTMH SUBTASK



AA2 A1
NOLOCK
CLEAR THE DESTINATION QCB ADDRESS IN THE SCB
IS THIS A STOP LINE REQUEST
YES
AA1 C5
NO
IS THIS A BSC LINE
NO
AA-3 A5
LINKA1
DETERMINE IF OLT SEQUENCE IS IN BUFFER
RETURN CODE =
4
0
YES
OLTFLAG
IS THE 'OLT' FLAG ON IN THE LCB
NO
AA1 C5
YES
AA-3 A3
OLTERTST
TEST FOR OLT ERRORS
SSPRES
AA-3 A3
OLTERTST
TEST FOR OLT ERRORS
TURN OFF THE 'OLT' FLAG; SET THE 'PREFIX' FLAG TO X'04'
SET THE 'PREFIX' FLAG TO X'08'
SETOTT
PUT THE OLT QCB ADDRESS AND THE PRIORITY IN THE BUFFER
PUT THE 'PREFIX' FLAG IN THE BUFFER
SET LCB 'NO SOURCE' FLAG, NEGATIVE RESP, & STOP LINE STATUS
EXIT TO DSPPOST
AA2 A3
HDRSEND
IS THE TERMINAL IN LOCK MODE
YES
IS EXTENDED LOCK SPECIFIED
NO
TURN OFF THE 'LOCK' FLAGS IN THE SCB
YES
NO
TESTINIT
IS THE LINE IN INITIATE MODE
YES
NO
NO
IS THIS A ZERO-LENGTH BUFFER
YES
IS THERE A TEXT TRANSFER ERROR
YES
NO
UPFEFO
GET THE ADDRESS OF THE MASTER QCB
CLEAR THE 'CURRENT SENDING' FLAGS
GET THE ADDRESS OF THE PRIORITY QCB FROM THE MASTER QCB
UPDATE THE FEFO POINTER IN THE PRIORITY QCB FROM THE SCB
COMPIDLS
SET THE RESERVE CHARACTERS COUNT TO ZERO
IS THIS A ZERO-LENGTH BUFFER
NO
RESET THE RESERVE CHARACTERS COUNT FROM THE BUFFER
YES
STORIDLS
SET THE RESERVE CHARACTERS COUNT IN THE LCB; SET THE SCAN POINTER
IS THIS A ZERO-LENGTH BUFFER
YES
NO
NT A3
IEDQTNT
GET THE TERMINAL TABLE ENTRY ADDRESS
PUT THE OUTPUT SEQUENCE NUMBER IN THE SCB AND ADD ONE
IS THE MAXIMUM COUNT EXCEEDED
NO
YES
UPCOUNT
RESET THE COUNT TO ONE IN THE TERMINAL TABLE ENTRY
AA1 C5
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart AA-3 STARTMH SUBTASK

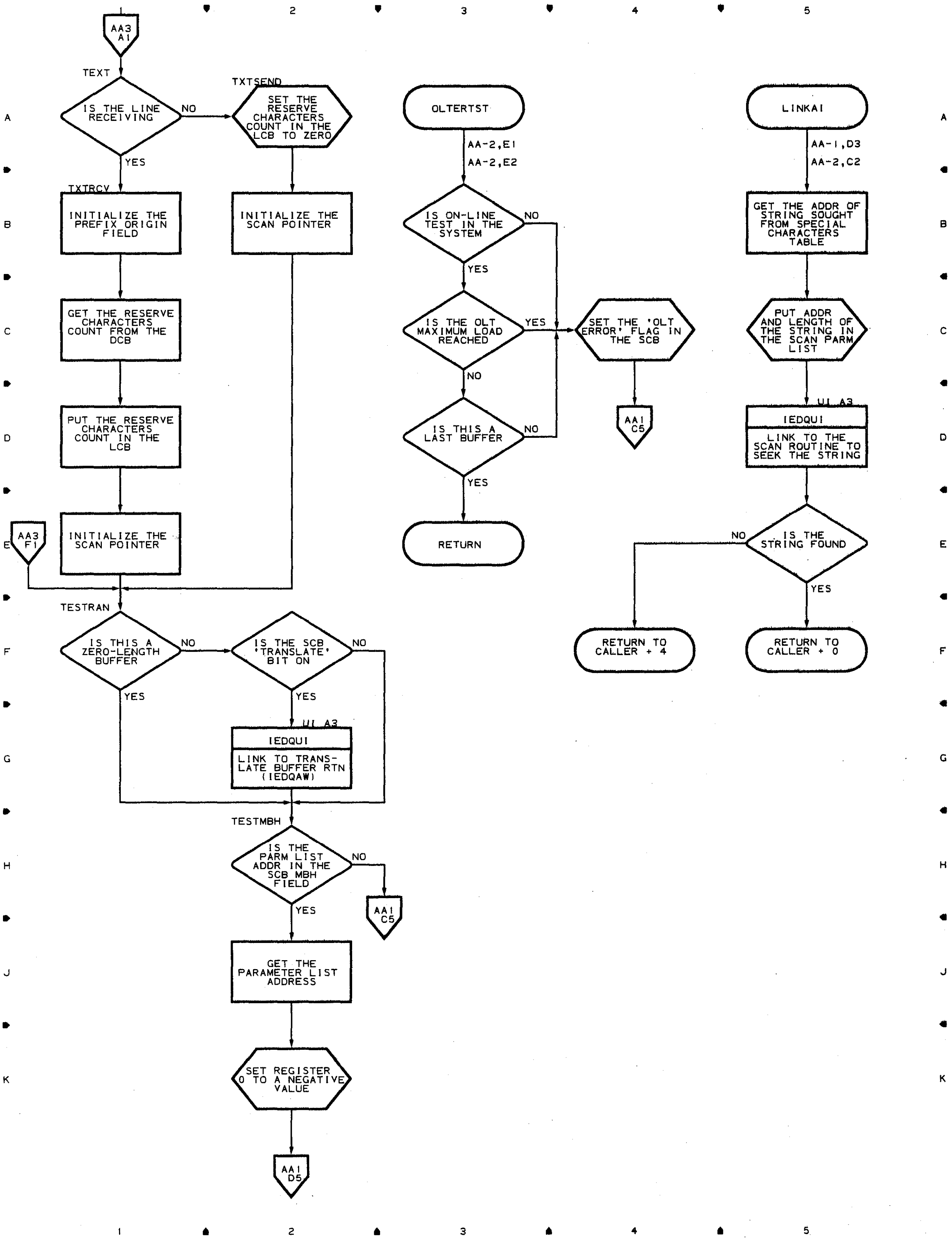

AA3 A1
TEXT
IS THE LINE RECEIVING
NO
YES
TXTSEND
SET THE RESERVE CHARACTERS COUNT IN THE LCB TO ZERO
TXTRCV
INITIALIZE THE PREFIX ORIGIN FIELD
INITIALIZE THE SCAN POINTER
GET THE RESERVE CHARACTERS COUNT FROM THE DCB
PUT THE RESERVE CHARACTERS COUNT IN THE LCB
AA3 F1
INITIALIZE THE SCAN POINTER
TESTRAN
IS THIS A ZERO-LENGTH BUFFER
NO
YES
IS THE SCB 'TRANSLATE' BIT ON
NO
YES
UI A3
IEDQUI
LINK TO TRANSLATE BUFFER RTN (IEDQAW)
TESTMBH
IS THE PARM LIST ADDR IN THE SCB MBH FIELD
NO
YES
AA1 C5
GET THE PARAMETER LIST ADDRESS
SET REGISTER 0 TO A NEGATIVE VALUE
AA1 D5
OLTERTST
AA-2,E1
AA-2,E2
IS ON-LINE TEST IN THE SYSTEM
NO
YES
IS THE OLT MAXIMUM LOAD REACHED
YES
NO
IS THIS A LAST BUFFER
NO
YES
RETURN
SET THE 'OLT ERROR' FLAG IN THE SCB
AA1 C5
LINKA1
AA-1,D3
AA-2,C2
GET THE ADDR OF STRING SOUGHT FROM SPECIAL CHARACTERS TABLE
PUT ADDR AND LENGTH OF THE STRING IN THE SCAN PARM LIST
UI A3
IEDQUI
LINK TO THE SCAN ROUTINE TO SEEK THE STRING
IS THE STRING FOUND
NO
YES
RETURN TO CALLER + 4
RETURN TO CALLER + 0
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart AC DATE AND TIME PROVISION ROUTINE

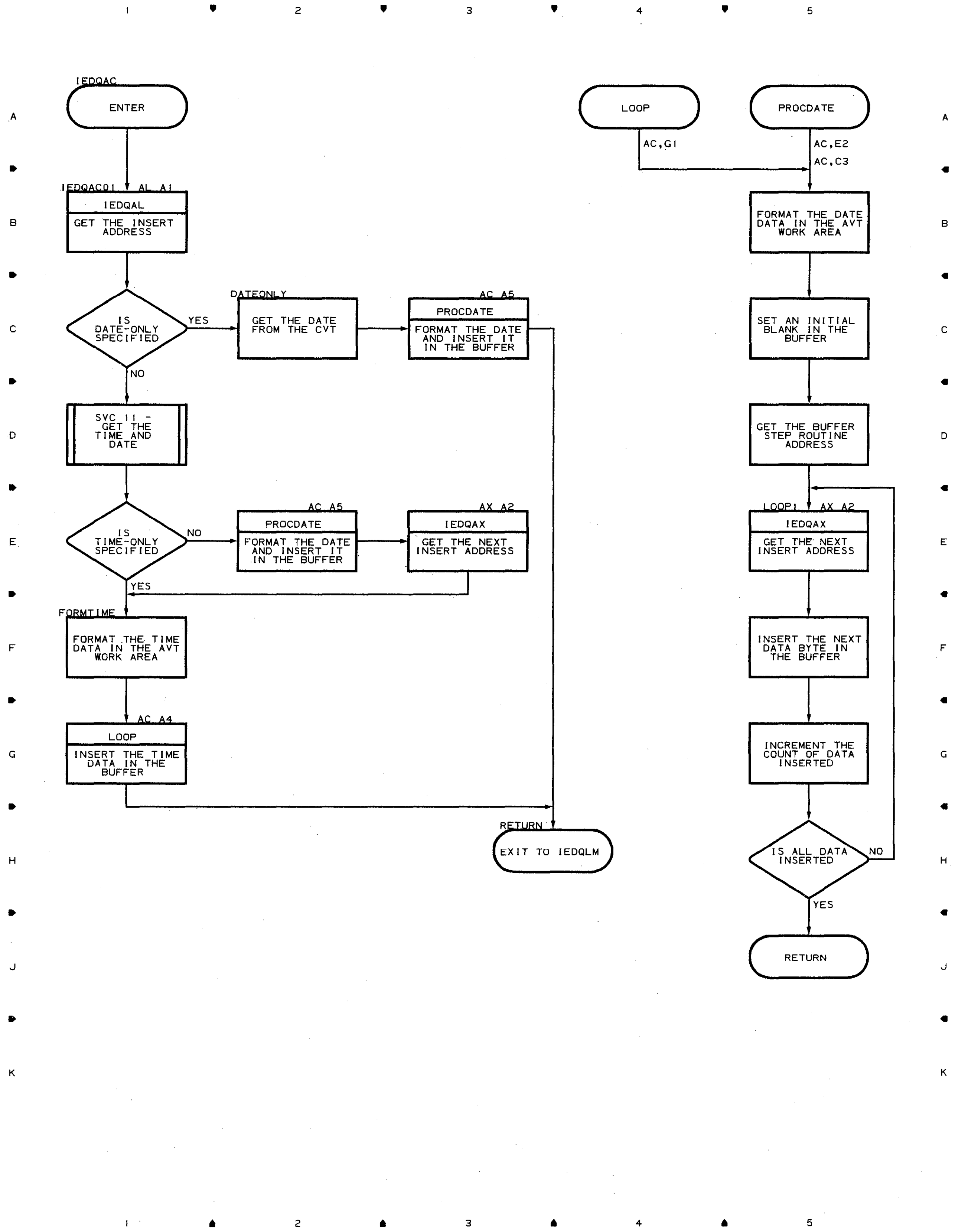
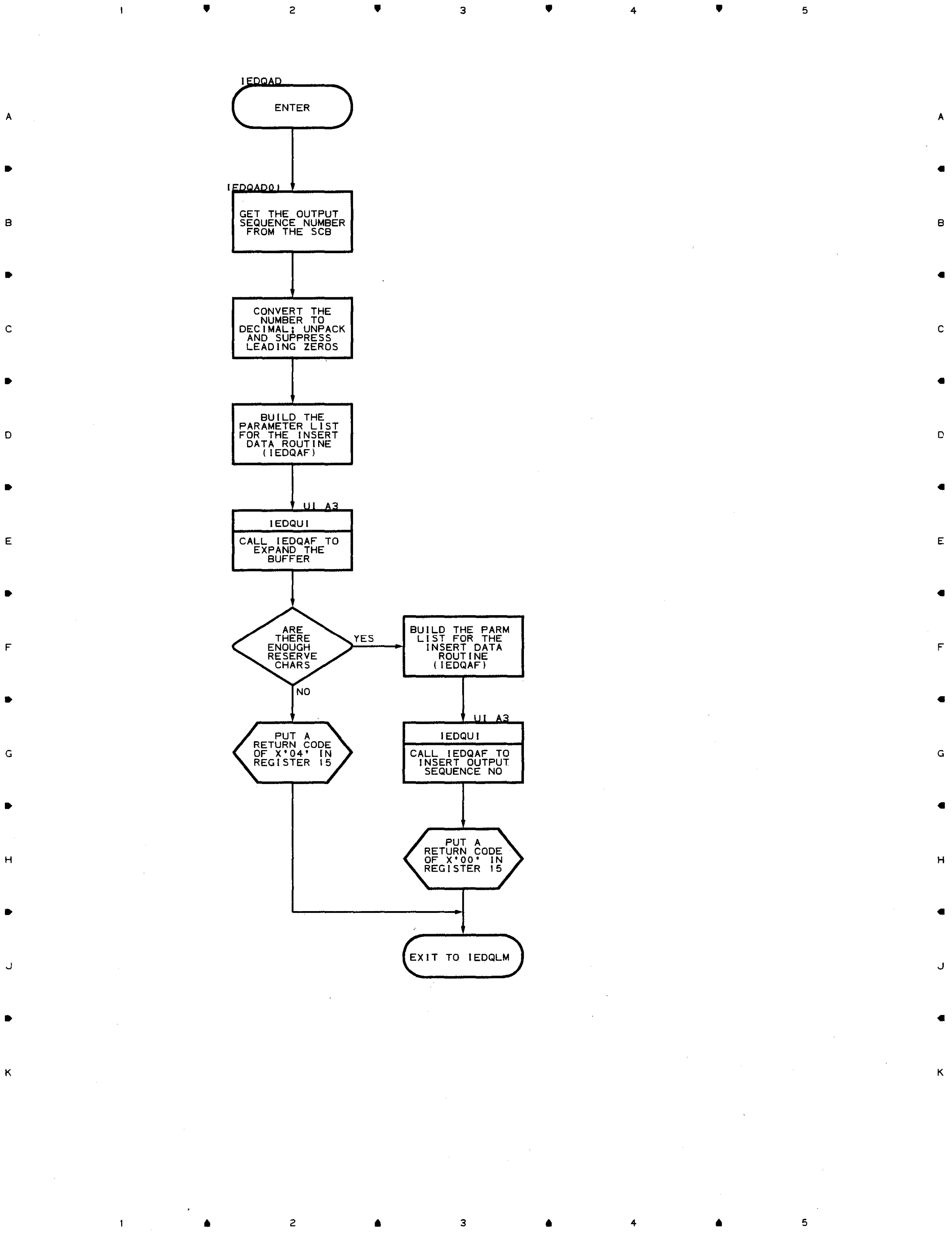

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQAC
ENTER
LOOP
AC,G1
PROCDATE
AC,E2
AC,C3
IEDQAC01 AL A1
IEDQAL
GET THE INSERT ADDRESS
FORMAT THE DATE DATA IN THE AVT WORK AREA
IS DATE-ONLY SPECIFIED
YES
NO
DATEONLY
GET THE DATE FROM THE CVT
AC A5
PROCDATE
FORMAT THE DATE AND INSERT IT IN THE BUFFER
SET AN INITIAL BLANK IN THE BUFFER
SVC 11 - GET THE TIME AND DATE
GET THE BUFFER STEP ROUTINE ADDRESS
IS TIME-ONLY SPECIFIED
NO
YES
AC A5
PROCDATE
FORMAT THE DATE AND INSERT IT IN THE BUFFER
AX A2
IEDQAX
GET THE NEXT INSERT ADDRESS
LOOP1 AX A2
IEDQAX
GET THE NEXT INSERT ADDRESS
FORMTIME
FORMAT THE TIME DATA IN THE AVT WORK AREA
INSERT THE NEXT DATA BYTE IN THE BUFFER
AC A4
LOOP
INSERT THE TIME DATA IN THE BUFFER
INCREMENT THE COUNT OF DATA INSERTED
RETURN
EXIT TO IEDQLM
IS ALL DATA INSERTED
NO
YES
RETURN

Chart AD OUTPUT SEQUENCE NUMBER PROVISION ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQAD
ENTER
IEDQAD01
GET THE OUTPUT SEQUENCE NUMBER FROM THE SCB
CONVERT THE NUMBER TO DECIMAL; UNPACK AND SUPPRESS LEADING ZEROS
BUILD THE PARAMETER LIST FOR THE INSERT DATA ROUTINE (IEDQAF)
UI A3
IEDQUI
CALL IEDQAF TO EXPAND THE BUFFER
ARE THERE ENOUGH RESERVE CHARS
YES
NO
BUILD THE PARM LIST FOR THE INSERT DATA ROUTINE (IEDQAF)
UI A3
IEDQUI
CALL IEDQAF TO INSERT OUTPUT SEQUENCE NO
PUT A RETURN CODE OF X'04' IN REGISTER 15
PUT A RETURN CODE OF X'00' IN REGISTER 15
EXIT TO IEDQLM

Chart AE LOCATE OPTION FIELD ADDRESS ROUTINE

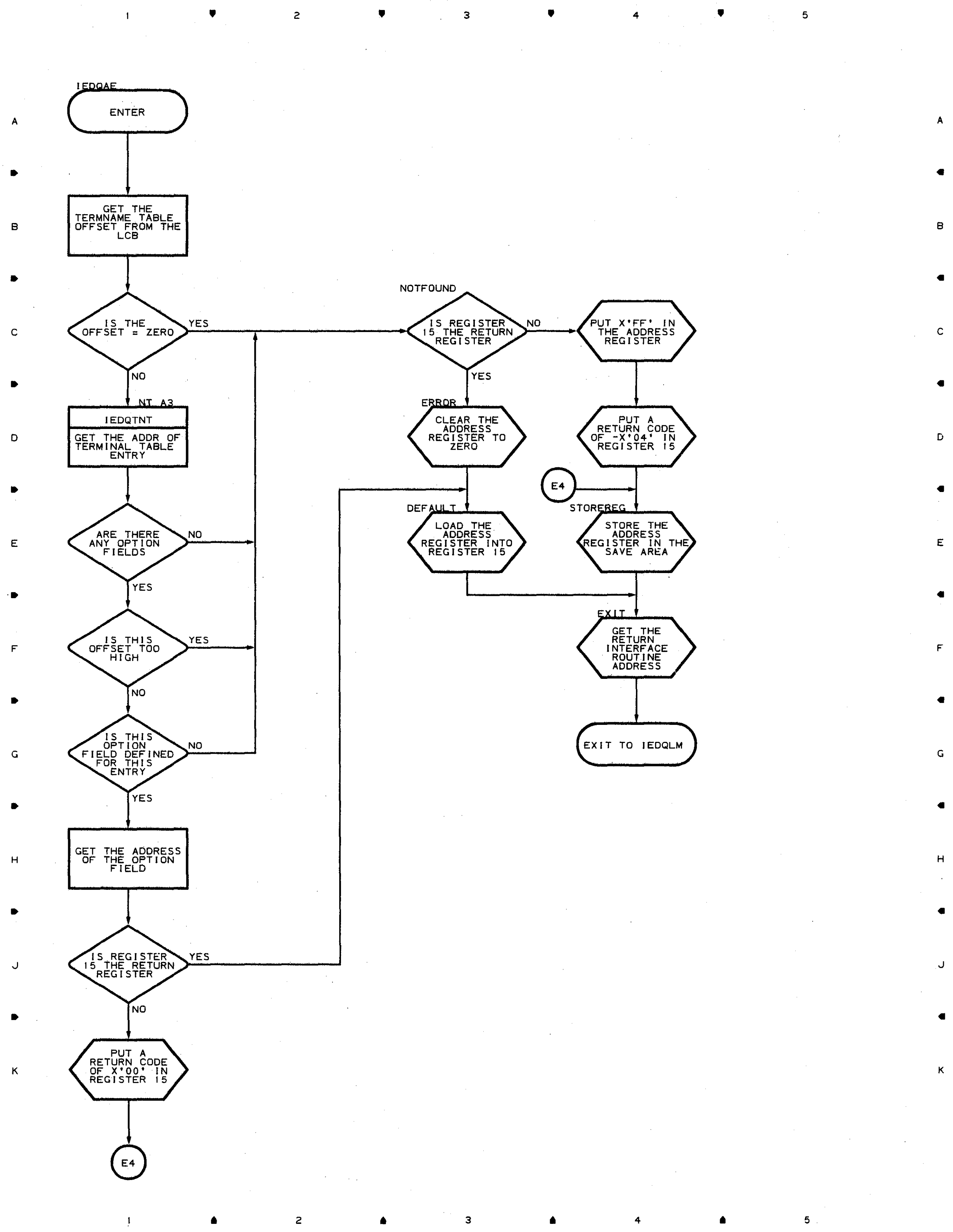
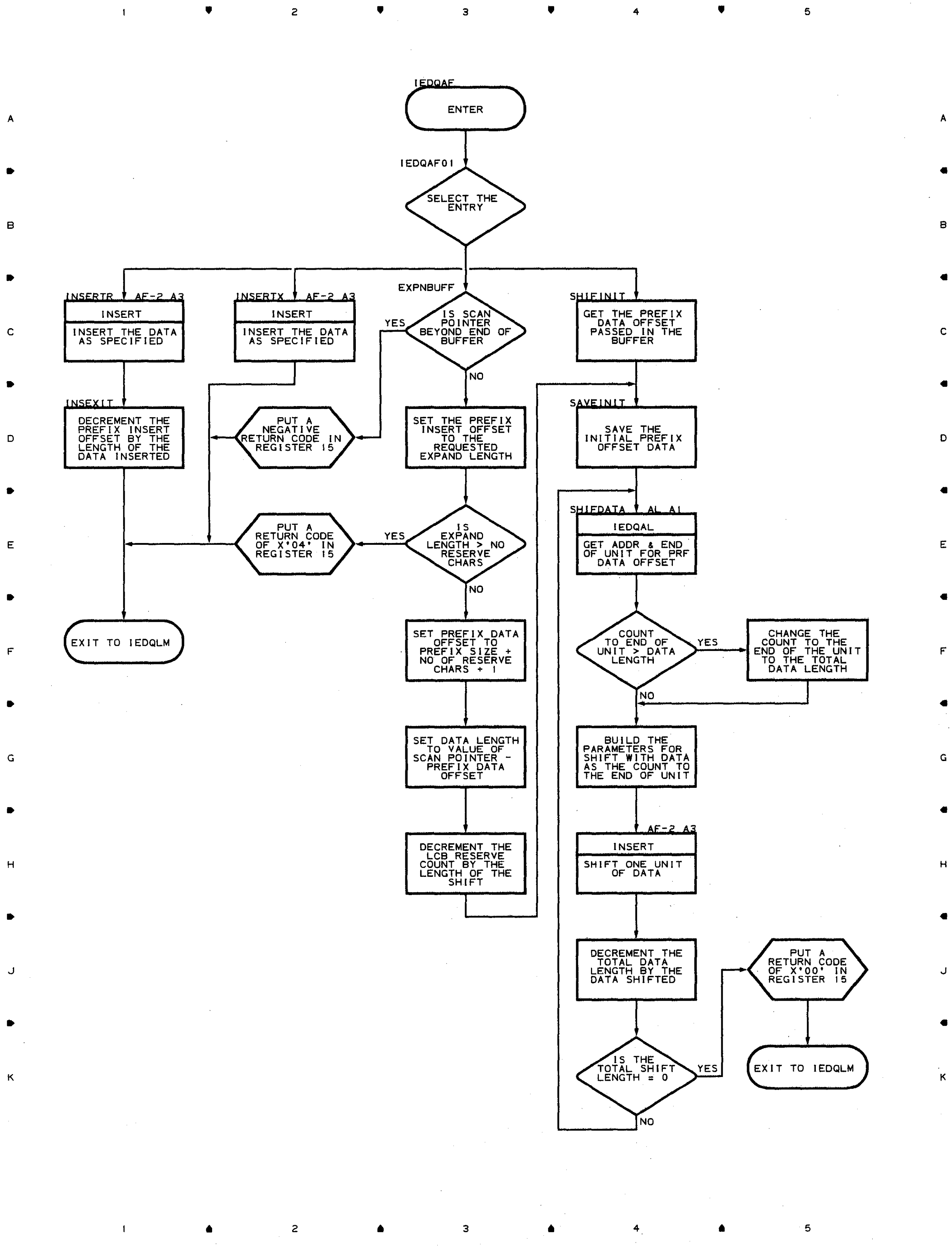

IEDQAE
ENTER
GET THE TERMNAME TABLE OFFSET FROM THE LCB
IS THE OFFSET = ZERO
YES
NO
NT A3
IEDQTNT
GET THE ADDR OF TERMINAL TABLE ENTRY
ARE THERE ANY OPTION FIELDS
NO
YES
IS THIS OFFSET TOO HIGH
YES
NO
IS THIS OPTION FIELD DEFINED FOR THIS ENTRY
NO
YES
GET THE ADDRESS OF THE OPTION FIELD
IS REGISTER 15 THE RETURN REGISTER
YES
NO
PUT A RETURN CODE OF X'00' IN REGISTER 15
E4
NOTFOUND
IS REGISTER 15 THE RETURN REGISTER
NO
YES
ERROR
CLEAR THE ADDRESS REGISTER TO ZERO
DEFAULT
LOAD THE ADDRESS REGISTER INTO REGISTER 15
PUT X'FF' IN THE ADDRESS REGISTER
PUT A RETURN CODE OF -X'04' IN REGISTER 15
E4
STOREREG
STORE THE ADDRESS REGISTER IN THE SAVE AREA
EXIT
GET THE RETURN INTERFACE ROUTINE ADDRESS
EXIT TO IEDQLM
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart AF-1 INSERT DATA ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQAF
ENTER
IEDQAF01
SELECT THE ENTRY
INSERTR
AF-2 A3
INSERT
INSERT THE DATA AS SPECIFIED
INSERTX
AF-2 A3
INSERT
INSERT THE DATA AS SPECIFIED
EXPNBUFF
IS SCAN POINTER BEYOND END OF BUFFER
YES
NO
SHIFINIT
GET THE PREFIX DATA OFFSET PASSED IN THE BUFFER
INSEXIT
DECREMENT THE PREFIX INSERT OFFSET BY THE LENGTH OF THE DATA INSERTED
PUT A NEGATIVE RETURN CODE IN REGISTER 15
SET THE PREFIX INSERT OFFSET TO THE REQUESTED EXPAND LENGTH
SAVEINIT
SAVE THE INITIAL PREFIX OFFSET DATA
PUT A RETURN CODE OF X'04' IN REGISTER 15
YES
IS EXPAND LENGTH > NO RESERVE CHARS
NO
SHIFDATA
AL A1
IEDQAL
GET ADDR & END OF UNIT FOR PRF DATA OFFSET
EXIT TO IEDQLM
SET PREFIX DATA OFFSET TO PREFIX SIZE + NO OF RESERVE CHARS + 1
COUNT TO END OF UNIT > DATA LENGTH
YES
NO
CHANGE THE COUNT TO THE END OF THE UNIT TO THE TOTAL DATA LENGTH
SET DATA LENGTH TO VALUE OF SCAN POINTER - PREFIX DATA OFFSET
BUILD THE PARAMETERS FOR SHIFT WITH DATA AS THE COUNT TO THE END OF UNIT
DECREMENT THE LCB RESERVE COUNT BY THE LENGTH OF THE SHIFT
AF-2 A3
INSERT
SHIFT ONE UNIT OF DATA
DECREMENT THE TOTAL DATA LENGTH BY THE DATA SHIFTED
PUT A RETURN CODE OF X'00' IN REGISTER 15
IS THE TOTAL SHIFT LENGTH = 0
YES
NO
EXIT TO IEDQLM

Chart AF-2 INSERT DATA ROUTINE

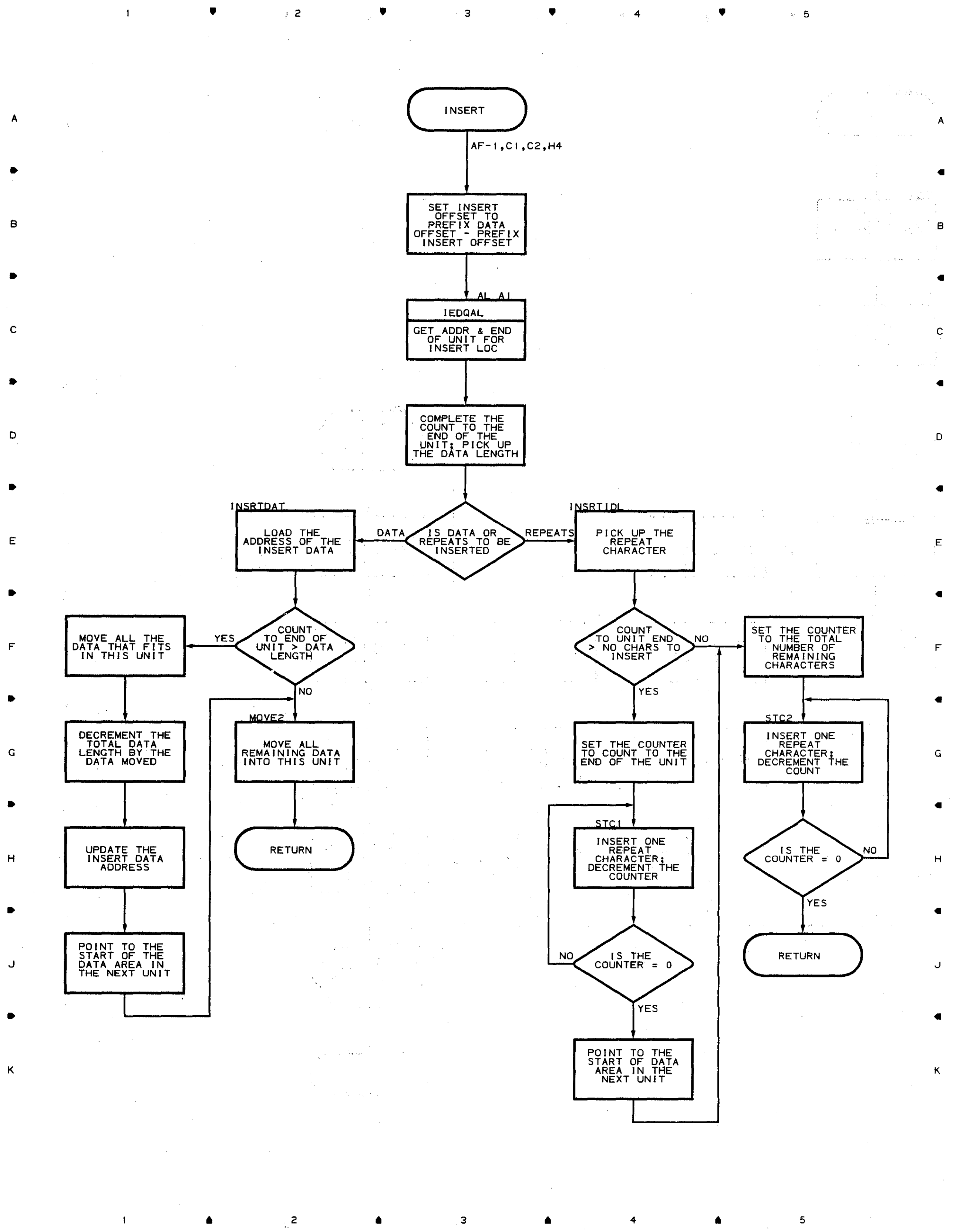

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
INSERT
AF-1,C1,C2,H4
SET INSERT OFFSET TO PREFIX DATA OFFSET - PREFIX INSERT OFFSET
AL A1
IEDQAL
GET ADDR & END OF UNIT FOR INSERT LOC
COMPLETE THE COUNT TO THE END OF THE UNIT; PICK UP THE DATA LENGTH
IS DATA OR REPEATS TO BE INSERTED
DATA
REPEATS
INSRTDAT
LOAD THE ADDRESS OF THE INSERT DATA
INSRTIDL
PICK UP THE REPEAT CHARACTER
COUNT TO END OF UNIT > DATA LENGTH
YES
NO
MOVE ALL THE DATA THAT FITS IN THIS UNIT
DECREMENT THE TOTAL DATA LENGTH BY THE DATA MOVED
UPDATE THE INSERT DATA ADDRESS
POINT TO THE START OF THE DATA AREA IN THE NEXT UNIT
MOVE2
MOVE ALL REMAINING DATA INTO THIS UNIT
RETURN
COUNT TO UNIT END > NO CHARS TO INSERT
NO
YES
SET THE COUNTER TO COUNT TO THE END OF THE UNIT
STC1
INSERT ONE REPEAT CHARACTER; DECREMENT THE COUNTER
IS THE COUNTER = 0
NO
YES
POINT TO THE START OF DATA AREA IN THE NEXT UNIT
SET THE COUNTER TO THE TOTAL NUMBER OF REMAINING CHARACTERS
STC2
INSERT ONE REPEAT CHARACTER; DECREMENT THE COUNT
IS THE COUNTER = 0
NO
YES
RETURN

Chart AG MESSAGE LIMIT ROUTINE

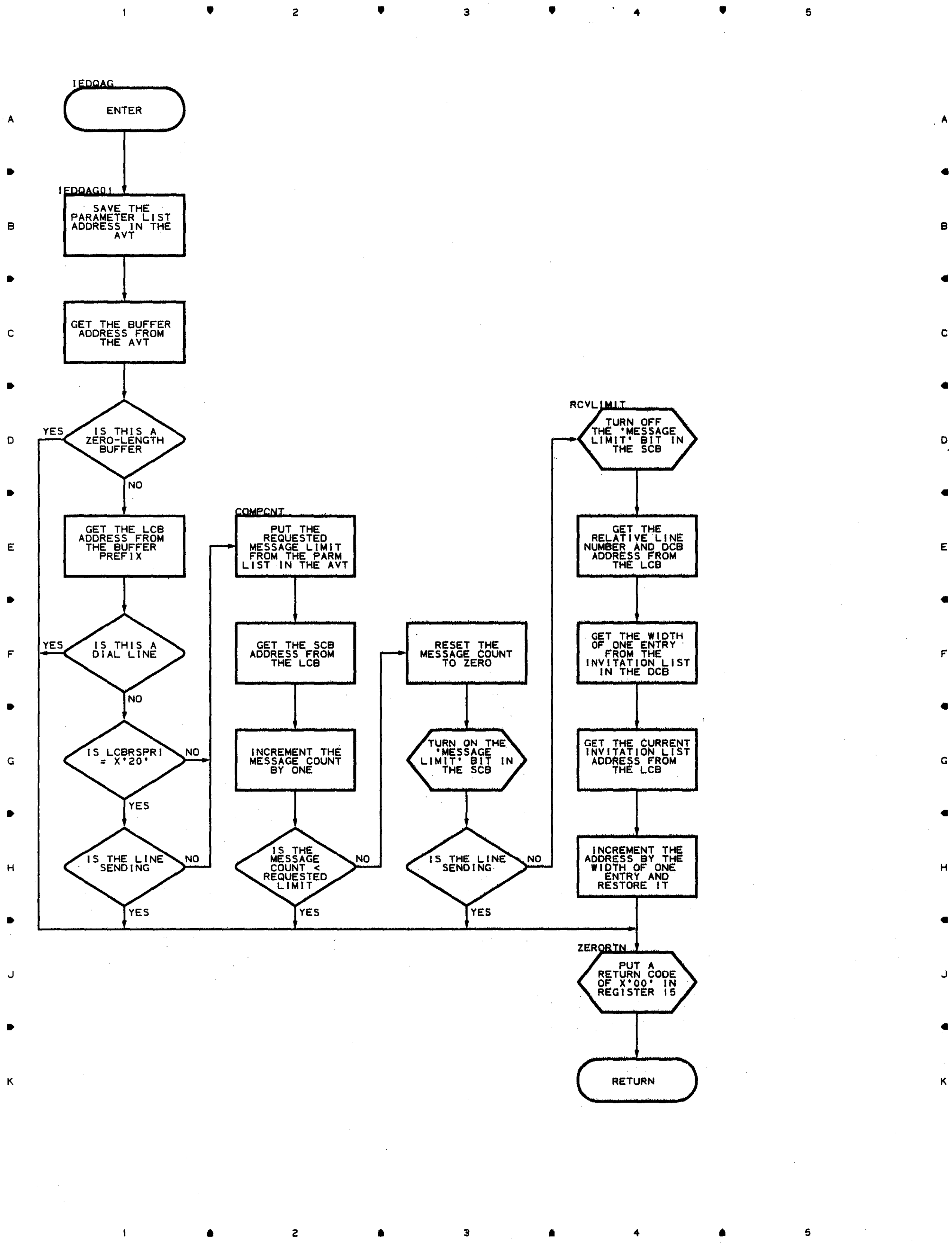

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQAG
ENTER
IEDQAG01
SAVE THE PARAMETER LIST ADDRESS IN THE AVT
GET THE BUFFER ADDRESS FROM THE AVT
YES
IS THIS A ZERO-LENGTH BUFFER
NO
GET THE LCB ADDRESS FROM THE BUFFER PREFIX
YES
IS THIS A DIAL LINE
NO
IS LCBRSPRI = X'20'
NO
YES
IS THE LINE SENDING
NO
YES
COMPCNT
PUT THE REQUESTED MESSAGE LIMIT FROM THE PARM LIST IN THE AVT
GET THE SCB ADDRESS FROM THE LCB
INCREMENT THE MESSAGE COUNT BY ONE
IS THE MESSAGE COUNT < REQUESTED LIMIT
NO
YES
RESET THE MESSAGE COUNT TO ZERO
TURN ON THE 'MESSAGE LIMIT' BIT IN THE SCB
IS THE LINE SENDING
NO
YES
RCVLIMIT
TURN OFF THE 'MESSAGE LIMIT' BIT IN THE SCB
GET THE RELATIVE LINE NUMBER AND DCB ADDRESS FROM THE LCB
GET THE WIDTH OF ONE ENTRY FROM THE INVITATION LIST IN THE DCB
GET THE CURRENT INVITATION LIST ADDRESS FROM THE LCB
INCREMENT THE ADDRESS BY THE WIDTH OF ONE ENTRY AND RESTORE IT
ZERORTN
PUT A RETURN CODE OF X'00' IN REGISTER 15
RETURN

Chart AH INPUT SEQUENCE NUMBER INSERTION ROUTINE

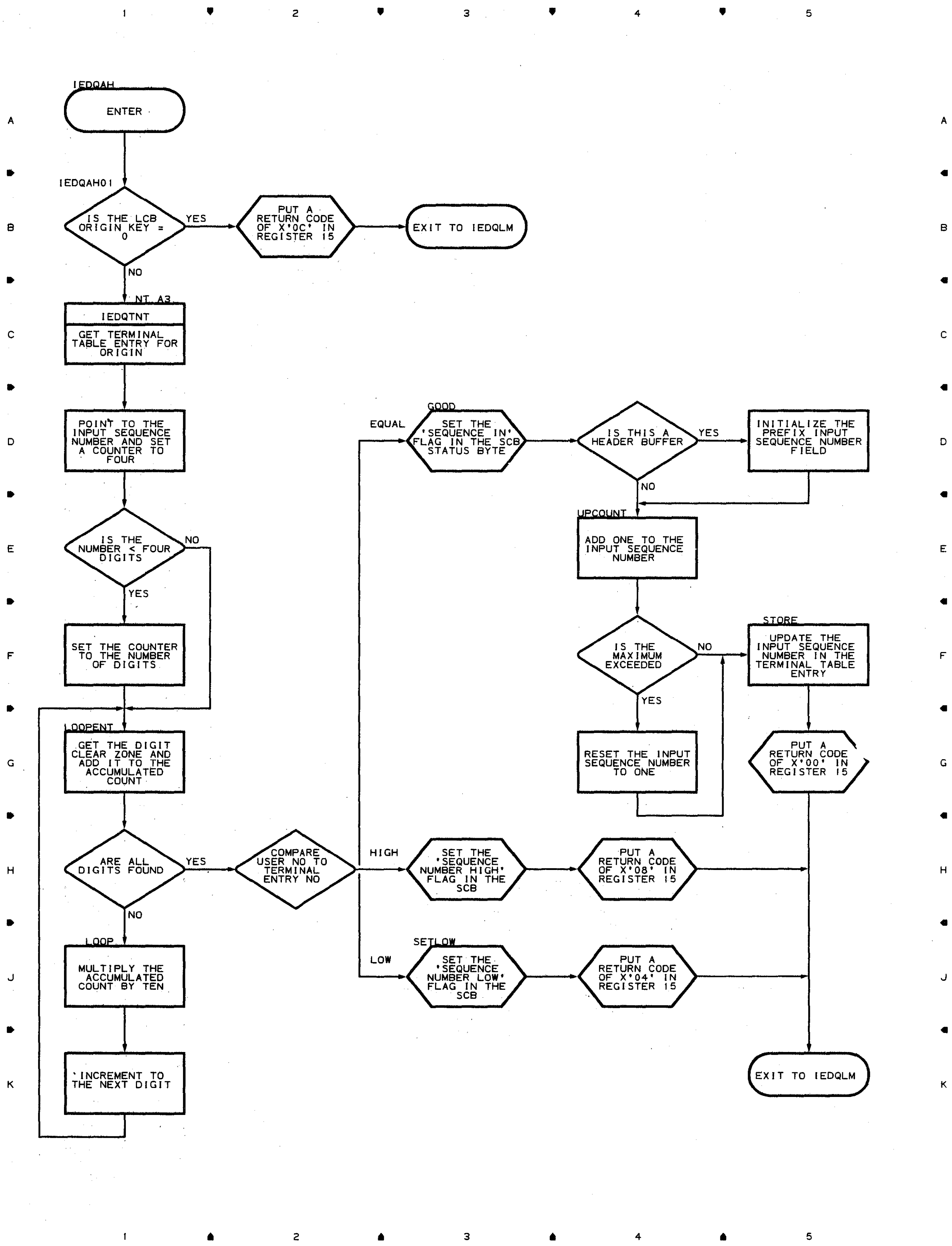
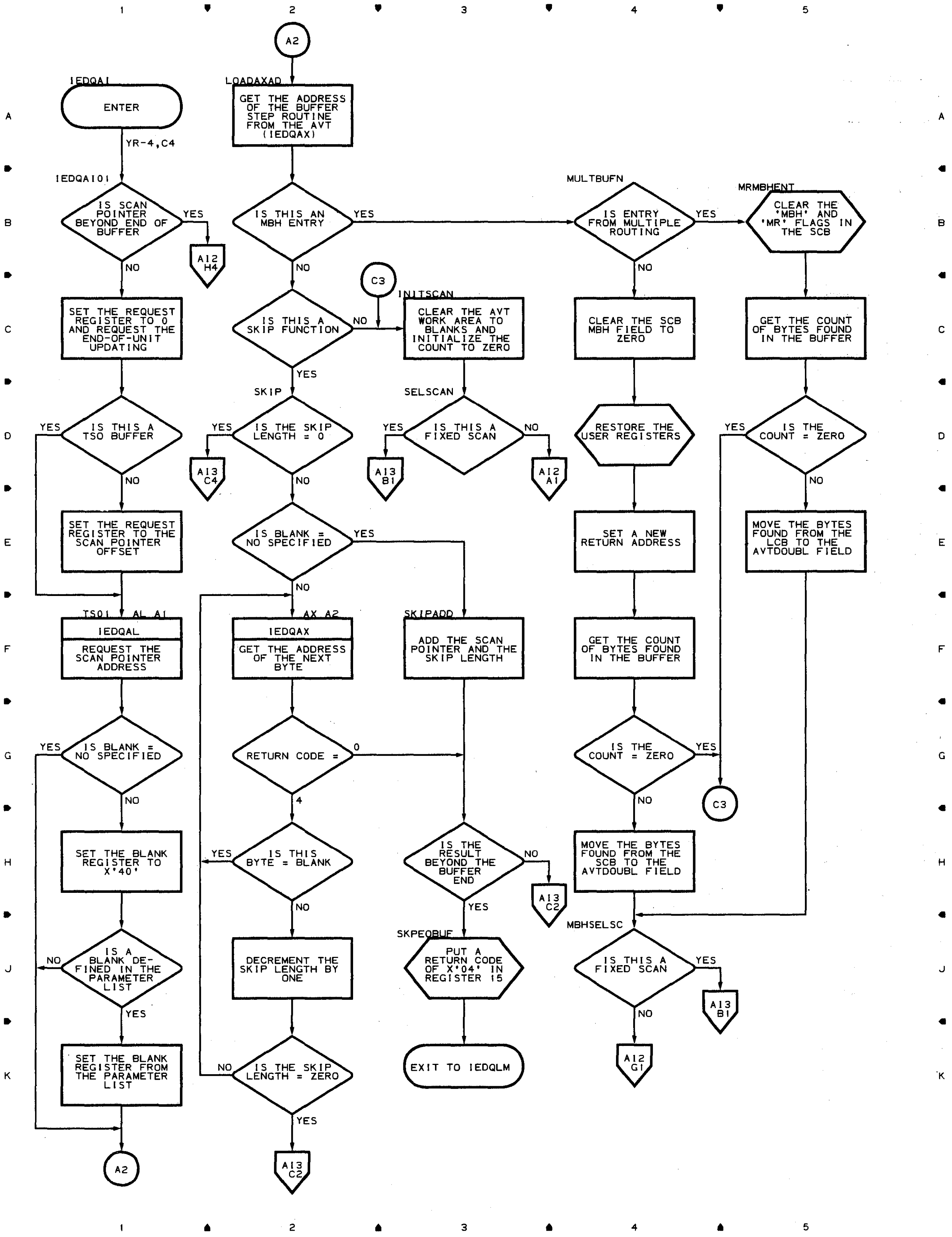

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQAH
ENTER
IEDQAH01
IS THE LCB ORIGIN KEY = 0
YES
PUT A RETURN CODE OF X'0C' IN REGISTER 15
EXIT TO IEDQLM
NO
NT A3
IEDQTNT
GET TERMINAL TABLE ENTRY FOR ORIGIN
POINT TO THE INPUT SEQUENCE NUMBER AND SET A COUNTER TO FOUR
IS THE NUMBER < FOUR DIGITS
NO
YES
SET THE COUNTER TO THE NUMBER OF DIGITS
LOOPENT
GET THE DIGIT CLEAR ZONE AND ADD IT TO THE ACCUMULATED COUNT
ARE ALL DIGITS FOUND
YES
NO
LOOP
MULTIPLY THE ACCUMULATED COUNT BY TEN
INCREMENT TO THE NEXT DIGIT
COMPARE USER NO TO TERMINAL ENTRY NO
EQUAL
HIGH
LOW
GOOD
SET THE 'SEQUENCE IN' FLAG IN THE SCB STATUS BYTE
IS THIS A HEADER BUFFER
YES
INITIALIZE THE PREFIX INPUT SEQUENCE NUMBER FIELD
NO
UPCOUNT
ADD ONE TO THE INPUT SEQUENCE NUMBER
IS THE MAXIMUM EXCEEDED
NO
YES
STORE
UPDATE THE INPUT SEQUENCE NUMBER IN THE TERMINAL TABLE ENTRY
RESET THE INPUT SEQUENCE NUMBER TO ONE
PUT A RETURN CODE OF X'00' IN REGISTER 15
SET THE 'SEQUENCE NUMBER HIGH' FLAG IN THE SCB
PUT A RETURN CODE OF X'08' IN REGISTER 15
SETLOW
SET THE 'SEQUENCE NUMBER LOW' FLAG IN THE SCB
PUT A RETURN CODE OF X'04' IN REGISTER 15
EXIT TO IEDQLM

Chart AI-1 SKIP FORWARD AND SCAN ROUTINE



IEDQAI
ENTER
YR-4,C4
IEDQAI01
IS SCAN POINTER BEYOND END OF BUFFER
YES
AI2 H4
NO
SET THE REQUEST REGISTER TO 0 AND REQUEST THE END-OF-UNIT UPDATING
IS THIS A TSO BUFFER
YES
NO
SET THE REQUEST REGISTER TO THE SCAN POINTER OFFSET
TSO1
AL A1
IEDQAL
REQUEST THE SCAN POINTER ADDRESS
IS BLANK = NO SPECIFIED
YES
NO
SET THE BLANK REGISTER TO X'40'
IS A BLANK DE-FINED IN THE PARAMETER LIST
NO
YES
SET THE BLANK REGISTER FROM THE PARAMETER LIST
A2
A2
LOADAXAD
GET THE ADDRESS OF THE BUFFER STEP ROUTINE FROM THE AVT (IEDQAX)
IS THIS AN MBH ENTRY
YES
NO
IS THIS A SKIP FUNCTION
NO
YES
SKIP
IS THE SKIP LENGTH = 0
YES
AI3 C4
NO
IS BLANK = NO SPECIFIED
YES
NO
AX A2
IEDQAX
GET THE ADDRESS OF THE NEXT BYTE
RETURN CODE =
0
4
IS THIS BYTE = BLANK
YES
NO
DECREMENT THE SKIP LENGTH BY ONE
IS THE SKIP LENGTH = ZERO
NO
YES
AI3 C2
C3
INITSCAN
CLEAR THE AVT WORK AREA TO BLANKS AND INITIALIZE THE COUNT TO ZERO
SELSCAN
IS THIS A FIXED SCAN
YES
AI3 B1
NO
AI2 A1
SKIPADD
ADD THE SCAN POINTER AND THE SKIP LENGTH
IS THE RESULT BEYOND THE BUFFER END
NO
AI3 C2
YES
SKPEOBUF
PUT A RETURN CODE OF X'04' IN REGISTER 15
EXIT TO IEDQLM
MULTBUFN
IS ENTRY FROM MULTIPLE ROUTING
YES
NO
CLEAR THE SCB MBH FIELD TO ZERO
RESTORE THE USER REGISTERS
SET A NEW RETURN ADDRESS
GET THE COUNT OF BYTES FOUND IN THE BUFFER
IS THE COUNT = ZERO
YES
C3
NO
MOVE THE BYTES FOUND FROM THE SCB TO THE AVTDOUBL FIELD
MBHSELSC
IS THIS A FIXED SCAN
YES
AI3 B1
NO
AI2 G1
MRMBHENT
CLEAR THE 'MBH' AND 'MR' FLAGS IN THE SCB
GET THE COUNT OF BYTES FOUND IN THE BUFFER
IS THE COUNT = ZERO
YES
NO
MOVE THE BYTES FOUND FROM THE LCB TO THE AVTDOUBL FIELD
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart AI-2 SKIP FORWARD AND SCAN ROUTINE

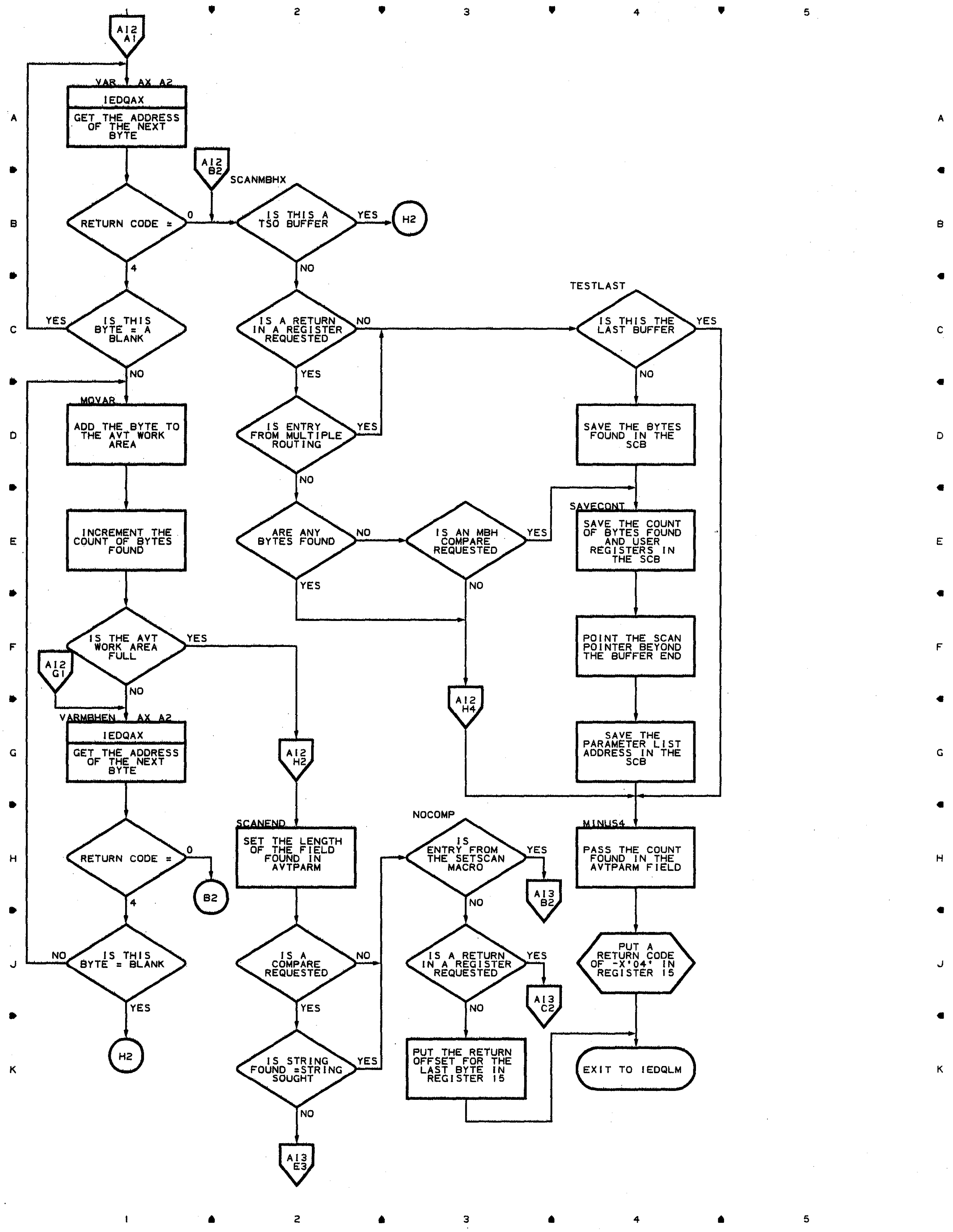
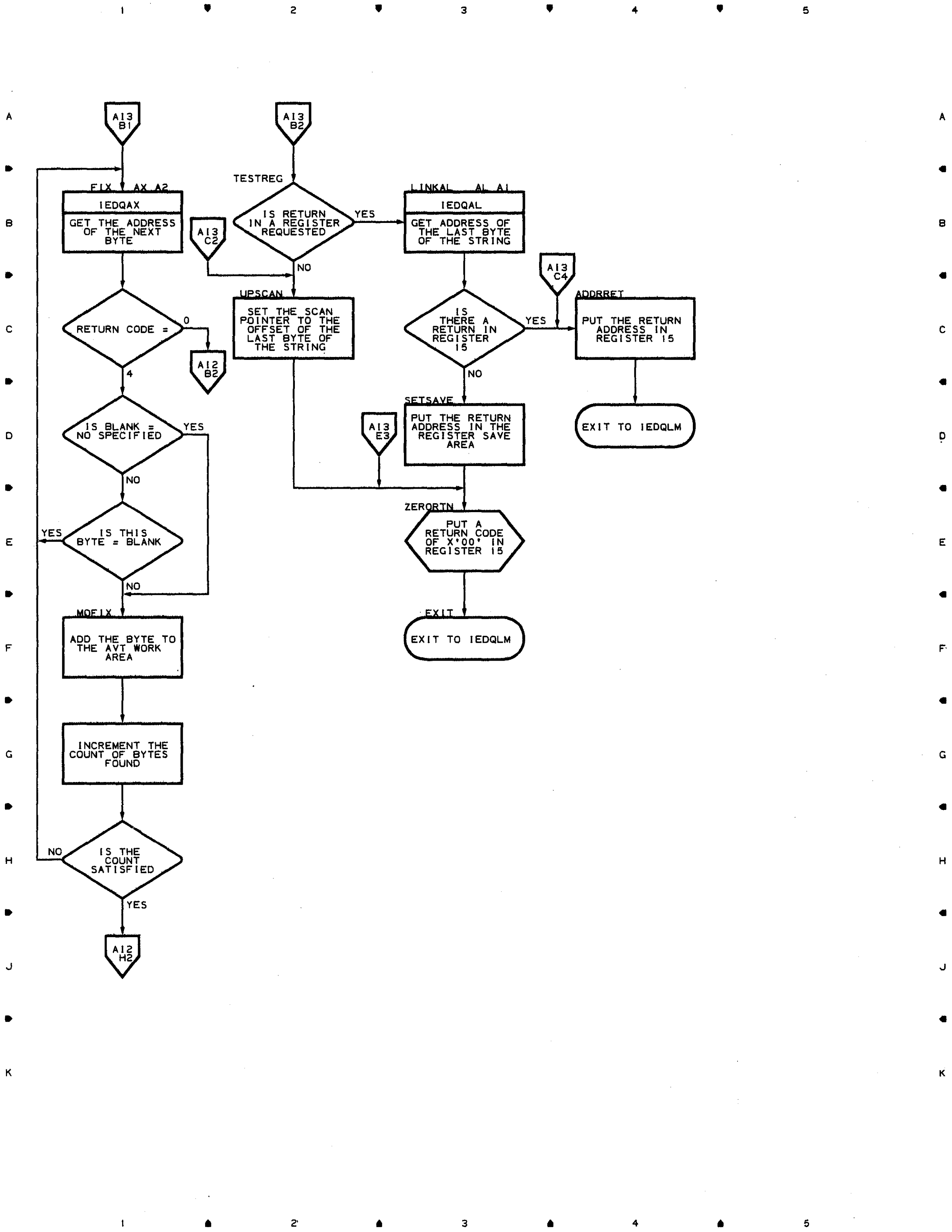
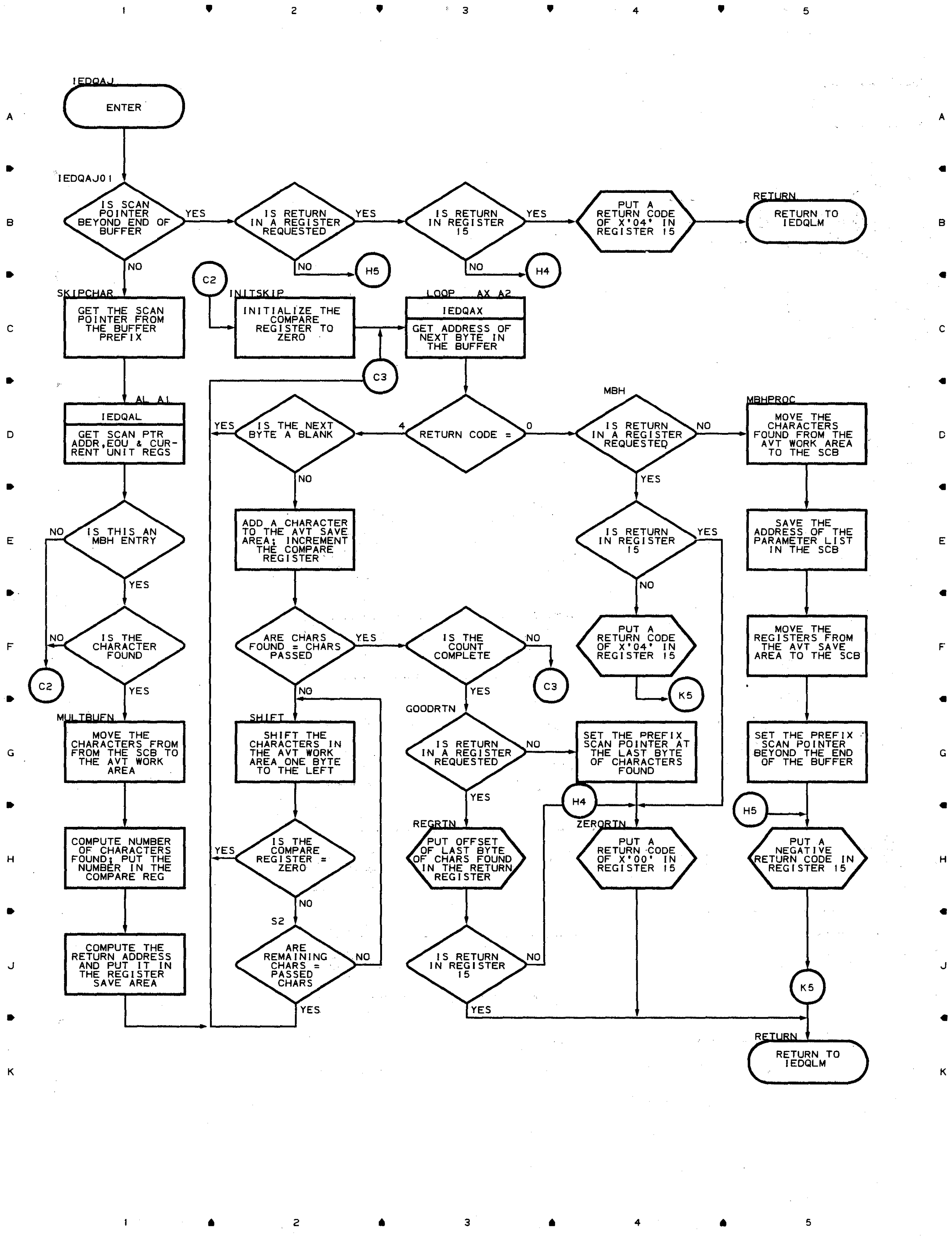

A12 A1
VAR AX A2
IEDQAX
GET THE ADDRESS OF THE NEXT BYTE
A12 B2
SCANMBHX
RETURN CODE =
0
4
IS THIS A TSO BUFFER
YES
H2
NO
IS THIS BYTE = A BLANK
YES
NO
IS A RETURN IN A REGISTER REQUESTED
NO
YES
TESTLAST
IS THIS THE LAST BUFFER
YES
NO
MOVAR
ADD THE BYTE TO THE AVT WORK AREA
IS ENTRY FROM MULTIPLE ROUTING
YES
NO
SAVE THE BYTES FOUND IN THE SCB
SAVECONT
INCREMENT THE COUNT OF BYTES FOUND
ARE ANY BYTES FOUND
NO
IS AN MBH COMPARE REQUESTED
YES
SAVE THE COUNT OF BYTES FOUND AND USER REGISTERS IN THE SCB
YES
NO
IS THE AVT WORK AREA FULL
YES
A12 G1
POINT THE SCAN POINTER BEYOND THE BUFFER END
NO
A12 H4
VARMBHEN AX A2
IEDQAX
GET THE ADDRESS OF THE NEXT BYTE
A12 H2
SAVE THE PARAMETER LIST ADDRESS IN THE SCB
SCANEND
NOCOMP
MINUS4
RETURN CODE =
0
B2
SET THE LENGTH OF THE FIELD FOUND IN AVTPARM
IS ENTRY FROM THE SETSCAN MACRO
YES
A13 B2
PASS THE COUNT FOUND IN THE AVTPARM FIELD
4
NO
IS THIS BYTE = BLANK
NO
IS A COMPARE REQUESTED
NO
IS A RETURN IN A REGISTER REQUESTED
YES
A13 C2
PUT A RETURN CODE OF -X'04' IN REGISTER 15
YES
YES
NO
H2
IS STRING FOUND =STRING SOUGHT
YES
PUT THE RETURN OFFSET FOR THE LAST BYTE IN REGISTER 15
EXIT TO IEDQLM
NO
A13 E3
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart AI-3 SKIP FORWARD AND SCAN ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
AI3 B1
AI3 B2
TESTREG
FIX AX A2
IEDQAX
GET THE ADDRESS OF THE NEXT BYTE
IS RETURN IN A REGISTER REQUESTED
YES
NO
AI3 C2
LINKAL AL A1
IEDQAL
GET ADDRESS OF THE LAST BYTE OF THE STRING
AI3 C4
RETURN CODE =
0
4
AI2 B2
UPSCAN
SET THE SCAN POINTER TO THE OFFSET OF THE LAST BYTE OF THE STRING
IS THERE A RETURN IN REGISTER 15
YES
NO
ADDRRET
PUT THE RETURN ADDRESS IN REGISTER 15
EXIT TO IEDQLM
IS BLANK = NO SPECIFIED
YES
NO
AI3 E3
SETSAVE
PUT THE RETURN ADDRESS IN THE REGISTER SAVE AREA
ZERORTN
PUT A RETURN CODE OF X'00' IN REGISTER 15
IS THIS BYTE = BLANK
YES
NO
MOFIX
ADD THE BYTE TO THE AVT WORK AREA
EXIT
EXIT TO IEDQLM
INCREMENT THE COUNT OF BYTES FOUND
IS THE COUNT SATISFIED
NO
YES
AI2 H2

Chart AJ SKIP TO CHARACTER SET ROUTINE



IEDQAJ
ENTER
IEDQAJ01
IS SCAN POINTER BEYOND END OF BUFFER
YES
NO
IS RETURN IN A REGISTER REQUESTED
YES
NO
H5
IS RETURN IN REGISTER 15
YES
NO
H4
PUT A RETURN CODE OF X'04' IN REGISTER 15
RETURN
RETURN TO IEDQLM
SKIPCHAR
GET THE SCAN POINTER FROM THE BUFFER PREFIX
C2
INITSKIP
INITIALIZE THE COMPARE REGISTER TO ZERO
LOOP AX A2
IEDQAX
GET ADDRESS OF NEXT BYTE IN THE BUFFER
C3
AL A1
IEDQAL
GET SCAN PTR ADDR,EOU & CURRENT UNIT REGS
YES
IS THE NEXT BYTE A BLANK
4
RETURN CODE =
0
MBH
IS RETURN IN A REGISTER REQUESTED
NO
MBHPROC
MOVE THE CHARACTERS FOUND FROM THE AVT WORK AREA TO THE SCB
NO
YES
NO
IS THIS AN MBH ENTRY
ADD A CHARACTER TO THE AVT SAVE AREA; INCREMENT THE COMPARE REGISTER
IS RETURN IN REGISTER 15
YES
SAVE THE ADDRESS OF THE PARAMETER LIST IN THE SCB
YES
NO
NO
IS THE CHARACTER FOUND
ARE CHARS FOUND = CHARS PASSED
YES
IS THE COUNT COMPLETE
NO
PUT A RETURN CODE OF X'04' IN REGISTER 15
MOVE THE REGISTERS FROM THE AVT SAVE AREA TO THE SCB
C2
YES
NO
C3
YES
K5
GOODRTN
MULTBUFN
MOVE THE CHARACTERS FROM FROM THE SCB TO THE AVT WORK AREA
SHIFT
SHIFT THE CHARACTERS IN THE AVT WORK AREA ONE BYTE TO THE LEFT
IS RETURN IN A REGISTER REQUESTED
NO
SET THE PREFIX SCAN POINTER AT THE LAST BYTE OF CHARACTERS FOUND
SET THE PREFIX SCAN POINTER BEYOND THE END OF THE BUFFER
YES
H4
H5
REGRTN
ZERORTN
COMPUTE NUMBER OF CHARACTERS FOUND; PUT THE NUMBER IN THE COMPARE REG
YES
IS THE COMPARE REGISTER = ZERO
PUT OFFSET OF LAST BYTE OF CHARS FOUND IN THE RETURN REGISTER
PUT A RETURN CODE OF X'00' IN REGISTER 15
PUT A NEGATIVE RETURN CODE IN REGISTER 15
NO
S2
COMPUTE THE RETURN ADDRESS AND PUT IT IN THE REGISTER SAVE AREA
ARE REMAINING CHARS = PASSED CHARS
NO
IS RETURN IN REGISTER 15
NO
K5
YES
YES
RETURN
RETURN TO IEDQLM
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart AK-1 LINE CONTROL INSERTION ROUTINE

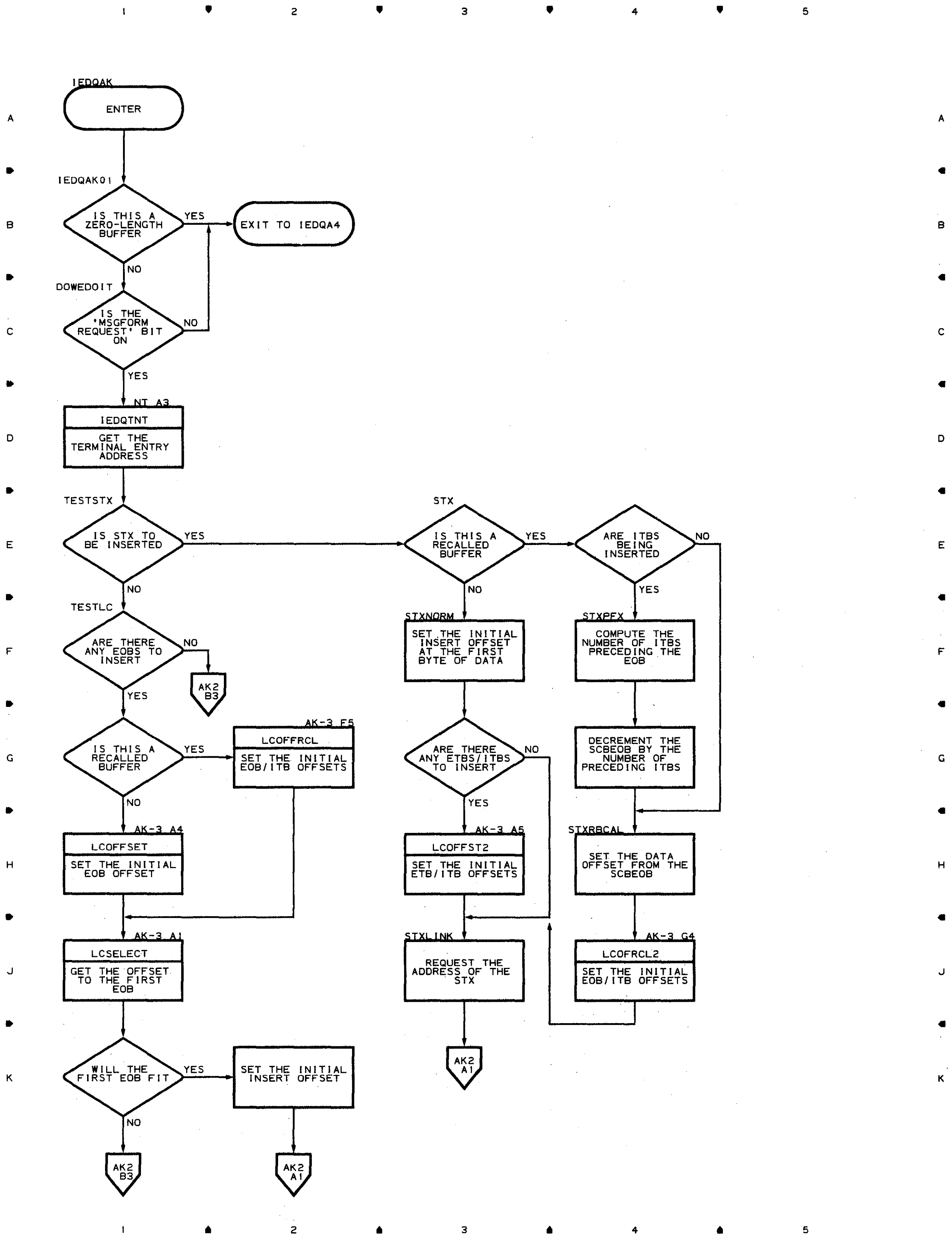

IEDQAK
ENTER
IEDQAK01
IS THIS A ZERO-LENGTH BUFFER
YES
EXIT TO IEDQA4
NO
DOWEDOIT
IS THE 'MSGFORM REQUEST' BIT ON
NO
YES
NT A3
IEDQTNT
GET THE TERMINAL ENTRY ADDRESS
TESTSTX
IS STX TO BE INSERTED
YES
NO
TESTLC
ARE THERE ANY EOBS TO INSERT
NO
AK2 B3
YES
IS THIS A RECALLED BUFFER
YES
AK-3 F5
LCOFFRCL
SET THE INITIAL EOB/ITB OFFSETS
NO
AK-3 A4
LCOFFSET
SET THE INITIAL EOB OFFSET
AK-3 A1
LCSELECT
GET THE OFFSET TO THE FIRST EOB
WILL THE FIRST EOB FIT
YES
SET THE INITIAL INSERT OFFSET
NO
AK2 B3
AK2 A1
STX
IS THIS A RECALLED BUFFER
YES
NO
STXNORM
SET THE INITIAL INSERT OFFSET AT THE FIRST BYTE OF DATA
ARE THERE ANY ETBS/ITBS TO INSERT
NO
YES
AK-3 A5
LCOFFST2
SET THE INITIAL ETB/ITB OFFSETS
STXLINK
REQUEST THE ADDRESS OF THE STX
AK2 A1
ARE ITBS BEING INSERTED
NO
YES
STXPFX
COMPUTE THE NUMBER OF ITBS PRECEDING THE EOB
DECREMENT THE SCBEOB BY THE NUMBER OF PRECEDING ITBS
STXRBCAL
SET THE DATA OFFSET FROM THE SCBEOB
AK-3 G4
LCOFRCL2
SET THE INITIAL EOB/ITB OFFSETS
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart AK-2 LINE CONTROL INSERTION ROUTINE

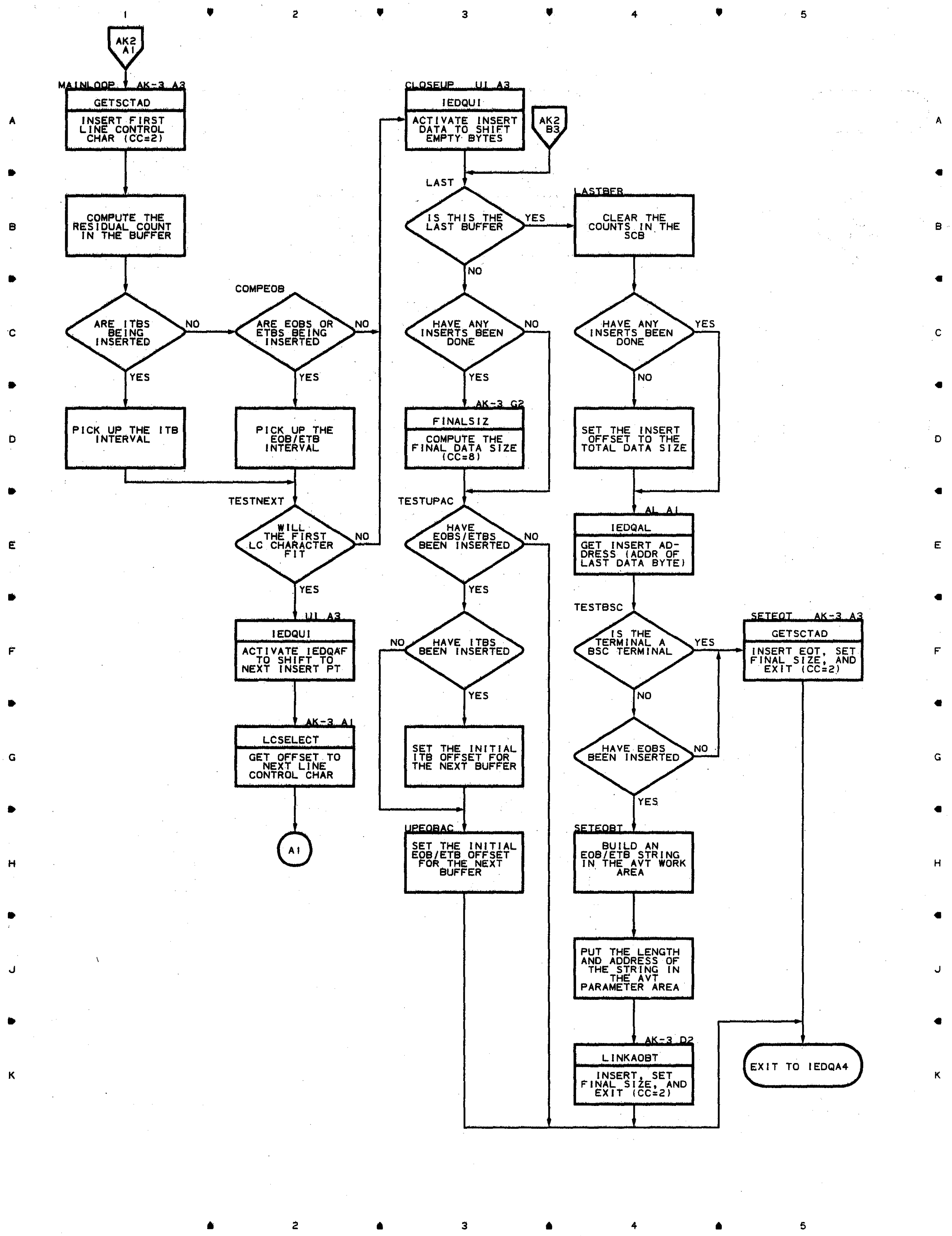

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
AK2 A1
MAINLOOP AK-3 A3
GETSCTAD
INSERT FIRST LINE CONTROL CHAR (CC=2)
COMPUTE THE RESIDUAL COUNT IN THE BUFFER
ARE ITBS BEING INSERTED
NO
YES
PICK UP THE ITB INTERVAL
COMPEOB
ARE EOBS OR ETBS BEING INSERTED
NO
YES
PICK UP THE EOB/ETB INTERVAL
TESTNEXT
WILL THE FIRST LC CHARACTER FIT
NO
YES
UI A3
IEDQUI
ACTIVATE IEDQAF TO SHIFT TO NEXT INSERT PT
AK-3 A1
LCSELECT
GET OFFSET TO NEXT LINE CONTROL CHAR
A1
CLOSEUP UI A3
IEDQUI
ACTIVATE INSERT DATA TO SHIFT EMPTY BYTES
AK2 B3
LAST
IS THIS THE LAST BUFFER
YES
NO
HAVE ANY INSERTS BEEN DONE
NO
YES
AK-3 G2
FINALSIZ
COMPUTE THE FINAL DATA SIZE (CC=8)
TESTUPAC
HAVE EOBS/ETBS BEEN INSERTED
NO
YES
HAVE ITBS BEEN INSERTED
NO
YES
SET THE INITIAL ITB OFFSET FOR THE NEXT BUFFER
UPEOBAC
SET THE INITIAL EOB/ETB OFFSET FOR THE NEXT BUFFER
LASTBFR
CLEAR THE COUNTS IN THE SCB
HAVE ANY INSERTS BEEN DONE
YES
NO
SET THE INSERT OFFSET TO THE TOTAL DATA SIZE
AL A1
IEDQAL
GET INSERT ADDRESS (ADDR OF LAST DATA BYTE)
TESTBSC
IS THE TERMINAL A BSC TERMINAL
YES
NO
HAVE EOBS BEEN INSERTED
NO
YES
SETEOBT
BUILD AN EOB/ETB STRING IN THE AVT WORK AREA
PUT THE LENGTH AND ADDRESS OF THE STRING IN THE AVT PARAMETER AREA
AK-3 D2
LINKAOBT
INSERT, SET FINAL SIZE, AND EXIT (CC=2)
SETEOT AK-3 A3
GETSCTAD
INSERT EOT, SET FINAL SIZE, AND EXIT (CC=2)
EXIT TO IEDQA4
2
3
4
5

Chart AK-3 LINE CONTROL INSERTION ROUTINE

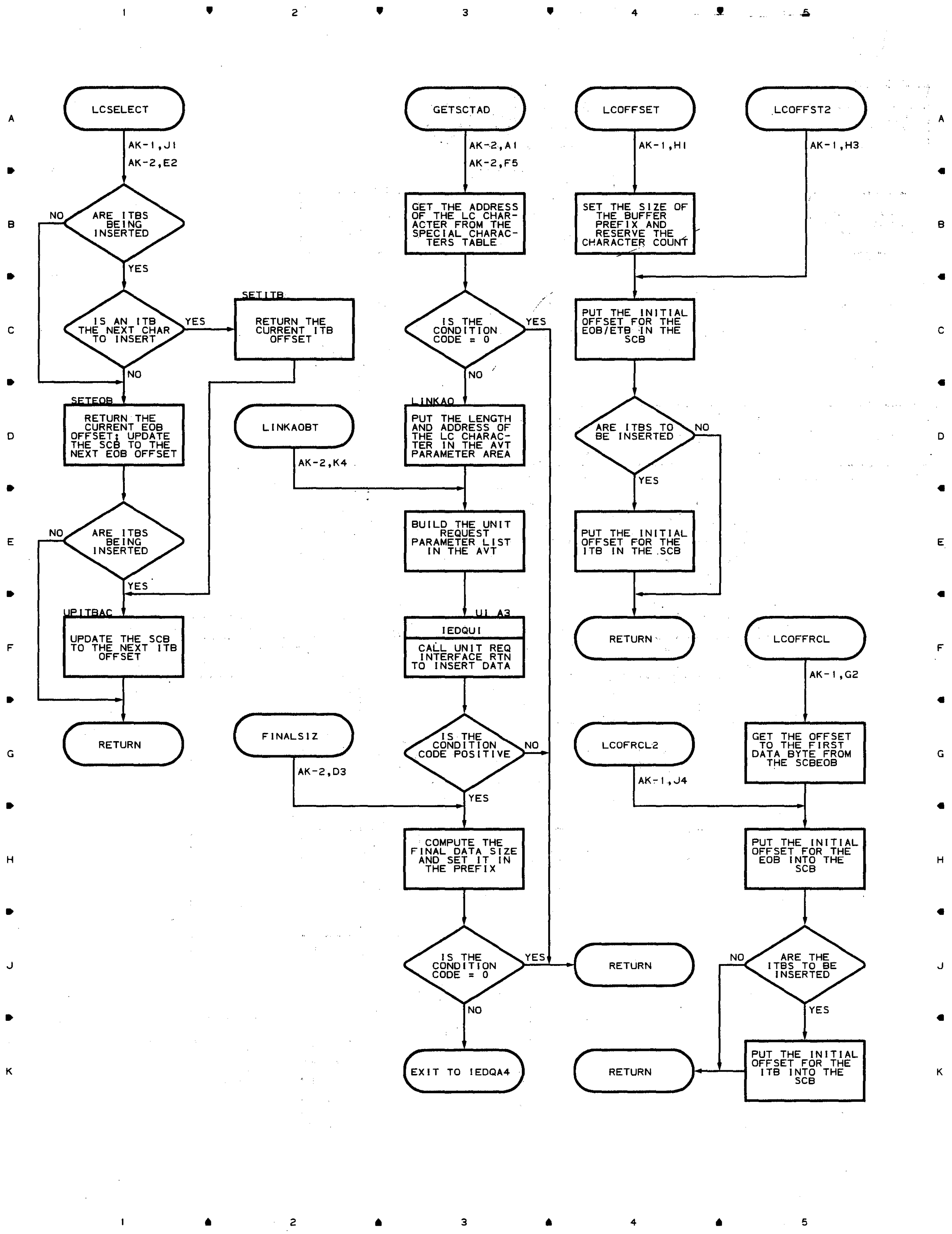
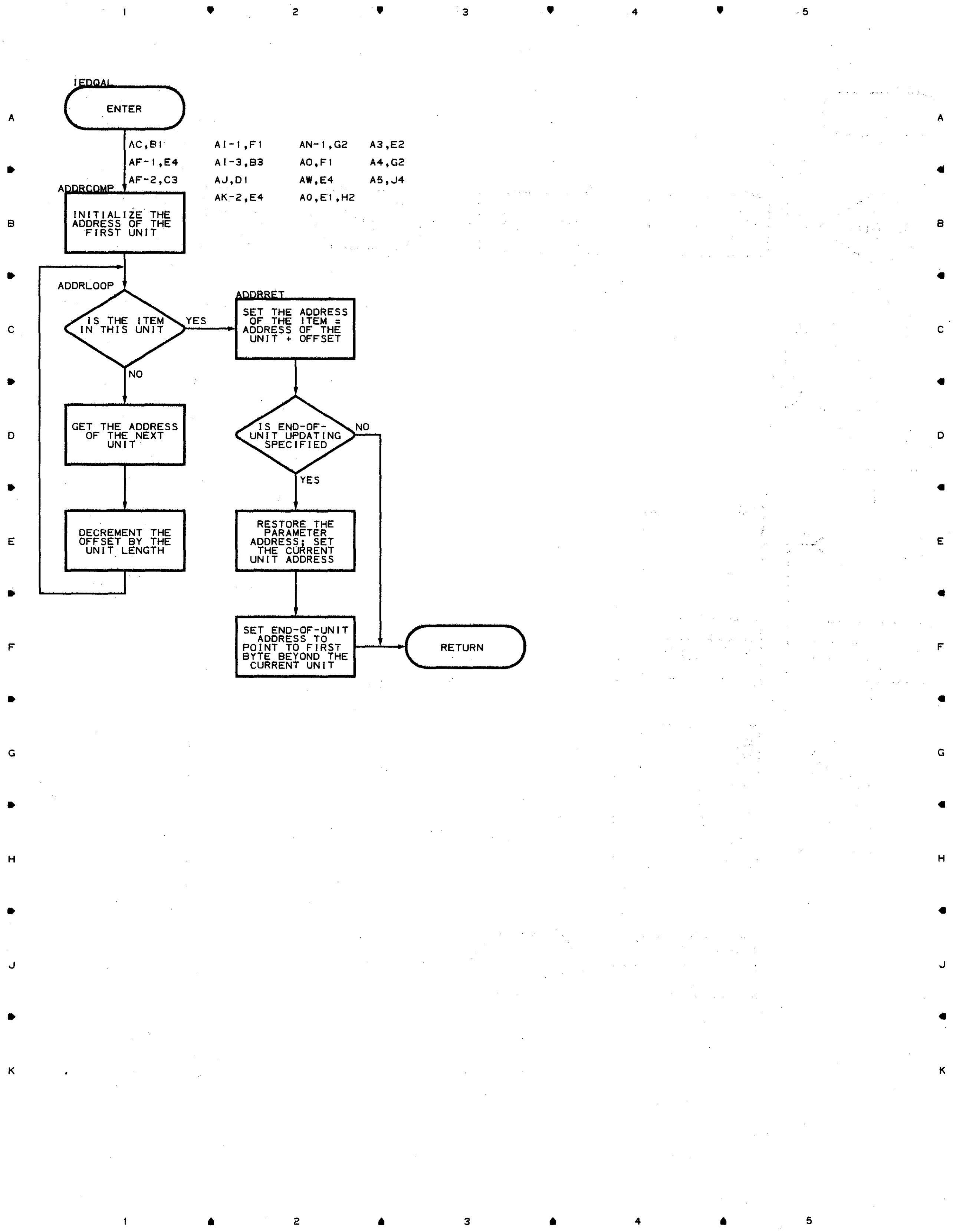
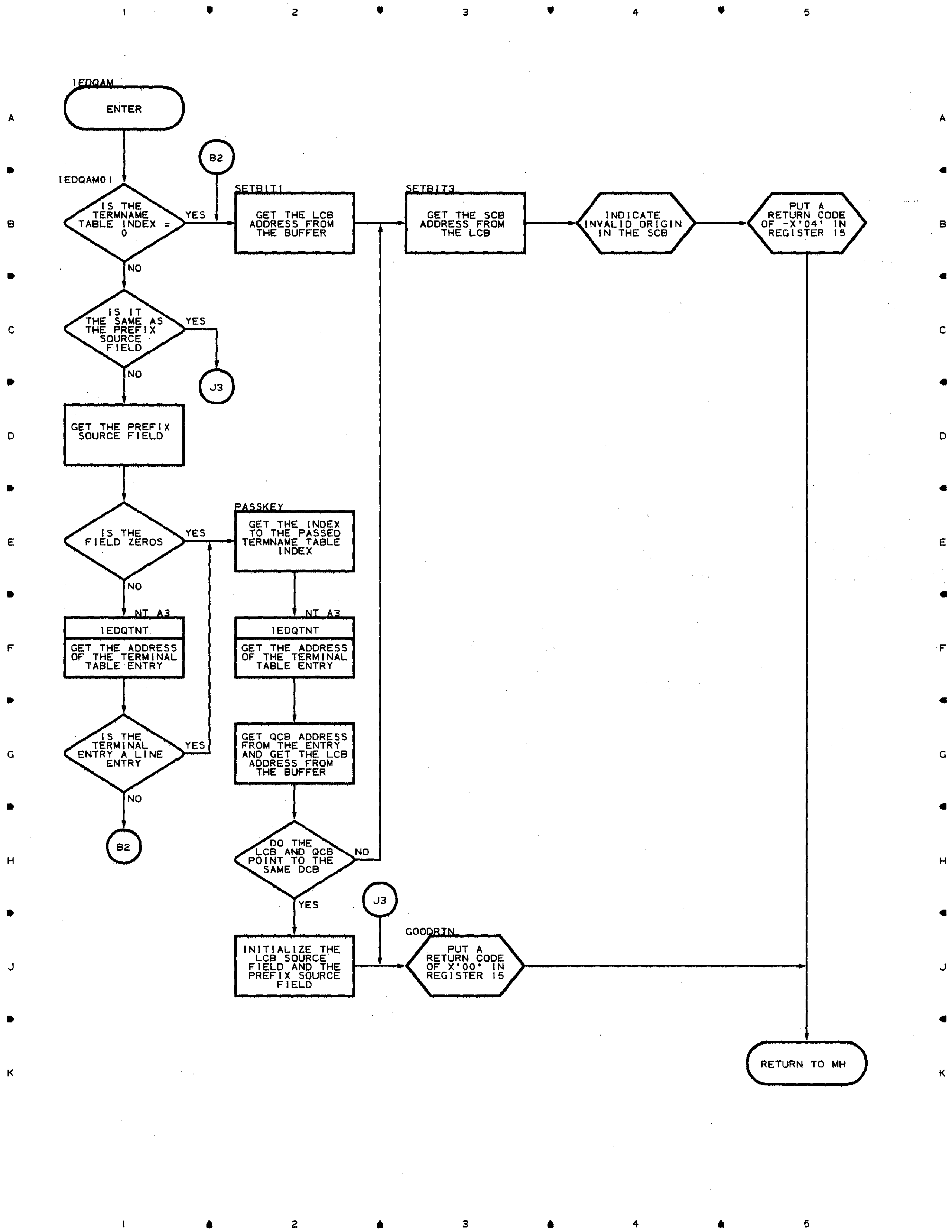

LCSELECT
AK-1,J1
AK-2,E2
ARE ITBS BEING INSERTED
NO
YES
IS AN ITB THE NEXT CHAR TO INSERT
YES
NO
SETITB
RETURN THE CURRENT ITB OFFSET
SETEOB
RETURN THE CURRENT EOB OFFSET; UPDATE THE SCB TO THE NEXT EOB OFFSET
ARE ITBS BEING INSERTED
NO
YES
UPITBAC
UPDATE THE SCB TO THE NEXT ITB OFFSET
RETURN
LINKAOBT
AK-2,K4
FINALSIZ
AK-2,D3
GETSCTAD
AK-2,A1
AK-2,F5
GET THE ADDRESS OF THE LC CHARACTER FROM THE SPECIAL CHARACTERS TABLE
IS THE CONDITION CODE = 0
YES
NO
LINKAO
PUT THE LENGTH AND ADDRESS OF THE LC CHARACTER IN THE AVT PARAMETER AREA
BUILD THE UNIT REQUEST PARAMETER LIST IN THE AVT
UI A3
IEDQUI
CALL UNIT REQ INTERFACE RTN TO INSERT DATA
IS THE CONDITION CODE POSITIVE
NO
YES
COMPUTE THE FINAL DATA SIZE AND SET IT IN THE PREFIX
IS THE CONDITION CODE = 0
YES
NO
EXIT TO IEDQA4
LCOFFSET
AK-1,H1
SET THE SIZE OF THE BUFFER PREFIX AND RESERVE THE CHARACTER COUNT
PUT THE INITIAL OFFSET FOR THE EOB/ETB IN THE SCB
ARE ITBS TO BE INSERTED
NO
YES
PUT THE INITIAL OFFSET FOR THE ITB IN THE SCB
RETURN
LCOFRCL2
AK-1,J4
RETURN
RETURN
LCOFFST2
AK-1,H3
LCOFFRCL
AK-1,G2
GET THE OFFSET TO THE FIRST DATA BYTE FROM THE SCBEOB
PUT THE INITIAL OFFSET FOR THE EOB INTO THE SCB
ARE THE ITBS TO BE INSERTED
NO
YES
PUT THE INITIAL OFFSET FOR THE ITB INTO THE SCB
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart AL   ADDRESS FINDER ROUTINE



IEDQAL
ENTER
AC,B1
AF-1,E4
AF-2,C3
AI-1,F1
AI-3,B3
AJ,D1
AK-2,E4
AN-1,G2
AO,F1
AW,E4
AO,E1,H2
A3,E2
A4,G2
A5,J4
ADDRCOMP
INITIALIZE THE ADDRESS OF THE FIRST UNIT
ADDRLOOP
IS THE ITEM IN THIS UNIT
YES
NO
ADDRRET
SET THE ADDRESS OF THE ITEM = ADDRESS OF THE UNIT + OFFSET
GET THE ADDRESS OF THE NEXT UNIT
IS END-OF-UNIT UPDATING SPECIFIED
NO
YES
DECREMENT THE OFFSET BY THE UNIT LENGTH
RESTORE THE PARAMETER ADDRESS; SET THE CURRENT UNIT ADDRESS
SET END-OF-UNIT ADDRESS TO POINT TO FIRST BYTE BEYOND THE CURRENT UNIT
RETURN

Chart AM ORIGIN ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQAM
ENTER
IEDQAM01
IS THE TERMNAME TABLE INDEX = 0
YES
NO
B2
SETBIT1
GET THE LCB ADDRESS FROM THE BUFFER
SETBIT3
GET THE SCB ADDRESS FROM THE LCB
INDICATE INVALID ORIGIN IN THE SCB
PUT A RETURN CODE OF -X'04' IN REGISTER 15
IS IT THE SAME AS THE PREFIX SOURCE FIELD
YES
NO
J3
GET THE PREFIX SOURCE FIELD
IS THE FIELD ZEROS
YES
NO
PASSKEY
GET THE INDEX TO THE PASSED TERMNAME TABLE INDEX
NT A3
IEDQTNT
GET THE ADDRESS OF THE TERMINAL TABLE ENTRY
NT A3
IEDQTNT
GET THE ADDRESS OF THE TERMINAL TABLE ENTRY
IS THE TERMINAL ENTRY A LINE ENTRY
YES
NO
GET QCB ADDRESS FROM THE ENTRY AND GET THE LCB ADDRESS FROM THE BUFFER
B2
DO THE LCB AND QCB POINT TO THE SAME DCB
NO
YES
J3
INITIALIZE THE LCB SOURCE FIELD AND THE PREFIX SOURCE FIELD
GOODRTN
PUT A RETURN CODE OF X'00' IN REGISTER 15
RETURN TO MH

Chart AN-1 MULTIPLE INSERT/REMOVE ROUTINE

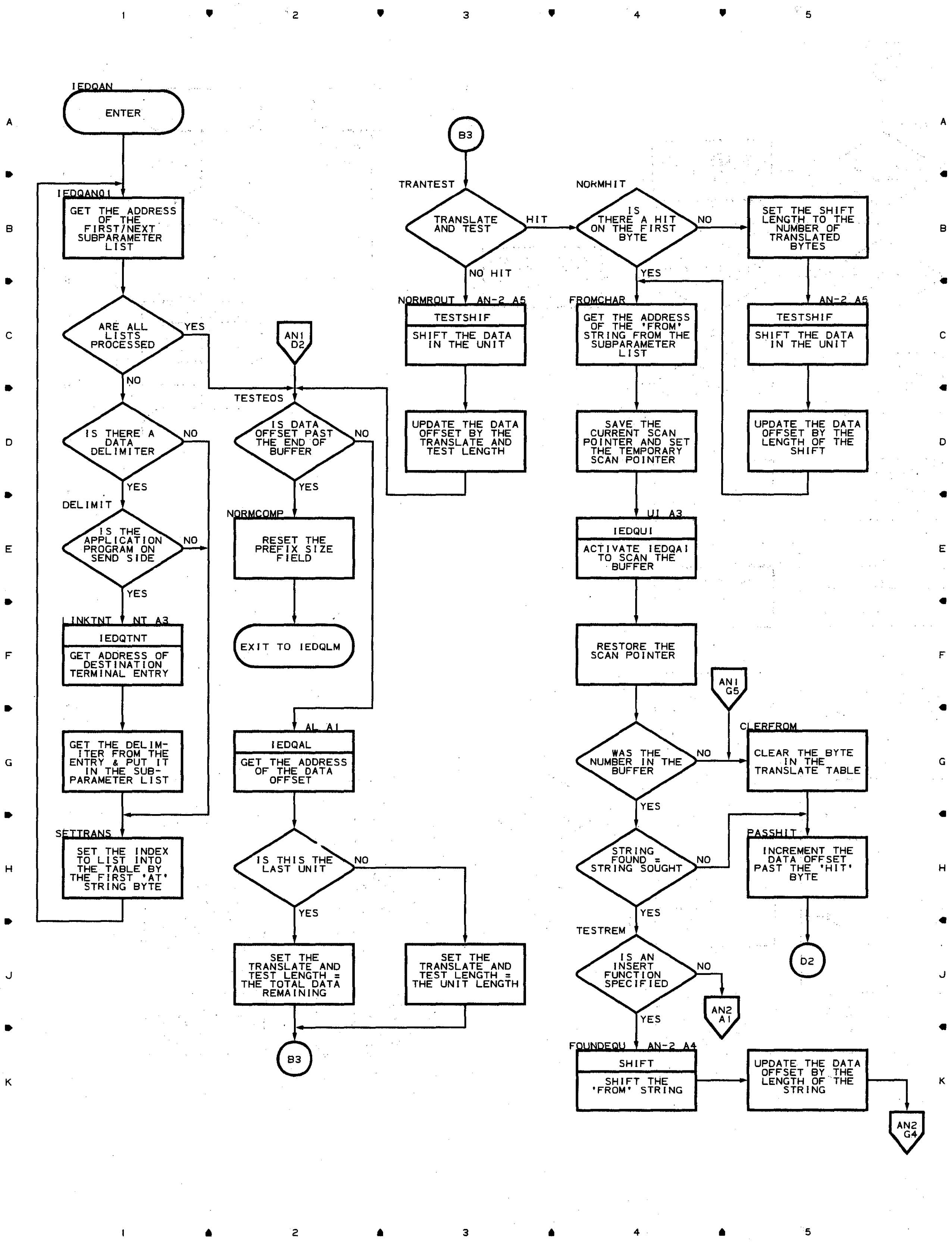
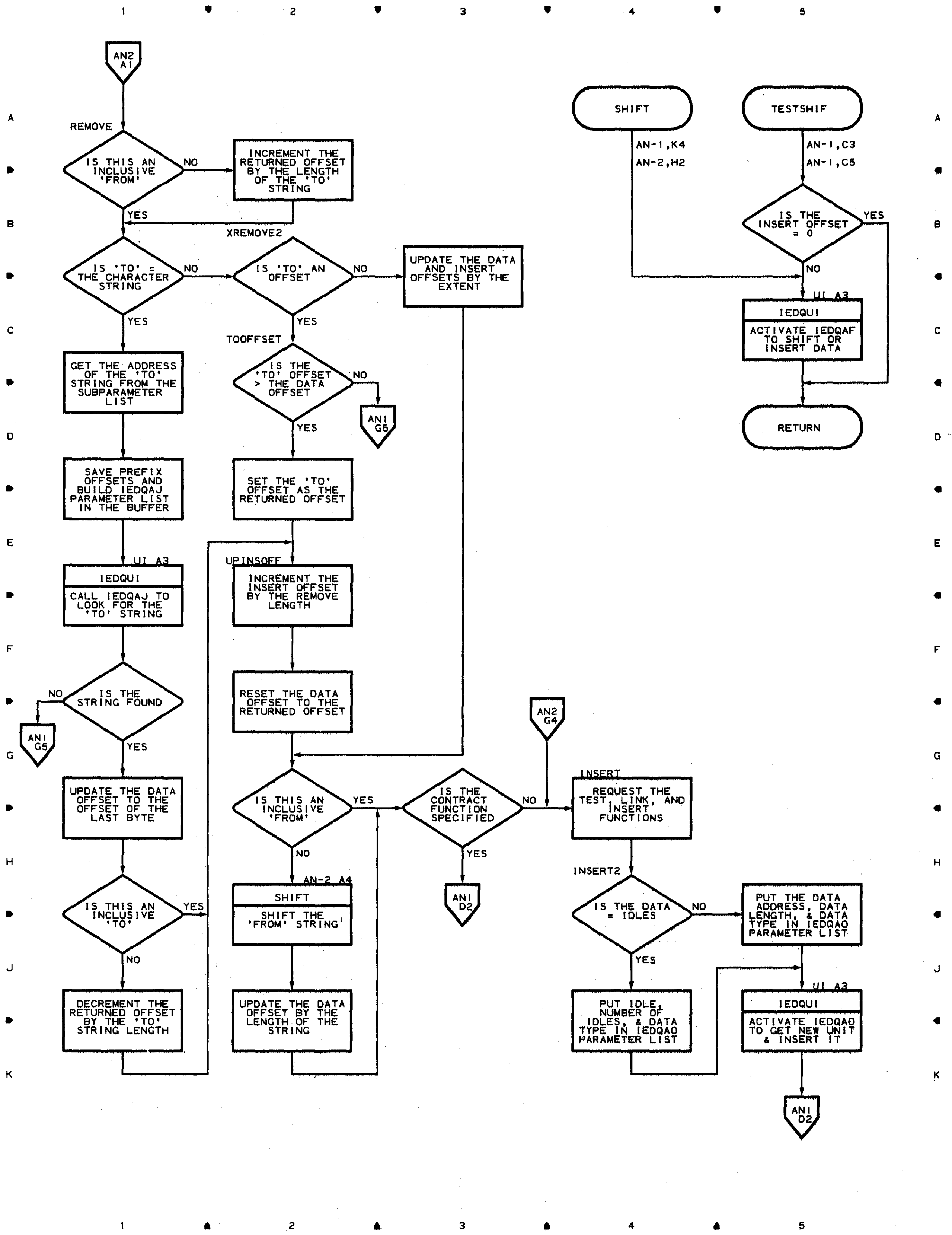

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQAN
ENTER
IEDQAN01
GET THE ADDRESS OF THE FIRST/NEXT SUBPARAMETER LIST
ARE ALL LISTS PROCESSED
YES
NO
IS THERE A DATA DELIMITER
NO
YES
DELIMIT
IS THE APPLICATION PROGRAM ON SEND SIDE
NO
YES
LINKTNT
NT A3
IEDQTNT
GET ADDRESS OF DESTINATION TERMINAL ENTRY
GET THE DELIMITER FROM THE ENTRY & PUT IT IN THE SUB-PARAMETER LIST
SETTRANS
SET THE INDEX TO LIST INTO THE TABLE BY THE FIRST 'AT' STRING BYTE
AN1 D2
TESTEOS
IS DATA OFFSET PAST THE END OF BUFFER
NO
YES
NORMCOMP
RESET THE PREFIX SIZE FIELD
EXIT TO IEDQLM
AL A1
IEDQAL
GET THE ADDRESS OF THE DATA OFFSET
IS THIS THE LAST UNIT
NO
YES
SET THE TRANSLATE AND TEST LENGTH = THE TOTAL DATA REMAINING
B3
B3
TRANTEST
TRANSLATE AND TEST
HIT
NO HIT
NORMROUT
AN-2 A5
TESTSHIF
SHIFT THE DATA IN THE UNIT
UPDATE THE DATA OFFSET BY THE TRANSLATE AND TEST LENGTH
SET THE TRANSLATE AND TEST LENGTH = THE UNIT LENGTH
NORMHIT
IS THERE A HIT ON THE FIRST BYTE
NO
YES
FROMCHAR
GET THE ADDRESS OF THE 'FROM' STRING FROM THE SUBPARAMETER LIST
SAVE THE CURRENT SCAN POINTER AND SET THE TEMPORARY SCAN POINTER
UI A3
IEDQUI
ACTIVATE IEDQAI TO SCAN THE BUFFER
RESTORE THE SCAN POINTER
WAS THE NUMBER IN THE BUFFER
NO
YES
STRING FOUND = STRING SOUGHT
NO
YES
TESTREM
IS AN INSERT FUNCTION SPECIFIED
NO
YES
AN2 A1
FOUNDEQU
AN-2 A4
SHIFT
SHIFT THE 'FROM' STRING
UPDATE THE DATA OFFSET BY THE LENGTH OF THE STRING
AN2 G4
SET THE SHIFT LENGTH TO THE NUMBER OF TRANSLATED BYTES
AN-2 A5
TESTSHIF
SHIFT THE DATA IN THE UNIT
UPDATE THE DATA OFFSET BY THE LENGTH OF THE SHIFT
AN1 G5
CLERFROM
CLEAR THE BYTE IN THE TRANSLATE TABLE
PASSHIT
INCREMENT THE DATA OFFSET PAST THE 'HIT' BYTE
D2

Chart AN-2 MULTIPLE INSERT/REMOVE ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
AN2 A1
REMOVE
IS THIS AN INCLUSIVE 'FROM'
NO
INCREMENT THE RETURNED OFFSET BY THE LENGTH OF THE 'TO' STRING
YES
XREMOVE2
IS 'TO' = THE CHARACTER STRING
NO
IS 'TO' AN OFFSET
NO
UPDATE THE DATA AND INSERT OFFSETS BY THE EXTENT
YES
YES
TOOFFSET
GET THE ADDRESS OF THE 'TO' STRING FROM THE SUBPARAMETER LIST
IS THE 'TO' OFFSET > THE DATA OFFSET
NO
AN1 G5
YES
SAVE PREFIX OFFSETS AND BUILD IEDQAJ PARAMETER LIST IN THE BUFFER
SET THE 'TO' OFFSET AS THE RETURNED OFFSET
UI A3
IEDQUI
CALL IEDQAJ TO LOOK FOR THE 'TO' STRING
UPINSOFF
INCREMENT THE INSERT OFFSET BY THE REMOVE LENGTH
NO
IS THE STRING FOUND
AN1 G5
YES
RESET THE DATA OFFSET TO THE RETURNED OFFSET
UPDATE THE DATA OFFSET TO THE OFFSET OF THE LAST BYTE
IS THIS AN INCLUSIVE 'TO'
YES
NO
DECREMENT THE RETURNED OFFSET BY THE 'TO' STRING LENGTH
IS THIS AN INCLUSIVE 'FROM'
YES
NO
AN-2 A4
SHIFT
SHIFT THE 'FROM' STRING
UPDATE THE DATA OFFSET BY THE LENGTH OF THE STRING
IS THE CONTRACT FUNCTION SPECIFIED
NO
YES
AN1 D2
AN2 G4
INSERT
REQUEST THE TEST, LINK, AND INSERT FUNCTIONS
INSERT2
IS THE DATA = IDLES
NO
YES
PUT THE DATA ADDRESS, DATA LENGTH, & DATA TYPE IN IEDQAO PARAMETER LIST
PUT IDLE, NUMBER OF IDLES, & DATA TYPE IN IEDQAO PARAMETER LIST
UI A3
IEDQUI
ACTIVATE IEDQAO TO GET NEW UNIT & INSERT IT
AN1 D2
SHIFT
AN-1,K4
AN-2,H2
TESTSHIF
AN-1,C3
AN-1,C5
IS THE INSERT OFFSET = 0
YES
NO
UI A3
IEDQUI
ACTIVATE IEDQAF TO SHIFT OR INSERT DATA
RETURN

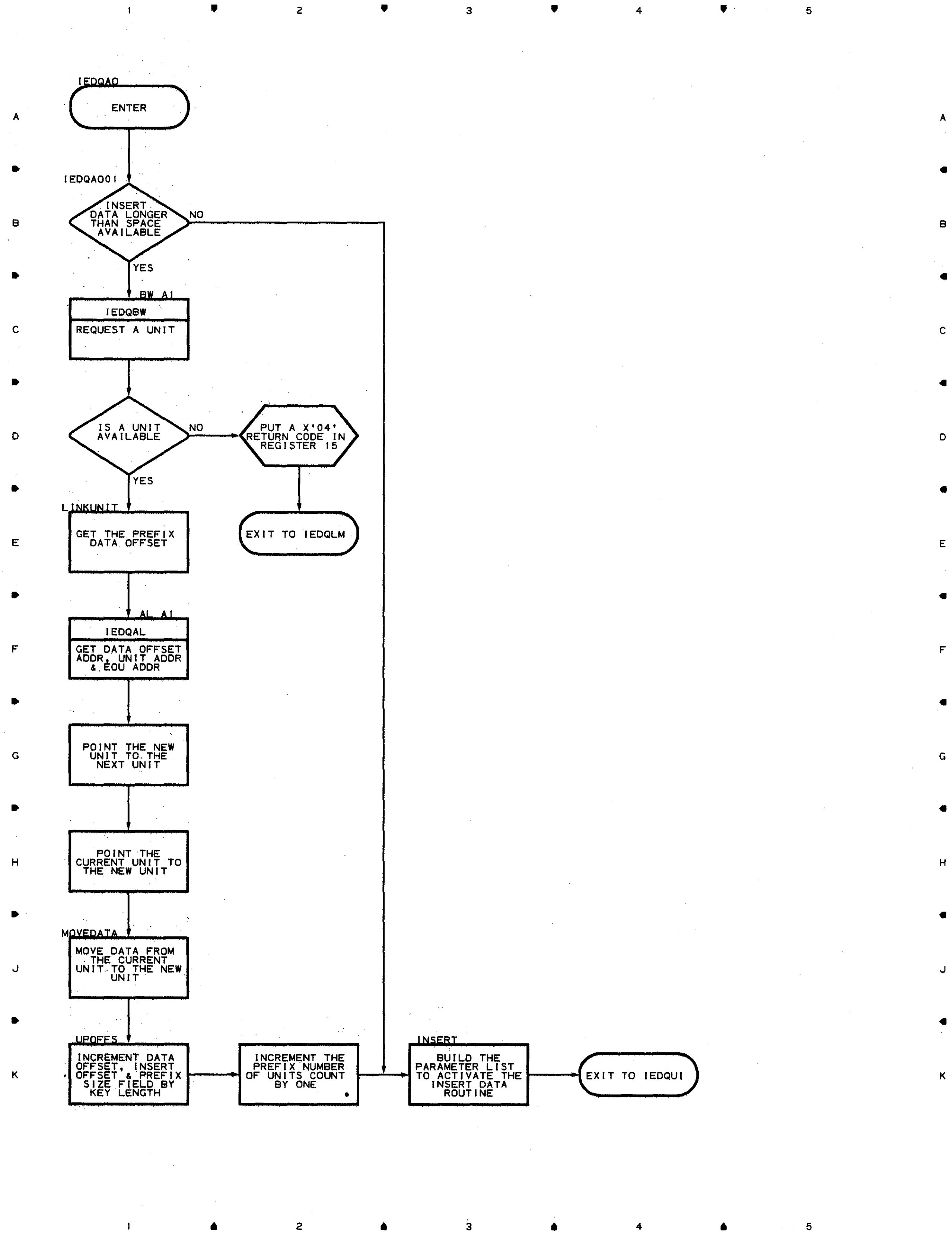

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQAO
ENTER
IEDQAO01
INSERT DATA LONGER THAN SPACE AVAILABLE
NO
YES
BW A1
IEDQBW
REQUEST A UNIT
IS A UNIT AVAILABLE
NO
PUT A X'04' RETURN CODE IN REGISTER 15
EXIT TO IEDQLM
YES
LINKUNIT
GET THE PREFIX DATA OFFSET
AL A1
IEDQAL
GET DATA OFFSET ADDR, UNIT ADDR & EOU ADDR
POINT THE NEW UNIT TO THE NEXT UNIT
POINT THE CURRENT UNIT TO THE NEW UNIT
MOVEDATA
MOVE DATA FROM THE CURRENT UNIT TO THE NEW UNIT
UPOFFS
INCREMENT DATA OFFSET, INSERT OFFSET & PREFIX SIZE FIELD BY KEY LENGTH
INCREMENT THE PREFIX NUMBER OF UNITS COUNT BY ONE
INSERT
BUILD THE PARAMETER LIST TO ACTIVATE THE INSERT DATA ROUTINE
EXIT TO IEDQUI

Chart AP-1 REMOVE AT OFFSET ROUTINE

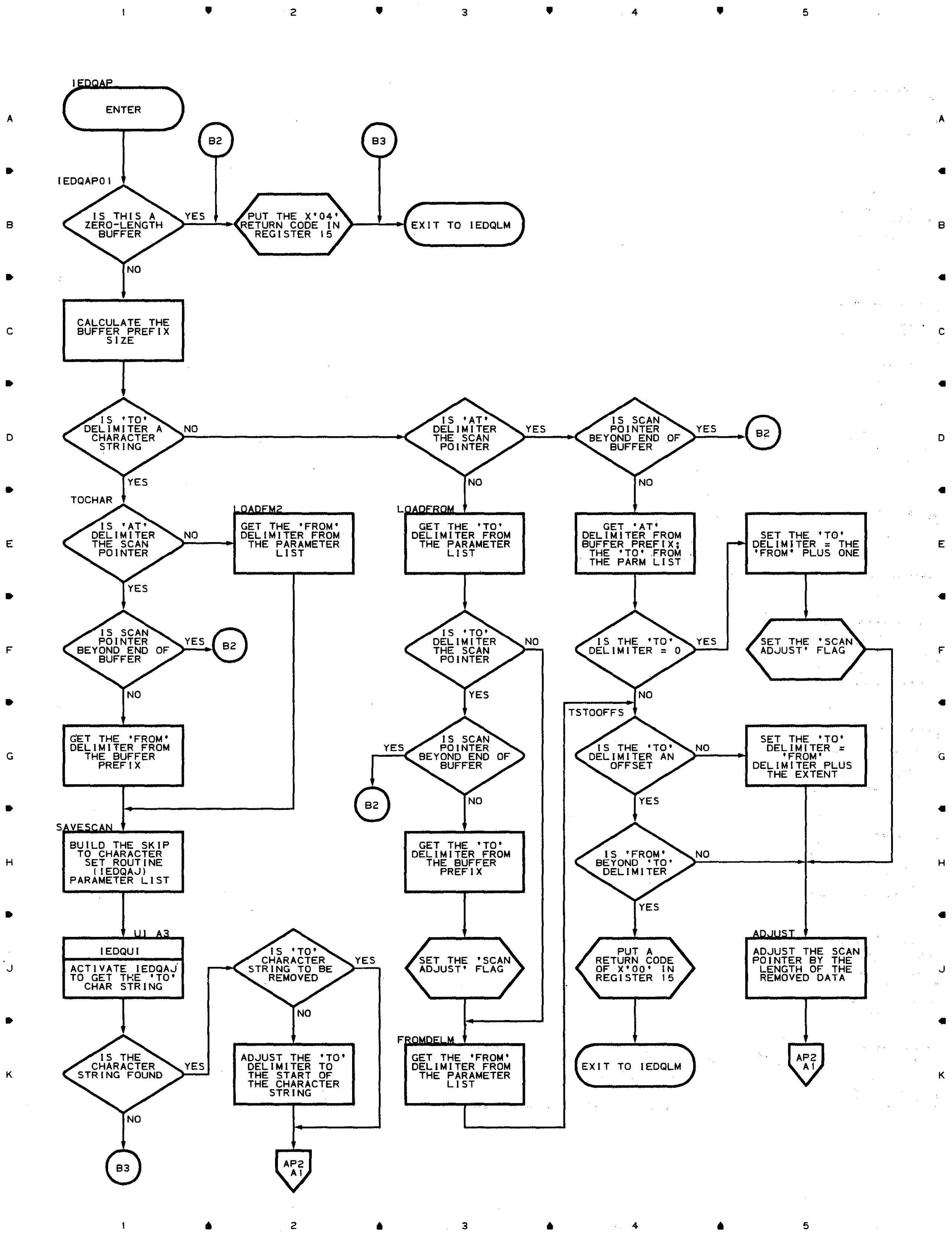
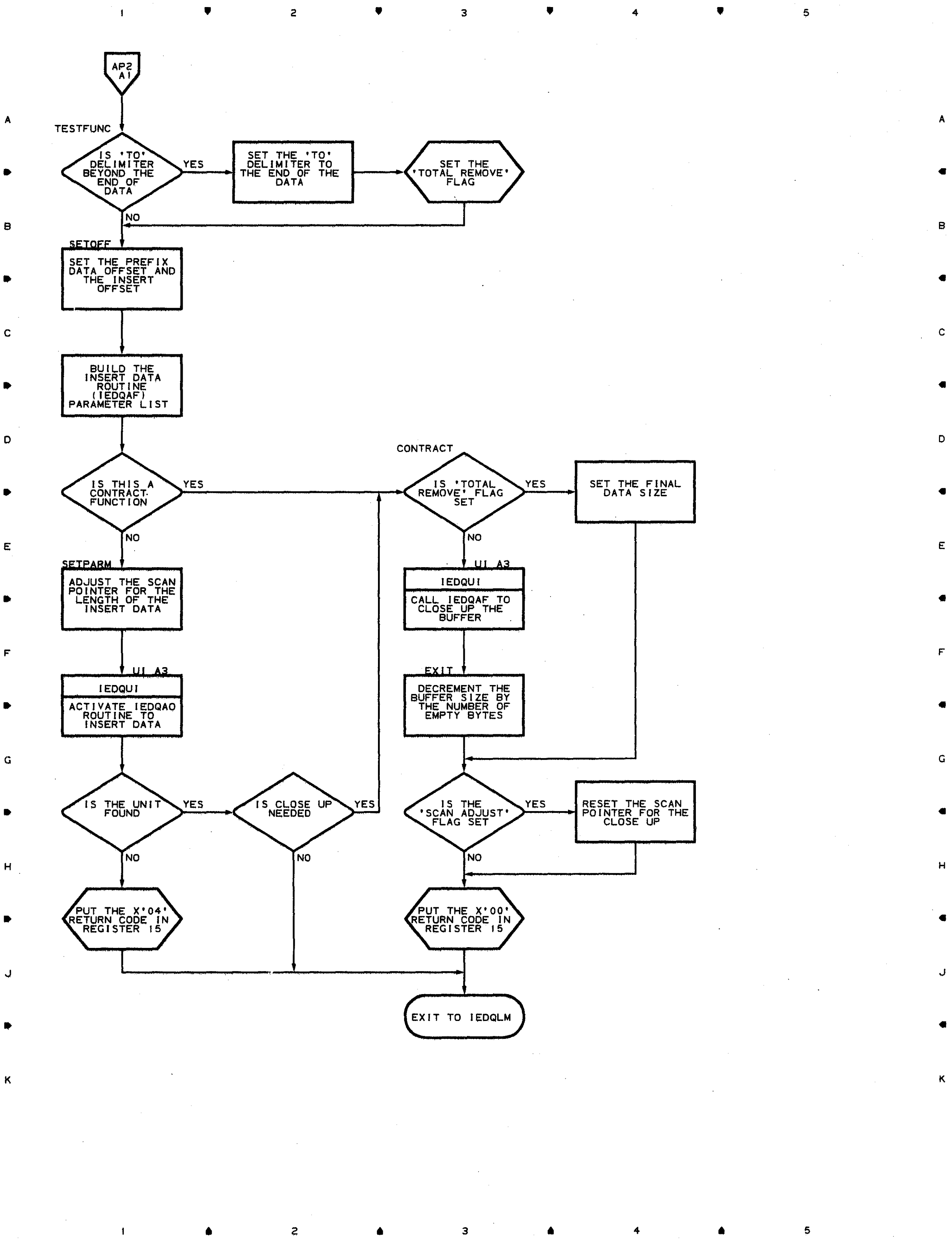

IEDQAP
ENTER
B2
B3
IEDQAP01
IS THIS A ZERO-LENGTH BUFFER
YES
NO
PUT THE X'04' RETURN CODE IN REGISTER 15
EXIT TO IEDQLM
CALCULATE THE BUFFER PREFIX SIZE
IS 'TO' DELIMITER A CHARACTER STRING
NO
YES
IS 'AT' DELIMITER THE SCAN POINTER
YES
NO
IS SCAN POINTER BEYOND END OF BUFFER
YES
NO
B2
TOCHAR
IS 'AT' DELIMITER THE SCAN POINTER
NO
YES
LOADFM2
GET THE 'FROM' DELIMITER FROM THE PARAMETER LIST
LOADFROM
GET THE 'TO' DELIMITER FROM THE PARAMETER LIST
GET 'AT' DELIMITER FROM BUFFER PREFIX; THE 'TO' FROM THE PARM LIST
SET THE 'TO' DELIMITER = THE 'FROM' PLUS ONE
IS SCAN POINTER BEYOND END OF BUFFER
YES
NO
B2
IS 'TO' DELIMITER THE SCAN POINTER
NO
YES
IS THE 'TO' DELIMITER = 0
YES
NO
SET THE 'SCAN ADJUST' FLAG
TSTOOFFS
GET THE 'FROM' DELIMITER FROM THE BUFFER PREFIX
IS SCAN POINTER BEYOND END OF BUFFER
YES
NO
B2
IS THE 'TO' DELIMITER AN OFFSET
NO
YES
SET THE 'TO' DELIMITER = 'FROM' DELIMITER PLUS THE EXTENT
SAVESCAN
BUILD THE SKIP TO CHARACTER SET ROUTINE (IEDQAJ) PARAMETER LIST
GET THE 'TO' DELIMITER FROM THE BUFFER PREFIX
IS 'FROM' BEYOND 'TO' DELIMITER
NO
YES
UI A3
IEDQUI
ACTIVATE IEDQAJ TO GET THE 'TO' CHAR STRING
IS 'TO' CHARACTER STRING TO BE REMOVED
YES
NO
SET THE 'SCAN ADJUST' FLAG
PUT A RETURN CODE OF X'00' IN REGISTER 15
ADJUST
ADJUST THE SCAN POINTER BY THE LENGTH OF THE REMOVED DATA
IS THE CHARACTER STRING FOUND
YES
NO
ADJUST THE 'TO' DELIMITER TO THE START OF THE CHARACTER STRING
FROMDELM
GET THE 'FROM' DELIMITER FROM THE PARAMETER LIST
EXIT TO IEDQLM
AP2 A1
B3
AP2 A1
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart AP-2 REMOVE AT OFFSET ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
AP2 A1
TESTFUNC
IS 'TO' DELIMITER BEYOND THE END OF DATA
YES
SET THE 'TO' DELIMITER TO THE END OF THE DATA
SET THE 'TOTAL REMOVE' FLAG
NO
SETOFF
SET THE PREFIX DATA OFFSET AND THE INSERT OFFSET
BUILD THE INSERT DATA ROUTINE (IEDQAF) PARAMETER LIST
IS THIS A CONTRACT FUNCTION
YES
CONTRACT
IS 'TOTAL REMOVE' FLAG SET
YES
SET THE FINAL DATA SIZE
NO
NO
SETPARM
ADJUST THE SCAN POINTER FOR THE LENGTH OF THE INSERT DATA
UI A3
IEDQUI
CALL IEDQAF TO CLOSE UP THE BUFFER
UI A3
IEDQUI
ACTIVATE IEDQAO ROUTINE TO INSERT DATA
EXIT
DECREMENT THE BUFFER SIZE BY THE NUMBER OF EMPTY BYTES
IS THE UNIT FOUND
YES
IS CLOSE UP NEEDED
YES
IS THE 'SCAN ADJUST' FLAG SET
YES
RESET THE SCAN POINTER FOR THE CLOSE UP
NO
NO
NO
PUT THE X'04' RETURN CODE IN REGISTER 15
PUT THE X'00' RETURN CODE IN REGISTER 15
EXIT TO IEDQLM

OPERATOR CONTROL INTERFACE ROUTINE

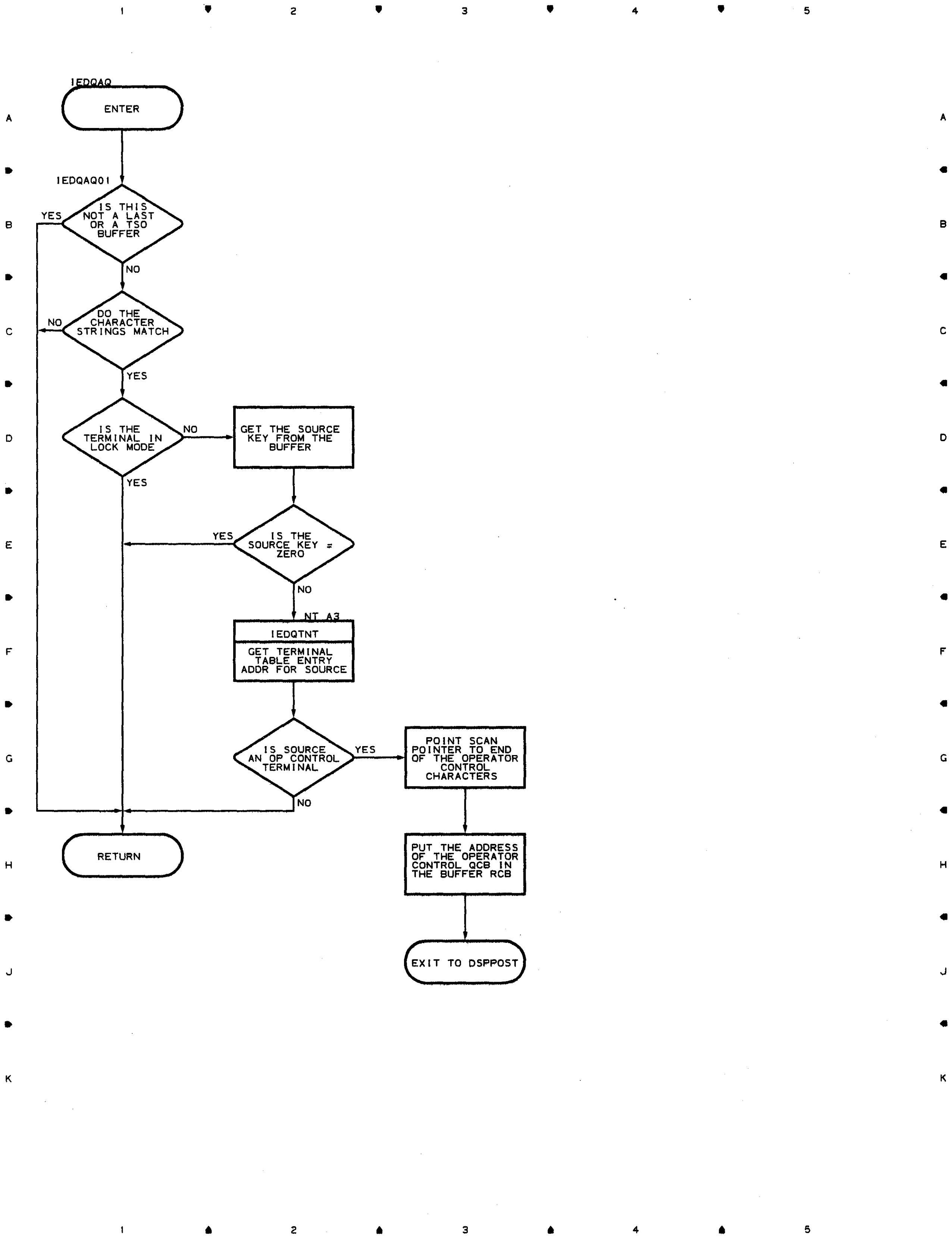

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQAQ
ENTER
IEDQAQ01
IS THIS NOT A LAST OR A TSO BUFFER
YES
NO
DO THE CHARACTER STRINGS MATCH
NO
YES
IS THE TERMINAL IN LOCK MODE
NO
GET THE SOURCE KEY FROM THE BUFFER
YES
IS THE SOURCE KEY = ZERO
YES
NO
NT A3
IEDQTNT
GET TERMINAL TABLE ENTRY ADDR FOR SOURCE
IS SOURCE AN OP CONTROL TERMINAL
YES
POINT SCAN POINTER TO END OF THE OPERATOR CONTROL CHARACTERS
NO
RETURN
PUT THE ADDRESS OF THE OPERATOR CONTROL QCB IN THE BUFFER RCB
EXIT TO DSPPOST

Chart AR CANCEL MESSAGE ROUTINE

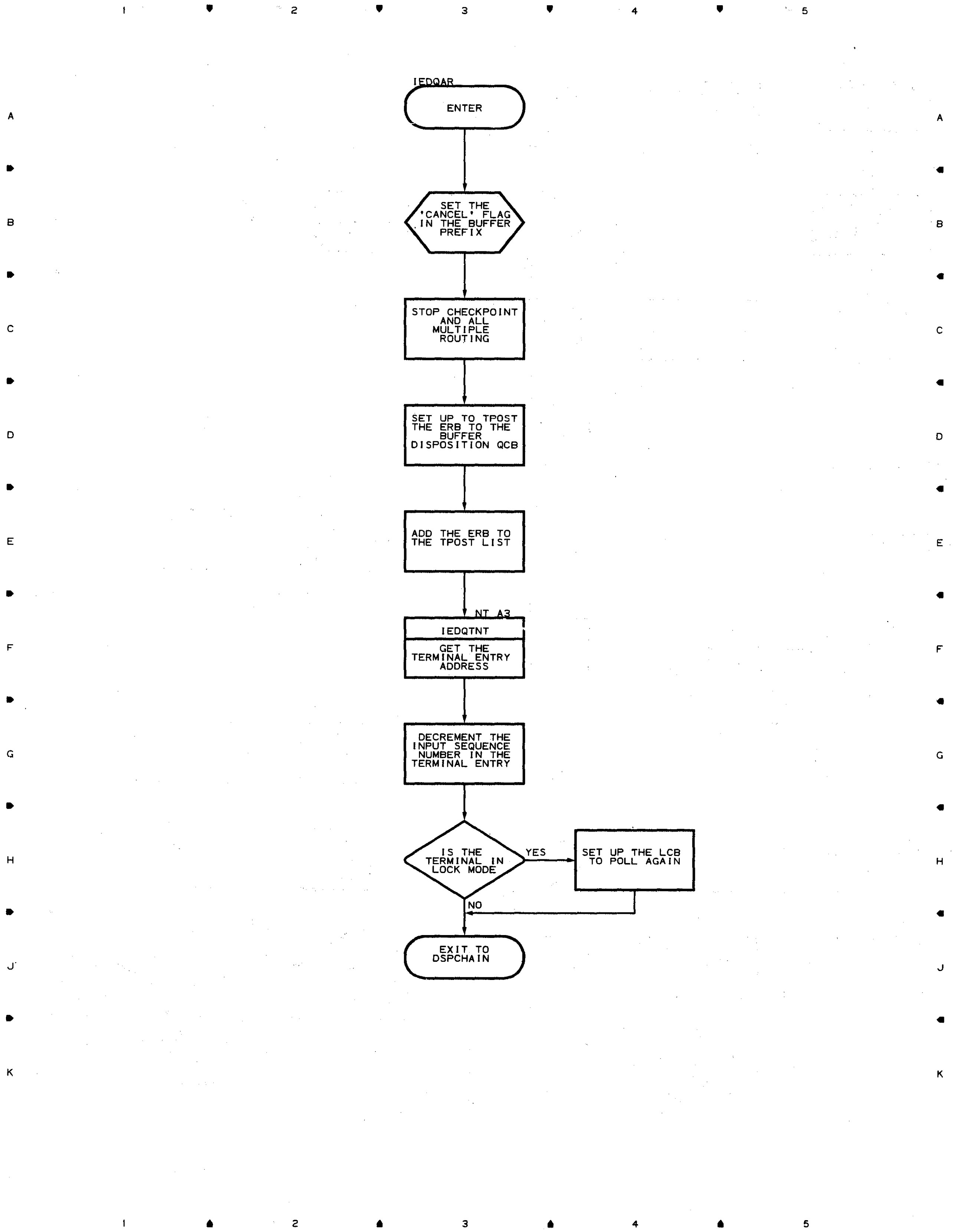

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQAR
ENTER
SET THE 'CANCEL' FLAG IN THE BUFFER PREFIX
STOP CHECKPOINT AND ALL MULTIPLE ROUTING
SET UP TO TPOST THE ERB TO THE BUFFER DISPOSITION QCB
ADD THE ERB TO THE TPOST LIST
NT A3
IEDQTNT
GET THE TERMINAL ENTRY ADDRESS
DECREMENT THE INPUT SEQUENCE NUMBER IN THE TERMINAL ENTRY
IS THE TERMINAL IN LOCK MODE
YES
SET UP THE LCB TO POLL AGAIN
NO
EXIT TO DSPCHAIN

Chart AS-1 HOLD/RELEASE TERMINAL ROUTINE

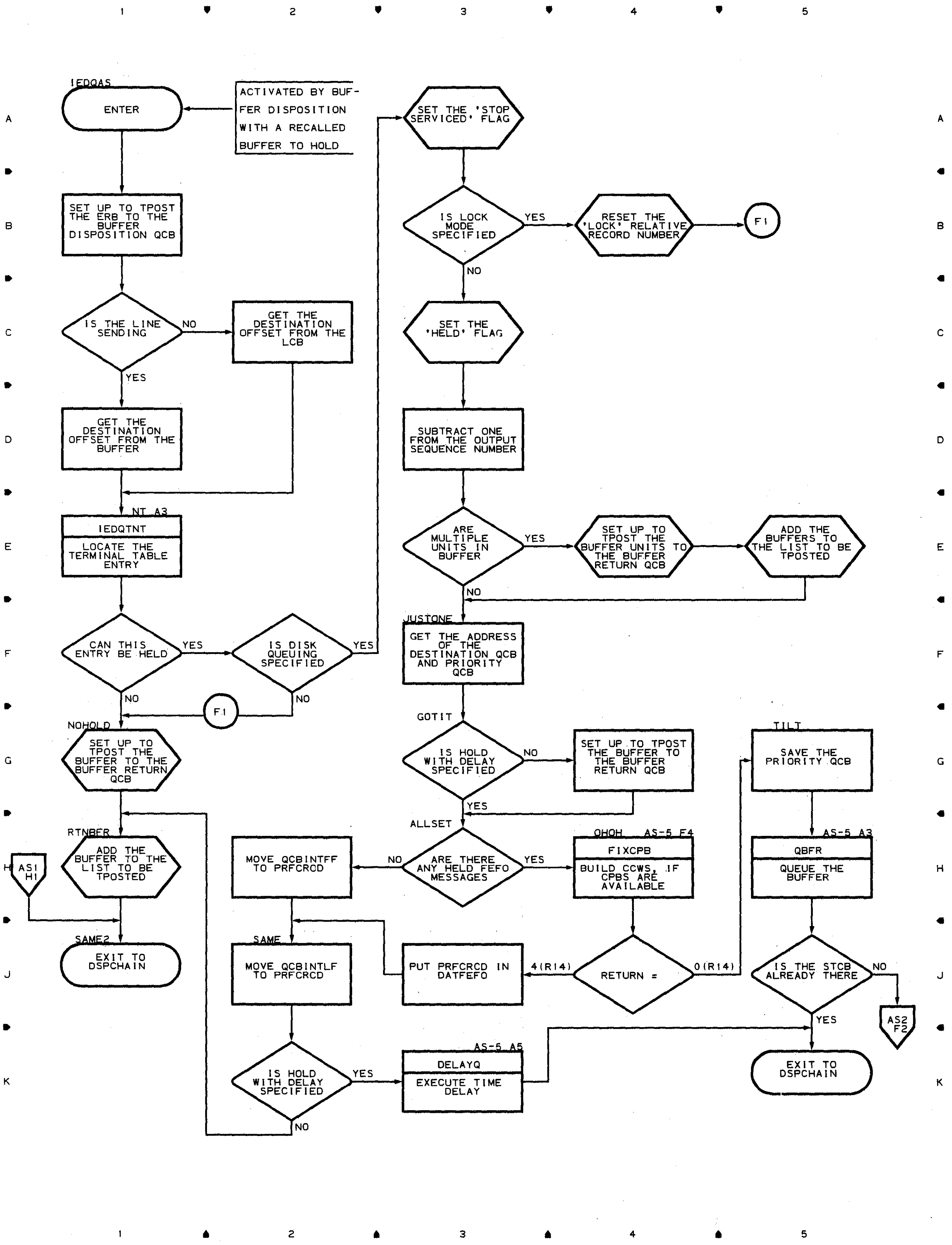
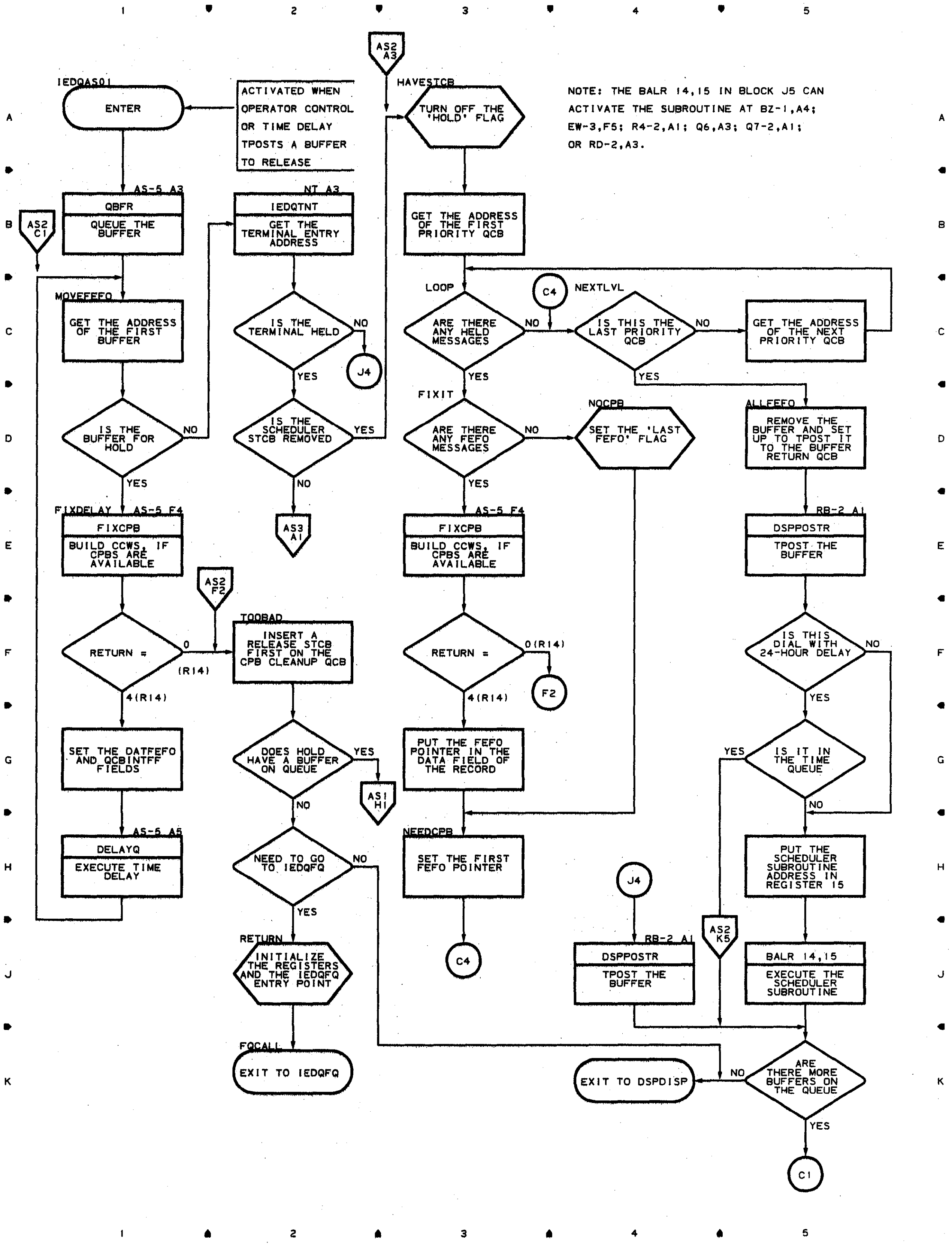

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQAS
ENTER
ACTIVATED BY BUFFER DISPOSITION WITH A RECALLED BUFFER TO HOLD
SET UP TO TPOST THE ERB TO THE BUFFER DISPOSITION QCB
IS THE LINE SENDING
NO
YES
GET THE DESTINATION OFFSET FROM THE LCB
GET THE DESTINATION OFFSET FROM THE BUFFER
NT A3
IEDQTNT
LOCATE THE TERMINAL TABLE ENTRY
CAN THIS ENTRY BE HELD
YES
NO
IS DISK QUEUING SPECIFIED
YES
NO
F1
NOHOLD
SET UP TO TPOST THE BUFFER TO THE BUFFER RETURN QCB
RTNBFR
ADD THE BUFFER TO THE LIST TO BE TPOSTED
AS1 H1
SAME2
EXIT TO DSPCHAIN
SET THE 'STOP SERVICED' FLAG
IS LOCK MODE SPECIFIED
YES
NO
RESET THE 'LOCK' RELATIVE RECORD NUMBER
F1
SET THE 'HELD' FLAG
SUBTRACT ONE FROM THE OUTPUT SEQUENCE NUMBER
ARE MULTIPLE UNITS IN BUFFER
YES
NO
SET UP TO TPOST THE BUFFER UNITS TO THE BUFFER RETURN QCB
ADD THE BUFFERS TO THE LIST TO BE TPOSTED
JUSTONE
GET THE ADDRESS OF THE DESTINATION QCB AND PRIORITY QCB
GOTIT
IS HOLD WITH DELAY SPECIFIED
NO
YES
SET UP TO TPOST THE BUFFER TO THE BUFFER RETURN QCB
ALLSET
ARE THERE ANY HELD FEFO MESSAGES
NO
YES
MOVE QCBINTFF TO PRFCRCD
QHQH AS-5 F4
FIXCPB
BUILD CCWS, IF CPBS ARE AVAILABLE
SAME
MOVE QCBINTLF TO PRFCRCD
PUT PRFCRCD IN DATFEFO
4(R14)
RETURN =
0(R14)
IS HOLD WITH DELAY SPECIFIED
YES
NO
AS-5 A5
DELAYQ
EXECUTE TIME DELAY
TILT
SAVE THE PRIORITY QCB
AS-5 A3
QBFR
QUEUE THE BUFFER
IS THE STCB ALREADY THERE
NO
YES
AS2 F2
EXIT TO DSPCHAIN

Chart AS-2 HOLD/RELEASE TERMINAL ROUTINE



NOTE: THE BALR 14,15 IN BLOCK J5 CAN ACTIVATE THE SUBROUTINE AT BZ-1,A4; EW-3,F5; R4-2,A1; Q6,A3; Q7-2,A1; OR RD-2,A3.

IEDQAS01
ENTER
ACTIVATED WHEN OPERATOR CONTROL OR TIME DELAY TPOSTS A BUFFER TO RELEASE
AS2 A3
HAVESTCB
TURN OFF THE 'HOLD' FLAG
AS-5 A3
QBFR
QUEUE THE BUFFER
AS2 C1
NT A3
IEDQTNT
GET THE TERMINAL ENTRY ADDRESS
GET THE ADDRESS OF THE FIRST PRIORITY QCB
MOVEFEFO
GET THE ADDRESS OF THE FIRST BUFFER
IS THE TERMINAL HELD
NO
YES
J4
LOOP
ARE THERE ANY HELD MESSAGES
C4
NEXTLVL
IS THIS THE LAST PRIORITY QCB
GET THE ADDRESS OF THE NEXT PRIORITY QCB
FIXIT
NOCPB
ALLFEFO
IS THE BUFFER FOR HOLD
IS THE SCHEDULER STCB REMOVED
ARE THERE ANY FEFO MESSAGES
SET THE 'LAST FEFO' FLAG
REMOVE THE BUFFER AND SET UP TO TPOST IT TO THE BUFFER RETURN QCB
FIXDELAY
AS-5 F4
FIXCPB
BUILD CCWS, IF CPBS ARE AVAILABLE
AS3 A1
RB-2 A1
DSPPOSTR
TPOST THE BUFFER
AS2 F2
TOOBAD
RETURN =
0
(R14)
4(R14)
INSERT A RELEASE STCB FIRST ON THE CPB CLEANUP QCB
0(R14)
F2
IS THIS DIAL WITH 24-HOUR DELAY
SET THE DATFEFO AND QCBINTFF FIELDS
DOES HOLD HAVE A BUFFER ON QUEUE
AS1 H1
PUT THE FEFO POINTER IN THE DATA FIELD OF THE RECORD
IS IT IN THE TIME QUEUE
AS-5 A5
DELAYQ
EXECUTE TIME DELAY
NEED TO GO TO IEDQFQ
NEEDCPB
SET THE FIRST FEFO POINTER
PUT THE SCHEDULER SUBROUTINE ADDRESS IN REGISTER 15
RETURN
INITIALIZE THE REGISTERS AND THE IEDQFQ ENTRY POINT
AS2 K5
BALR 14,15
EXECUTE THE SCHEDULER SUBROUTINE
FQCALL
EXIT TO IEDQFQ
EXIT TO DSPDISP
ARE THERE MORE BUFFERS ON THE QUEUE
C1
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart AS-3 HOLD/RELEASE TERMINAL ROUTINE

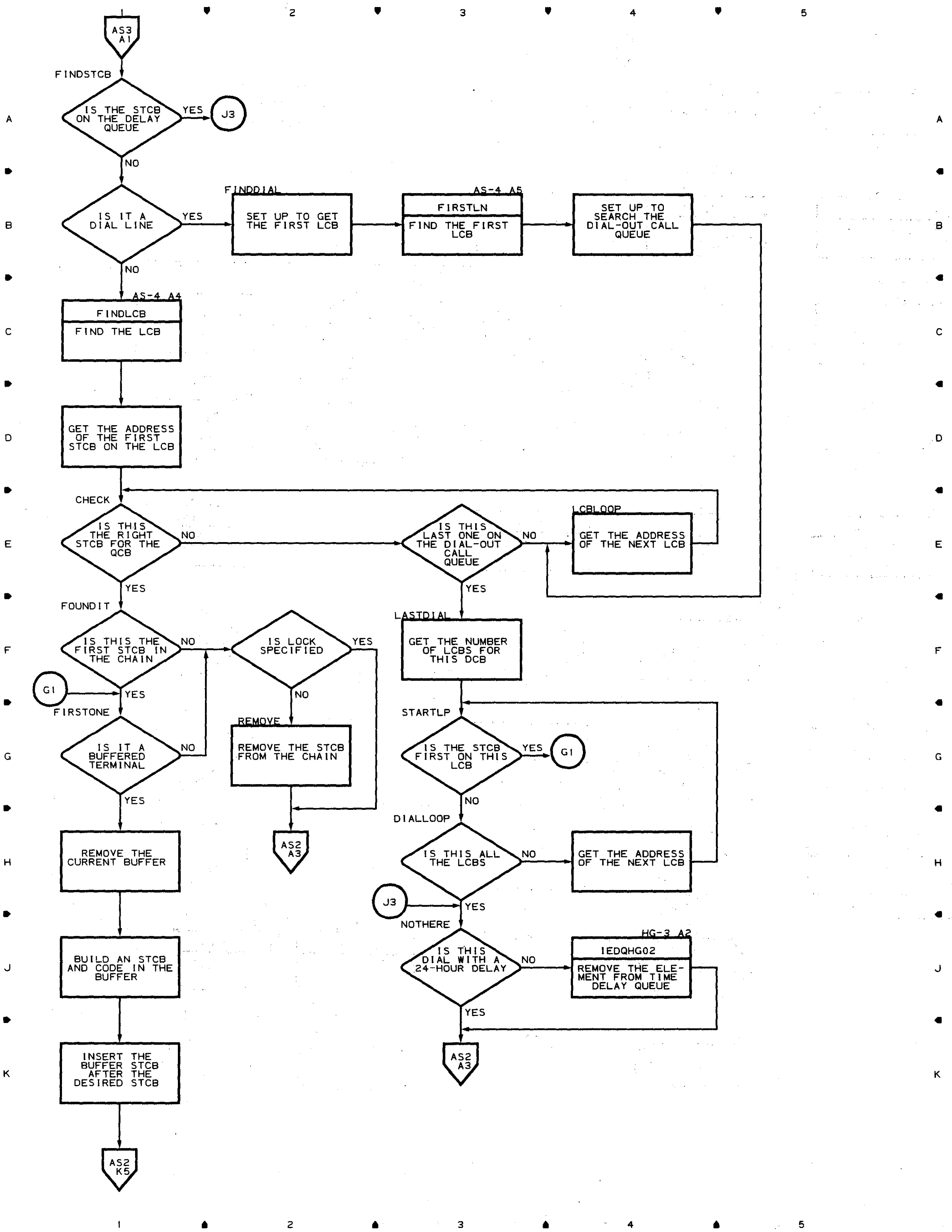
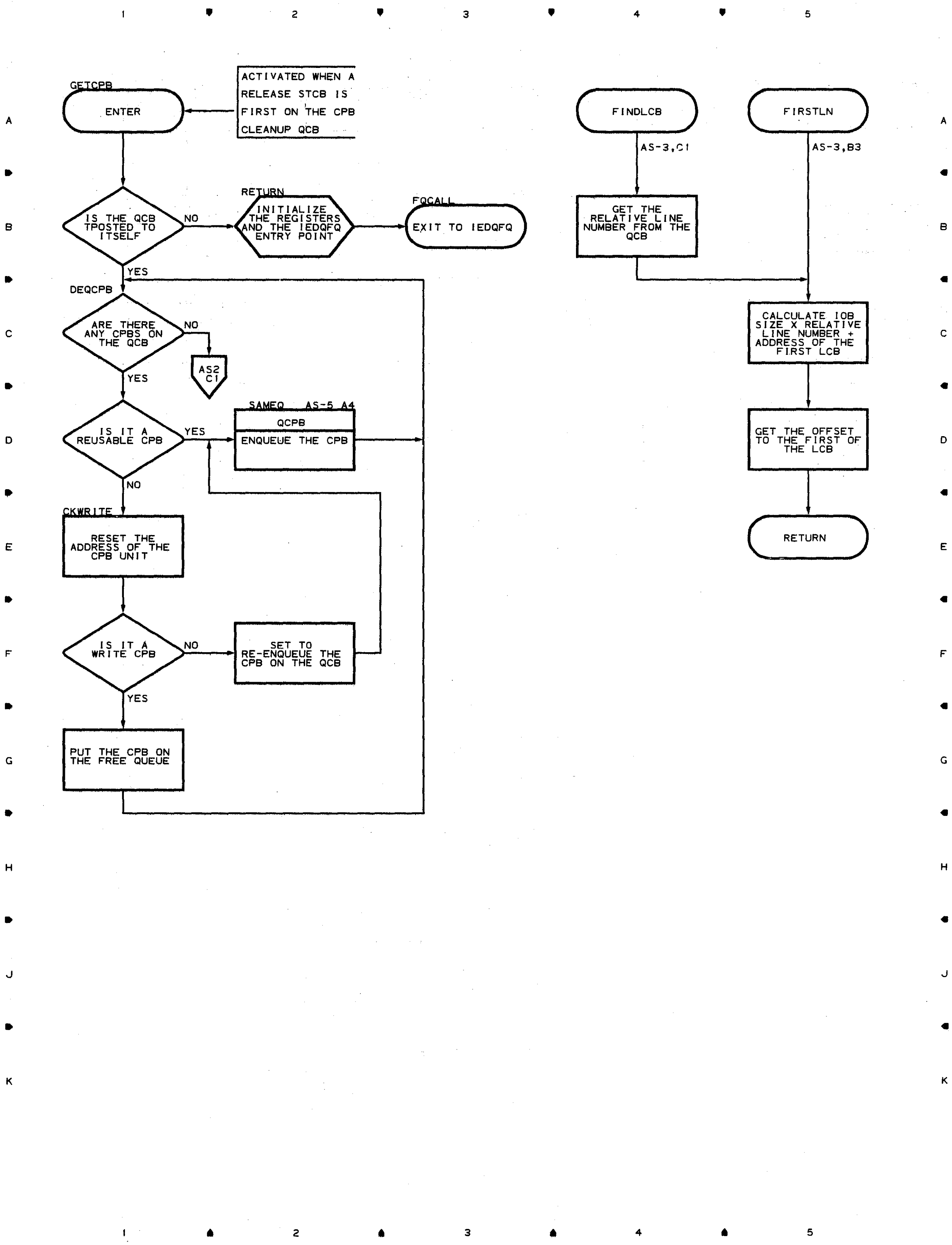

AS3 A1
FINDSTCB
IS THE STCB ON THE DELAY QUEUE
YES
J3
NO
IS IT A DIAL LINE
YES
FINDDIAL
SET UP TO GET THE FIRST LCB
AS-4 A5
FIRSTLN
FIND THE FIRST LCB
SET UP TO SEARCH THE DIAL-OUT CALL QUEUE
NO
AS-4 A4
FINDLCB
FIND THE LCB
GET THE ADDRESS OF THE FIRST STCB ON THE LCB
CHECK
IS THIS THE RIGHT STCB FOR THE QCB
NO
IS THIS LAST ONE ON THE DIAL-OUT CALL QUEUE
NO
LCBLOOP
GET THE ADDRESS OF THE NEXT LCB
YES
FOUNDIT
IS THIS THE FIRST STCB IN THE CHAIN
NO
IS LOCK SPECIFIED
YES
YES
LASTDIAL
GET THE NUMBER OF LCBS FOR THIS DCB
G1
YES
FIRSTONE
NO
REMOVE
REMOVE THE STCB FROM THE CHAIN
STARTLP
IS IT A BUFFERED TERMINAL
NO
IS THE STCB FIRST ON THIS LCB
YES
G1
YES
NO
DIALLOOP
REMOVE THE CURRENT BUFFER
AS2 A3
IS THIS ALL THE LCBS
NO
GET THE ADDRESS OF THE NEXT LCB
J3
YES
NOTHERE
BUILD AN STCB AND CODE IN THE BUFFER
IS THIS DIAL WITH A 24-HOUR DELAY
NO
HG-3 A2
IEDQHG02
REMOVE THE ELEMENT FROM TIME DELAY QUEUE
YES
INSERT THE BUFFER STCB AFTER THE DESIRED STCB
AS2 A3
AS2 K5
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart AS-4 HOLD/RELEASE TERMINAL ROUTINE



GETCPB
ENTER
ACTIVATED WHEN A RELEASE STCB IS FIRST ON THE CPB CLEANUP QCB
FINDLCB
FIRSTLN
AS-3,C1
AS-3,B3
IS THE QCB TPOSTED TO ITSELF
NO
RETURN
INITIALIZE THE REGISTERS AND THE IEDQFQ ENTRY POINT
FQCALL
EXIT TO IEDQFQ
GET THE RELATIVE LINE NUMBER FROM THE QCB
YES
DEQCPB
ARE THERE ANY CPBS ON THE QCB
NO
AS2 C1
YES
CALCULATE IOB SIZE X RELATIVE LINE NUMBER + ADDRESS OF THE FIRST LCB
SAMEQ AS-5 A4
QCPB
IS IT A REUSABLE CPB
YES
ENQUEUE THE CPB
GET THE OFFSET TO THE FIRST OF THE LCB
NO
CKWRITE
RESET THE ADDRESS OF THE CPB UNIT
RETURN
IS IT A WRITE CPB
NO
SET TO RE-ENQUEUE THE CPB ON THE QCB
YES
PUT THE CPB ON THE FREE QUEUE
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart AS-5 HOLD/RELEASE TERMINAL ROUTINE

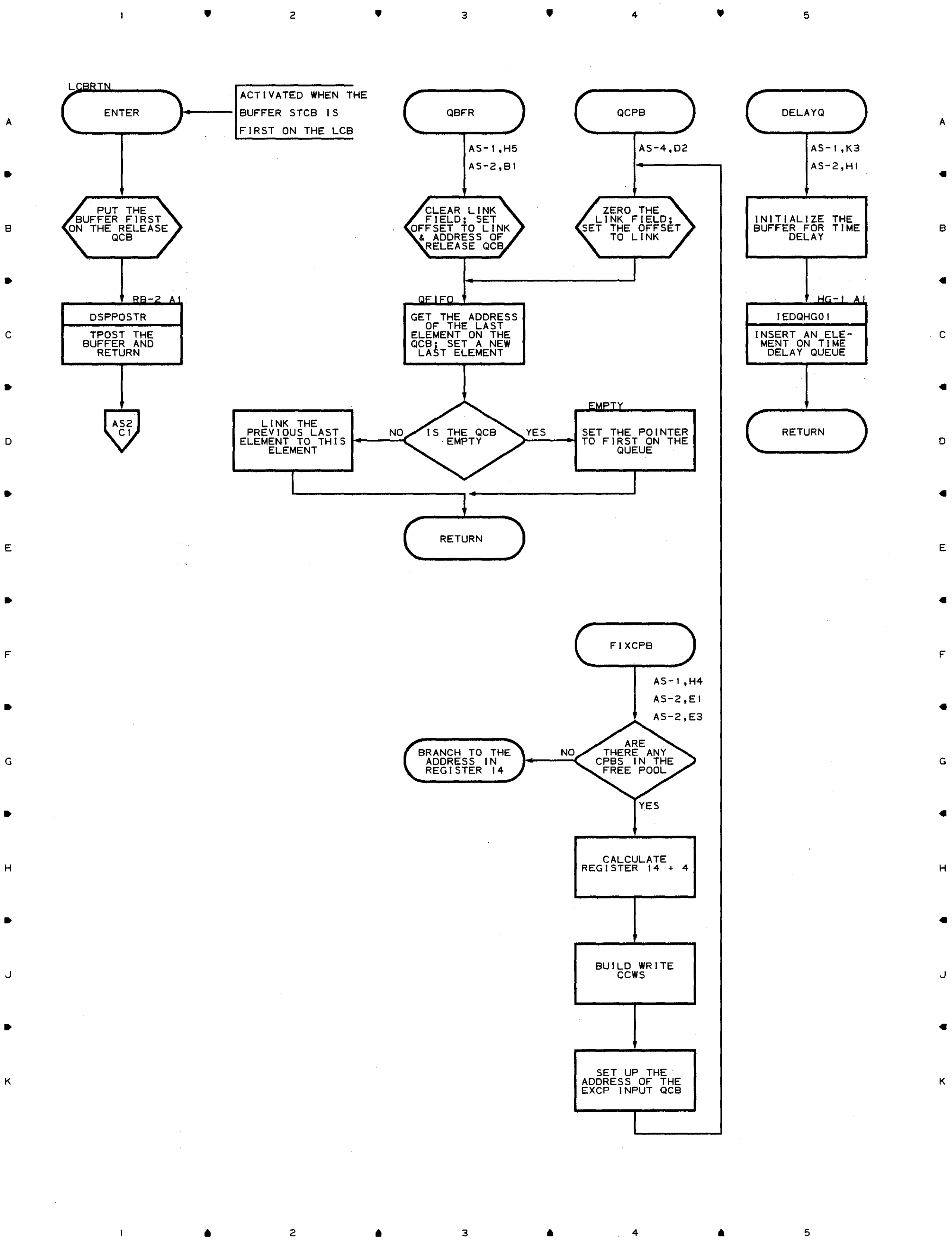
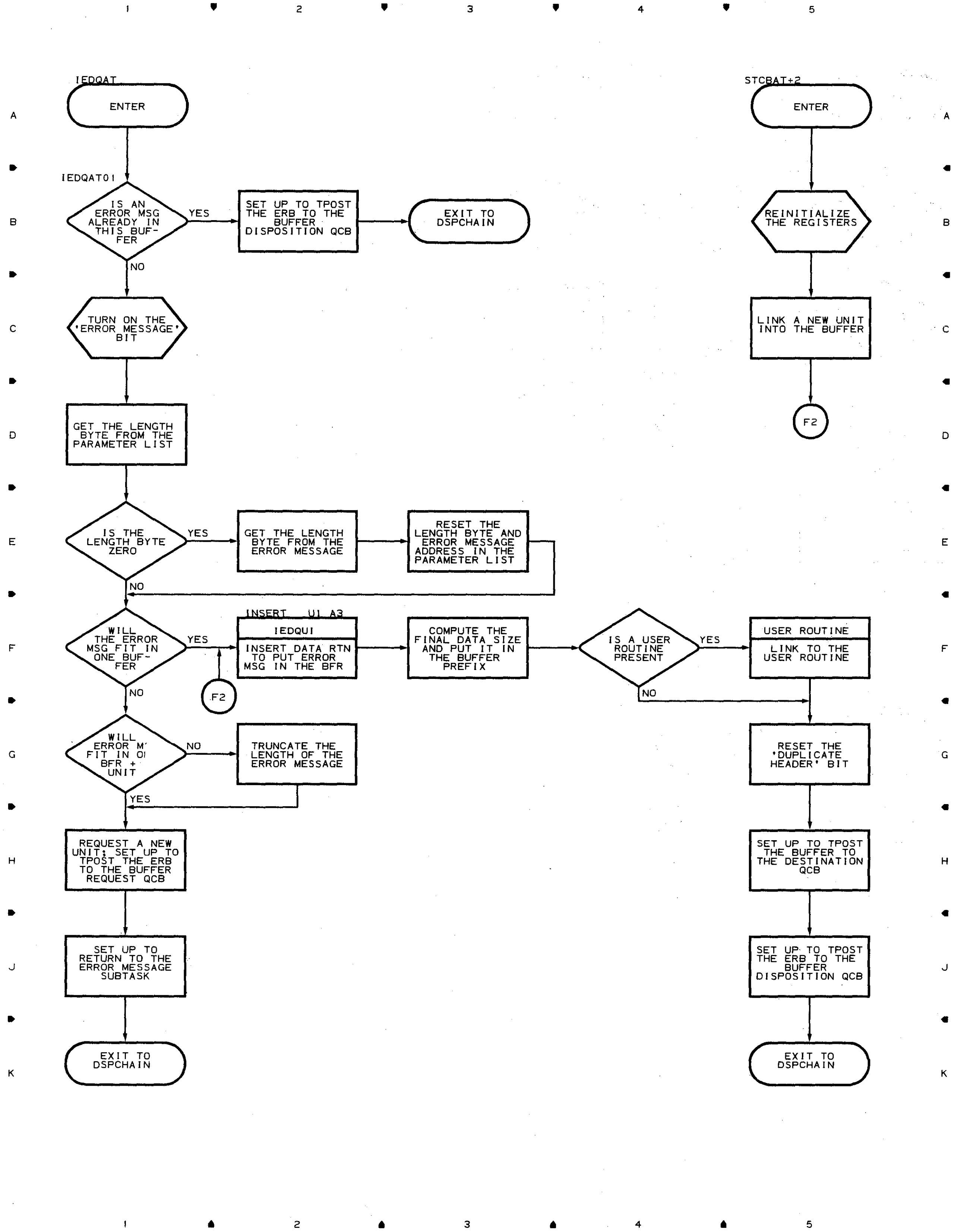

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
LCBRTN
ENTER
ACTIVATED WHEN THE BUFFER STCB IS FIRST ON THE LCB
PUT THE BUFFER FIRST ON THE RELEASE QCB
RB-2 A1
DSPPOSTR
TPOST THE BUFFER AND RETURN
AS2 C1
QBFR
AS-1,H5
AS-2,B1
CLEAR LINK FIELD; SET OFFSET TO LINK & ADDRESS OF RELEASE QCB
QFIFO
GET THE ADDRESS OF THE LAST ELEMENT ON THE QCB; SET A NEW LAST ELEMENT
LINK THE PREVIOUS LAST ELEMENT TO THIS ELEMENT
NO
IS THE QCB EMPTY
YES
EMPTY
SET THE POINTER TO FIRST ON THE QUEUE
RETURN
QCPB
AS-4,D2
ZERO THE LINK FIELD; SET THE OFFSET TO LINK
DELAYQ
AS-1,K3
AS-2,H1
INITIALIZE THE BUFFER FOR TIME DELAY
HG-1 A1
IEDQHG01
INSERT AN ELEMENT ON TIME DELAY QUEUE
RETURN
FIXCPB
AS-1,H4
AS-2,E1
AS-2,E3
ARE THERE ANY CPBS IN THE FREE POOL
NO
BRANCH TO THE ADDRESS IN REGISTER 14
YES
CALCULATE REGISTER 14 + 4
BUILD WRITE CCWS
SET UP THE ADDRESS OF THE EXCP INPUT QCB

Chart AT CREATE AN ERROR MESSAGE ROUTINE AND SUBTASK



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQAT
ENTER
IEDQAT01
IS AN ERROR MSG ALREADY IN THIS BUFFER
YES
SET UP TO TPOST THE ERB TO THE BUFFER DISPOSITION QCB
EXIT TO DSPCHAIN
NO
TURN ON THE 'ERROR MESSAGE' BIT
GET THE LENGTH BYTE FROM THE PARAMETER LIST
IS THE LENGTH BYTE ZERO
YES
GET THE LENGTH BYTE FROM THE ERROR MESSAGE
RESET THE LENGTH BYTE AND ERROR MESSAGE ADDRESS IN THE PARAMETER LIST
NO
WILL THE ERROR MSG FIT IN ONE BUFFER
YES
INSERT U1 A3
IEDQUI
INSERT DATA RTN TO PUT ERROR MSG IN THE BFR
COMPUTE THE FINAL DATA SIZE AND PUT IT IN THE BUFFER PREFIX
IS A USER ROUTINE PRESENT
YES
USER ROUTINE
LINK TO THE USER ROUTINE
NO
NO
F2
WILL ERROR M' FIT IN O! BFR + UNIT
NO
TRUNCATE THE LENGTH OF THE ERROR MESSAGE
YES
REQUEST A NEW UNIT; SET UP TO TPOST THE ERB TO THE BUFFER REQUEST QCB
SET UP TO RETURN TO THE ERROR MESSAGE SUBTASK
EXIT TO DSPCHAIN
STCBAT+2
ENTER
REINITIALIZE THE REGISTERS
LINK A NEW UNIT INTO THE BUFFER
F2
RESET THE 'DUPLICATE HEADER' BIT
SET UP TO TPOST THE BUFFER TO THE DESTINATION QCB
SET UP TO TPOST THE ERB TO THE BUFFER DISPOSITION QCB
EXIT TO DSPCHAIN

Chart AU CUTOFF MESSAGE TRANSMISSION ROUTINE AND SUBTASK

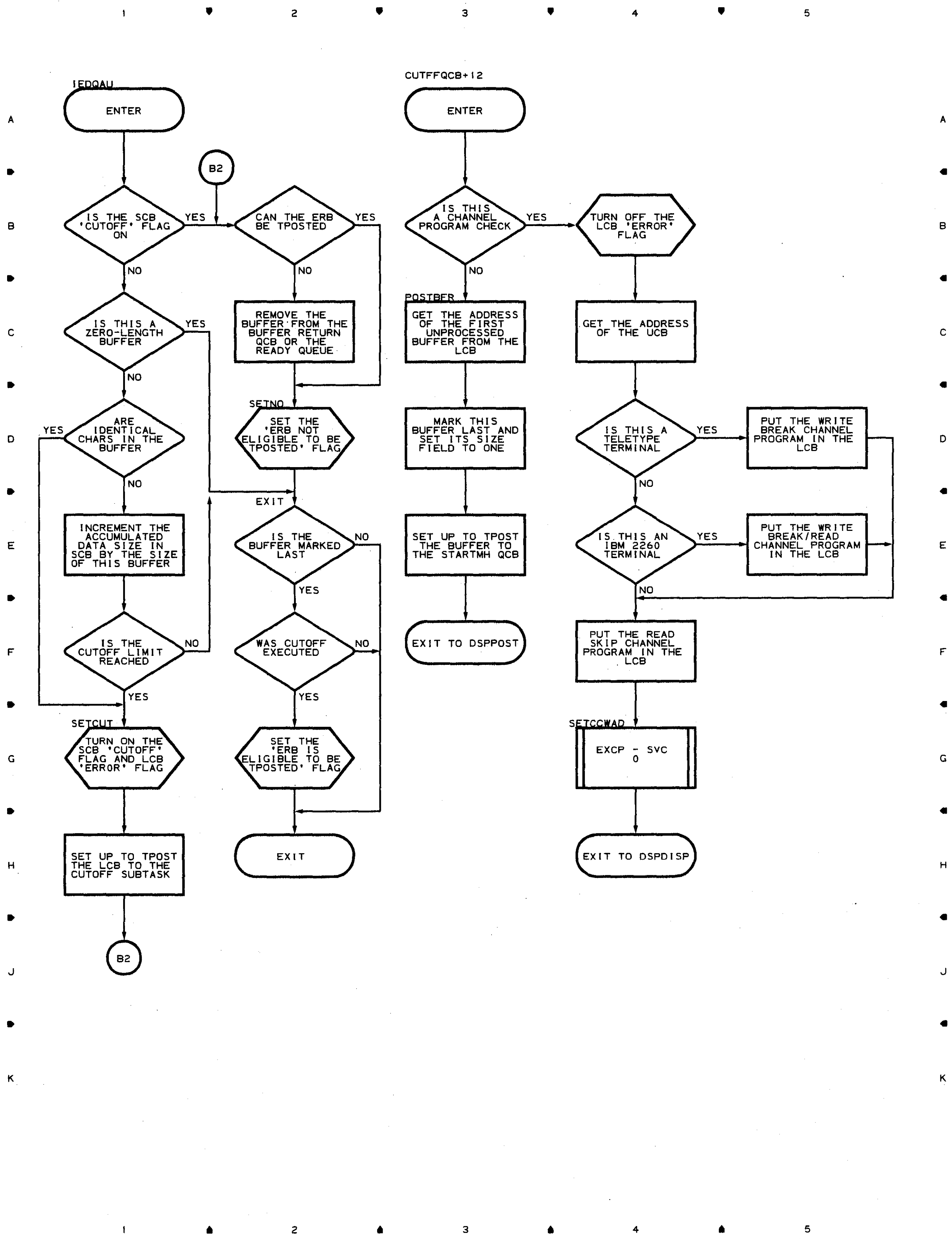

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQAU
ENTER
B2
IS THE SCB 'CUTOFF' FLAG ON
YES
NO
IS THIS A ZERO-LENGTH BUFFER
YES
NO
ARE IDENTICAL CHARS IN THE BUFFER
YES
NO
INCREMENT THE ACCUMULATED DATA SIZE IN SCB BY THE SIZE OF THIS BUFFER
IS THE CUTOFF LIMIT REACHED
NO
YES
SETCUT
TURN ON THE SCB 'CUTOFF' FLAG AND LCB 'ERROR' FLAG
SET UP TO TPOST THE LCB TO THE CUTOFF SUBTASK
B2
CAN THE ERB BE TPOSTED
YES
NO
REMOVE THE BUFFER FROM THE BUFFER RETURN QCB OR THE READY QUEUE
SETNO
SET THE 'ERB NOT ELIGIBLE TO BE TPOSTED' FLAG
EXIT
IS THE BUFFER MARKED LAST
NO
YES
WAS CUTOFF EXECUTED
NO
YES
SET THE 'ERB IS ELIGIBLE TO BE TPOSTED' FLAG
EXIT
CUTFFQCB+12
ENTER
IS THIS A CHANNEL PROGRAM CHECK
YES
NO
POSTBFR
GET THE ADDRESS OF THE FIRST UNPROCESSED BUFFER FROM THE LCB
MARK THIS BUFFER LAST AND SET ITS SIZE FIELD TO ONE
SET UP TO TPOST THE BUFFER TO THE STARTMH QCB
EXIT TO DSPPOST
TURN OFF THE LCB 'ERROR' FLAG
GET THE ADDRESS OF THE UCB
IS THIS A TELETYPE TERMINAL
YES
NO
PUT THE WRITE BREAK CHANNEL PROGRAM IN THE LCB
IS THIS AN IBM 2260 TERMINAL
YES
NO
PUT THE WRITE BREAK/READ CHANNEL PROGRAM IN THE LCB
PUT THE READ SKIP CHANNEL PROGRAM IN THE LCB
SETCCWAD
EXCP - SVC 0
EXIT TO DSPDISP

Chart AV LOOKUP TERMINAL ENTRY ROUTINE

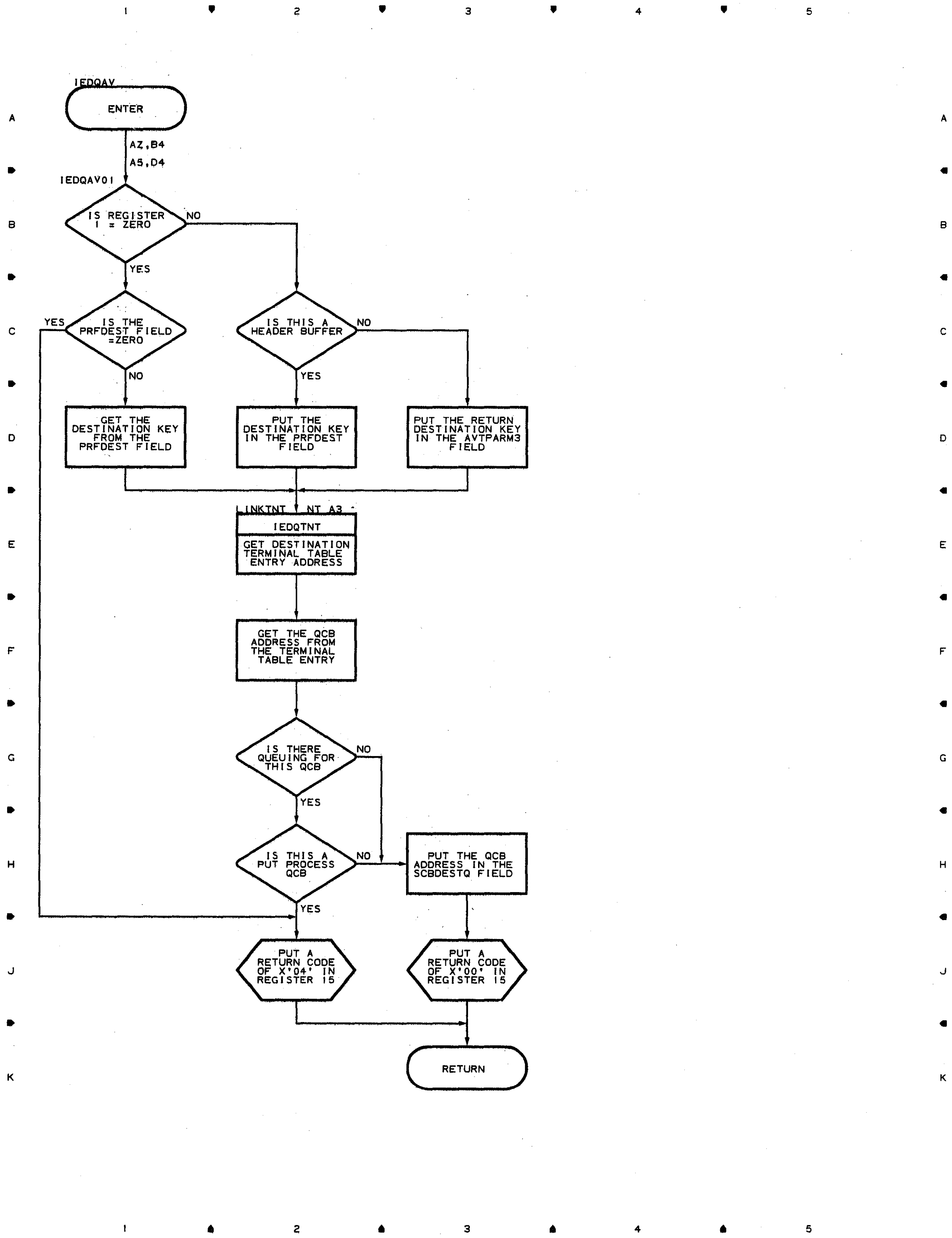
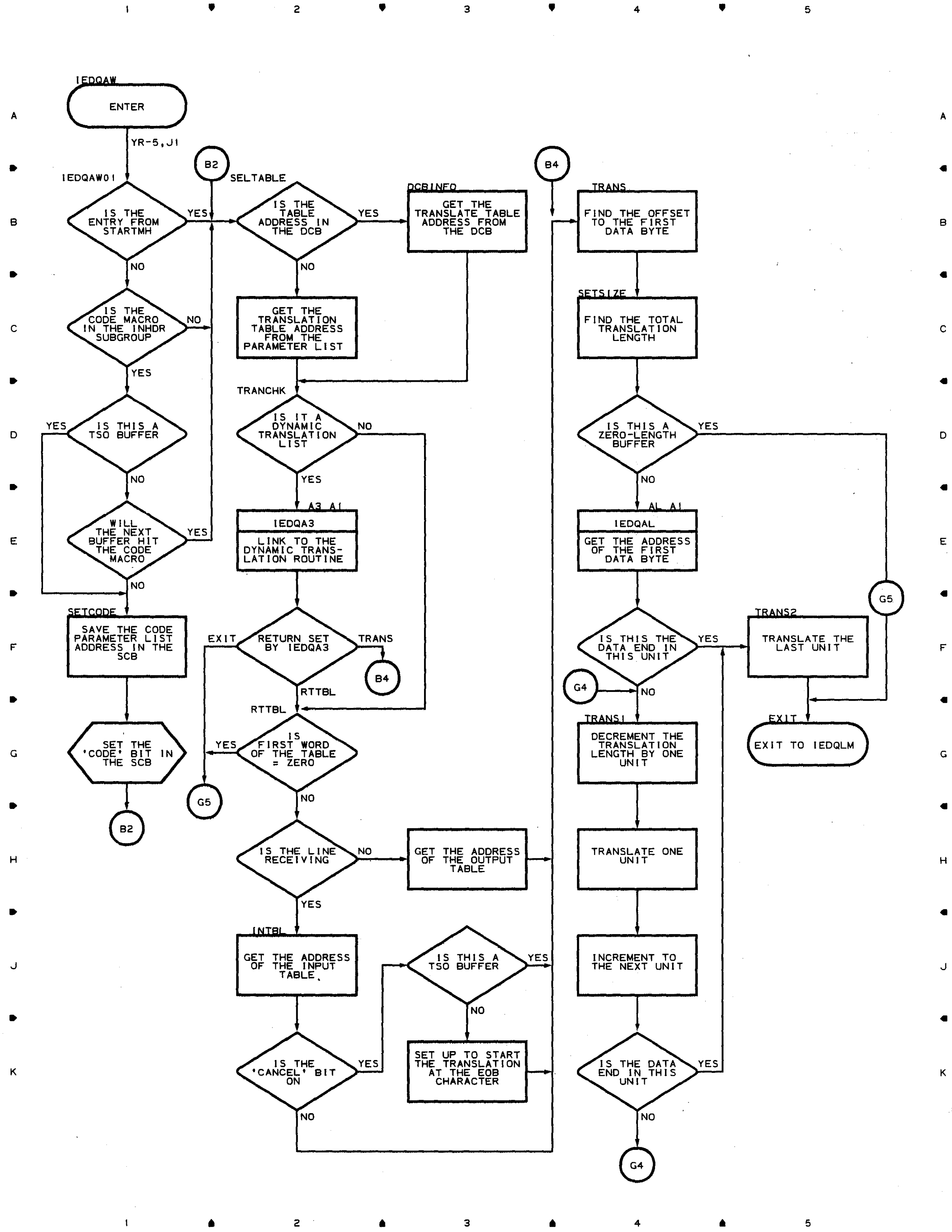
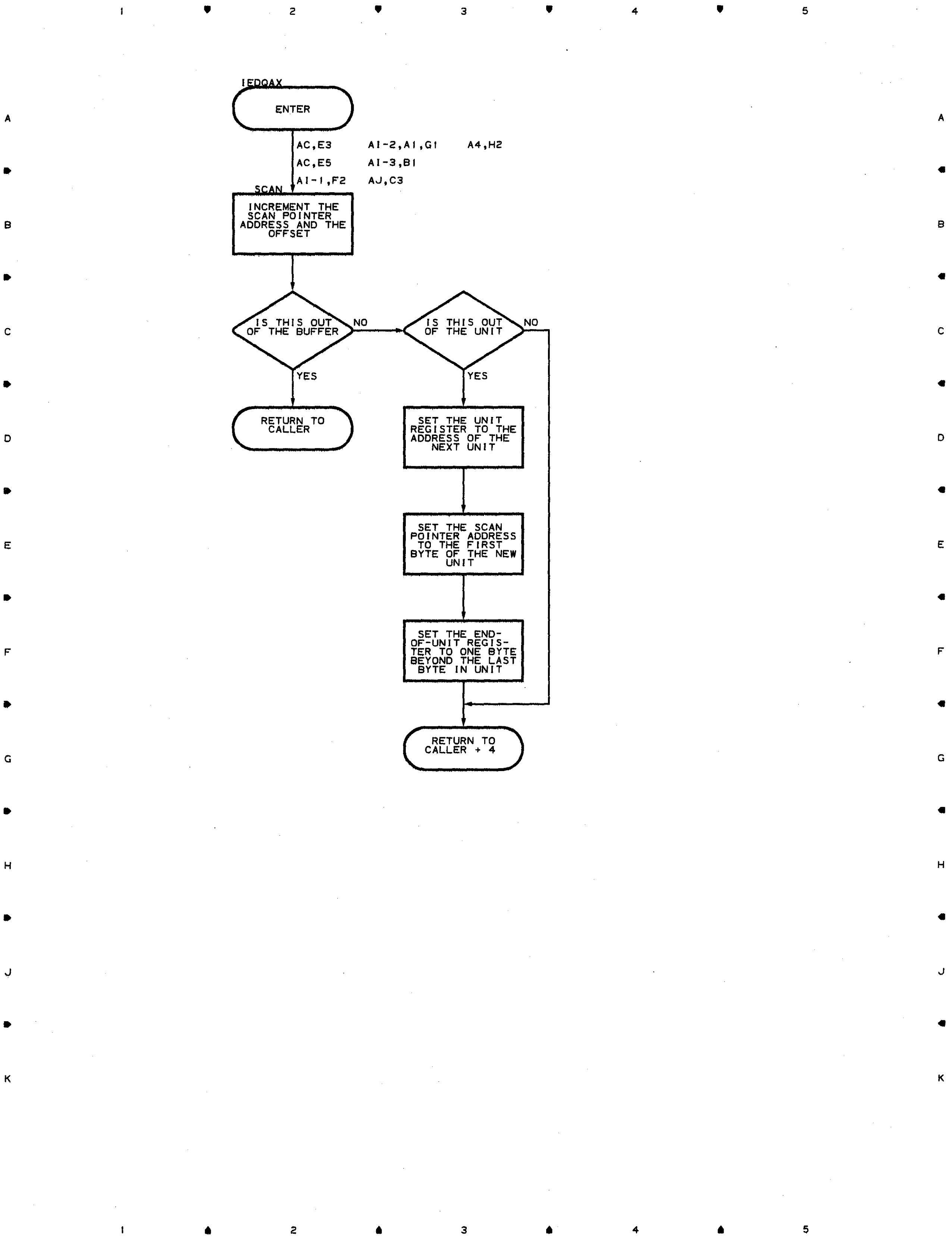

IEDQAV
ENTER
AZ,B4
A5,D4
IEDQAV01
IS REGISTER 1 = ZERO
NO
YES
IS THE PRFDEST FIELD =ZERO
YES
NO
IS THIS A HEADER BUFFER
NO
YES
GET THE DESTINATION KEY FROM THE PRFDEST FIELD
PUT THE DESTINATION KEY IN THE PRFDEST FIELD
PUT THE RETURN DESTINATION KEY IN THE AVTPARM3 FIELD
LINKTNT
NT A3
IEDQTNT
GET DESTINATION TERMINAL TABLE ENTRY ADDRESS
GET THE QCB ADDRESS FROM THE TERMINAL TABLE ENTRY
IS THERE QUEUING FOR THIS QCB
NO
YES
IS THIS A PUT PROCESS QCB
NO
YES
PUT THE QCB ADDRESS IN THE SCBDESTQ FIELD
PUT A RETURN CODE OF X'04' IN REGISTER 15
PUT A RETURN CODE OF X'00' IN REGISTER 15
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart AW TRANSLATE BUFFER ROUTINE



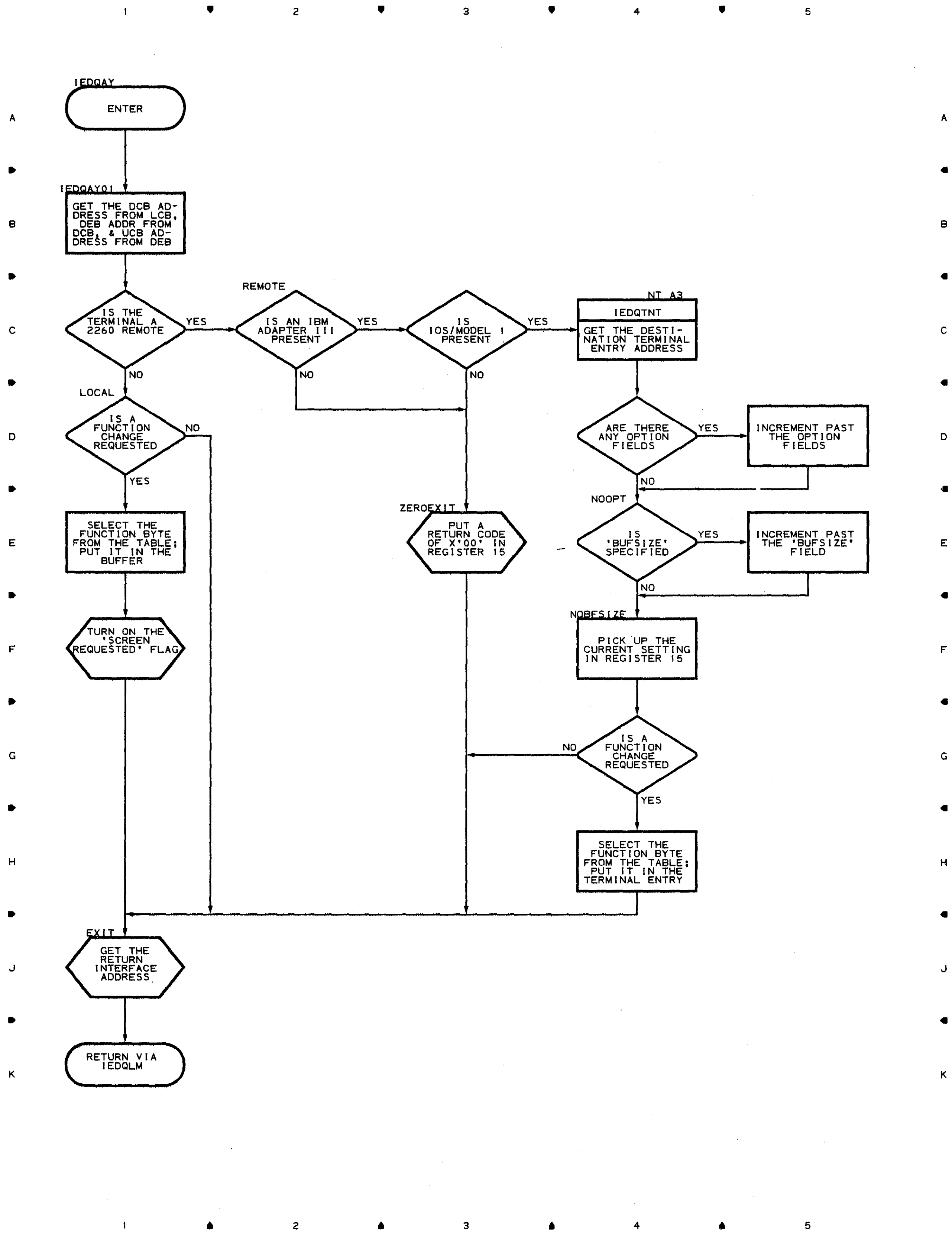
IEDQAW
ENTER
YR-5,J1
IEDQAW01
IS THE ENTRY FROM STARTMH
YES
NO
IS THE CODE MACRO IN THE INHDR SUBGROUP
NO
YES
IS THIS A TSO BUFFER
YES
NO
WILL THE NEXT BUFFER HIT THE CODE MACRO
YES
NO
SETCODE
SAVE THE CODE PARAMETER LIST ADDRESS IN THE SCB
SET THE 'CODE' BIT IN THE SCB
B2
B2
SELTABLE
IS THE TABLE ADDRESS IN THE DCB
YES
NO
DCBINFO
GET THE TRANSLATE TABLE ADDRESS FROM THE DCB
GET THE TRANSLATION TABLE ADDRESS FROM THE PARAMETER LIST
TRANCHK
IS IT A DYNAMIC TRANSLATION LIST
NO
YES
A3 A1
IEDQA3
LINK TO THE DYNAMIC TRANSLATION ROUTINE
EXIT
RETURN SET BY IEDQA3
TRANS
B4
RTTBL
RTTBL
IS FIRST WORD OF THE TABLE = ZERO
YES
G5
NO
IS THE LINE RECEIVING
NO
GET THE ADDRESS OF THE OUTPUT TABLE
YES
INTBL
GET THE ADDRESS OF THE INPUT TABLE
IS THE 'CANCEL' BIT ON
YES
NO
IS THIS A TSO BUFFER
YES
NO
SET UP TO START THE TRANSLATION AT THE EOB CHARACTER
B4
TRANS
FIND THE OFFSET TO THE FIRST DATA BYTE
SETSIZE
FIND THE TOTAL TRANSLATION LENGTH
IS THIS A ZERO-LENGTH BUFFER
YES
NO
AL A1
IEDQAL
GET THE ADDRESS OF THE FIRST DATA BYTE
IS THIS THE DATA END IN THIS UNIT
YES
G4
NO
TRANS1
DECREMENT THE TRANSLATION LENGTH BY ONE UNIT
TRANSLATE ONE UNIT
INCREMENT TO THE NEXT UNIT
IS THE DATA END IN THIS UNIT
YES
NO
G4
G5
TRANS2
TRANSLATE THE LAST UNIT
EXIT
EXIT TO IEDQLM
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart AX BUFFER STEP ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQAX
ENTER
AC,E3
AC,E5
AI-1,F2
AI-2,AI,GI
AI-3,BI
AJ,C3
A4,H2
SCAN
INCREMENT THE SCAN POINTER ADDRESS AND THE OFFSET
IS THIS OUT OF THE BUFFER
NO
YES
RETURN TO CALLER
IS THIS OUT OF THE UNIT
NO
YES
SET THE UNIT REGISTER TO THE ADDRESS OF THE NEXT UNIT
SET THE SCAN POINTER ADDRESS TO THE FIRST BYTE OF THE NEW UNIT
SET THE END-OF-UNIT REGISTER TO ONE BYTE BEYOND THE LAST BYTE IN UNIT
RETURN TO CALLER + 4

Chart AY SCREEN ROUTINE



IEDQAY
ENTER
IEDQAY01
GET THE DCB ADDRESS FROM LCB, DEB ADDR FROM DCB, & UCB ADDRESS FROM DEB
IS THE TERMINAL A 2260 REMOTE
YES
NO
REMOTE
IS AN IBM ADAPTER III PRESENT
YES
NO
IS IOS/MODEL 1 PRESENT
YES
NO
NT A3
IEDQTNT
GET THE DESTINATION TERMINAL ENTRY ADDRESS
LOCAL
IS A FUNCTION CHANGE REQUESTED
NO
YES
ARE THERE ANY OPTION FIELDS
YES
INCREMENT PAST THE OPTION FIELDS
NO
NOOPT
SELECT THE FUNCTION BYTE FROM THE TABLE; PUT IT IN THE BUFFER
ZEROEXIT
PUT A RETURN CODE OF X'00' IN REGISTER 15
IS 'BUFSIZE' SPECIFIED
YES
INCREMENT PAST THE 'BUFSIZE' FIELD
NO
NOBFSIZE
TURN ON THE 'SCREEN REQUESTED' FLAG
PICK UP THE CURRENT SETTING IN REGISTER 15
NO
IS A FUNCTION CHANGE REQUESTED
YES
SELECT THE FUNCTION BYTE FROM THE TABLE; PUT IT IN THE TERMINAL ENTRY
EXIT
GET THE RETURN INTERFACE ADDRESS
RETURN VIA IEDQLM
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart AZ REDIRECT A MESSAGE ROUTINE

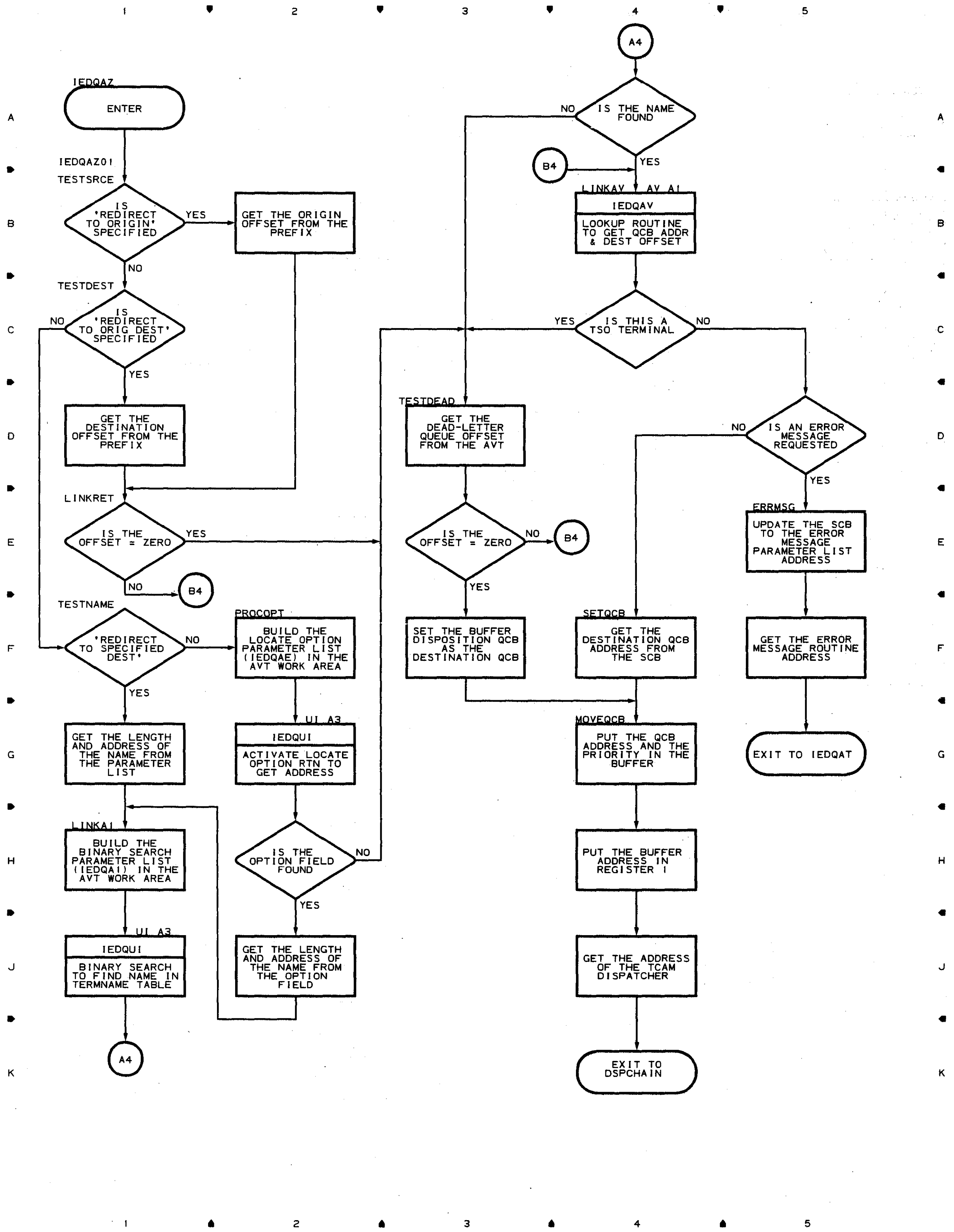

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQAZ
ENTER
IEDQAZ01
TESTSRCE
IS 'REDIRECT TO ORIGIN' SPECIFIED
YES
GET THE ORIGIN OFFSET FROM THE PREFIX
NO
TESTDEST
IS 'REDIRECT TO ORIG DEST' SPECIFIED
NO
YES
GET THE DESTINATION OFFSET FROM THE PREFIX
LINKRET
IS THE OFFSET = ZERO
YES
NO
B4
TESTNAME
'REDIRECT TO SPECIFIED DEST'
NO
YES
PROCOPT
BUILD THE LOCATE OPTION PARAMETER LIST (IEDQAE) IN THE AVT WORK AREA
UI A3
IEDQUI
ACTIVATE LOCATE OPTION RTN TO GET ADDRESS
IS THE OPTION FIELD FOUND
NO
YES
GET THE LENGTH AND ADDRESS OF THE NAME FROM THE OPTION FIELD
GET THE LENGTH AND ADDRESS OF THE NAME FROM THE PARAMETER LIST
LINKAI
BUILD THE BINARY SEARCH PARAMETER LIST (IEDQAI) IN THE AVT WORK AREA
UI A3
IEDQUI
BINARY SEARCH TO FIND NAME IN TERMNAME TABLE
A4
TESTDEAD
GET THE DEAD-LETTER QUEUE OFFSET FROM THE AVT
IS THE OFFSET = ZERO
NO
B4
YES
SET THE BUFFER DISPOSITION QCB AS THE DESTINATION QCB
A4
IS THE NAME FOUND
NO
YES
B4
LINKAV
AV A1
IEDQAV
LOOKUP ROUTINE TO GET QCB ADDR & DEST OFFSET
IS THIS A TSO TERMINAL
YES
NO
IS AN ERROR MESSAGE REQUESTED
NO
YES
ERRMSG
UPDATE THE SCB TO THE ERROR MESSAGE PARAMETER LIST ADDRESS
GET THE ERROR MESSAGE ROUTINE ADDRESS
EXIT TO IEDQAT
SETQCB
GET THE DESTINATION QCB ADDRESS FROM THE SCB
MOVEQCB
PUT THE QCB ADDRESS AND THE PRIORITY IN THE BUFFER
PUT THE BUFFER ADDRESS IN REGISTER 1
GET THE ADDRESS OF THE TCAM DISPATCHER
EXIT TO DSPCHAIN

Chart A0 SKIP BACKWARD ROUTINE

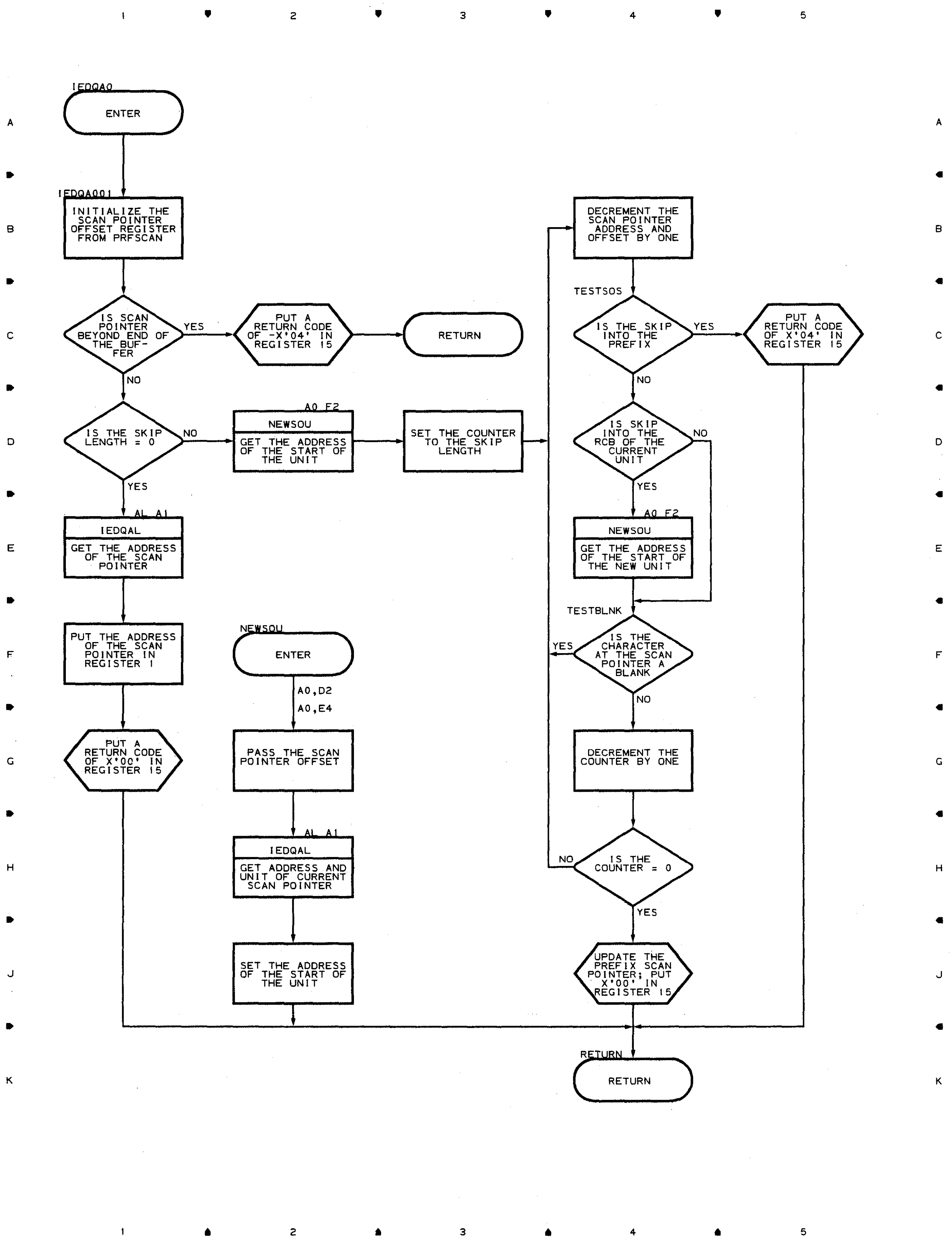
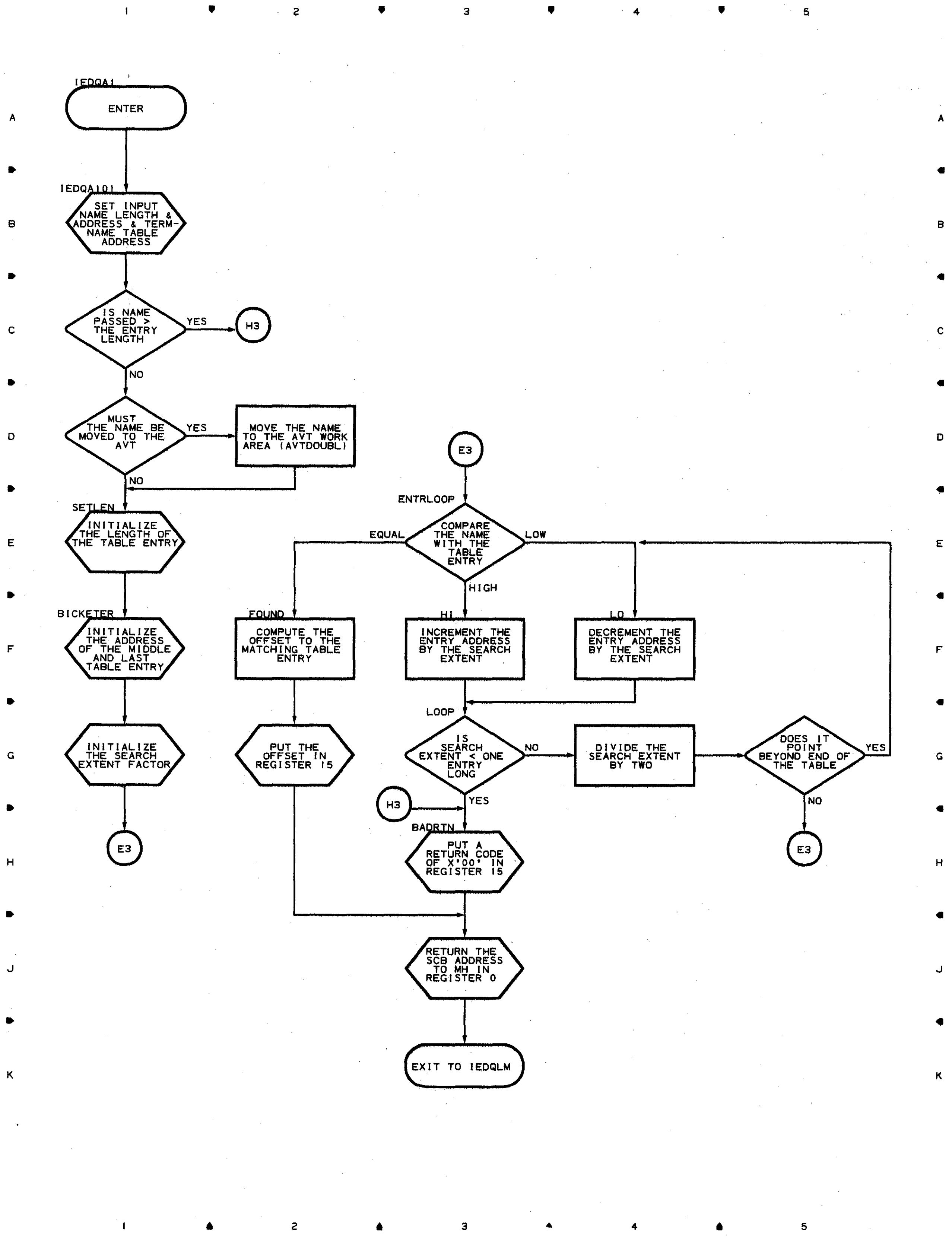

IEDQA0
ENTER
IEDQA001
INITIALIZE THE SCAN POINTER OFFSET REGISTER FROM PRFSCAN
IS SCAN POINTER BEYOND END OF THE BUF-FER
YES
PUT A RETURN CODE OF -X'04' IN REGISTER 15
RETURN
NO
IS THE SKIP LENGTH = 0
NO
A0 F2
NEWSOU
GET THE ADDRESS OF THE START OF THE UNIT
SET THE COUNTER TO THE SKIP LENGTH
YES
AL A1
IEDQAL
GET THE ADDRESS OF THE SCAN POINTER
PUT THE ADDRESS OF THE SCAN POINTER IN REGISTER 1
PUT A RETURN CODE OF X'00' IN REGISTER 15
NEWSOU
ENTER
A0,D2
A0,E4
PASS THE SCAN POINTER OFFSET
AL A1
IEDQAL
GET ADDRESS AND UNIT OF CURRENT SCAN POINTER
SET THE ADDRESS OF THE START OF THE UNIT
DECREMENT THE SCAN POINTER ADDRESS AND OFFSET BY ONE
TESTSOS
IS THE SKIP INTO THE PREFIX
YES
PUT A RETURN CODE OF X'04' IN REGISTER 15
NO
IS SKIP INTO THE RCB OF THE CURRENT UNIT
NO
YES
A0 F2
NEWSOU
GET THE ADDRESS OF THE START OF THE NEW UNIT
TESTBLNK
IS THE CHARACTER AT THE SCAN POINTER A BLANK
YES
NO
DECREMENT THE COUNTER BY ONE
IS THE COUNTER = 0
NO
YES
UPDATE THE PREFIX SCAN POINTER; PUT X'00' IN REGISTER 15
RETURN
RETURN
1 2 3 4 5
A B C D E F G H J K

Chart A1 BINARY SEARCH ROUTINE



IEDQA1
ENTER
IEDQA101
SET INPUT NAME LENGTH & ADDRESS & TERM-NAME TABLE ADDRESS
IS NAME PASSED > THE ENTRY LENGTH
YES
H3
NO
MUST THE NAME BE MOVED TO THE AVT
YES
MOVE THE NAME TO THE AVT WORK AREA (AVTDOUBL)
NO
SETLEN
INITIALIZE THE LENGTH OF THE TABLE ENTRY
BICKETER
INITIALIZE THE ADDRESS OF THE MIDDLE AND LAST TABLE ENTRY
INITIALIZE THE SEARCH EXTENT FACTOR
E3
E3
ENTRLOOP
COMPARE THE NAME WITH THE TABLE ENTRY
EQUAL
LOW
HIGH
FOUND
COMPUTE THE OFFSET TO THE MATCHING TABLE ENTRY
HI
INCREMENT THE ENTRY ADDRESS BY THE SEARCH EXTENT
LO
DECREMENT THE ENTRY ADDRESS BY THE SEARCH EXTENT
PUT THE OFFSET IN REGISTER 15
LOOP
IS SEARCH EXTENT < ONE ENTRY LONG
NO
DIVIDE THE SEARCH EXTENT BY TWO
DOES IT POINT BEYOND END OF THE TABLE
YES
NO
E3
YES
H3
BADRTN
PUT A RETURN CODE OF X'00' IN REGISTER 15
RETURN THE SCB ADDRESS TO MH IN REGISTER 0
EXIT TO IEDQLM
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart A2 INSERT AT OFFSET ROUTINE

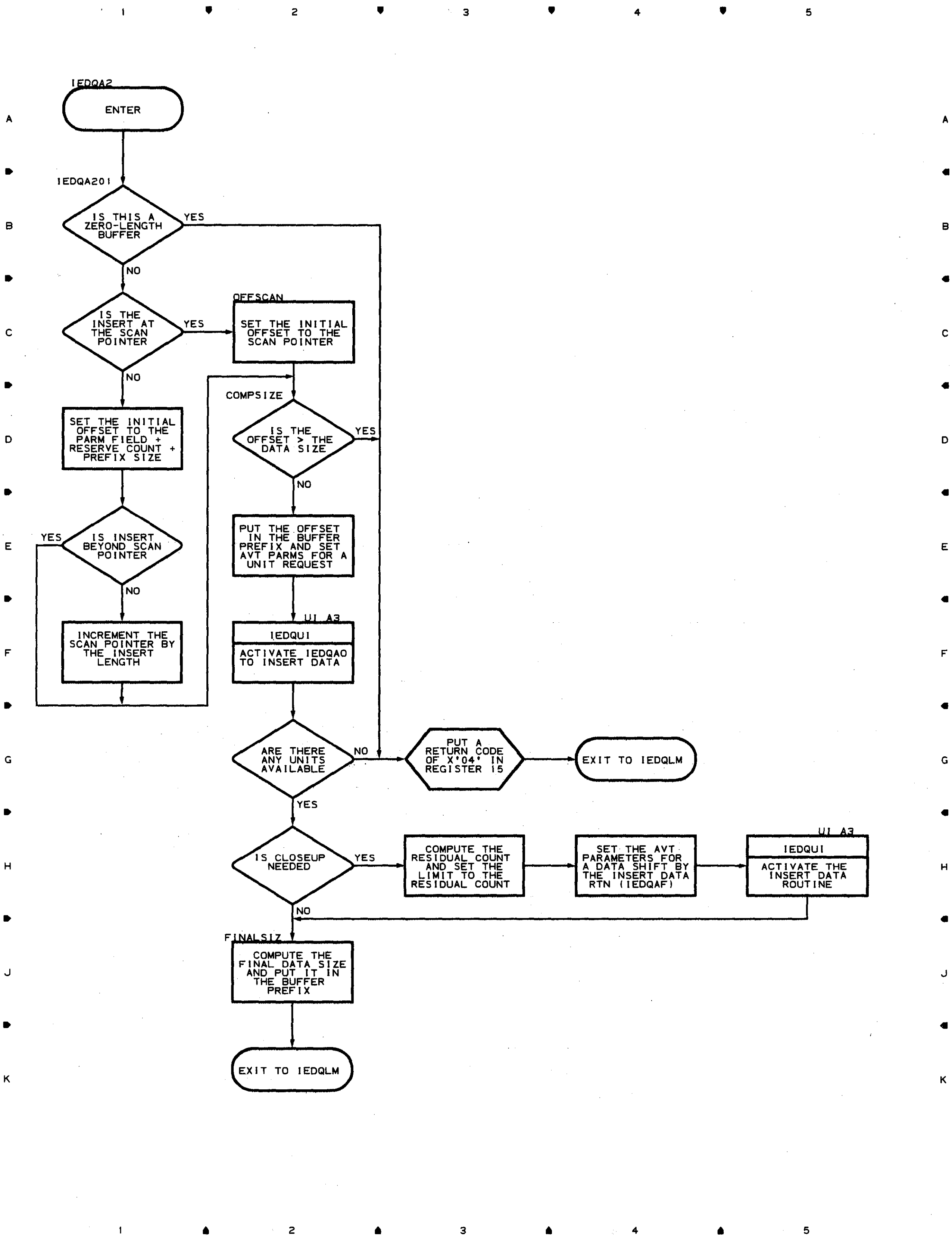

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQA2
ENTER
IEDQA201
IS THIS A ZERO-LENGTH BUFFER
YES
NO
IS THE INSERT AT THE SCAN POINTER
YES
NO
OFFSCAN
SET THE INITIAL OFFSET TO THE SCAN POINTER
SET THE INITIAL OFFSET TO THE PARM FIELD + RESERVE COUNT + PREFIX SIZE
COMPSIZE
IS THE OFFSET > THE DATA SIZE
YES
NO
IS INSERT BEYOND SCAN POINTER
YES
NO
PUT THE OFFSET IN THE BUFFER PREFIX AND SET AVT PARMS FOR A UNIT REQUEST
INCREMENT THE SCAN POINTER BY THE INSERT LENGTH
UI A3
IEDQUI
ACTIVATE IEDQAO TO INSERT DATA
ARE THERE ANY UNITS AVAILABLE
NO
YES
PUT A RETURN CODE OF X'04' IN REGISTER 15
EXIT TO IEDQLM
IS CLOSEUP NEEDED
YES
NO
COMPUTE THE RESIDUAL COUNT AND SET THE LIMIT TO THE RESIDUAL COUNT
SET THE AVT PARAMETERS FOR A DATA SHIFT BY THE INSERT DATA RTN (IEDQAF)
UI A3
IEDQUI
ACTIVATE THE INSERT DATA ROUTINE
FINALSIZ
COMPUTE THE FINAL DATA SIZE AND PUT IT IN THE BUFFER PREFIX
EXIT TO IEDQLM

Chart A3 DYNAMIC TRANSLATION ROUTINE

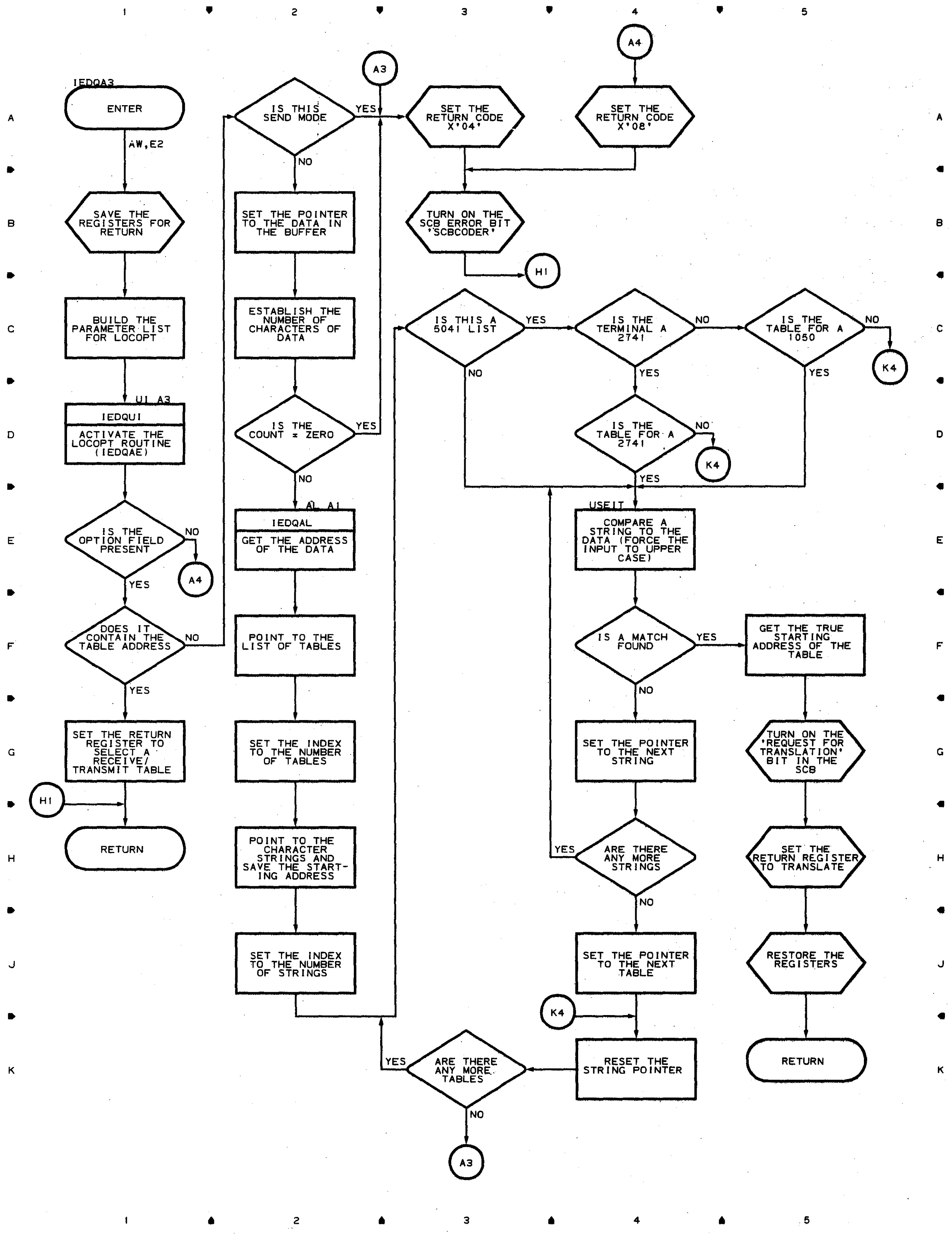

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQA3
ENTER
AW,E2
SAVE THE REGISTERS FOR RETURN
BUILD THE PARAMETER LIST FOR LOCOPT
UI A3
IEDQUI
ACTIVATE THE LOCOPT ROUTINE (IEDQAE)
IS THE OPTION FIELD PRESENT
NO
A4
YES
DOES IT CONTAIN THE TABLE ADDRESS
NO
YES
SET THE RETURN REGISTER TO SELECT A RECEIVE/ TRANSMIT TABLE
H1
RETURN
IS THIS SEND MODE
YES
NO
SET THE POINTER TO THE DATA IN THE BUFFER
ESTABLISH THE NUMBER OF CHARACTERS OF DATA
IS THE COUNT = ZERO
YES
NO
AL A1
IEDQAL
GET THE ADDRESS OF THE DATA
POINT TO THE LIST OF TABLES
SET THE INDEX TO THE NUMBER OF TABLES
POINT TO THE CHARACTER STRINGS AND SAVE THE START-ING ADDRESS
SET THE INDEX TO THE NUMBER OF STRINGS
A3
SET THE RETURN CODE X'04'
TURN ON THE SCB ERROR BIT 'SCBCODER'
H1
IS THIS A 5041 LIST
YES
NO
ARE THERE ANY MORE TABLES
YES
NO
A3
A4
SET THE RETURN CODE X'08'
IS THE TERMINAL A 2741
NO
YES
IS THE TABLE FOR A 2741
NO
K4
YES
USEIT
COMPARE A STRING TO THE DATA (FORCE THE INPUT TO UPPER CASE)
IS A MATCH FOUND
YES
NO
SET THE POINTER TO THE NEXT STRING
ARE THERE ANY MORE STRINGS
YES
NO
SET THE POINTER TO THE NEXT TABLE
K4
RESET THE STRING POINTER
IS THE TABLE FOR A 1050
NO
K4
YES
GET THE TRUE STARTING ADDRESS OF THE TABLE
TURN ON THE 'REQUEST FOR TRANSLATION' BIT IN THE SCB
SET THE RETURN REGISTER TO TRANSLATE
RESTORE THE REGISTERS
RETURN

Chart A4 INCOMING/OUTGOING MESSAGE DELIMITER ROUTINE

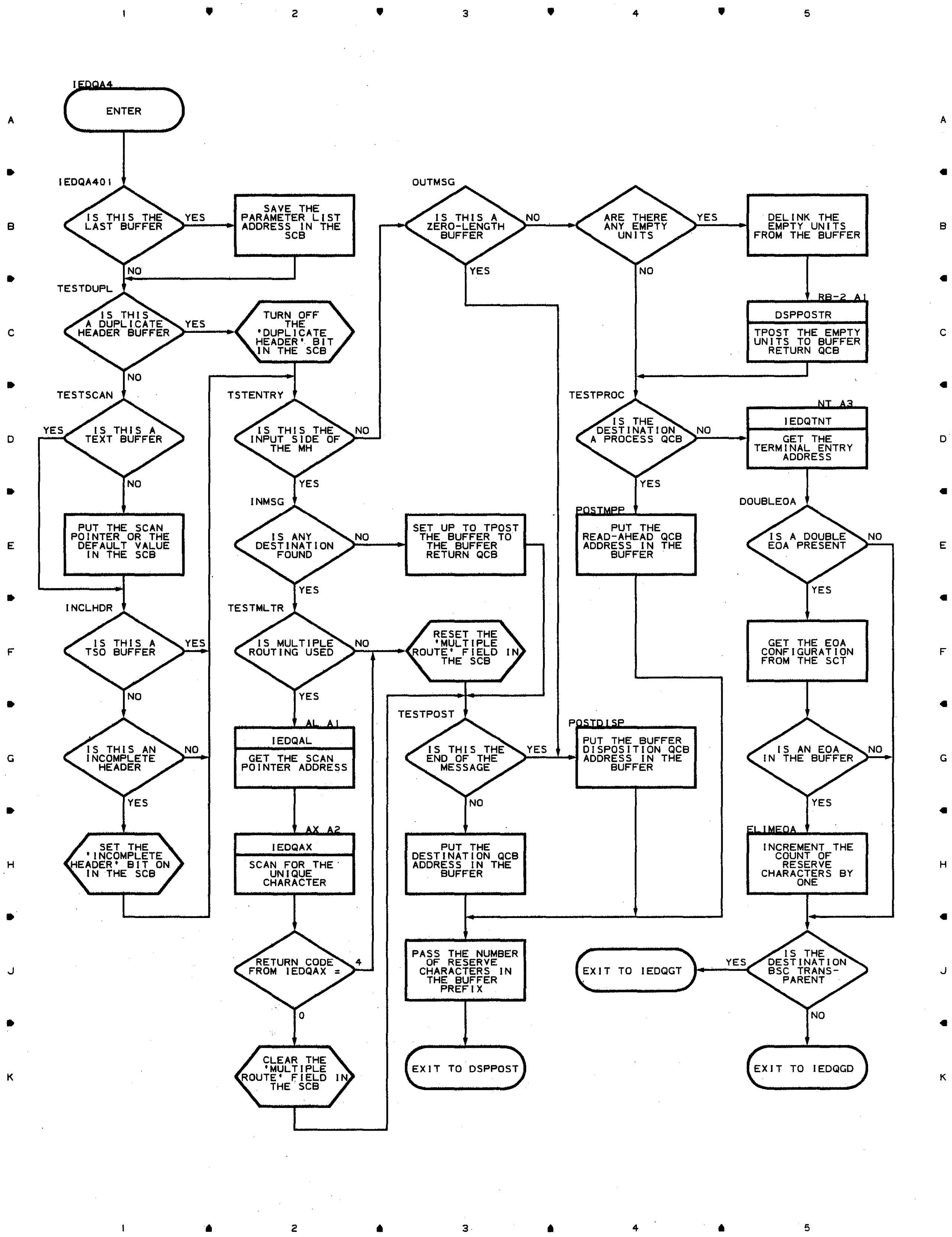
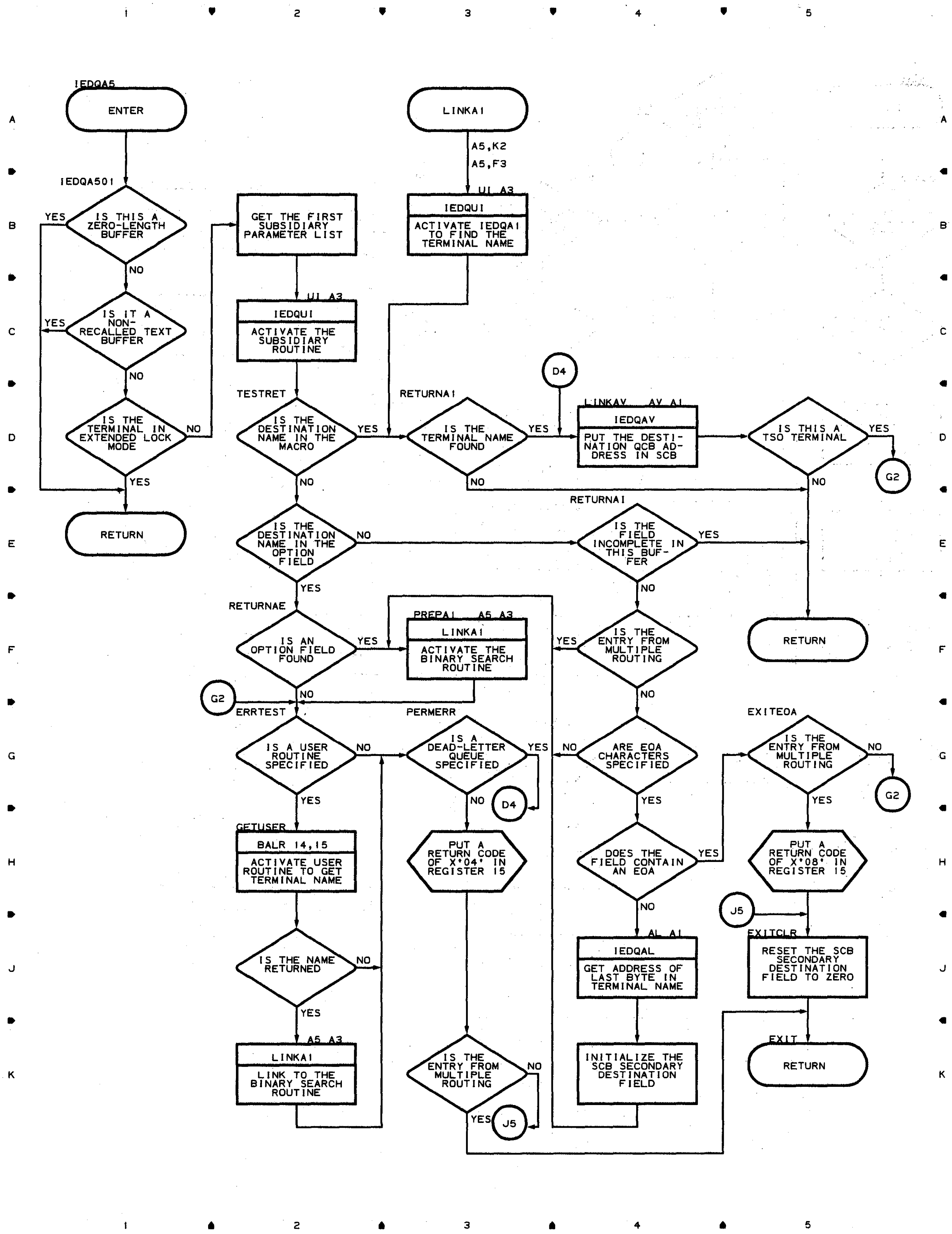
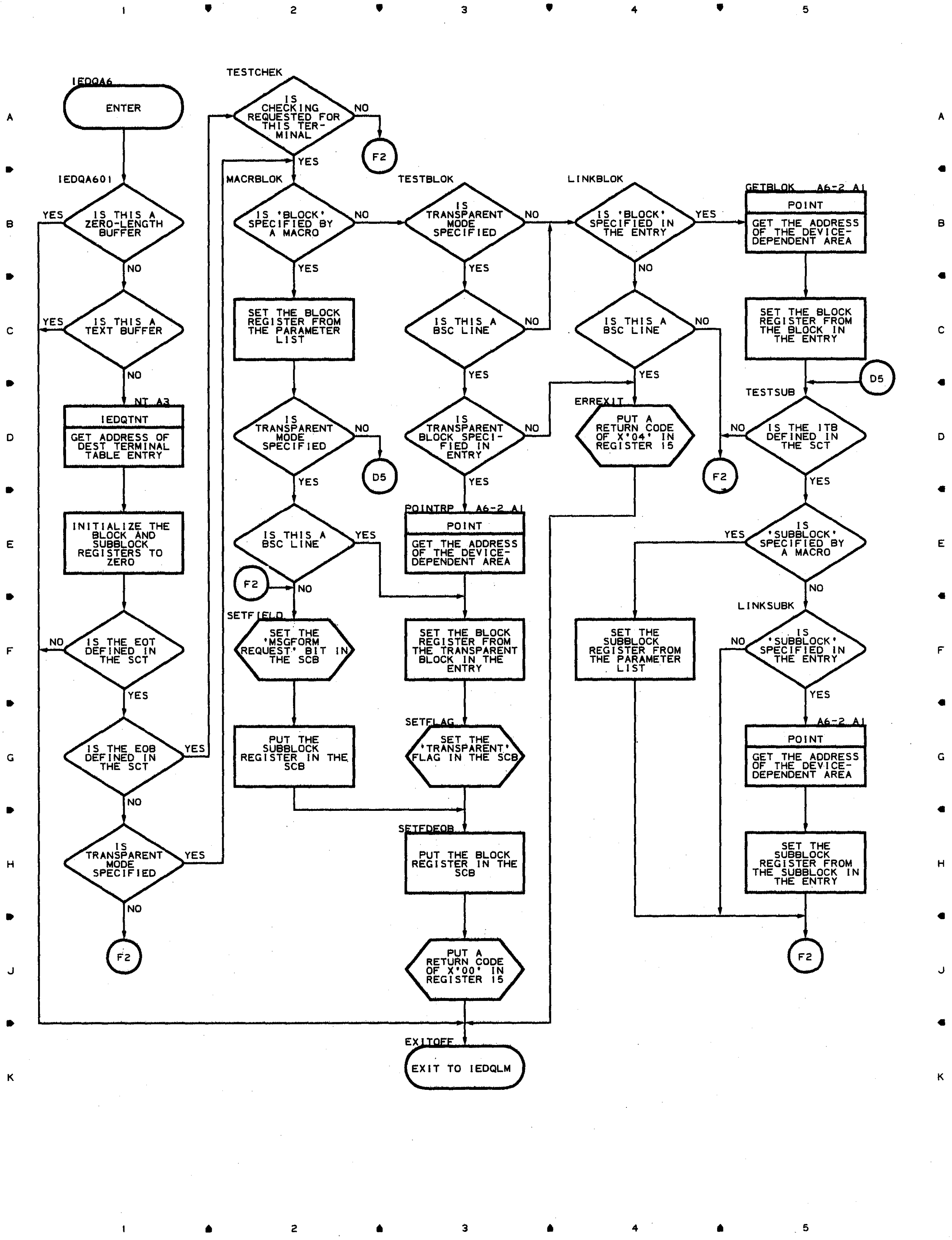

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQA4
ENTER
IEDQA401
IS THIS THE LAST BUFFER
YES
SAVE THE PARAMETER LIST ADDRESS IN THE SCB
NO
TESTDUPL
IS THIS A DUPLICATE HEADER BUFFER
YES
TURN OFF THE 'DUPLICATE HEADER' BIT IN THE SCB
NO
TESTSCAN
IS THIS A TEXT BUFFER
YES
NO
PUT THE SCAN POINTER OR THE DEFAULT VALUE IN THE SCB
INCLHDR
IS THIS A TSO BUFFER
YES
NO
IS THIS AN INCOMPLETE HEADER
NO
YES
SET THE 'INCOMPLETE HEADER' BIT ON IN THE SCB
TSTENTRY
IS THIS THE INPUT SIDE OF THE MH
NO
YES
INMSG
IS ANY DESTINATION FOUND
NO
SET UP TO TPOST THE BUFFER TO THE BUFFER RETURN QCB
YES
TESTMLTR
IS MULTIPLE ROUTING USED
NO
RESET THE 'MULTIPLE ROUTE' FIELD IN THE SCB
YES
AL A1
IEDQAL
GET THE SCAN POINTER ADDRESS
AX A2
IEDQAX
SCAN FOR THE UNIQUE CHARACTER
RETURN CODE FROM IEDQAX =
4
0
CLEAR THE 'MULTIPLE ROUTE' FIELD IN THE SCB
OUTMSG
IS THIS A ZERO-LENGTH BUFFER
NO
YES
TESTPOST
IS THIS THE END OF THE MESSAGE
YES
NO
PUT THE DESTINATION QCB ADDRESS IN THE BUFFER
PASS THE NUMBER OF RESERVE CHARACTERS IN THE BUFFER PREFIX
EXIT TO DSPPOST
ARE THERE ANY EMPTY UNITS
YES
NO
DELINK THE EMPTY UNITS FROM THE BUFFER
RB-2 A1
DSPPOSTR
TPOST THE EMPTY UNITS TO BUFFER RETURN QCB
TESTPROC
IS THE DESTINATION A PROCESS QCB
NO
YES
NT A3
IEDQTNT
GET THE TERMINAL ENTRY ADDRESS
POSTMPP
PUT THE READ-AHEAD QCB ADDRESS IN THE BUFFER
POSTDISP
PUT THE BUFFER DISPOSITION QCB ADDRESS IN THE BUFFER
EXIT TO IEDQGT
DOUBLEOA
IS A DOUBLE EOA PRESENT
NO
YES
GET THE EOA CONFIGURATION FROM THE SCT
IS AN EOA IN THE BUFFER
NO
YES
ELIMEOA
INCREMENT THE COUNT OF RESERVE CHARACTERS BY ONE
IS THE DESTINATION BSC TRANS-PARENT
YES
NO
EXIT TO IEDQGD

Chart A5 FORWARD ROUTINE



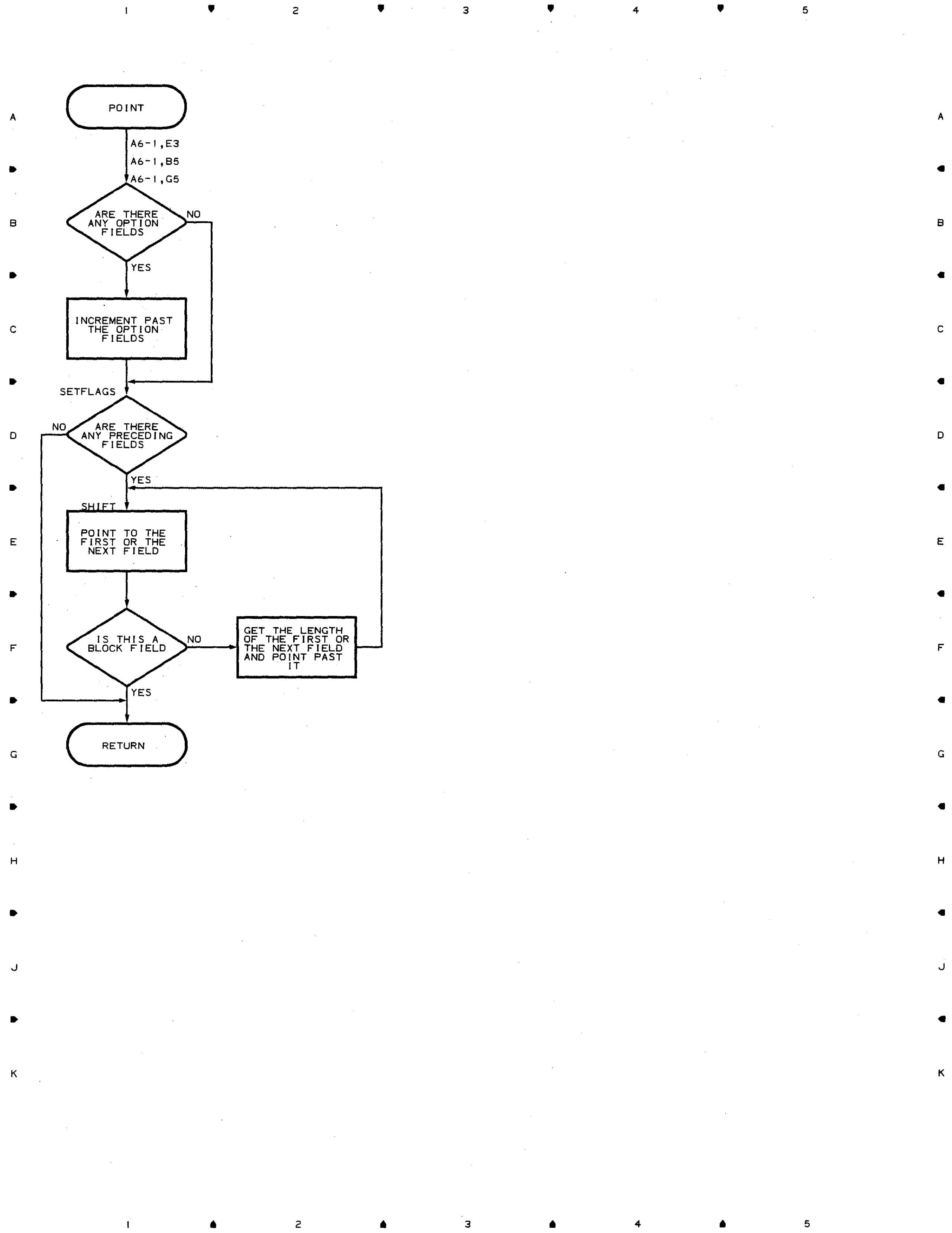
1 2 3 4 5
A B C D E F G H J K
IEDQA5
ENTER
IEDQA501
IS THIS A ZERO-LENGTH BUFFER
YES
NO
IS IT A NON-RECALLED TEXT BUFFER
YES
NO
IS THE TERMINAL IN EXTENDED LOCK MODE
NO
YES
RETURN
GET THE FIRST SUBSIDIARY PARAMETER LIST
UI A3
IEDQUI
ACTIVATE THE SUBSIDIARY ROUTINE
TESTRET
IS THE DESTINATION NAME IN THE MACRO
YES
NO
IS THE DESTINATION NAME IN THE OPTION FIELD
NO
YES
RETURNAE
IS AN OPTION FIELD FOUND
YES
NO
G2
ERRTEST
IS A USER ROUTINE SPECIFIED
NO
YES
GETUSER
BALR 14,15
ACTIVATE USER ROUTINE TO GET TERMINAL NAME
IS THE NAME RETURNED
NO
YES
A5 A3
LINKA1
LINK TO THE BINARY SEARCH ROUTINE
LINKA1
A5,K2
A5,F3
UI A3
IEDQUI
ACTIVATE IEDQA1 TO FIND THE TERMINAL NAME
RETURNA1
IS THE TERMINAL NAME FOUND
YES
NO
D4
LINKAV AV A1
IEDQAV
PUT THE DESTINATION QCB ADDRESS IN SCB
IS THIS A TSO TERMINAL
YES
G2
NO
RETURNA1
IS THE FIELD INCOMPLETE IN THIS BUFFER
YES
NO
RETURN
PREPA1 A5 A3
LINKA1
ACTIVATE THE BINARY SEARCH ROUTINE
IS THE ENTRY FROM MULTIPLE ROUTING
YES
NO
PERMERR
IS A DEAD-LETTER QUEUE SPECIFIED
YES
D4
NO
PUT A RETURN CODE OF X'04' IN REGISTER 15
IS THE ENTRY FROM MULTIPLE ROUTING
NO
YES
J5
ARE EOA CHARACTERS SPECIFIED
NO
YES
DOES THE FIELD CONTAIN AN EOA
YES
NO
AL A1
IEDQAL
GET ADDRESS OF LAST BYTE IN TERMINAL NAME
INITIALIZE THE SCB SECONDARY DESTINATION FIELD
EXITEOA
IS THE ENTRY FROM MULTIPLE ROUTING
NO
G2
YES
PUT A RETURN CODE OF X'08' IN REGISTER 15
J5
EXITCLR
RESET THE SCB SECONDARY DESTINATION FIELD TO ZERO
EXIT
RETURN

Chart A6-1 LINE CONTROL INITIALIZATION ROUTINE



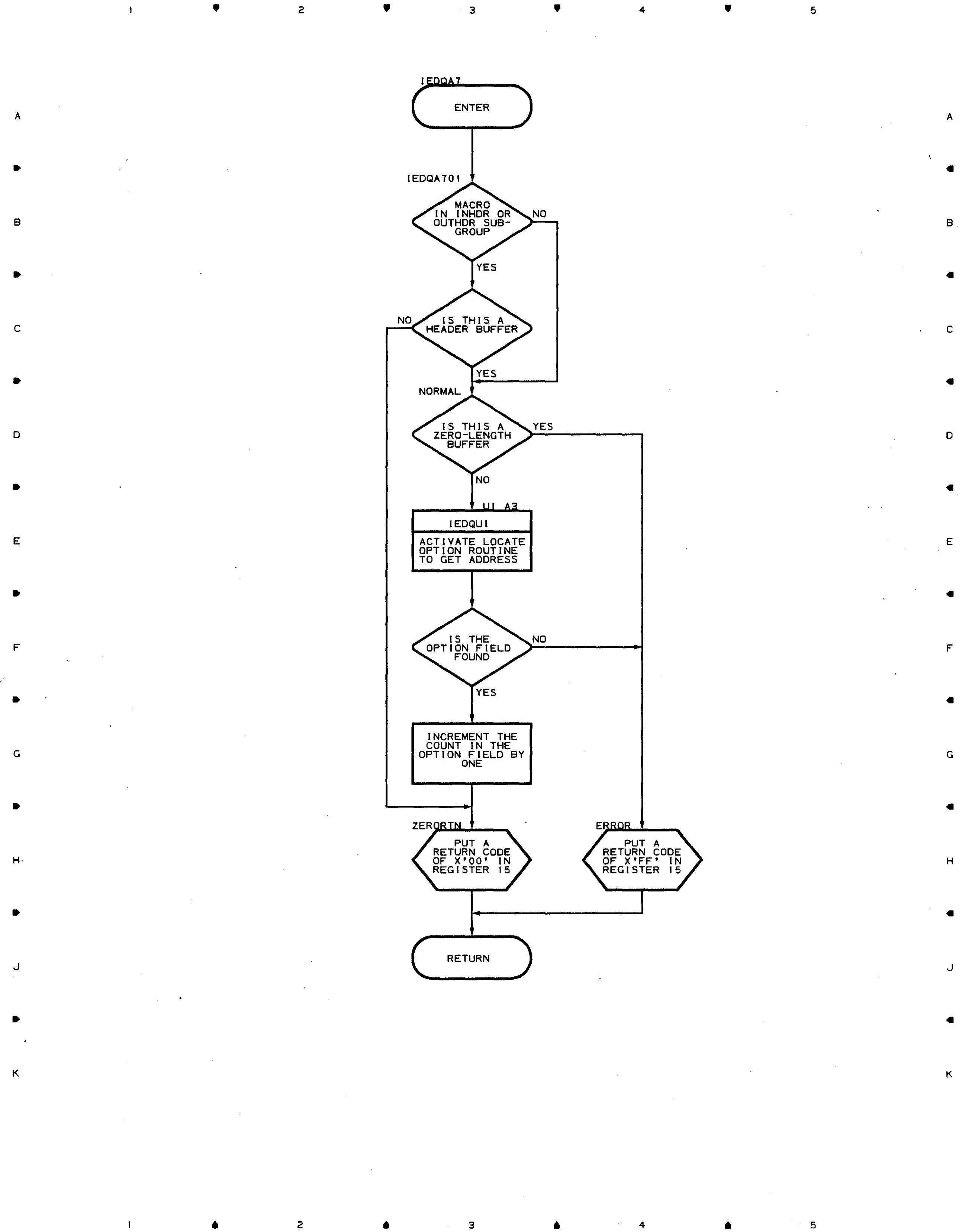
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQA6
ENTER
IEDQA601
IS THIS A ZERO-LENGTH BUFFER
YES
NO
IS THIS A TEXT BUFFER
YES
NO
NT A3
IEDQTNT
GET ADDRESS OF DEST TERMINAL TABLE ENTRY
INITIALIZE THE BLOCK AND SUBBLOCK REGISTERS TO ZERO
IS THE EOT DEFINED IN THE SCT
NO
YES
IS THE EOB DEFINED IN THE SCT
YES
NO
IS TRANSPARENT MODE SPECIFIED
YES
NO
F2
TESTCHEK
IS CHECKING REQUESTED FOR THIS TERMINAL
NO
F2
YES
MACRBLOK
IS 'BLOCK' SPECIFIED BY A MACRO
NO
YES
SET THE BLOCK REGISTER FROM THE PARAMETER LIST
IS TRANSPARENT MODE SPECIFIED
NO
D5
YES
IS THIS A BSC LINE
YES
F2
NO
SETFIELD
SET THE 'MSGFORM REQUEST' BIT IN THE SCB
PUT THE SUBBLOCK REGISTER IN THE SCB
TESTBLOK
IS TRANSPARENT MODE SPECIFIED
NO
YES
IS THIS A BSC LINE
NO
YES
IS TRANSPARENT BLOCK SPECI-FIED IN ENTRY
NO
YES
POINTRP A6-2 A1
POINT
GET THE ADDRESS OF THE DEVICE-DEPENDENT AREA
SET THE BLOCK REGISTER FROM THE TRANSPARENT BLOCK IN THE ENTRY
SETFLAG
SET THE 'TRANSPARENT' FLAG IN THE SCB
SETFDEOB
PUT THE BLOCK REGISTER IN THE SCB
PUT A RETURN CODE OF X'00' IN REGISTER 15
EXITOFF
EXIT TO IEDQLM
LINKBLOK
IS 'BLOCK' SPECIFIED IN THE ENTRY
YES
NO
IS THIS A BSC LINE
NO
YES
ERREXIT
PUT A RETURN CODE OF X'04' IN REGISTER 15
F2
SET THE SUBBLOCK REGISTER FROM THE PARAMETER LIST
GETBLOK A6-2 A1
POINT
GET THE ADDRESS OF THE DEVICE-DEPENDENT AREA
SET THE BLOCK REGISTER FROM THE BLOCK IN THE ENTRY
D5
TESTSUB
IS THE ITB DEFINED IN THE SCT
NO
YES
IS 'SUBBLOCK' SPECIFIED BY A MACRO
YES
NO
LINKSUBK
IS 'SUBBLOCK' SPECIFIED IN THE ENTRY
NO
YES
A6-2 A1
POINT
GET THE ADDRESS OF THE DEVICE-DEPENDENT AREA
SET THE SUBBLOCK REGISTER FROM THE SUBBLOCK IN THE ENTRY
F2

Chart A6-2 LINE CONTROL INITIALIZATION ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
POINT
A6-1,E3
A6-1,B5
A6-1,G5
ARE THERE ANY OPTION FIELDS
NO
YES
INCREMENT PAST THE OPTION FIELDS
SETFLAGS
NO
ARE THERE ANY PRECEDING FIELDS
YES
SHIFT
POINT TO THE FIRST OR THE NEXT FIELD
IS THIS A BLOCK FIELD
NO
GET THE LENGTH OF THE FIRST OR THE NEXT FIELD AND POINT PAST IT
YES
RETURN

Chart A7 COUNTER ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQA7
ENTER
IEDQA701
MACRO IN INHDR OR OUTHDR SUB-GROUP
NO
YES
IS THIS A HEADER BUFFER
NO
YES
NORMAL
IS THIS A ZERO-LENGTH BUFFER
YES
NO
UI A3
IEDQUI
ACTIVATE LOCATE OPTION ROUTINE TO GET ADDRESS
IS THE OPTION FIELD FOUND
NO
YES
INCREMENT THE COUNT IN THE OPTION FIELD BY ONE
ZERORTN
PUT A RETURN CODE OF X'00' IN REGISTER 15
ERROR
PUT A RETURN CODE OF X'FF' IN REGISTER 15
RETURN

Chart A8 MULTIPLE INSERT AT OFFSET ROUTINE

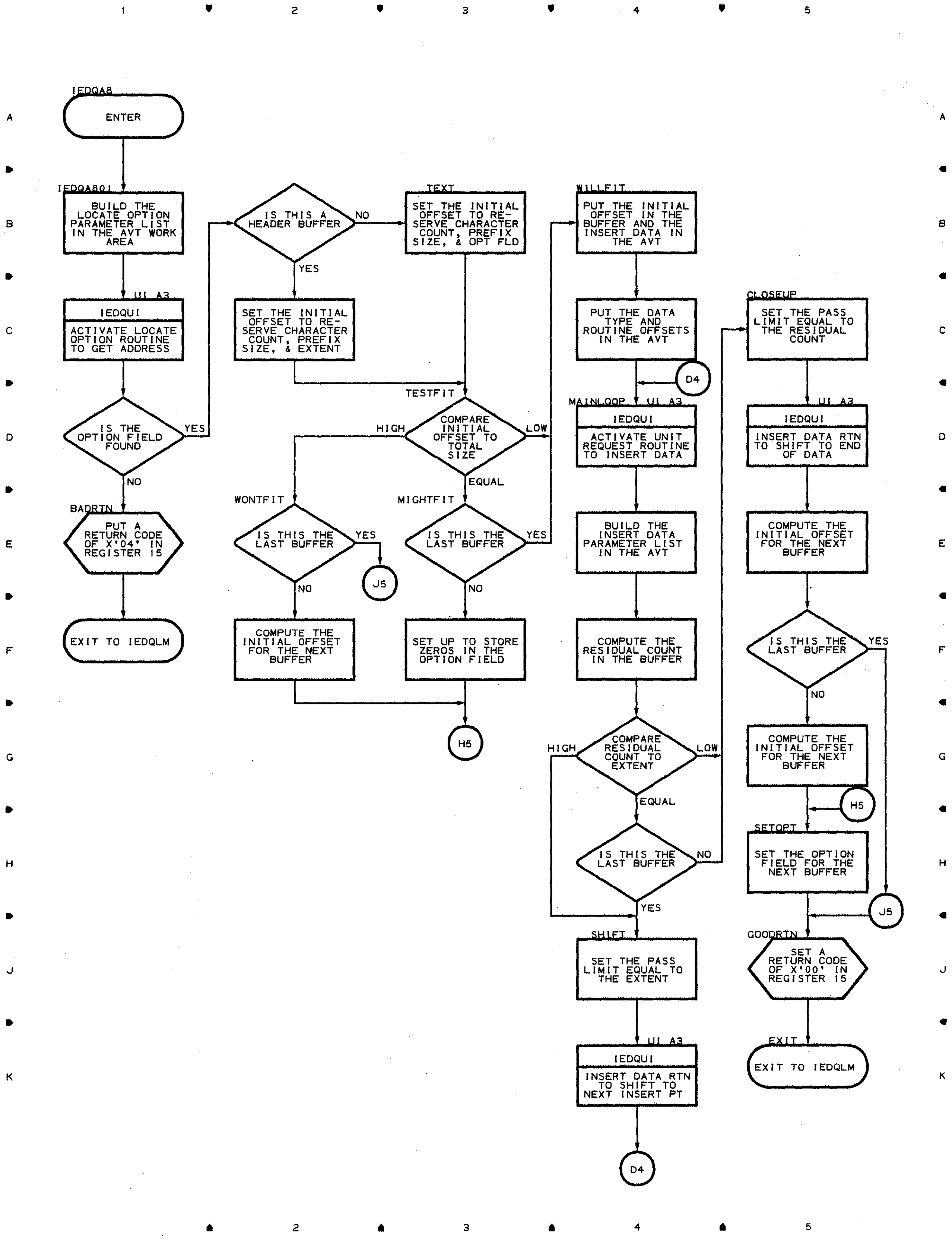

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQA8
ENTER
IEDQA801
BUILD THE LOCATE OPTION PARAMETER LIST IN THE AVT WORK AREA
UI A3
IEDQUI
ACTIVATE LOCATE OPTION ROUTINE TO GET ADDRESS
IS THE OPTION FIELD FOUND
YES
NO
BADRTN
PUT A RETURN CODE OF X'04' IN REGISTER 15
EXIT TO IEDQLM
IS THIS A HEADER BUFFER
NO
YES
SET THE INITIAL OFFSET TO RESERVE CHARACTER COUNT, PREFIX SIZE, & EXTENT
TEXT
SET THE INITIAL OFFSET TO RESERVE CHARACTER COUNT, PREFIX SIZE, & OPT FLD
TESTFIT
COMPARE INITIAL OFFSET TO TOTAL SIZE
HIGH
LOW
EQUAL
WONTFIT
IS THIS THE LAST BUFFER
YES
J5
NO
COMPUTE THE INITIAL OFFSET FOR THE NEXT BUFFER
MIGHTFIT
IS THIS THE LAST BUFFER
YES
NO
SET UP TO STORE ZEROS IN THE OPTION FIELD
H5
WILLFIT
PUT THE INITIAL OFFSET IN THE BUFFER AND THE INSERT DATA IN THE AVT
PUT THE DATA TYPE AND ROUTINE OFFSETS IN THE AVT
D4
MAINLOOP
UI A3
IEDQUI
ACTIVATE UNIT REQUEST ROUTINE TO INSERT DATA
BUILD THE INSERT DATA PARAMETER LIST IN THE AVT
COMPUTE THE RESIDUAL COUNT IN THE BUFFER
COMPARE RESIDUAL COUNT TO EXTENT
HIGH
LOW
EQUAL
IS THIS THE LAST BUFFER
NO
YES
SHIFT
SET THE PASS LIMIT EQUAL TO THE EXTENT
UI A3
IEDQUI
INSERT DATA RTN TO SHIFT TO NEXT INSERT PT
D4
CLOSEUP
SET THE PASS LIMIT EQUAL TO THE RESIDUAL COUNT
UI A3
IEDQUI
INSERT DATA RTN TO SHIFT TO END OF DATA
COMPUTE THE INITIAL OFFSET FOR THE NEXT BUFFER
IS THIS THE LAST BUFFER
YES
NO
COMPUTE THE INITIAL OFFSET FOR THE NEXT BUFFER
H5
SETOPT
SET THE OPTION FIELD FOR THE NEXT BUFFER
J5
GOODRTN
SET A RETURN CODE OF X'00' IN REGISTER 15
EXIT
EXIT TO IEDQLM

Chart BA MULTIPLE ROUTING SUBTASK

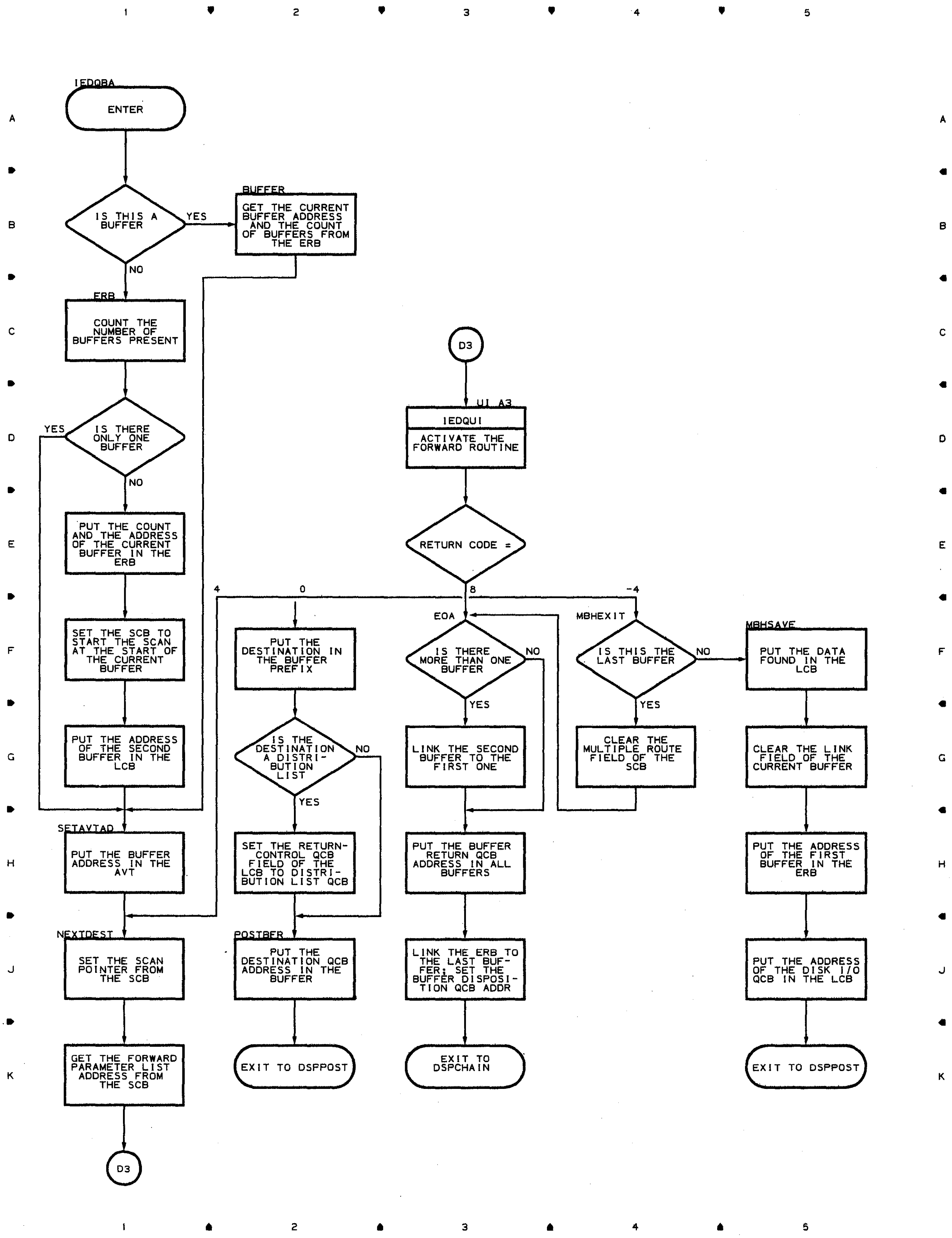
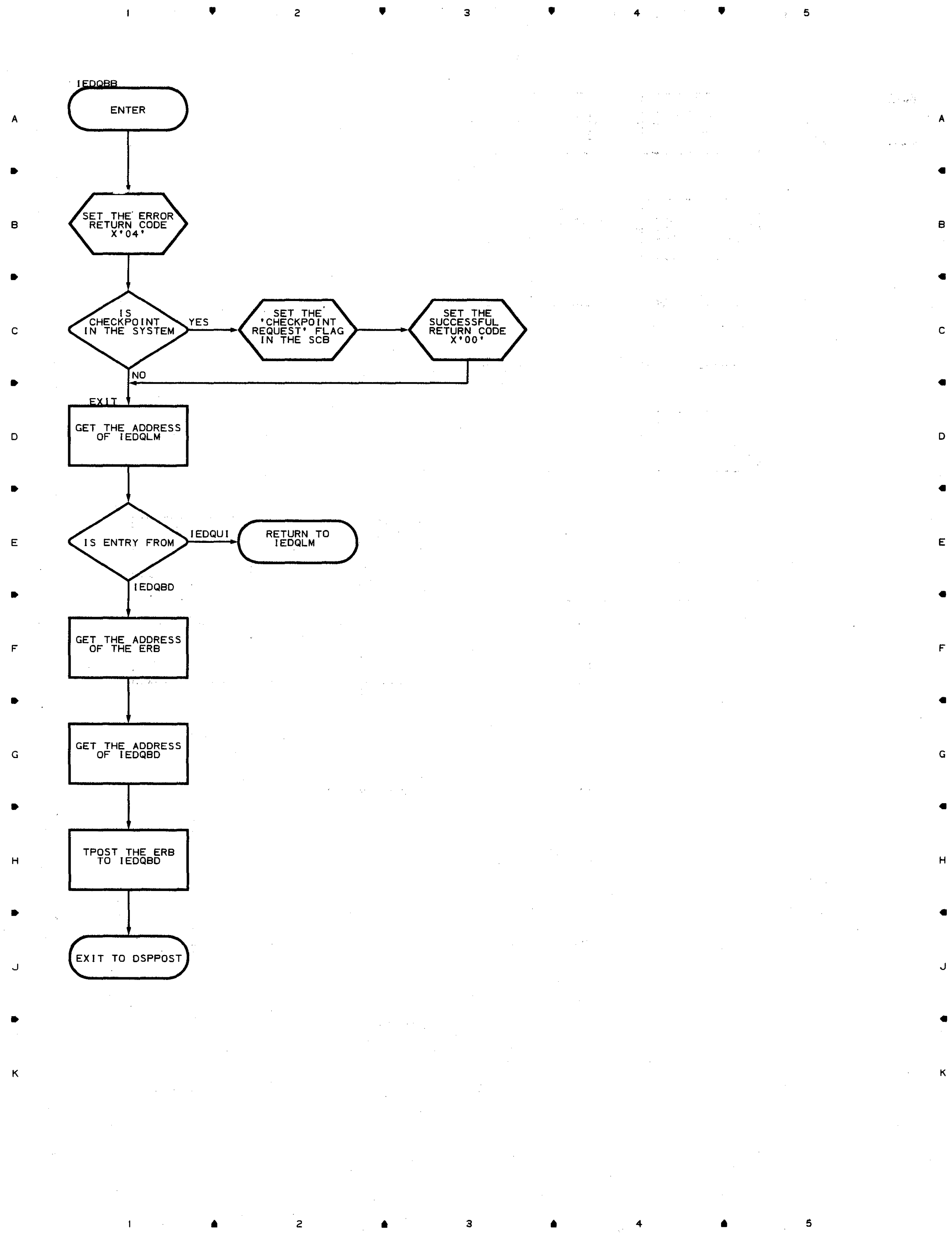

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQBA
ENTER
IS THIS A BUFFER
YES
NO
BUFFER
GET THE CURRENT BUFFER ADDRESS AND THE COUNT OF BUFFERS FROM THE ERB
ERB
COUNT THE NUMBER OF BUFFERS PRESENT
IS THERE ONLY ONE BUFFER
YES
NO
PUT THE COUNT AND THE ADDRESS OF THE CURRENT BUFFER IN THE ERB
SET THE SCB TO START THE SCAN AT THE START OF THE CURRENT BUFFER
PUT THE ADDRESS OF THE SECOND BUFFER IN THE LCB
SETAVTAD
PUT THE BUFFER ADDRESS IN THE AVT
NEXTDEST
SET THE SCAN POINTER FROM THE SCB
GET THE FORWARD PARAMETER LIST ADDRESS FROM THE SCB
D3
D3
UI A3
IEDQUI
ACTIVATE THE FORWARD ROUTINE
RETURN CODE =
4
0
8
-4
PUT THE DESTINATION IN THE BUFFER PREFIX
IS THE DESTINATION A DISTRIBUTION LIST
NO
YES
SET THE RETURN-CONTROL QCB FIELD OF THE LCB TO DISTRIBUTION LIST QCB
POSTBFR
PUT THE DESTINATION QCB ADDRESS IN THE BUFFER
EXIT TO DSPPOST
EOA
IS THERE MORE THAN ONE BUFFER
NO
YES
LINK THE SECOND BUFFER TO THE FIRST ONE
PUT THE BUFFER RETURN QCB ADDRESS IN ALL BUFFERS
LINK THE ERB TO THE LAST BUFFER; SET THE BUFFER DISPOSITION QCB ADDR
EXIT TO DSPCHAIN
MBHEXIT
IS THIS THE LAST BUFFER
NO
YES
CLEAR THE MULTIPLE ROUTE FIELD OF THE SCB
MBHSAVE
PUT THE DATA FOUND IN THE LCB
CLEAR THE LINK FIELD OF THE CURRENT BUFFER
PUT THE ADDRESS OF THE FIRST BUFFER IN THE ERB
PUT THE ADDRESS OF THE DISK I/O QCB IN THE LCB
EXIT TO DSPPOST

Chart BB CHECKPOINT REQUEST ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQBB
ENTER
SET THE ERROR RETURN CODE X'04'
IS CHECKPOINT IN THE SYSTEM
YES
SET THE 'CHECKPOINT REQUEST' FLAG IN THE SCB
SET THE SUCCESSFUL RETURN CODE X'00'
NO
EXIT
GET THE ADDRESS OF IEDQLM
IS ENTRY FROM
IEDQUI
RETURN TO IEDQLM
IEDQBD
GET THE ADDRESS OF THE ERB
GET THE ADDRESS OF IEDQBD
TPOST THE ERB TO IEDQBD
EXIT TO DSPPOST

Chart BC DISTRIBUTION LIST SUBTASK

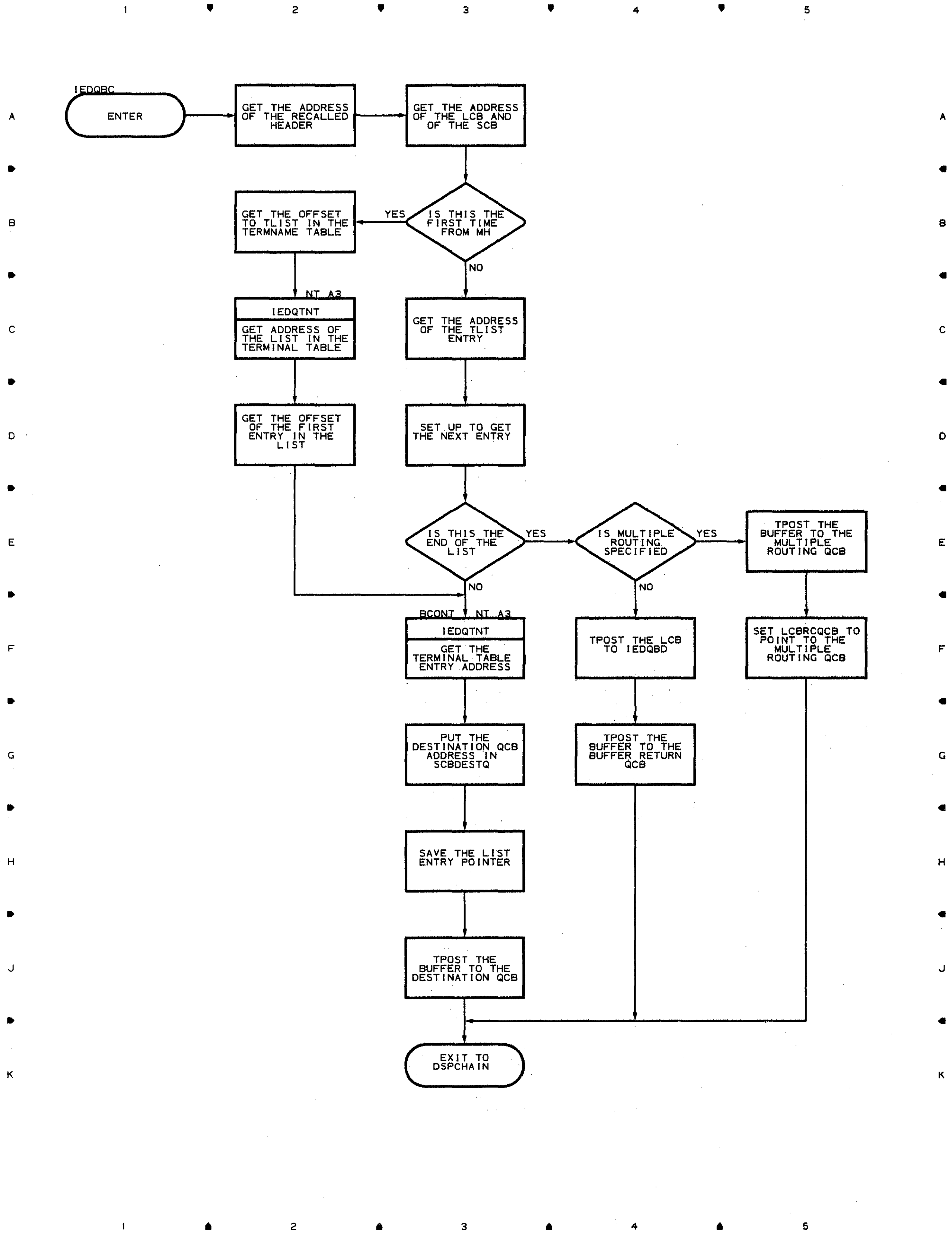
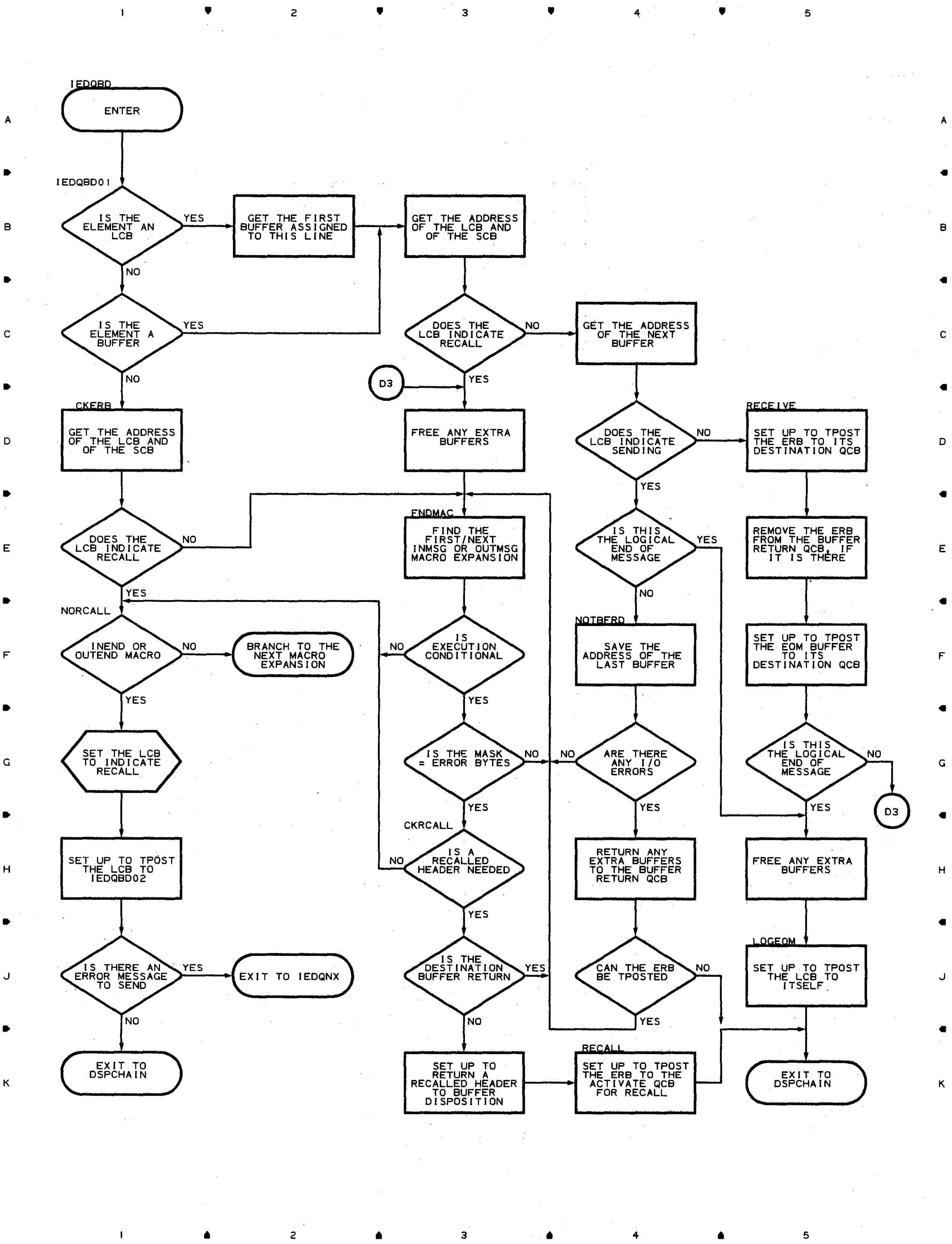
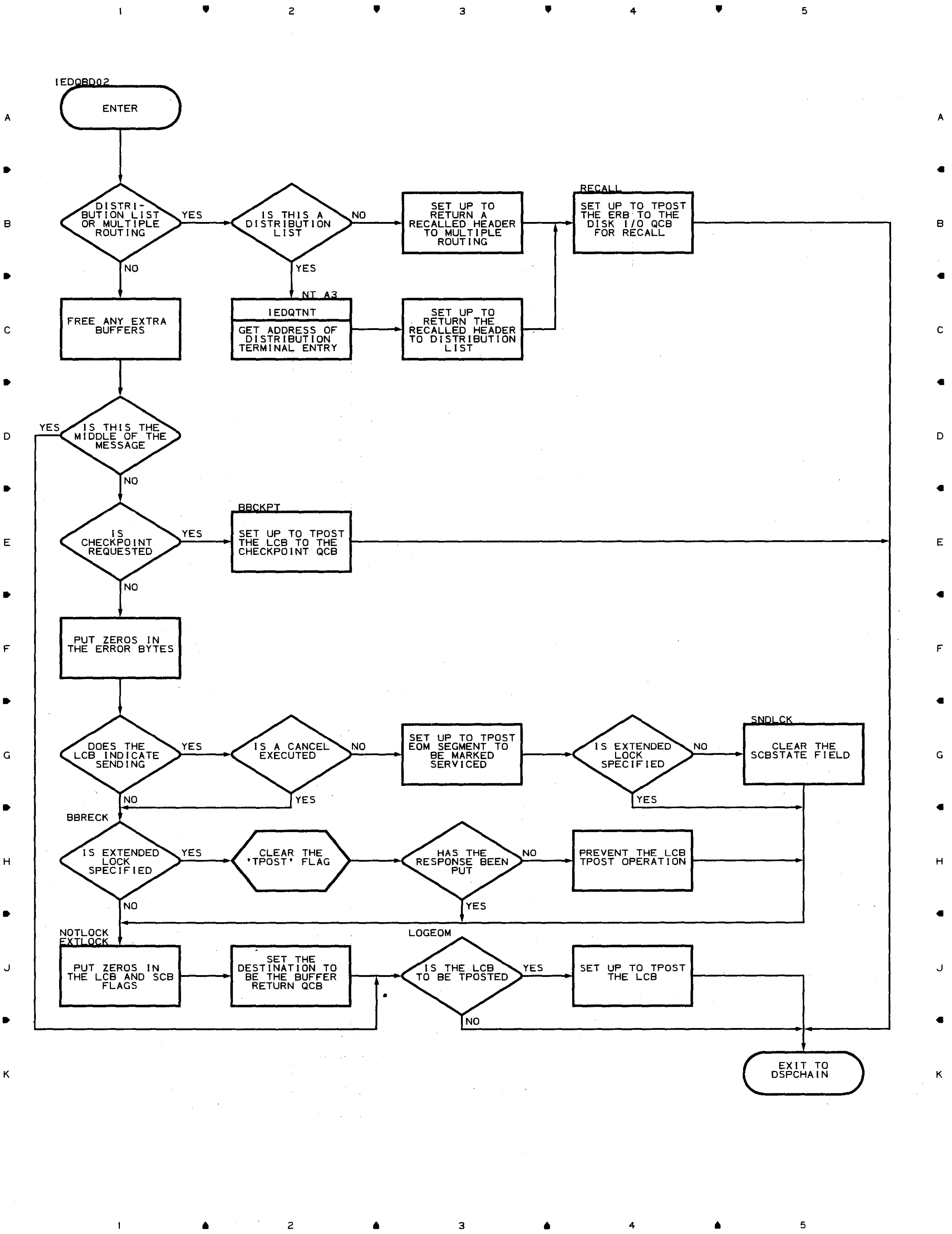

IEDQBC
ENTER
GET THE ADDRESS OF THE RECALLED HEADER
GET THE ADDRESS OF THE LCB AND OF THE SCB
IS THIS THE FIRST TIME FROM MH
YES
NO
GET THE OFFSET TO TLIST IN THE TERMNAME TABLE
NT A3
IEDQTNT
GET ADDRESS OF THE LIST IN THE TERMINAL TABLE
GET THE ADDRESS OF THE TLIST ENTRY
GET THE OFFSET OF THE FIRST ENTRY IN THE LIST
SET UP TO GET THE NEXT ENTRY
IS THIS THE END OF THE LIST
YES
NO
IS MULTIPLE ROUTING SPECIFIED
YES
NO
TPOST THE BUFFER TO THE MULTIPLE ROUTING QCB
BCONT
NT A3
IEDQTNT
GET THE TERMINAL TABLE ENTRY ADDRESS
TPOST THE LCB TO IEDQBD
SET LCBRCQCB TO POINT TO THE MULTIPLE ROUTING QCB
PUT THE DESTINATION QCB ADDRESS IN SCBDESTQ
TPOST THE BUFFER TO THE BUFFER RETURN QCB
SAVE THE LIST ENTRY POINTER
TPOST THE BUFFER TO THE DESTINATION QCB
EXIT TO DSPCHAIN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart BD-1 BUFFER DISPOSITION SUBTASK



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQBD
ENTER
IEDQBD01
IS THE ELEMENT AN LCB
YES
GET THE FIRST BUFFER ASSIGNED TO THIS LINE
GET THE ADDRESS OF THE LCB AND OF THE SCB
NO
IS THE ELEMENT A BUFFER
YES
DOES THE LCB INDICATE RECALL
NO
GET THE ADDRESS OF THE NEXT BUFFER
NO
D3
YES
CKERB
GET THE ADDRESS OF THE LCB AND OF THE SCB
FREE ANY EXTRA BUFFERS
DOES THE LCB INDICATE SENDING
NO
RECEIVE
SET UP TO TPOST THE ERB TO ITS DESTINATION QCB
YES
ENDMAC
DOES THE LCB INDICATE RECALL
NO
FIND THE FIRST/NEXT INMSG OR OUTMSG MACRO EXPANSION
IS THIS THE LOGICAL END OF MESSAGE
YES
REMOVE THE ERB FROM THE BUFFER RETURN QCB, IF IT IS THERE
YES
NORCALL
NO
NOTBFRD
INEND OR OUTEND MACRO
NO
BRANCH TO THE NEXT MACRO EXPANSION
NO
IS EXECUTION CONDITIONAL
SAVE THE ADDRESS OF THE LAST BUFFER
SET UP TO TPOST THE EOM BUFFER TO ITS DESTINATION QCB
YES
YES
SET THE LCB TO INDICATE RECALL
IS THE MASK = ERROR BYTES
NO
NO
ARE THERE ANY I/O ERRORS
IS THIS THE LOGICAL END OF MESSAGE
NO
YES
YES
YES
D3
CKRCALL
SET UP TO TPOST THE LCB TO IEDQBD02
NO
IS A RECALLED HEADER NEEDED
RETURN ANY EXTRA BUFFERS TO THE BUFFER RETURN QCB
FREE ANY EXTRA BUFFERS
YES
LOGEOM
IS THERE AN ERROR MESSAGE TO SEND
YES
EXIT TO IEDQNX
IS THE DESTINATION BUFFER RETURN
YES
CAN THE ERB BE TPOSTED
NO
SET UP TO TPOST THE LCB TO ITSELF
NO
NO
YES
RECALL
EXIT TO DSPCHAIN
SET UP TO RETURN A RECALLED HEADER TO BUFFER DISPOSITION
SET UP TO TPOST THE ERB TO THE ACTIVATE QCB FOR RECALL
EXIT TO DSPCHAIN

Chart BD-2 BUFFER DISPOSITION SUBTASK



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQBD02
ENTER
DISTRI-BUTION LIST OR MULTIPLE ROUTING
YES
NO
IS THIS A DISTRIBUTION LIST
NO
YES
SET UP TO RETURN A RECALLED HEADER TO MULTIPLE ROUTING
RECALL
SET UP TO TPOST THE ERB TO THE DISK I/O QCB FOR RECALL
FREE ANY EXTRA BUFFERS
NT A3
IEDQTNT
GET ADDRESS OF DISTRIBUTION TERMINAL ENTRY
SET UP TO RETURN THE RECALLED HEADER TO DISTRIBUTION LIST
YES
IS THIS THE MIDDLE OF THE MESSAGE
NO
IS CHECKPOINT REQUESTED
YES
NO
BBCKPT
SET UP TO TPOST THE LCB TO THE CHECKPOINT QCB
PUT ZEROS IN THE ERROR BYTES
DOES THE LCB INDICATE SENDING
YES
NO
IS A CANCEL EXECUTED
NO
YES
SET UP TO TPOST EOM SEGMENT TO BE MARKED SERVICED
IS EXTENDED LOCK SPECIFIED
NO
YES
SNDLCK
CLEAR THE SCBSTATE FIELD
BBRECK
IS EXTENDED LOCK SPECIFIED
YES
NO
CLEAR THE 'TPOST' FLAG
HAS THE RESPONSE BEEN PUT
NO
YES
PREVENT THE LCB TPOST OPERATION
NOTLOCK
EXTLOCK
PUT ZEROS IN THE LCB AND SCB FLAGS
SET THE DESTINATION TO BE THE BUFFER RETURN QCB
LOGEOM
IS THE LCB TO BE TPOSTED
YES
NO
SET UP TO TPOST THE LCB
EXIT TO DSPCHAIN

Chart BE LOCK ROUTINE

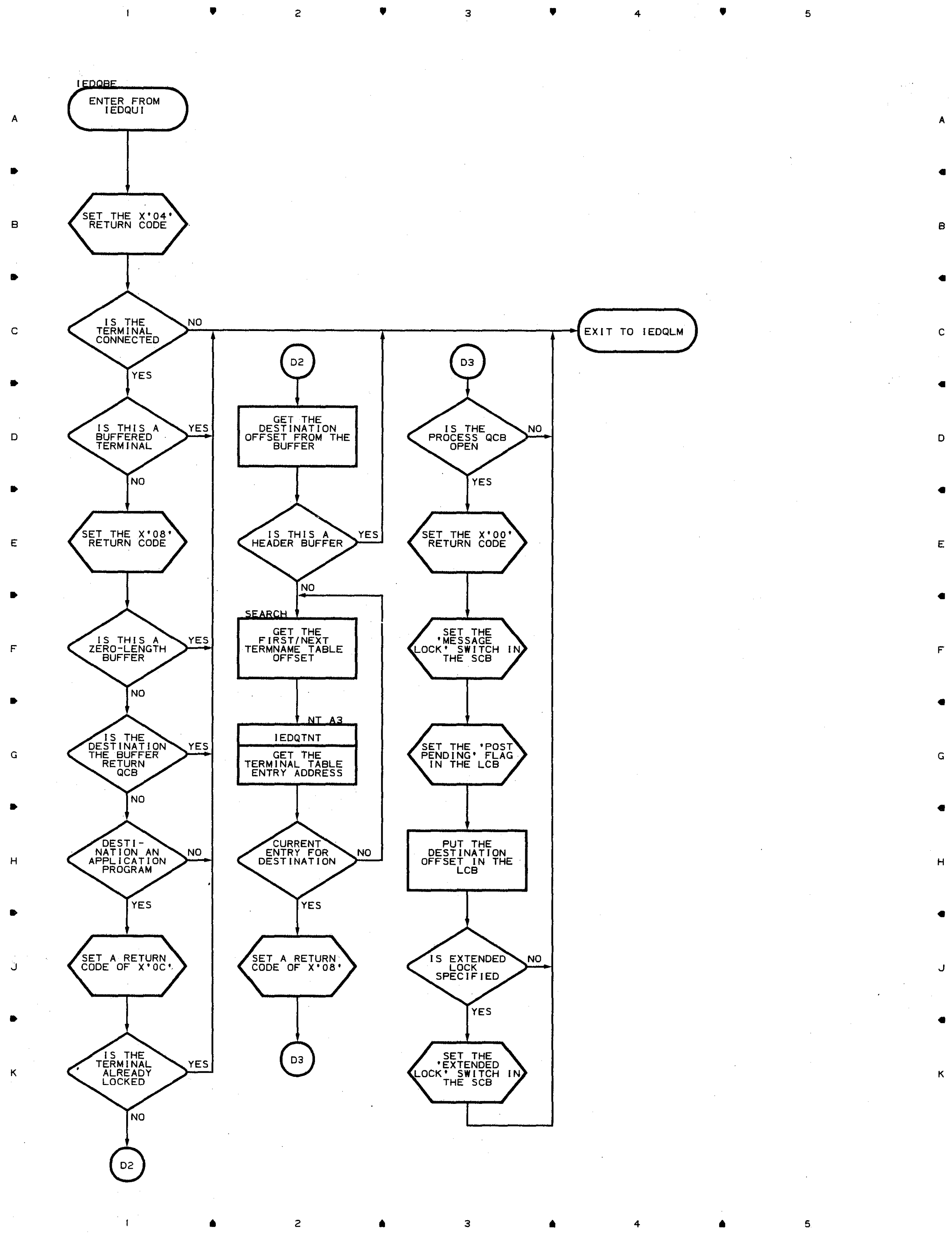

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQBE
ENTER FROM IEDQUI
SET THE X'04' RETURN CODE
IS THE TERMINAL CONNECTED
NO
EXIT TO IEDQLM
YES
IS THIS A BUFFERED TERMINAL
YES
NO
SET THE X'08' RETURN CODE
IS THIS A ZERO-LENGTH BUFFER
YES
NO
IS THE DESTINATION THE BUFFER RETURN QCB
YES
NO
DESTI- NATION AN APPLICATION PROGRAM
NO
YES
SET A RETURN CODE OF X'0C'
IS THE TERMINAL ALREADY LOCKED
YES
NO
D2
D2
GET THE DESTINATION OFFSET FROM THE BUFFER
IS THIS A HEADER BUFFER
YES
NO
SEARCH
GET THE FIRST/NEXT TERMNAME TABLE OFFSET
NT A3
IEDQTNT
GET THE TERMINAL TABLE ENTRY ADDRESS
CURRENT ENTRY FOR DESTINATION
NO
YES
SET A RETURN CODE OF X'08'
D3
D3
IS THE PROCESS QCB OPEN
NO
YES
SET THE X'00' RETURN CODE
SET THE 'MESSAGE LOCK' SWITCH IN THE SCB
SET THE 'POST PENDING' FLAG IN THE LCB
PUT THE DESTINATION OFFSET IN THE LCB
IS EXTENDED LOCK SPECIFIED
NO
YES
SET THE 'EXTENDED LOCK' SWITCH IN THE SCB

Chart BF UNLOCK ROUTINE

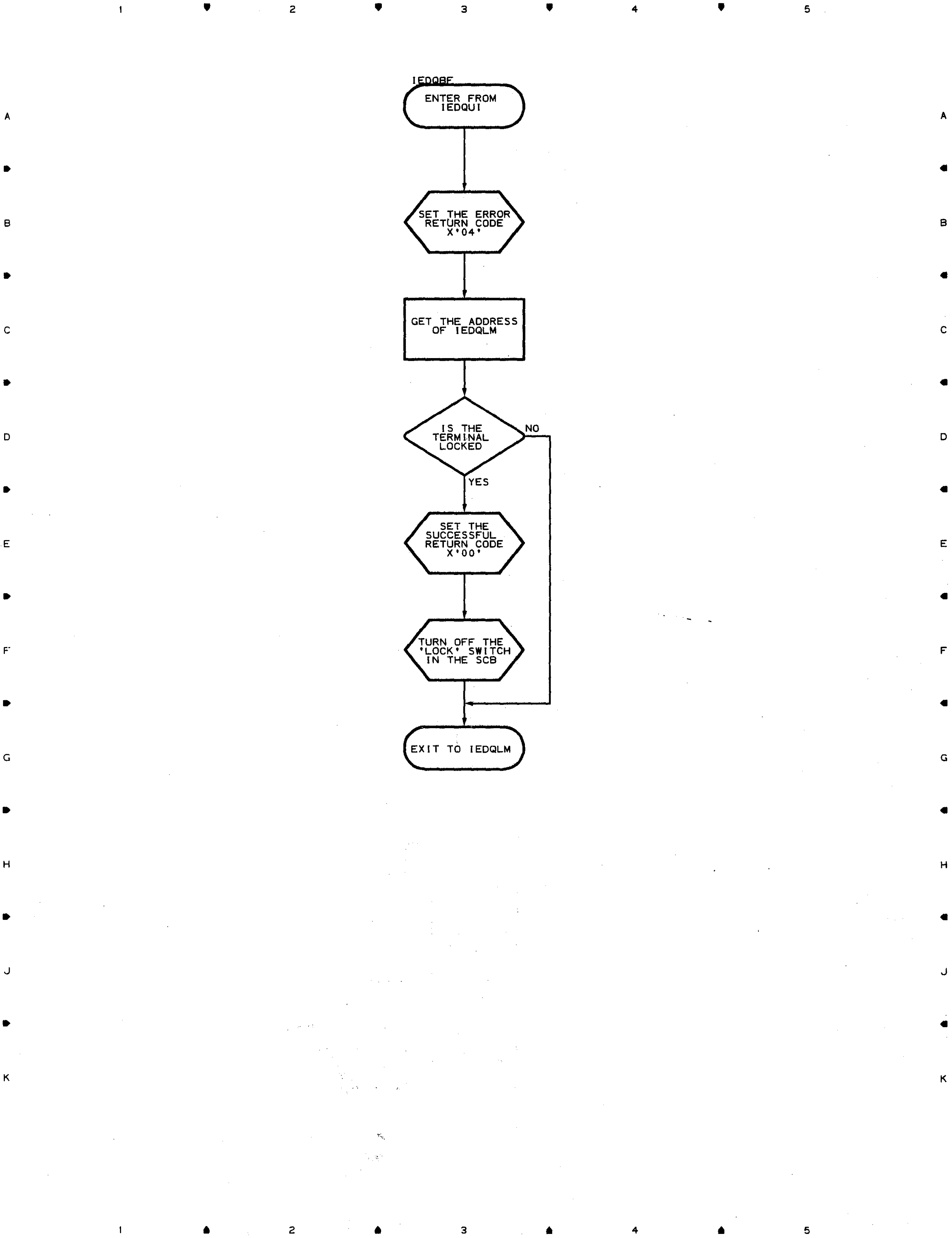

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQBF
ENTER FROM IEDQUI
SET THE ERROR RETURN CODE X'04'
GET THE ADDRESS OF IEDQLM
IS THE TERMINAL LOCKED
NO
YES
SET THE SUCCESSFUL RETURN CODE X'00'
TURN OFF THE 'LOCK' SWITCH IN THE SCB
EXIT TO IEDQLM

Chart BG CASCADE LIST SUBTASK

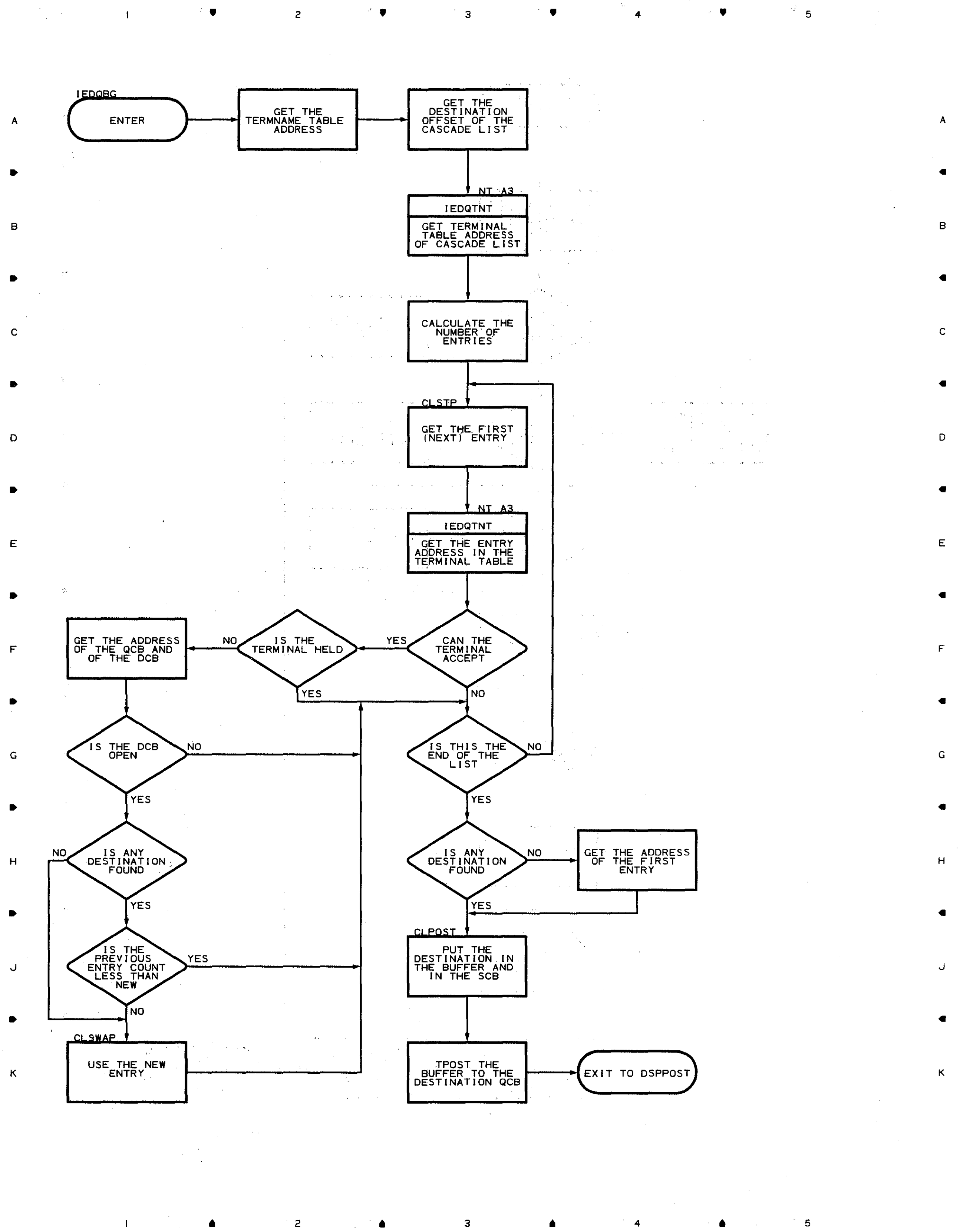

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQBG
ENTER
GET THE TERMNAME TABLE ADDRESS
GET THE DESTINATION OFFSET OF THE CASCADE LIST
NT A3
IEDQTNT
GET TERMINAL TABLE ADDRESS OF CASCADE LIST
CALCULATE THE NUMBER OF ENTRIES
CLSTP
GET THE FIRST (NEXT) ENTRY
NT A3
IEDQTNT
GET THE ENTRY ADDRESS IN THE TERMINAL TABLE
CAN THE TERMINAL ACCEPT
YES
NO
IS THE TERMINAL HELD
NO
YES
GET THE ADDRESS OF THE QCB AND OF THE DCB
IS THE DCB OPEN
NO
YES
IS THIS THE END OF THE LIST
NO
YES
IS ANY DESTINATION FOUND
NO
YES
IS ANY DESTINATION FOUND
NO
YES
GET THE ADDRESS OF THE FIRST ENTRY
IS THE PREVIOUS ENTRY COUNT LESS THAN NEW
YES
NO
CLPOST
PUT THE DESTINATION IN THE BUFFER AND IN THE SCB
CLSWAP
USE THE NEW ENTRY
TPOST THE BUFFER TO THE DESTINATION QCB
EXIT TO DSPPOST

Chart BL MESSAGE GENERATION ROUTINE

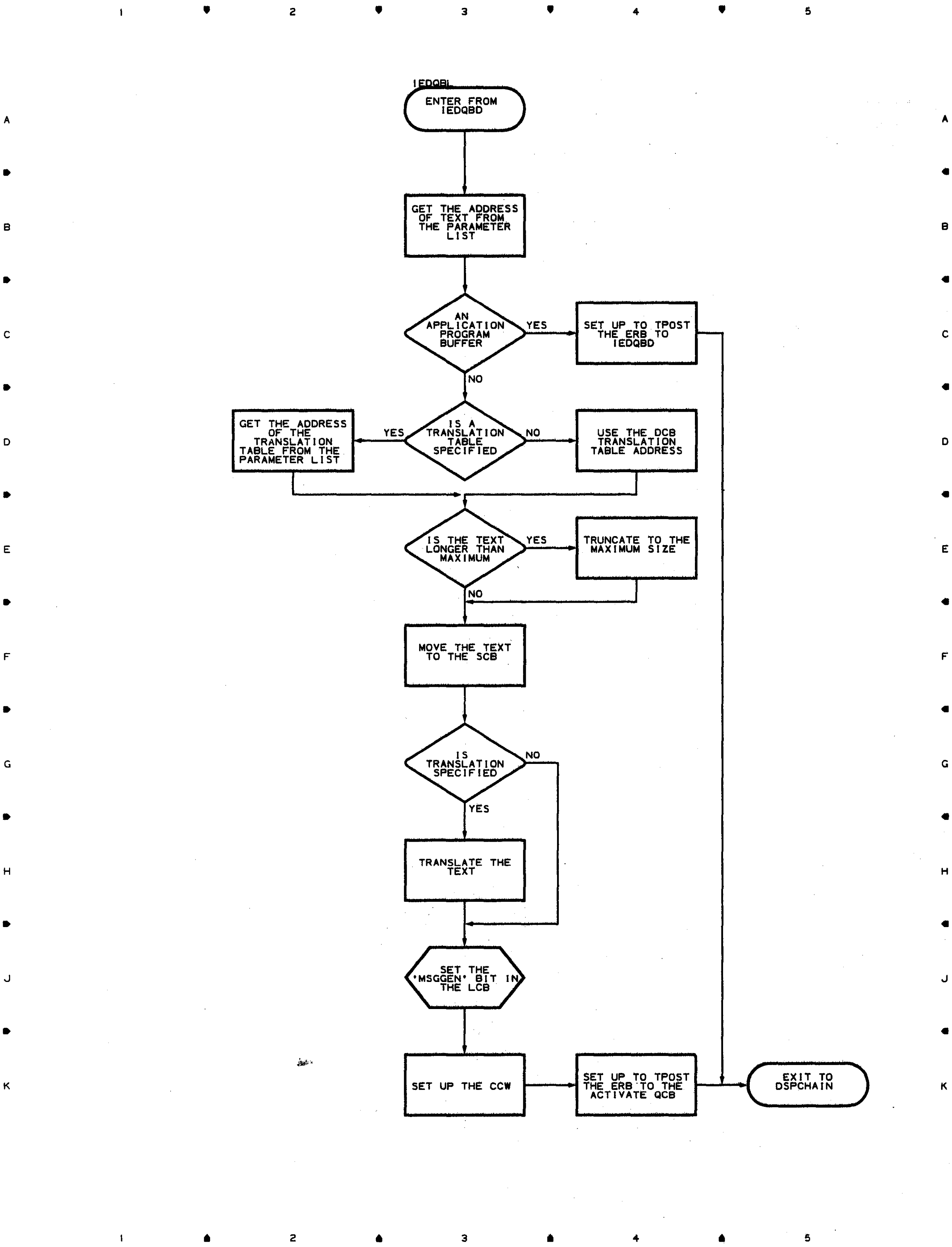
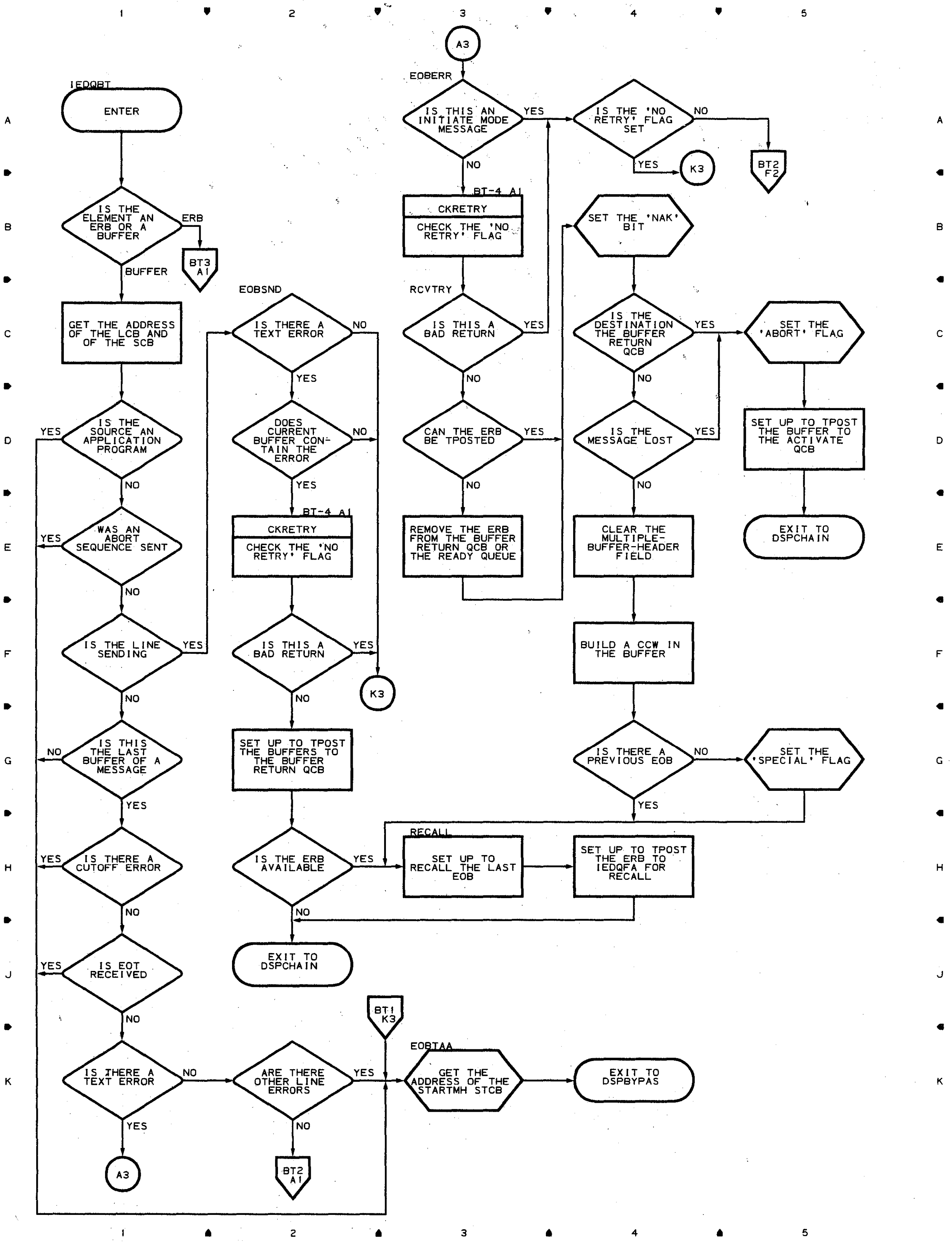

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQBL
ENTER FROM IEDQBD
GET THE ADDRESS OF TEXT FROM THE PARAMETER LIST
AN APPLICATION PROGRAM BUFFER
YES
SET UP TO TPOST THE ERB TO IEDQBD
NO
IS A TRANSLATION TABLE SPECIFIED
YES
GET THE ADDRESS OF THE TRANSLATION TABLE FROM THE PARAMETER LIST
NO
USE THE DCB TRANSLATION TABLE ADDRESS
IS THE TEXT LONGER THAN MAXIMUM
YES
TRUNCATE TO THE MAXIMUM SIZE
NO
MOVE THE TEXT TO THE SCB
IS TRANSLATION SPECIFIED
NO
YES
TRANSLATE THE TEXT
SET THE 'MSGGEN' BIT IN THE LCB
SET UP THE CCW
SET UP TO TPOST THE ERB TO THE ACTIVATE QCB
EXIT TO DSPCHAIN

Chart BT-1 EOB/ETB HANDLING SUBTASK



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQBT
ENTER
IS THE ELEMENT AN ERB OR A BUFFER
ERB
BT3 A1
BUFFER
GET THE ADDRESS OF THE LCB AND OF THE SCB
IS THE SOURCE AN APPLICATION PROGRAM
YES
NO
WAS AN ABORT SEQUENCE SENT
YES
NO
IS THE LINE SENDING
YES
NO
IS THIS THE LAST BUFFER OF A MESSAGE
NO
YES
IS THERE A CUTOFF ERROR
YES
NO
IS EOT RECEIVED
YES
NO
IS THERE A TEXT ERROR
NO
YES
A3
EOBSND
IS THERE A TEXT ERROR
NO
YES
DOES CURRENT BUFFER CONTAIN THE ERROR
NO
YES
BT-4 A1
CKRETRY
CHECK THE 'NO RETRY' FLAG
IS THIS A BAD RETURN
YES
NO
K3
SET UP TO TPOST THE BUFFERS TO THE BUFFER RETURN QCB
IS THE ERB AVAILABLE
YES
NO
EXIT TO DSPCHAIN
ARE THERE OTHER LINE ERRORS
YES
NO
BT2 A1
A3
EOBERR
IS THIS AN INITIATE MODE MESSAGE
YES
NO
BT-4 A1
CKRETRY
CHECK THE 'NO RETRY' FLAG
RCVTRY
IS THIS A BAD RETURN
YES
NO
CAN THE ERB BE TPOSTED
YES
NO
REMOVE THE ERB FROM THE BUFFER RETURN QCB OR THE READY QUEUE
RECALL
SET UP TO RECALL THE LAST EOB
BT1 K3
EOBTAA
GET THE ADDRESS OF THE STARTMH STCB
IS THE 'NO RETRY' FLAG SET
NO
YES
K3
BT2 F2
SET THE 'NAK' BIT
IS THE DESTINATION THE BUFFER RETURN QCB
YES
NO
IS THE MESSAGE LOST
YES
NO
CLEAR THE MULTIPLE-BUFFER-HEADER FIELD
BUILD A CCW IN THE BUFFER
IS THERE A PREVIOUS EOB
NO
YES
SET UP TO TPOST THE ERB TO IEDQFA FOR RECALL
EXIT TO DSPBYPAS
SET THE 'ABORT' FLAG
SET UP TO TPOST THE BUFFER TO THE ACTIVATE QCB
EXIT TO DSPCHAIN
SET THE 'SPECIAL' FLAG

Chart BT-2 EOB/ETB HANDLING SUBTASK

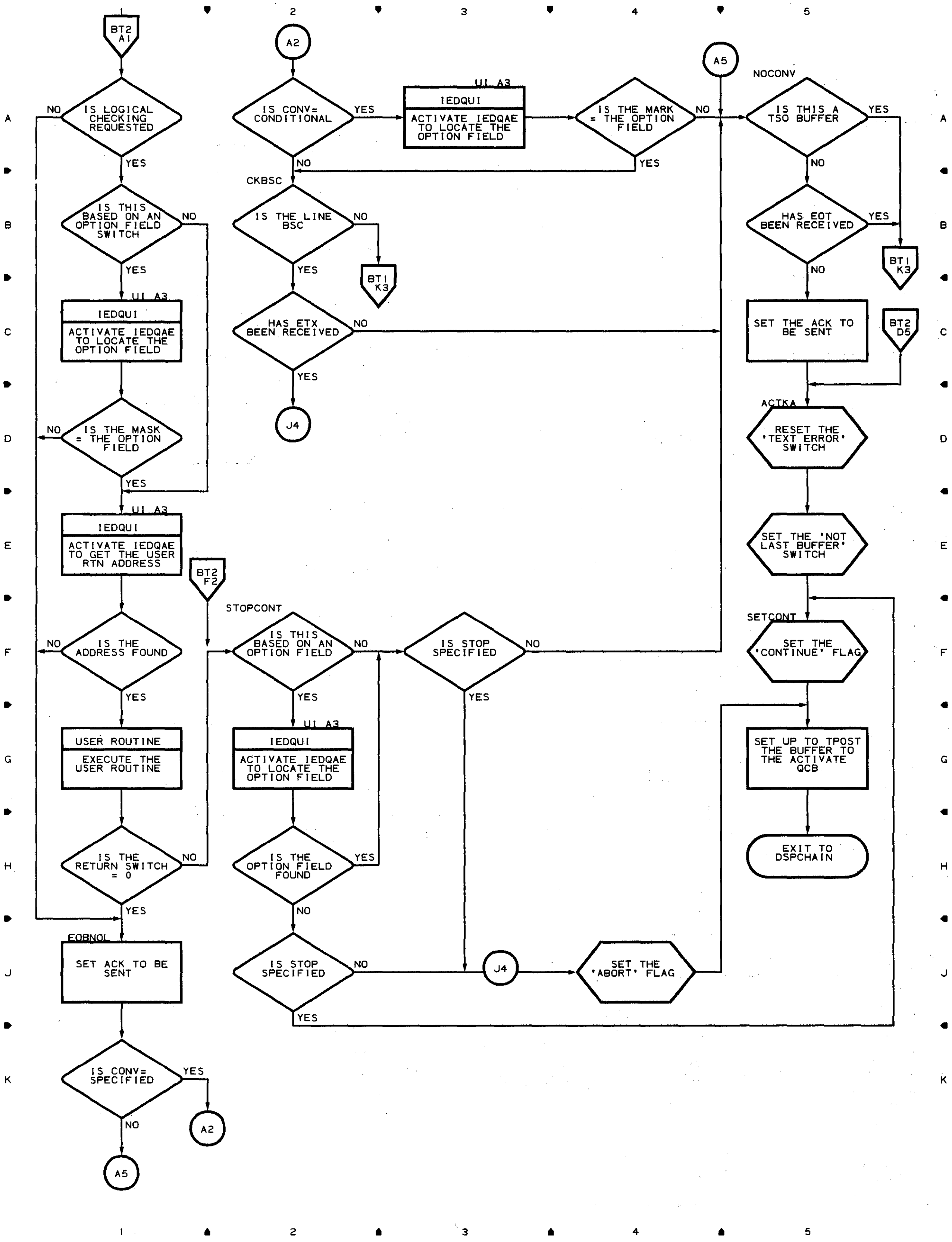


Chart BT-3 EOB/ETB HANDLING SUBTASK

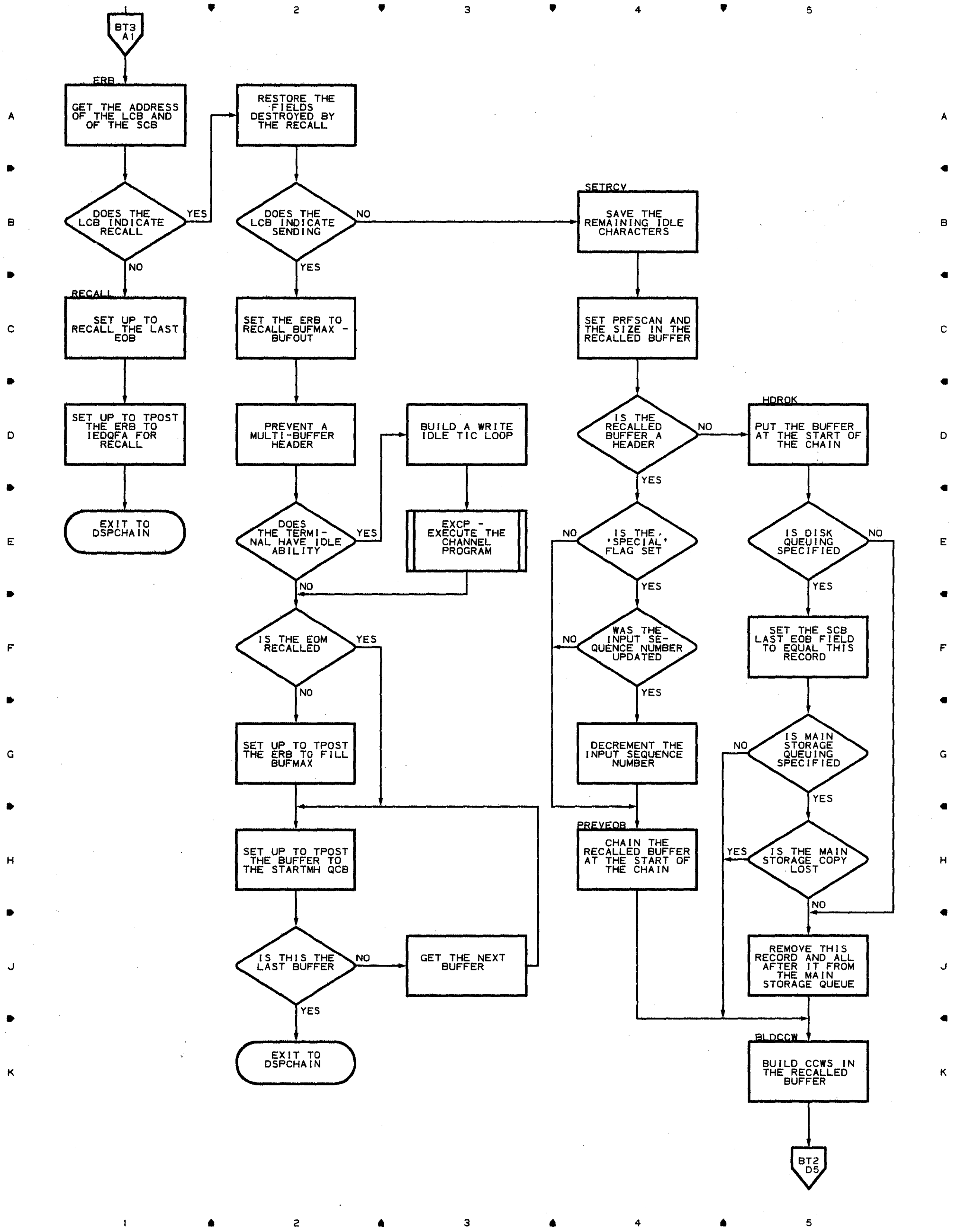

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
BT3 A1
ERB
GET THE ADDRESS OF THE LCB AND OF THE SCB
DOES THE LCB INDICATE RECALL
YES
NO
RECALL
SET UP TO RECALL THE LAST EOB
SET UP TO TPOST THE ERB TO IEDQFA FOR RECALL
EXIT TO DSPCHAIN
RESTORE THE FIELDS DESTROYED BY THE RECALL
DOES THE LCB INDICATE SENDING
NO
YES
SET THE ERB TO RECALL BUFMAX - BUFOUT
PREVENT A MULTI-BUFFER HEADER
DOES THE TERMINAL HAVE IDLE ABILITY
YES
NO
BUILD A WRITE IDLE TIC LOOP
EXCP - EXECUTE THE CHANNEL PROGRAM
IS THE EOM RECALLED
YES
NO
SET UP TO TPOST THE ERB TO FILL BUFMAX
SET UP TO TPOST THE BUFFER TO THE STARTMH QCB
IS THIS THE LAST BUFFER
NO
YES
GET THE NEXT BUFFER
EXIT TO DSPCHAIN
SETRCV
SAVE THE REMAINING IDLE CHARACTERS
SET PRFSCAN AND THE SIZE IN THE RECALLED BUFFER
IS THE RECALLED BUFFER A HEADER
NO
YES
IS THE 'SPECIAL' FLAG SET
NO
YES
WAS THE INPUT SEQUENCE NUMBER UPDATED
NO
YES
DECREMENT THE INPUT SEQUENCE NUMBER
PREVEOB
CHAIN THE RECALLED BUFFER AT THE START OF THE CHAIN
HDROK
PUT THE BUFFER AT THE START OF THE CHAIN
IS DISK QUEUING SPECIFIED
NO
YES
SET THE SCB LAST EOB FIELD TO EQUAL THIS RECORD
IS MAIN STORAGE QUEUING SPECIFIED
NO
YES
IS THE MAIN STORAGE COPY LOST
YES
NO
REMOVE THIS RECORD AND ALL AFTER IT FROM THE MAIN STORAGE QUEUE
BLDCCW
BUILD CCWS IN THE RECALLED BUFFER
BT2 D5

Chart BT-4 EOB/ETB HANDLING SUBTASK

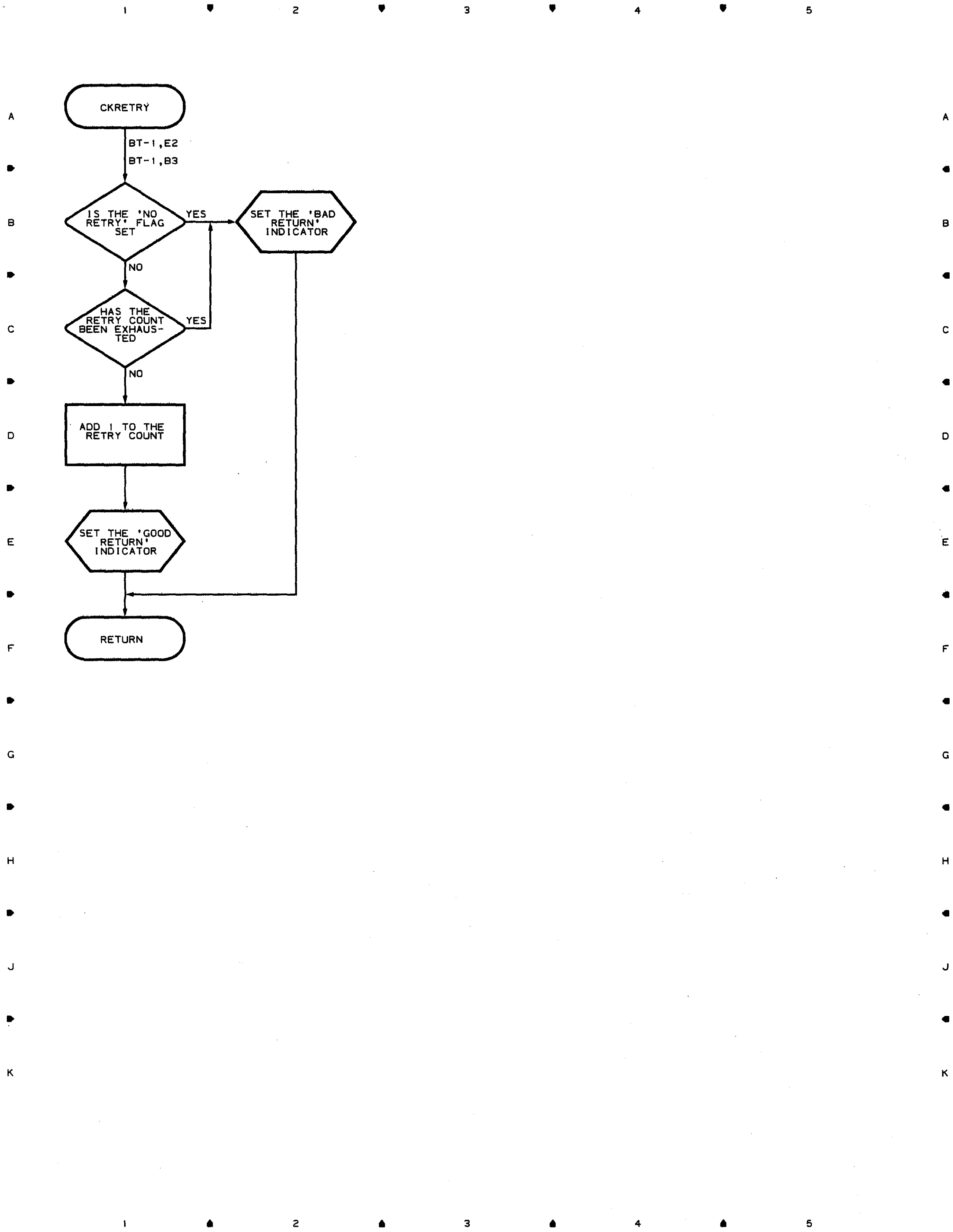

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
CKRETRY
BT-1,E2
BT-1,B3
IS THE 'NO RETRY' FLAG SET
YES
SET THE 'BAD RETURN' INDICATOR
NO
HAS THE RETRY COUNT BEEN EXHAUS-TED
YES
NO
ADD 1 TO THE RETRY COUNT
SET THE 'GOOD RETURN' INDICATOR
RETURN

Chart BW UNIT REQUEST ROUTINE

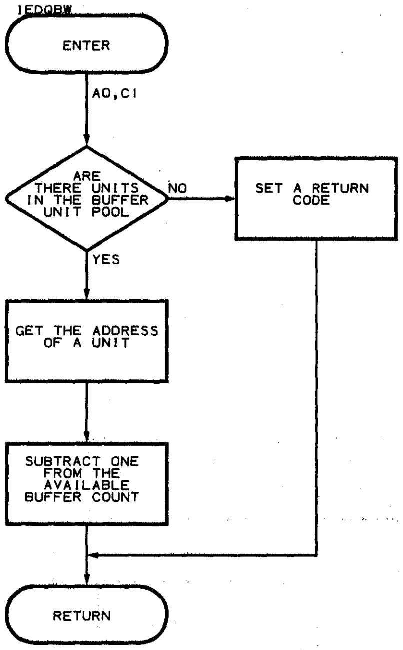

IEDQBW
ENTER
AO,C1
ARE THERE UNITS IN THE BUFFER UNIT POOL
NO
SET A RETURN CODE
YES
GET THE ADDRESS OF A UNIT
SUBTRACT ONE FROM THE AVAILABLE BUFFER COUNT
RETURN

Chart BX LOG SEGMENT ROUTINE

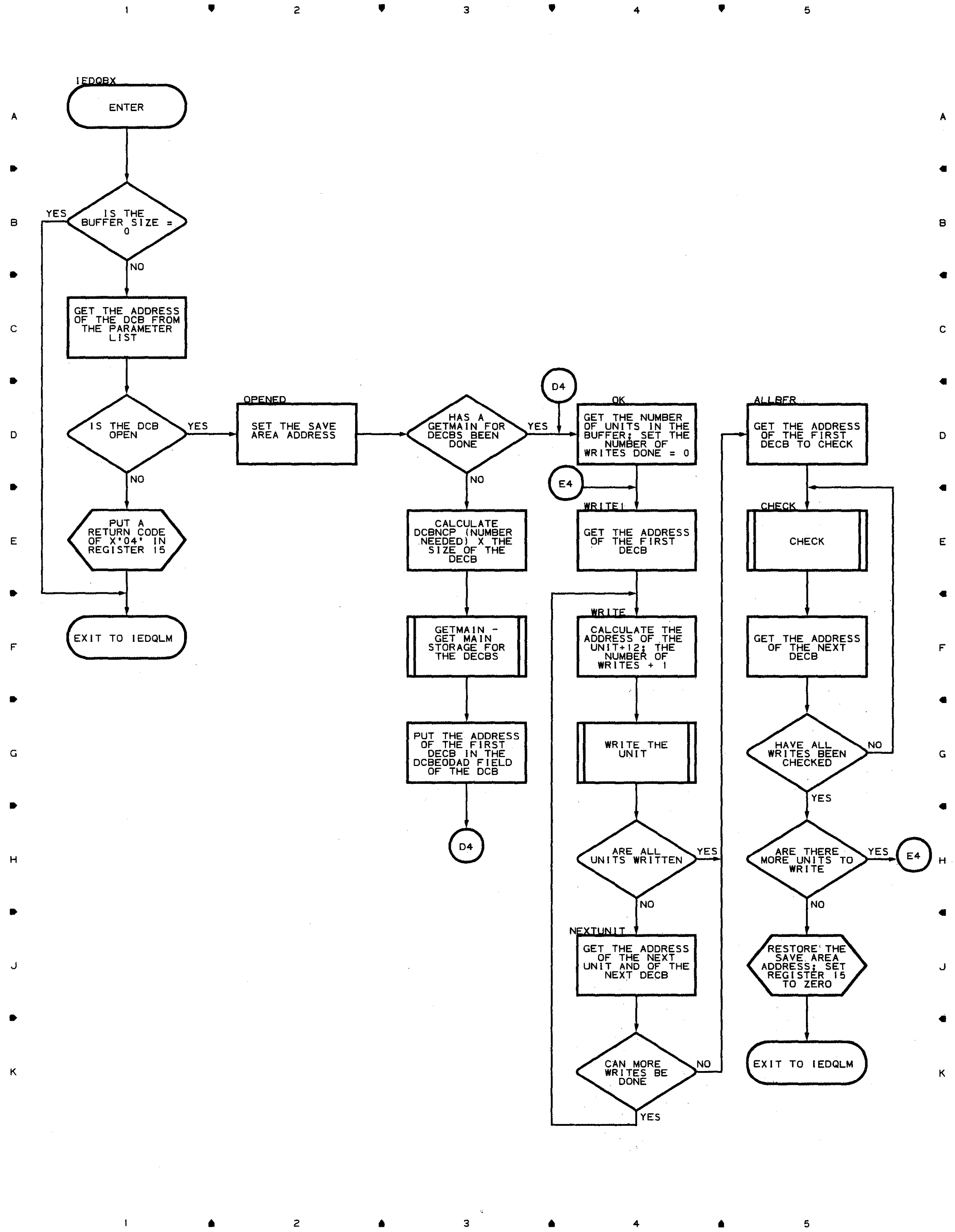
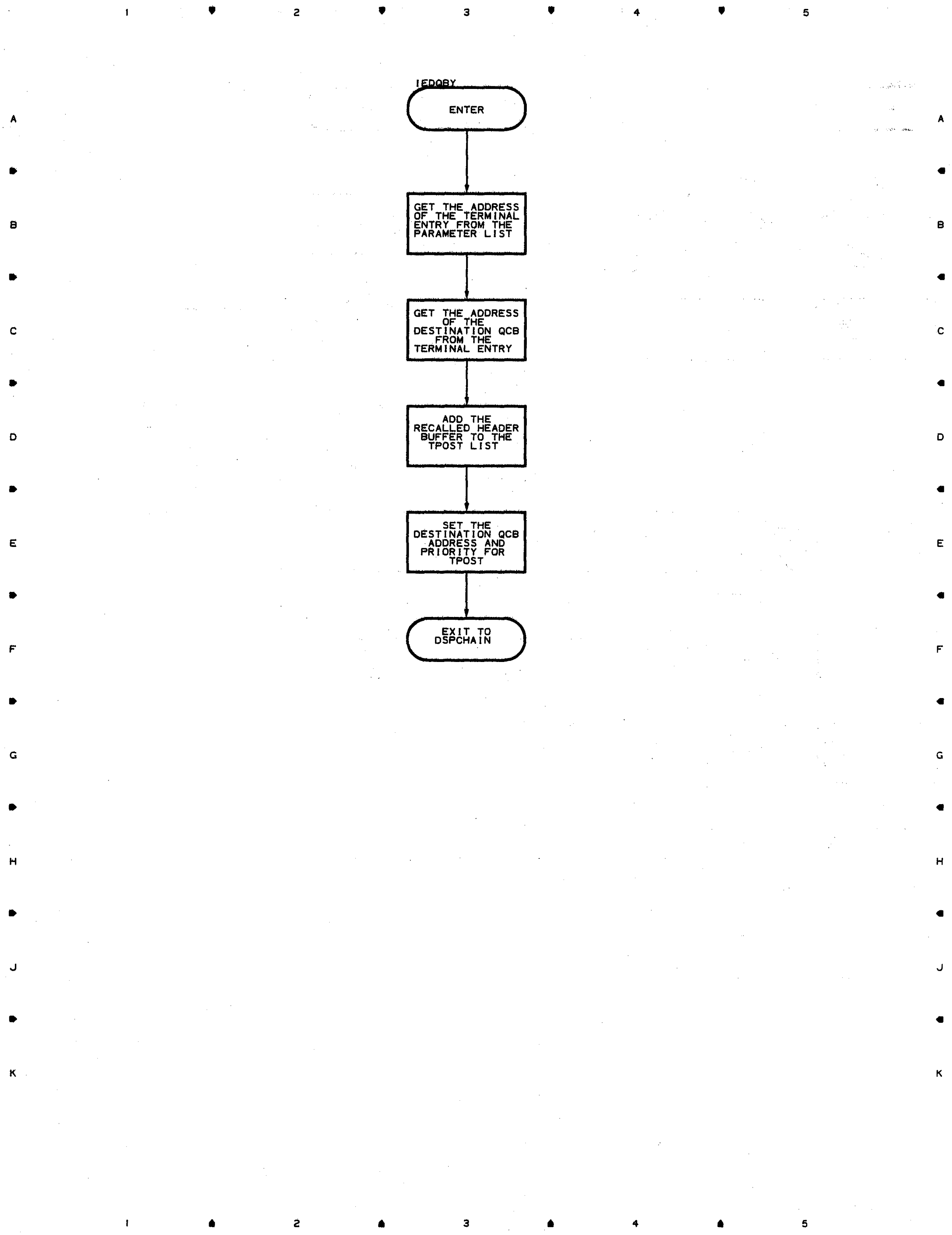
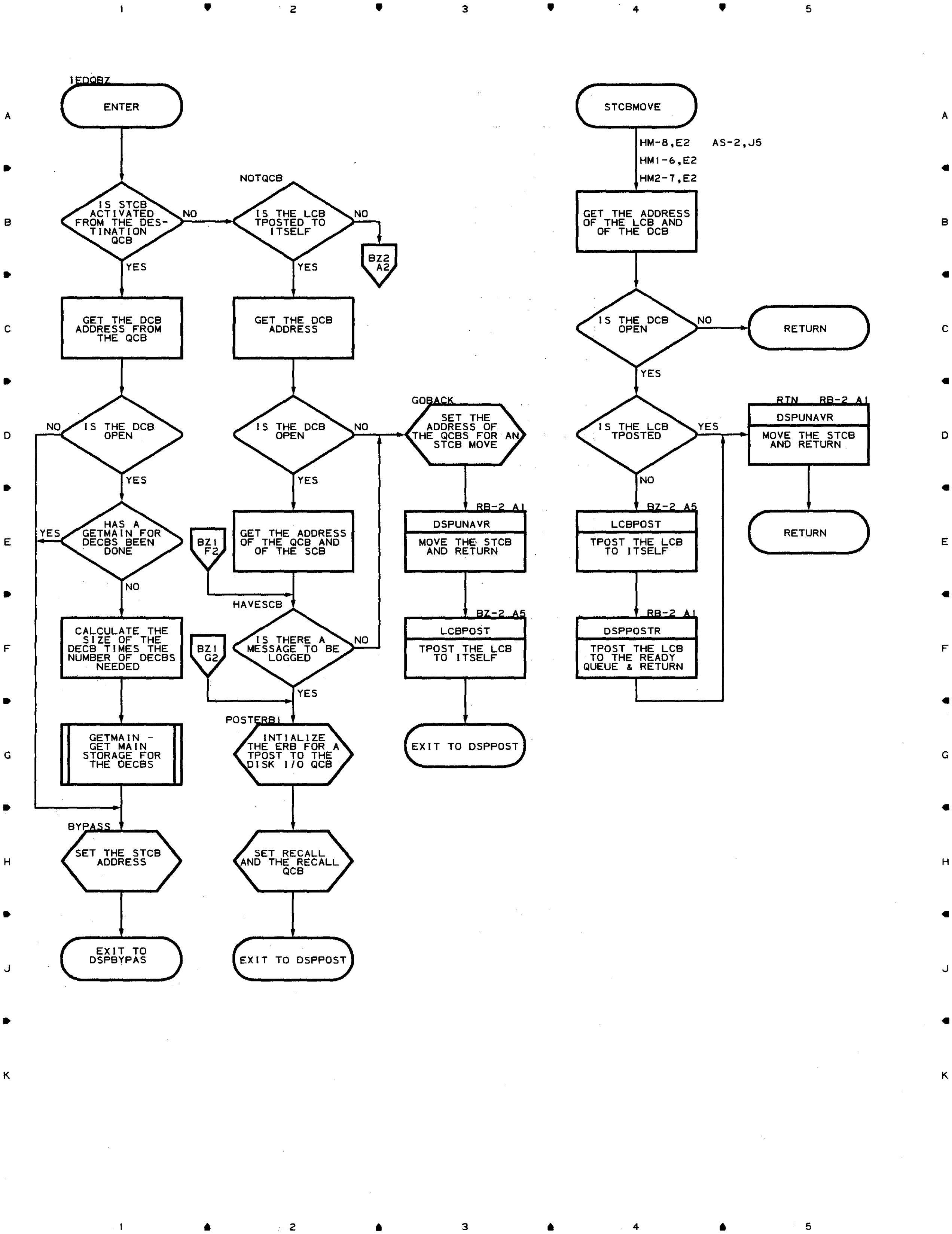

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQBX
ENTER
IS THE BUFFER SIZE = 0
YES
NO
GET THE ADDRESS OF THE DCB FROM THE PARAMETER LIST
IS THE DCB OPEN
YES
NO
PUT A RETURN CODE OF X'04' IN REGISTER 15
EXIT TO IEDQLM
OPENED
SET THE SAVE AREA ADDRESS
HAS A GETMAIN FOR DECBS BEEN DONE
YES
NO
CALCULATE DCBNCP (NUMBER NEEDED) X THE SIZE OF THE DECB
GETMAIN - GET MAIN STORAGE FOR THE DECBS
PUT THE ADDRESS OF THE FIRST DECB IN THE DCBEODAD FIELD OF THE DCB
D4
D4
OK
GET THE NUMBER OF UNITS IN THE BUFFER; SET THE NUMBER OF WRITES DONE = 0
E4
WRITE1
GET THE ADDRESS OF THE FIRST DECB
WRITE
CALCULATE THE ADDRESS OF THE UNIT+12; THE NUMBER OF WRITES + 1
WRITE THE UNIT
ARE ALL UNITS WRITTEN
YES
NO
NEXTUNIT
GET THE ADDRESS OF THE NEXT UNIT AND OF THE NEXT DECB
CAN MORE WRITES BE DONE
NO
YES
ALLBFR
GET THE ADDRESS OF THE FIRST DECB TO CHECK
CHECK
CHECK
GET THE ADDRESS OF THE NEXT DECB
HAVE ALL WRITES BEEN CHECKED
NO
YES
ARE THERE MORE UNITS TO WRITE
YES
E4
NO
RESTORE THE SAVE AREA ADDRESS; SET REGISTER 15 TO ZERO
EXIT TO IEDQLM

Chart BY LOG MESSAGE ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQBY
ENTER
GET THE ADDRESS OF THE TERMINAL ENTRY FROM THE PARAMETER LIST
GET THE ADDRESS OF THE DESTINATION QCB FROM THE TERMINAL ENTRY
ADD THE RECALLED HEADER BUFFER TO THE TPOST LIST
SET THE DESTINATION QCB ADDRESS AND PRIORITY FOR TPOST
EXIT TO DSPCHAIN

Chart BZ-1 LOG SCHEDULER ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQBZ
ENTER
IS STCB ACTIVATED FROM THE DES-TINATION QCB
NO
YES
GET THE DCB ADDRESS FROM THE QCB
IS THE DCB OPEN
NO
YES
HAS A GETMAIN FOR DECBS BEEN DONE
YES
NO
CALCULATE THE SIZE OF THE DECB TIMES THE NUMBER OF DECBS NEEDED
GETMAIN - GET MAIN STORAGE FOR THE DECBS
BYPASS
SET THE STCB ADDRESS
EXIT TO DSPBYPAS
NOTQCB
IS THE LCB TPOSTED TO ITSELF
NO
BZ2 A2
YES
GET THE DCB ADDRESS
IS THE DCB OPEN
NO
YES
GET THE ADDRESS OF THE QCB AND OF THE SCB
BZ1 F2
HAVESCB
IS THERE A MESSAGE TO BE LOGGED
NO
YES
BZ1 G2
POSTERB1
INTIALIZE THE ERB FOR A TPOST TO THE DISK I/O QCB
SET RECALL AND THE RECALL QCB
EXIT TO DSPPOST
GOBACK
SET THE ADDRESS OF THE QCBS FOR AN STCB MOVE
RB-2 A1
DSPUNAVR
MOVE THE STCB AND RETURN
BZ-2 A5
LCBPOST
TPOST THE LCB TO ITSELF
EXIT TO DSPPOST
STCBMOVE
HM-8,E2
AS-2,J5
HM1-6,E2
HM2-7,E2
GET THE ADDRESS OF THE LCB AND OF THE DCB
IS THE DCB OPEN
NO
RETURN
YES
IS THE LCB TPOSTED
YES
NO
BZ-2 A5
LCBPOST
TPOST THE LCB TO ITSELF
RB-2 A1
DSPPOSTR
TPOST THE LCB TO THE READY QUEUE & RETURN
RTN RB-2 A1
DSPUNAVR
MOVE THE STCB AND RETURN
RETURN
1
2
3
4
5

Chart BZ-2 LOG SCHEDULER ROUTINE

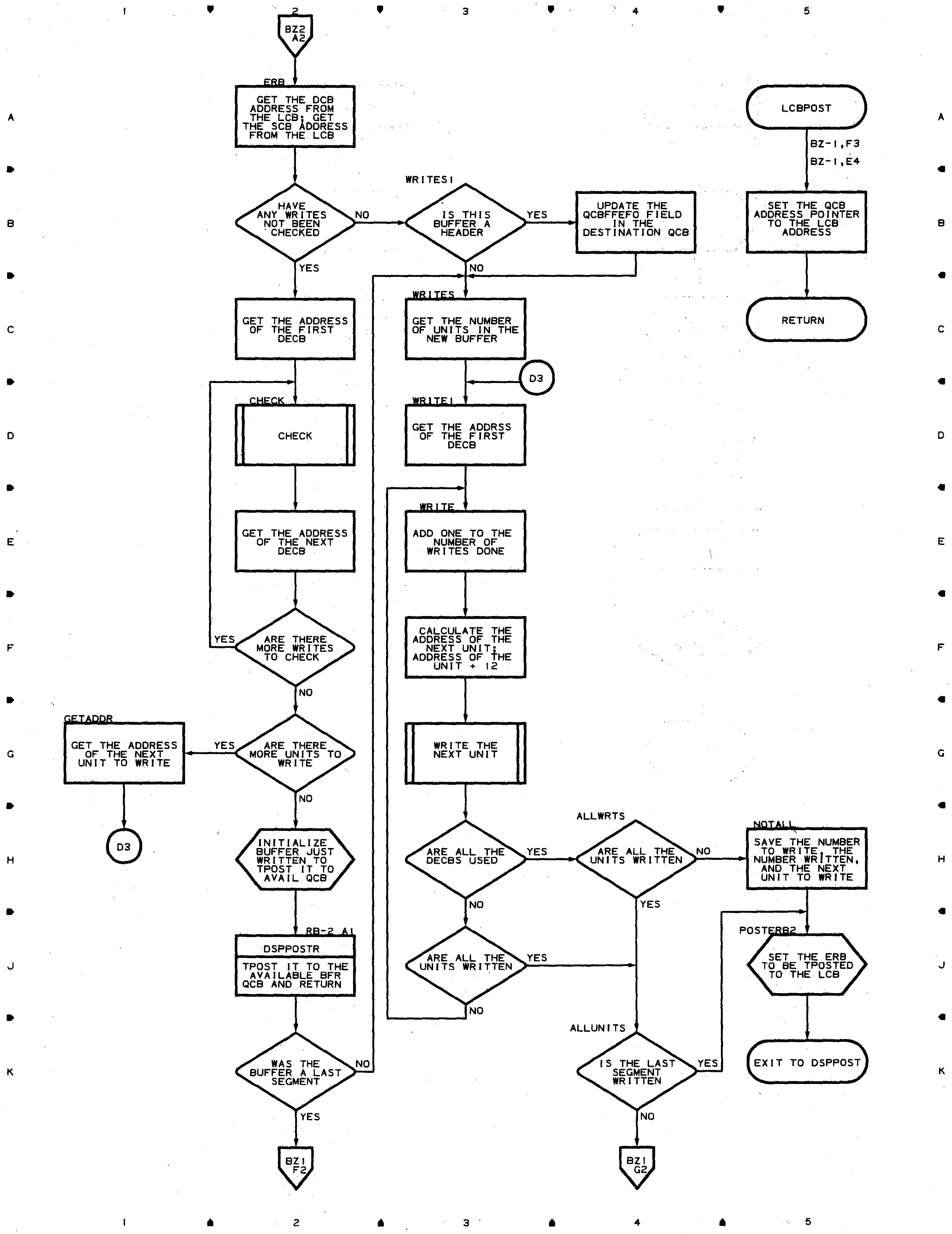

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
BZ2 A2
ERB
GET THE DCB ADDRESS FROM THE LCB; GET THE SCB ADDRESS FROM THE LCB
HAVE ANY WRITES NOT BEEN CHECKED
NO
YES
WRITES1
IS THIS BUFFER A HEADER
YES
NO
UPDATE THE QCBFFEFO FIELD IN THE DESTINATION QCB
LCBPOST
BZ-1,F3
BZ-1,E4
SET THE QCB ADDRESS POINTER TO THE LCB ADDRESS
RETURN
GET THE ADDRESS OF THE FIRST DECB
WRITES
GET THE NUMBER OF UNITS IN THE NEW BUFFER
D3
CHECK
CHECK
WRITE1
GET THE ADDRSS OF THE FIRST DECB
GET THE ADDRESS OF THE NEXT DECB
WRITE
ADD ONE TO THE NUMBER OF WRITES DONE
YES
ARE THERE MORE WRITES TO CHECK
NO
CALCULATE THE ADDRESS OF THE NEXT UNIT; ADDRESS OF THE UNIT + 12
GETADDR
GET THE ADDRESS OF THE NEXT UNIT TO WRITE
YES
ARE THERE MORE UNITS TO WRITE
NO
WRITE THE NEXT UNIT
D3
INITIALIZE BUFFER JUST WRITTEN TO TPOST IT TO AVAIL QCB
ARE ALL THE DECBS USED
YES
NO
ALLWRTS
ARE ALL THE UNITS WRITTEN
NO
YES
NOTALL
SAVE THE NUMBER TO WRITE, THE NUMBER WRITTEN, AND THE NEXT UNIT TO WRITE
RB-2 A1
DSPPOSTR
TPOST IT TO THE AVAILABLE BFR QCB AND RETURN
ARE ALL THE UNITS WRITTEN
YES
NO
POSTERB2
SET THE ERB TO BE TPOSTED TO THE LCB
ALLUNITS
WAS THE BUFFER A LAST SEGMENT
NO
YES
IS THE LAST SEGMENT WRITTEN
YES
NO
EXIT TO DSPPOST
BZ1 F2
BZ1 G2

Chart CA RESIDENT OPERATOR CONTROL MODULE

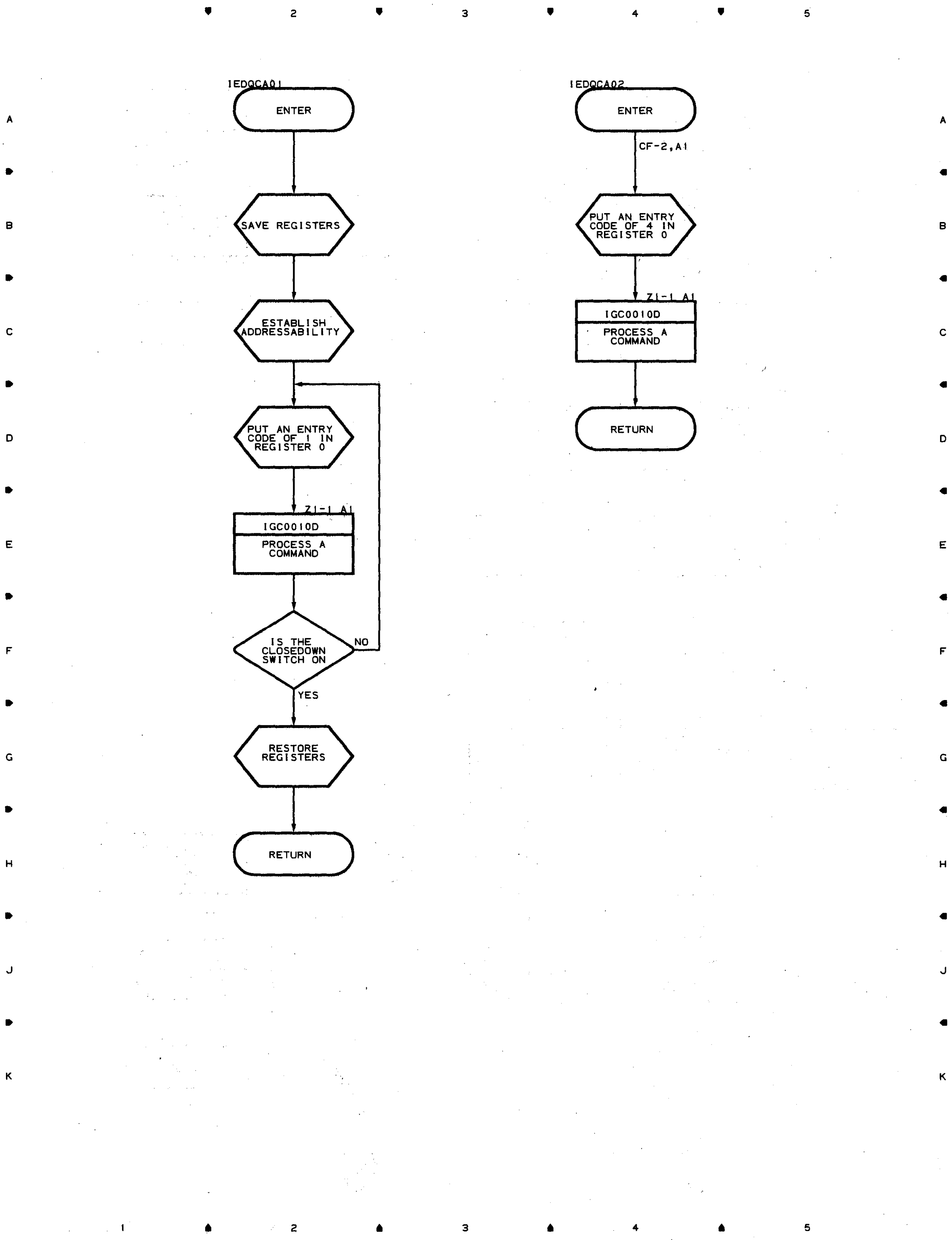


Chart CF-1 MODIFY OPTIONS ROUTINE

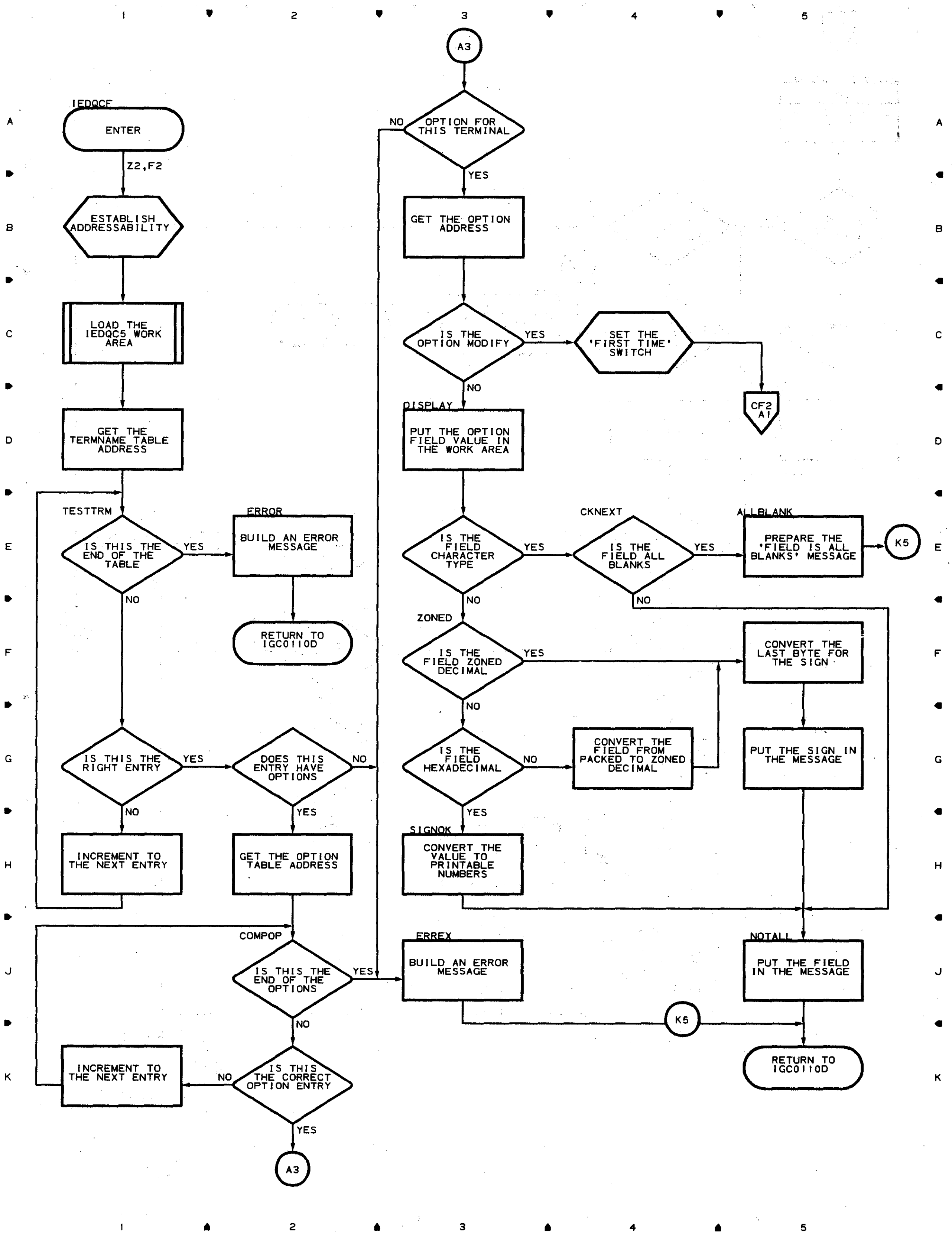

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
A3
IEDQCF
ENTER
Z2,F2
ESTABLISH ADDRESSABILITY
LOAD THE IEDQC5 WORK AREA
GET THE TERMNAME TABLE ADDRESS
TESTTRM
IS THIS THE END OF THE TABLE
YES
NO
ERROR
BUILD AN ERROR MESSAGE
RETURN TO IGC0110D
IS THIS THE RIGHT ENTRY
YES
NO
DOES THIS ENTRY HAVE OPTIONS
NO
YES
INCREMENT TO THE NEXT ENTRY
GET THE OPTION TABLE ADDRESS
COMPOP
IS THIS THE END OF THE OPTIONS
YES
NO
IS THIS THE CORRECT OPTION ENTRY
NO
YES
INCREMENT TO THE NEXT ENTRY
A3
NO
OPTION FOR THIS TERMINAL
YES
GET THE OPTION ADDRESS
IS THE OPTION MODIFY
YES
SET THE 'FIRST TIME' SWITCH
CF2 A1
NO
DISPLAY
PUT THE OPTION FIELD VALUE IN THE WORK AREA
IS THE FIELD CHARACTER TYPE
YES
NO
CKNEXT
IS THE FIELD ALL BLANKS
YES
NO
ALLBLANK
PREPARE THE 'FIELD IS ALL BLANKS' MESSAGE
K5
ZONED
IS THE FIELD ZONED DECIMAL
YES
NO
CONVERT THE LAST BYTE FOR THE SIGN
IS THE FIELD HEXADECIMAL
NO
YES
CONVERT THE FIELD FROM PACKED TO ZONED DECIMAL
PUT THE SIGN IN THE MESSAGE
SIGNOK
CONVERT THE VALUE TO PRINTABLE NUMBERS
ERREX
BUILD AN ERROR MESSAGE
NOTALL
PUT THE FIELD IN THE MESSAGE
K5
RETURN TO IGC0110D

Chart CF-2 MODIFY OPTIONS ROUTINE

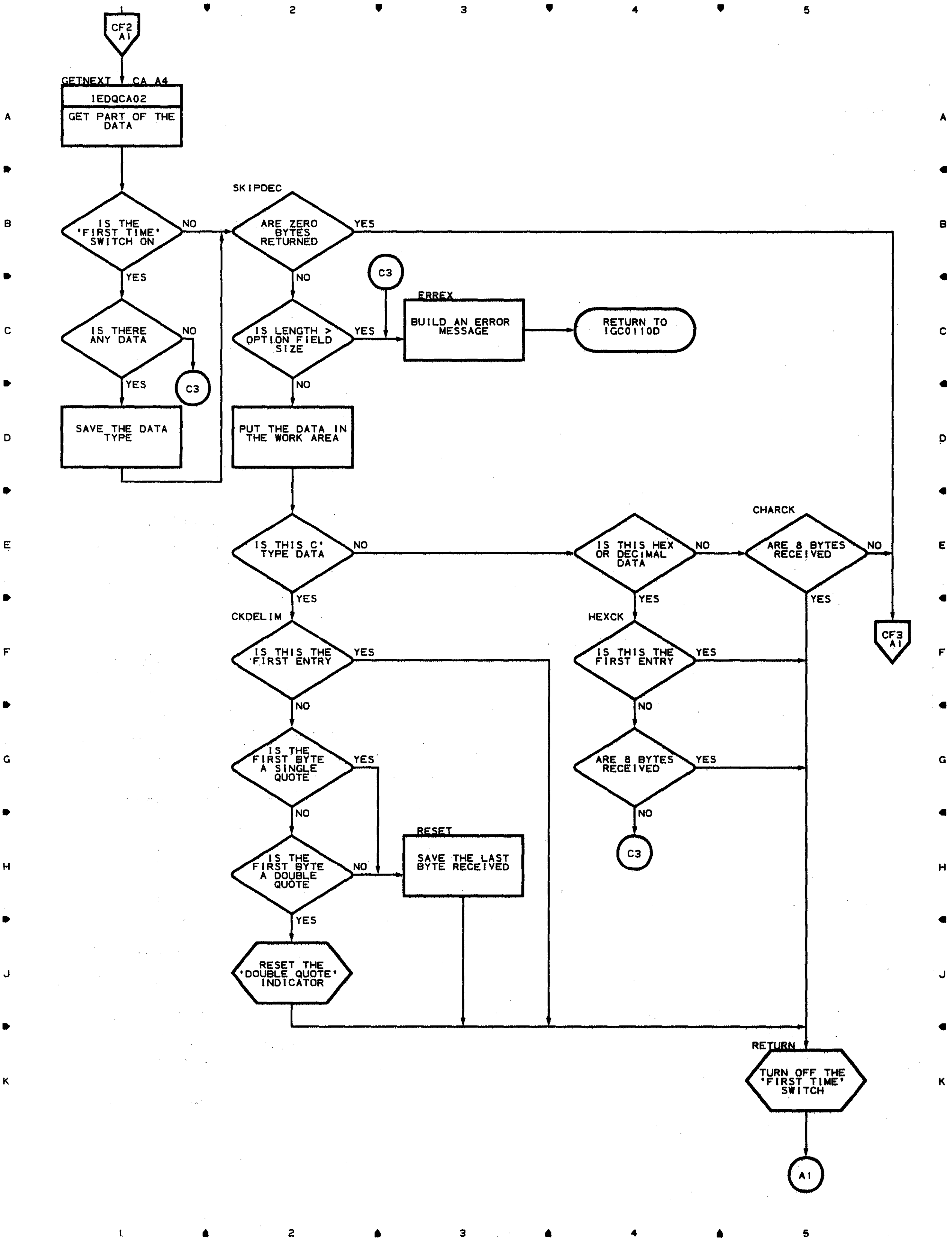
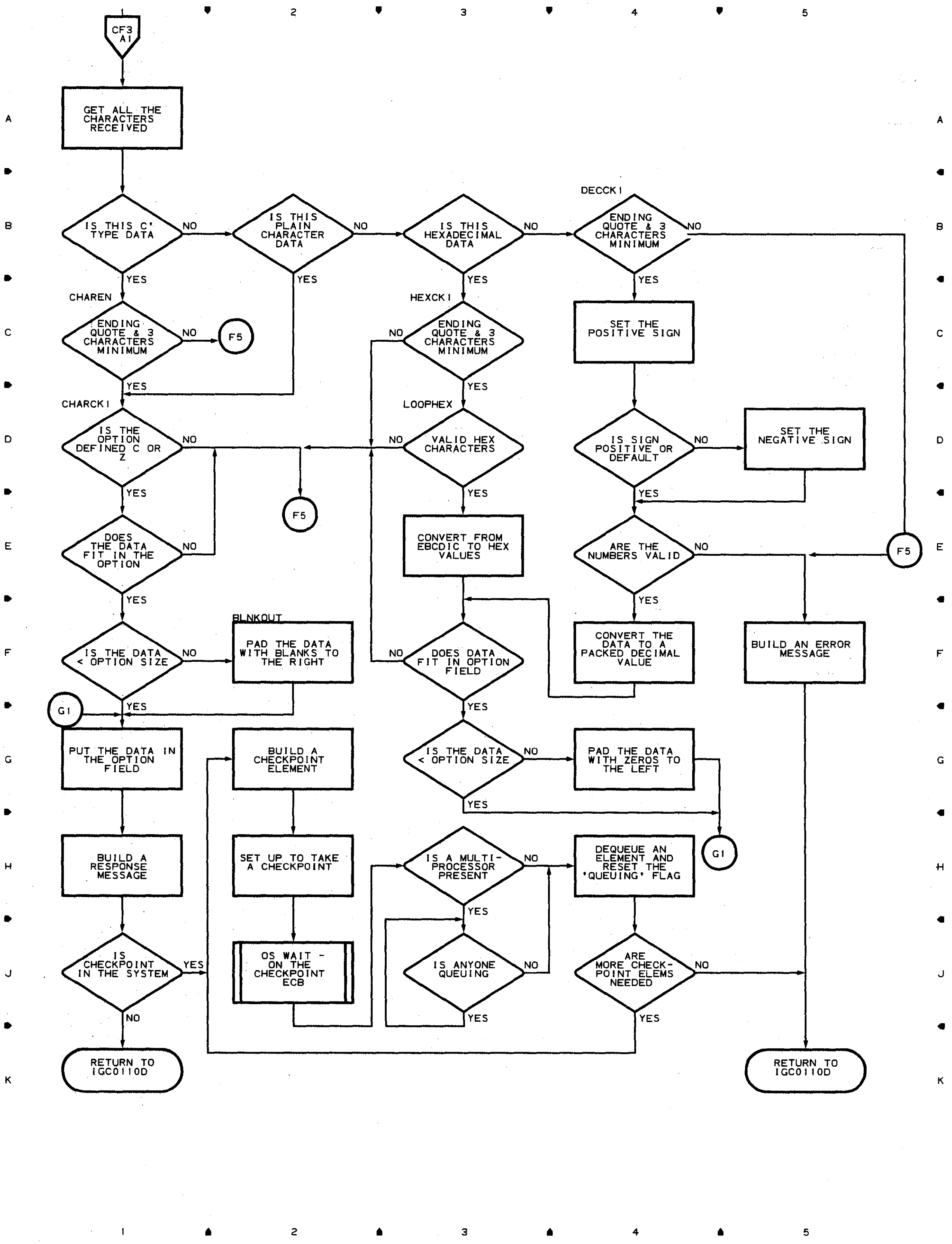
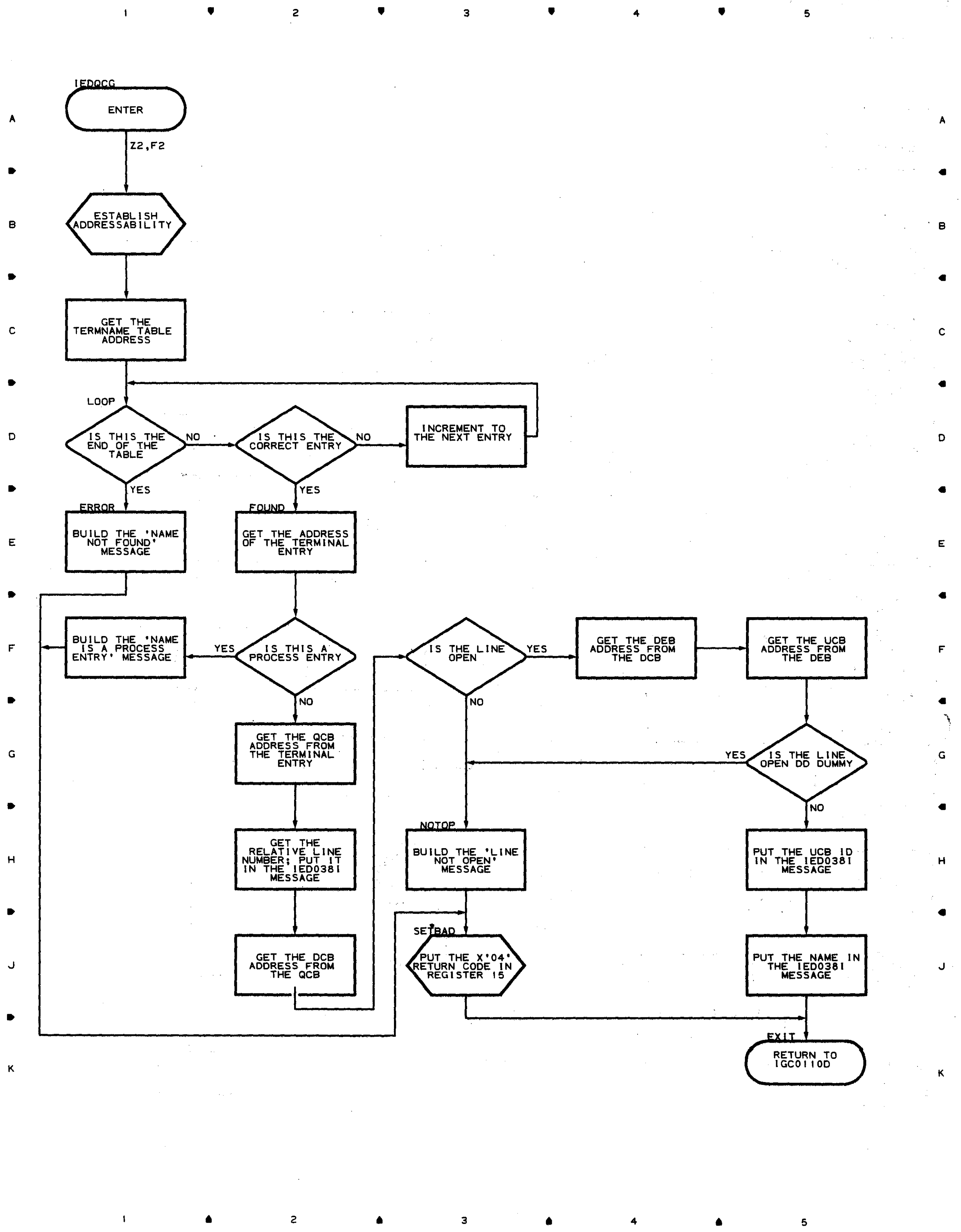


Chart CF-3 MODIFY OPTIONS ROUTINE

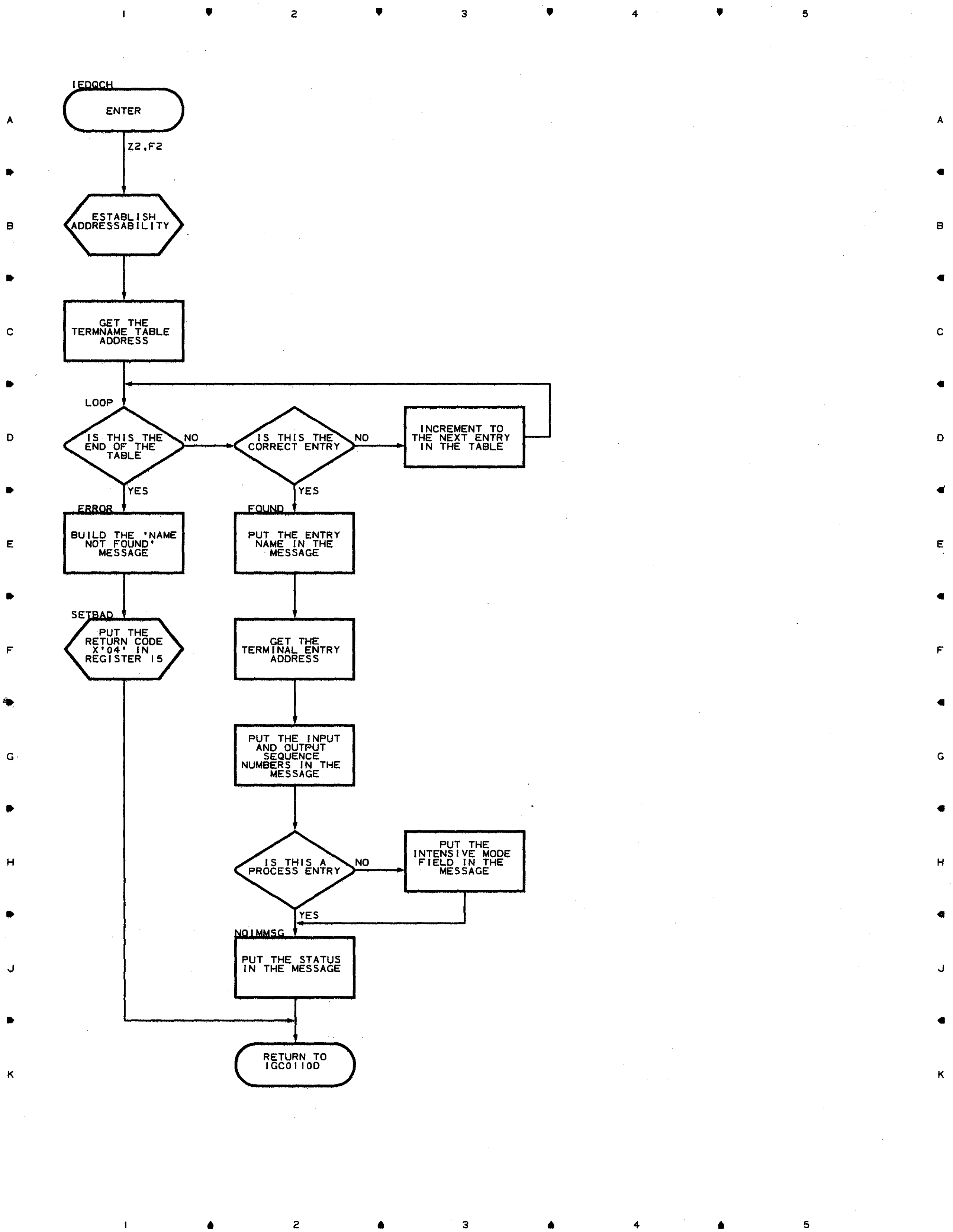


1
2
3
4
5
CF3 A1
A
B
C
D
E
F
G
H
J
K
GET ALL THE CHARACTERS RECEIVED
IS THIS C' TYPE DATA
NO
YES
IS THIS PLAIN CHARACTER DATA
NO
YES
IS THIS HEXADECIMAL DATA
NO
YES
DECCK1
ENDING QUOTE & 3 CHARACTERS MINIMUM
NO
YES
CHAREN
ENDING QUOTE & 3 CHARACTERS MINIMUM
NO
F5
YES
HEXCK1
ENDING QUOTE & 3 CHARACTERS MINIMUM
NO
YES
SET THE POSITIVE SIGN
CHARCK1
IS THE OPTION DEFINED C OR Z
NO
YES
LOOPHEX
VALID HEX CHARACTERS
NO
YES
IS SIGN POSITIVE OR DEFAULT
NO
YES
SET THE NEGATIVE SIGN
F5
DOES THE DATA FIT IN THE OPTION
NO
YES
CONVERT FROM EBCDIC TO HEX VALUES
ARE THE NUMBERS VALID
NO
YES
F5
BLNKOUT
IS THE DATA < OPTION SIZE
NO
YES
PAD THE DATA WITH BLANKS TO THE RIGHT
DOES DATA FIT IN OPTION FIELD
NO
YES
CONVERT THE DATA TO A PACKED DECIMAL VALUE
BUILD AN ERROR MESSAGE
G1
PUT THE DATA IN THE OPTION FIELD
BUILD A CHECKPOINT ELEMENT
IS THE DATA < OPTION SIZE
NO
YES
PAD THE DATA WITH ZEROS TO THE LEFT
G1
BUILD A RESPONSE MESSAGE
SET UP TO TAKE A CHECKPOINT
IS A MULTI-PROCESSOR PRESENT
NO
YES
DEQUEUE AN ELEMENT AND RESET THE 'QUEUING' FLAG
IS CHECKPOINT IN THE SYSTEM
YES
NO
OS WAIT - ON THE CHECKPOINT ECB
IS ANYONE QUEUING
NO
YES
ARE MORE CHECK-POINT ELEMS NEEDED
NO
YES
RETURN TO IGC0110D
RETURN TO IGC0110D


1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQCG
ENTER
Z2,F2
ESTABLISH ADDRESSABILITY
GET THE TERMNAME TABLE ADDRESS
LOOP
IS THIS THE END OF THE TABLE
NO
IS THIS THE CORRECT ENTRY
NO
INCREMENT TO THE NEXT ENTRY
YES
ERROR
BUILD THE 'NAME NOT FOUND' MESSAGE
YES
FOUND
GET THE ADDRESS OF THE TERMINAL ENTRY
BUILD THE 'NAME IS A PROCESS ENTRY' MESSAGE
YES
IS THIS A PROCESS ENTRY
NO
GET THE QCB ADDRESS FROM THE TERMINAL ENTRY
GET THE RELATIVE LINE NUMBER; PUT IT IN THE IED038I MESSAGE
GET THE DCB ADDRESS FROM THE QCB
IS THE LINE OPEN
YES
NO
GET THE DEB ADDRESS FROM THE DCB
GET THE UCB ADDRESS FROM THE DEB
YES
IS THE LINE OPEN DD DUMMY
NO
NOTOP
BUILD THE 'LINE NOT OPEN' MESSAGE
PUT THE UCB ID IN THE IED038I MESSAGE
SETBAD
PUT THE X'04' RETURN CODE IN REGISTER 15
PUT THE NAME IN THE IED038I MESSAGE
EXIT
RETURN TO IGC0110D

]

Chart CH COPY TERMINAL INFORMATION ROUTINE



IEDQCH
ENTER
Z2,F2
ESTABLISH ADDRESSABILITY
GET THE TERMNAME TABLE ADDRESS
LOOP
IS THIS THE END OF THE TABLE
NO
IS THIS THE CORRECT ENTRY
NO
INCREMENT TO THE NEXT ENTRY IN THE TABLE
YES
YES
ERROR
BUILD THE 'NAME NOT FOUND' MESSAGE
FOUND
PUT THE ENTRY NAME IN THE MESSAGE
SETBAD
PUT THE RETURN CODE X'04' IN REGISTER 15
GET THE TERMINAL ENTRY ADDRESS
PUT THE INPUT AND OUTPUT SEQUENCE NUMBERS IN THE MESSAGE
IS THIS A PROCESS ENTRY
NO
PUT THE INTENSIVE MODE FIELD IN THE MESSAGE
YES
NOIMMSG
PUT THE STATUS IN THE MESSAGE
RETURN TO IGC0110D
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart CI COPY LCB INFORMATION ROUTINE

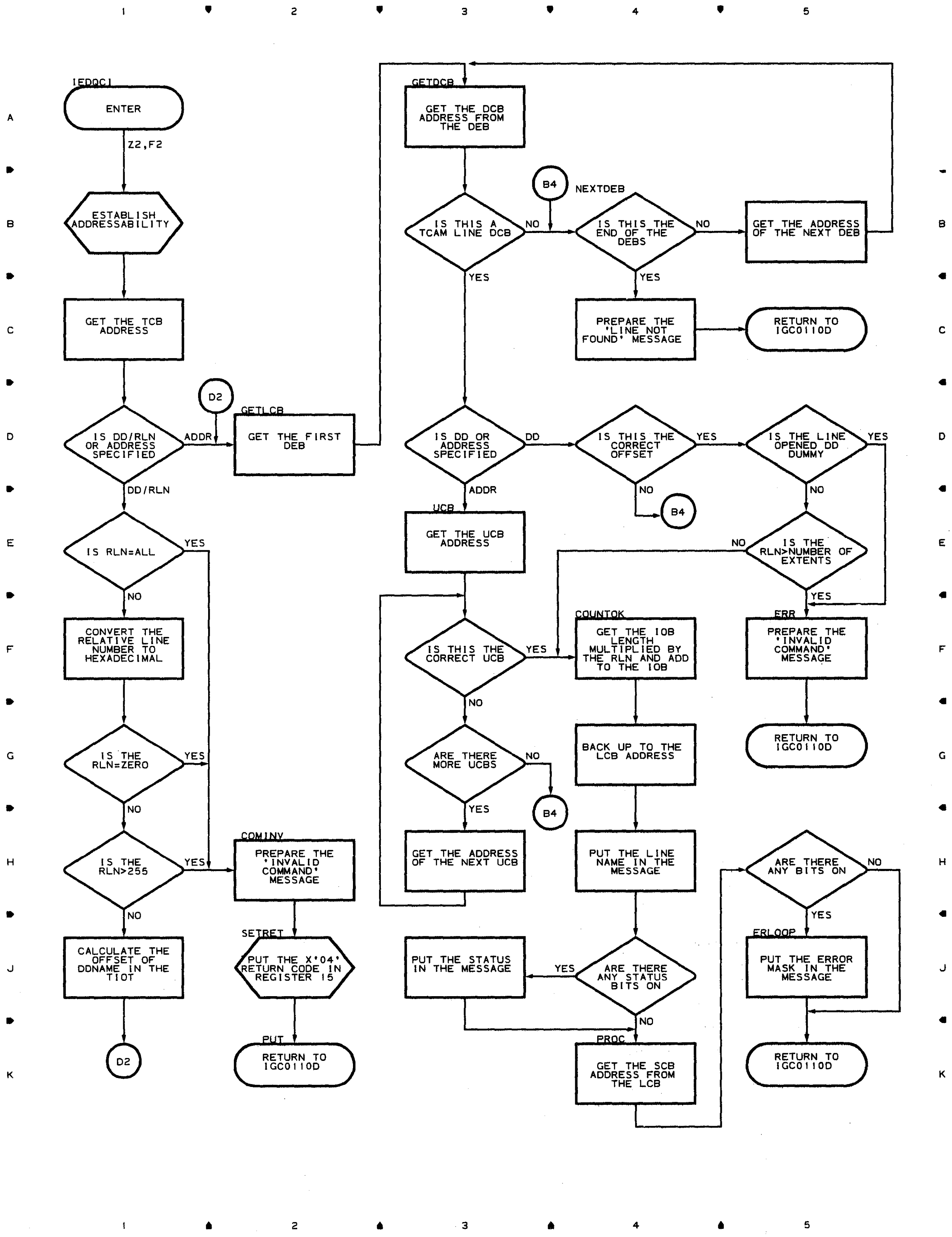
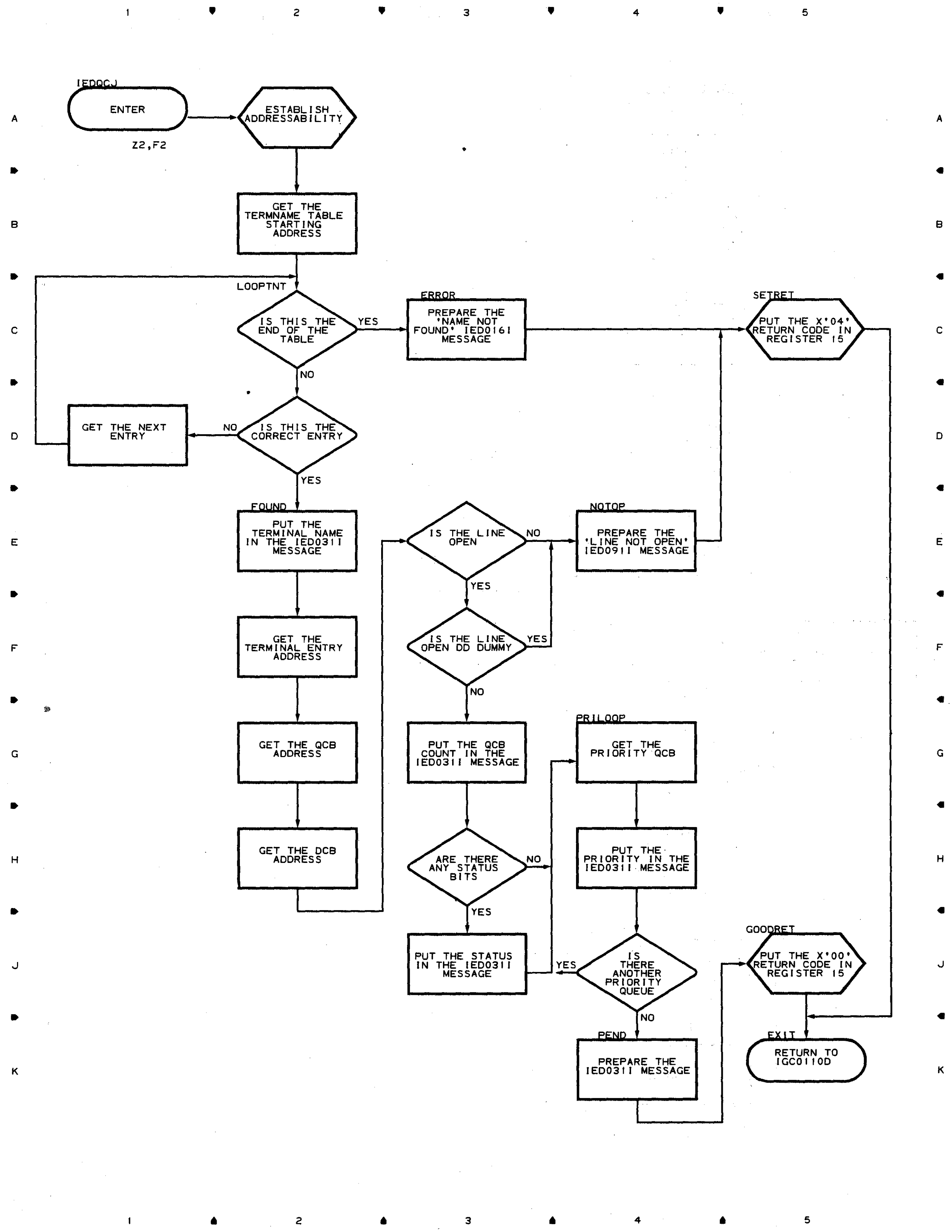

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQCI
ENTER
Z2,F2
ESTABLISH ADDRESSABILITY
GET THE TCB ADDRESS
D2
IS DD/RLN OR ADDRESS SPECIFIED
ADDR
DD/RLN
GETLCB
GET THE FIRST DEB
IS RLN=ALL
YES
NO
CONVERT THE RELATIVE LINE NUMBER TO HEXADECIMAL
IS THE RLN=ZERO
YES
NO
IS THE RLN>255
YES
NO
CALCULATE THE OFFSET OF DDNAME IN THE TIOT
D2
COMINV
PREPARE THE 'INVALID COMMAND' MESSAGE
SETRET
PUT THE X'04' RETURN CODE IN REGISTER 15
PUT
RETURN TO IGC0110D
GETDCB
GET THE DCB ADDRESS FROM THE DEB
IS THIS A TCAM LINE DCB
NO
YES
B4
NEXTDEB
IS THIS THE END OF THE DEBS
NO
YES
GET THE ADDRESS OF THE NEXT DEB
PREPARE THE 'LINE NOT FOUND' MESSAGE
RETURN TO IGC0110D
IS DD OR ADDRESS SPECIFIED
DD
ADDR
IS THIS THE CORRECT OFFSET
YES
NO
B4
IS THE LINE OPENED DD DUMMY
YES
NO
UCB
GET THE UCB ADDRESS
IS THE RLN>NUMBER OF EXTENTS
NO
YES
IS THIS THE CORRECT UCB
YES
NO
COUNTOK
GET THE IOB LENGTH MULTIPLIED BY THE RLN AND ADD TO THE IOB
ERR
PREPARE THE 'INVALID COMMAND' MESSAGE
RETURN TO IGC0110D
ARE THERE MORE UCBS
NO
YES
B4
BACK UP TO THE LCB ADDRESS
GET THE ADDRESS OF THE NEXT UCB
PUT THE LINE NAME IN THE MESSAGE
ARE THERE ANY BITS ON
NO
YES
ERLOOP
PUT THE ERROR MASK IN THE MESSAGE
PUT THE STATUS IN THE MESSAGE
ARE THERE ANY STATUS BITS ON
YES
NO
PROC
GET THE SCB ADDRESS FROM THE LCB
RETURN TO IGC0110D

Chart CJ COPY QCB INFORMATION ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQCJ
ENTER
Z2,F2
ESTABLISH ADDRESSABILITY
GET THE TERMNAME TABLE STARTING ADDRESS
LOOPTNT
IS THIS THE END OF THE TABLE
YES
NO
ERROR
PREPARE THE 'NAME NOT FOUND' IED016I MESSAGE
SETRET
PUT THE X'04' RETURN CODE IN REGISTER 15
GET THE NEXT ENTRY
NO
IS THIS THE CORRECT ENTRY
YES
FOUND
PUT THE TERMINAL NAME IN THE IED031I MESSAGE
IS THE LINE OPEN
NO
NOTOP
PREPARE THE 'LINE NOT OPEN' IED091I MESSAGE
YES
GET THE TERMINAL ENTRY ADDRESS
IS THE LINE OPEN DD DUMMY
YES
NO
GET THE QCB ADDRESS
PUT THE QCB COUNT IN THE IED031I MESSAGE
PRILOOP
GET THE PRIORITY QCB
GET THE DCB ADDRESS
ARE THERE ANY STATUS BITS
NO
PUT THE PRIORITY IN THE IED031I MESSAGE
YES
PUT THE STATUS IN THE IED031I MESSAGE
YES
IS THERE ANOTHER PRIORITY QUEUE
NO
GOODRET
PUT THE X'00' RETURN CODE IN REGISTER 15
PEND
PREPARE THE IED031I MESSAGE
EXIT
RETURN TO IGC0110D

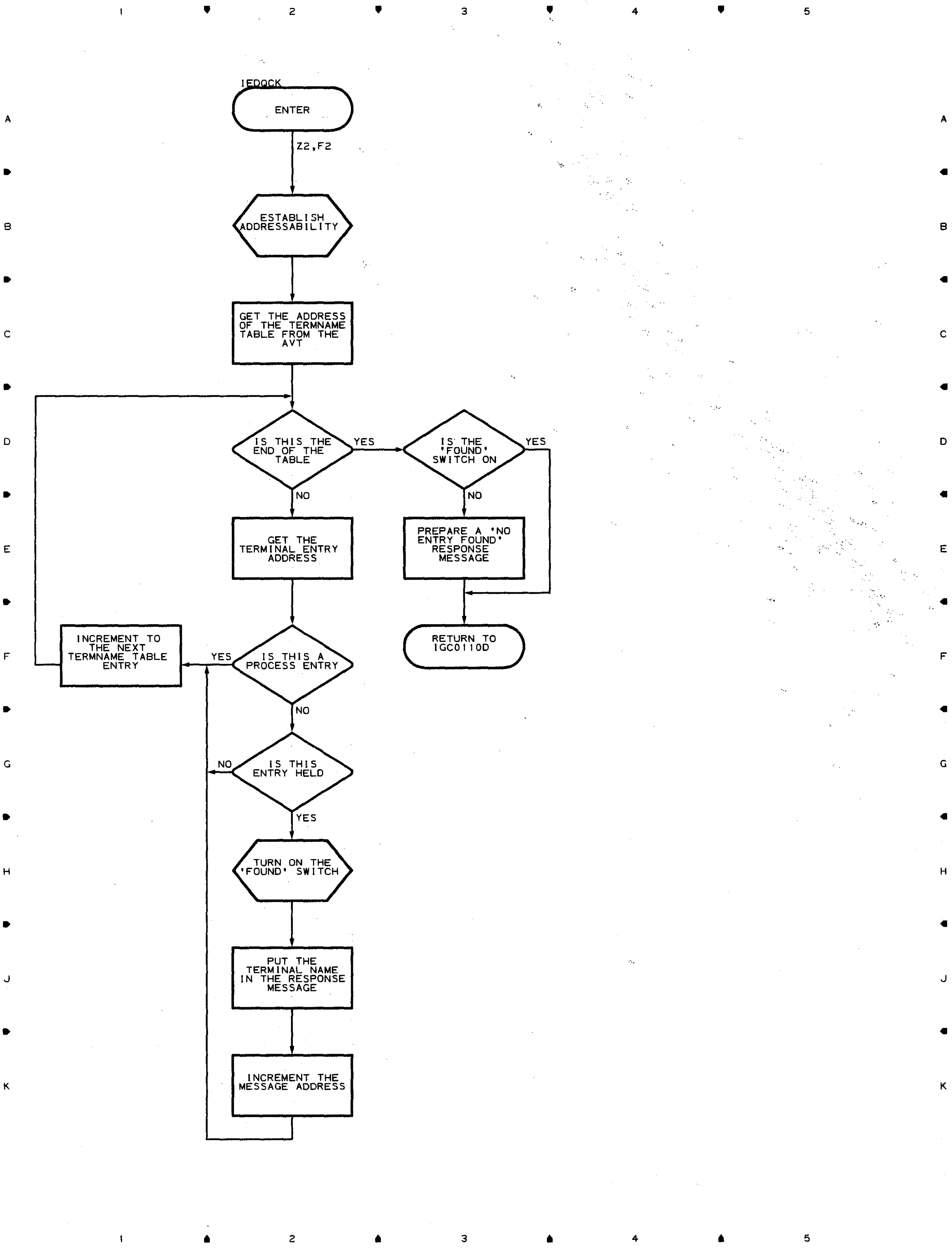

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQCK
ENTER
Z2,F2
ESTABLISH ADDRESSABILITY
GET THE ADDRESS OF THE TERMNAME TABLE FROM THE AVT
IS THIS THE END OF THE TABLE
YES
IS THE 'FOUND' SWITCH ON
YES
NO
NO
GET THE TERMINAL ENTRY ADDRESS
PREPARE A 'NO ENTRY FOUND' RESPONSE MESSAGE
RETURN TO IGC0110D
INCREMENT TO THE NEXT TERMNAME TABLE ENTRY
YES
IS THIS A PROCESS ENTRY
NO
NO
IS THIS ENTRY HELD
YES
TURN ON THE 'FOUND' SWITCH
PUT THE TERMINAL NAME IN THE RESPONSE MESSAGE
INCREMENT THE MESSAGE ADDRESS

Chart CL-1 COPY INVITATION LIST ENTRY ROUTINE

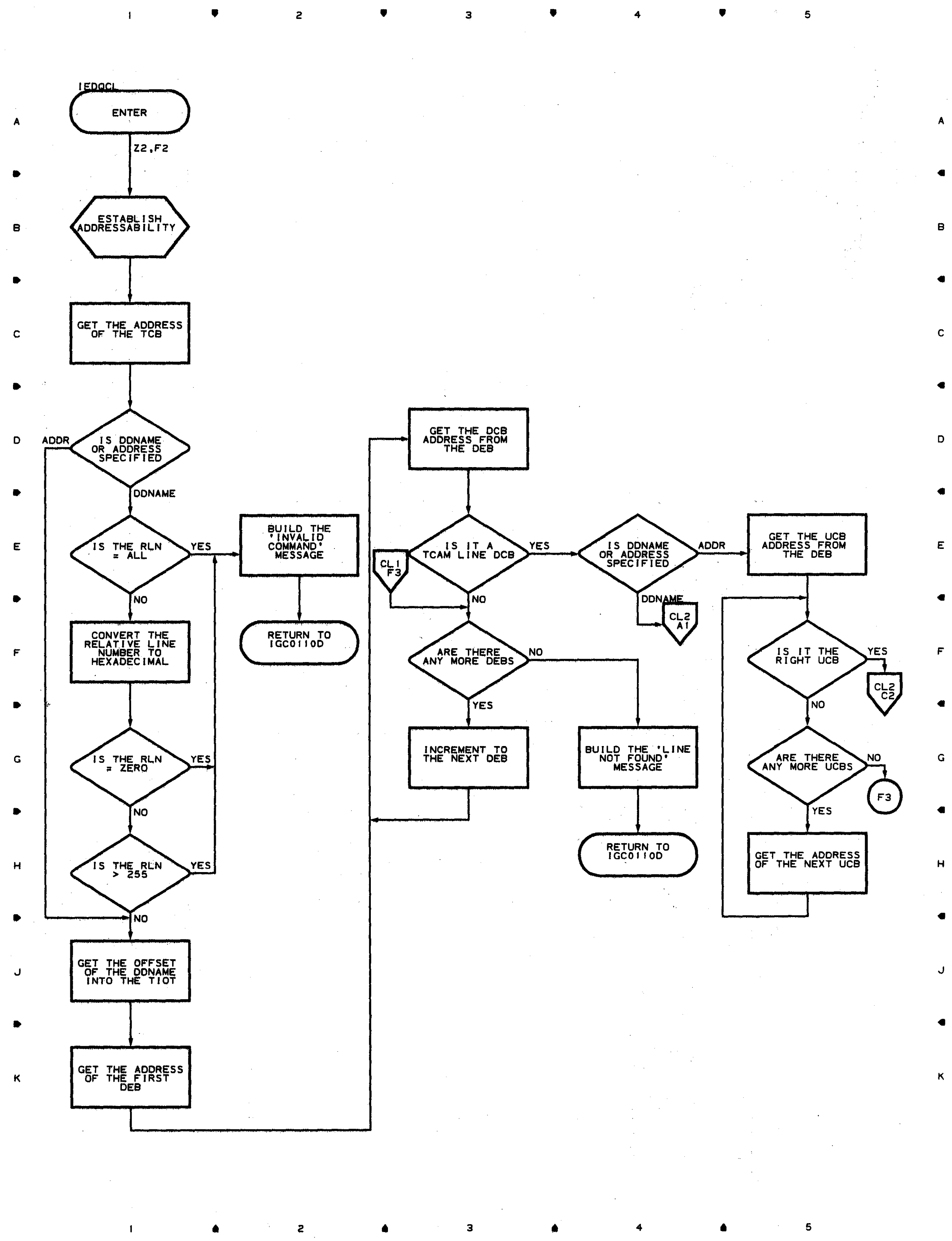

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQCL
ENTER
Z2,F2
ESTABLISH ADDRESSABILITY
GET THE ADDRESS OF THE TCB
ADDR
IS DDNAME OR ADDRESS SPECIFIED
DDNAME
IS THE RLN = ALL
YES
NO
CONVERT THE RELATIVE LINE NUMBER TO HEXADECIMAL
IS THE RLN = ZERO
YES
NO
IS THE RLN > 255
YES
NO
GET THE OFFSET OF THE DDNAME INTO THE TIOT
GET THE ADDRESS OF THE FIRST DEB
BUILD THE 'INVALID COMMAND' MESSAGE
RETURN TO IGC0110D
GET THE DCB ADDRESS FROM THE DEB
CL1 F3
IS IT A TCAM LINE DCB
YES
NO
ARE THERE ANY MORE DEBS
NO
YES
INCREMENT TO THE NEXT DEB
IS DDNAME OR ADDRESS SPECIFIED
ADDR
DDNAME
CL2 A1
BUILD THE 'LINE NOT FOUND' MESSAGE
RETURN TO IGC0110D
GET THE UCB ADDRESS FROM THE DEB
IS IT THE RIGHT UCB
YES
CL2 C2
NO
ARE THERE ANY MORE UCBS
NO
F3
YES
GET THE ADDRESS OF THE NEXT UCB

Chart CL-2 COPY INVITATION LIST ENTRY ROUTINE

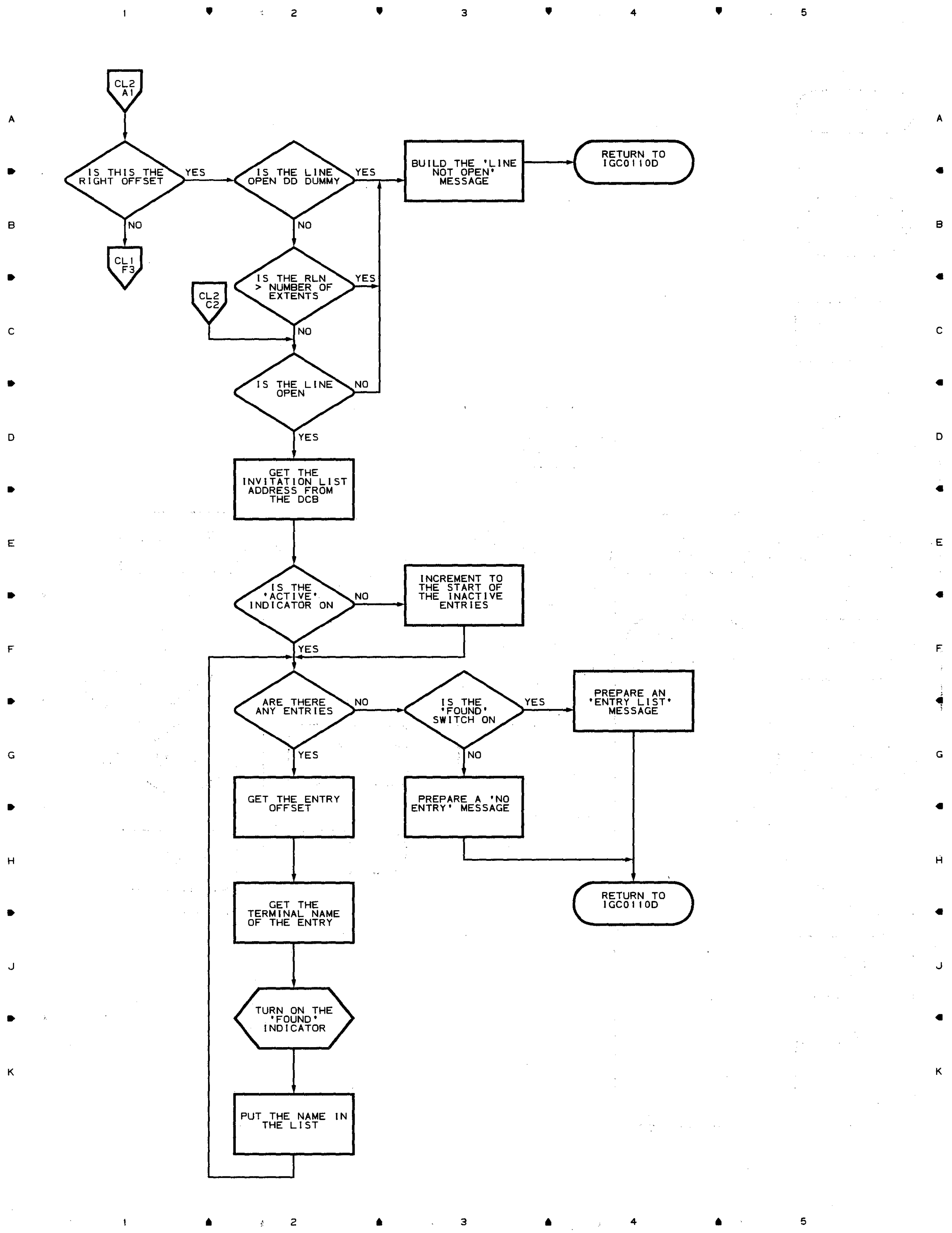
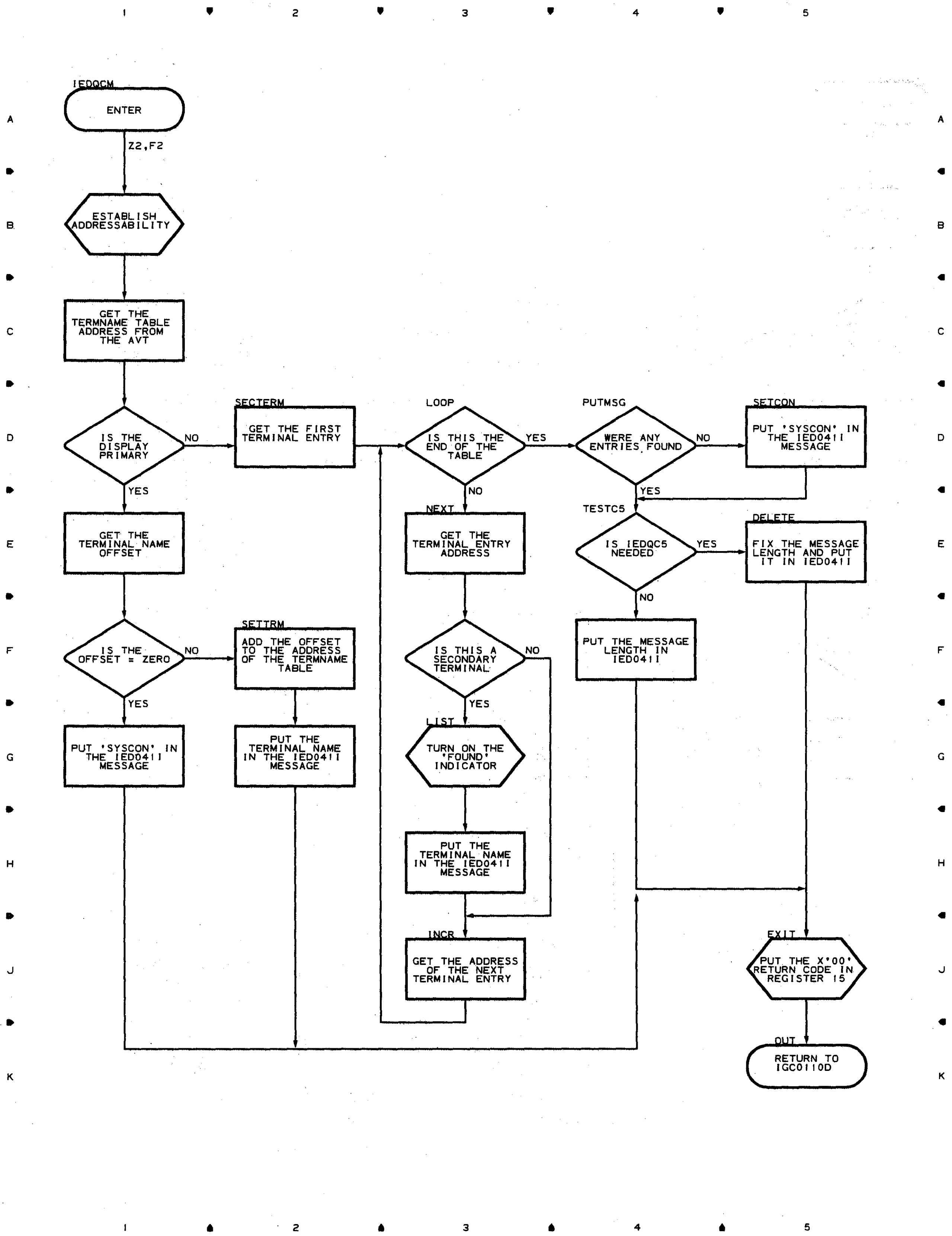
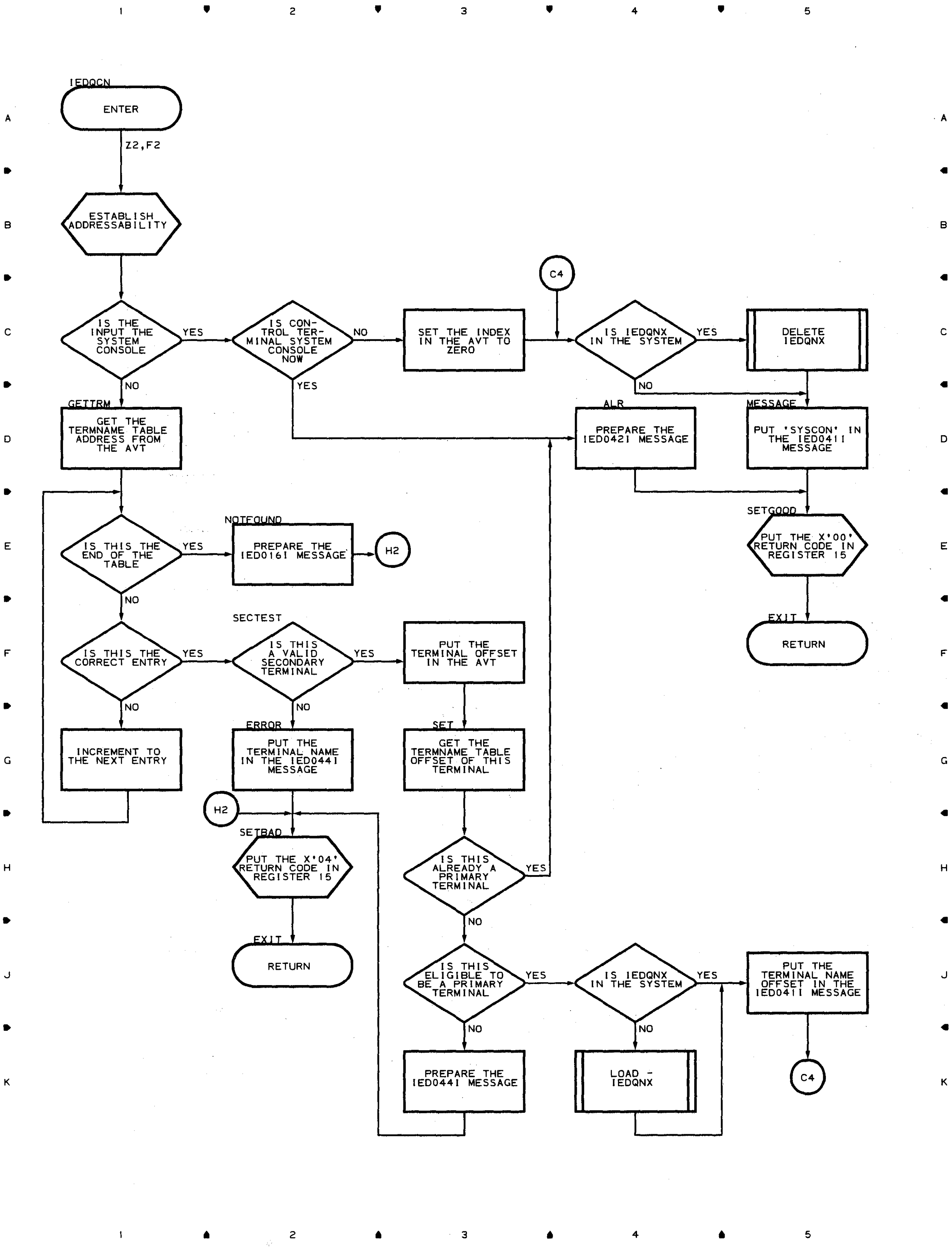

CL2 A1
IS THIS THE RIGHT OFFSET
YES
NO
CL1 F3
IS THE LINE OPEN DD DUMMY
YES
NO
BUILD THE 'LINE NOT OPEN' MESSAGE
RETURN TO IGC0110D
IS THE RLN > NUMBER OF EXTENTS
YES
NO
CL2 C2
IS THE LINE OPEN
NO
YES
GET THE INVITATION LIST ADDRESS FROM THE DCB
IS THE 'ACTIVE' INDICATOR ON
NO
INCREMENT TO THE START OF THE INACTIVE ENTRIES
YES
ARE THERE ANY ENTRIES
NO
IS THE 'FOUND' SWITCH ON
YES
PREPARE AN 'ENTRY LIST' MESSAGE
YES
NO
GET THE ENTRY OFFSET
PREPARE A 'NO ENTRY' MESSAGE
RETURN TO IGC0110D
GET THE TERMINAL NAME OF THE ENTRY
TURN ON THE 'FOUND' INDICATOR
PUT THE NAME IN THE LIST
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart CM COPY OPERATOR CONTROL TERMINAL ROUTINE



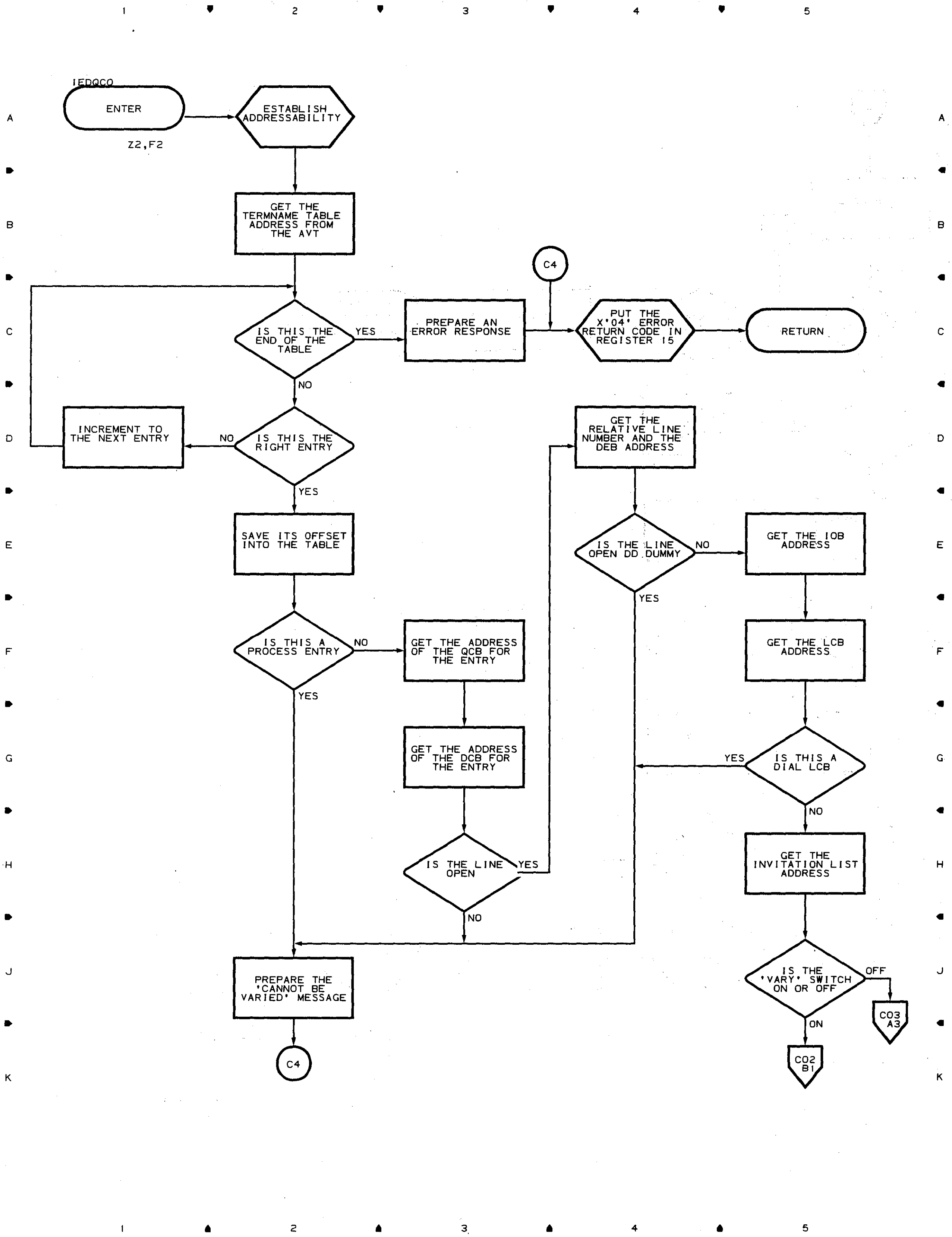
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQCM
ENTER
Z2,F2
ESTABLISH ADDRESSABILITY
GET THE TERMNAME TABLE ADDRESS FROM THE AVT
IS THE DISPLAY PRIMARY
NO
YES
SECTERM
GET THE FIRST TERMINAL ENTRY
LOOP
IS THIS THE END OF THE TABLE
YES
NO
PUTMSG
WERE ANY ENTRIES FOUND
NO
YES
SETCON
PUT 'SYSCON' IN THE IED041I MESSAGE
GET THE TERMINAL NAME OFFSET
NEXT
GET THE TERMINAL ENTRY ADDRESS
TESTC5
IS IEDQC5 NEEDED
YES
NO
DELETE
FIX THE MESSAGE LENGTH AND PUT IT IN IED041I
IS THE OFFSET = ZERO
NO
YES
SETTRM
ADD THE OFFSET TO THE ADDRESS OF THE TERMNAME TABLE
IS THIS A SECONDARY TERMINAL
NO
YES
PUT THE MESSAGE LENGTH IN IED041I
PUT 'SYSCON' IN THE IED041I MESSAGE
PUT THE TERMINAL NAME IN THE IED041I MESSAGE
LIST
TURN ON THE 'FOUND' INDICATOR
PUT THE TERMINAL NAME IN THE IED041I MESSAGE
INCR
GET THE ADDRESS OF THE NEXT TERMINAL ENTRY
EXIT
PUT THE X'00' RETURN CODE IN REGISTER 15
OUT
RETURN TO IGC0110D

Chart CN CHANGE CONTROL TERMINAL ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQCN
ENTER
Z2,F2
ESTABLISH ADDRESSABILITY
IS THE INPUT THE SYSTEM CONSOLE
YES
IS CONTROL TERMINAL SYSTEM CONSOLE NOW
NO
SET THE INDEX IN THE AVT TO ZERO
C4
IS IEDQNX IN THE SYSTEM
YES
DELETE IEDQNX
NO
YES
NO
GETTRM
GET THE TERMNAME TABLE ADDRESS FROM THE AVT
ALR
PREPARE THE IED042I MESSAGE
MESSAGE
PUT 'SYSCON' IN THE IED041I MESSAGE
SETGOOD
PUT THE X'00' RETURN CODE IN REGISTER 15
EXIT
RETURN
IS THIS THE END OF THE TABLE
YES
NOTFOUND
PREPARE THE IED016I MESSAGE
H2
NO
SECTEST
IS THIS THE CORRECT ENTRY
YES
IS THIS A VALID SECONDARY TERMINAL
YES
PUT THE TERMINAL OFFSET IN THE AVT
NO
NO
INCREMENT TO THE NEXT ENTRY
ERROR
PUT THE TERMINAL NAME IN THE IED044I MESSAGE
SET
GET THE TERMNAME TABLE OFFSET OF THIS TERMINAL
H2
SETBAD
PUT THE X'04' RETURN CODE IN REGISTER 15
IS THIS ALREADY A PRIMARY TERMINAL
YES
NO
EXIT
RETURN
IS THIS ELIGIBLE TO BE A PRIMARY TERMINAL
YES
IS IEDQNX IN THE SYSTEM
YES
PUT THE TERMINAL NAME OFFSET IN THE IED041I MESSAGE
NO
NO
PREPARE THE IED044I MESSAGE
LOAD - IEDQNX
C4
1
2
3
4
5

Chart CO-1 CHANGE TERMINAL ROUTINE



IEDQCO
ENTER
Z2,F2
ESTABLISH ADDRESSABILITY
GET THE TERMNAME TABLE ADDRESS FROM THE AVT
C4
IS THIS THE END OF THE TABLE
YES
NO
PREPARE AN ERROR RESPONSE
PUT THE X'04' ERROR RETURN CODE IN REGISTER 15
RETURN
INCREMENT TO THE NEXT ENTRY
NO
IS THIS THE RIGHT ENTRY
YES
GET THE RELATIVE LINE NUMBER AND THE DEB ADDRESS
SAVE ITS OFFSET INTO THE TABLE
IS THE LINE OPEN DD DUMMY
NO
YES
GET THE IOB ADDRESS
IS THIS A PROCESS ENTRY
NO
YES
GET THE ADDRESS OF THE QCB FOR THE ENTRY
GET THE LCB ADDRESS
GET THE ADDRESS OF THE DCB FOR THE ENTRY
YES
IS THIS A DIAL LCB
NO
IS THE LINE OPEN
YES
NO
GET THE INVITATION LIST ADDRESS
PREPARE THE 'CANNOT BE VARIED' MESSAGE
IS THE 'VARY' SWITCH ON OR OFF
OFF
ON
C4
CO3 A3
CO2 B1
1 2 3 4 5
A B C D E F G H J K

Chart CO-2 CHANGE TERMINAL ROUTINE

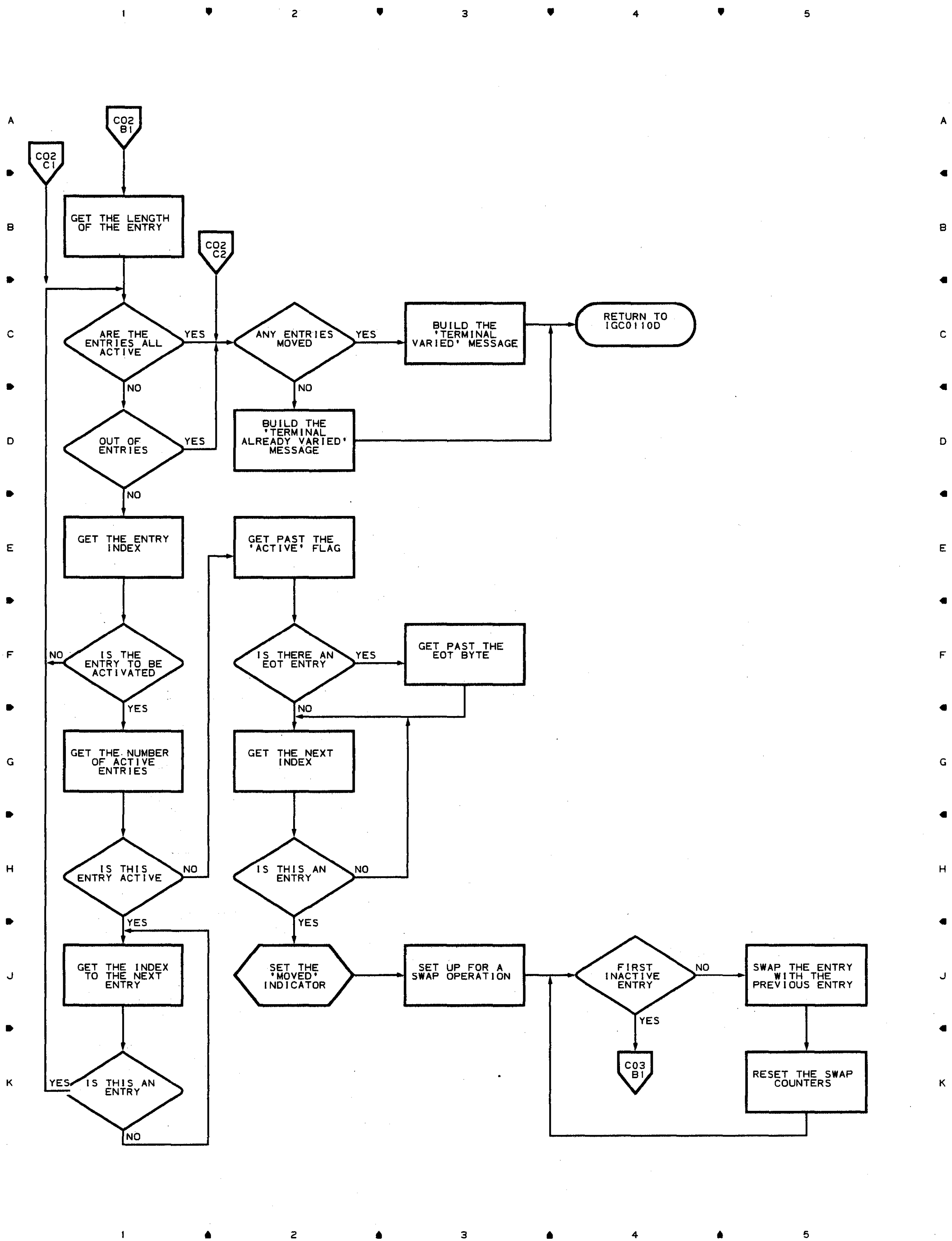

CO2 B1
CO2 C1
GET THE LENGTH OF THE ENTRY
CO2 C2
ARE THE ENTRIES ALL ACTIVE
YES
NO
ANY ENTRIES MOVED
YES
NO
BUILD THE 'TERMINAL VARIED' MESSAGE
RETURN TO IGC0110D
OUT OF ENTRIES
YES
NO
BUILD THE 'TERMINAL ALREADY VARIED' MESSAGE
GET THE ENTRY INDEX
GET PAST THE 'ACTIVE' FLAG
IS THE ENTRY TO BE ACTIVATED
NO
YES
IS THERE AN EOT ENTRY
YES
NO
GET PAST THE EOT BYTE
GET THE NUMBER OF ACTIVE ENTRIES
GET THE NEXT INDEX
IS THIS ENTRY ACTIVE
NO
YES
IS THIS AN ENTRY
NO
YES
GET THE INDEX TO THE NEXT ENTRY
SET THE 'MOVED' INDICATOR
SET UP FOR A SWAP OPERATION
FIRST INACTIVE ENTRY
NO
YES
SWAP THE ENTRY WITH THE PREVIOUS ENTRY
CO3 B1
RESET THE SWAP COUNTERS
IS THIS AN ENTRY
YES
NO
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart CO-3 CHANGE TERMINAL ROUTINE

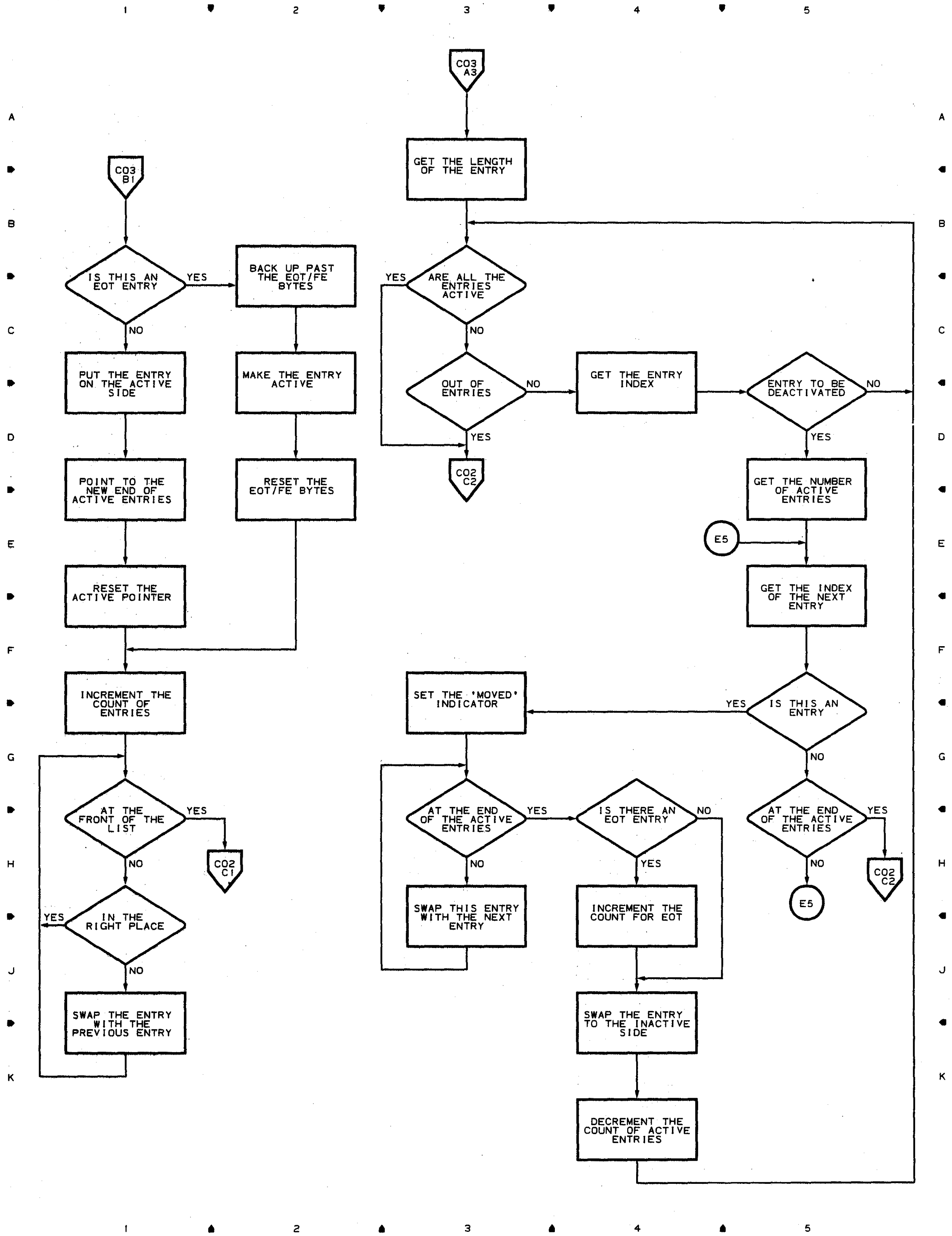
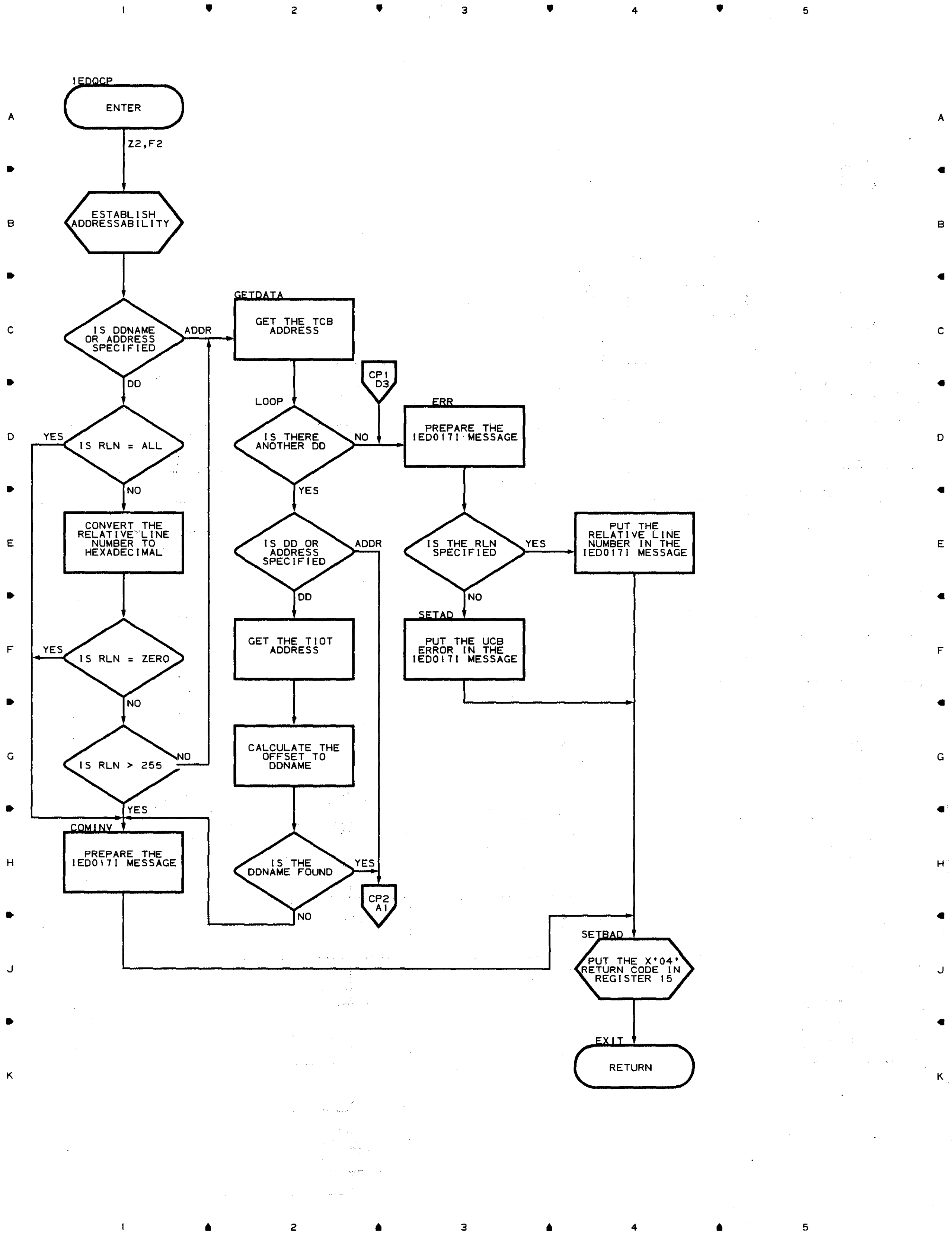
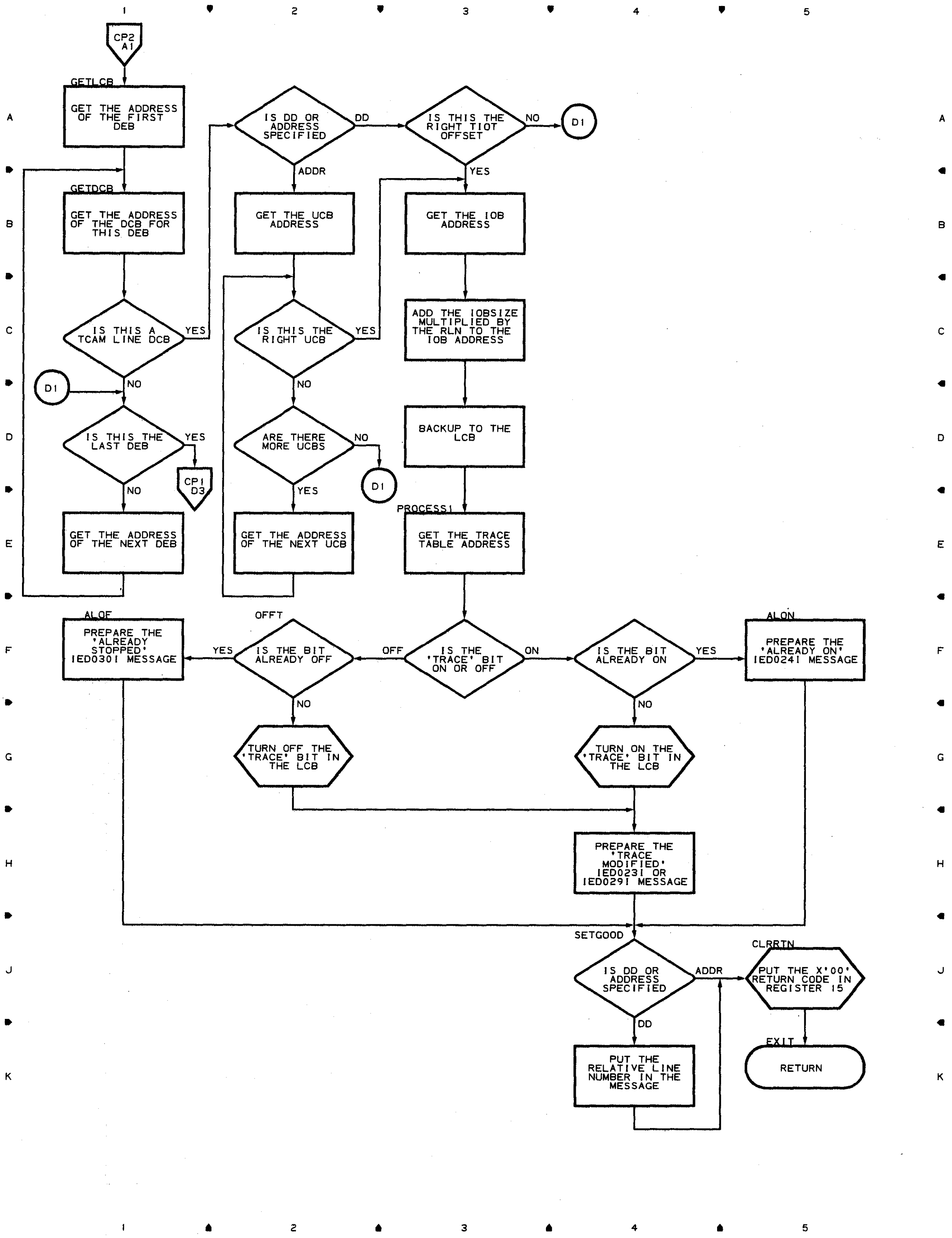

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
CO3 A3
GET THE LENGTH OF THE ENTRY
CO3 B1
IS THIS AN EOT ENTRY
YES
NO
BACK UP PAST THE EOT/FE BYTES
ARE ALL THE ENTRIES ACTIVE
YES
NO
PUT THE ENTRY ON THE ACTIVE SIDE
MAKE THE ENTRY ACTIVE
OUT OF ENTRIES
NO
YES
GET THE ENTRY INDEX
ENTRY TO BE DEACTIVATED
NO
YES
POINT TO THE NEW END OF ACTIVE ENTRIES
RESET THE EOT/FE BYTES
CO2 C2
GET THE NUMBER OF ACTIVE ENTRIES
E5
RESET THE ACTIVE POINTER
GET THE INDEX OF THE NEXT ENTRY
INCREMENT THE COUNT OF ENTRIES
SET THE 'MOVED' INDICATOR
IS THIS AN ENTRY
YES
NO
AT THE FRONT OF THE LIST
YES
NO
CO2 C1
AT THE END OF THE ACTIVE ENTRIES
YES
NO
IS THERE AN EOT ENTRY
NO
YES
AT THE END OF THE ACTIVE ENTRIES
YES
NO
CO2 C2
IN THE RIGHT PLACE
YES
NO
SWAP THIS ENTRY WITH THE NEXT ENTRY
INCREMENT THE COUNT FOR EOT
E5
SWAP THE ENTRY WITH THE PREVIOUS ENTRY
SWAP THE ENTRY TO THE INACTIVE SIDE
DECREMENT THE COUNT OF ACTIVE ENTRIES

Chart CP-1 ALTER TRACE STATUS ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQCP
ENTER
Z2,F2
ESTABLISH ADDRESSABILITY
IS DDNAME OR ADDRESS SPECIFIED
ADDR
DD
GETDATA
GET THE TCB ADDRESS
CP1 D3
LOOP
IS THERE ANOTHER DD
NO
YES
ERR
PREPARE THE IED017I MESSAGE
IS RLN = ALL
YES
NO
CONVERT THE RELATIVE LINE NUMBER TO HEXADECIMAL
IS DD OR ADDRESS SPECIFIED
ADDR
DD
IS THE RLN SPECIFIED
YES
NO
PUT THE RELATIVE LINE NUMBER IN THE IED017I MESSAGE
SETAD
PUT THE UCB ERROR IN THE IED017I MESSAGE
GET THE TIOT ADDRESS
IS RLN = ZERO
YES
NO
IS RLN > 255
NO
YES
CALCULATE THE OFFSET TO DDNAME
COMINV
PREPARE THE IED017I MESSAGE
IS THE DDNAME FOUND
YES
NO
CP2 A1
SETBAD
PUT THE X'04' RETURN CODE IN REGISTER 15
EXIT
RETURN

Chart CP-2 ALTER TRACE STATUS ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
CP2 A1
GETLCB
GET THE ADDRESS OF THE FIRST DEB
IS DD OR ADDRESS SPECIFIED
DD
IS THIS THE RIGHT TIOT OFFSET
NO
D1
ADDR
YES
GETDCB
GET THE ADDRESS OF THE DCB FOR THIS DEB
GET THE UCB ADDRESS
GET THE IOB ADDRESS
IS THIS A TCAM LINE DCB
YES
IS THIS THE RIGHT UCB
YES
ADD THE IOBSIZE MULTIPLIED BY THE RLN TO THE IOB ADDRESS
D1
NO
NO
IS THIS THE LAST DEB
YES
ARE THERE MORE UCBS
NO
BACKUP TO THE LCB
NO
CP1 D3
YES
D1
PROCESS1
GET THE ADDRESS OF THE NEXT DEB
GET THE ADDRESS OF THE NEXT UCB
GET THE TRACE TABLE ADDRESS
ALOF
OFFT
ALON
PREPARE THE 'ALREADY STOPPED' IED030I MESSAGE
YES
IS THE BIT ALREADY OFF
OFF
IS THE 'TRACE' BIT ON OR OFF
ON
IS THE BIT ALREADY ON
YES
PREPARE THE 'ALREADY ON' IED024I MESSAGE
NO
NO
TURN OFF THE 'TRACE' BIT IN THE LCB
TURN ON THE 'TRACE' BIT IN THE LCB
PREPARE THE 'TRACE MODIFIED' IED023I OR IED029I MESSAGE
SETGOOD
CLRRTN
IS DD OR ADDRESS SPECIFIED
ADDR
PUT THE X'00' RETURN CODE IN REGISTER 15
DD
EXIT
PUT THE RELATIVE LINE NUMBER IN THE MESSAGE
RETURN

STOP/RESUME TERMINAL TRANSMISSION ROUTINE

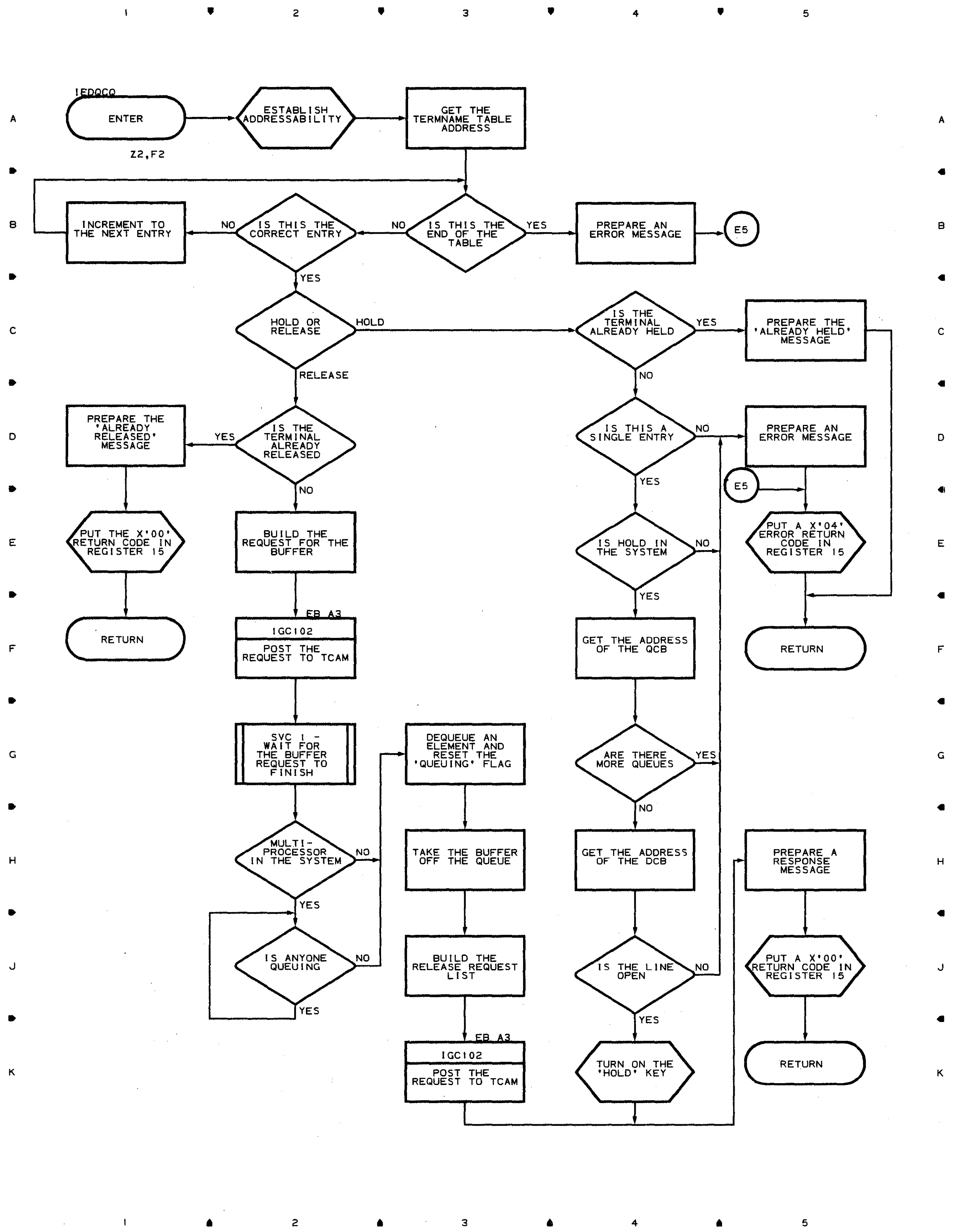

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQCQ
ENTER
Z2,F2
ESTABLISH ADDRESSABILITY
GET THE TERMNAME TABLE ADDRESS
INCREMENT TO THE NEXT ENTRY
NO
IS THIS THE CORRECT ENTRY
NO
IS THIS THE END OF THE TABLE
YES
PREPARE AN ERROR MESSAGE
E5
YES
HOLD OR RELEASE
HOLD
IS THE TERMINAL ALREADY HELD
YES
PREPARE THE 'ALREADY HELD' MESSAGE
RELEASE
NO
PREPARE THE 'ALREADY RELEASED' MESSAGE
YES
IS THE TERMINAL ALREADY RELEASED
IS THIS A SINGLE ENTRY
NO
PREPARE AN ERROR MESSAGE
NO
YES
E5
PUT THE X'00' RETURN CODE IN REGISTER 15
BUILD THE REQUEST FOR THE BUFFER
IS HOLD IN THE SYSTEM
NO
PUT A X'04' ERROR RETURN CODE IN REGISTER 15
YES
RETURN
EB A3
IGC102
POST THE REQUEST TO TCAM
GET THE ADDRESS OF THE QCB
RETURN
SVC 1 - WAIT FOR THE BUFFER REQUEST TO FINISH
DEQUEUE AN ELEMENT AND RESET THE 'QUEUING' FLAG
ARE THERE MORE QUEUES
YES
NO
MULTI-PROCESSOR IN THE SYSTEM
NO
TAKE THE BUFFER OFF THE QUEUE
GET THE ADDRESS OF THE DCB
PREPARE A RESPONSE MESSAGE
YES
IS ANYONE QUEUING
NO
BUILD THE RELEASE REQUEST LIST
IS THE LINE OPEN
NO
PUT A X'00' RETURN CODE IN REGISTER 15
YES
YES
EB A3
IGC102
POST THE REQUEST TO TCAM
TURN ON THE 'HOLD' KEY
RETURN

Chart CU-1 START LINE ROUTINE

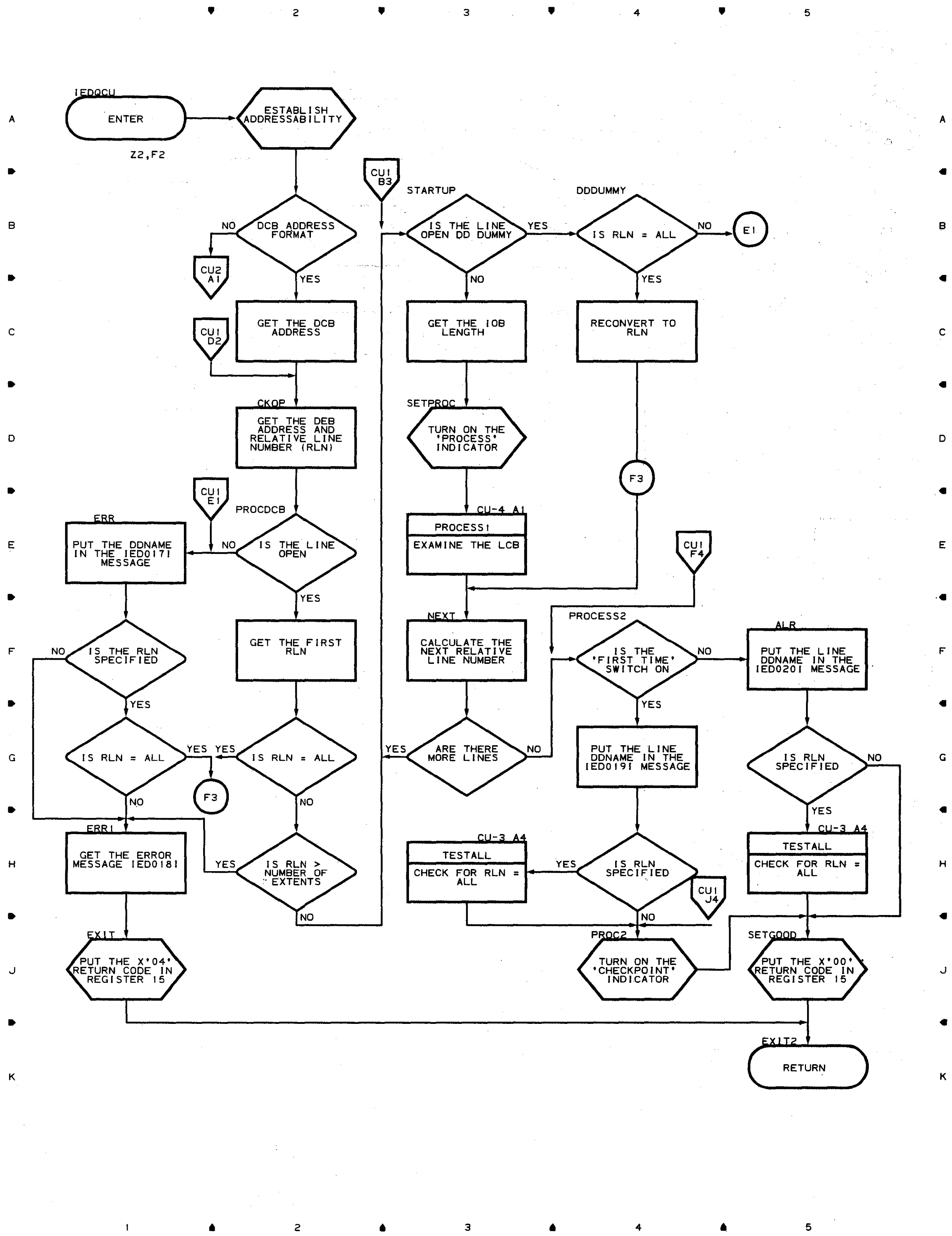


Chart CU-2 START LINE ROUTINE

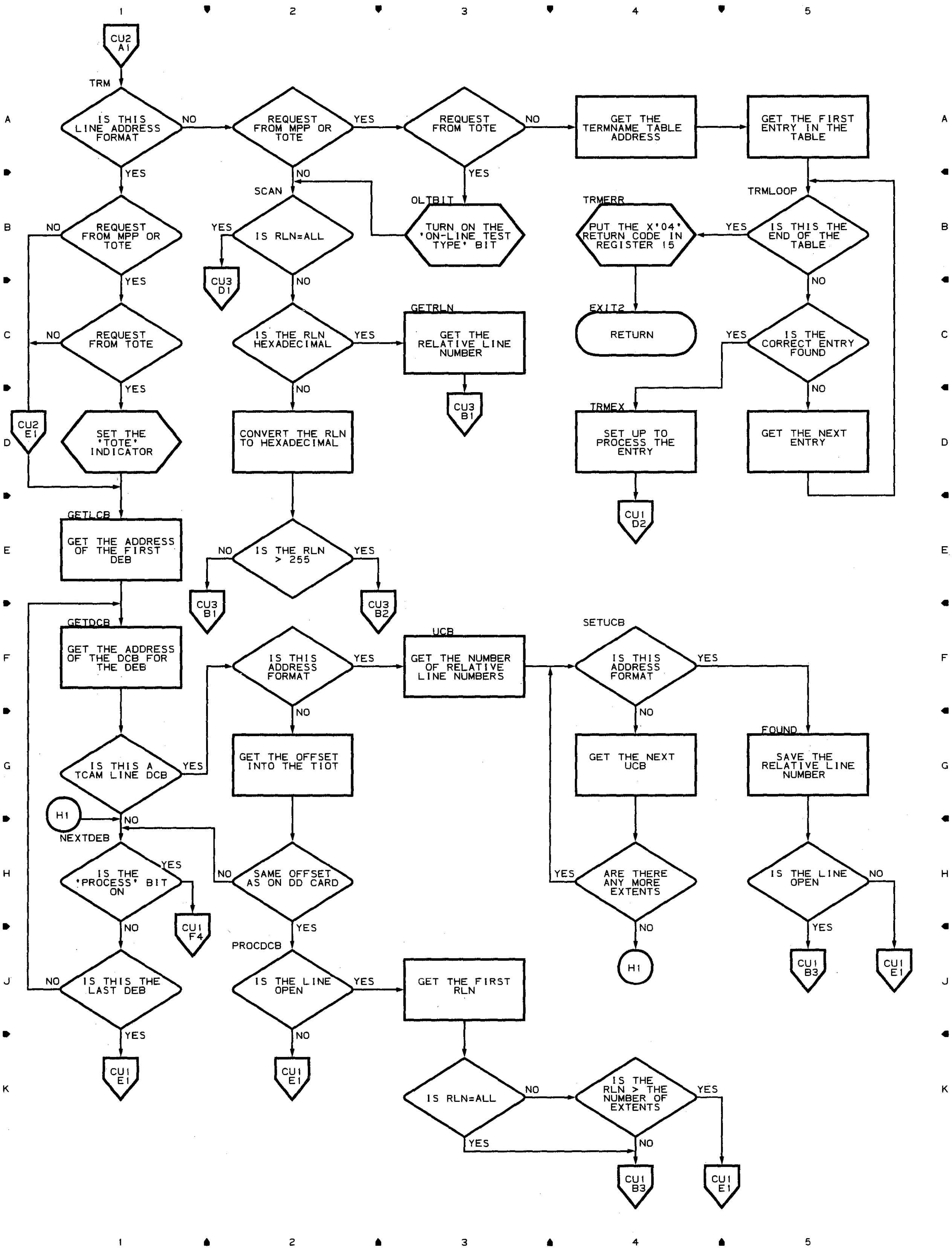
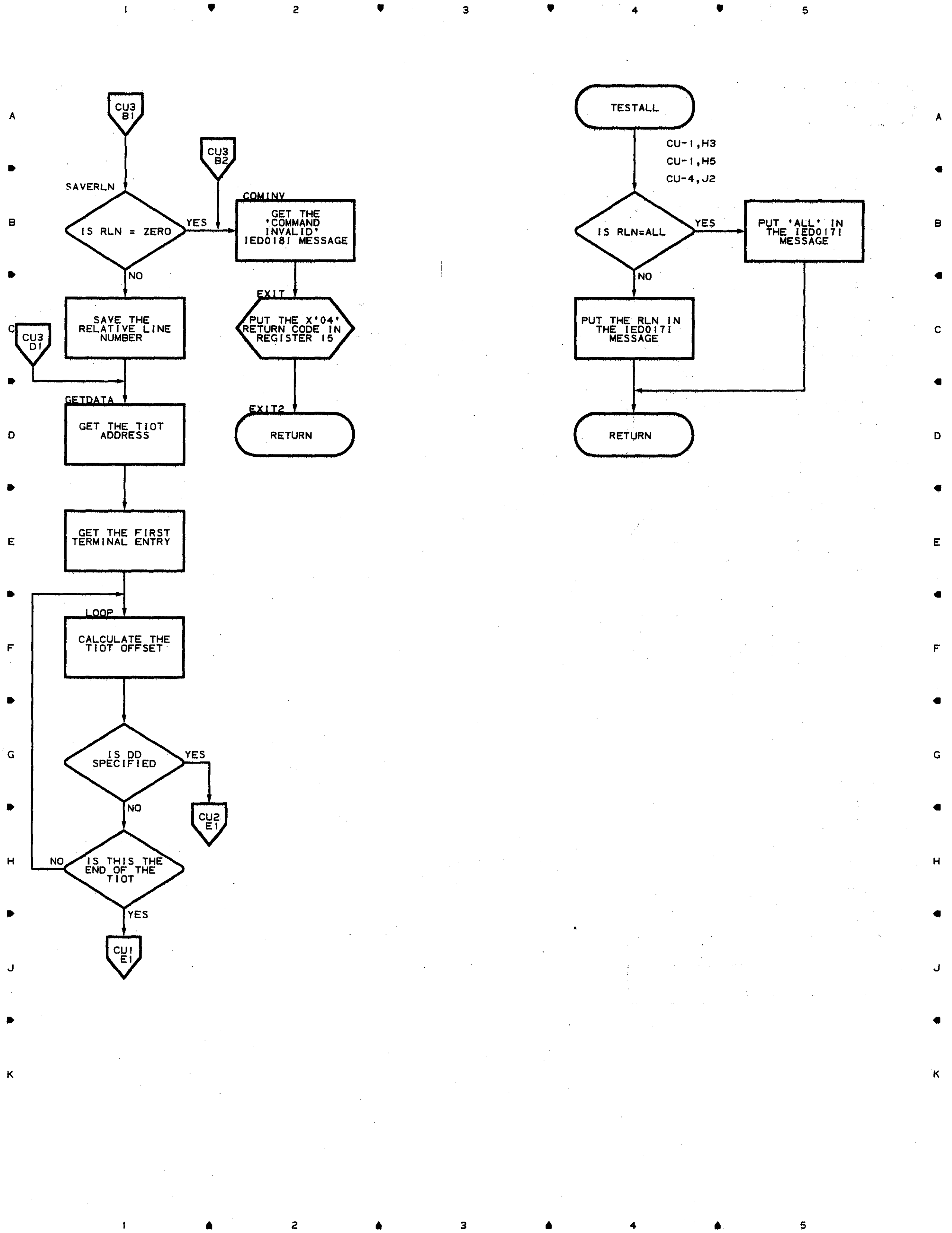
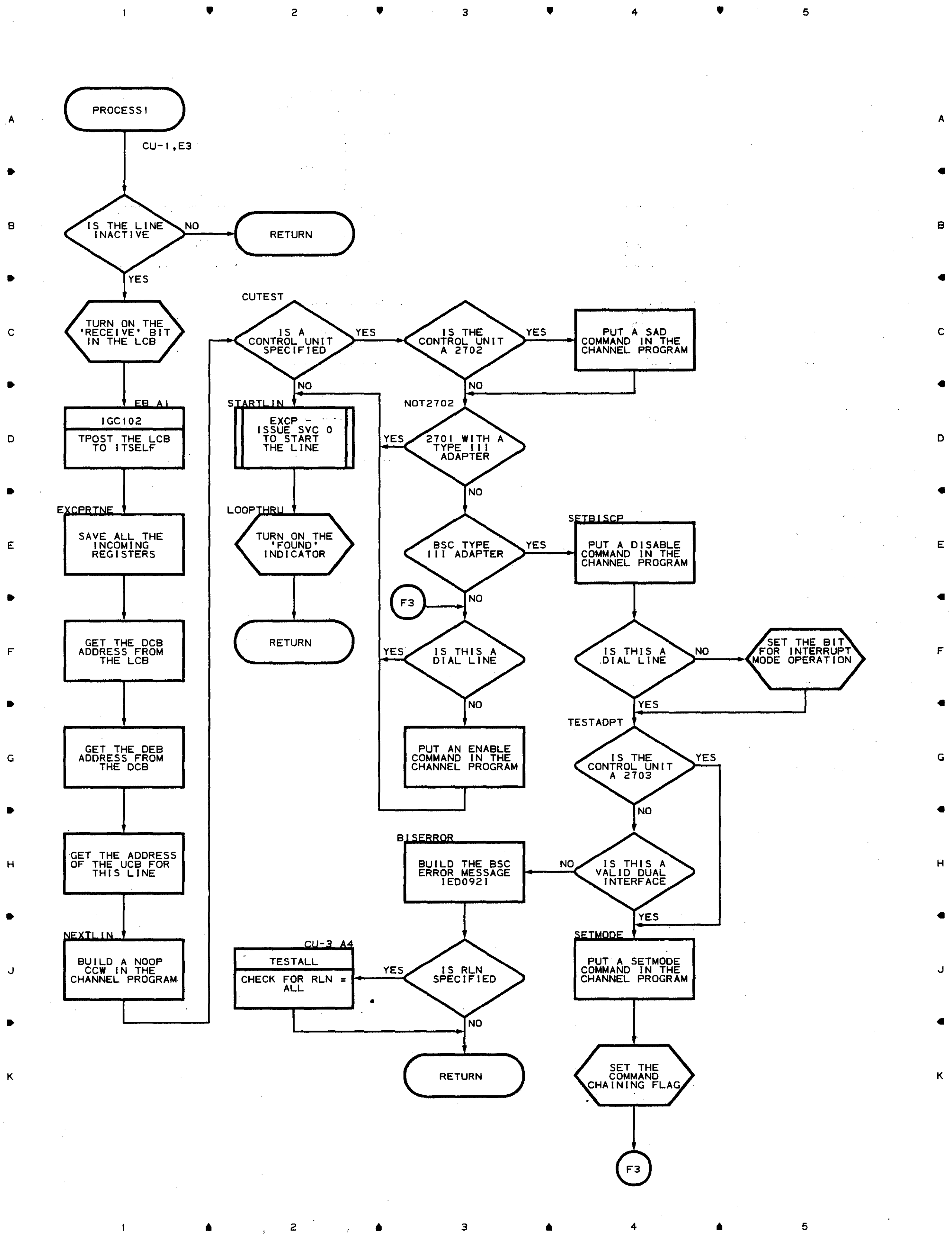


Chart CU-3 START LINE ROUTINE



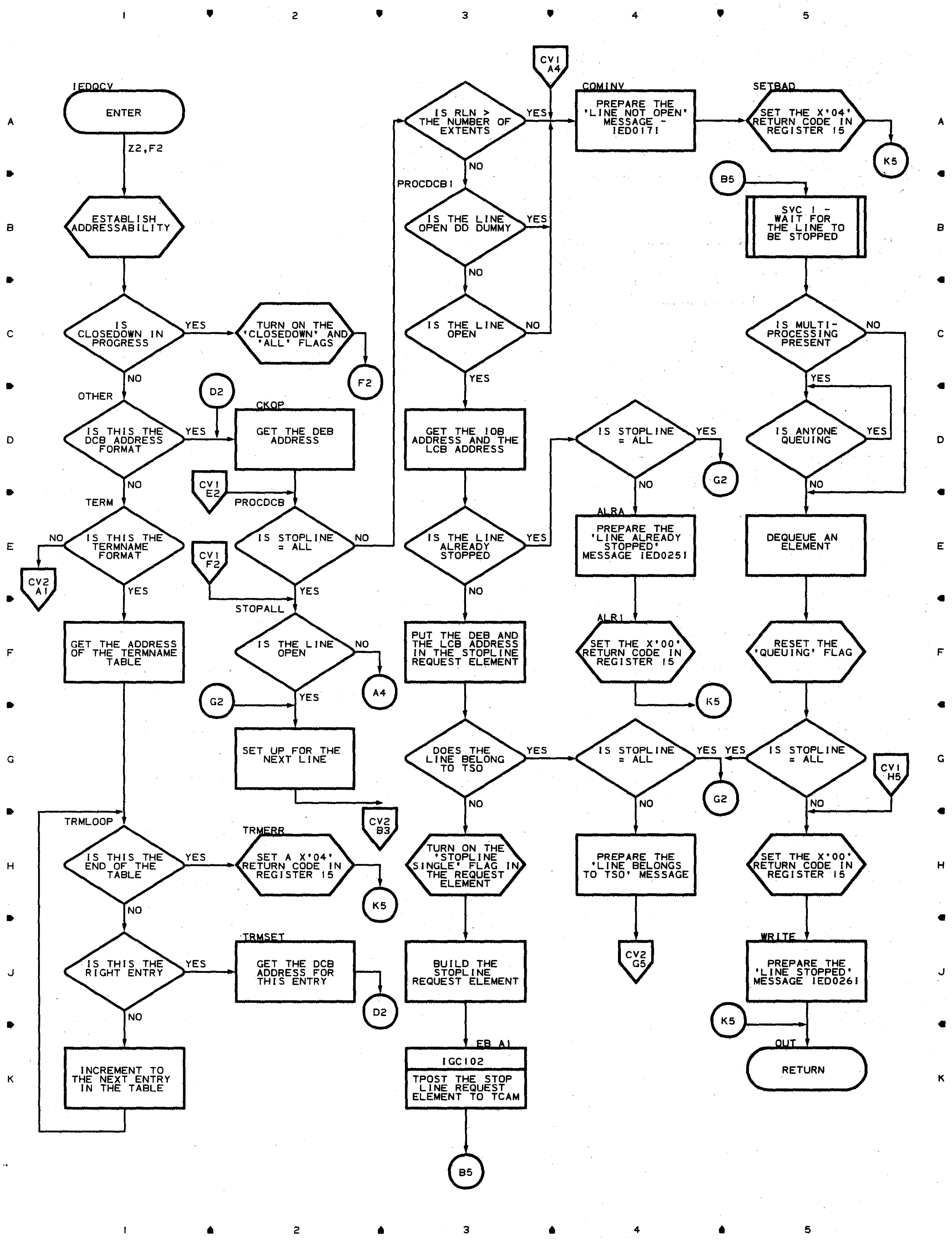
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
CU3 B1
CU3 B2
SAVERLN
IS RLN = ZERO
YES
NO
COMINV
GET THE 'COMMAND INVALID' IED018I MESSAGE
EXIT
PUT THE X'04' RETURN CODE IN REGISTER 15
EXIT2
RETURN
SAVE THE RELATIVE LINE NUMBER
CU3 D1
GETDATA
GET THE TIOT ADDRESS
GET THE FIRST TERMINAL ENTRY
LOOP
CALCULATE THE TIOT OFFSET
IS DD SPECIFIED
YES
CU2 E1
NO
IS THIS THE END OF THE TIOT
NO
YES
CU1 E1
TESTALL
CU-1,H3
CU-1,H5
CU-4,J2
IS RLN=ALL
YES
PUT 'ALL' IN THE IED017I MESSAGE
NO
PUT THE RLN IN THE IED017I MESSAGE
RETURN

Chart CU-4 START LINE ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
PROCESSI
CU-1,E3
IS THE LINE INACTIVE
NO
RETURN
YES
TURN ON THE 'RECEIVE' BIT IN THE LCB
EB A1
IGC102
TPOST THE LCB TO ITSELF
EXCPRTNE
SAVE ALL THE INCOMING REGISTERS
GET THE DCB ADDRESS FROM THE LCB
GET THE DEB ADDRESS FROM THE DCB
GET THE ADDRESS OF THE UCB FOR THIS LINE
NEXTLIN
BUILD A NOOP CCW IN THE CHANNEL PROGRAM
CUTEST
IS A CONTROL UNIT SPECIFIED
YES
NO
STARTLIN
EXCP - ISSUE SVC 0 TO START THE LINE
LOOPTHRU
TURN ON THE 'FOUND' INDICATOR
RETURN
IS THE CONTROL UNIT A 2702
YES
PUT A SAD COMMAND IN THE CHANNEL PROGRAM
NO
NOT2702
2701 WITH A TYPE III ADAPTER
YES
NO
BSC TYPE III ADAPTER
YES
SETBISCP
PUT A DISABLE COMMAND IN THE CHANNEL PROGRAM
F3
NO
IS THIS A DIAL LINE
YES
NO
PUT AN ENABLE COMMAND IN THE CHANNEL PROGRAM
IS THIS A DIAL LINE
NO
SET THE BIT FOR INTERRUPT MODE OPERATION
YES
TESTADPT
IS THE CONTROL UNIT A 2703
YES
NO
BISERROR
BUILD THE BSC ERROR MESSAGE IED092I
NO
IS THIS A VALID DUAL INTERFACE
YES
SETMODE
PUT A SETMODE COMMAND IN THE CHANNEL PROGRAM
SET THE COMMAND CHAINING FLAG
F3
CU-3 A4
TESTALL
CHECK FOR RLN = ALL
YES
IS RLN SPECIFIED
NO
RETURN
1
2
3
4
5

Chart CV-1 STOP LINE ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQCV
ENTER
Z2,F2
ESTABLISH ADDRESSABILITY
IS CLOSEDOWN IN PROGRESS
YES
TURN ON THE 'CLOSEDOWN' AND 'ALL' FLAGS
F2
NO
OTHER
D2
IS THIS THE DCB ADDRESS FORMAT
YES
CKOP
GET THE DEB ADDRESS
NO
TERM
CV1 E2
PROCDCB
IS THIS THE TERMNAME FORMAT
NO
CV2 A1
YES
CV1 F2
IS STOPLINE = ALL
NO
YES
STOPALL
GET THE ADDRESS OF THE TERMNAME TABLE
IS THE LINE OPEN
NO
A4
YES
G2
SET UP FOR THE NEXT LINE
CV2 B3
TRMLOOP
IS THIS THE END OF THE TABLE
YES
TRMERR
SET A X'04' RETURN CODE IN REGISTER 15
K5
NO
IS THIS THE RIGHT ENTRY
YES
TRMSET
GET THE DCB ADDRESS FOR THIS ENTRY
D2
NO
INCREMENT TO THE NEXT ENTRY IN THE TABLE
CV1 A4
IS RLN > THE NUMBER OF EXTENTS
YES
NO
PROCDCB1
IS THE LINE OPEN DD DUMMY
YES
NO
IS THE LINE OPEN
NO
YES
GET THE IOB ADDRESS AND THE LCB ADDRESS
IS THE LINE ALREADY STOPPED
YES
NO
PUT THE DEB AND THE LCB ADDRESS IN THE STOPLINE REQUEST ELEMENT
DOES THE LINE BELONG TO TSO
YES
NO
TURN ON THE 'STOPLINE SINGLE' FLAG IN THE REQUEST ELEMENT
BUILD THE STOPLINE REQUEST ELEMENT
EB A1
IGC102
TPOST THE STOP LINE REQUEST ELEMENT TO TCAM
B5
COMINV
PREPARE THE 'LINE NOT OPEN' MESSAGE - IED017I
IS STOPLINE = ALL
YES
G2
NO
ALRA
PREPARE THE 'LINE ALREADY STOPPED' MESSAGE IED025I
ALR1
SET THE X'00' RETURN CODE IN REGISTER 15
K5
IS STOPLINE = ALL
YES
G2
NO
PREPARE THE 'LINE BELONGS TO TSO' MESSAGE
CV2 G5
SETBAD
SET THE X'04' RETURN CODE IN REGISTER 15
K5
B5
SVC 1 - WAIT FOR THE LINE TO BE STOPPED
IS MULTI-PROCESSING PRESENT
NO
YES
IS ANYONE QUEUING
YES
NO
DEQUEUE AN ELEMENT
RESET THE 'QUEUING' FLAG
IS STOPLINE = ALL
YES
CV1 H5
NO
SET THE X'00' RETURN CODE IN REGISTER 15
WRITE
PREPARE THE 'LINE STOPPED' MESSAGE IED026I
K5
OUT
RETURN

Chart CV-2 STOPLINE ROUTINE

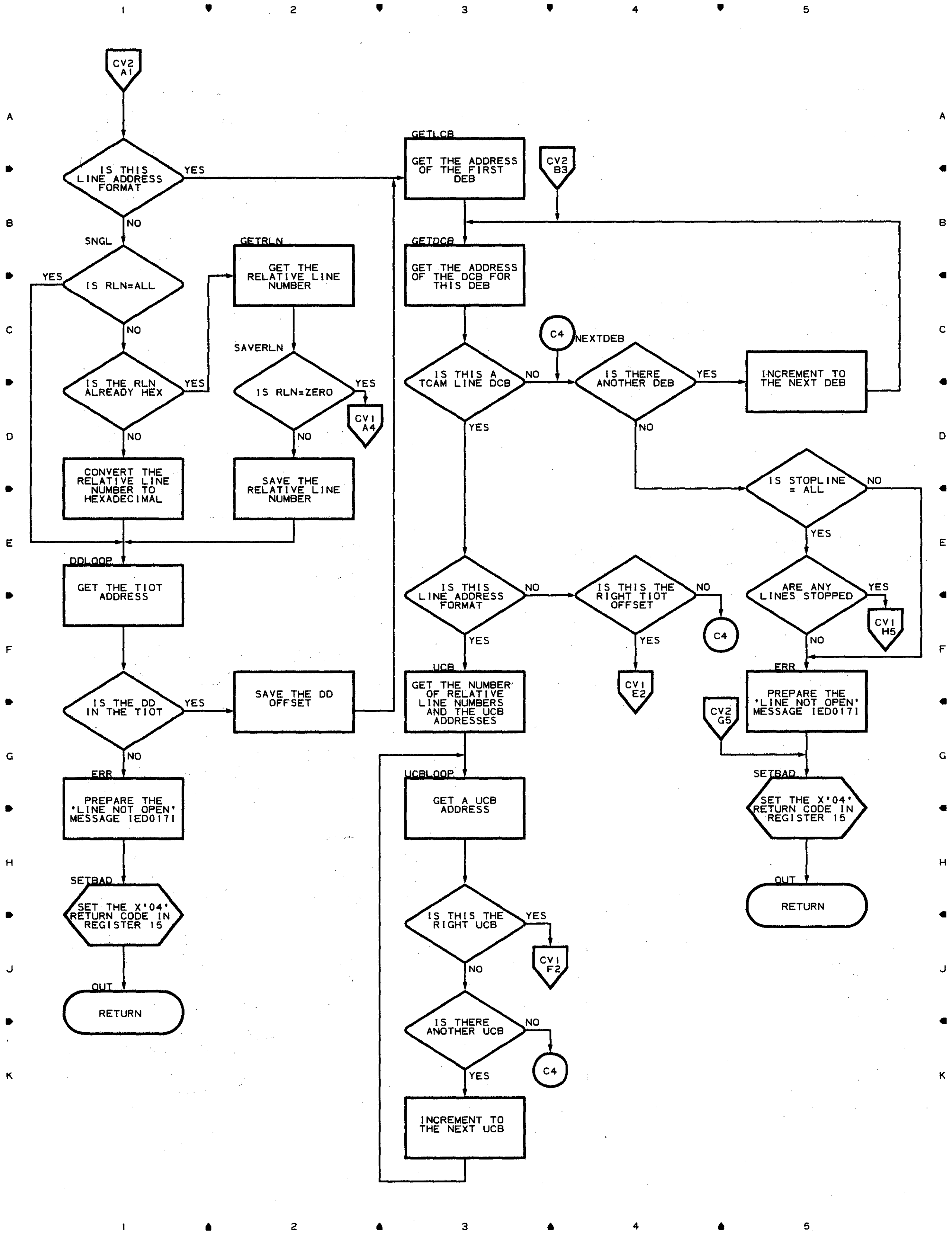
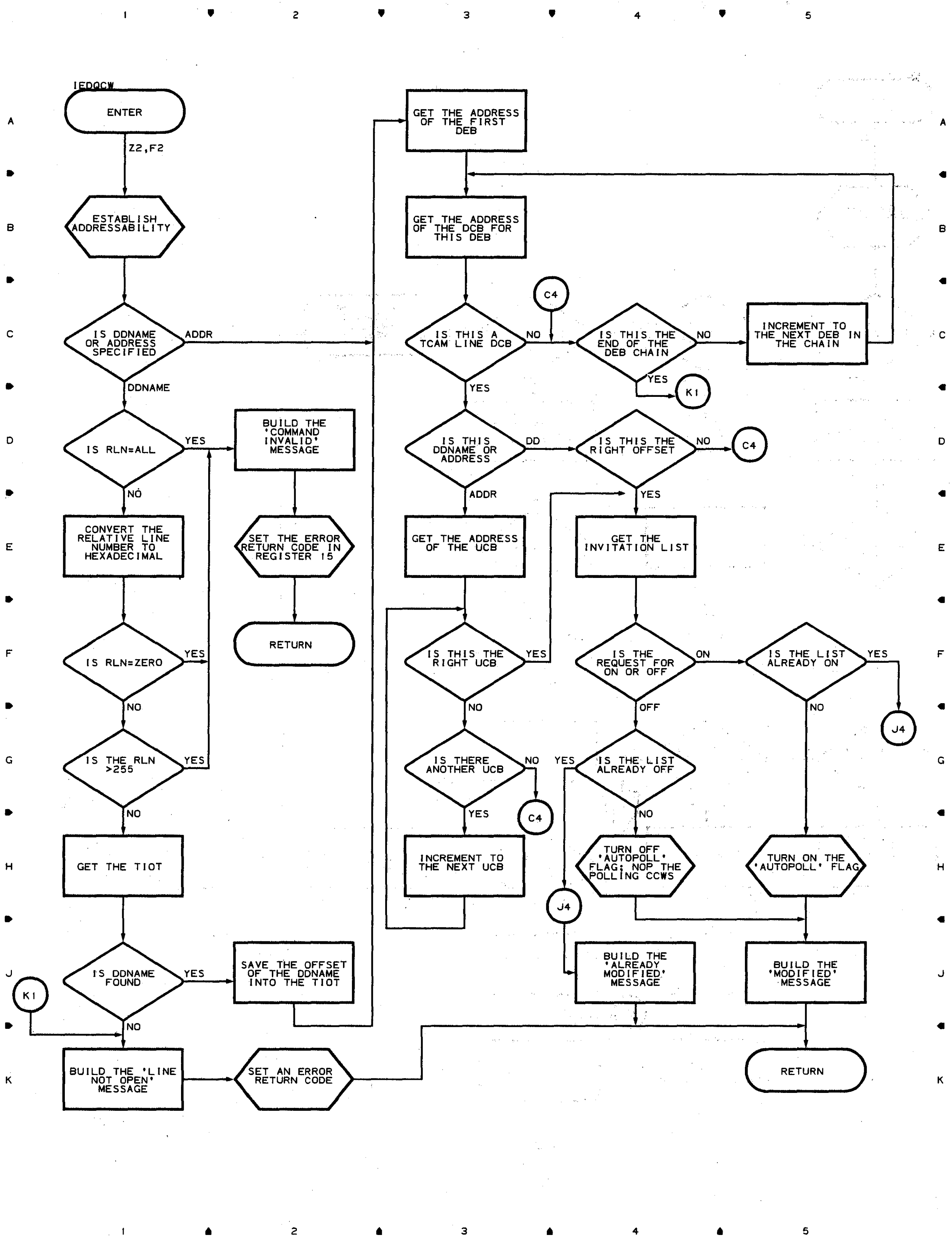
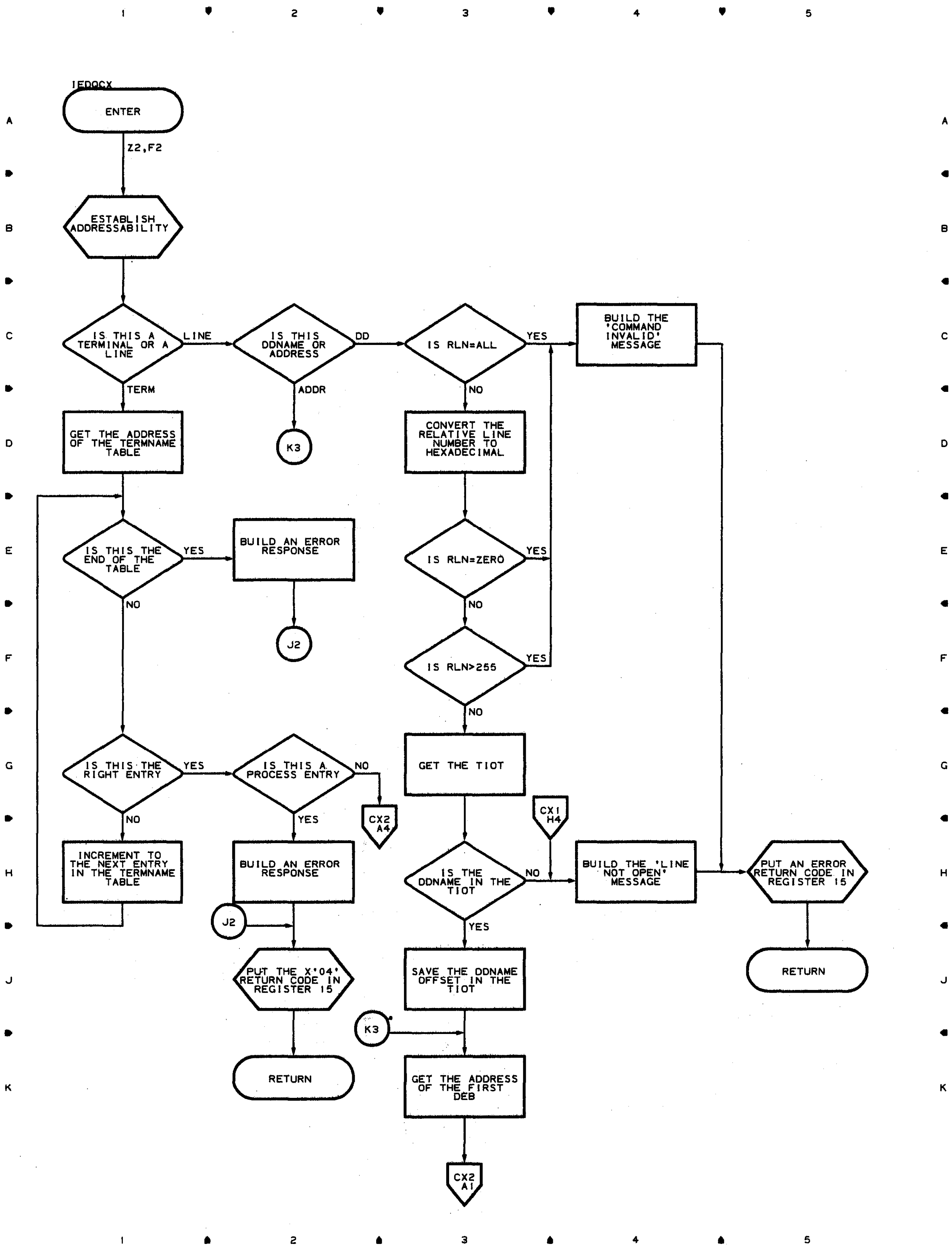

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
CV2 A1
IS THIS LINE ADDRESS FORMAT
YES
NO
GETLCB
GET THE ADDRESS OF THE FIRST DEB
CV2 B3
SNGL
IS RLN=ALL
YES
NO
GETRLN
GET THE RELATIVE LINE NUMBER
GETDCB
GET THE ADDRESS OF THE DCB FOR THIS DEB
C4
NEXTDEB
IS THE RLN ALREADY HEX
YES
NO
SAVERLN
IS RLN=ZERO
YES
NO
CV1 A4
IS THIS A TCAM LINE DCB
NO
YES
IS THERE ANOTHER DEB
YES
NO
INCREMENT TO THE NEXT DEB
CONVERT THE RELATIVE LINE NUMBER TO HEXADECIMAL
SAVE THE RELATIVE LINE NUMBER
IS STOPLINE = ALL
NO
YES
DDLOOP
GET THE TIOT ADDRESS
IS THIS LINE ADDRESS FORMAT
NO
YES
IS THIS THE RIGHT TIOT OFFSET
NO
YES
C4
ARE ANY LINES STOPPED
YES
NO
CV1 H5
IS THE DD IN THE TIOT
YES
NO
SAVE THE DD OFFSET
UCB
GET THE NUMBER OF RELATIVE LINE NUMBERS AND THE UCB ADDRESSES
CV1 E2
ERR
PREPARE THE 'LINE NOT OPEN' MESSAGE IED017I
CV2 G5
ERR
PREPARE THE 'LINE NOT OPEN' MESSAGE IED017I
UCBLOOP
GET A UCB ADDRESS
SETBAD
SET THE X'04' RETURN CODE IN REGISTER 15
OUT
RETURN
SETBAD
SET THE X'04' RETURN CODE IN REGISTER 15
IS THIS THE RIGHT UCB
YES
NO
CV1 F2
OUT
RETURN
IS THERE ANOTHER UCB
NO
YES
C4
INCREMENT TO THE NEXT UCB

Chart CW MODIFY POLL ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQCW
ENTER
Z2,F2
ESTABLISH ADDRESSABILITY
IS DDNAME OR ADDRESS SPECIFIED
ADDR
DDNAME
IS RLN=ALL
YES
NO
BUILD THE 'COMMAND INVALID' MESSAGE
SET THE ERROR RETURN CODE IN REGISTER 15
RETURN
CONVERT THE RELATIVE LINE NUMBER TO HEXADECIMAL
IS RLN=ZERO
YES
NO
IS THE RLN >255
YES
NO
GET THE TIOT
IS DDNAME FOUND
YES
NO
SAVE THE OFFSET OF THE DDNAME INTO THE TIOT
K1
BUILD THE 'LINE NOT OPEN' MESSAGE
SET AN ERROR RETURN CODE
GET THE ADDRESS OF THE FIRST DEB
GET THE ADDRESS OF THE DCB FOR THIS DEB
C4
IS THIS A TCAM LINE DCB
NO
YES
IS THIS THE END OF THE DEB CHAIN
NO
YES
K1
INCREMENT TO THE NEXT DEB IN THE CHAIN
IS THIS DDNAME OR ADDRESS
DD
ADDR
IS THIS THE RIGHT OFFSET
NO
C4
YES
GET THE ADDRESS OF THE UCB
GET THE INVITATION LIST
IS THIS THE RIGHT UCB
YES
NO
IS THE REQUEST FOR ON OR OFF
ON
OFF
IS THE LIST ALREADY ON
YES
NO
J4
IS THERE ANOTHER UCB
NO
YES
C4
IS THE LIST ALREADY OFF
YES
NO
INCREMENT TO THE NEXT UCB
TURN OFF 'AUTOPOLL' FLAG; NOP THE POLLING CCWS
TURN ON THE 'AUTOPOLL' FLAG
J4
BUILD THE 'ALREADY MODIFIED' MESSAGE
BUILD THE 'MODIFIED' MESSAGE
RETURN

Chart CX-1 MODIFY INTENSE ROUTINE



IEDQCX
ENTER
Z2,F2
ESTABLISH ADDRESSABILITY
IS THIS A TERMINAL OR A LINE
LINE
TERM
IS THIS DDNAME OR ADDRESS
DD
ADDR
K3
IS RLN=ALL
YES
NO
BUILD THE 'COMMAND INVALID' MESSAGE
GET THE ADDRESS OF THE TERMNAME TABLE
CONVERT THE RELATIVE LINE NUMBER TO HEXADECIMAL
IS THIS THE END OF THE TABLE
YES
NO
BUILD AN ERROR RESPONSE
J2
IS RLN=ZERO
YES
NO
IS RLN>255
YES
NO
IS THIS THE RIGHT ENTRY
YES
NO
IS THIS A PROCESS ENTRY
NO
YES
CX2 A4
GET THE TIOT
CX1 H4
INCREMENT TO THE NEXT ENTRY IN THE TERMNAME TABLE
BUILD AN ERROR RESPONSE
IS THE DDNAME IN THE TIOT
NO
YES
BUILD THE 'LINE NOT OPEN' MESSAGE
PUT AN ERROR RETURN CODE IN REGISTER 15
J2
PUT THE X'04' RETURN CODE IN REGISTER 15
SAVE THE DDNAME OFFSET IN THE TIOT
RETURN
K3
RETURN
GET THE ADDRESS OF THE FIRST DEB
CX2 A1
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

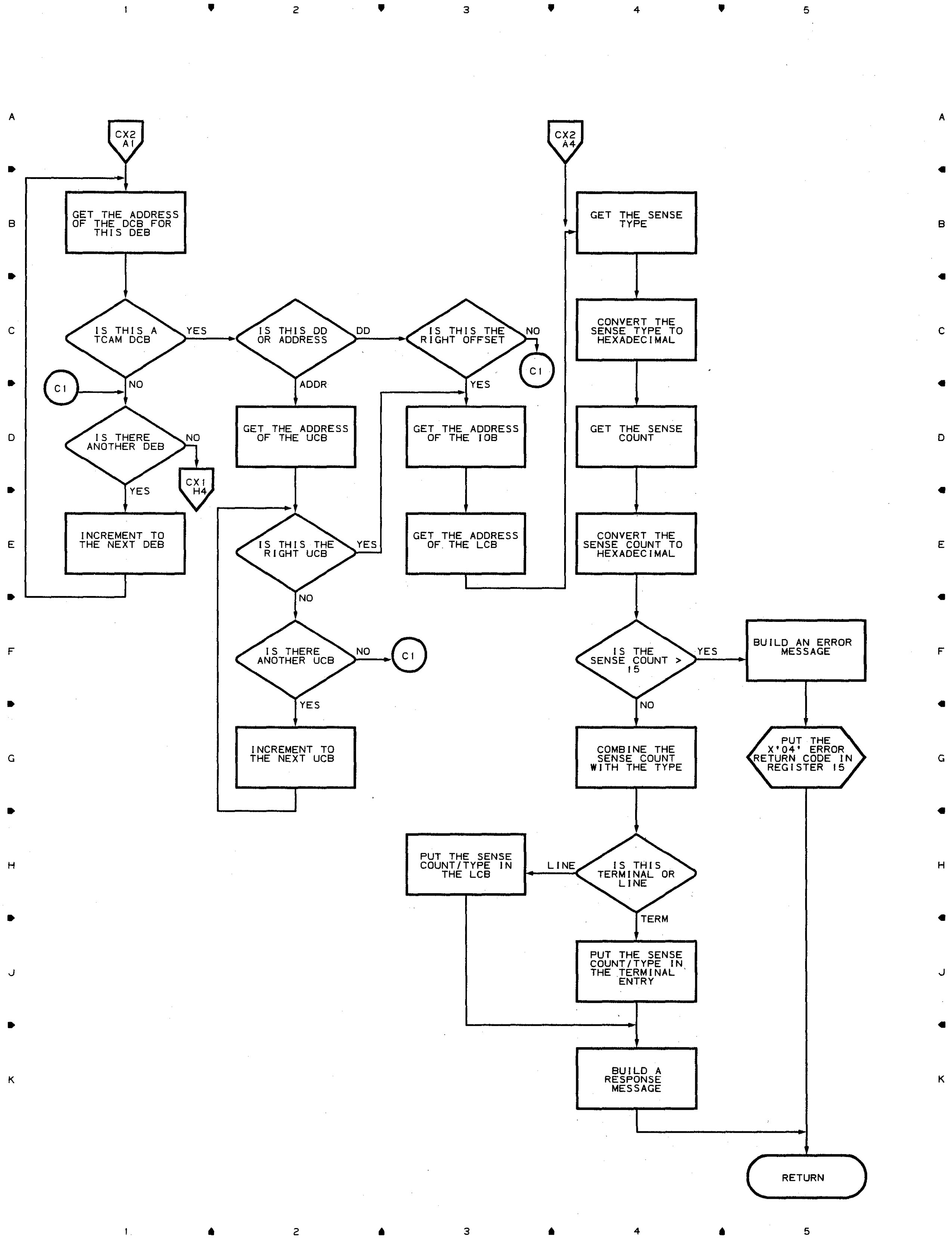
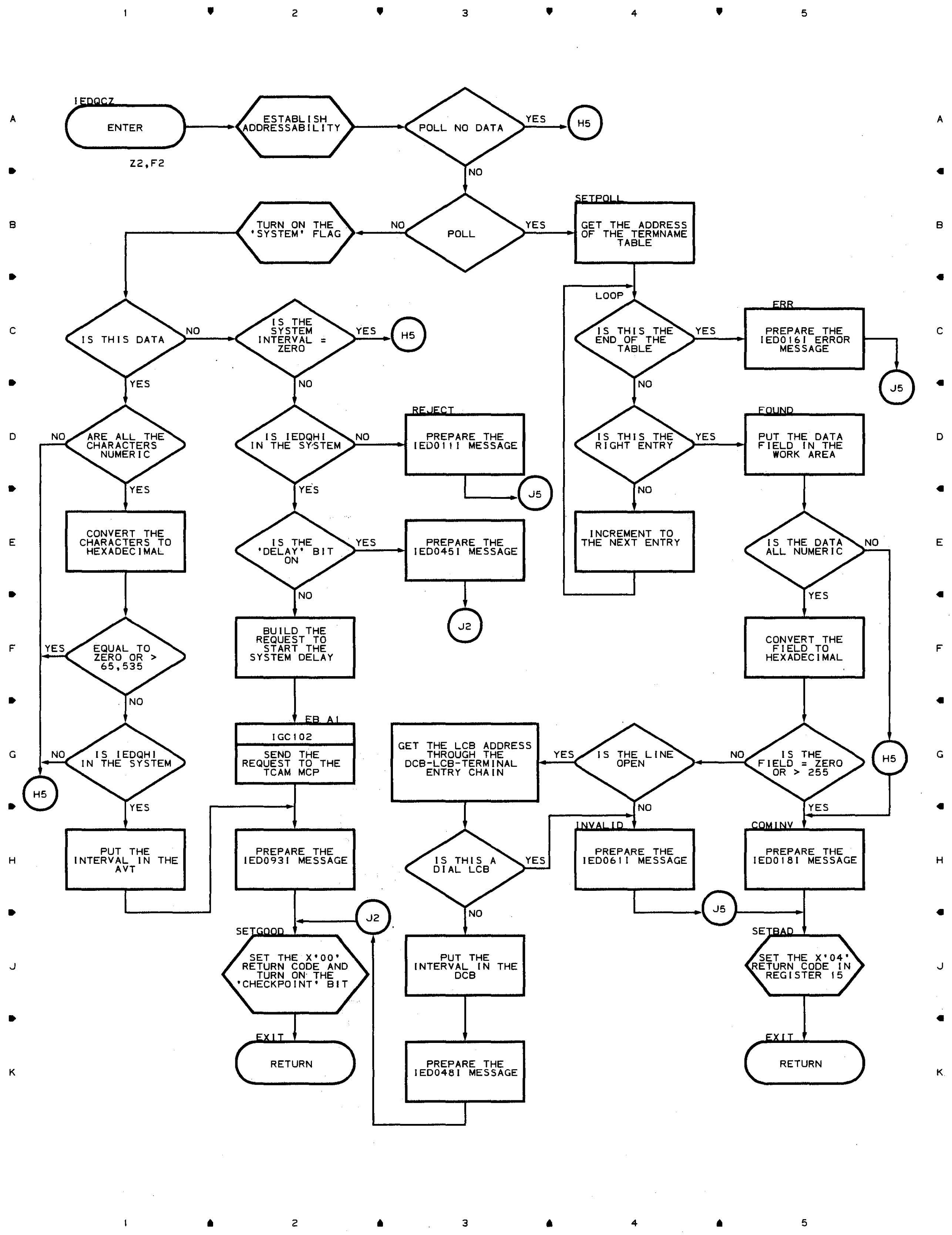
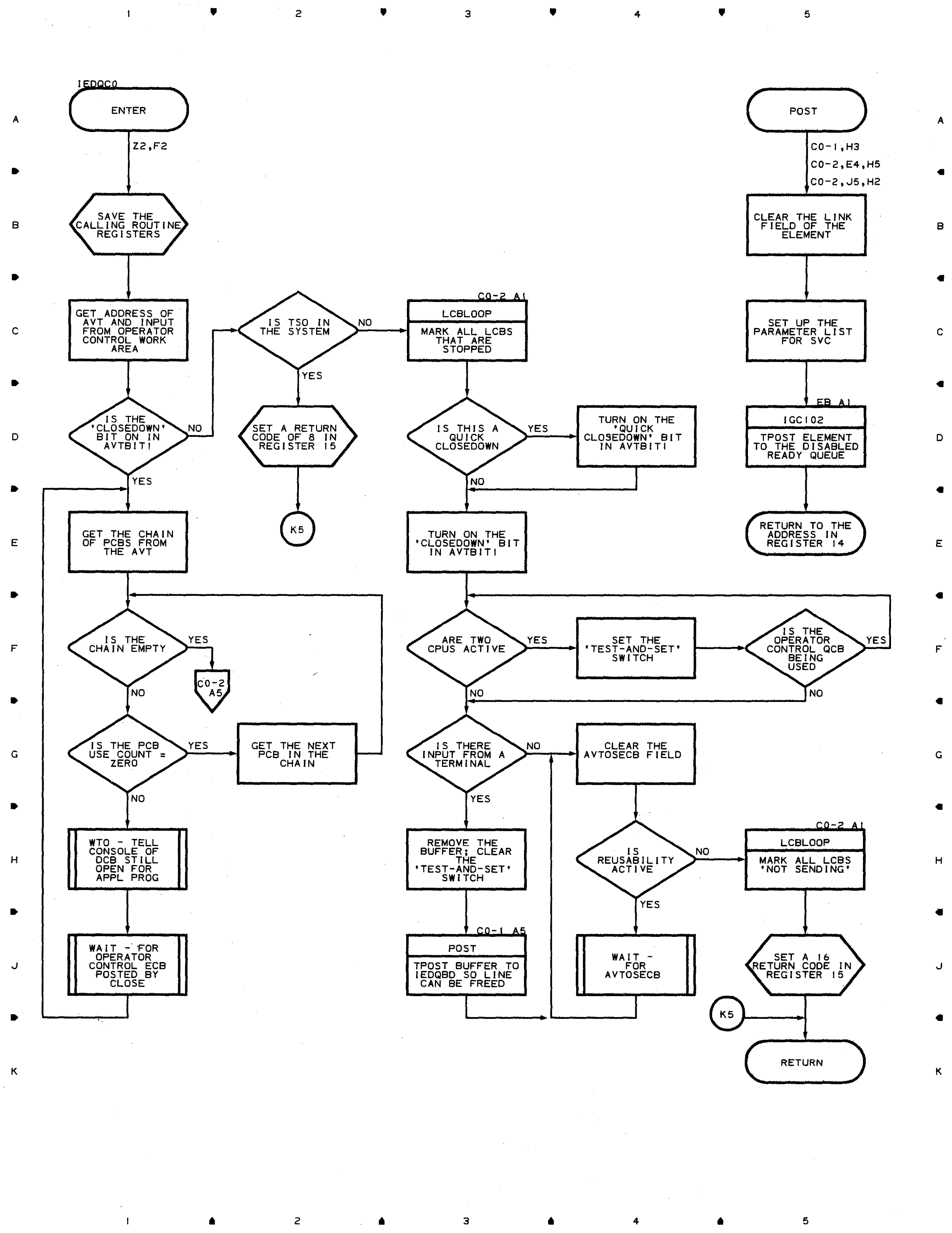

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
CX2 A1
CX2 A4
GET THE ADDRESS OF THE DCB FOR THIS DEB
GET THE SENSE TYPE
IS THIS A TCAM DCB
YES
IS THIS DD OR ADDRESS
DD
IS THIS THE RIGHT OFFSET
NO
C1
CONVERT THE SENSE TYPE TO HEXADECIMAL
C1
NO
ADDR
YES
IS THERE ANOTHER DEB
NO
GET THE ADDRESS OF THE UCB
GET THE ADDRESS OF THE IOB
GET THE SENSE COUNT
YES
CX1 H4
INCREMENT TO THE NEXT DEB
IS THIS THE RIGHT UCB
YES
GET THE ADDRESS OF THE LCB
CONVERT THE SENSE COUNT TO HEXADECIMAL
NO
IS THERE ANOTHER UCB
NO
C1
IS THE SENSE COUNT > 15
YES
BUILD AN ERROR MESSAGE
YES
NO
INCREMENT TO THE NEXT UCB
COMBINE THE SENSE COUNT WITH THE TYPE
PUT THE X'04' ERROR RETURN CODE IN REGISTER 15
PUT THE SENSE COUNT/TYPE IN THE LCB
LINE
IS THIS TERMINAL OR LINE
TERM
PUT THE SENSE COUNT/TYPE IN THE TERMINAL ENTRY
BUILD A RESPONSE MESSAGE
RETURN

Chart CZ CHANGE INTERVAL TYPE ROUTINE



IEDQCZ
ENTER
Z2,F2
ESTABLISH ADDRESSABILITY
POLL NO DATA
YES
H5
NO
POLL
NO
TURN ON THE 'SYSTEM' FLAG
YES
SETPOLL
GET THE ADDRESS OF THE TERMNAME TABLE
LOOP
IS THIS DATA
NO
IS THE SYSTEM INTERVAL = ZERO
YES
H5
IS THIS THE END OF THE TABLE
YES
ERR
PREPARE THE IED016I ERROR MESSAGE
J5
YES
NO
NO
NO
ARE ALL THE CHARACTERS NUMERIC
IS IEDQHI IN THE SYSTEM
NO
REJECT
PREPARE THE IED011I MESSAGE
J5
IS THIS THE RIGHT ENTRY
YES
FOUND
PUT THE DATA FIELD IN THE WORK AREA
YES
YES
NO
CONVERT THE CHARACTERS TO HEXADECIMAL
IS THE 'DELAY' BIT ON
YES
PREPARE THE IED045I MESSAGE
INCREMENT TO THE NEXT ENTRY
IS THE DATA ALL NUMERIC
NO
NO
J2
YES
YES
EQUAL TO ZERO OR > 65,535
BUILD THE REQUEST TO START THE SYSTEM DELAY
CONVERT THE FIELD TO HEXADECIMAL
NO
FB A1
IGC102
SEND THE REQUEST TO THE TCAM MCP
NO
IS IEDQHI IN THE SYSTEM
GET THE LCB ADDRESS THROUGH THE DCB-LCB-TERMINAL ENTRY CHAIN
YES
IS THE LINE OPEN
NO
IS THE FIELD = ZERO OR > 255
H5
H5
YES
NO
YES
INVALID
COMINV
PUT THE INTERVAL IN THE AVT
PREPARE THE IED093I MESSAGE
IS THIS A DIAL LCB
YES
PREPARE THE IED061I MESSAGE
PREPARE THE IED018I MESSAGE
J2
J5
NO
SETGOOD
SETBAD
SET THE X'00' RETURN CODE AND TURN ON THE 'CHECKPOINT' BIT
PUT THE INTERVAL IN THE DCB
SET THE X'04' RETURN CODE IN REGISTER 15
EXIT
EXIT
RETURN
PREPARE THE IED048I MESSAGE
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart C0-1 MCP CLOSEDOWN PROCESSING ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQC0
ENTER
Z2,F2
SAVE THE CALLING ROUTINE REGISTERS
GET ADDRESS OF AVT AND INPUT FROM OPERATOR CONTROL WORK AREA
IS THE 'CLOSEDOWN' BIT ON IN AVTBIT1
NO
YES
GET THE CHAIN OF PCBS FROM THE AVT
IS THE CHAIN EMPTY
YES
C0-2 A5
NO
IS THE PCB USE COUNT = ZERO
YES
GET THE NEXT PCB IN THE CHAIN
NO
WTO - TELL CONSOLE OF DCB STILL OPEN FOR APPL PROG
WAIT - FOR OPERATOR CONTROL ECB POSTED BY CLOSE
IS TSO IN THE SYSTEM
NO
YES
SET A RETURN CODE OF 8 IN REGISTER 15
K5
C0-2 A1
LCBLOOP
MARK ALL LCBS THAT ARE STOPPED
IS THIS A QUICK CLOSEDOWN
YES
TURN ON THE 'QUICK CLOSEDOWN' BIT IN AVTBIT1
NO
TURN ON THE 'CLOSEDOWN' BIT IN AVTBIT1
ARE TWO CPUS ACTIVE
YES
SET THE 'TEST-AND-SET' SWITCH
IS THE OPERATOR CONTROL QCB BEING USED
YES
NO
NO
IS THERE INPUT FROM A TERMINAL
NO
CLEAR THE AVTOSECB FIELD
YES
REMOVE THE BUFFER; CLEAR THE 'TEST-AND-SET' SWITCH
C0-1 A5
POST
TPOST BUFFER TO IEDQBD SO LINE CAN BE FREED
IS REUSABILITY ACTIVE
NO
YES
WAIT - FOR AVTOSECB
C0-2 A1
LCBLOOP
MARK ALL LCBS 'NOT SENDING'
SET A 16 RETURN CODE IN REGISTER 15
K5
RETURN
POST
C0-1,H3
C0-2,E4,H5
C0-2,J5,H2
CLEAR THE LINK FIELD OF THE ELEMENT
SET UP THE PARAMETER LIST FOR SVC
EB A1
IGC102
TPOST ELEMENT TO THE DISABLED READY QUEUE
RETURN TO THE ADDRESS IN REGISTER 14

Chart CO-2 MCP CLOSEDOWN PROCESSING ROUTINE

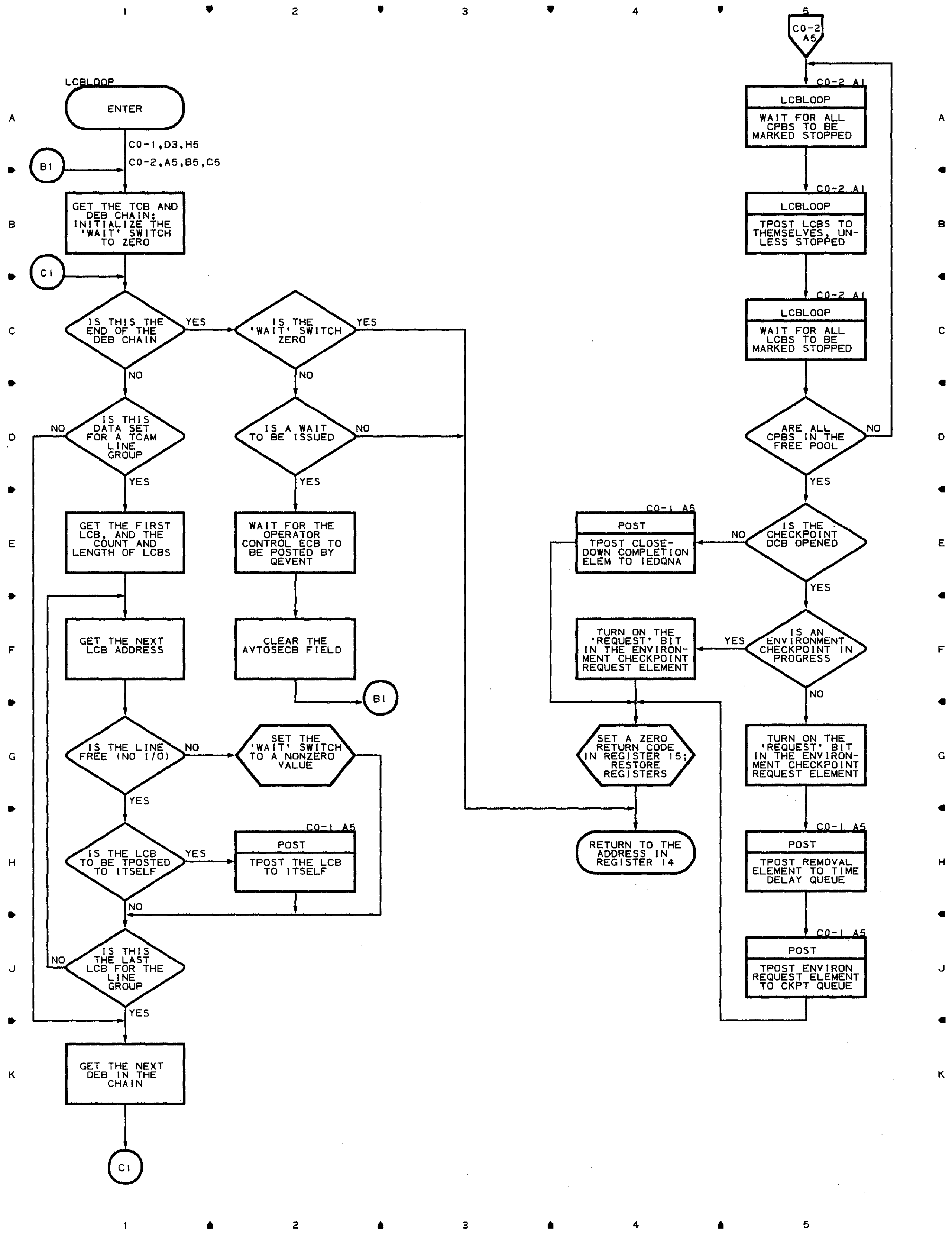
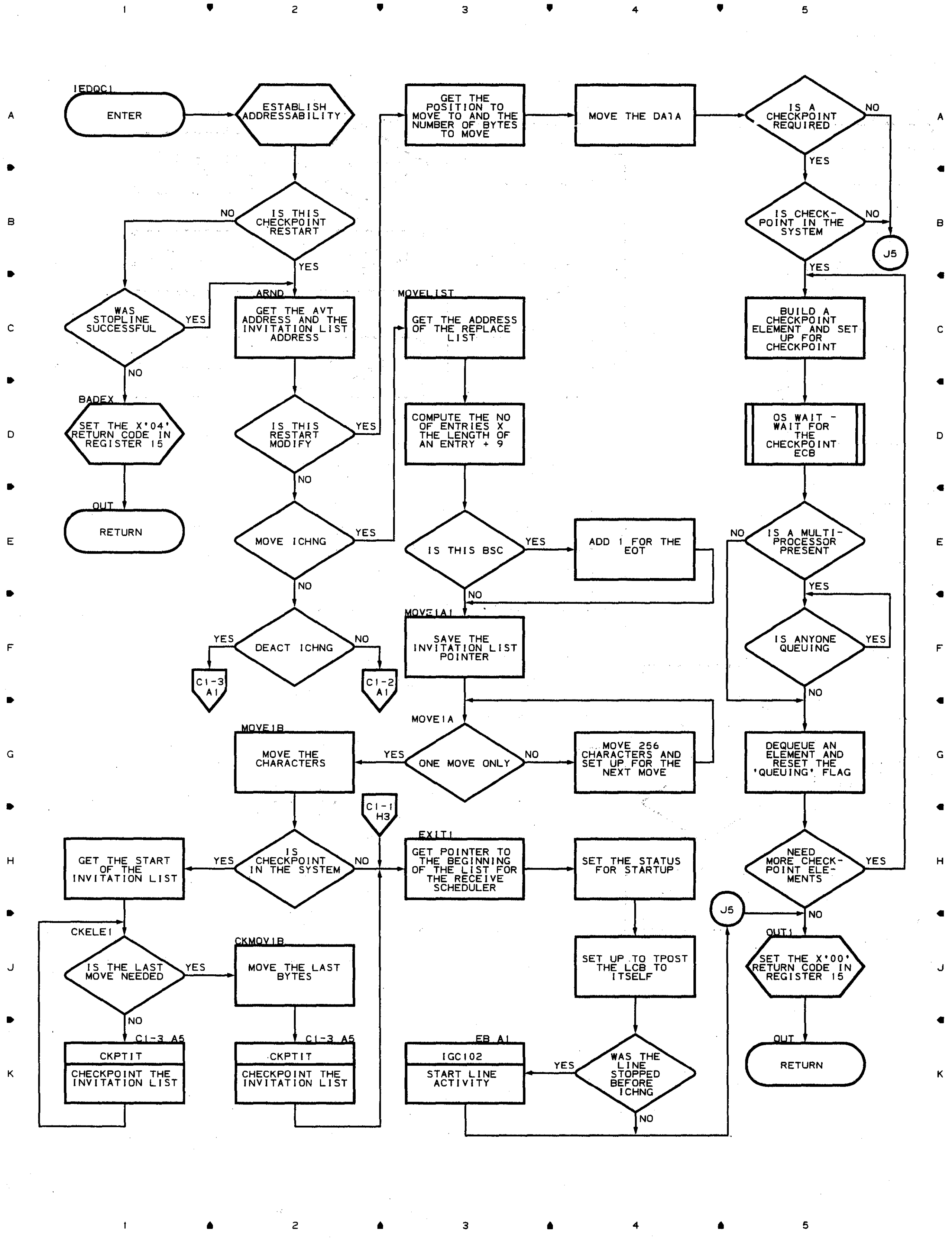

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
LCBLOOP
ENTER
CO-1,D3,H5
CO-2,A5,B5,C5
B1
GET THE TCB AND DEB CHAIN; INITIALIZE THE 'WAIT' SWITCH TO ZERO
C1
IS THIS THE END OF THE DEB CHAIN
YES
NO
IS THE 'WAIT' SWITCH ZERO
YES
NO
IS THIS DATA SET FOR A TCAM LINE GROUP
NO
YES
IS A WAIT TO BE ISSUED
NO
YES
GET THE FIRST LCB, AND THE COUNT AND LENGTH OF LCBS
WAIT FOR THE OPERATOR CONTROL ECB TO BE POSTED BY QEVENT
GET THE NEXT LCB ADDRESS
CLEAR THE AVTOSECB FIELD
B1
IS THE LINE FREE (NO I/O)
NO
YES
SET THE 'WAIT' SWITCH TO A NONZERO VALUE
IS THE LCB TO BE TPOSTED TO ITSELF
YES
NO
CO-1 A5
POST
TPOST THE LCB TO ITSELF
IS THIS THE LAST LCB FOR THE LINE GROUP
NO
YES
GET THE NEXT DEB IN THE CHAIN
C1
CO-2 A5
CO-2 A1
LCBLOOP
WAIT FOR ALL CPBS TO BE MARKED STOPPED
CO-2 A1
LCBLOOP
TPOST LCBS TO THEMSELVES, UNLESS STOPPED
CO-2 A1
LCBLOOP
WAIT FOR ALL LCBS TO BE MARKED STOPPED
ARE ALL CPBS IN THE FREE POOL
NO
YES
IS THE CHECKPOINT DCB OPENED
NO
YES
CO-1 A5
POST
TPOST CLOSEDOWN COMPLETION ELEM TO IEDQNA
IS AN ENVIRONMENT CHECKPOINT IN PROGRESS
YES
NO
TURN ON THE 'REQUEST' BIT IN THE ENVIRONMENT CHECKPOINT REQUEST ELEMENT
SET A ZERO RETURN CODE IN REGISTER 15; RESTORE REGISTERS
TURN ON THE 'REQUEST' BIT IN THE ENVIRONMENT CHECKPOINT REQUEST ELEMENT
RETURN TO THE ADDRESS IN REGISTER 14
CO-1 A5
POST
TPOST REMOVAL ELEMENT TO TIME DELAY QUEUE
CO-1 A5
POST
TPOST ENVIRON REQUEST ELEMENT TO CKPT QUEUE

Chart C1-1 ICHNG PROCESSING ROUTINE



IEDQC1
ENTER
ESTABLISH ADDRESSABILITY
GET THE POSITION TO MOVE TO AND THE NUMBER OF BYTES TO MOVE
MOVE THE DATA
IS A CHECKPOINT REQUIRED
NO
YES
IS THIS CHECKPOINT RESTART
NO
YES
IS CHECK-POINT IN THE SYSTEM
NO
YES
J5
ARND
WAS STOPLINE SUCCESSFUL
YES
NO
GET THE AVT ADDRESS AND THE INVITATION LIST ADDRESS
MOVELIST
GET THE ADDRESS OF THE REPLACE LIST
BUILD A CHECKPOINT ELEMENT AND SET UP FOR CHECKPOINT
BADEX
SET THE X'04' RETURN CODE IN REGISTER 15
IS THIS RESTART MODIFY
YES
NO
COMPUTE THE NO OF ENTRIES X THE LENGTH OF AN ENTRY + 9
OS WAIT - WAIT FOR THE CHECKPOINT ECB
OUT
RETURN
MOVE ICHNG
YES
NO
IS THIS BSC
YES
NO
ADD 1 FOR THE EOT
IS A MULTI-PROCESSOR PRESENT
NO
YES
DEACT ICHNG
YES
NO
C1-3 A1
C1-2 A1
MOVE1A1
SAVE THE INVITATION LIST POINTER
IS ANYONE QUEUING
YES
NO
MOVE1A
MOVE1B
MOVE THE CHARACTERS
ONE MOVE ONLY
YES
NO
MOVE 256 CHARACTERS AND SET UP FOR THE NEXT MOVE
DEQUEUE AN ELEMENT AND RESET THE 'QUEUING' FLAG
C1-1 H3
GET THE START OF THE INVITATION LIST
IS CHECKPOINT IN THE SYSTEM
YES
NO
EXIT1
GET POINTER TO THE BEGINNING OF THE LIST FOR THE RECEIVE SCHEDULER
SET THE STATUS FOR STARTUP
NEED MORE CHECK-POINT ELE-MENTS
YES
NO
J5
CKELE1
IS THE LAST MOVE NEEDED
YES
NO
CKMOV1B
MOVE THE LAST BYTES
SET UP TO TPOST THE LCB TO ITSELF
OUT1
SET THE X'00' RETURN CODE IN REGISTER 15
C1-3 A5
CKPTIT
CHECKPOINT THE INVITATION LIST
C1-3 A5
CKPTIT
CHECKPOINT THE INVITATION LIST
EB A1
IGC102
START LINE ACTIVITY
WAS THE LINE STOPPED BEFORE ICHNG
YES
NO
OUT
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart C1-2 ICHNG PROCESSING ROUTINE

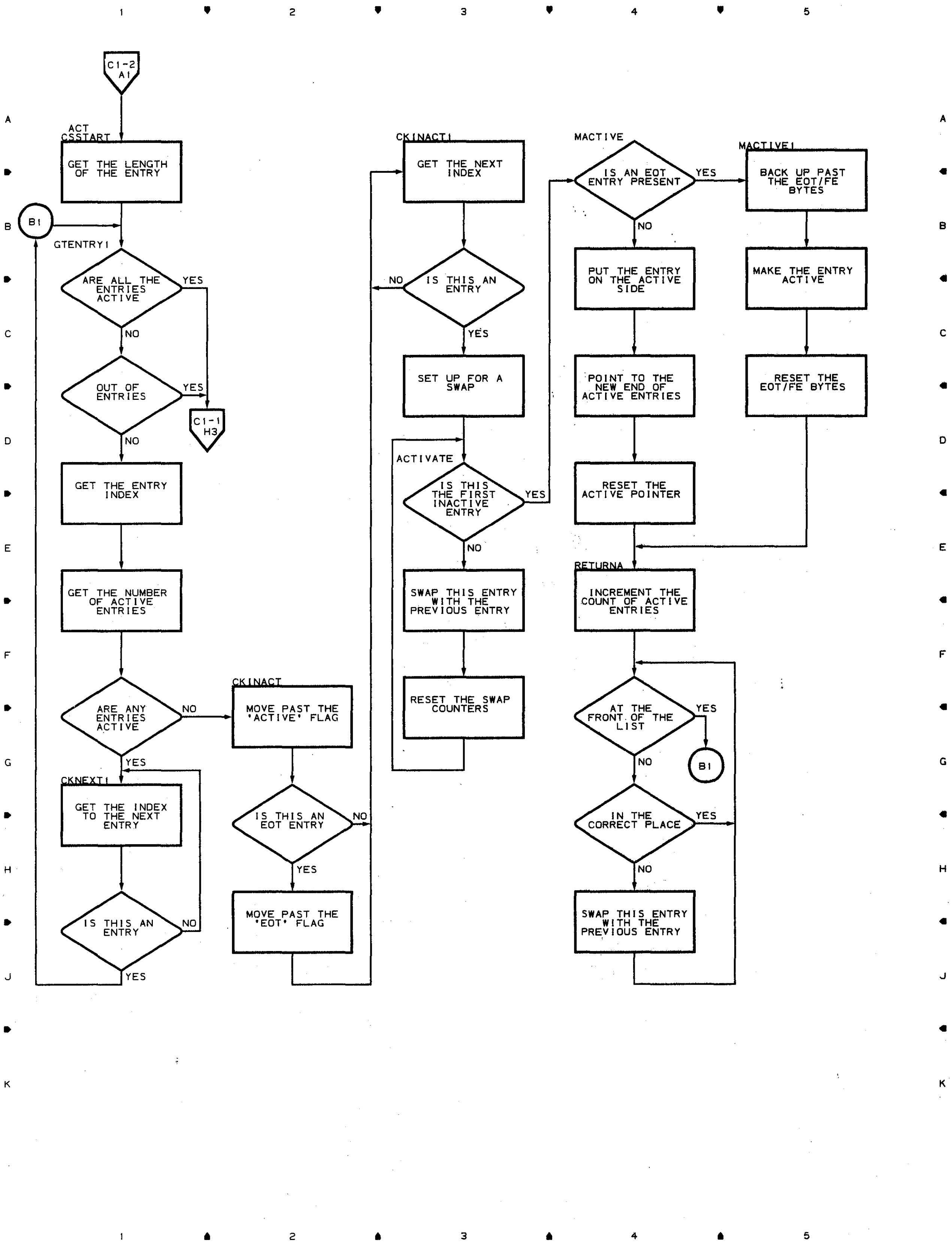


Chart C1-3 ICHNG PROCESSING ROUTINE

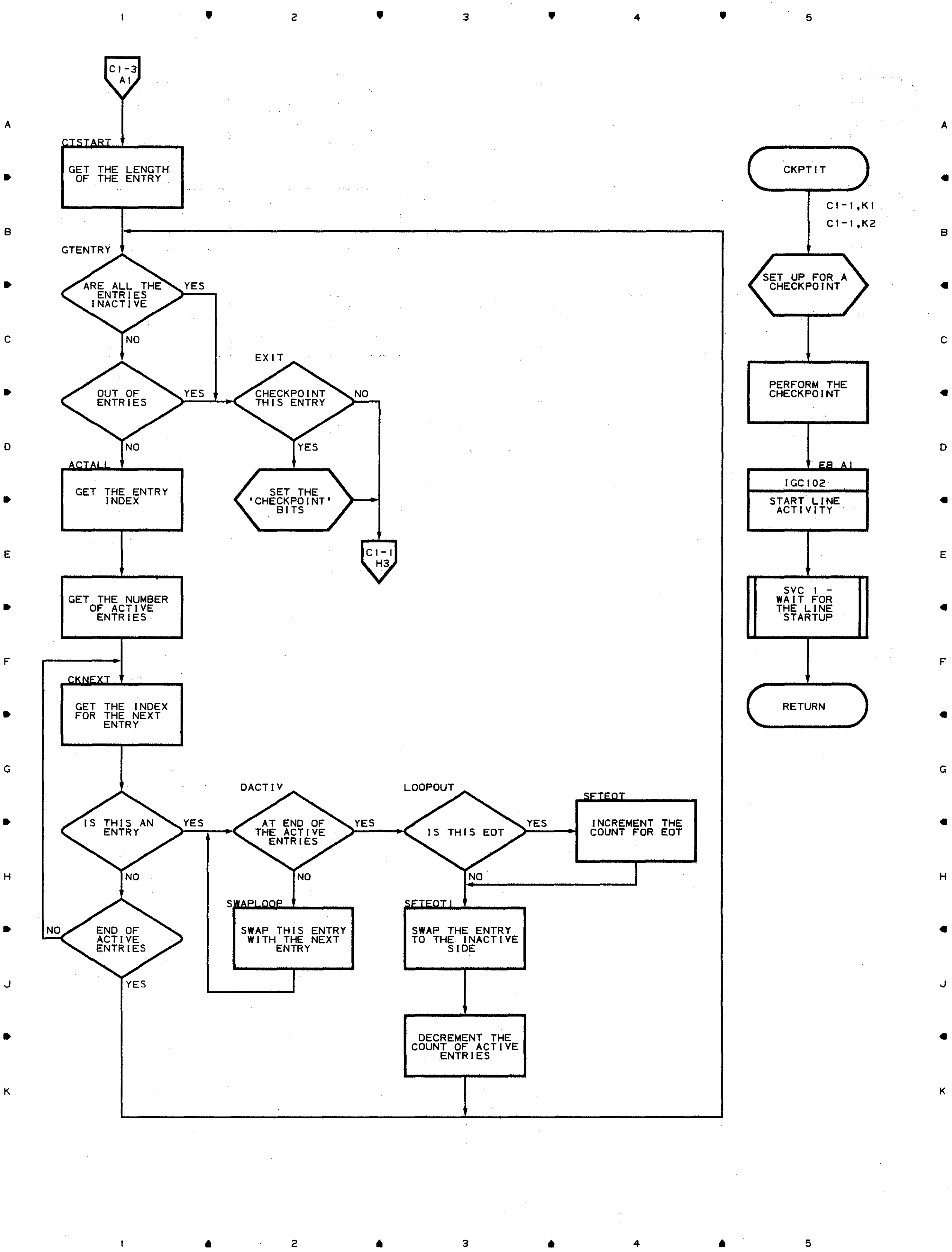

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
C1-3 A1
CTSTART
GET THE LENGTH OF THE ENTRY
GTENTRY
ARE ALL THE ENTRIES INACTIVE
YES
NO
OUT OF ENTRIES
YES
NO
EXIT
CHECKPOINT THIS ENTRY
NO
YES
ACTALL
GET THE ENTRY INDEX
SET THE 'CHECKPOINT' BITS
C1-1 H3
GET THE NUMBER OF ACTIVE ENTRIES
CKNEXT
GET THE INDEX FOR THE NEXT ENTRY
IS THIS AN ENTRY
YES
NO
DACTIV
AT END OF THE ACTIVE ENTRIES
YES
NO
LOOPOUT
IS THIS EOT
YES
NO
SETEOT
INCREMENT THE COUNT FOR EOT
SWAPLOOP
SWAP THIS ENTRY WITH THE NEXT ENTRY
SETEOT1
SWAP THE ENTRY TO THE INACTIVE SIDE
NO
END OF ACTIVE ENTRIES
YES
DECREMENT THE COUNT OF ACTIVE ENTRIES
CKPTIT
C1-1,K1
C1-1,K2
SET UP FOR A CHECKPOINT
PERFORM THE CHECKPOINT
EB A1
IGC102
START LINE ACTIVITY
SVC 1 - WAIT FOR THE LINE STARTUP
RETURN

Chart C2-1 ON-LINE TEST INTERFACE ROUTINE

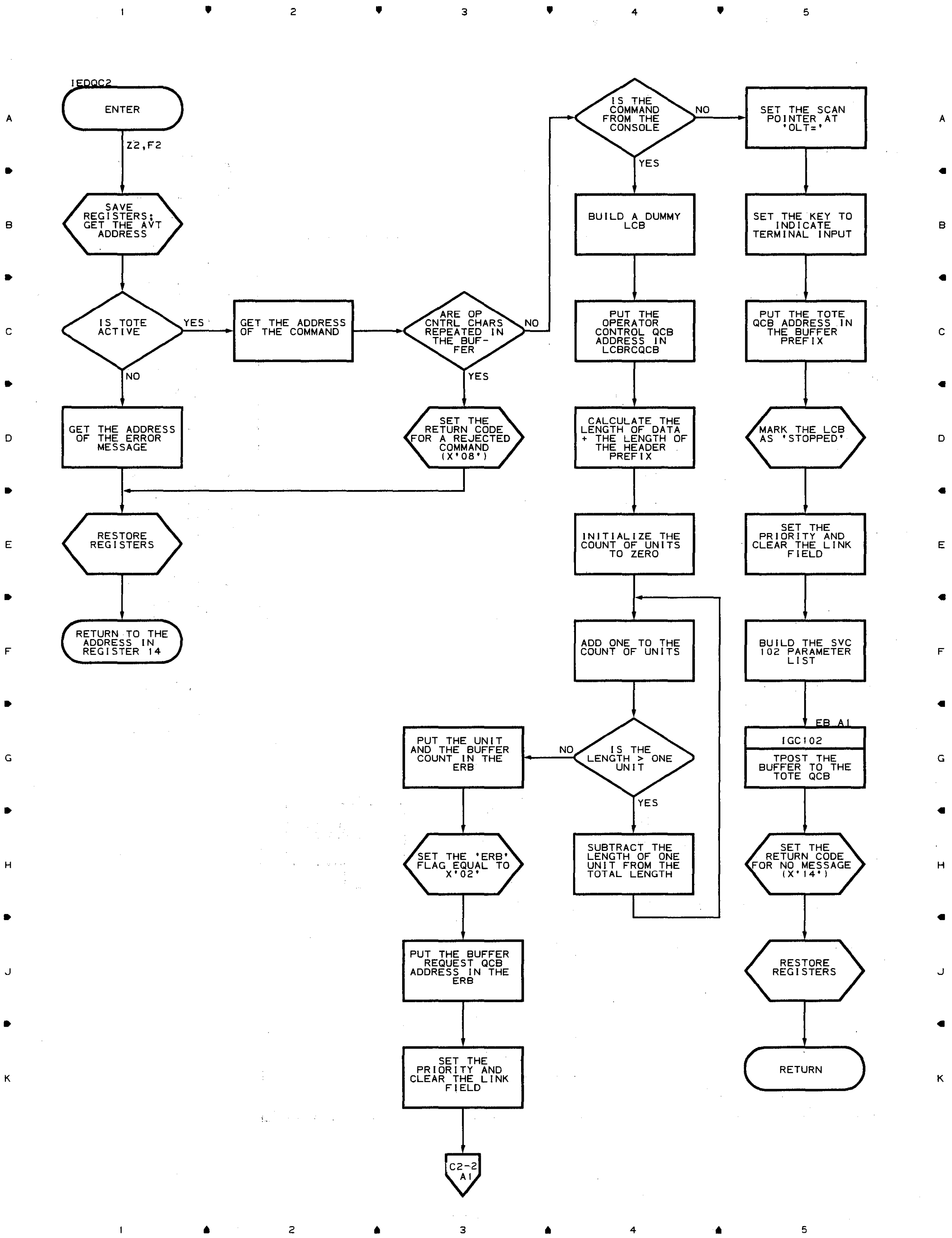

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQC2
ENTER
Z2,F2
SAVE REGISTERS; GET THE AVT ADDRESS
IS TOTE ACTIVE
YES
NO
GET THE ADDRESS OF THE COMMAND
ARE OP CNTRL CHARS REPEATED IN THE BUF-FER
NO
YES
SET THE RETURN CODE FOR A REJECTED COMMAND (X'08')
GET THE ADDRESS OF THE ERROR MESSAGE
RESTORE REGISTERS
RETURN TO THE ADDRESS IN REGISTER 14
IS THE COMMAND FROM THE CONSOLE
NO
YES
BUILD A DUMMY LCB
PUT THE OPERATOR CONTROL QCB ADDRESS IN LCBRCQCB
CALCULATE THE LENGTH OF DATA + THE LENGTH OF THE HEADER PREFIX
INITIALIZE THE COUNT OF UNITS TO ZERO
ADD ONE TO THE COUNT OF UNITS
IS THE LENGTH > ONE UNIT
NO
YES
SUBTRACT THE LENGTH OF ONE UNIT FROM THE TOTAL LENGTH
PUT THE UNIT AND THE BUFFER COUNT IN THE ERB
SET THE 'ERB' FLAG EQUAL TO X'02'
PUT THE BUFFER REQUEST QCB ADDRESS IN THE ERB
SET THE PRIORITY AND CLEAR THE LINK FIELD
C2-2 A1
SET THE SCAN POINTER AT 'OLT='
SET THE KEY TO INDICATE TERMINAL INPUT
PUT THE TOTE QCB ADDRESS IN THE BUFFER PREFIX
MARK THE LCB AS 'STOPPED'
SET THE PRIORITY AND CLEAR THE LINK FIELD
BUILD THE SVC 102 PARAMETER LIST
EB A1
IGC102
TPOST THE BUFFER TO THE TOTE QCB
SET THE RETURN CODE FOR NO MESSAGE (X'14')
RESTORE REGISTERS
RETURN

Chart C2-2 ON-LINE TEST INTERFACE ROUTINE

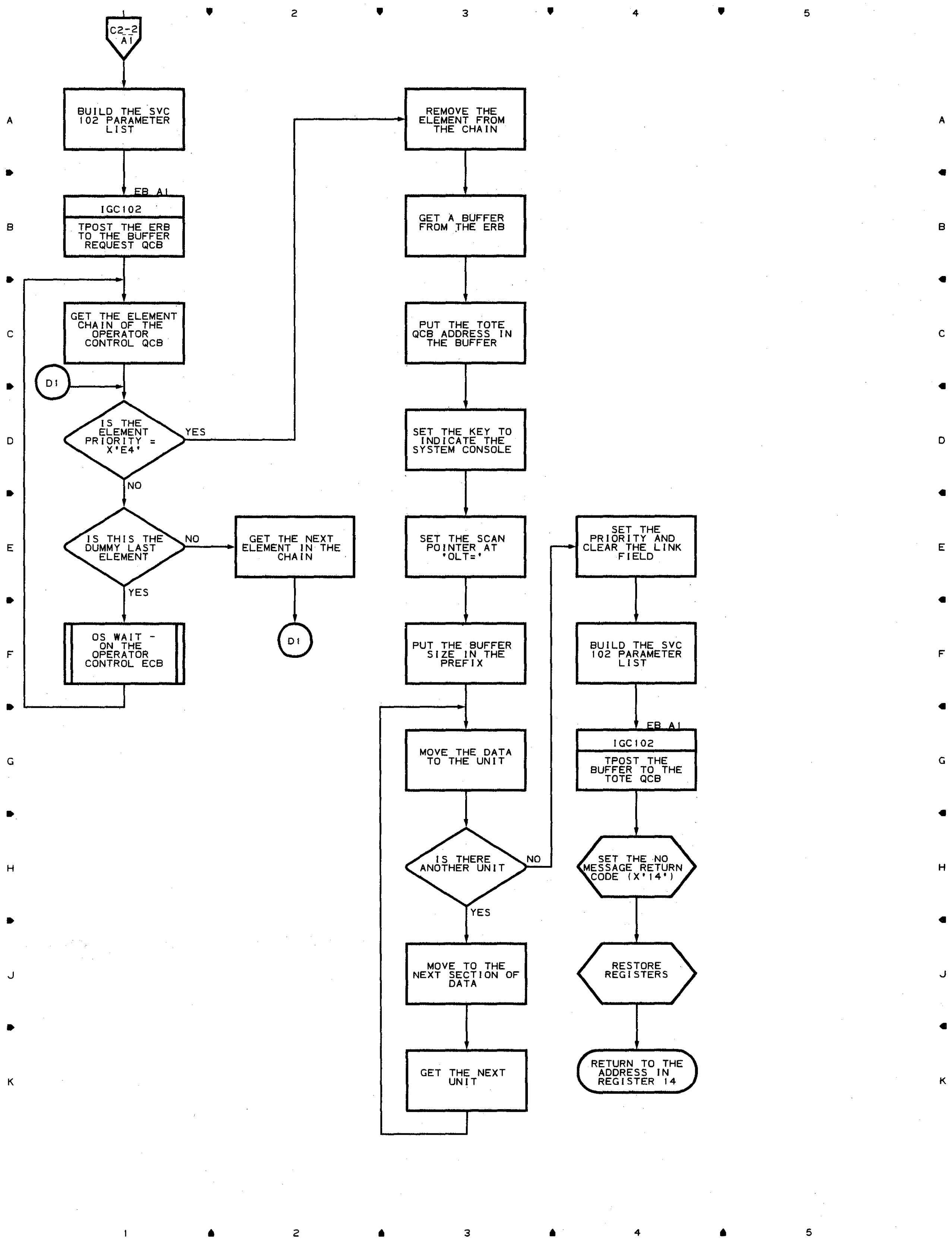

C2-2 A1
BUILD THE SVC 102 PARAMETER LIST
EB A1
IGC102
TPOST THE ERB TO THE BUFFER REQUEST QCB
GET THE ELEMENT CHAIN OF THE OPERATOR CONTROL QCB
D1
IS THE ELEMENT PRIORITY = X'E4'
YES
NO
IS THIS THE DUMMY LAST ELEMENT
NO
YES
GET THE NEXT ELEMENT IN THE CHAIN
D1
OS WAIT - ON THE OPERATOR CONTROL ECB
REMOVE THE ELEMENT FROM THE CHAIN
GET A BUFFER FROM THE ERB
PUT THE TOTE QCB ADDRESS IN THE BUFFER
SET THE KEY TO INDICATE THE SYSTEM CONSOLE
SET THE SCAN POINTER AT 'OLT='
PUT THE BUFFER SIZE IN THE PREFIX
MOVE THE DATA TO THE UNIT
IS THERE ANOTHER UNIT
NO
YES
MOVE TO THE NEXT SECTION OF DATA
GET THE NEXT UNIT
SET THE PRIORITY AND CLEAR THE LINK FIELD
BUILD THE SVC 102 PARAMETER LIST
EB A1
IGC102
TPOST THE BUFFER TO THE TOTE QCB
SET THE NO MESSAGE RETURN CODE (X'14')
RESTORE REGISTERS
RETURN TO THE ADDRESS IN REGISTER 14
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart C3 COPY INVITATION LIST STATUS ROUTINE

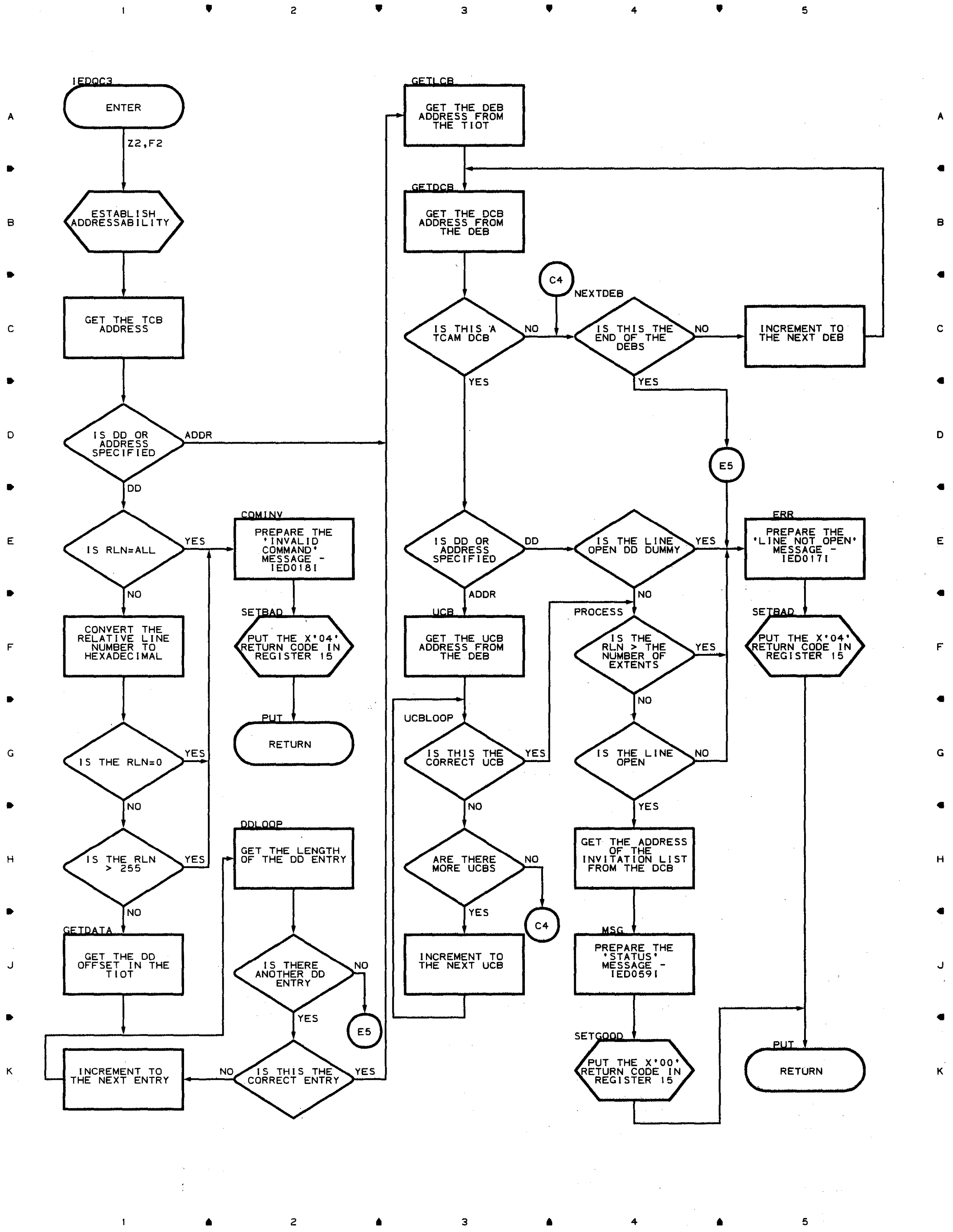

IEDQC3
ENTER
Z2,F2
ESTABLISH ADDRESSABILITY
GET THE TCB ADDRESS
IS DD OR ADDRESS SPECIFIED
ADDR
DD
IS RLN=ALL
YES
NO
CONVERT THE RELATIVE LINE NUMBER TO HEXADECIMAL
IS THE RLN=0
YES
NO
IS THE RLN > 255
YES
NO
GETDATA
GET THE DD OFFSET IN THE TIOT
INCREMENT TO THE NEXT ENTRY
COMINV
PREPARE THE 'INVALID COMMAND' MESSAGE - IED018I
SETBAD
PUT THE X'04' RETURN CODE IN REGISTER 15
PUT
RETURN
DDLOOP
GET THE LENGTH OF THE DD ENTRY
IS THERE ANOTHER DD ENTRY
NO
E5
YES
IS THIS THE CORRECT ENTRY
NO
YES
GETLCB
GET THE DEB ADDRESS FROM THE TIOT
GETDCB
GET THE DCB ADDRESS FROM THE DEB
IS THIS A TCAM DCB
NO
YES
IS DD OR ADDRESS SPECIFIED
DD
ADDR
UCB
GET THE UCB ADDRESS FROM THE DEB
UCBLOOP
IS THIS THE CORRECT UCB
YES
NO
ARE THERE MORE UCBS
NO
C4
YES
INCREMENT TO THE NEXT UCB
C4
NEXTDEB
IS THIS THE END OF THE DEBS
NO
YES
INCREMENT TO THE NEXT DEB
E5
IS THE LINE OPEN DD DUMMY
YES
NO
PROCESS
IS THE RLN > THE NUMBER OF EXTENTS
YES
NO
IS THE LINE OPEN
NO
YES
GET THE ADDRESS OF THE INVITATION LIST FROM THE DCB
MSG
PREPARE THE 'STATUS' MESSAGE - IED059I
SETGOOD
PUT THE X'00' RETURN CODE IN REGISTER 15
ERR
PREPARE THE 'LINE NOT OPEN' MESSAGE - IED017I
SETBAD
PUT THE X'04' RETURN CODE IN REGISTER 15
PUT
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart C6-1 DEBUG SERVICE AID ROUTER

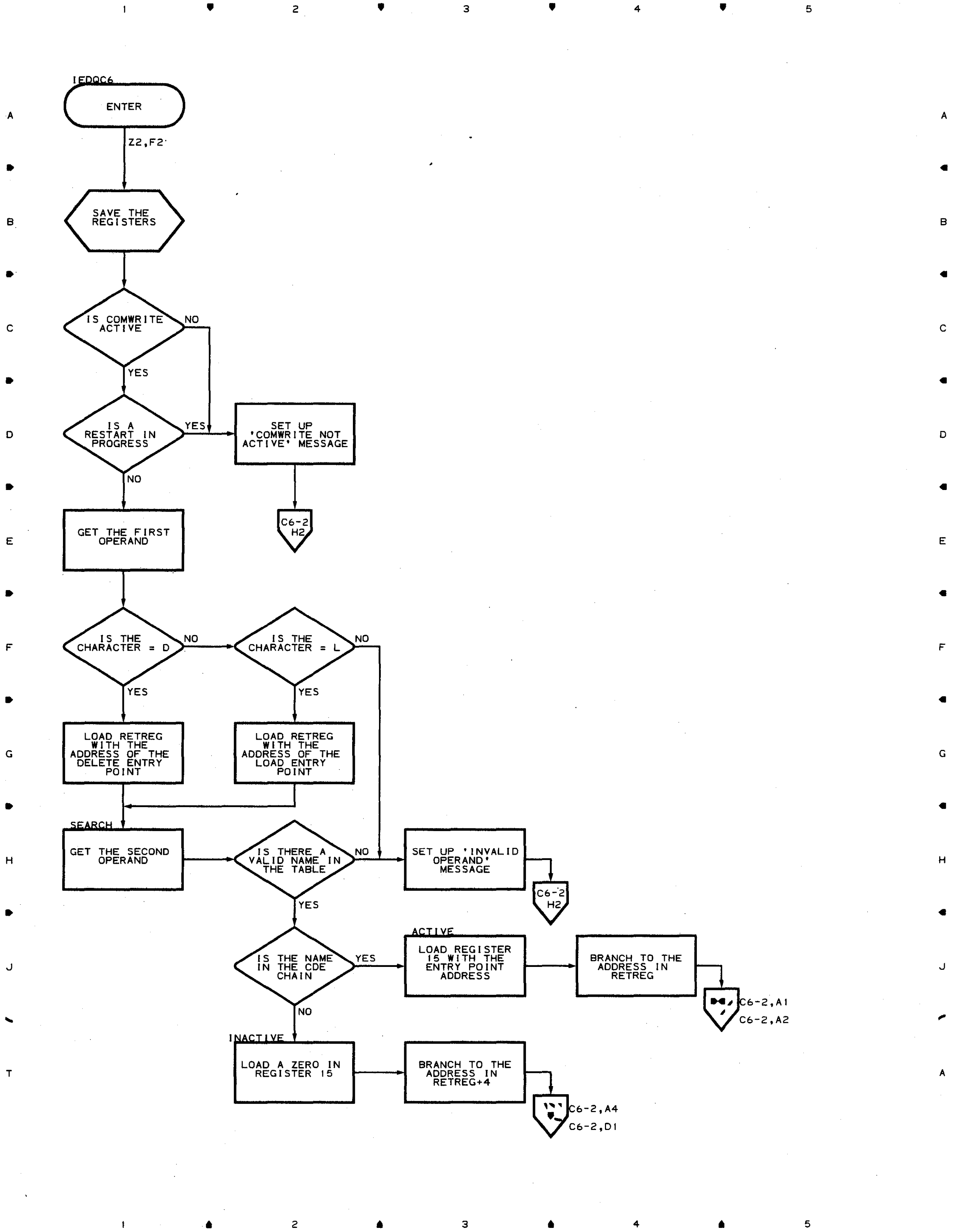
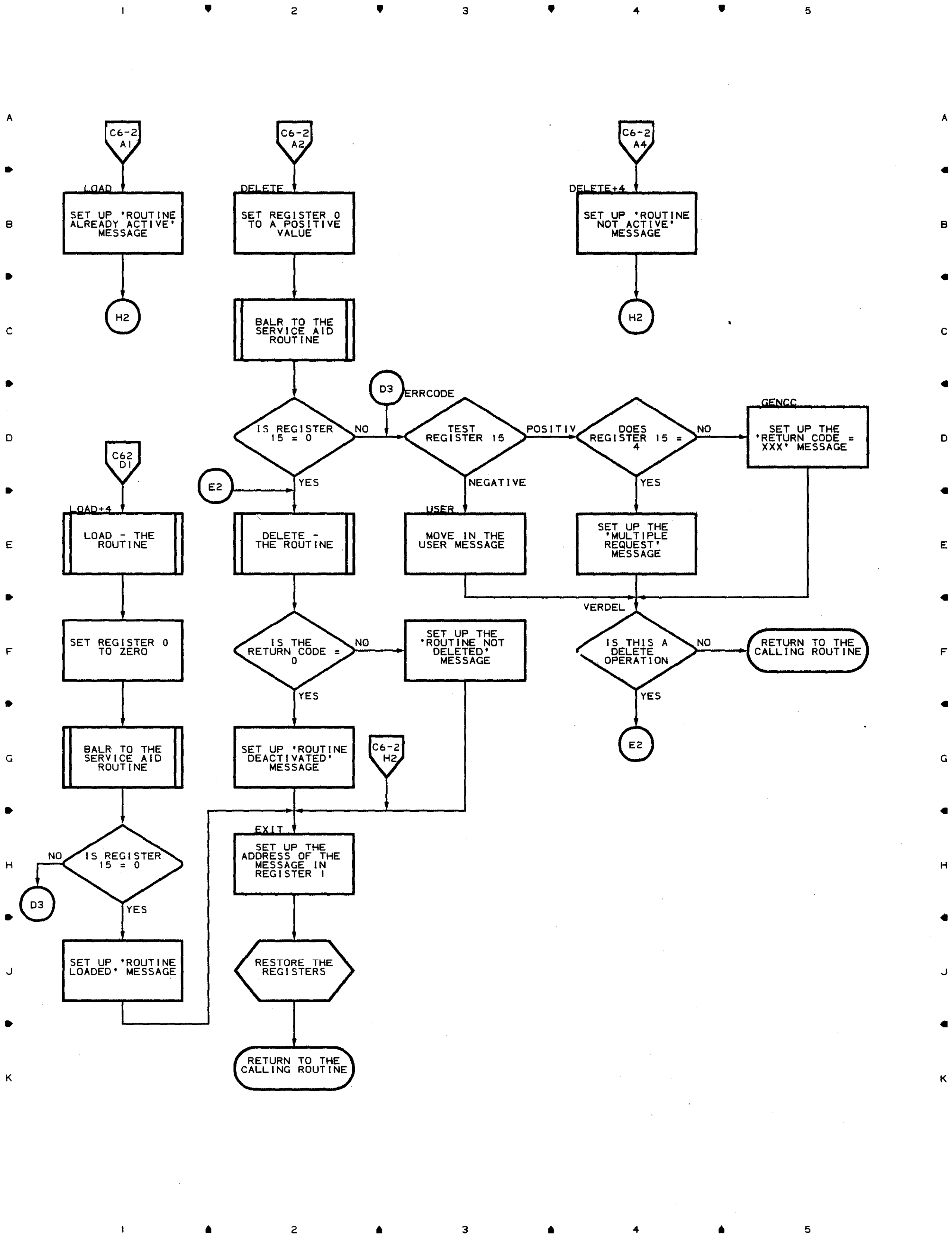

IEDQC6
ENTER
Z2,F2
SAVE THE REGISTERS
IS COMWRITE ACTIVE
NO
YES
IS A RESTART IN PROGRESS
YES
SET UP 'COMWRITE NOT ACTIVE' MESSAGE
NO
C6-2 H2
GET THE FIRST OPERAND
IS THE CHARACTER = D
NO
IS THE CHARACTER = L
NO
YES
YES
LOAD RETREG WITH THE ADDRESS OF THE DELETE ENTRY POINT
LOAD RETREG WITH THE ADDRESS OF THE LOAD ENTRY POINT
SEARCH
GET THE SECOND OPERAND
IS THERE A VALID NAME IN THE TABLE
NO
SET UP 'INVALID OPERAND' MESSAGE
C6-2 H2
YES
ACTIVE
IS THE NAME IN THE CDE CHAIN
YES
LOAD REGISTER 15 WITH THE ENTRY POINT ADDRESS
BRANCH TO THE ADDRESS IN RETREG
C6-2,A1
C6-2,A2
NO
INACTIVE
LOAD A ZERO IN REGISTER 15
BRANCH TO THE ADDRESS IN RETREG+4
C6-2,A4
C6-2,D1
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
A

Chart C6-2 DEBUG SERVICE AID ROUTER



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
C6-2 A1
LOAD
SET UP 'ROUTINE ALREADY ACTIVE' MESSAGE
H2
C6-2 A2
DELETE
SET REGISTER 0 TO A POSITIVE VALUE
BALR TO THE SERVICE AID ROUTINE
C6-2 A4
DELETE+4
SET UP 'ROUTINE NOT ACTIVE' MESSAGE
H2
D3
ERRCODE
IS REGISTER 15 = 0
NO
TEST REGISTER 15
POSITIV
DOES REGISTER 15 = 4
NO
GENCC
SET UP THE 'RETURN CODE = XXX' MESSAGE
C62 D1
E2
YES
NEGATIVE
YES
LOAD+4
LOAD - THE ROUTINE
DELETE - THE ROUTINE
USER
MOVE IN THE USER MESSAGE
SET UP THE 'MULTIPLE REQUEST' MESSAGE
VERDEL
SET REGISTER 0 TO ZERO
IS THE RETURN CODE = 0
NO
SET UP THE 'ROUTINE NOT DELETED' MESSAGE
IS THIS A DELETE OPERATION
NO
RETURN TO THE CALLING ROUTINE
YES
YES
E2
BALR TO THE SERVICE AID ROUTINE
SET UP 'ROUTINE DEACTIVATED' MESSAGE
C6-2 H2
EXIT
NO
IS REGISTER 15 = 0
SET UP THE ADDRESS OF THE MESSAGE IN REGISTER 1
D3
YES
SET UP 'ROUTINE LOADED' MESSAGE
RESTORE THE REGISTERS
RETURN TO THE CALLING ROUTINE
1
2
3
4
5

Chart EB AQCTL SVC 102 ROUTINE

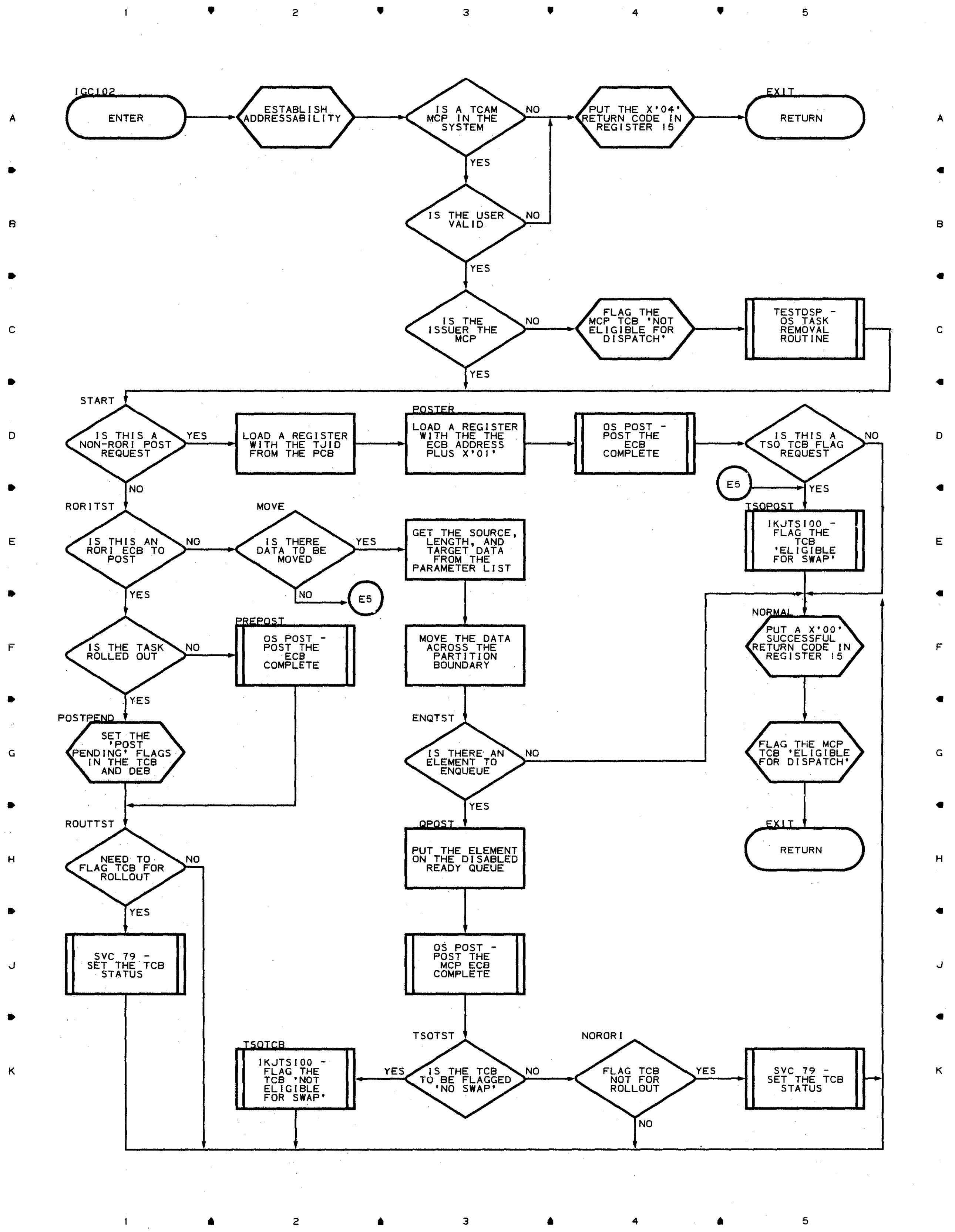

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGC102
ENTER
ESTABLISH ADDRESSABILITY
IS A TCAM MCP IN THE SYSTEM
NO
PUT THE X'04' RETURN CODE IN REGISTER 15
EXIT
RETURN
YES
IS THE USER VALID
NO
YES
IS THE ISSUER THE MCP
NO
FLAG THE MCP TCB 'NOT ELIGIBLE FOR DISPATCH'
TESTDSP - OS TASK REMOVAL ROUTINE
YES
START
IS THIS A NON-RORI POST REQUEST
YES
LOAD A REGISTER WITH THE TJID FROM THE PCB
POSTER
LOAD A REGISTER WITH THE ECB ADDRESS PLUS X'01'
OS POST - POST THE ECB COMPLETE
IS THIS A TSO TCB FLAG REQUEST
NO
NO
E5
YES
RORITST
IS THIS AN RORI ECB TO POST
NO
MOVE
IS THERE DATA TO BE MOVED
YES
GET THE SOURCE, LENGTH, AND TARGET DATA FROM THE PARAMETER LIST
TSOPOST
IKJTSI00 - FLAG THE TCB 'ELIGIBLE FOR SWAP'
YES
NO
E5
IS THE TASK ROLLED OUT
NO
PREPOST
OS POST - POST THE ECB COMPLETE
MOVE THE DATA ACROSS THE PARTITION BOUNDARY
NORMAL
PUT A X'00' SUCCESSFUL RETURN CODE IN REGISTER 15
YES
POSTPEND
SET THE 'POST PENDING' FLAGS IN THE TCB AND DEB
ENQTST
IS THERE AN ELEMENT TO ENQUEUE
NO
FLAG THE MCP TCB 'ELIGIBLE FOR DISPATCH'
YES
ROUTTST
NEED TO FLAG TCB FOR ROLLOUT
NO
QPOST
PUT THE ELEMENT ON THE DISABLED READY QUEUE
EXIT
RETURN
YES
SVC 79 - SET THE TCB STATUS
OS POST - POST THE MCP ECB COMPLETE
TSOTCB
IKJTSI00 - FLAG THE TCB 'NOT ELIGIBLE FOR SWAP'
YES
TSOTST
IS THE TCB TO BE FLAGGED 'NO SWAP'
NO
NORORI
FLAG TCB NOT FOR ROLLOUT
YES
SVC 79 - SET THE TCB STATUS
NO

Chart EC PUT SCHEDULER

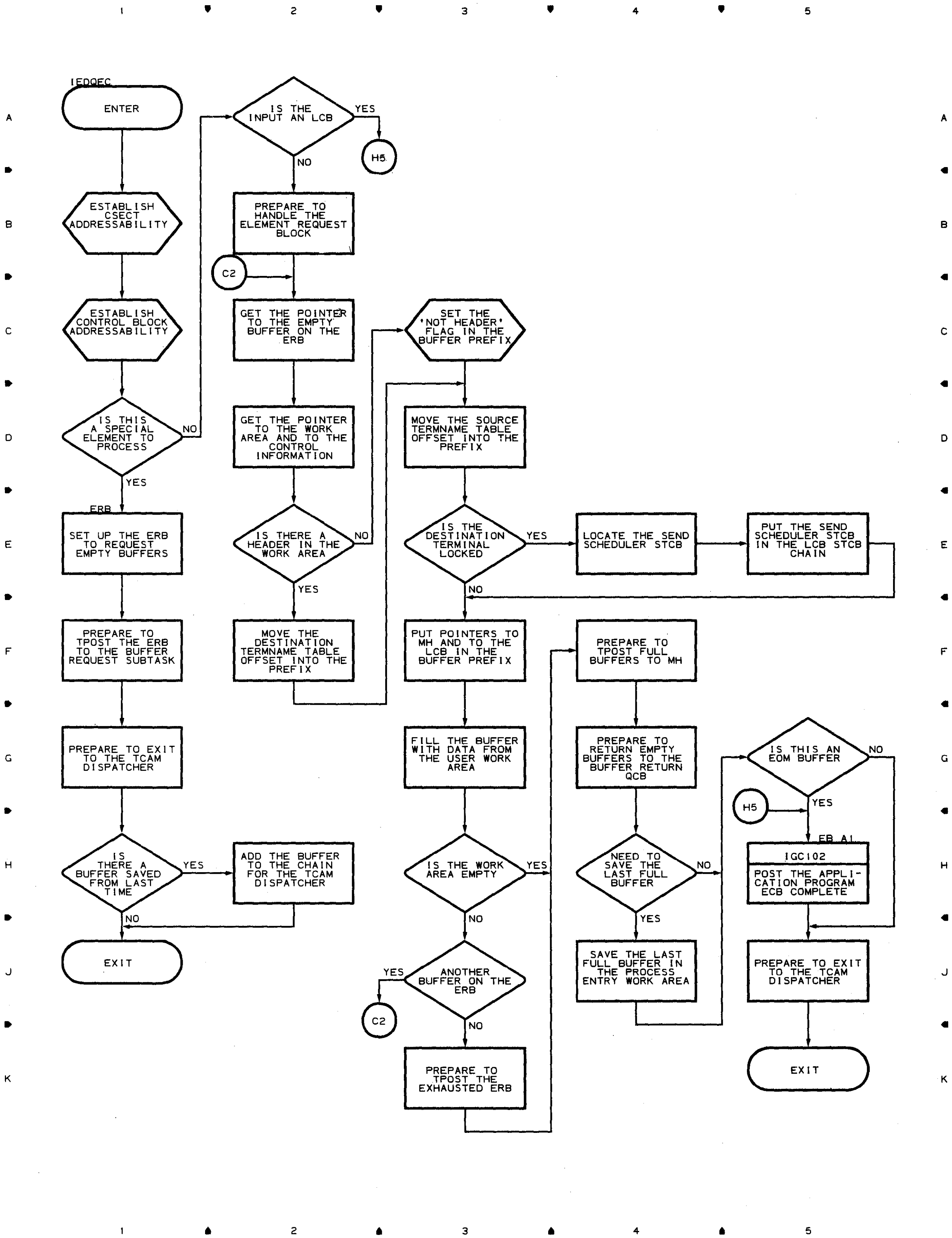

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQEC
ENTER
ESTABLISH CSECT ADDRESSABILITY
ESTABLISH CONTROL BLOCK ADDRESSABILITY
IS THIS A SPECIAL ELEMENT TO PROCESS
NO
YES
ERB
SET UP THE ERB TO REQUEST EMPTY BUFFERS
PREPARE TO TPOST THE ERB TO THE BUFFER REQUEST SUBTASK
PREPARE TO EXIT TO THE TCAM DISPATCHER
IS THERE A BUFFER SAVED FROM LAST TIME
YES
NO
EXIT
IS THE INPUT AN LCB
YES
H5
NO
PREPARE TO HANDLE THE ELEMENT REQUEST BLOCK
C2
GET THE POINTER TO THE EMPTY BUFFER ON THE ERB
GET THE POINTER TO THE WORK AREA AND TO THE CONTROL INFORMATION
IS THERE A HEADER IN THE WORK AREA
NO
YES
MOVE THE DESTINATION TERMNAME TABLE OFFSET INTO THE PREFIX
ADD THE BUFFER TO THE CHAIN FOR THE TCAM DISPATCHER
SET THE 'NOT HEADER' FLAG IN THE BUFFER PREFIX
MOVE THE SOURCE TERMNAME TABLE OFFSET INTO THE PREFIX
IS THE DESTINATION TERMINAL LOCKED
YES
NO
PUT POINTERS TO MH AND TO THE LCB IN THE BUFFER PREFIX
FILL THE BUFFER WITH DATA FROM THE USER WORK AREA
IS THE WORK AREA EMPTY
YES
NO
ANOTHER BUFFER ON THE ERB
YES
C2
NO
PREPARE TO TPOST THE EXHAUSTED ERB
LOCATE THE SEND SCHEDULER STCB
PREPARE TO TPOST FULL BUFFERS TO MH
PREPARE TO RETURN EMPTY BUFFERS TO THE BUFFER RETURN QCB
NEED TO SAVE THE LAST FULL BUFFER
NO
YES
SAVE THE LAST FULL BUFFER IN THE PROCESS ENTRY WORK AREA
PUT THE SEND SCHEDULER STCB IN THE LCB STCB CHAIN
IS THIS AN EOM BUFFER
NO
YES
H5
EB A1
IGC102
POST THE APPLI-CATION PROGRAM ECB COMPLETE
PREPARE TO EXIT TO THE TCAM DISPATCHER
EXIT

Chart ES RETRIEVE SERVICE ROUTINE

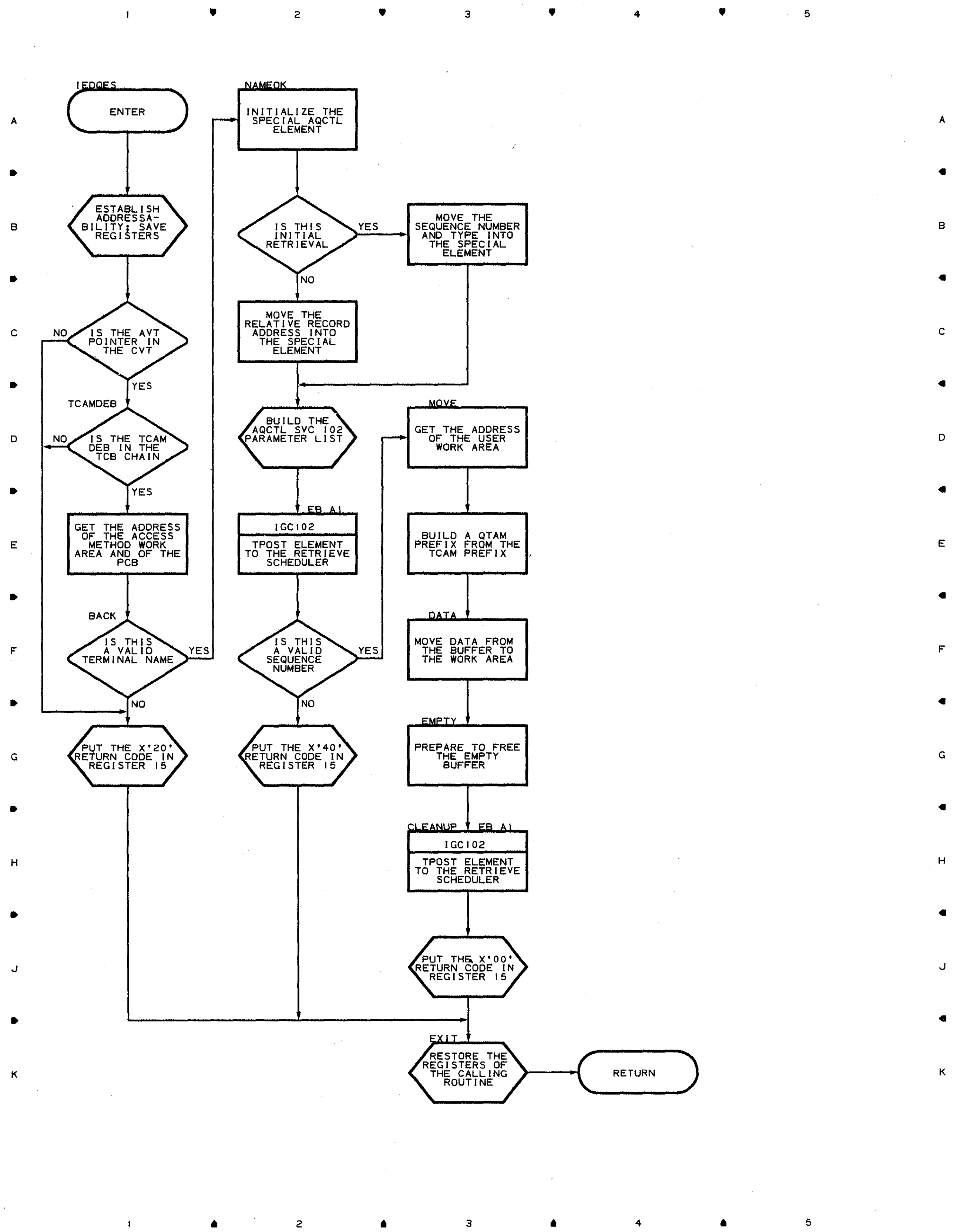

IEDQES
ENTER
ESTABLISH ADDRESSABILITY; SAVE REGISTERS
IS THE AVT POINTER IN THE CVT
NO
YES
TCAMDEB
IS THE TCAM DEB IN THE TCB CHAIN
NO
YES
GET THE ADDRESS OF THE ACCESS METHOD WORK AREA AND OF THE PCB
BACK
IS THIS A VALID TERMINAL NAME
YES
NO
PUT THE X'20' RETURN CODE IN REGISTER 15
NAMEOK
INITIALIZE THE SPECIAL AQCTL ELEMENT
IS THIS INITIAL RETRIEVAL
YES
NO
MOVE THE SEQUENCE NUMBER AND TYPE INTO THE SPECIAL ELEMENT
MOVE THE RELATIVE RECORD ADDRESS INTO THE SPECIAL ELEMENT
BUILD THE AQCTL SVC 102 PARAMETER LIST
EB A1
IGC102
TPOST ELEMENT TO THE RETRIEVE SCHEDULER
IS THIS A VALID SEQUENCE NUMBER
YES
NO
PUT THE X'40' RETURN CODE IN REGISTER 15
MOVE
GET THE ADDRESS OF THE USER WORK AREA
BUILD A QTAM PREFIX FROM THE TCAM PREFIX
DATA
MOVE DATA FROM THE BUFFER TO THE WORK AREA
EMPTY
PREPARE TO FREE THE EMPTY BUFFER
CLEANUP
EB A1
IGC102
TPOST ELEMENT TO THE RETRIEVE SCHEDULER
PUT THE X'00' RETURN CODE IN REGISTER 15
EXIT
RESTORE THE REGISTERS OF THE CALLING ROUTINE
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart ET OPERATOR CONTROL/APPLICATION PROGRAM INTERFACE ROUTINE

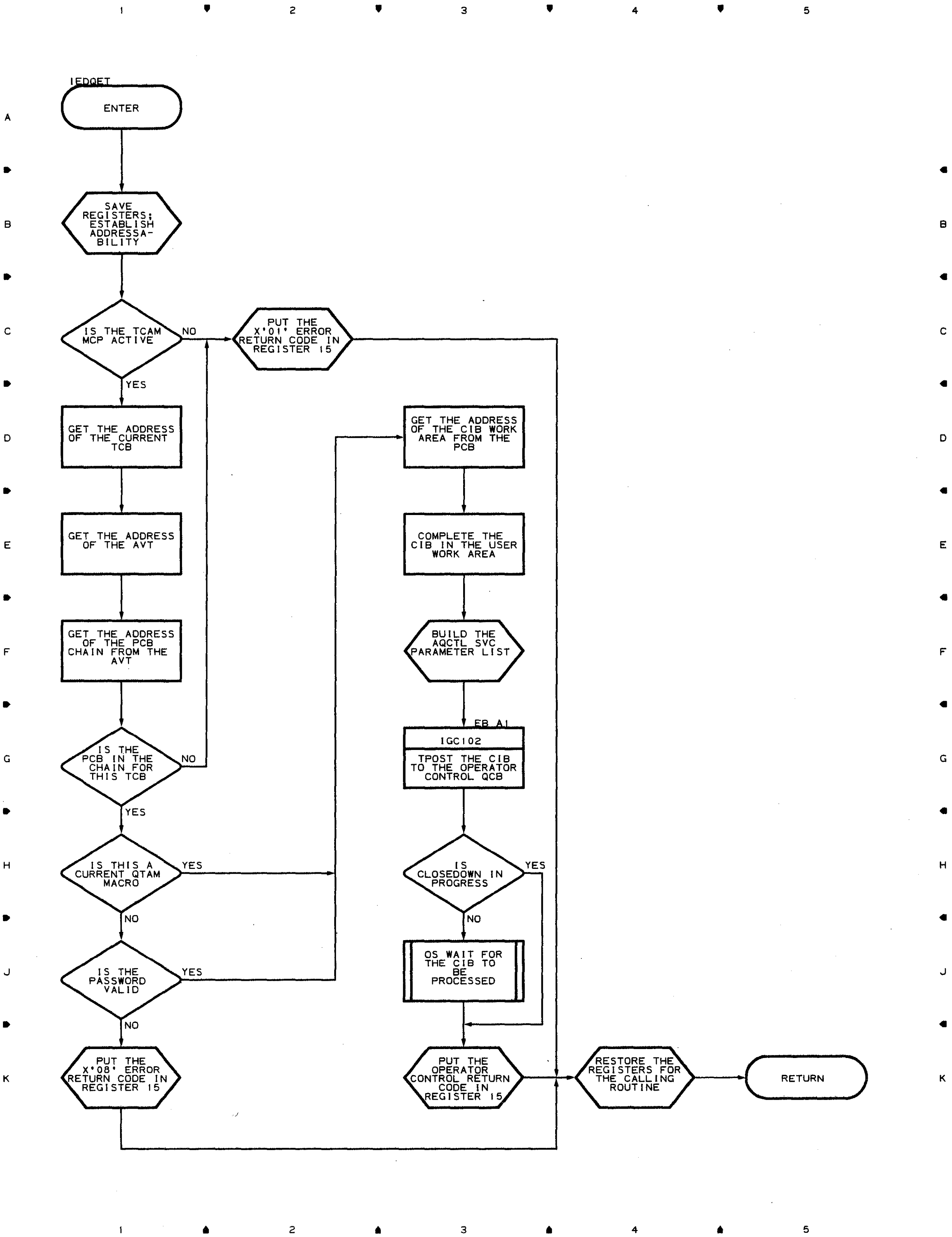

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQET
ENTER
SAVE REGISTERS; ESTABLISH ADDRESSA-BILITY
IS THE TCAM MCP ACTIVE
NO
YES
PUT THE X'01' ERROR RETURN CODE IN REGISTER 15
GET THE ADDRESS OF THE CURRENT TCB
GET THE ADDRESS OF THE AVT
GET THE ADDRESS OF THE PCB CHAIN FROM THE AVT
IS THE PCB IN THE CHAIN FOR THIS TCB
NO
YES
IS THIS A CURRENT QTAM MACRO
YES
NO
IS THE PASSWORD VALID
YES
NO
PUT THE X'08' ERROR RETURN CODE IN REGISTER 15
GET THE ADDRESS OF THE CIB WORK AREA FROM THE PCB
COMPLETE THE CIB IN THE USER WORK AREA
BUILD THE AQCTL SVC PARAMETER LIST
EB A1
IGC102
TPOST THE CIB TO THE OPERATOR CONTROL QCB
IS CLOSEDOWN IN PROGRESS
YES
NO
OS WAIT FOR THE CIB TO BE PROCESSED
PUT THE OPERATOR CONTROL RETURN CODE IN REGISTER 15
RESTORE THE REGISTERS FOR THE CALLING ROUTINE
RETURN

Chart EU OPEN/CLOSE SUBTASK

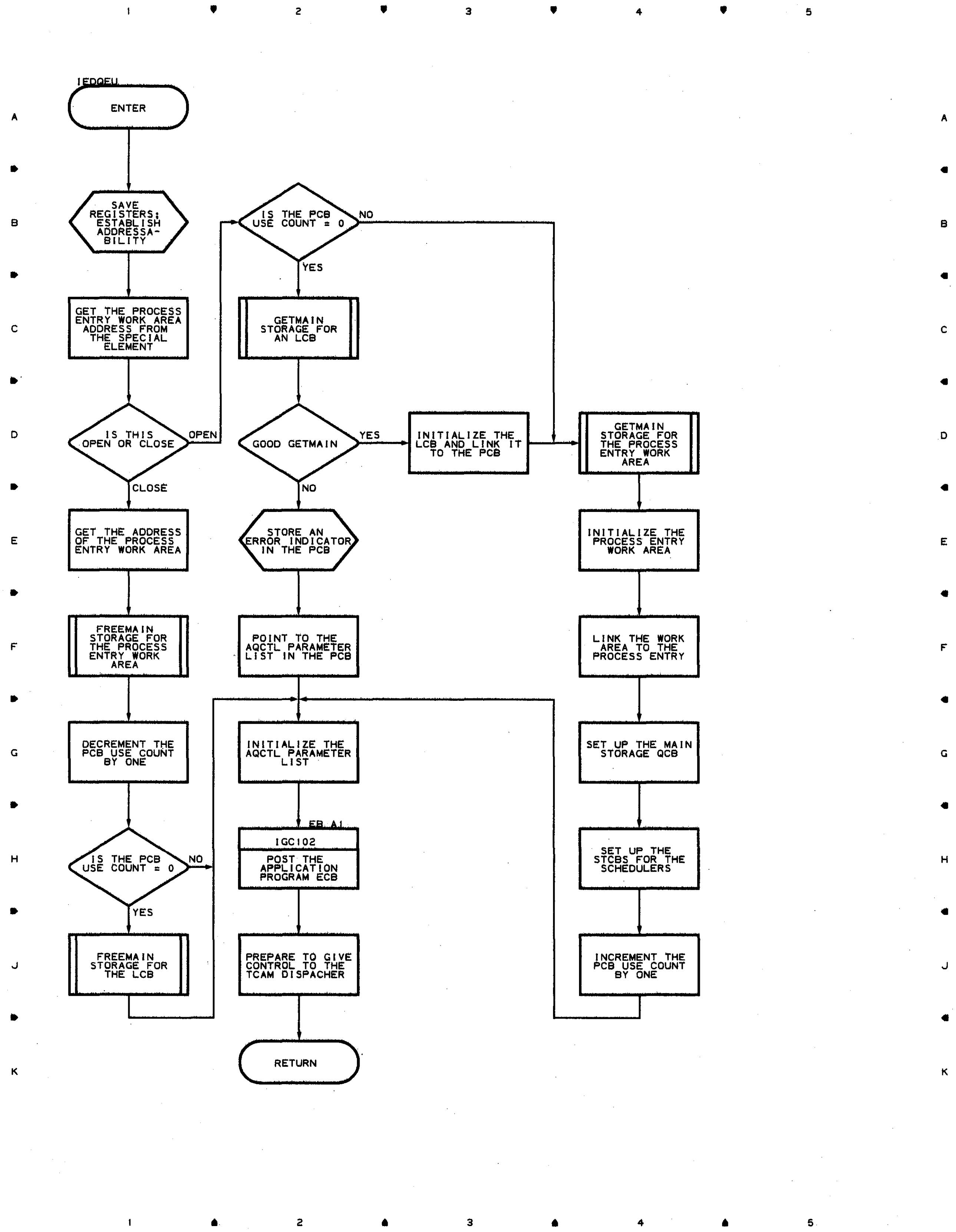

IEDQEU
ENTER
SAVE REGISTERS; ESTABLISH ADDRESSA-BILITY
GET THE PROCESS ENTRY WORK AREA ADDRESS FROM THE SPECIAL ELEMENT
IS THIS OPEN OR CLOSE
OPEN
CLOSE
GET THE ADDRESS OF THE PROCESS ENTRY WORK AREA
FREEMAIN STORAGE FOR THE PROCESS ENTRY WORK AREA
DECREMENT THE PCB USE COUNT BY ONE
IS THE PCB USE COUNT = 0
NO
YES
FREEMAIN STORAGE FOR THE LCB
IS THE PCB USE COUNT = 0
NO
YES
GETMAIN STORAGE FOR AN LCB
GOOD GETMAIN
YES
NO
STORE AN ERROR INDICATOR IN THE PCB
POINT TO THE AQCTL PARAMETER LIST IN THE PCB
INITIALIZE THE AQCTL PARAMETER LIST
EB A1
IGC102
POST THE APPLICATION PROGRAM ECB
PREPARE TO GIVE CONTROL TO THE TCAM DISPACHER
RETURN
INITIALIZE THE LCB AND LINK IT TO THE PCB
GETMAIN STORAGE FOR THE PROCESS ENTRY WORK AREA
INITIALIZE THE PROCESS ENTRY WORK AREA
LINK THE WORK AREA TO THE PROCESS ENTRY
SET UP THE MAIN STORAGE QCB
SET UP THE STCBS FOR THE SCHEDULERS
INCREMENT THE PCB USE COUNT BY ONE
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart EW-1 GET SCHEDULER

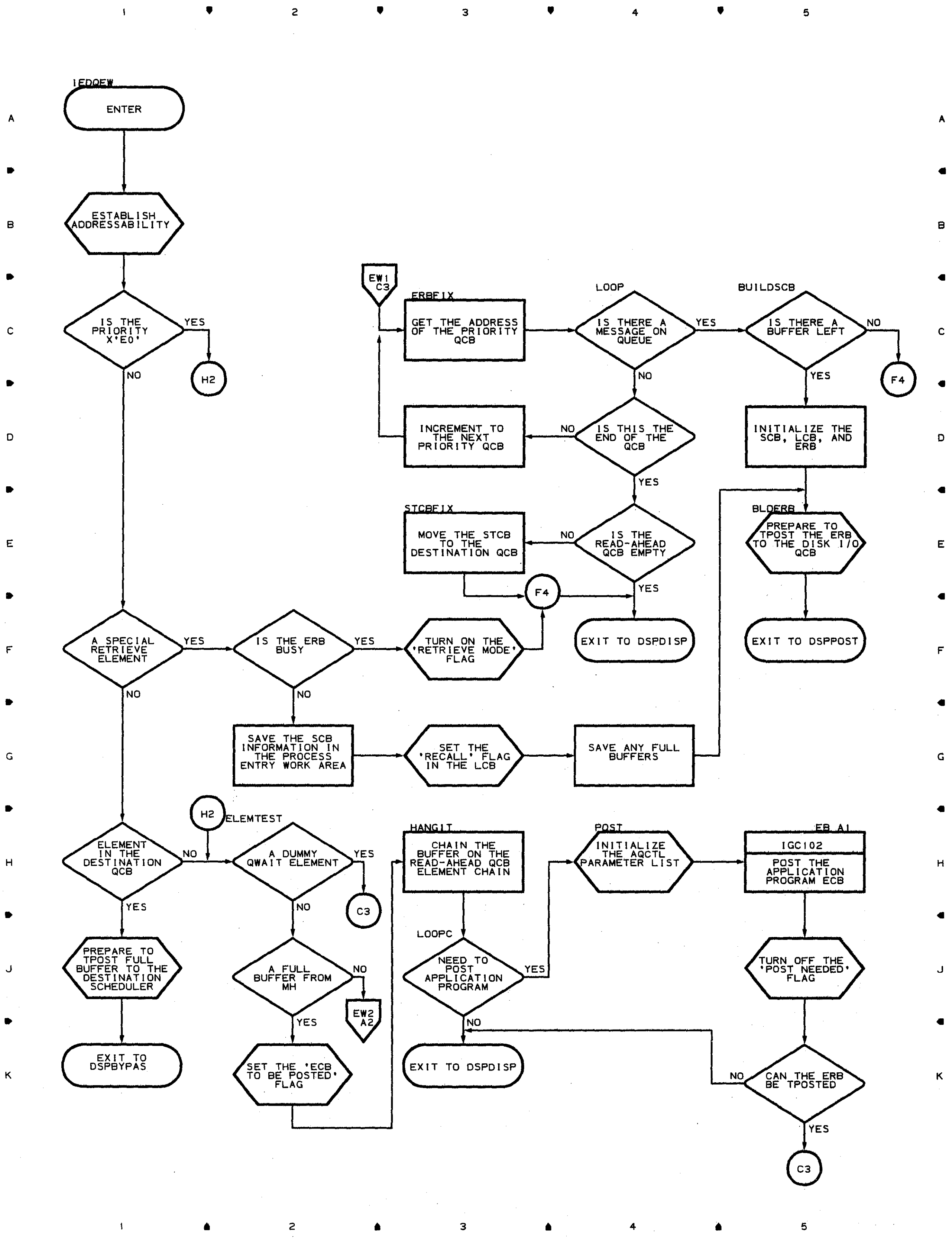
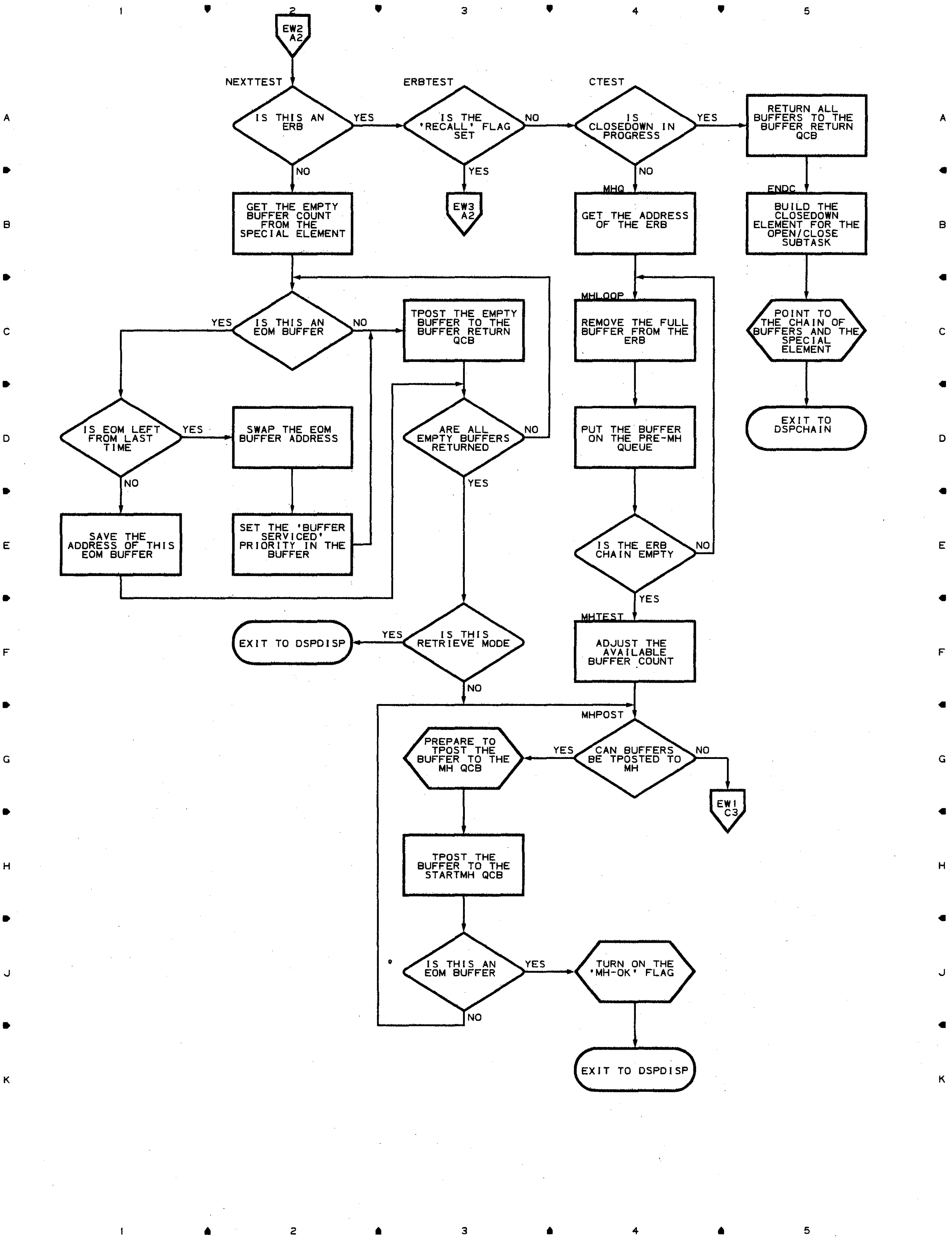

IEDQEW
ENTER
ESTABLISH ADDRESSABILITY
IS THE PRIORITY X'E0'
YES
NO
H2
EW1 C3
ERBFIX
GET THE ADDRESS OF THE PRIORITY QCB
LOOP
IS THERE A MESSAGE ON QUEUE
YES
NO
BUILDSCB
IS THERE A BUFFER LEFT
NO
YES
F4
INCREMENT TO THE NEXT PRIORITY QCB
NO
IS THIS THE END OF THE QCB
YES
INITIALIZE THE SCB, LCB, AND ERB
STCBFIX
MOVE THE STCB TO THE DESTINATION QCB
NO
IS THE READ-AHEAD QCB EMPTY
YES
BLDERB
PREPARE TO TPOST THE ERB TO THE DISK I/O QCB
F4
A SPECIAL RETRIEVE ELEMENT
YES
NO
IS THE ERB BUSY
YES
NO
TURN ON THE 'RETRIEVE MODE' FLAG
EXIT TO DSPDISP
EXIT TO DSPPOST
SAVE THE SCB INFORMATION IN THE PROCESS ENTRY WORK AREA
SET THE 'RECALL' FLAG IN THE LCB
SAVE ANY FULL BUFFERS
H2
ELEMTEST
ELEMENT IN THE DESTINATION QCB
NO
YES
A DUMMY QWAIT ELEMENT
YES
NO
C3
HANGIT
CHAIN THE BUFFER ON THE READ-AHEAD QCB ELEMENT CHAIN
POST
INITIALIZE THE AQCTL PARAMETER LIST
EB A1
IGC102
POST THE APPLICATION PROGRAM ECB
PREPARE TO TPOST FULL BUFFER TO THE DESTINATION SCHEDULER
A FULL BUFFER FROM MH
NO
YES
EW2 A2
LOOPC
NEED TO POST APPLICATION PROGRAM
YES
NO
TURN OFF THE 'POST NEEDED' FLAG
EXIT TO DSPBYPAS
SET THE 'ECB TO BE POSTED' FLAG
EXIT TO DSPDISP
CAN THE ERB BE TPOSTED
NO
YES
C3
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart EW-2 GET SCHEDULER



1
2
3
4
5
EW2 A2
NEXTTEST
IS THIS AN ERB
YES
NO
ERBTEST
IS THE 'RECALL' FLAG SET
NO
YES
EW3 A2
CTEST
IS CLOSEDOWN IN PROGRESS
YES
NO
RETURN ALL BUFFERS TO THE BUFFER RETURN QCB
ENDC
BUILD THE CLOSEDOWN ELEMENT FOR THE OPEN/CLOSE SUBTASK
POINT TO THE CHAIN OF BUFFERS AND THE SPECIAL ELEMENT
EXIT TO DSPCHAIN
GET THE EMPTY BUFFER COUNT FROM THE SPECIAL ELEMENT
MHQ
GET THE ADDRESS OF THE ERB
MHLOOP
REMOVE THE FULL BUFFER FROM THE ERB
PUT THE BUFFER ON THE PRE-MH QUEUE
IS THE ERB CHAIN EMPTY
NO
YES
MHTEST
ADJUST THE AVAILABLE BUFFER COUNT
IS THIS AN EOM BUFFER
YES
NO
TPOST THE EMPTY BUFFER TO THE BUFFER RETURN QCB
IS EOM LEFT FROM LAST TIME
YES
NO
SWAP THE EOM BUFFER ADDRESS
ARE ALL EMPTY BUFFERS RETURNED
NO
YES
SAVE THE ADDRESS OF THIS EOM BUFFER
SET THE 'BUFFER SERVICED' PRIORITY IN THE BUFFER
EXIT TO DSPDISP
YES
IS THIS RETRIEVE MODE
NO
MHPOST
PREPARE TO TPOST THE BUFFER TO THE MH QCB
YES
CAN BUFFERS BE TPOSTED TO MH
NO
EW1 C3
TPOST THE BUFFER TO THE STARTMH QCB
IS THIS AN EOM BUFFER
YES
NO
TURN ON THE 'MH-OK' FLAG
EXIT TO DSPDISP
A
B
C
D
E
F
G
H
J
K

Chart EW-3 GET SCHEDULER

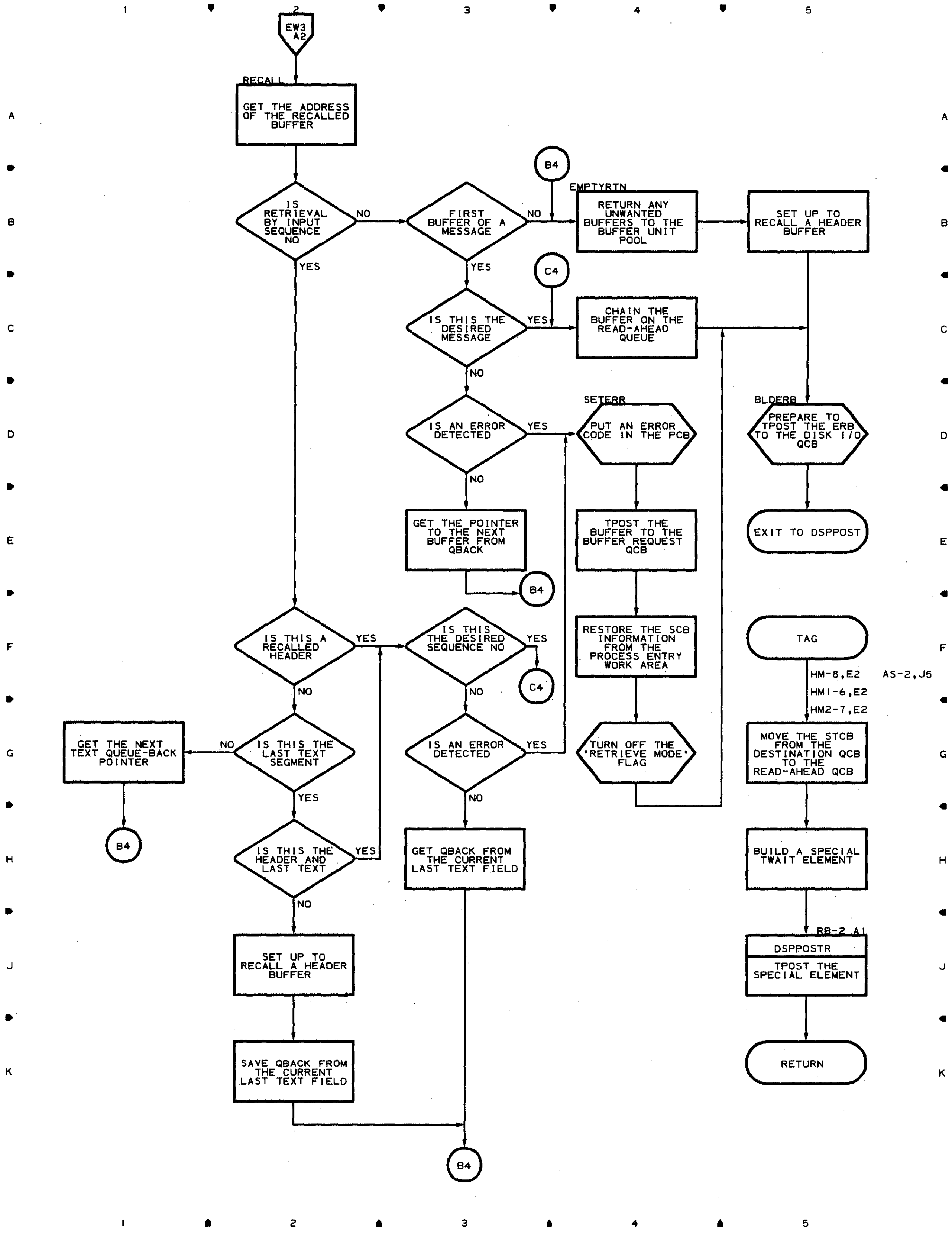

EW3 A2
RECALL
GET THE ADDRESS OF THE RECALLED BUFFER
IS RETRIEVAL BY INPUT SEQUENCE NO
NO
YES
FIRST BUFFER OF A MESSAGE
NO
YES
B4
EMPTYRTN
RETURN ANY UNWANTED BUFFERS TO THE BUFFER UNIT POOL
SET UP TO RECALL A HEADER BUFFER
C4
IS THIS THE DESIRED MESSAGE
YES
NO
CHAIN THE BUFFER ON THE READ-AHEAD QUEUE
SETERR
BLDERB
IS AN ERROR DETECTED
YES
NO
PUT AN ERROR CODE IN THE PCB
PREPARE TO TPOST THE ERB TO THE DISK I/O QCB
GET THE POINTER TO THE NEXT BUFFER FROM QBACK
B4
TPOST THE BUFFER TO THE BUFFER REQUEST QCB
EXIT TO DSPPOST
IS THIS A RECALLED HEADER
YES
NO
IS THIS THE DESIRED SEQUENCE NO
YES
NO
C4
RESTORE THE SCB INFORMATION FROM THE PROCESS ENTRY WORK AREA
TAG
HM-8,E2
HM1-6,E2
HM2-7,E2
AS-2,J5
GET THE NEXT TEXT QUEUE-BACK POINTER
NO
IS THIS THE LAST TEXT SEGMENT
YES
IS AN ERROR DETECTED
YES
NO
TURN OFF THE 'RETRIEVE MODE' FLAG
MOVE THE STCB FROM THE DESTINATION QCB TO THE READ-AHEAD QCB
B4
IS THIS THE HEADER AND LAST TEXT
YES
NO
GET QBACK FROM THE CURRENT LAST TEXT FIELD
BUILD A SPECIAL TWAIT ELEMENT
RB-2 A1
DSPPOSTR
TPOST THE SPECIAL ELEMENT
SET UP TO RECALL A HEADER BUFFER
SAVE QBACK FROM THE CURRENT LAST TEXT FIELD
RETURN
B4
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart EZ GET SCHEDULER FIFO ROUTINE

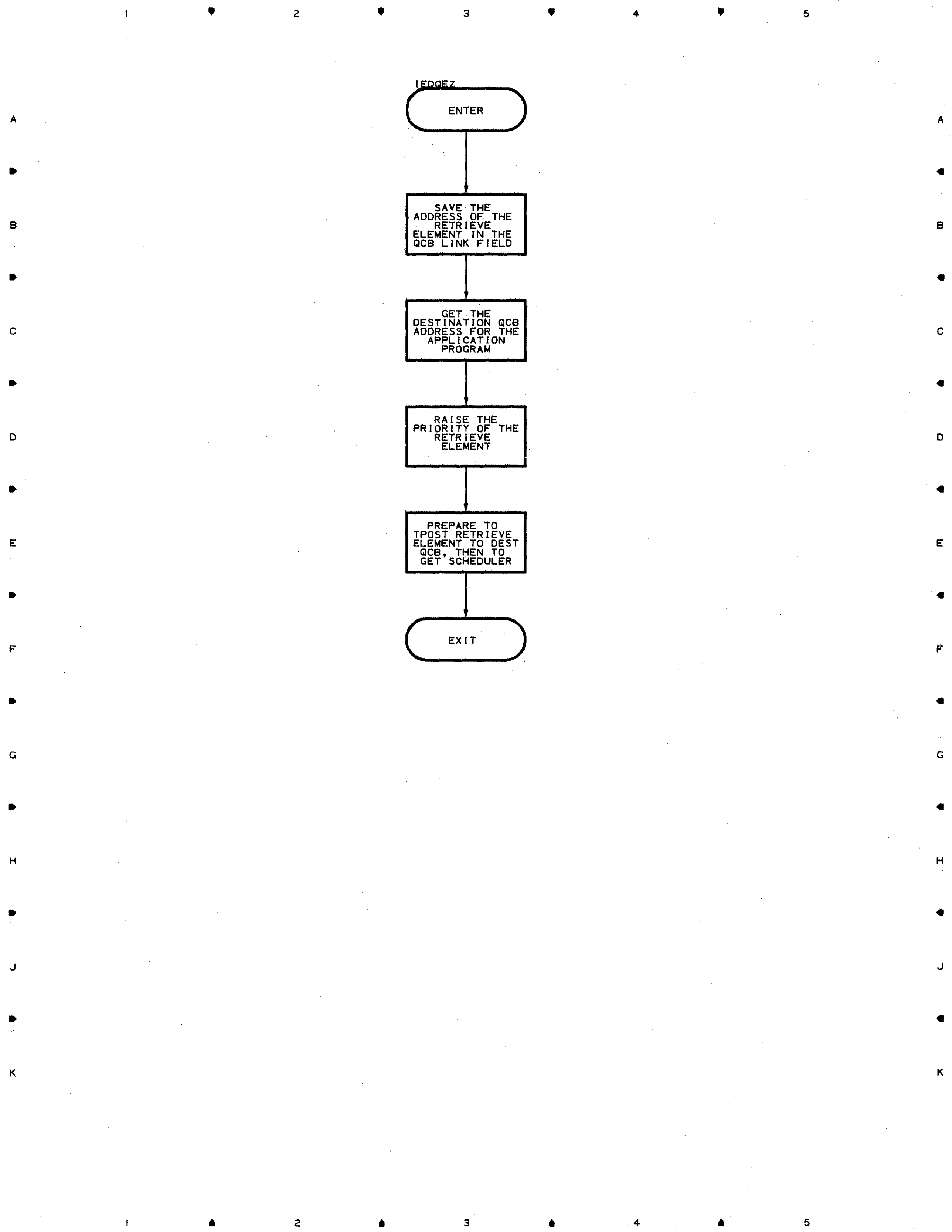

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQEZ
ENTER
SAVE THE ADDRESS OF THE RETRIEVE ELEMENT IN THE QCB LINK FIELD
GET THE DESTINATION QCB ADDRESS FOR THE APPLICATION PROGRAM
RAISE THE PRIORITY OF THE RETRIEVE ELEMENT
PREPARE TO TPOST RETRIEVE ELEMENT TO DEST QCB, THEN TO GET SCHEDULER
EXIT

Chart E1 TCOPY SERVICE ROUTINE

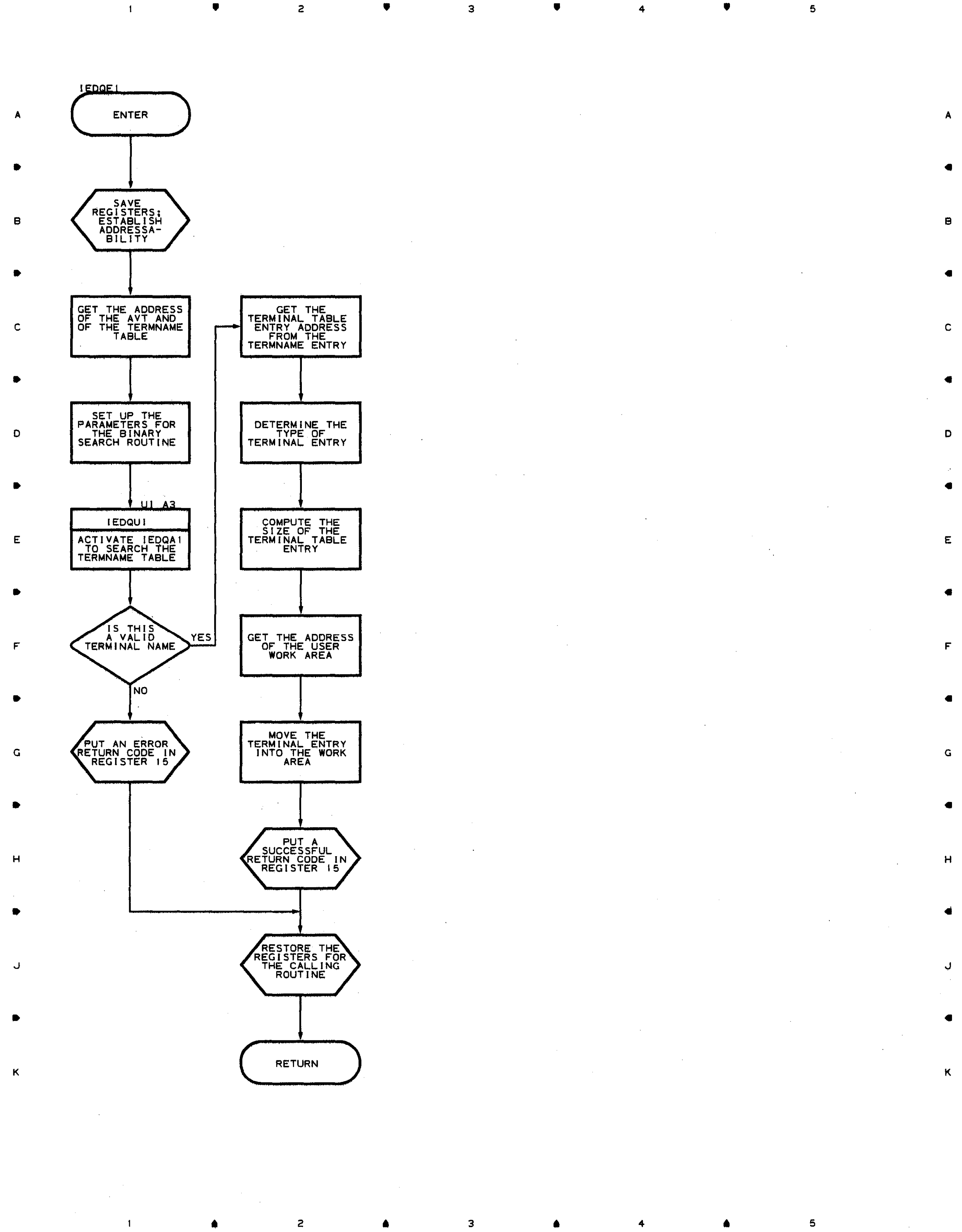

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQE1
ENTER
SAVE REGISTERS; ESTABLISH ADDRESSA-BILITY
GET THE ADDRESS OF THE AVT AND OF THE TERMNAME TABLE
SET UP THE PARAMETERS FOR THE BINARY SEARCH ROUTINE
U1 A3
IEDQU1
ACTIVATE IEDQA1 TO SEARCH THE TERMNAME TABLE
IS THIS A VALID TERMINAL NAME
YES
NO
PUT AN ERROR RETURN CODE IN REGISTER 15
GET THE TERMINAL TABLE ENTRY ADDRESS FROM THE TERMNAME ENTRY
DETERMINE THE TYPE OF TERMINAL ENTRY
COMPUTE THE SIZE OF THE TERMINAL TABLE ENTRY
GET THE ADDRESS OF THE USER WORK AREA
MOVE THE TERMINAL ENTRY INTO THE WORK AREA
PUT A SUCCESSFUL RETURN CODE IN REGISTER 15
RESTORE THE REGISTERS FOR THE CALLING ROUTINE
RETURN

Chart E2 QCOPY SERVICE ROUTINE

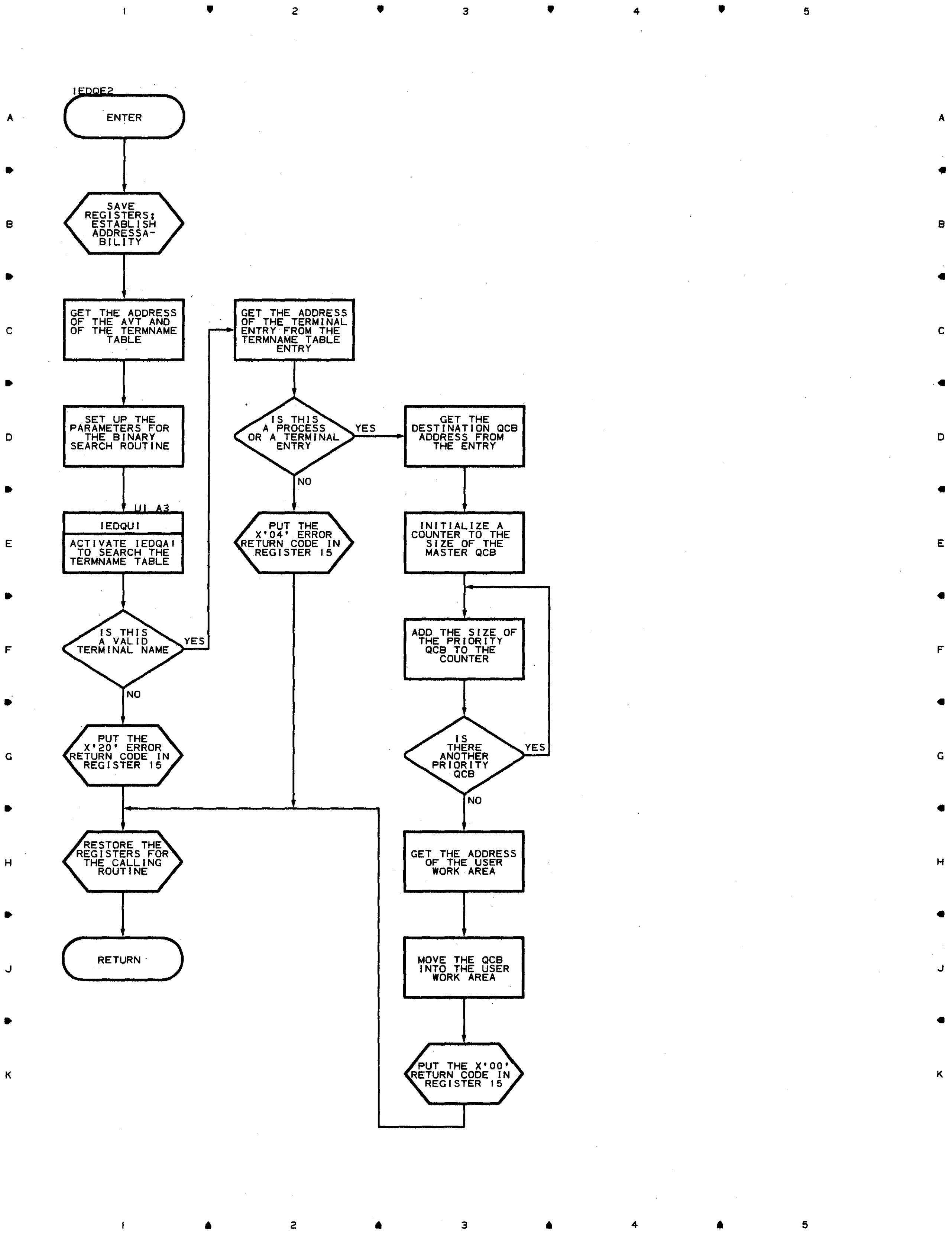

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQE2
ENTER
SAVE REGISTERS; ESTABLISH ADDRESSABILITY
GET THE ADDRESS OF THE AVT AND OF THE TERMNAME TABLE
SET UP THE PARAMETERS FOR THE BINARY SEARCH ROUTINE
UI A3
IEDQUI
ACTIVATE IEDQAI TO SEARCH THE TERMNAME TABLE
IS THIS A VALID TERMINAL NAME
YES
NO
PUT THE X'20' ERROR RETURN CODE IN REGISTER 15
RESTORE THE REGISTERS FOR THE CALLING ROUTINE
RETURN
GET THE ADDRESS OF THE TERMINAL ENTRY FROM THE TERMNAME TABLE ENTRY
IS THIS A PROCESS OR A TERMINAL ENTRY
YES
NO
PUT THE X'04' ERROR RETURN CODE IN REGISTER 15
GET THE DESTINATION QCB ADDRESS FROM THE ENTRY
INITIALIZE A COUNTER TO THE SIZE OF THE MASTER QCB
ADD THE SIZE OF THE PRIORITY QCB TO THE COUNTER
IS THERE ANOTHER PRIORITY QCB
YES
NO
GET THE ADDRESS OF THE USER WORK AREA
MOVE THE QCB INTO THE USER WORK AREA
PUT THE X'00' RETURN CODE IN REGISTER 15

Chart E3 TCHNG SERVICE ROUTINE

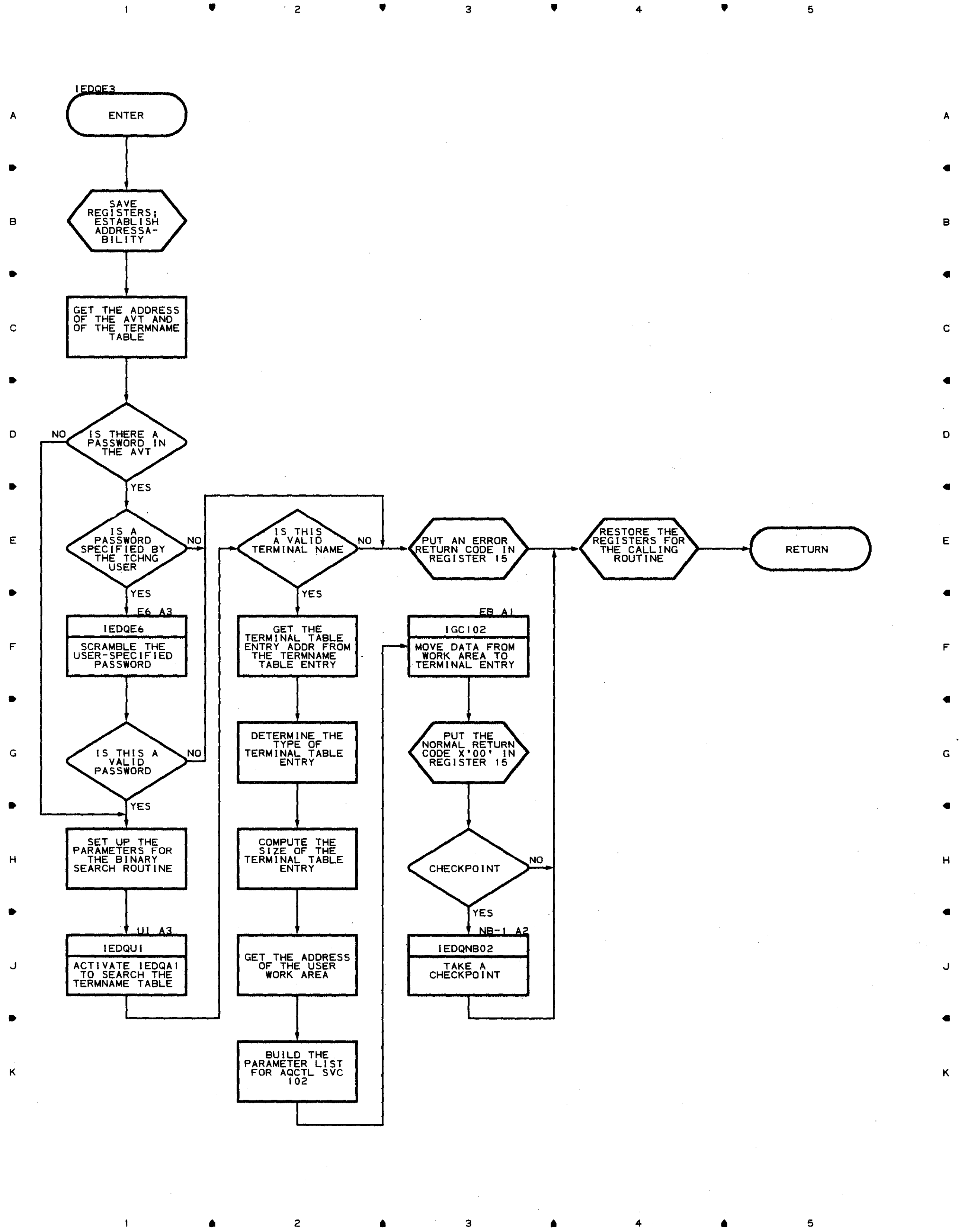

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQE3
ENTER
SAVE REGISTERS; ESTABLISH ADDRESSA-BILITY
GET THE ADDRESS OF THE AVT AND OF THE TERMNAME TABLE
IS THERE A PASSWORD IN THE AVT
NO
YES
IS A PASSWORD SPECIFIED BY THE TCHNG USER
NO
YES
E6 A3
IEDQE6
SCRAMBLE THE USER-SPECIFIED PASSWORD
IS THIS A VALID PASSWORD
NO
YES
SET UP THE PARAMETERS FOR THE BINARY SEARCH ROUTINE
UI A3
IEDQUI
ACTIVATE IEDQAI TO SEARCH THE TERMNAME TABLE
IS THIS A VALID TERMINAL NAME
NO
YES
GET THE TERMINAL TABLE ENTRY ADDR FROM THE TERMNAME TABLE ENTRY
DETERMINE THE TYPE OF TERMINAL TABLE ENTRY
COMPUTE THE SIZE OF THE TERMINAL TABLE ENTRY
GET THE ADDRESS OF THE USER WORK AREA
BUILD THE PARAMETER LIST FOR AQCTL SVC 102
PUT AN ERROR RETURN CODE IN REGISTER 15
E8 A1
IGC102
MOVE DATA FROM WORK AREA TO TERMINAL ENTRY
PUT THE NORMAL RETURN CODE X'00' IN REGISTER 15
CHECKPOINT
NO
YES
NB-1 A2
IEDQNB02
TAKE A CHECKPOINT
RESTORE THE REGISTERS FOR THE CALLING ROUTINE
RETURN

Chart E4 ICOPY SERVICE ROUTINE

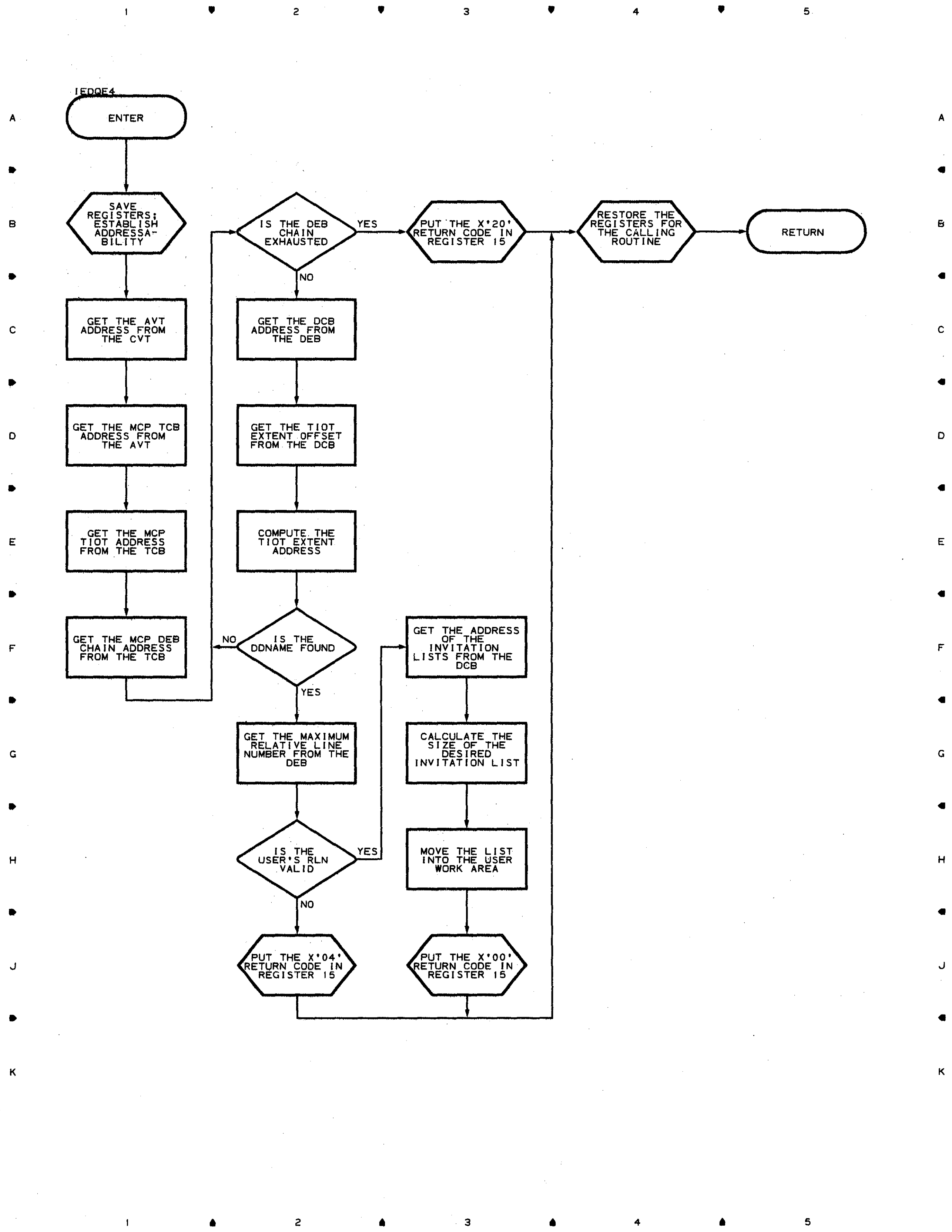
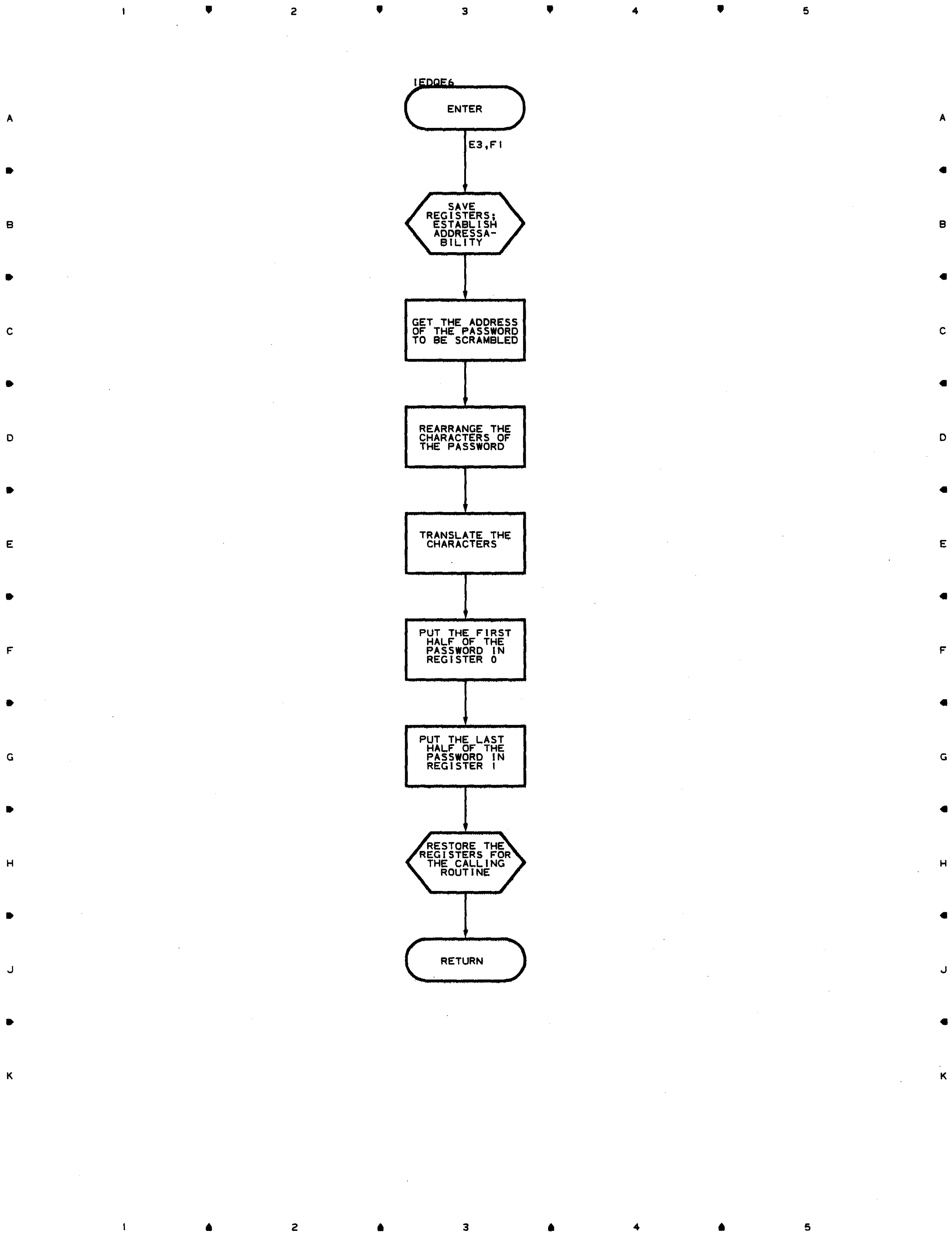

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQE4
ENTER
SAVE REGISTERS; ESTABLISH ADDRESSA-BILITY
GET THE AVT ADDRESS FROM THE CVT
GET THE MCP TCB ADDRESS FROM THE AVT
GET THE MCP TIOT ADDRESS FROM THE TCB
GET THE MCP DEB CHAIN ADDRESS FROM THE TCB
IS THE DEB CHAIN EXHAUSTED
YES
NO
GET THE DCB ADDRESS FROM THE DEB
GET THE TIOT EXTENT OFFSET FROM THE DCB
COMPUTE THE TIOT EXTENT ADDRESS
IS THE DDNAME FOUND
NO
YES
GET THE MAXIMUM RELATIVE LINE NUMBER FROM THE DEB
IS THE USER'S RLN VALID
YES
NO
PUT THE X'04' RETURN CODE IN REGISTER 15
PUT THE X'20' RETURN CODE IN REGISTER 15
GET THE ADDRESS OF THE INVITATION LISTS FROM THE DCB
CALCULATE THE SIZE OF THE DESIRED INVITATION LIST
MOVE THE LIST INTO THE USER WORK AREA
PUT THE X'00' RETURN CODE IN REGISTER 15
RESTORE THE REGISTERS FOR THE CALLING ROUTINE
RETURN

Chart E6 PASSWORD SCRAMBLER ROUTINE



IEDQE6
ENTER
E3,F1
SAVE REGISTERS; ESTABLISH ADDRESSABILITY
GET THE ADDRESS OF THE PASSWORD TO BE SCRAMBLED
REARRANGE THE CHARACTERS OF THE PASSWORD
TRANSLATE THE CHARACTERS
PUT THE FIRST HALF OF THE PASSWORD IN REGISTER 0
PUT THE LAST HALF OF THE PASSWORD IN REGISTER 1
RESTORE THE REGISTERS FOR THE CALLING ROUTINE
RETURN

Chart E7 RETRIEVE SCHEDULER

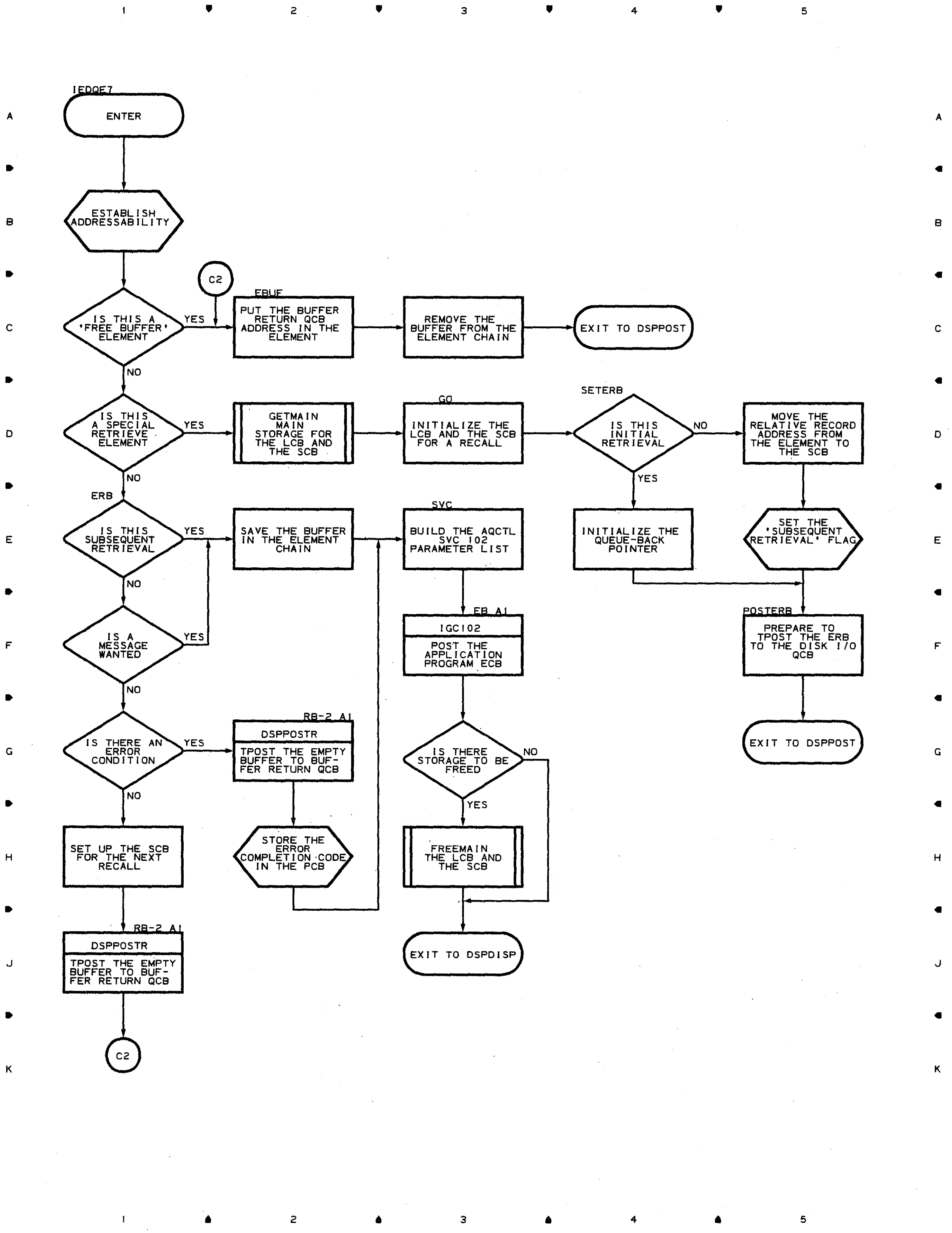

IEDQE7
ENTER
ESTABLISH ADDRESSABILITY
C2
EBUF
IS THIS A 'FREE BUFFER' ELEMENT
YES
NO
PUT THE BUFFER RETURN QCB ADDRESS IN THE ELEMENT
REMOVE THE BUFFER FROM THE ELEMENT CHAIN
EXIT TO DSPPOST
IS THIS A SPECIAL RETRIEVE ELEMENT
YES
NO
GETMAIN MAIN STORAGE FOR THE LCB AND THE SCB
GO
INITIALIZE THE LCB AND THE SCB FOR A RECALL
SETERB
IS THIS INITIAL RETRIEVAL
NO
YES
MOVE THE RELATIVE RECORD ADDRESS FROM THE ELEMENT TO THE SCB
ERB
IS THIS SUBSEQUENT RETRIEVAL
YES
NO
SAVE THE BUFFER IN THE ELEMENT CHAIN
SVC
BUILD THE AQCTL SVC 102 PARAMETER LIST
INITIALIZE THE QUEUE-BACK POINTER
SET THE 'SUBSEQUENT RETRIEVAL' FLAG
IS A MESSAGE WANTED
YES
NO
EB A1
IGC102
POST THE APPLICATION PROGRAM ECB
POSTERB
PREPARE TO TPOST THE ERB TO THE DISK I/O QCB
IS THERE AN ERROR CONDITION
YES
NO
RB-2 A1
DSPPOSTR
TPOST THE EMPTY BUFFER TO BUF-FER RETURN QCB
IS THERE STORAGE TO BE FREED
NO
YES
EXIT TO DSPPOST
SET UP THE SCB FOR THE NEXT RECALL
STORE THE ERROR COMPLETION CODE IN THE PCB
FREEMAIN THE LCB AND THE SCB
RB-2 A1
DSPPOSTR
TPOST THE EMPTY BUFFER TO BUF-FER RETURN QCB
EXIT TO DSPDISP
C2
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart FA-1 CPB INITIALIZATION

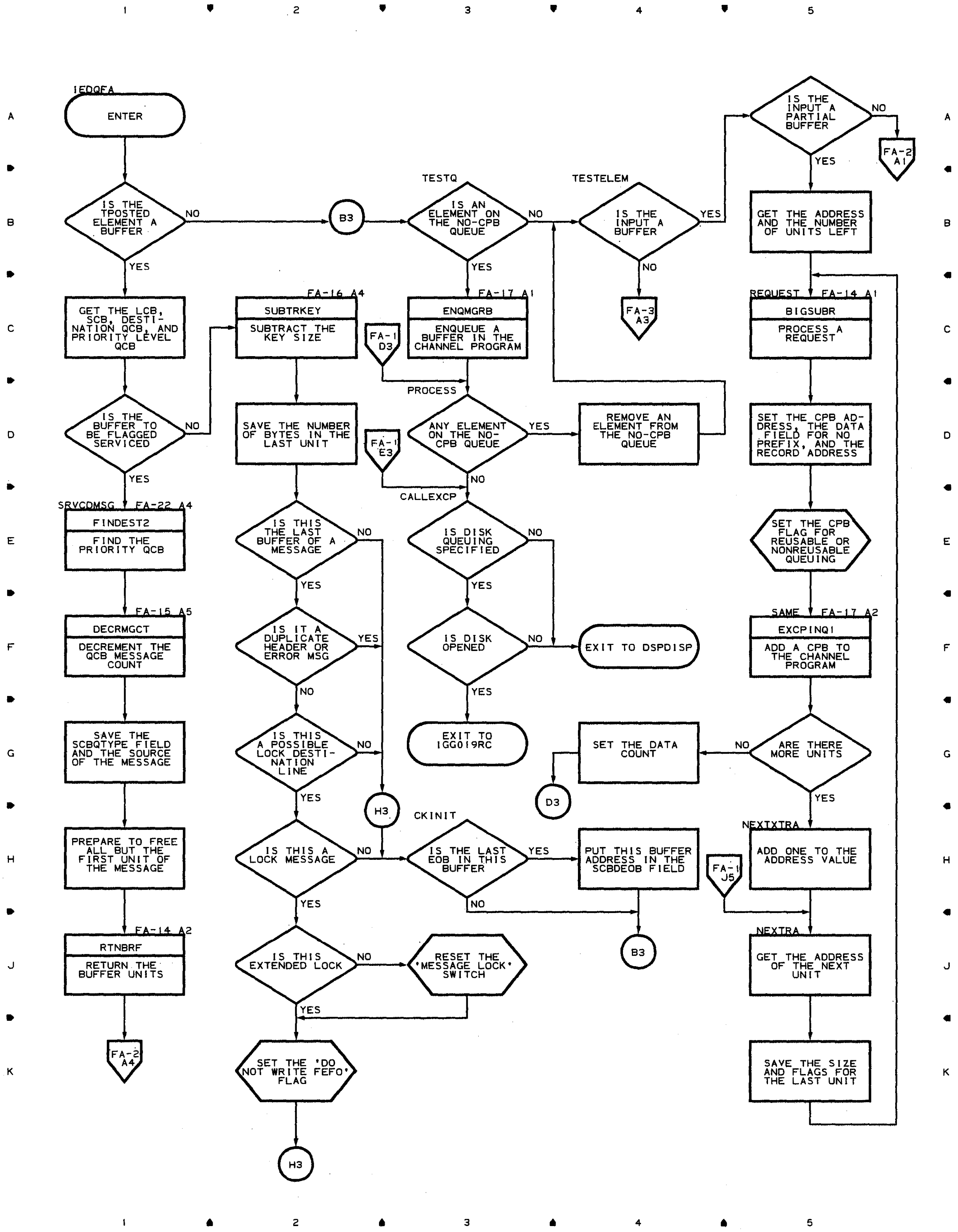

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQFA
ENTER
IS THE TPOSTED ELEMENT A BUFFER
NO
YES
B3
TESTQ
IS AN ELEMENT ON THE NO-CPB QUEUE
NO
YES
TESTELEM
IS THE INPUT A BUFFER
YES
NO
IS THE INPUT A PARTIAL BUFFER
NO
YES
FA-2 A1
GET THE ADDRESS AND THE NUMBER OF UNITS LEFT
GET THE LCB, SCB, DESTI-NATION QCB, AND PRIORITY LEVEL QCB
FA-16 A4
SUBTRKEY
SUBTRACT THE KEY SIZE
FA-1 D3
FA-17 A1
ENQMGRB
ENQUEUE A BUFFER IN THE CHANNEL PROGRAM
FA-3 A3
REQUEST FA-14 A1
BIGSUBR
PROCESS A REQUEST
IS THE BUFFER TO BE FLAGGED SERVICED
NO
YES
SAVE THE NUMBER OF BYTES IN THE LAST UNIT
FA-1 E3
PROCESS
ANY ELEMENT ON THE NO-CPB QUEUE
YES
NO
REMOVE AN ELEMENT FROM THE NO-CPB QUEUE
SET THE CPB AD-DRESS, THE DATA FIELD FOR NO PREFIX, AND THE RECORD ADDRESS
SRVCDMSG FA-22 A4
FINDEST2
FIND THE PRIORITY QCB
IS THIS THE LAST BUFFER OF A MESSAGE
NO
YES
CALLEXCP
IS DISK QUEUING SPECIFIED
NO
YES
SET THE CPB FLAG FOR REUSABLE OR NONREUSABLE QUEUING
FA-15 A5
DECRMGCT
DECREMENT THE QCB MESSAGE COUNT
IS IT A DUPLICATE HEADER OR ERROR MSG
YES
NO
IS DISK OPENED
NO
YES
EXIT TO DSPDISP
SAME FA-17 A2
EXCPINQ1
ADD A CPB TO THE CHANNEL PROGRAM
SAVE THE SCBQTYPE FIELD AND THE SOURCE OF THE MESSAGE
IS THIS A POSSIBLE LOCK DESTI-NATION LINE
NO
YES
EXIT TO IGG019RC
SET THE DATA COUNT
NO
ARE THERE MORE UNITS
YES
D3
H3
CKINIT
NEXTXTRA
PREPARE TO FREE ALL BUT THE FIRST UNIT OF THE MESSAGE
IS THIS A LOCK MESSAGE
NO
YES
IS THE LAST EOB IN THIS BUFFER
YES
NO
PUT THIS BUFFER ADDRESS IN THE SCBDEOB FIELD
FA-1 J5
ADD ONE TO THE ADDRESS VALUE
FA-14 A2
RTNBRF
RETURN THE BUFFER UNITS
IS THIS EXTENDED LOCK
NO
YES
RESET THE 'MESSAGE LOCK' SWITCH
B3
NEXTRA
GET THE ADDRESS OF THE NEXT UNIT
FA-2 A4
SET THE 'DO NOT WRITE FEFO' FLAG
SAVE THE SIZE AND FLAGS FOR THE LAST UNIT
H3

Chart FA-2 CPB INITIALIZATION

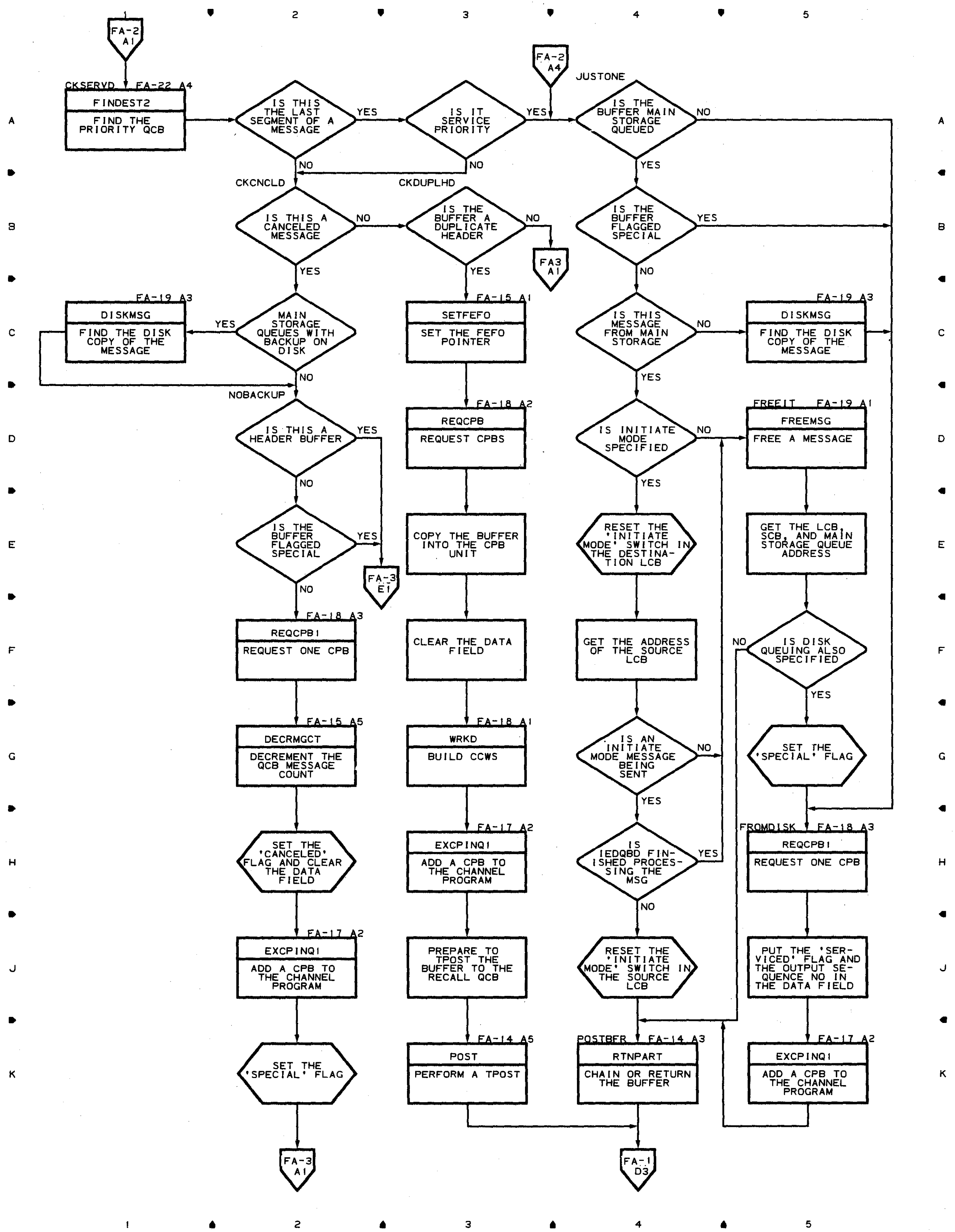

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
FA-2 A1
CKSERVD FA-22 A4
FINDEST2
FIND THE PRIORITY QCB
IS THIS THE LAST SEGMENT OF A MESSAGE
YES
NO
IS IT SERVICE PRIORITY
YES
NO
FA-2 A4
JUSTONE
IS THE BUFFER MAIN STORAGE QUEUED
NO
YES
CKCNCLD
CKDUPLHD
IS THIS A CANCELED MESSAGE
NO
YES
IS THE BUFFER A DUPLICATE HEADER
NO
YES
FA3 A1
IS THE BUFFER FLAGGED SPECIAL
YES
NO
FA-19 A3
DISKMSG
FIND THE DISK COPY OF THE MESSAGE
YES
MAIN STORAGE QUEUES WITH BACKUP ON DISK
NO
FA-15 A1
SETFEFO
SET THE FEFO POINTER
IS THIS MESSAGE FROM MAIN STORAGE
NO
YES
FA-19 A3
DISKMSG
FIND THE DISK COPY OF THE MESSAGE
NOBACKUP
IS THIS A HEADER BUFFER
YES
NO
FA-18 A2
REQCPB
REQUEST CPBS
IS INITIATE MODE SPECIFIED
NO
YES
FREEIT FA-19 A1
FREEMSG
FREE A MESSAGE
IS THE BUFFER FLAGGED SPECIAL
YES
NO
FA-3 E1
COPY THE BUFFER INTO THE CPB UNIT
RESET THE 'INITIATE MODE' SWITCH IN THE DESTINATION LCB
GET THE LCB, SCB, AND MAIN STORAGE QUEUE ADDRESS
FA-18 A3
REQCPB1
REQUEST ONE CPB
CLEAR THE DATA FIELD
GET THE ADDRESS OF THE SOURCE LCB
NO
IS DISK QUEUING ALSO SPECIFIED
YES
FA-15 A5
DECRMGCT
DECREMENT THE QCB MESSAGE COUNT
FA-18 A1
WRKD
BUILD CCWS
IS AN INITIATE MODE MESSAGE BEING SENT
NO
YES
SET THE 'SPECIAL' FLAG
FROMDISK FA-18 A3
SET THE 'CANCELED' FLAG AND CLEAR THE DATA FIELD
FA-17 A2
EXCPINQ1
ADD A CPB TO THE CHANNEL PROGRAM
IS IEDQBD FINISHED PROCESSING THE MSG
YES
NO
REQCPB1
REQUEST ONE CPB
FA-17 A2
EXCPINQ1
ADD A CPB TO THE CHANNEL PROGRAM
PREPARE TO TPOST THE BUFFER TO THE RECALL QCB
RESET THE 'INITIATE MODE' SWITCH IN THE SOURCE LCB
PUT THE 'SERVICED' FLAG AND THE OUTPUT SEQUENCE NO IN THE DATA FIELD
SET THE 'SPECIAL' FLAG
FA-14 A5
POST
PERFORM A TPOST
POSTBFR FA-14 A3
RTNPART
CHAIN OR RETURN THE BUFFER
FA-17 A2
EXCPINQ1
ADD A CPB TO THE CHANNEL PROGRAM
FA-3 A1
FA-1 D3

Chart FA-3 CPB INITIALIZATION

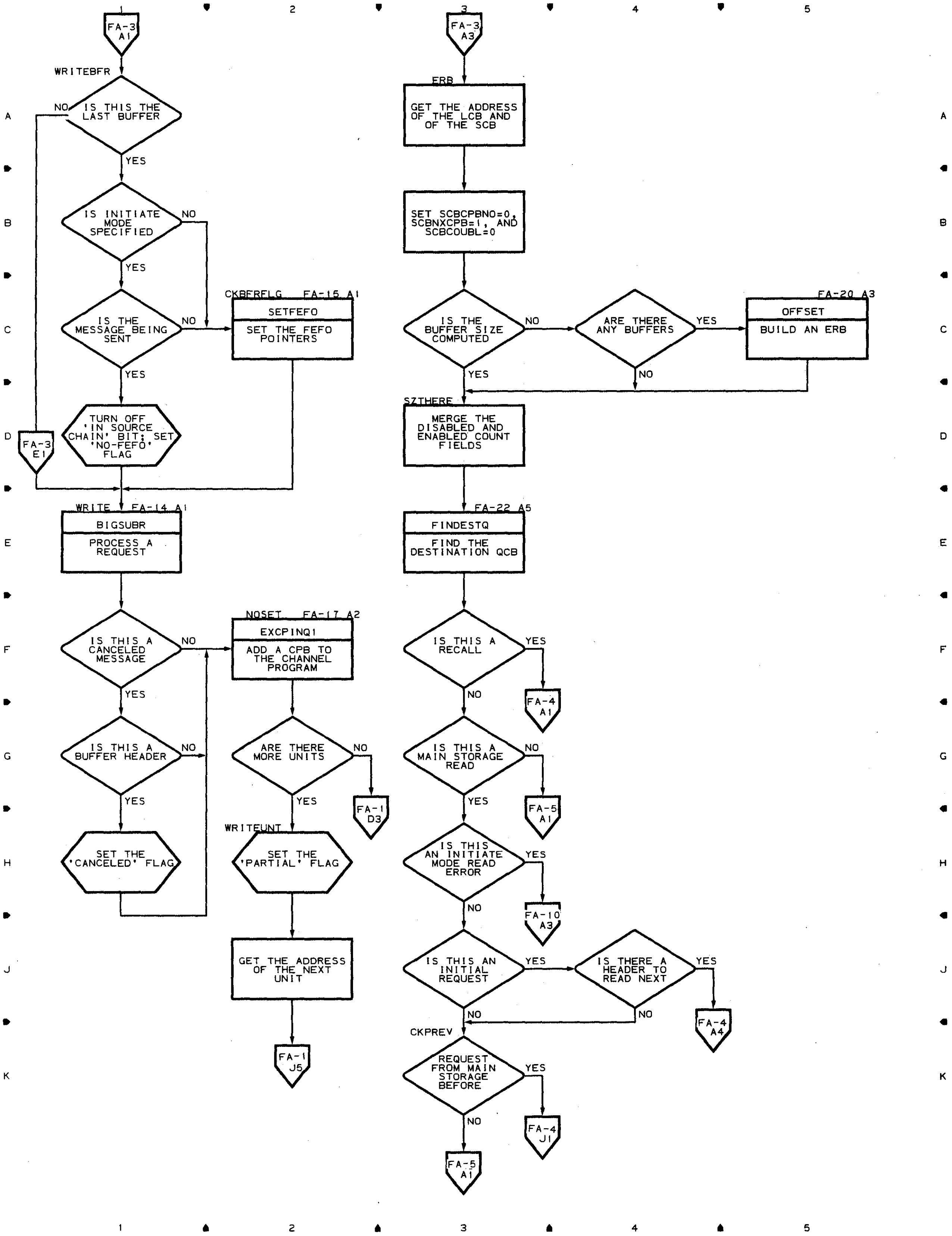


Chart FA-4 CPB INITIALIZATION

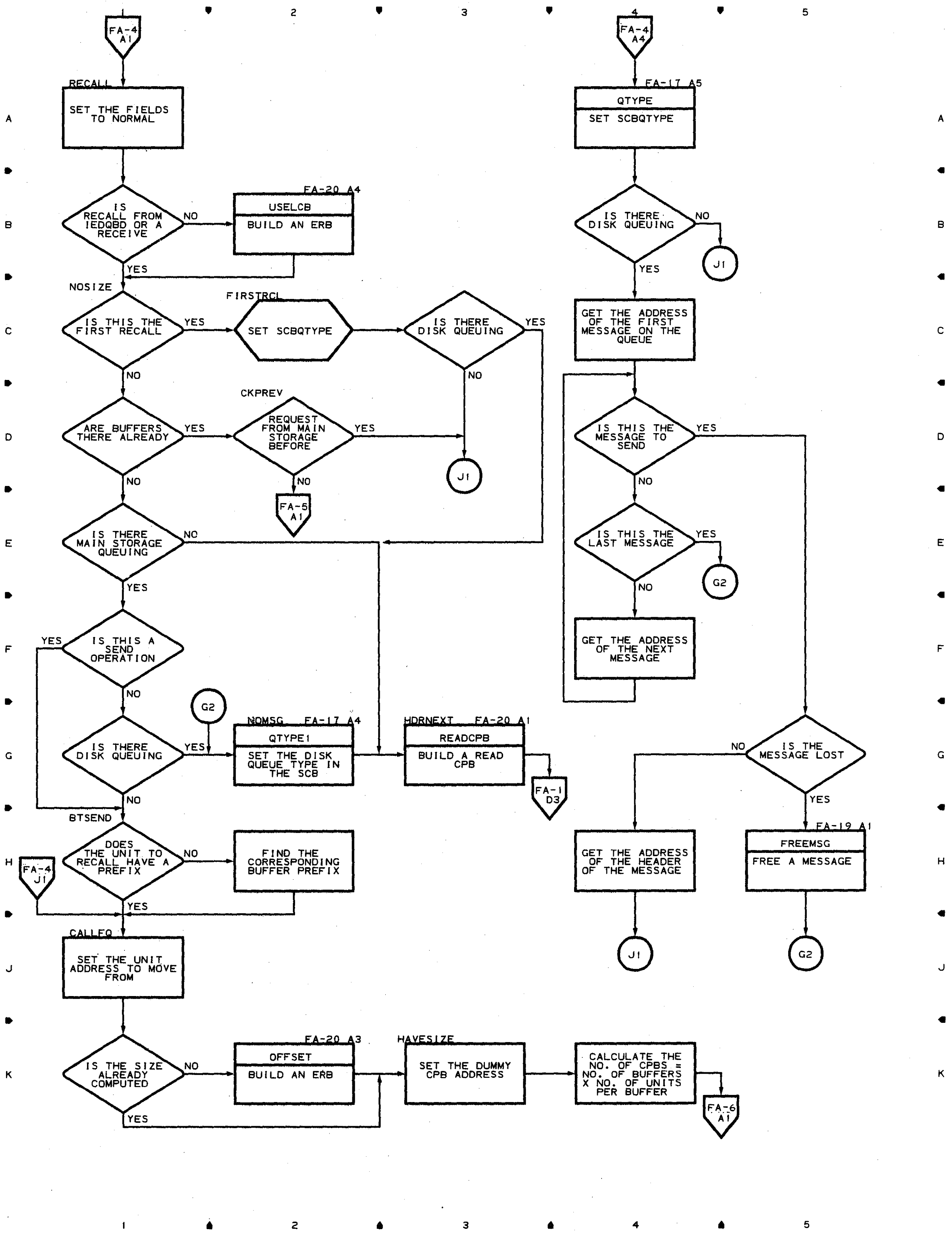


Chart FA-5 CPB INITIALIZATION

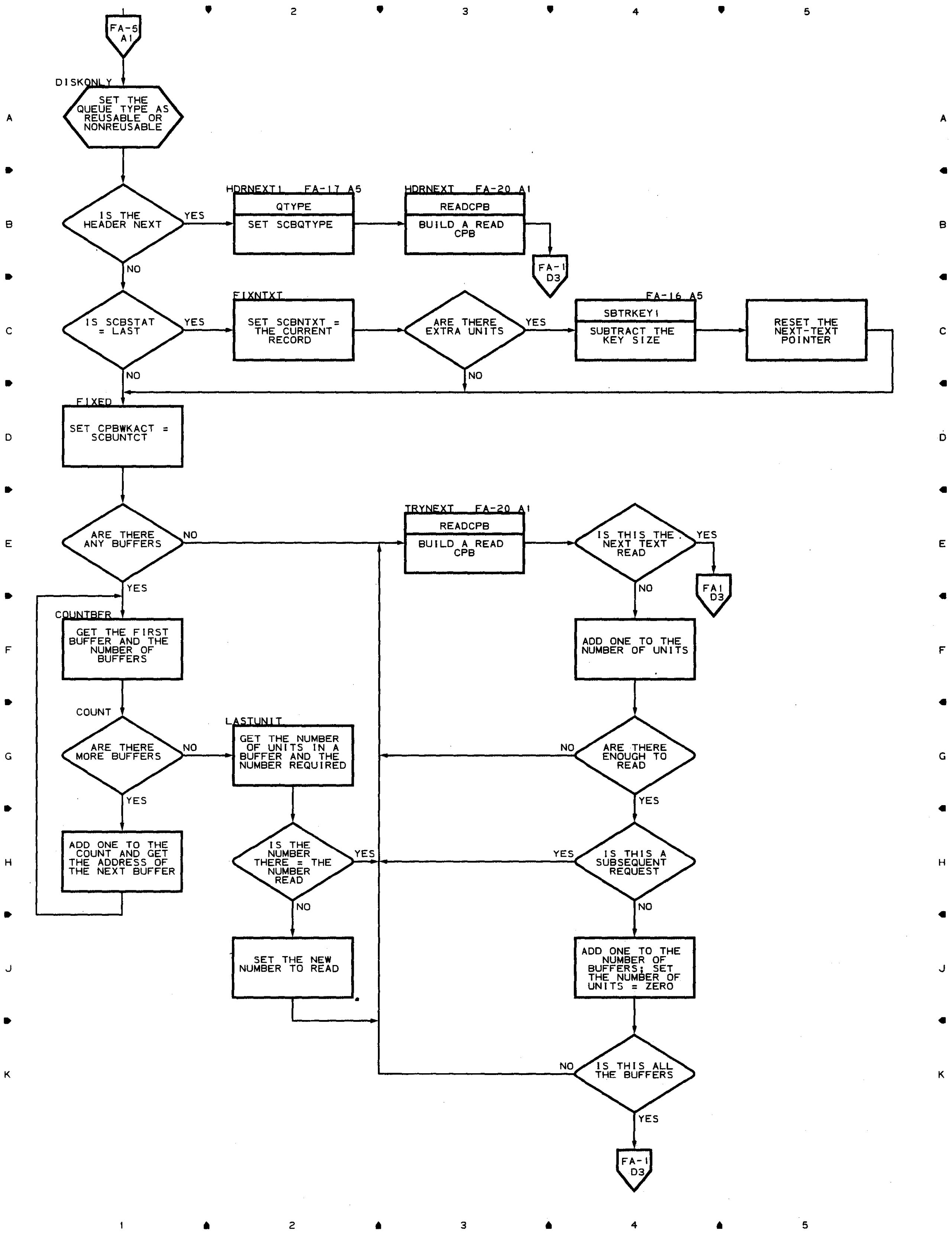

FA-5 A1
DISKONLY
SET THE QUEUE TYPE AS REUSABLE OR NONREUSABLE
IS THE HEADER NEXT
YES
NO
HDRNEXT1 FA-17 A5
QTYPE
SET SCBQTYPE
HDRNEXT FA-20 A1
READCPB
BUILD A READ CPB
FA-1 D3
IS SCBSTAT = LAST
YES
NO
FIXNTXT
SET SCBNTXT = THE CURRENT RECORD
ARE THERE EXTRA UNITS
YES
NO
FA-16 A5
SBTRKEY1
SUBTRACT THE KEY SIZE
RESET THE NEXT-TEXT POINTER
FIXED
SET CPBWKACT = SCBUNTCT
ARE THERE ANY BUFFERS
NO
YES
TRYNEXT FA-20 A1
READCPB
BUILD A READ CPB
IS THIS THE NEXT TEXT READ
YES
NO
FA1 D3
COUNTBFR
GET THE FIRST BUFFER AND THE NUMBER OF BUFFERS
ADD ONE TO THE NUMBER OF UNITS
COUNT
ARE THERE MORE BUFFERS
NO
YES
LASTUNIT
GET THE NUMBER OF UNITS IN A BUFFER AND THE NUMBER REQUIRED
NO
ARE THERE ENOUGH TO READ
YES
ADD ONE TO THE COUNT AND GET THE ADDRESS OF THE NEXT BUFFER
IS THE NUMBER THERE = THE NUMBER READ
YES
NO
YES
IS THIS A SUBSEQUENT REQUEST
NO
SET THE NEW NUMBER TO READ
ADD ONE TO THE NUMBER OF BUFFERS; SET THE NUMBER OF UNITS = ZERO
NO
IS THIS ALL THE BUFFERS
YES
FA-1 D3
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart FA-6 CPB INITIALIZATION

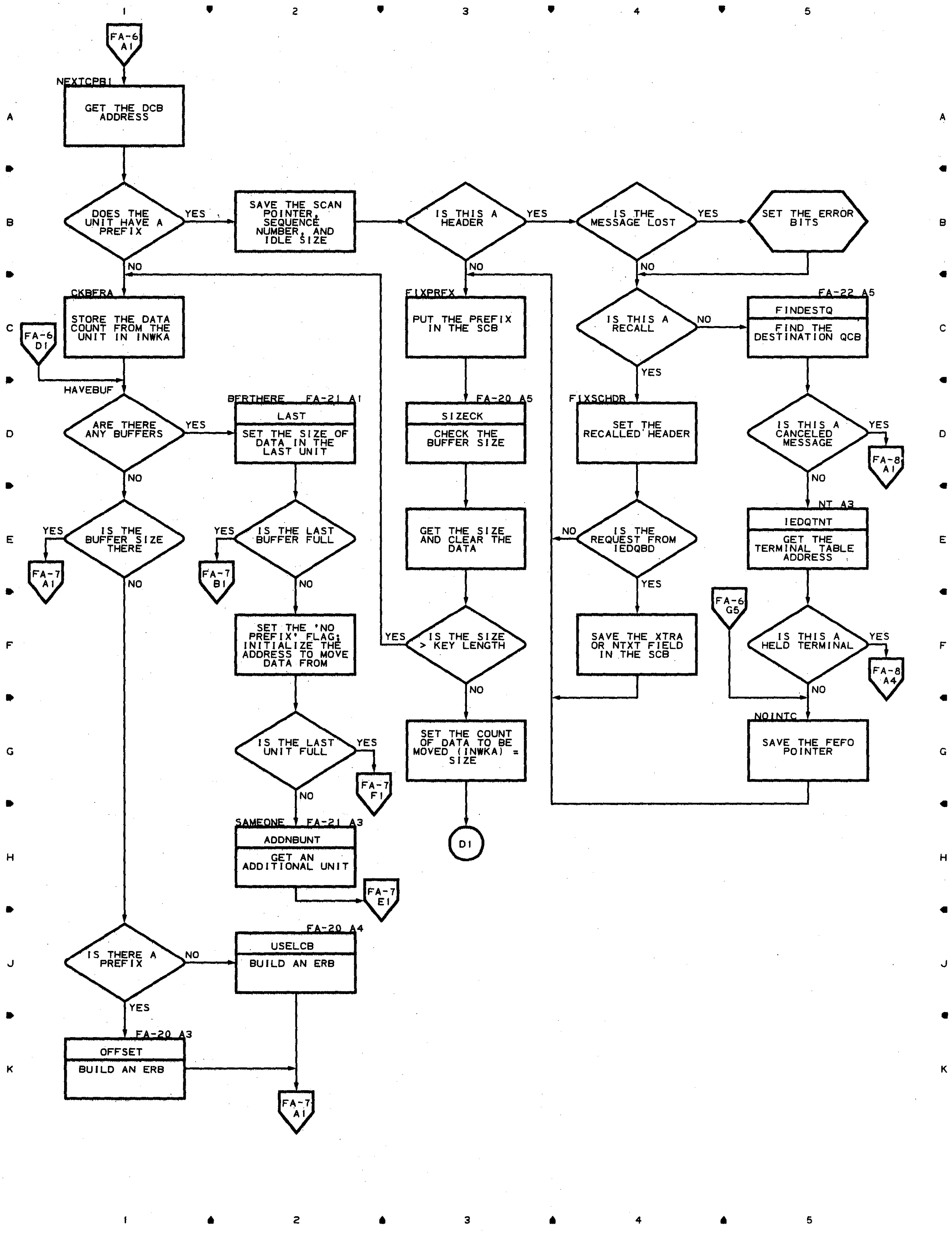


Chart FA-7 CPB INITIALIZATION

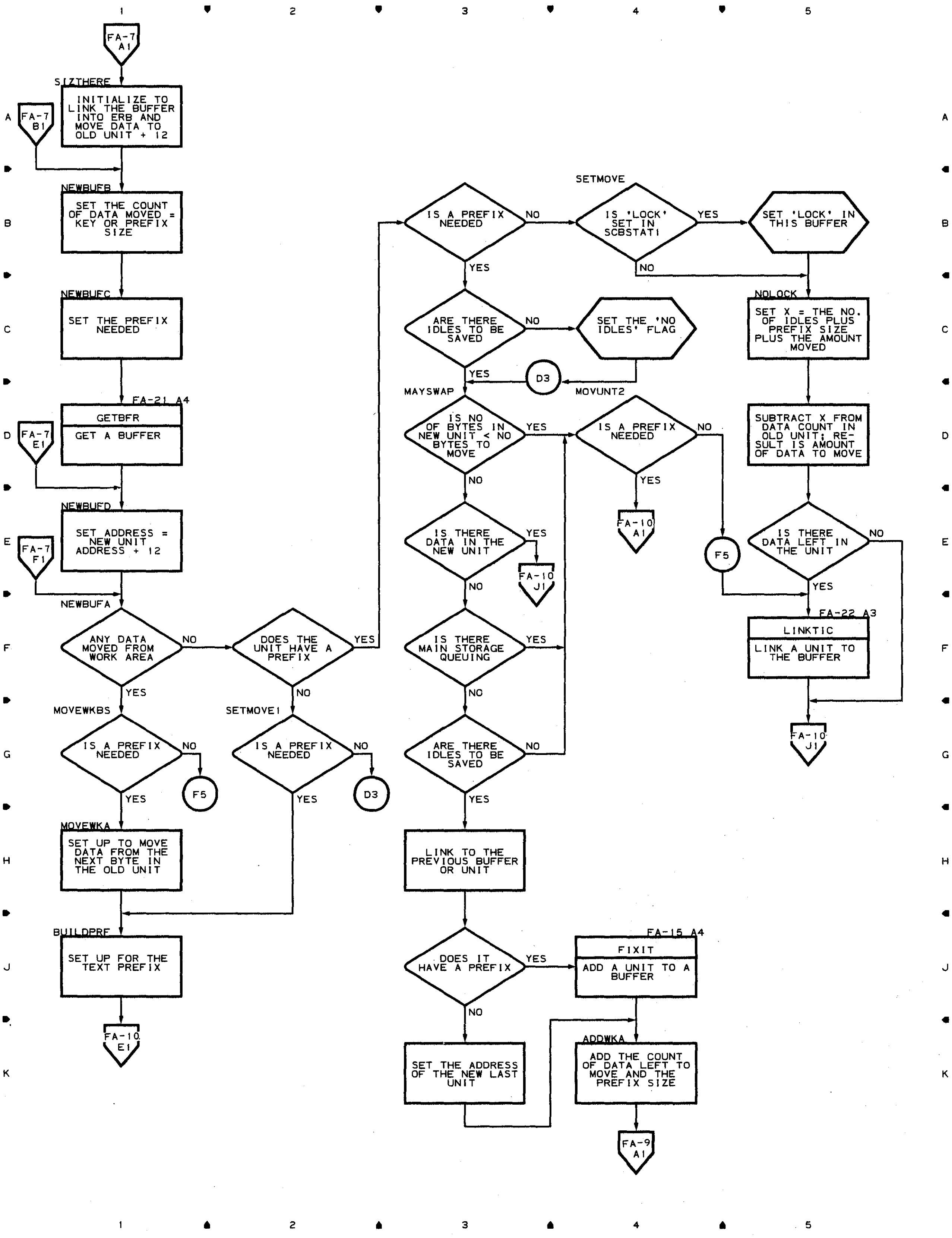

FA-7 A1
SIZTHERE
INITIALIZE TO LINK THE BUFFER INTO ERB AND MOVE DATA TO OLD UNIT + 12
FA-7 B1
NEWBUFB
SET THE COUNT OF DATA MOVED = KEY OR PREFIX SIZE
NEWBUFC
SET THE PREFIX NEEDED
FA-21 A4
GETBFR
GET A BUFFER
FA-7 E1
NEWBUFD
SET ADDRESS = NEW UNIT ADDRESS + 12
FA-7 F1
NEWBUFA
ANY DATA MOVED FROM WORK AREA
NO
YES
MOVEWKBS
IS A PREFIX NEEDED
NO
F5
YES
MOVEWKA
SET UP TO MOVE DATA FROM THE NEXT BYTE IN THE OLD UNIT
BUILDPRF
SET UP FOR THE TEXT PREFIX
FA-10 E1
DOES THE UNIT HAVE A PREFIX
YES
NO
SETMOVE1
IS A PREFIX NEEDED
NO
D3
YES
IS A PREFIX NEEDED
NO
YES
ARE THERE IDLES TO BE SAVED
NO
SET THE 'NO IDLES' FLAG
YES
D3
MAYSWAP
IS NO OF BYTES IN NEW UNIT < NO BYTES TO MOVE
YES
NO
IS THERE DATA IN THE NEW UNIT
YES
FA-10 J1
NO
IS THERE MAIN STORAGE QUEUING
YES
NO
ARE THERE IDLES TO BE SAVED
NO
YES
LINK TO THE PREVIOUS BUFFER OR UNIT
DOES IT HAVE A PREFIX
YES
FA-15 A4
FIXIT
ADD A UNIT TO A BUFFER
NO
SET THE ADDRESS OF THE NEW LAST UNIT
ADDWKA
ADD THE COUNT OF DATA LEFT TO MOVE AND THE PREFIX SIZE
FA-9 A1
SETMOVE
IS 'LOCK' SET IN SCBSTAT1
YES
SET 'LOCK' IN THIS BUFFER
NO
MOVUNT2
IS A PREFIX NEEDED
NO
YES
FA-10 A1
F5
NOLOCK
SET X = THE NO. OF IDLES PLUS PREFIX SIZE PLUS THE AMOUNT MOVED
SUBTRACT X FROM DATA COUNT IN OLD UNIT; RESULT IS AMOUNT OF DATA TO MOVE
IS THERE DATA LEFT IN THE UNIT
NO
YES
FA-22 A3
LINKTIC
LINK A UNIT TO THE BUFFER
FA-10 J1
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart FA-8 CPB INITIALIZATION

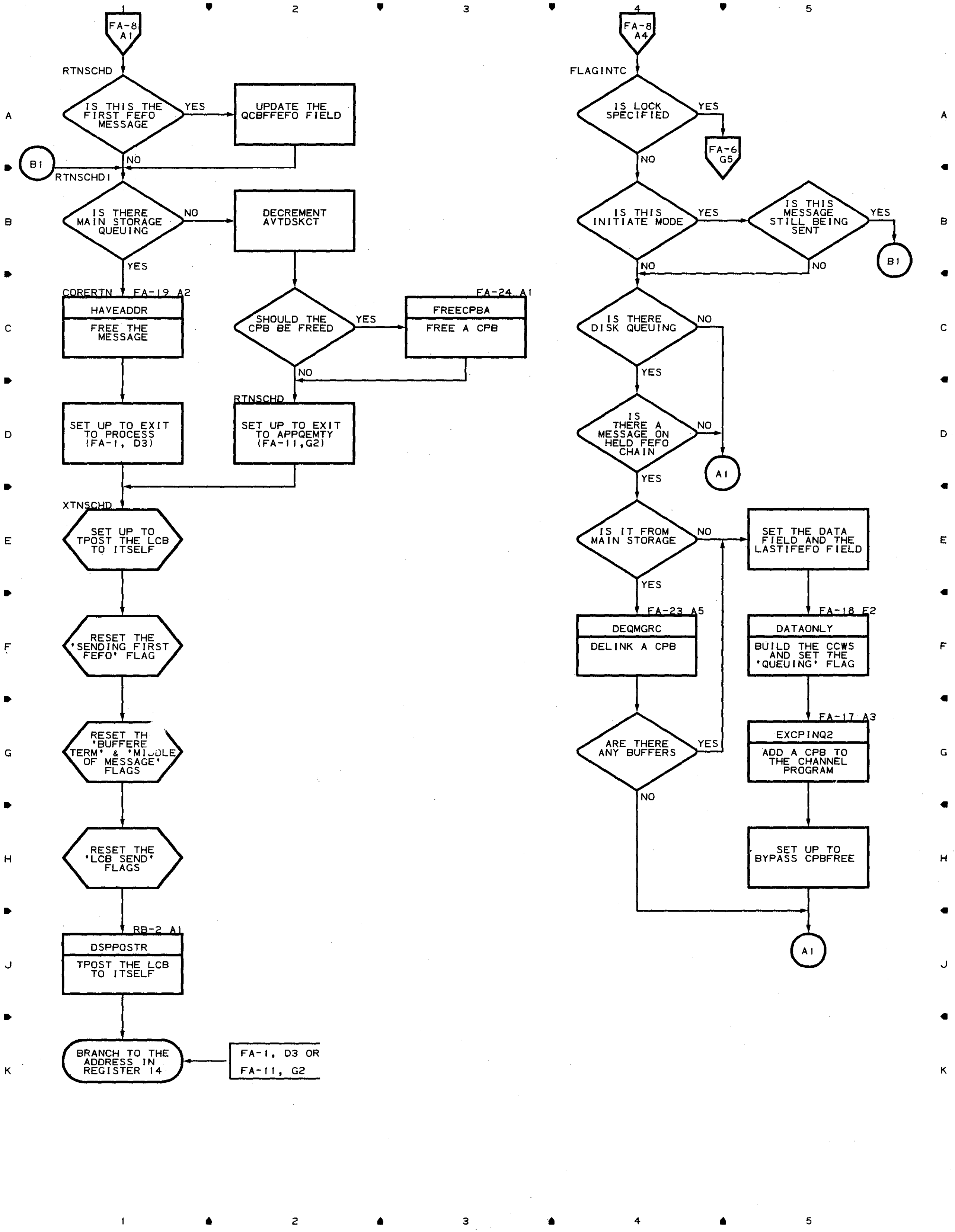

1
2
3
4
5
FA-8 A1
RTNSCHD
IS THIS THE FIRST FEFO MESSAGE
YES
UPDATE THE QCBFFEFO FIELD
NO
B1
RTNSCHD1
IS THERE MAIN STORAGE QUEUING
NO
DECREMENT AVTDSKCT
YES
CORERTN FA-19 A2
HAVEADDR
FREE THE MESSAGE
SHOULD THE CPB BE FREED
YES
FA-24 A1
FREECPBA
FREE A CPB
NO
RTNSCHD
SET UP TO EXIT TO PROCESS (FA-1, D3)
SET UP TO EXIT TO APPQEMTY (FA-11,G2)
XTNSCHD
SET UP TO TPOST THE LCB TO ITSELF
RESET THE 'SENDING FIRST FEFO' FLAG
RESET TH 'BUFFERE TERM' & 'MIDDLE OF MESSAGE' FLAGS
RESET THE 'LCB SEND' FLAGS
RB-2 A1
DSPPOSTR
TPOST THE LCB TO ITSELF
BRANCH TO THE ADDRESS IN REGISTER 14
FA-1, D3 OR
FA-11, G2
FA-8 A4
FLAGINTC
IS LOCK SPECIFIED
YES
FA-6 G5
NO
IS THIS INITIATE MODE
YES
IS THIS MESSAGE STILL BEING SENT
YES
B1
NO
NO
IS THERE DISK QUEUING
NO
YES
IS THERE A MESSAGE ON HELD FEFO CHAIN
NO
A1
YES
IS IT FROM MAIN STORAGE
NO
SET THE DATA FIELD AND THE LASTIFEFO FIELD
YES
FA-23 A5
DEQMGRC
DELINK A CPB
FA-18 E2
DATAONLY
BUILD THE CCWS AND SET THE 'QUEUING' FLAG
ARE THERE ANY BUFFERS
YES
FA-17 A3
EXCPINQ2
ADD A CPB TO THE CHANNEL PROGRAM
NO
SET UP TO BYPASS CPBFREE
A1
A
B
C
D
E
F
G
H
J
K
1
2
3
4
5

Chart FA-9 CPB INITIALIZATION

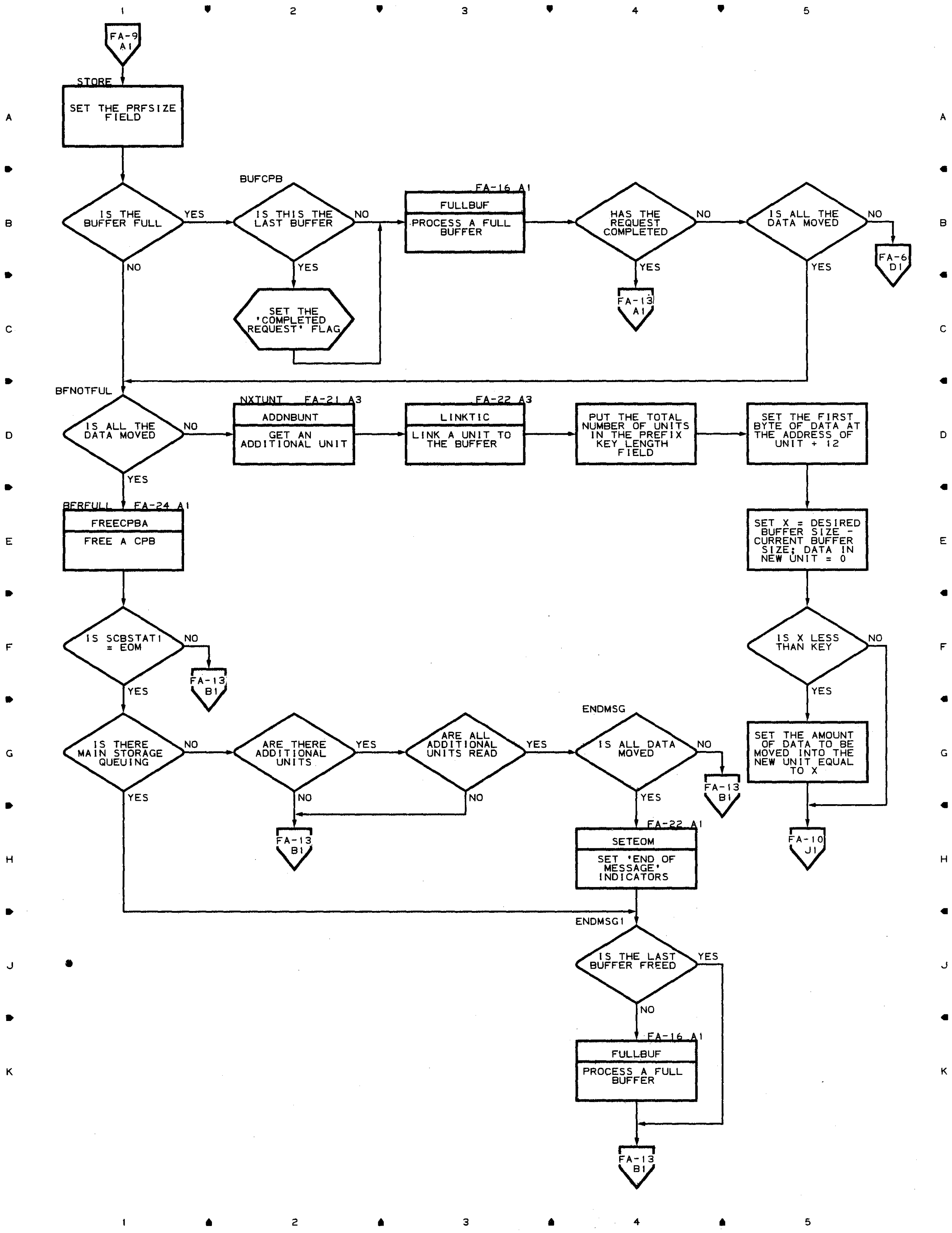


Chart FA-10 CPB INITIALIZATION

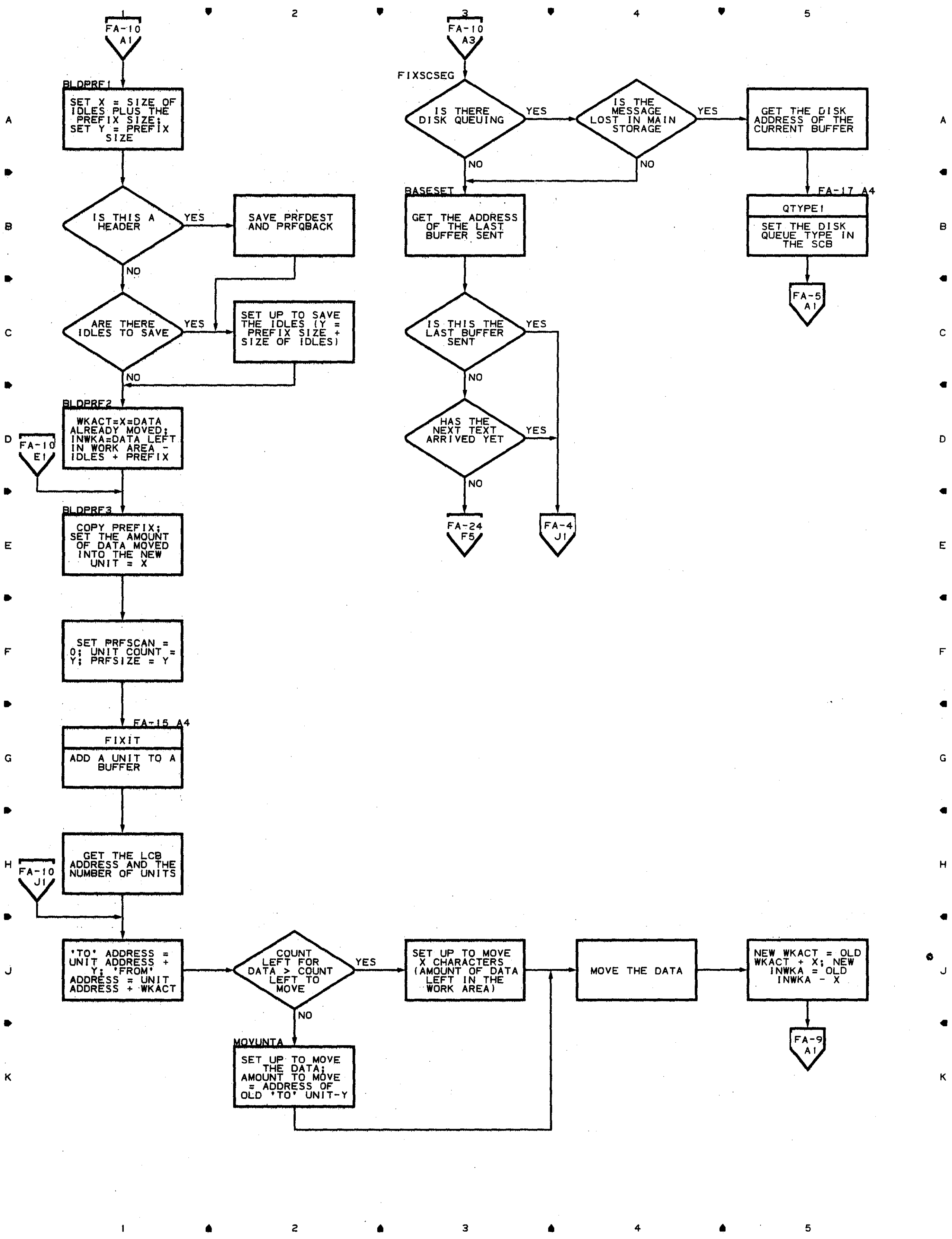

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
FA-10 A1
BLDPRF1
SET X = SIZE OF IDLES PLUS THE PREFIX SIZE; SET Y = PREFIX SIZE
IS THIS A HEADER
YES
SAVE PRFDEST AND PRFQBACK
NO
ARE THERE IDLES TO SAVE
YES
SET UP TO SAVE THE IDLES (Y = PREFIX SIZE + SIZE OF IDLES)
NO
BLDPRF2
WKACT=X=DATA ALREADY MOVED; INWKA=DATA LEFT IN WORK AREA - IDLES + PREFIX
FA-10 E1
BLDPRF3
COPY PREFIX; SET THE AMOUNT OF DATA MOVED INTO THE NEW UNIT = X
SET PRFSCAN = 0; UNIT COUNT = Y; PRFSIZE = Y
FA-15 A4
FIXIT
ADD A UNIT TO A BUFFER
GET THE LCB ADDRESS AND THE NUMBER OF UNITS
FA-10 J1
'TO' ADDRESS = UNIT ADDRESS + Y; 'FROM' ADDRESS = UNIT ADDRESS + WKACT
COUNT LEFT FOR DATA > COUNT LEFT TO MOVE
YES
NO
MOVUNTA
SET UP TO MOVE THE DATA; AMOUNT TO MOVE = ADDRESS OF OLD 'TO' UNIT-Y
SET UP TO MOVE X CHARACTERS (AMOUNT OF DATA LEFT IN THE WORK AREA)
MOVE THE DATA
NEW WKACT = OLD WKACT + X; NEW INWKA = OLD INWKA - X
FA-9 A1
FA-10 A3
FIXSCSEG
IS THERE DISK QUEUING
YES
IS THE MESSAGE LOST IN MAIN STORAGE
YES
GET THE DISK ADDRESS OF THE CURRENT BUFFER
NO
NO
FA-17 A4
QTYPE1
SET THE DISK QUEUE TYPE IN THE SCB
FA-5 A1
BASESET
GET THE ADDRESS OF THE LAST BUFFER SENT
IS THIS THE LAST BUFFER SENT
YES
NO
HAS THE NEXT TEXT ARRIVED YET
YES
NO
FA-24 F5
FA-4 J1

Chart FA-11 CPB INITIALIZATION

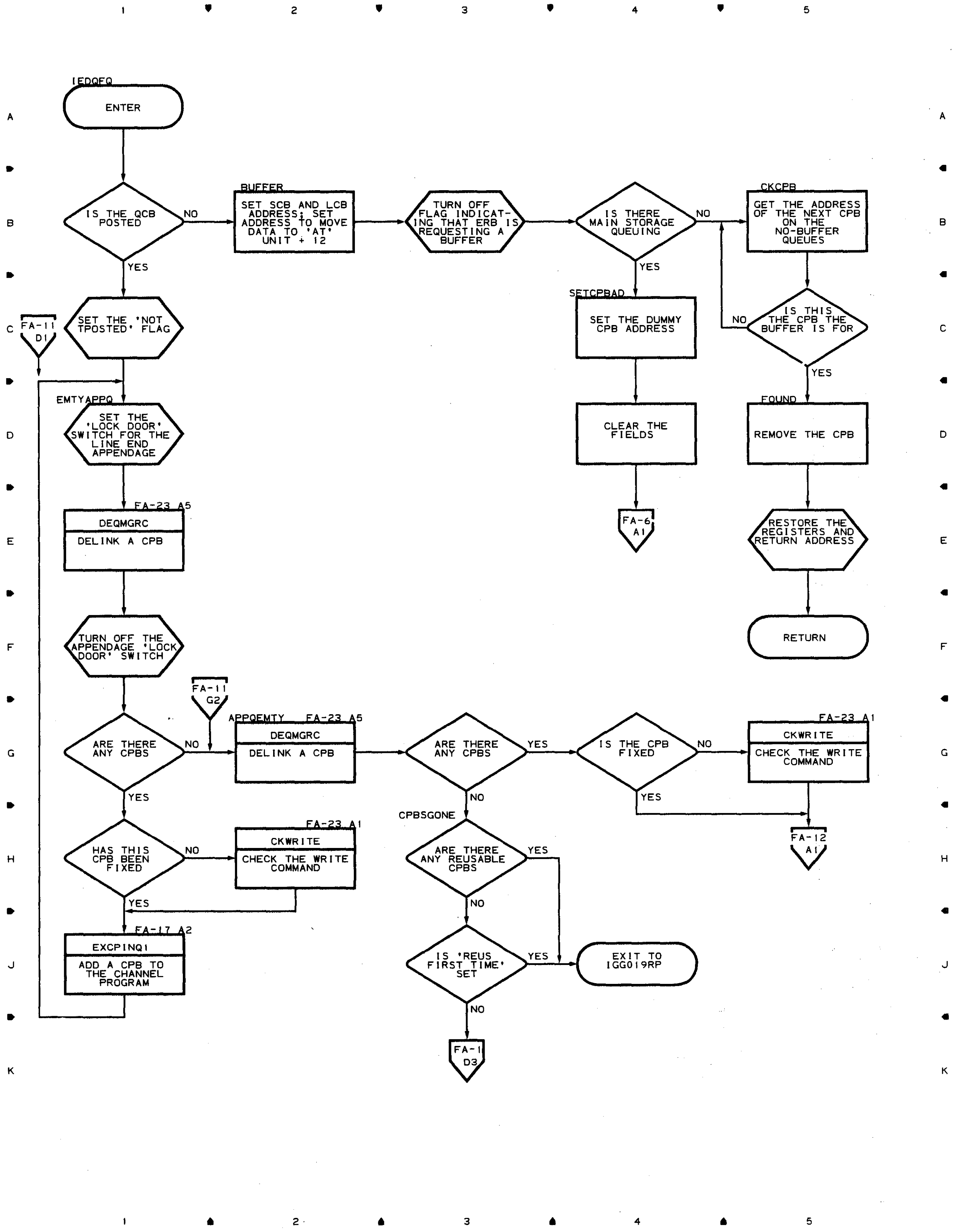

IEDQFQ
ENTER
IS THE QCB POSTED
NO
YES
BUFFER
SET SCB AND LCB ADDRESS; SET ADDRESS TO MOVE DATA TO 'AT' UNIT + 12
TURN OFF FLAG INDICATING THAT ERB IS REQUESTING A BUFFER
IS THERE MAIN STORAGE QUEUING
NO
YES
CKCPB
GET THE ADDRESS OF THE NEXT CPB ON THE NO-BUFFER QUEUES
IS THIS THE CPB THE BUFFER IS FOR
NO
YES
FA-11 D1
SET THE 'NOT TPOSTED' FLAG
SETCPBAD
SET THE DUMMY CPB ADDRESS
EMTYAPPQ
SET THE 'LOCK DOOR' SWITCH FOR THE LINE END APPENDAGE
CLEAR THE FIELDS
FOUND
REMOVE THE CPB
FA-23 A5
DEQMGRC
DELINK A CPB
FA-6 A1
RESTORE THE REGISTERS AND RETURN ADDRESS
TURN OFF THE APPENDAGE 'LOCK DOOR' SWITCH
RETURN
FA-11 G2
APPQEMTY FA-23 A5
DEQMGRC
DELINK A CPB
ARE THERE ANY CPBS
NO
YES
ARE THERE ANY CPBS
YES
NO
IS THE CPB FIXED
NO
YES
FA-23 A1
CKWRITE
CHECK THE WRITE COMMAND
CPBSGONE
FA-12 A1
HAS THIS CPB BEEN FIXED
NO
YES
FA-23 A1
CKWRITE
CHECK THE WRITE COMMAND
ARE THERE ANY REUSABLE CPBS
YES
NO
FA-17 A2
EXCPINQ1
ADD A CPB TO THE CHANNEL PROGRAM
IS 'REUS FIRST TIME' SET
YES
NO
EXIT TO IGG019RP
FA-1 D3
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart FA-12 CPB INITIALIZATION

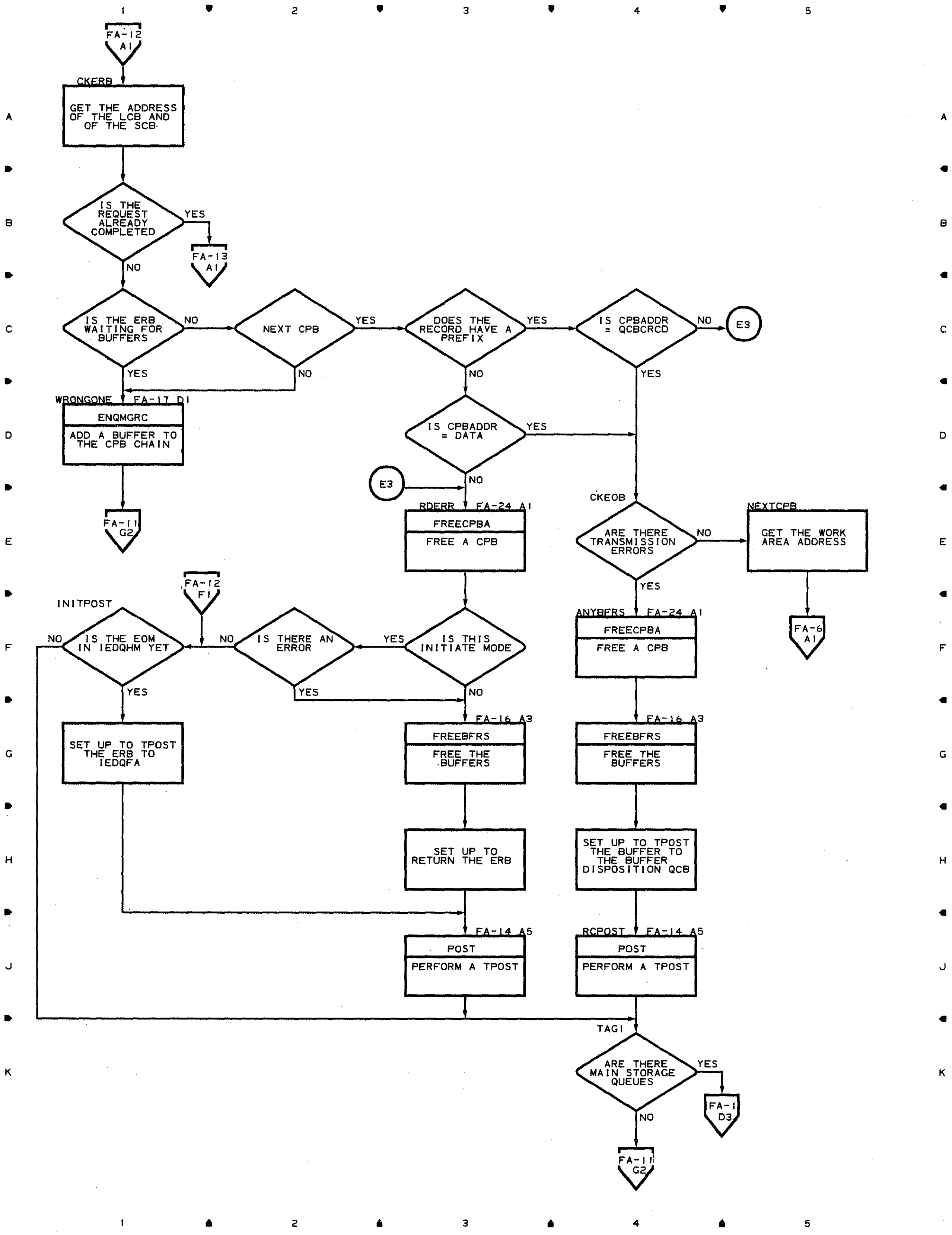


Chart FA-13 CPB INITIALIZATION

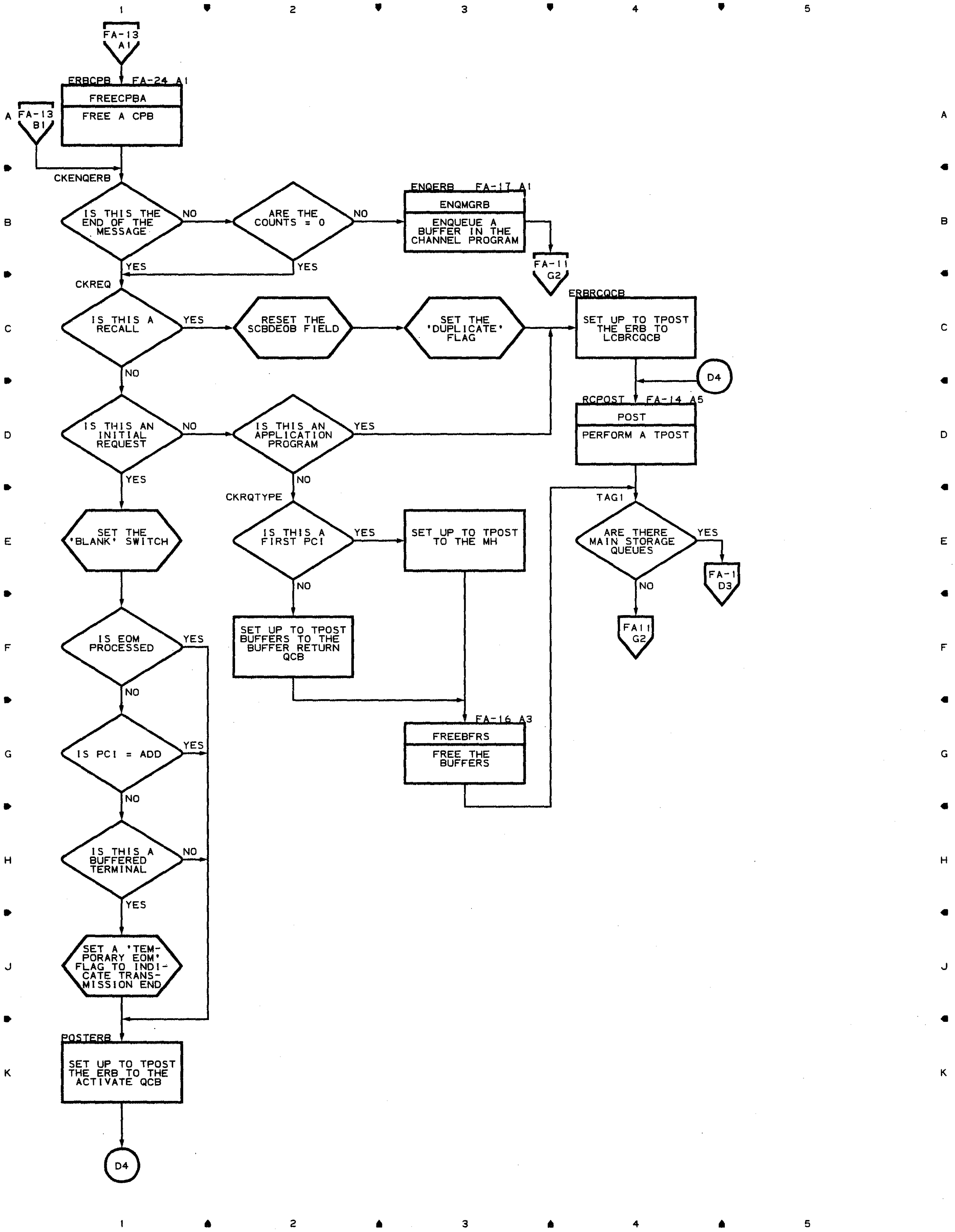


Chart FA-14 CPB INITIALIZATION

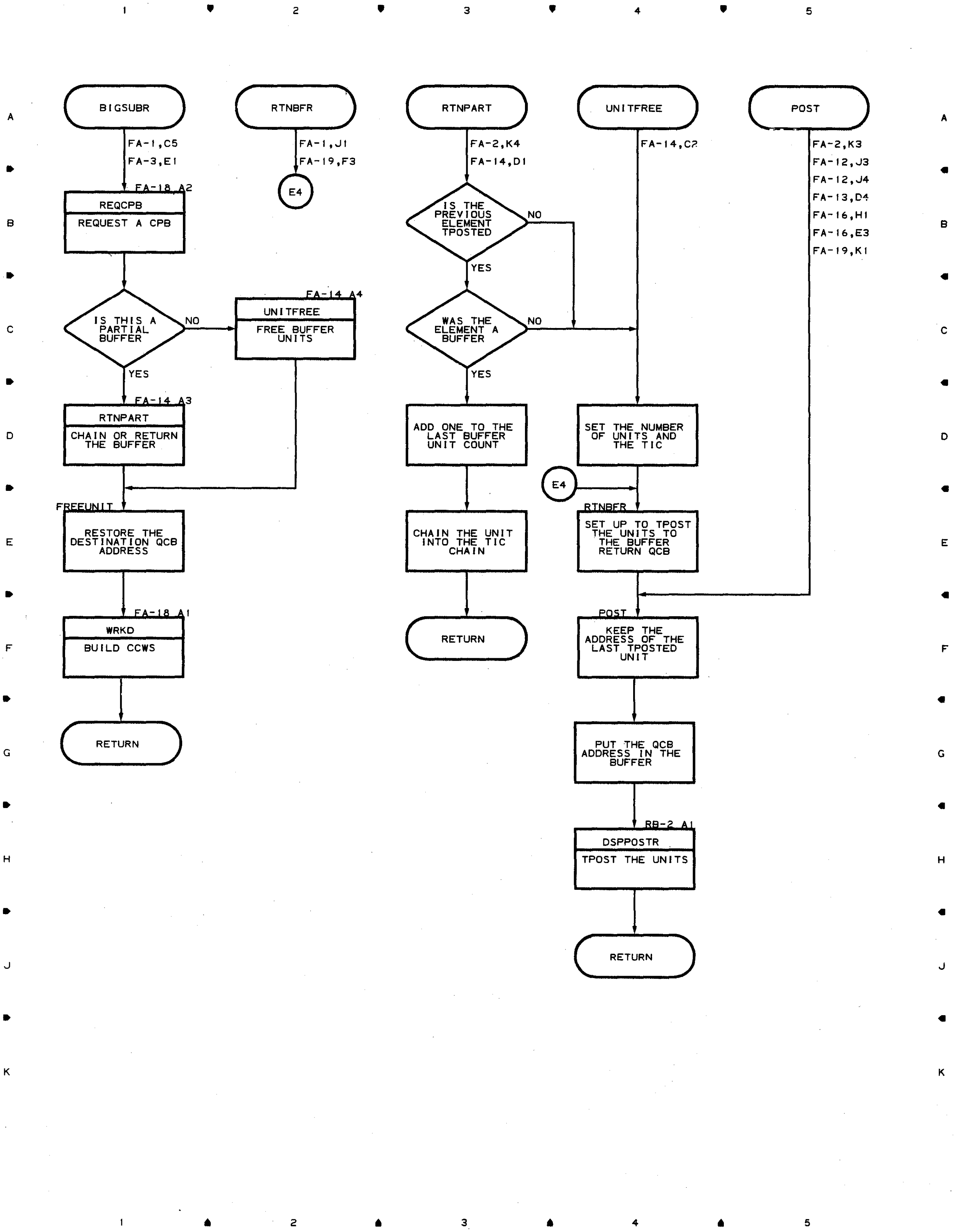
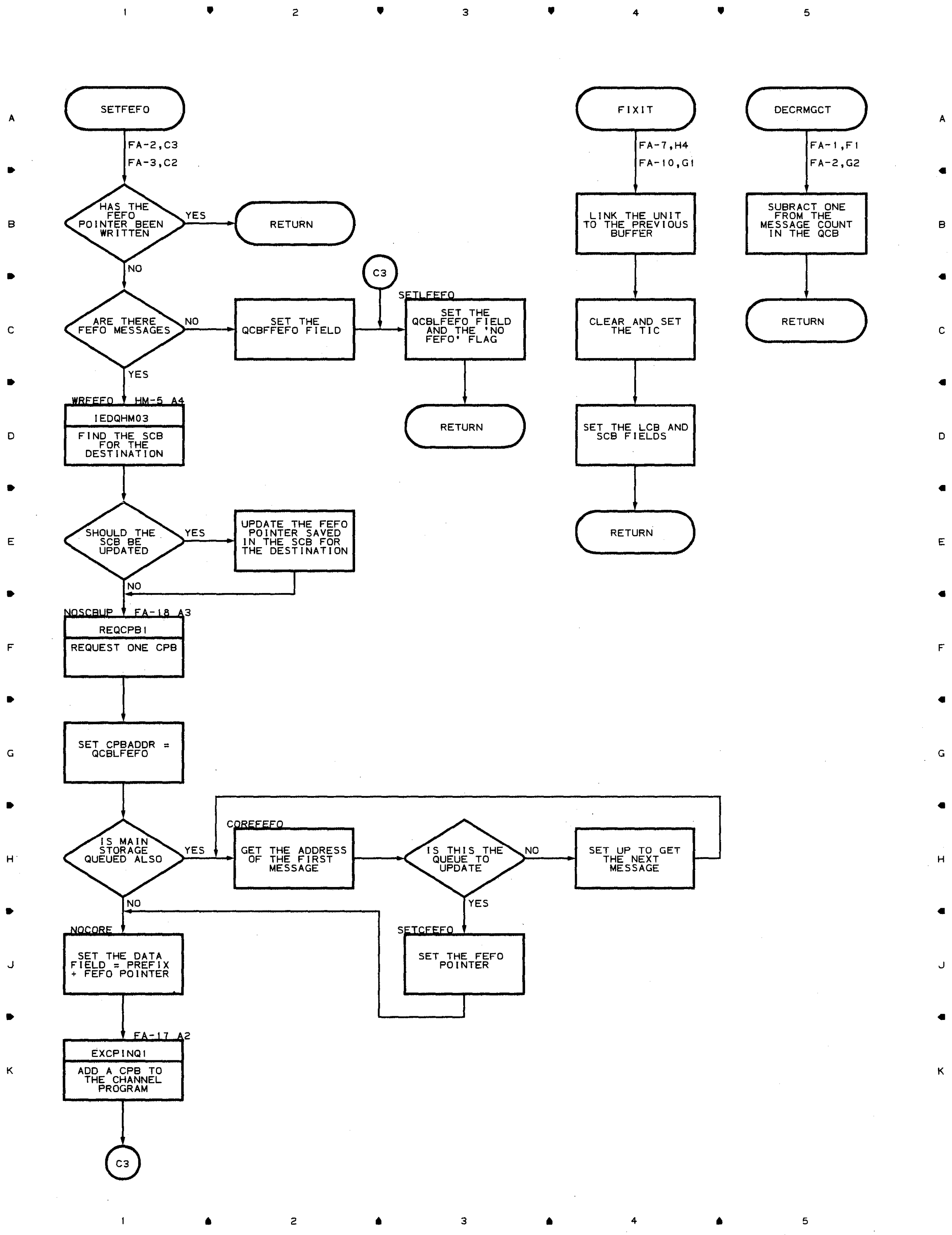


Chart FA-15 CPB INITIALIZATION



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
SETFEFO
FA-2,C3
FA-3,C2
HAS THE FEFO POINTER BEEN WRITTEN
YES
RETURN
NO
ARE THERE FEFO MESSAGES
NO
SET THE QCBFFEFO FIELD
C3
SETLFEFO
SET THE QCBLFEFO FIELD AND THE 'NO FEFO' FLAG
RETURN
YES
WRFEFO
HM-5 A4
IEDQHM03
FIND THE SCB FOR THE DESTINATION
SHOULD THE SCB BE UPDATED
YES
UPDATE THE FEFO POINTER SAVED IN THE SCB FOR THE DESTINATION
NO
NOSCBUP
FA-18 A3
REQCPB1
REQUEST ONE CPB
SET CPBADDR = QCBLFEFO
IS MAIN STORAGE QUEUED ALSO
YES
COREFEFO
GET THE ADDRESS OF THE FIRST MESSAGE
IS THIS THE QUEUE TO UPDATE
NO
SET UP TO GET THE NEXT MESSAGE
YES
SETCFEFO
SET THE FEFO POINTER
NO
NOCORE
SET THE DATA FIELD = PREFIX + FEFO POINTER
FA-17 A2
EXCPINQ1
ADD A CPB TO THE CHANNEL PROGRAM
C3
FIXIT
FA-7,H4
FA-10,G1
LINK THE UNIT TO THE PREVIOUS BUFFER
CLEAR AND SET THE TIC
SET THE LCB AND SCB FIELDS
RETURN
DECRMGCT
FA-1,F1
FA-2,G2
SUBRACT ONE FROM THE MESSAGE COUNT IN THE QCB
RETURN

Chart FA-16 CPB INITIALIZATION

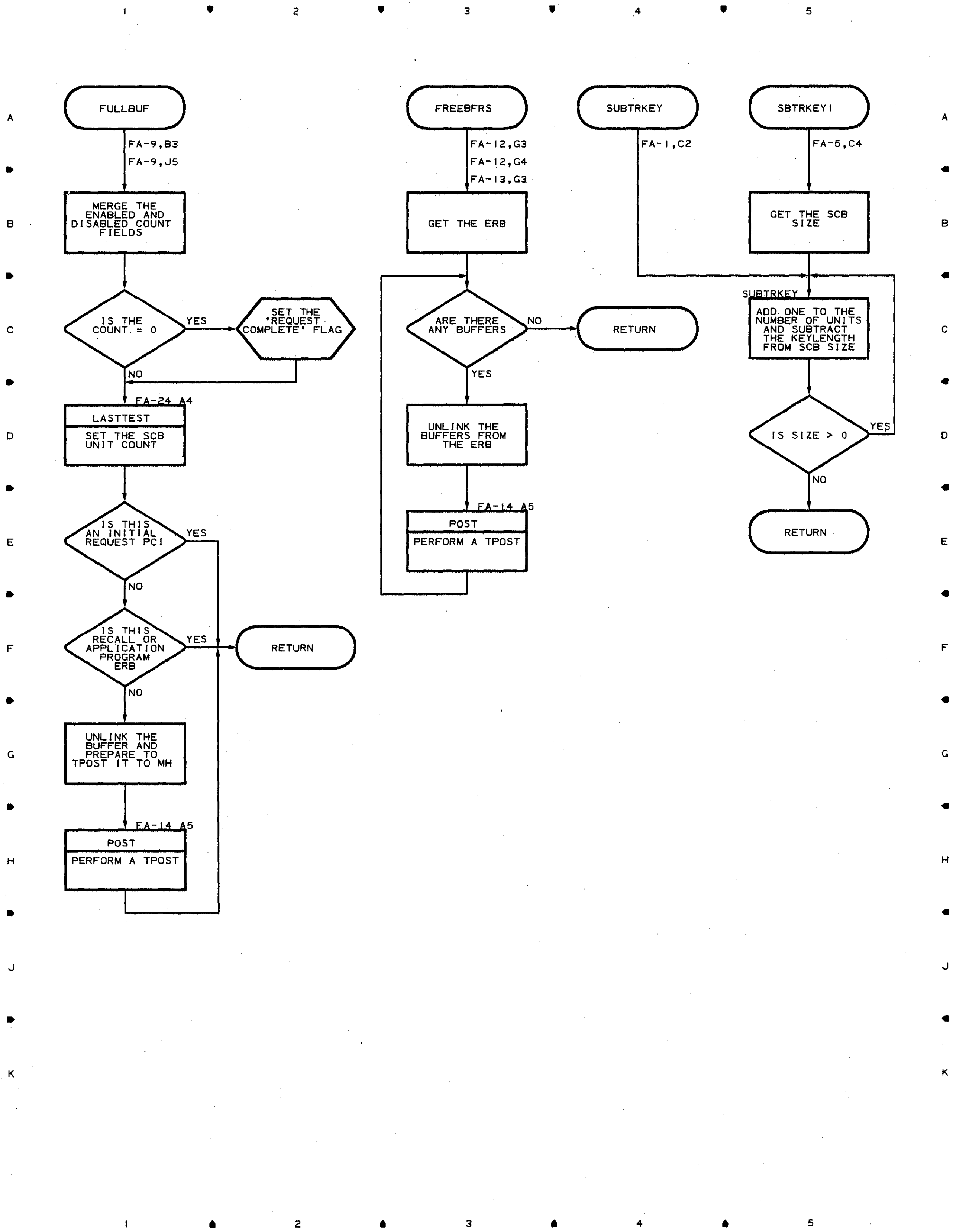


Chart FA-17 CPB INITIALIZATION

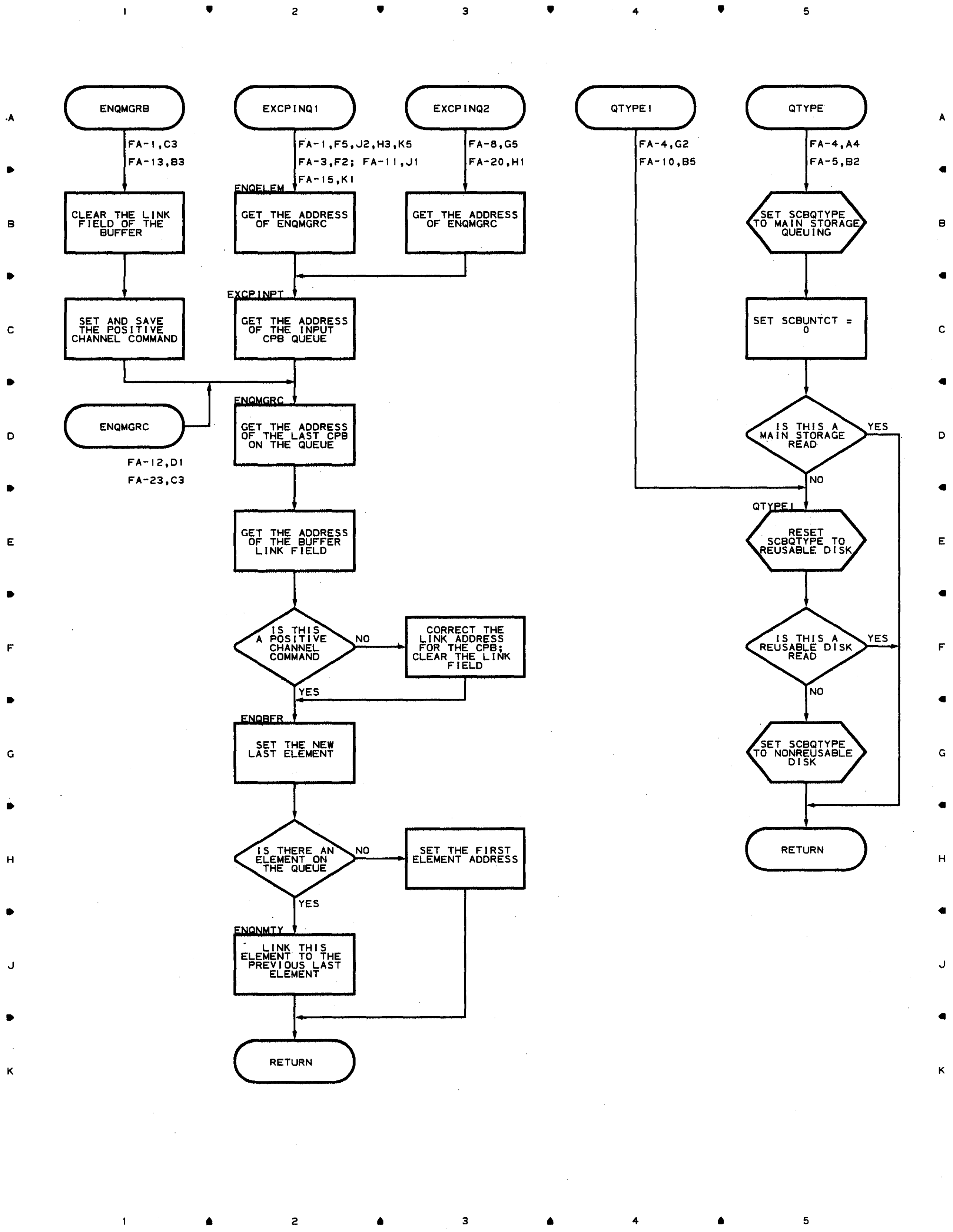
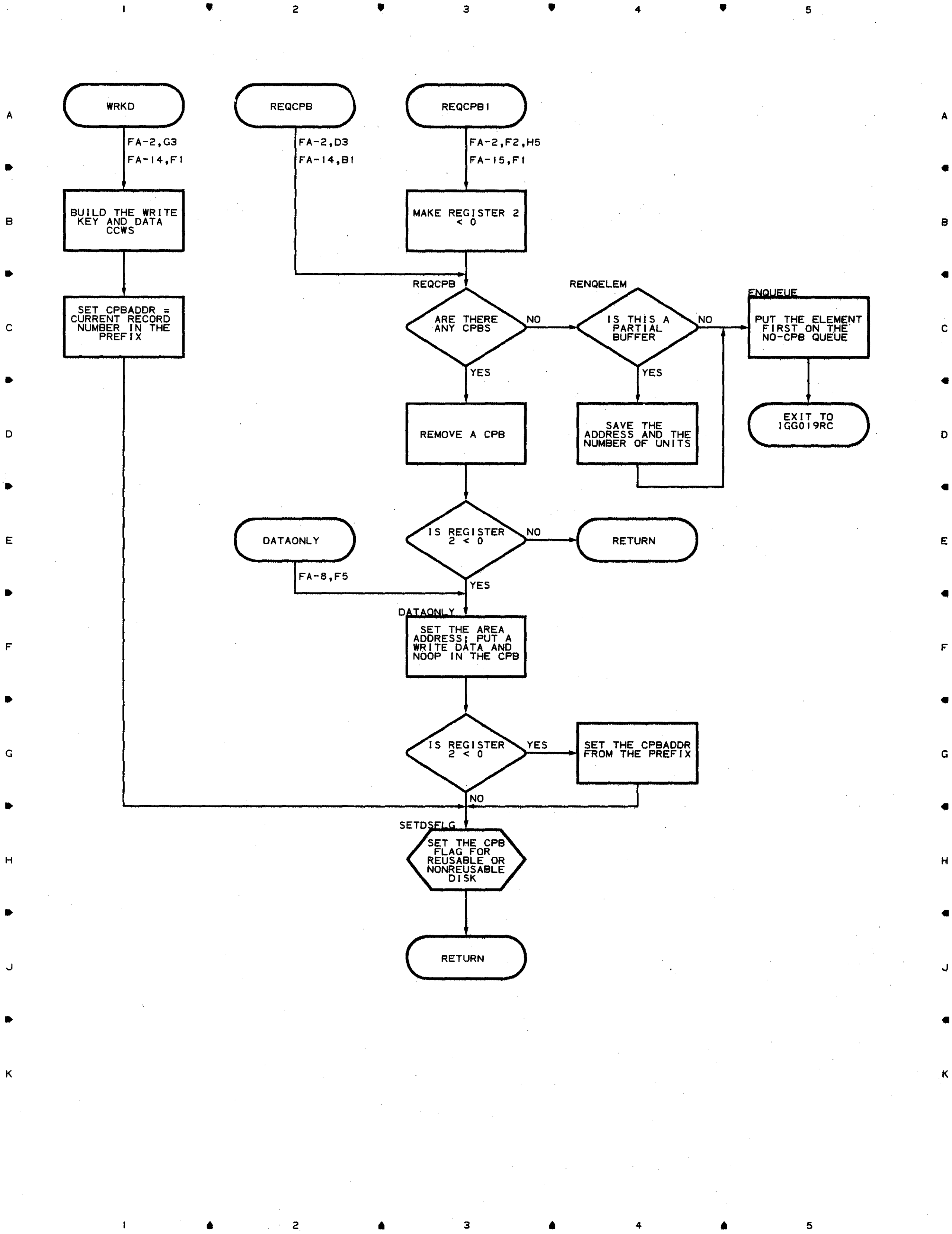

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
ENQMGRB
FA-1,C3
FA-13,B3
CLEAR THE LINK FIELD OF THE BUFFER
SET AND SAVE THE POSITIVE CHANNEL COMMAND
ENQMGRC
FA-12,D1
FA-23,C3
EXCPINQ1
FA-1,F5,J2,H3,K5
FA-3,F2; FA-11,J1
FA-15,K1
ENQELEM
GET THE ADDRESS OF ENQMGRC
EXCPINPT
GET THE ADDRESS OF THE INPUT CPB QUEUE
ENQMGRC
GET THE ADDRESS OF THE LAST CPB ON THE QUEUE
GET THE ADDRESS OF THE BUFFER LINK FIELD
IS THIS A POSITIVE CHANNEL COMMAND
NO
YES
ENQBFR
SET THE NEW LAST ELEMENT
IS THERE AN ELEMENT ON THE QUEUE
NO
YES
ENQNMTY
LINK THIS ELEMENT TO THE PREVIOUS LAST ELEMENT
RETURN
EXCPINQ2
FA-8,G5
FA-20,H1
GET THE ADDRESS OF ENQMGRC
CORRECT THE LINK ADDRESS FOR THE CPB; CLEAR THE LINK FIELD
SET THE FIRST ELEMENT ADDRESS
QTYPE1
FA-4,G2
FA-10,B5
QTYPE
FA-4,A4
FA-5,B2
SET SCBQTYPE TO MAIN STORAGE QUEUING
SET SCBUNTCT = 0
IS THIS A MAIN STORAGE READ
YES
NO
QTYPE1
RESET SCBQTYPE TO REUSABLE DISK
IS THIS A REUSABLE DISK READ
YES
NO
SET SCBQTYPE TO NONREUSABLE DISK
RETURN

Chart FA-18 CPB INITIALIZATION



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
WRKD
FA-2,G3
FA-14,F1
REQCPB
FA-2,D3
FA-14,B1
REQCPB1
FA-2,F2,H5
FA-15,F1
BUILD THE WRITE KEY AND DATA CCWS
MAKE REGISTER 2 < 0
SET CPBADDR = CURRENT RECORD NUMBER IN THE PREFIX
REQCPB
ARE THERE ANY CPBS
NO
RENQELEM
IS THIS A PARTIAL BUFFER
NO
ENQUEUE
PUT THE ELEMENT FIRST ON THE NO-CPB QUEUE
YES
YES
REMOVE A CPB
SAVE THE ADDRESS AND THE NUMBER OF UNITS
EXIT TO IGG019RC
DATAONLY
IS REGISTER 2 < 0
NO
RETURN
FA-8,F5
YES
DATAONLY
SET THE AREA ADDRESS; PUT A WRITE DATA AND NOOP IN THE CPB
IS REGISTER 2 < 0
YES
SET THE CPBADDR FROM THE PREFIX
NO
SETDSFLG
SET THE CPB FLAG FOR REUSABLE OR NONREUSABLE DISK
RETURN

Chart FA-19 CPB INITIALIZATION

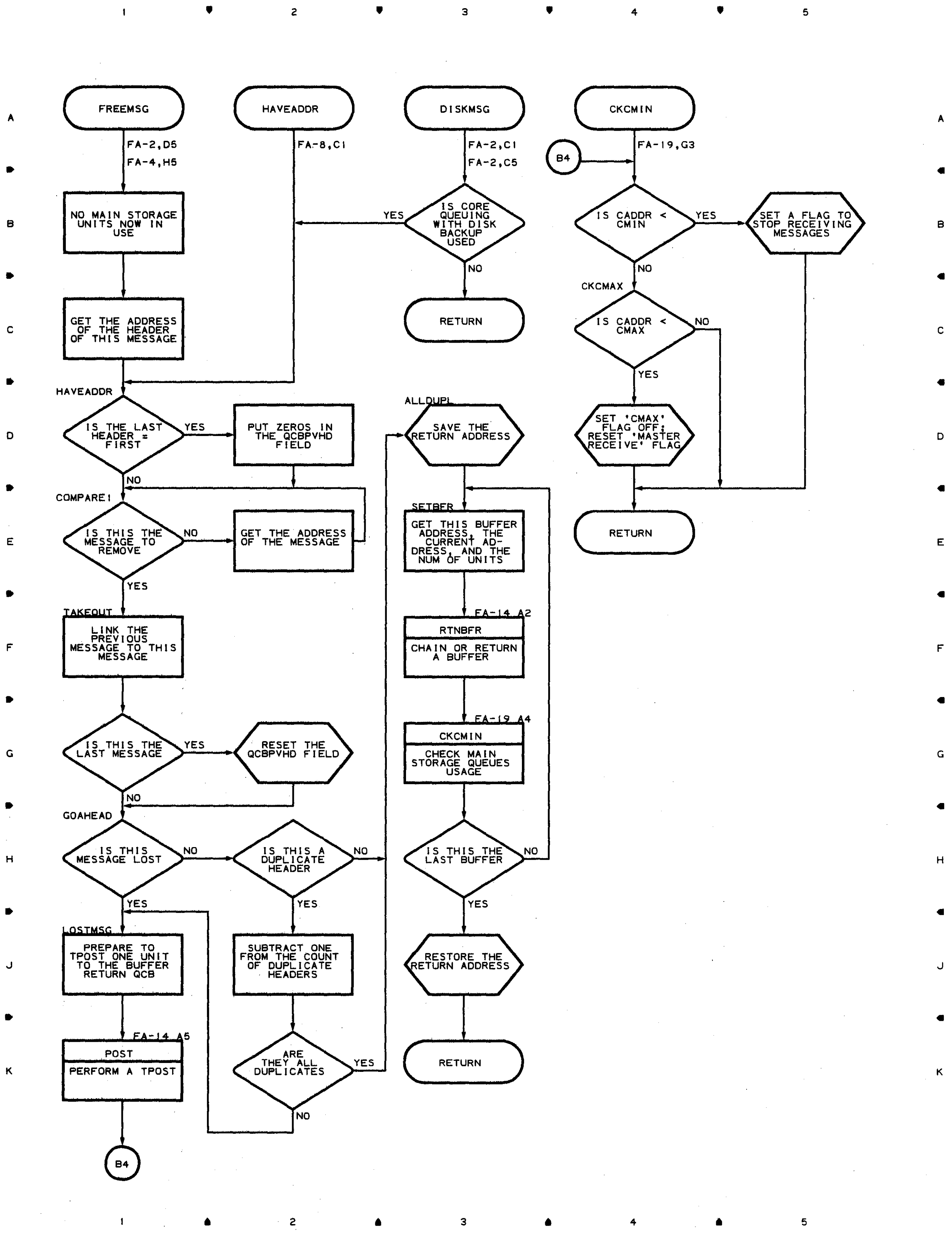
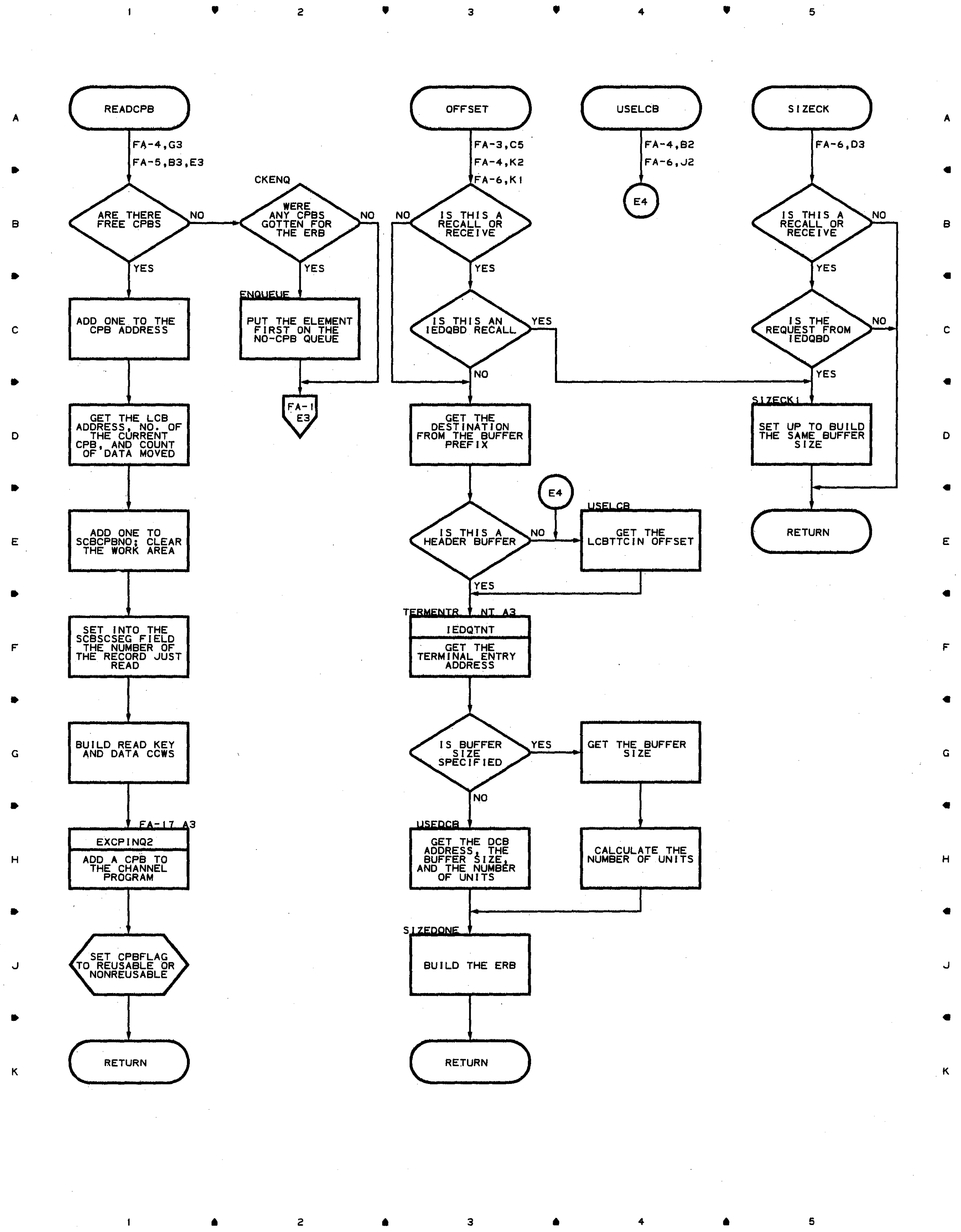

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
FREEMSG
FA-2,D5
FA-4,H5
NO MAIN STORAGE UNITS NOW IN USE
GET THE ADDRESS OF THE HEADER OF THIS MESSAGE
HAVEADDR
IS THE LAST HEADER = FIRST
YES
PUT ZEROS IN THE QCBPVHD FIELD
NO
COMPARE1
IS THIS THE MESSAGE TO REMOVE
NO
GET THE ADDRESS OF THE MESSAGE
YES
TAKEOUT
LINK THE PREVIOUS MESSAGE TO THIS MESSAGE
IS THIS THE LAST MESSAGE
YES
RESET THE QCBPVHD FIELD
NO
GOAHEAD
IS THIS MESSAGE LOST
NO
IS THIS A DUPLICATE HEADER
NO
YES
LOSTMSG
PREPARE TO TPOST ONE UNIT TO THE BUFFER RETURN QCB
YES
SUBTRACT ONE FROM THE COUNT OF DUPLICATE HEADERS
FA-14 A5
POST
PERFORM A TPOST
ARE THEY ALL DUPLICATES
YES
NO
B4
HAVEADDR
FA-8,C1
DISKMSG
FA-2,C1
FA-2,C5
IS CORE QUEUING WITH DISK BACKUP USED
YES
NO
RETURN
ALLDUPL
SAVE THE RETURN ADDRESS
SETBFR
GET THIS BUFFER ADDRESS, THE CURRENT AD-DRESS, AND THE NUM OF UNITS
FA-14 A2
RTNBFR
CHAIN OR RETURN A BUFFER
FA-19 A4
CKCMIN
CHECK MAIN STORAGE QUEUES USAGE
IS THIS THE LAST BUFFER
NO
YES
RESTORE THE RETURN ADDRESS
RETURN
CKCMIN
B4
FA-19,G3
IS CADDR < CMIN
YES
SET A FLAG TO STOP RECEIVING MESSAGES
NO
CKCMAX
IS CADDR < CMAX
NO
YES
SET 'CMAX' FLAG OFF; RESET 'MASTER RECEIVE' FLAG
RETURN

Chart FA-20 CPB INITIALIZATION



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
READCPB
FA-4,G3
FA-5,B3,E3
ARE THERE FREE CPBS
NO
YES
CKENQ
WERE ANY CPBS GOTTEN FOR THE ERB
NO
YES
ADD ONE TO THE CPB ADDRESS
ENQUEUE
PUT THE ELEMENT FIRST ON THE NO-CPB QUEUE
FA-1 E3
GET THE LCB ADDRESS, NO. OF THE CURRENT CPB, AND COUNT OF DATA MOVED
ADD ONE TO SCBCPBNO; CLEAR THE WORK AREA
SET INTO THE SCBSCSEG FIELD THE NUMBER OF THE RECORD JUST READ
BUILD READ KEY AND DATA CCWS
FA-17 A3
EXCPINQ2
ADD A CPB TO THE CHANNEL PROGRAM
SET CPBFLAG TO REUSABLE OR NONREUSABLE
RETURN
OFFSET
FA-3,C5
FA-4,K2
FA-6,K1
IS THIS A RECALL OR RECEIVE
NO
YES
IS THIS AN IEDQBD RECALL
YES
NO
GET THE DESTINATION FROM THE BUFFER PREFIX
E4
IS THIS A HEADER BUFFER
NO
YES
USELCB
GET THE LCBTTCIN OFFSET
TERMENTR
NT A3
IEDQTNT
GET THE TERMINAL ENTRY ADDRESS
IS BUFFER SIZE SPECIFIED
YES
NO
GET THE BUFFER SIZE
USEDCB
GET THE DCB ADDRESS, THE BUFFER SIZE, AND THE NUMBER OF UNITS
CALCULATE THE NUMBER OF UNITS
SIZEDONE
BUILD THE ERB
RETURN
USELCB
FA-4,B2
FA-6,J2
E4
SIZECK
FA-6,D3
IS THIS A RECALL OR RECEIVE
NO
YES
IS THE REQUEST FROM IEDQBD
NO
YES
SIZECK1
SET UP TO BUILD THE SAME BUFFER SIZE
RETURN

Chart FA-21 CPB INITIALIZATION

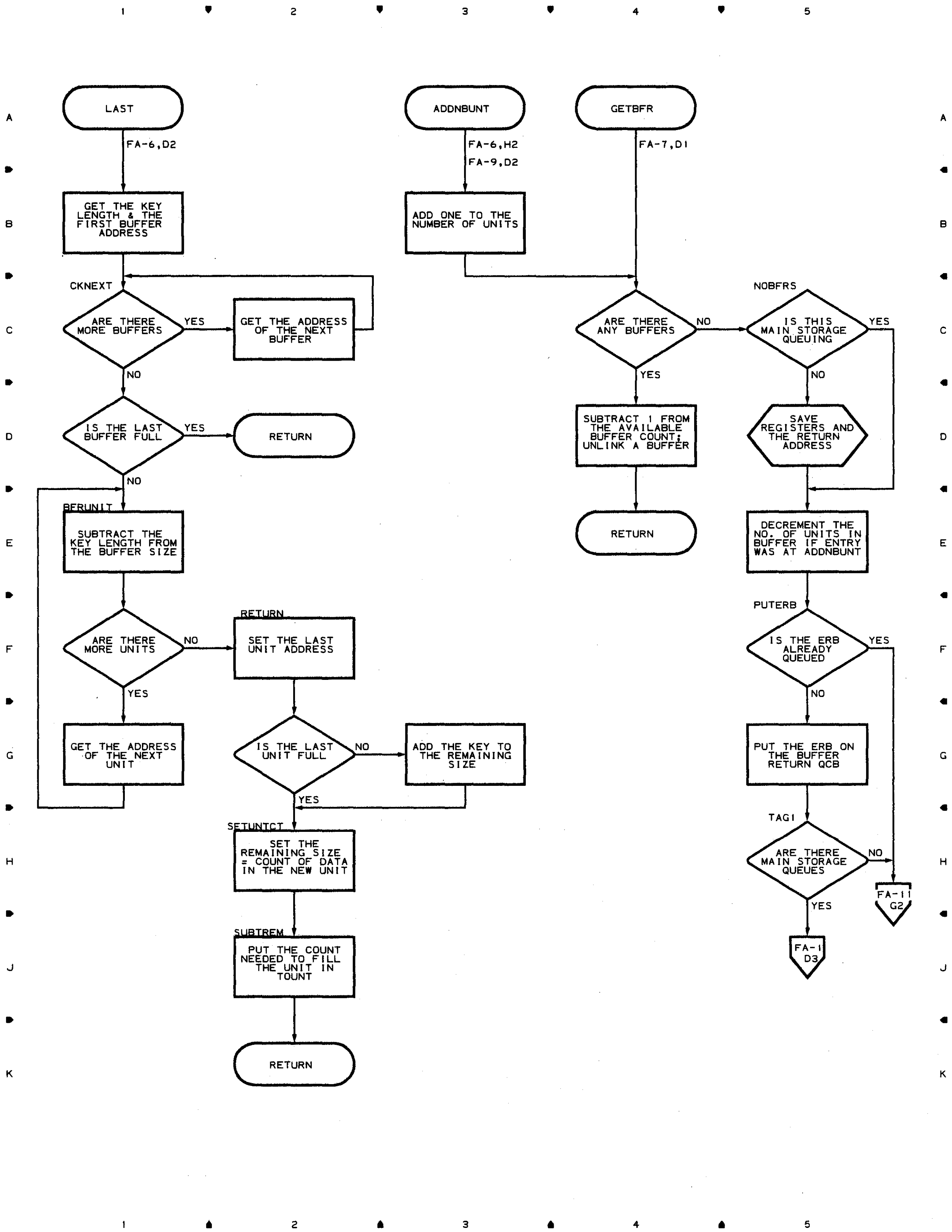

LAST
FA-6,D2
GET THE KEY LENGTH & THE FIRST BUFFER ADDRESS
CKNEXT
ARE THERE MORE BUFFERS
YES
GET THE ADDRESS OF THE NEXT BUFFER
NO
IS THE LAST BUFFER FULL
YES
RETURN
NO
BFRUNIT
SUBTRACT THE KEY LENGTH FROM THE BUFFER SIZE
ARE THERE MORE UNITS
NO
YES
GET THE ADDRESS OF THE NEXT UNIT
RETURN
SET THE LAST UNIT ADDRESS
IS THE LAST UNIT FULL
NO
ADD THE KEY TO THE REMAINING SIZE
YES
SETUNTCT
SET THE REMAINING SIZE = COUNT OF DATA IN THE NEW UNIT
SUBTREM
PUT THE COUNT NEEDED TO FILL THE UNIT IN TOUNT
RETURN
ADDNBUNT
FA-6,H2
FA-9,D2
ADD ONE TO THE NUMBER OF UNITS
GETBFR
FA-7,D1
ARE THERE ANY BUFFERS
NO
YES
SUBTRACT 1 FROM THE AVAILABLE BUFFER COUNT; UNLINK A BUFFER
RETURN
NOBFRS
IS THIS MAIN STORAGE QUEUING
YES
NO
SAVE REGISTERS AND THE RETURN ADDRESS
DECREMENT THE NO. OF UNITS IN BUFFER IF ENTRY WAS AT ADDNBUNT
PUTERB
IS THE ERB ALREADY QUEUED
YES
NO
PUT THE ERB ON THE BUFFER RETURN QCB
TAG1
ARE THERE MAIN STORAGE QUEUES
NO
FA-11 G2
YES
FA-1 D3
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart FA-22 CPB INITIALIZATION

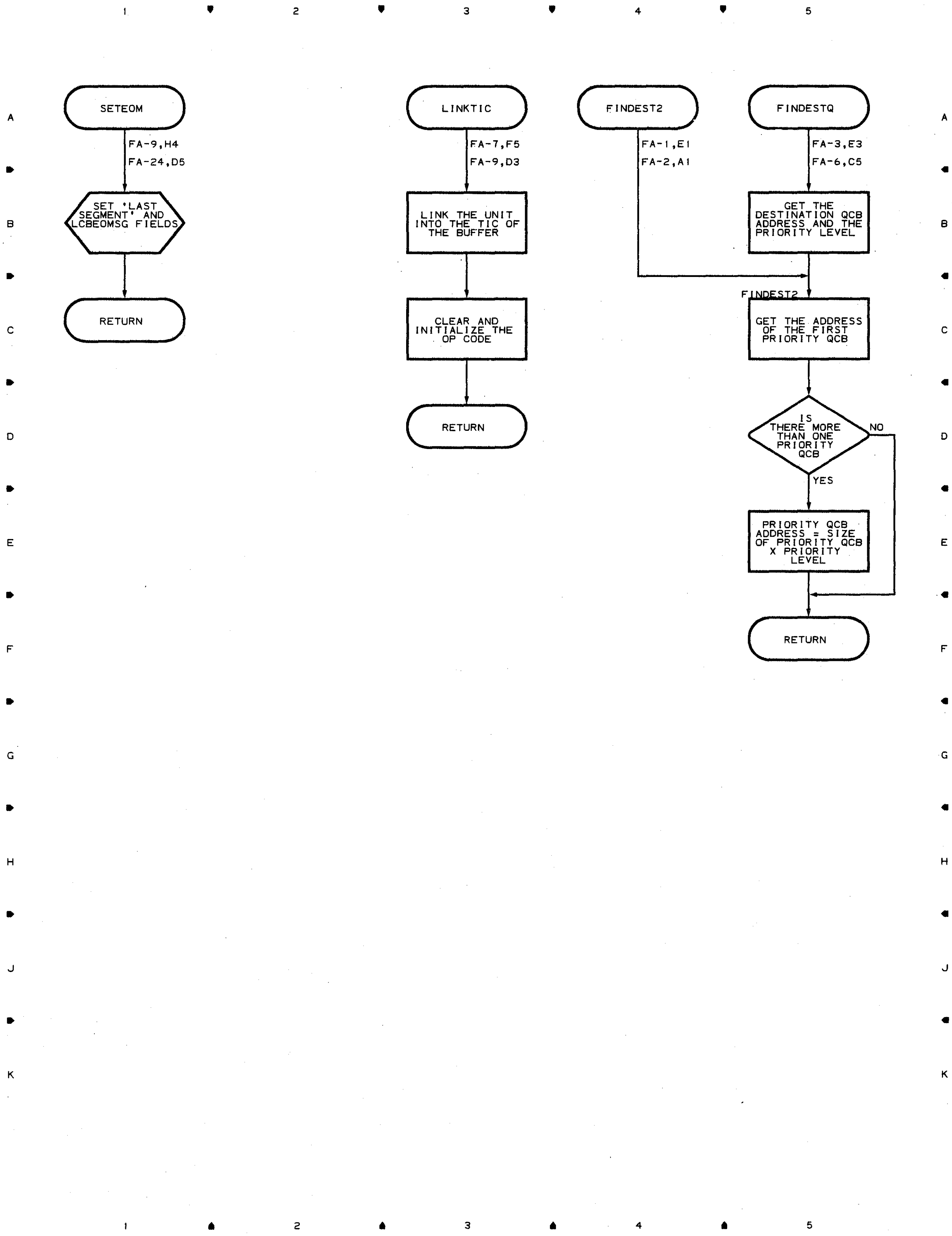
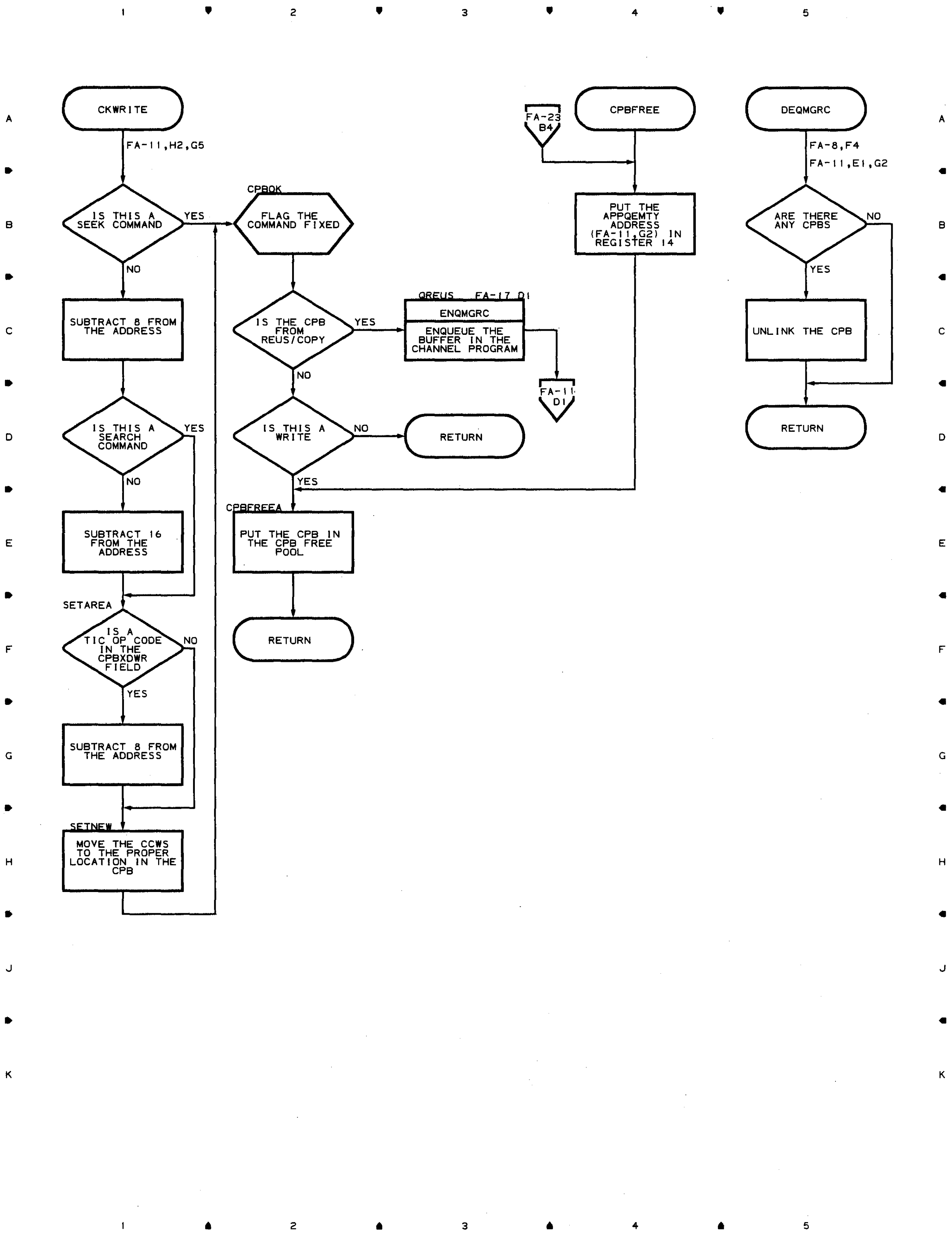

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
SETEOM
FA-9,H4
FA-24,D5
SET 'LAST SEGMENT' AND LCBEOMSG FIELDS
RETURN
LINKTIC
FA-7,F5
FA-9,D3
LINK THE UNIT INTO THE TIC OF THE BUFFER
CLEAR AND INITIALIZE THE OP CODE
RETURN
FINDEST2
FA-1,E1
FA-2,A1
FINDESTQ
FA-3,E3
FA-6,C5
GET THE DESTINATION QCB ADDRESS AND THE PRIORITY LEVEL
FINDEST2
GET THE ADDRESS OF THE FIRST PRIORITY QCB
IS THERE MORE THAN ONE PRIORITY QCB
NO
YES
PRIORITY QCB ADDRESS = SIZE OF PRIORITY QCB X PRIORITY LEVEL
RETURN

Chart FA-23 CPB INITIALIZATION



CKWRITE
FA-11,H2,G5
IS THIS A SEEK COMMAND
YES
NO
SUBTRACT 8 FROM THE ADDRESS
IS THIS A SEARCH COMMAND
YES
NO
SUBTRACT 16 FROM THE ADDRESS
SETAREA
IS A TIC OP CODE IN THE CPBXDWR FIELD
NO
YES
SUBTRACT 8 FROM THE ADDRESS
SETNEW
MOVE THE CCWS TO THE PROPER LOCATION IN THE CPB
CPBOK
FLAG THE COMMAND FIXED
IS THE CPB FROM REUS/COPY
YES
NO
QREUS FA-17 D1
ENQMGRC
ENQUEUE THE BUFFER IN THE CHANNEL PROGRAM
FA-11 D1
IS THIS A WRITE
NO
RETURN
YES
CPBFREEA
PUT THE CPB IN THE CPB FREE POOL
RETURN
FA-23 B4
CPBFREE
PUT THE APPQEMTY ADDRESS (FA-11,G2) IN REGISTER 14
DEQMGRC
FA-8,F4
FA-11,E1,G2
ARE THERE ANY CPBS
NO
YES
UNLINK THE CPB
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart FA-24 CPB INITIALIZATION

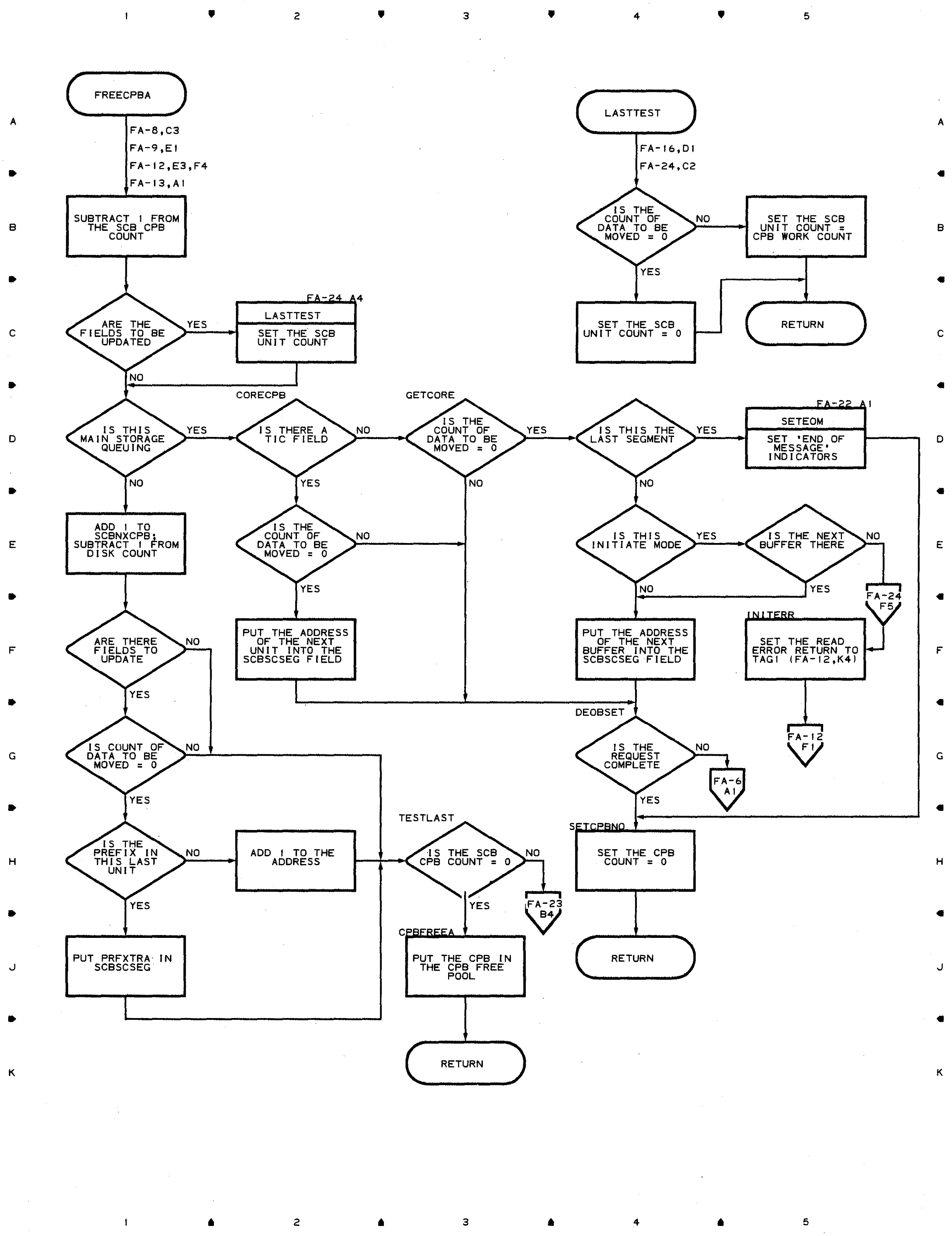

FREECPBA
FA-8,C3
FA-9,E1
FA-12,E3,F4
FA-13,A1
SUBTRACT 1 FROM THE SCB CPB COUNT
ARE THE FIELDS TO BE UPDATED
YES
FA-24 A4
LASTTEST
SET THE SCB UNIT COUNT
NO
IS THIS MAIN STORAGE QUEUING
YES
NO
CORECPB
IS THERE A TIC FIELD
NO
YES
GETCORE
IS THE COUNT OF DATA TO BE MOVED = 0
YES
NO
ADD 1 TO SCBNXCPB; SUBTRACT 1 FROM DISK COUNT
IS THE COUNT OF DATA TO BE MOVED = 0
NO
YES
PUT THE ADDRESS OF THE NEXT UNIT INTO THE SCBSCSEG FIELD
ARE THERE FIELDS TO UPDATE
NO
YES
IS COUNT OF DATA TO BE MOVED = 0
NO
YES
IS THE PREFIX IN THIS LAST UNIT
NO
ADD 1 TO THE ADDRESS
YES
PUT PRFXTRA IN SCBSCSEG
TESTLAST
IS THE SCB CPB COUNT = 0
NO
FA-23 B4
YES
CPBFREEA
PUT THE CPB IN THE CPB FREE POOL
RETURN
LASTTEST
FA-16,D1
FA-24,C2
IS THE COUNT OF DATA TO BE MOVED = 0
NO
SET THE SCB UNIT COUNT = CPB WORK COUNT
YES
SET THE SCB UNIT COUNT = 0
RETURN
IS THIS THE LAST SEGMENT
YES
FA-22 A1
SETEOM
SET 'END OF MESSAGE' INDICATORS
NO
IS THIS INITIATE MODE
YES
IS THE NEXT BUFFER THERE
NO
FA-24 F5
YES
NO
INITERR
SET THE READ ERROR RETURN TO TAG1 (FA-12,K4)
FA-12 F1
PUT THE ADDRESS OF THE NEXT BUFFER INTO THE SCBSCSEG FIELD
DEOBSET
IS THE REQUEST COMPLETE
NO
FA-6 A1
YES
SETCPBNO
SET THE CPB COUNT = 0
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart FA1-1 CPB INITIALIZATION - MAIN STORAGE ONLY QUEUING

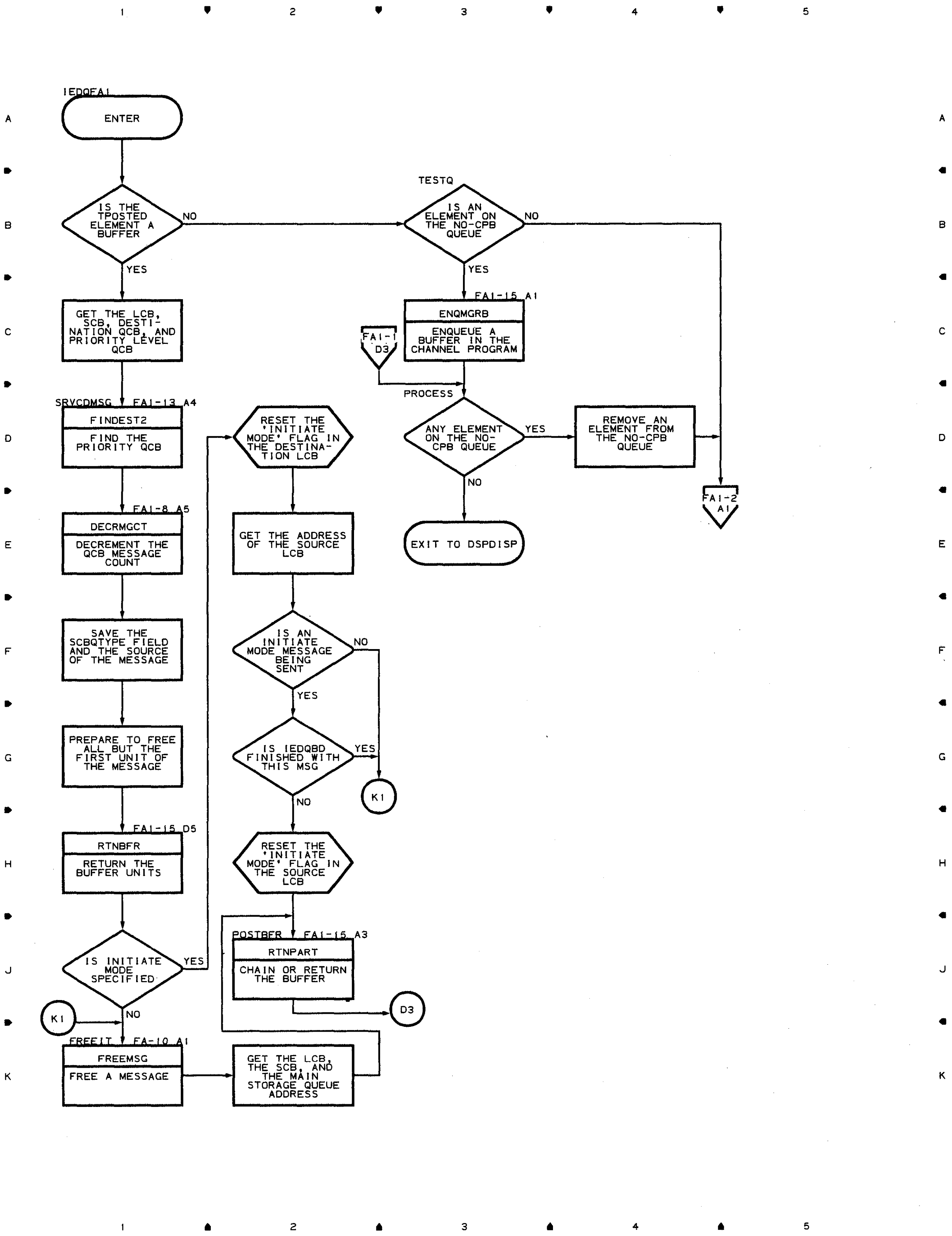
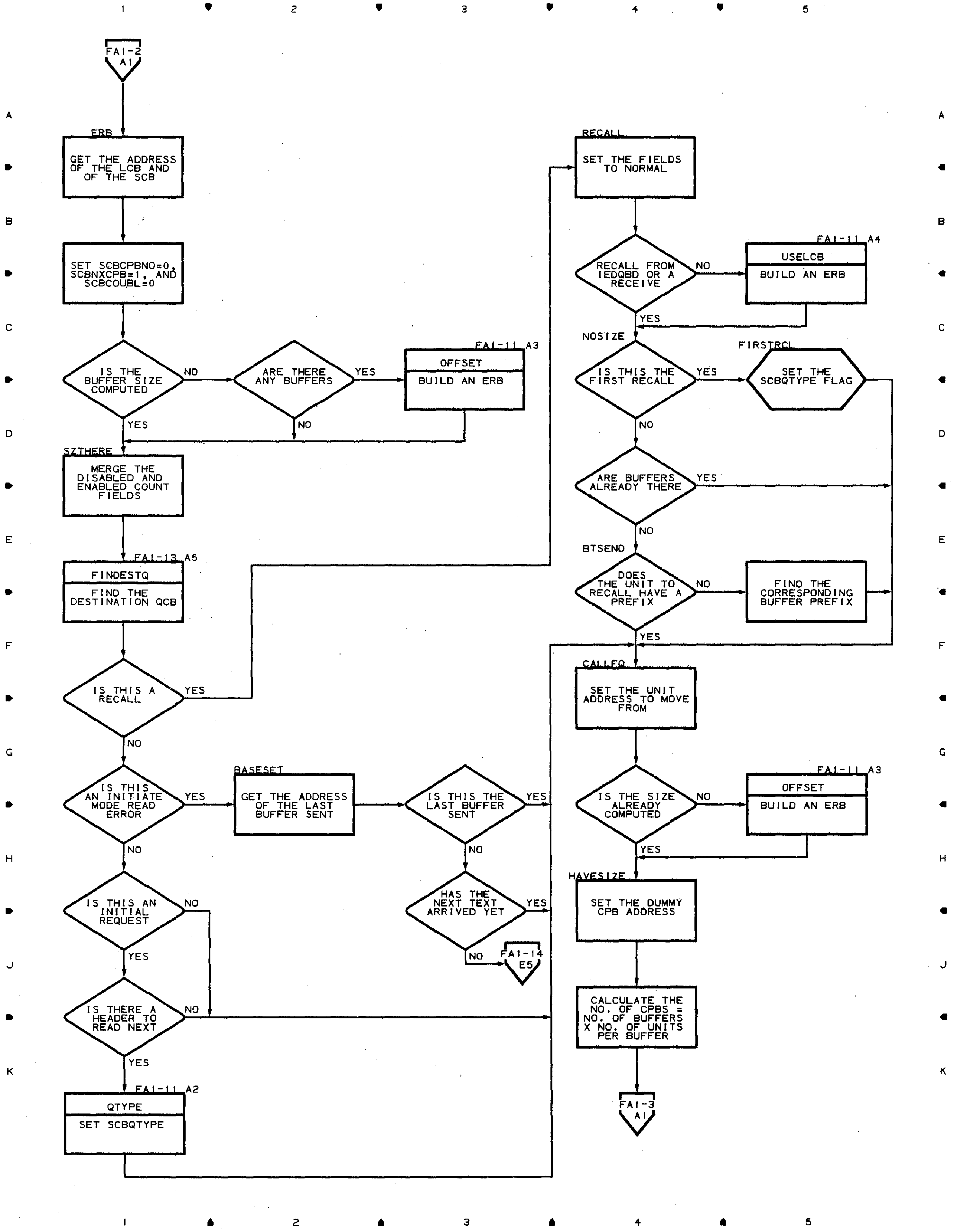

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQFA1
ENTER
IS THE TPOSTED ELEMENT A BUFFER
NO
YES
GET THE LCB, SCB, DESTINATION QCB, AND PRIORITY LEVEL QCB
SRVCDMSG
FA1-13 A4
FINDEST2
FIND THE PRIORITY QCB
FA1-8 A5
DECRMGCT
DECREMENT THE QCB MESSAGE COUNT
SAVE THE SCBQTYPE FIELD AND THE SOURCE OF THE MESSAGE
PREPARE TO FREE ALL BUT THE FIRST UNIT OF THE MESSAGE
FA1-15 D5
RTNBFR
RETURN THE BUFFER UNITS
IS INITIATE MODE SPECIFIED
YES
NO
K1
FREEIT
FA-10 A1
FREEMSG
FREE A MESSAGE
RESET THE 'INITIATE MODE' FLAG IN THE DESTINATION LCB
GET THE ADDRESS OF THE SOURCE LCB
IS AN INITIATE MODE MESSAGE BEING SENT
NO
YES
IS IEDQBD FINISHED WITH THIS MSG
YES
NO
K1
RESET THE 'INITIATE MODE' FLAG IN THE SOURCE LCB
POSTBFR
FA1-15 A3
RTNPART
CHAIN OR RETURN THE BUFFER
D3
GET THE LCB, THE SCB, AND THE MAIN STORAGE QUEUE ADDRESS
TESTQ
IS AN ELEMENT ON THE NO-CPB QUEUE
NO
YES
FA1-15 A1
ENQMGRB
ENQUEUE A BUFFER IN THE CHANNEL PROGRAM
FA1-1 D3
PROCESS
ANY ELEMENT ON THE NO-CPB QUEUE
YES
NO
REMOVE AN ELEMENT FROM THE NO-CPB QUEUE
EXIT TO DSPDISP
FA1-2 A1

Chart FA1-2 CPB INITIALIZATION - MAIN STORAGE ONLY QUEUING



FA1-2 A1
ERB
GET THE ADDRESS OF THE LCB AND OF THE SCB
SET SCBCPBNO=0, SCBNXCPB=1, AND SCBCOUBL=0
IS THE BUFFER SIZE COMPUTED
NO
YES
ARE THERE ANY BUFFERS
YES
NO
FA1-11 A3
OFFSET
BUILD AN ERB
SZTHERE
MERGE THE DISABLED AND ENABLED COUNT FIELDS
FA1-13 A5
FINDESTQ
FIND THE DESTINATION QCB
IS THIS A RECALL
YES
NO
IS THIS AN INITIATE MODE READ ERROR
YES
NO
BASESET
GET THE ADDRESS OF THE LAST BUFFER SENT
IS THIS THE LAST BUFFER SENT
YES
NO
HAS THE NEXT TEXT ARRIVED YET
YES
NO
FA1-14 E5
IS THIS AN INITIAL REQUEST
NO
YES
IS THERE A HEADER TO READ NEXT
NO
YES
FA1-11 A2
QTYPE
SET SCBQTYPE
RECALL
SET THE FIELDS TO NORMAL
RECALL FROM IEDQBD OR A RECEIVE
NO
YES
FA1-11 A4
USELCB
BUILD AN ERB
NOSIZE
IS THIS THE FIRST RECALL
YES
NO
FIRSTRCL
SET THE SCBQTYPE FLAG
ARE BUFFERS ALREADY THERE
YES
NO
BTSEND
DOES THE UNIT TO RECALL HAVE A PREFIX
NO
YES
FIND THE CORRESPONDING BUFFER PREFIX
CALLFQ
SET THE UNIT ADDRESS TO MOVE FROM
IS THE SIZE ALREADY COMPUTED
NO
YES
FA1-11 A3
OFFSET
BUILD AN ERB
HAVESIZE
SET THE DUMMY CPB ADDRESS
CALCULATE THE NO. OF CPBS = NO. OF BUFFERS X NO. OF UNITS PER BUFFER
FA1-3 A1
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart FA1-3 CPB INITIALIZATION - MAIN STORAGE ONLY QUEUING

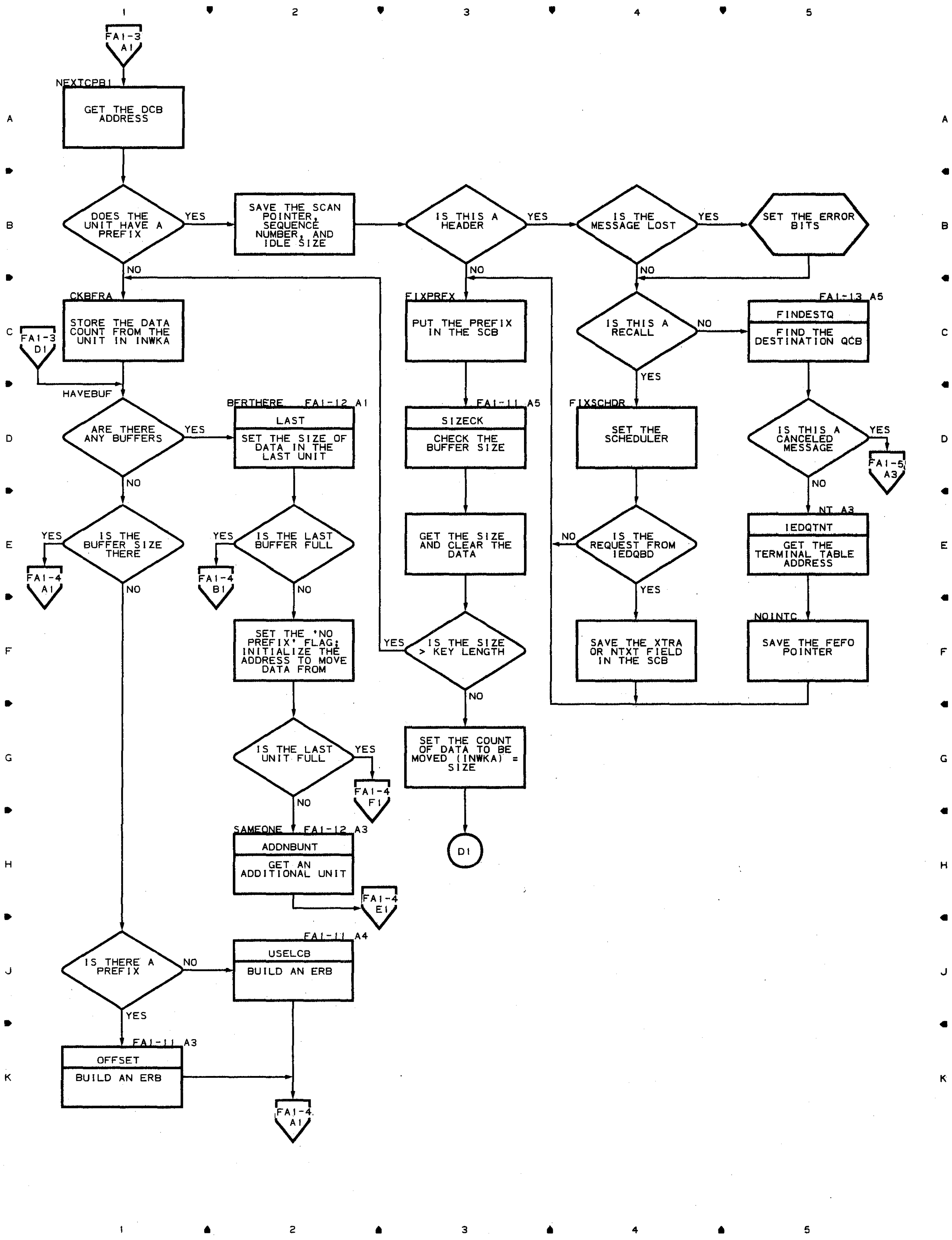


Chart FA1-4 CPB INITIALIZATION - MAIN STORAGE ONLY QUEUING

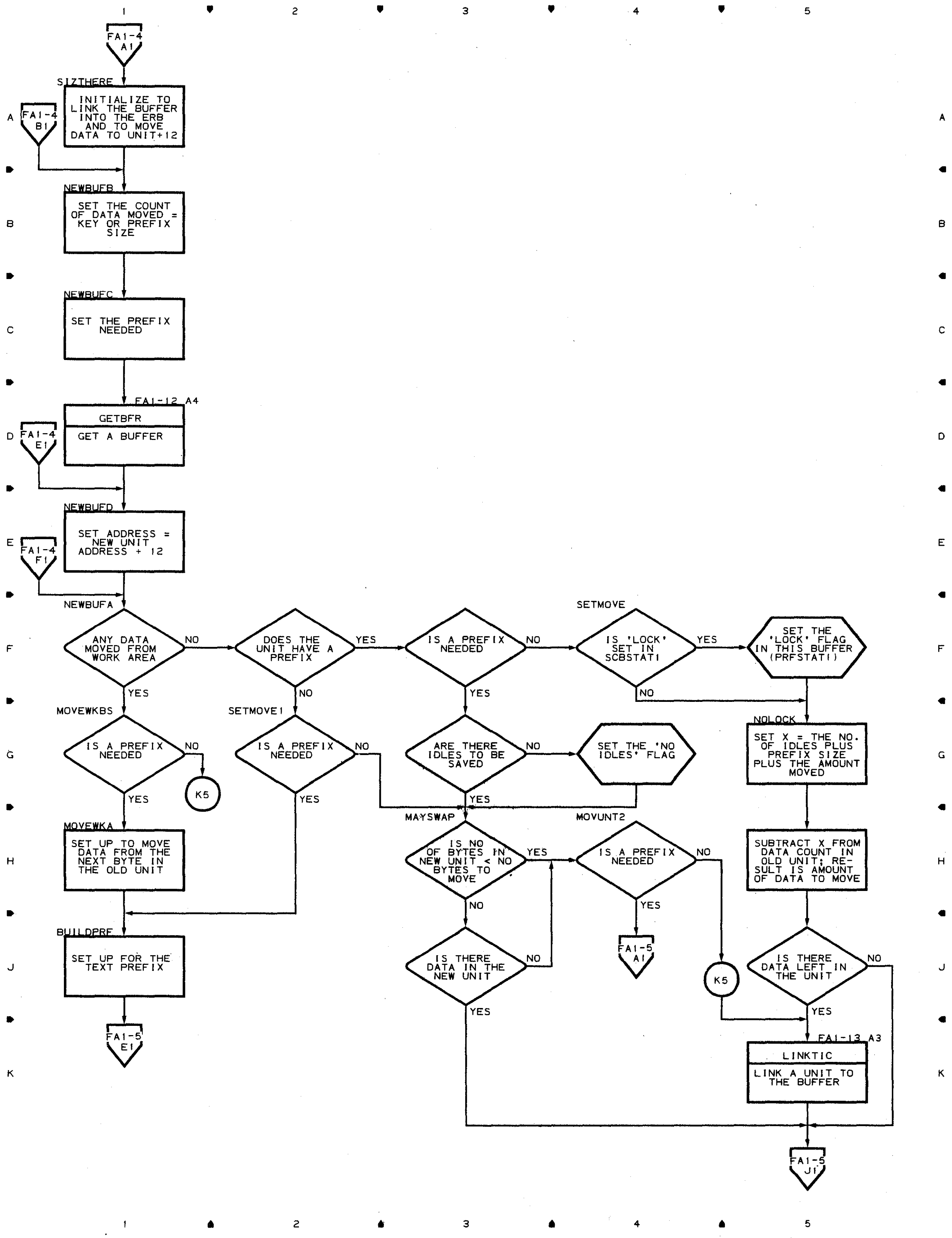
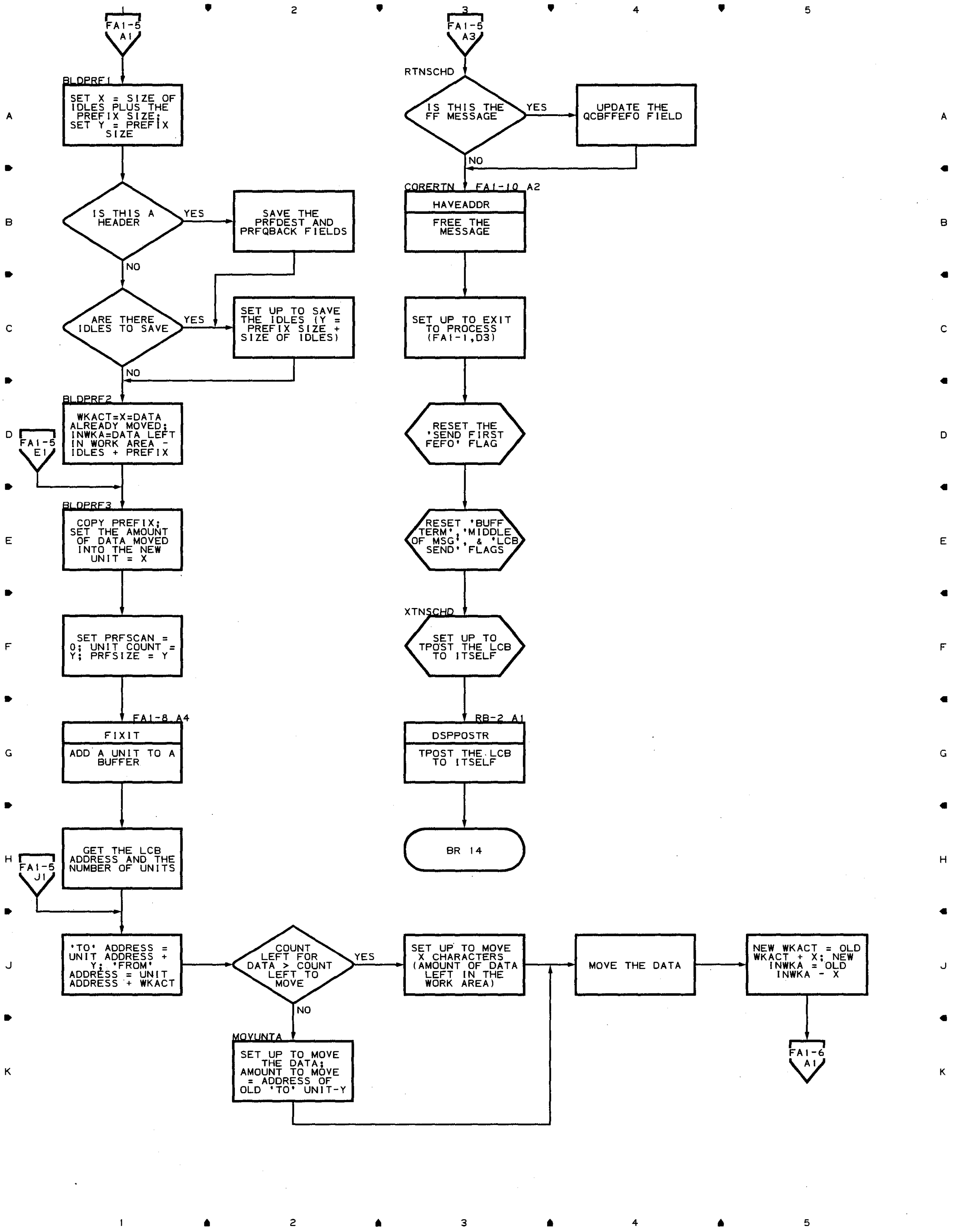

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
FA1-4 A1
SIZTHERE
INITIALIZE TO LINK THE BUFFER INTO THE ERB AND TO MOVE DATA TO UNIT+12
FA1-4 B1
NEWBUFB
SET THE COUNT OF DATA MOVED = KEY OR PREFIX SIZE
NEWBUFC
SET THE PREFIX NEEDED
FA1-12 A4
GETBFR
GET A BUFFER
FA1-4 E1
NEWBUFD
SET ADDRESS = NEW UNIT ADDRESS + 12
FA1-4 F1
NEWBUFA
ANY DATA MOVED FROM WORK AREA
NO
YES
DOES THE UNIT HAVE A PREFIX
YES
NO
IS A PREFIX NEEDED
NO
YES
SETMOVE
IS 'LOCK' SET IN SCBSTAT1
YES
NO
SET THE 'LOCK' FLAG IN THIS BUFFER (PRFSTAT1)
MOVEWKBS
IS A PREFIX NEEDED
NO
YES
K5
SETMOVE1
IS A PREFIX NEEDED
NO
YES
ARE THERE IDLES TO BE SAVED
NO
YES
SET THE 'NO IDLES' FLAG
NOLOCK
SET X = THE NO. OF IDLES PLUS PREFIX SIZE PLUS THE AMOUNT MOVED
MOVEWKA
SET UP TO MOVE DATA FROM THE NEXT BYTE IN THE OLD UNIT
MAYSWAP
IS NO OF BYTES IN NEW UNIT < NO BYTES TO MOVE
YES
NO
MOVUNT2
IS A PREFIX NEEDED
NO
YES
SUBTRACT X FROM DATA COUNT IN OLD UNIT; RESULT IS AMOUNT OF DATA TO MOVE
BUILDPRF
SET UP FOR THE TEXT PREFIX
IS THERE DATA IN THE NEW UNIT
NO
YES
FA1-5 A1
K5
IS THERE DATA LEFT IN THE UNIT
NO
YES
FA1-5 E1
FA1-13 A3
LINKTIC
LINK A UNIT TO THE BUFFER
FA1-5 J1

Chart FA1-5 CPB INITIALIZATION - MAIN STORAGE ONLY QUEUING


1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
FA1-5 A1
BLDPRF1
SET X = SIZE OF IDLES PLUS THE PREFIX SIZE; SET Y = PREFIX SIZE
IS THIS A HEADER
YES
SAVE THE PRFDEST AND PRFQBACK FIELDS
NO
ARE THERE IDLES TO SAVE
YES
SET UP TO SAVE THE IDLES (Y = PREFIX SIZE + SIZE OF IDLES)
NO
BLDPRF2
WKACT=X=DATA ALREADY MOVED; INWKA=DATA LEFT IN WORK AREA - IDLES + PREFIX
FA1-5 E1
BLDPRF3
COPY PREFIX; SET THE AMOUNT OF DATA MOVED INTO THE NEW UNIT = X
SET PRFSCAN = 0; UNIT COUNT = Y; PRFSIZE = Y
FA1-8 A4
FIXIT
ADD A UNIT TO A BUFFER
GET THE LCB ADDRESS AND THE NUMBER OF UNITS
FA1-5 J1
'TO' ADDRESS = UNIT ADDRESS + Y; 'FROM' ADDRESS = UNIT ADDRESS + WKACT
COUNT LEFT FOR DATA > COUNT LEFT TO MOVE
YES
NO
MOVUNTA
SET UP TO MOVE THE DATA; AMOUNT TO MOVE = ADDRESS OF OLD 'TO' UNIT-Y
FA1-5 A3
RTNSCHD
IS THIS THE FF MESSAGE
YES
UPDATE THE QCBFFEFO FIELD
NO
CORERTN
FA1-10 A2
HAVEADDR
FREE THE MESSAGE
SET UP TO EXIT TO PROCESS (FA1-1,D3)
RESET THE 'SEND FIRST FEFO' FLAG
RESET 'BUFF TERM', 'MIDDLE OF MSG', & 'LCB SEND' FLAGS
XTNSCHD
SET UP TO TPOST THE LCB TO ITSELF
RB-2 A1
DSPPOSTR
TPOST THE LCB TO ITSELF
BR 14
SET UP TO MOVE X CHARACTERS (AMOUNT OF DATA LEFT IN THE WORK AREA)
MOVE THE DATA
NEW WKACT = OLD WKACT + X; NEW INWKA = OLD INWKA - X
FA1-6 A1

CPB INITIALIZATION - MAIN STORAGE ONLY QUEUING

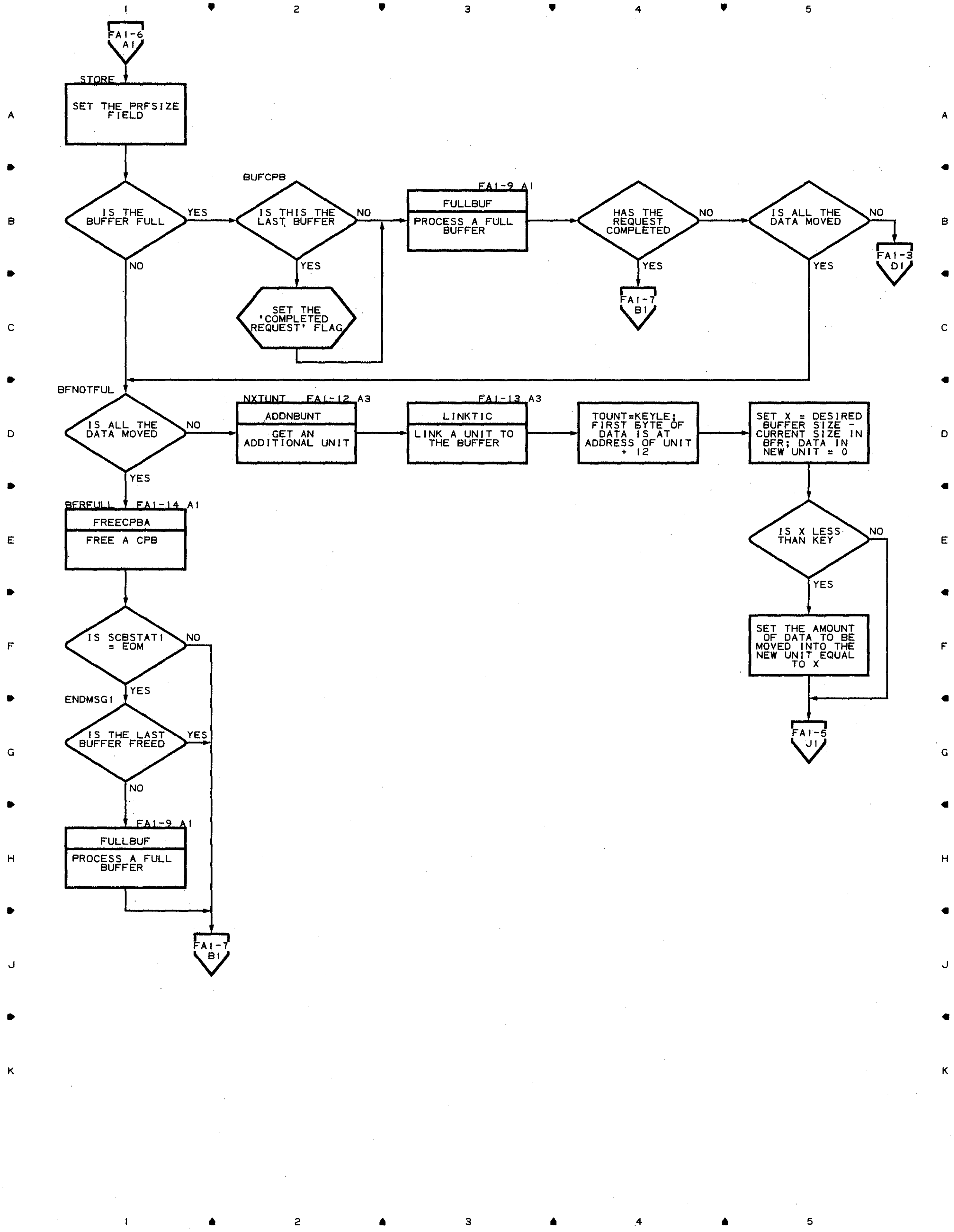


Chart FA1-7 CPB INITIALIZATION - MAIN STORAGE ONLY QUEUING

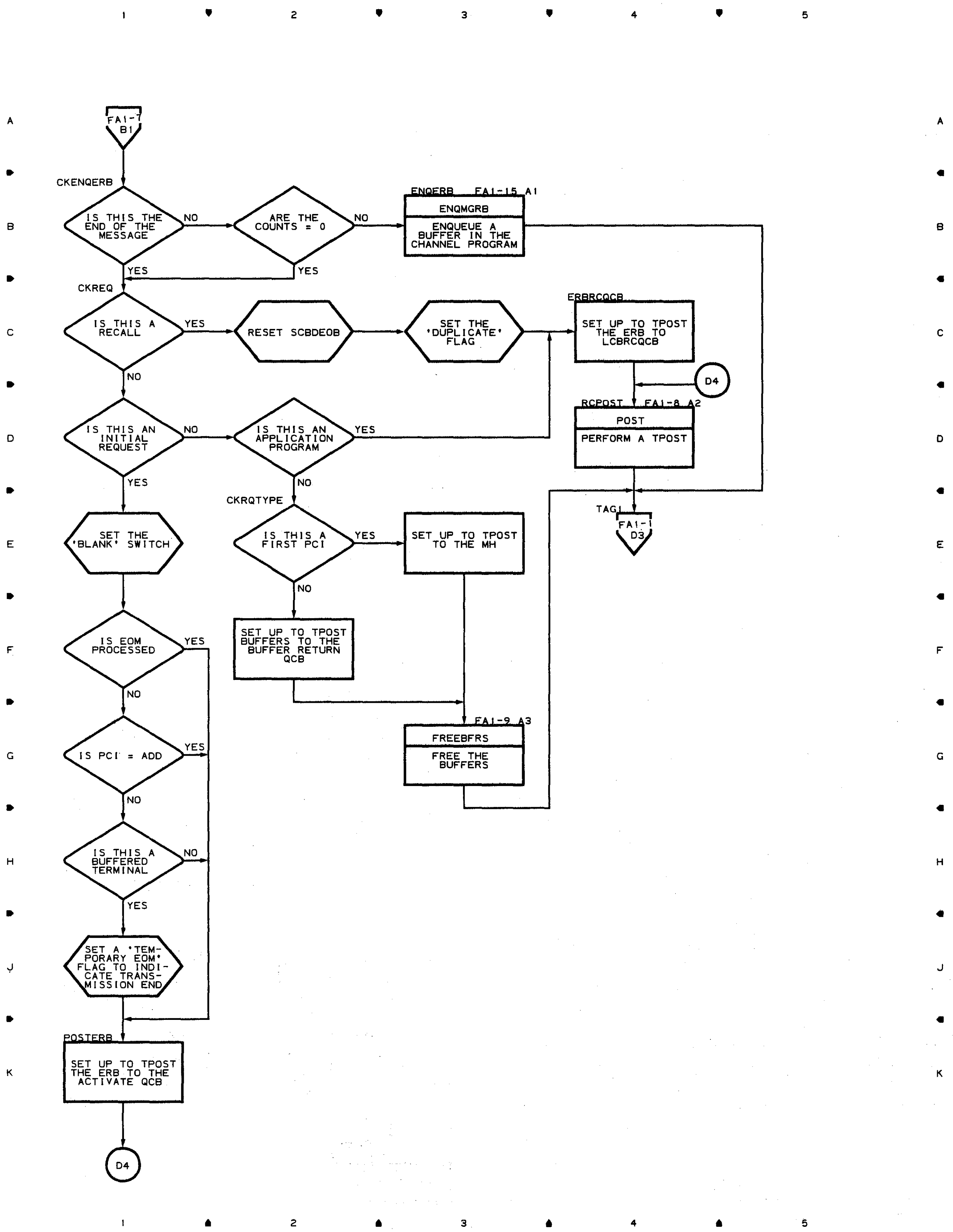

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
FA1-7 B1
CKENQERB
IS THIS THE END OF THE MESSAGE
NO
ARE THE COUNTS = 0
NO
ENQERB FA1-15 A1
ENQMGRB
ENQUEUE A BUFFER IN THE CHANNEL PROGRAM
YES
YES
CKREQ
IS THIS A RECALL
YES
RESET SCBDEOB
SET THE 'DUPLICATE' FLAG
ERBRCQCB
SET UP TO TPOST THE ERB TO LCBRCQCB
D4
NO
RCPOST FA1-8 A2
POST
PERFORM A TPOST
IS THIS AN INITIAL REQUEST
NO
IS THIS AN APPLICATION PROGRAM
YES
YES
NO
CKRQTYPE
TAG1
FA1-1 D3
SET THE 'BLANK' SWITCH
IS THIS A FIRST PCI
YES
SET UP TO TPOST TO THE MH
NO
IS EOM PROCESSED
YES
SET UP TO TPOST BUFFERS TO THE BUFFER RETURN QCB
NO
FA1-9 A3
FREEBFRS
FREE THE BUFFERS
IS PCI = ADD
YES
NO
IS THIS A BUFFERED TERMINAL
NO
YES
SET A 'TEMPORARY EOM' FLAG TO INDICATE TRANSMISSION END
POSTERB
SET UP TO TPOST THE ERB TO THE ACTIVATE QCB
D4

Chart FA1-8 CPB INITIALIZATION - MAIN STORAGE ONLY QUEUING

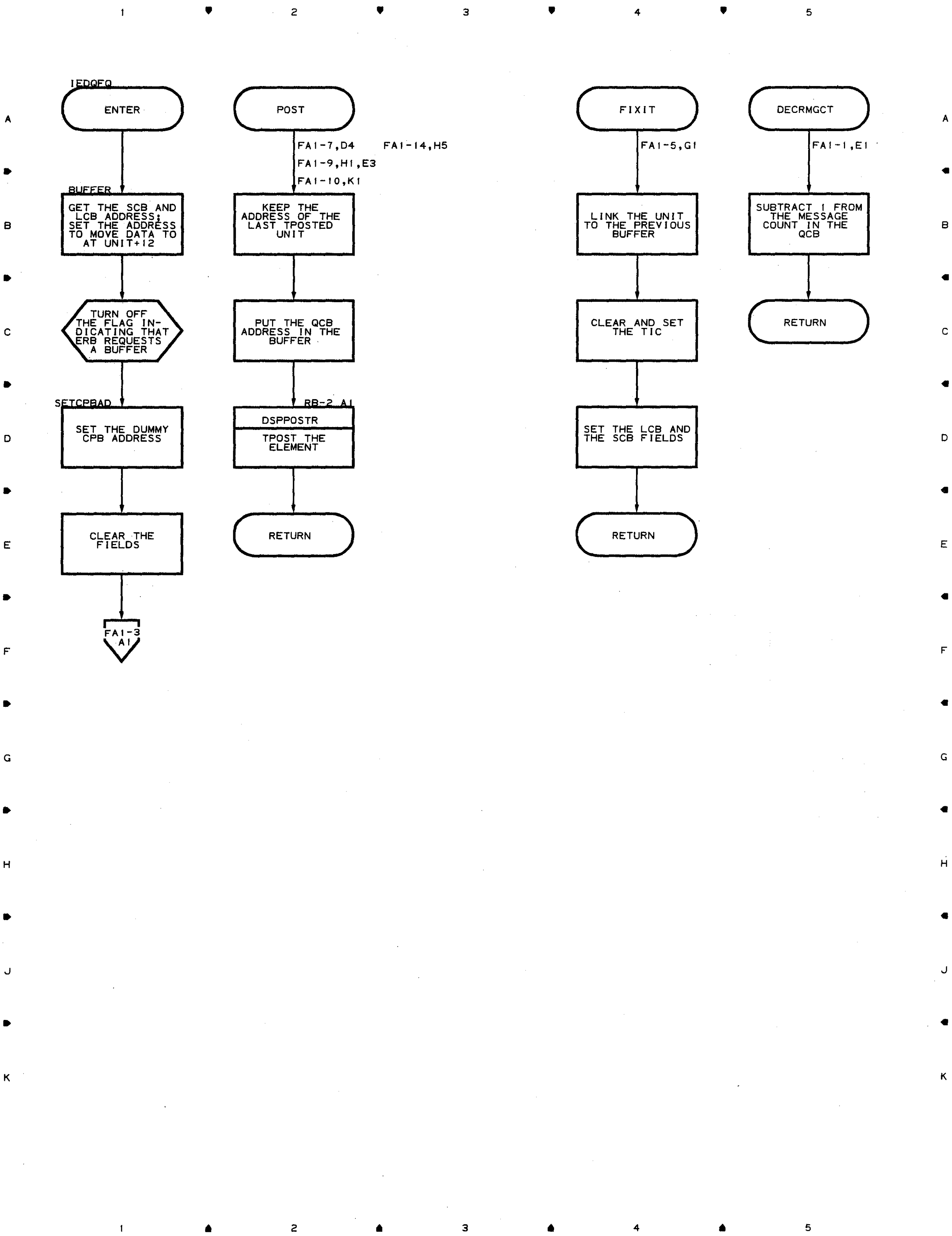
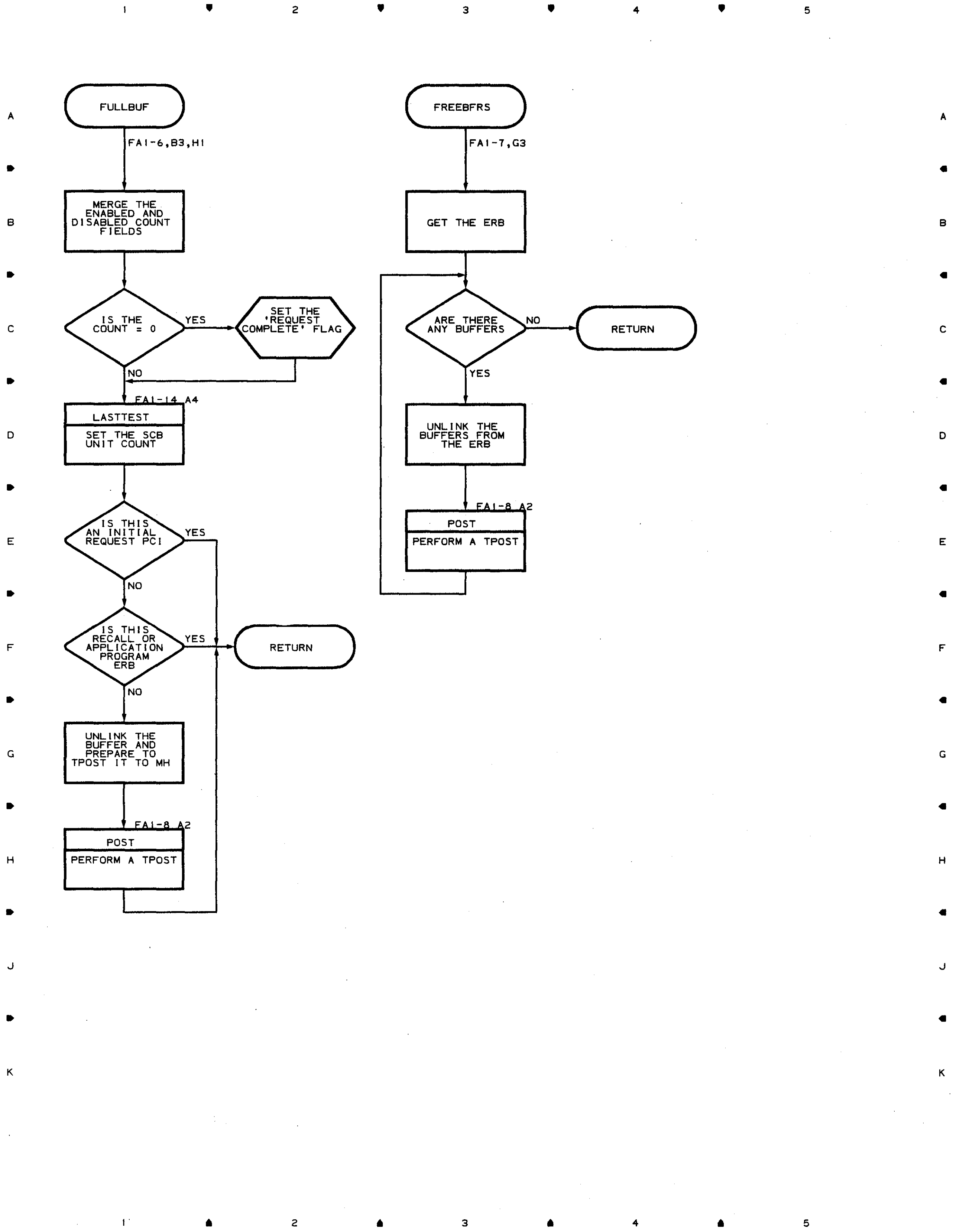

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQFQ
ENTER
BUFFER
GET THE SCB AND LCB ADDRESS; SET THE ADDRESS TO MOVE DATA TO AT UNIT+12
TURN OFF THE FLAG INDICATING THAT ERB REQUESTS A BUFFER
SETCPBAD
SET THE DUMMY CPB ADDRESS
CLEAR THE FIELDS
FA1-3 A1
POST
FA1-7,D4
FA1-14,H5
FA1-9,H1,E3
FA1-10,K1
KEEP THE ADDRESS OF THE LAST TPOSTED UNIT
PUT THE QCB ADDRESS IN THE BUFFER
RB-2 A1
DSPPOSTR
TPOST THE ELEMENT
RETURN
FIXIT
FA1-5,G1
LINK THE UNIT TO THE PREVIOUS BUFFER
CLEAR AND SET THE TIC
SET THE LCB AND THE SCB FIELDS
RETURN
DECRMGCT
FA1-1,E1
SUBTRACT 1 FROM THE MESSAGE COUNT IN THE QCB
RETURN

Chart FA1-9 CPB INITIALIZATION - MAIN STORAGE ONLY QUEUING



FULLBUF
FA1-6,B3,H1
MERGE THE ENABLED AND DISABLED COUNT FIELDS
IS THE COUNT = 0
YES
SET THE 'REQUEST COMPLETE' FLAG
NO
FA1-14 A4
LASTTEST
SET THE SCB UNIT COUNT
IS THIS AN INITIAL REQUEST PCI
YES
NO
IS THIS RECALL OR APPLICATION PROGRAM ERB
YES
RETURN
NO
UNLINK THE BUFFER AND PREPARE TO TPOST IT TO MH
FA1-8 A2
POST
PERFORM A TPOST
FREEBFRS
FA1-7,G3
GET THE ERB
ARE THERE ANY BUFFERS
NO
RETURN
YES
UNLINK THE BUFFERS FROM THE ERB
FA1-8 A2
POST
PERFORM A TPOST

Chart FA1-10 CPB INITIALIZATION - MAIN STORAGE ONLY QUEUING

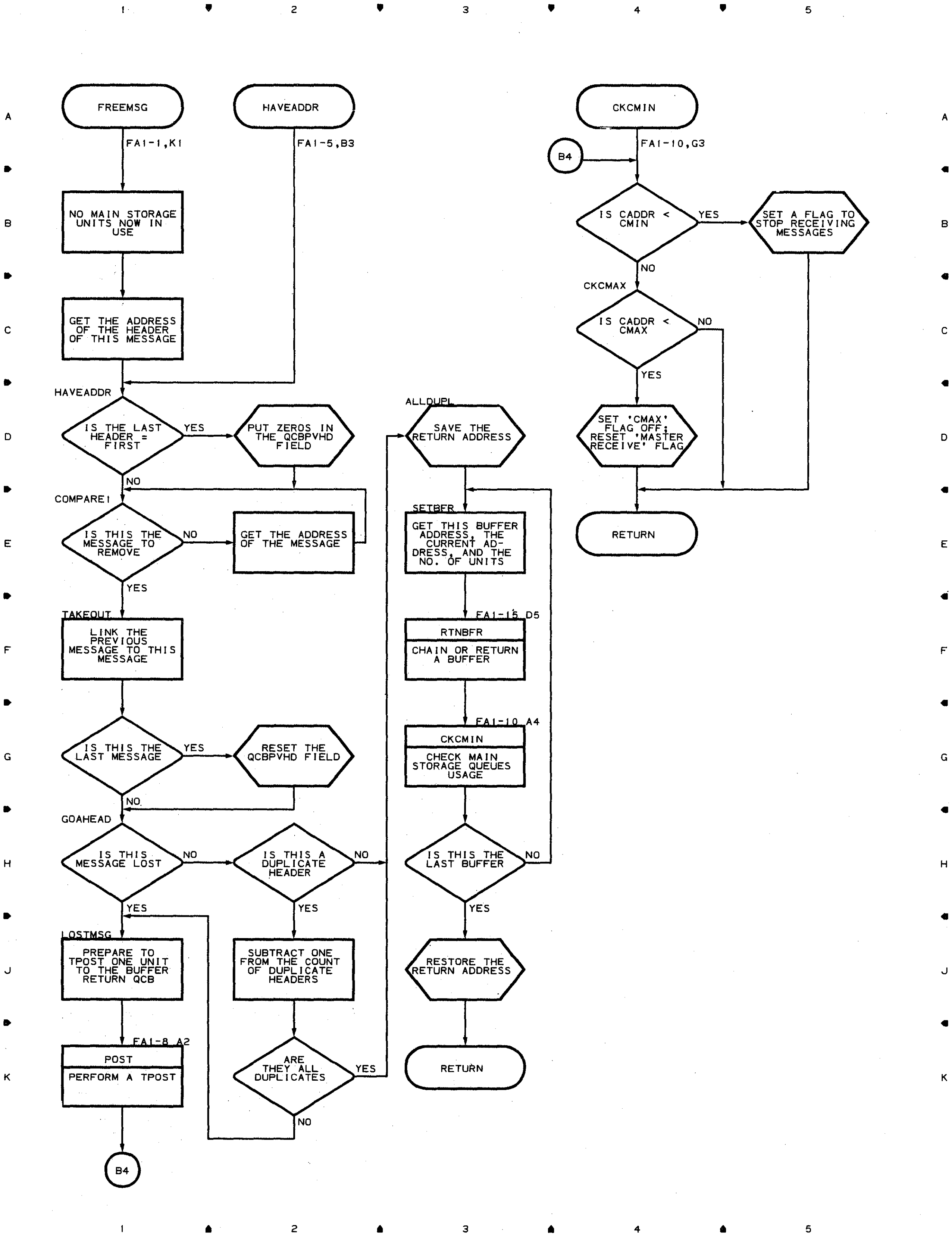

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
FREEMSG
FA1-1,K1
NO MAIN STORAGE UNITS NOW IN USE
GET THE ADDRESS OF THE HEADER OF THIS MESSAGE
HAVEADDR
FA1-5,B3
HAVEADDR
IS THE LAST HEADER = FIRST
YES
PUT ZEROS IN THE QCBPVHD FIELD
NO
COMPARE1
IS THIS THE MESSAGE TO REMOVE
NO
GET THE ADDRESS OF THE MESSAGE
YES
TAKEOUT
LINK THE PREVIOUS MESSAGE TO THIS MESSAGE
IS THIS THE LAST MESSAGE
YES
RESET THE QCBPVHD FIELD
NO
GOAHEAD
IS THIS MESSAGE LOST
NO
IS THIS A DUPLICATE HEADER
NO
YES
YES
LOSTMSG
PREPARE TO TPOST ONE UNIT TO THE BUFFER RETURN QCB
SUBTRACT ONE FROM THE COUNT OF DUPLICATE HEADERS
FA1-8 A2
POST
PERFORM A TPOST
ARE THEY ALL DUPLICATES
YES
NO
B4
ALLDUPL
SAVE THE RETURN ADDRESS
SETBFR
GET THIS BUFFER ADDRESS, THE CURRENT AD- DRESS, AND THE NO. OF UNITS
FA1-15 D5
RTNBFR
CHAIN OR RETURN A BUFFER
FA1-10 A4
CKCMIN
CHECK MAIN STORAGE QUEUES USAGE
IS THIS THE LAST BUFFER
NO
YES
RESTORE THE RETURN ADDRESS
RETURN
CKCMIN
B4
FA1-10,G3
IS CADDR < CMIN
YES
SET A FLAG TO STOP RECEIVING MESSAGES
NO
CKCMAX
IS CADDR < CMAX
NO
YES
SET 'CMAX' FLAG OFF; RESET 'MASTER RECEIVE' FLAG
RETURN

Chart FA1-11 CPB INTIALIZATION - MAIN STORAGE ONLY QUEUING

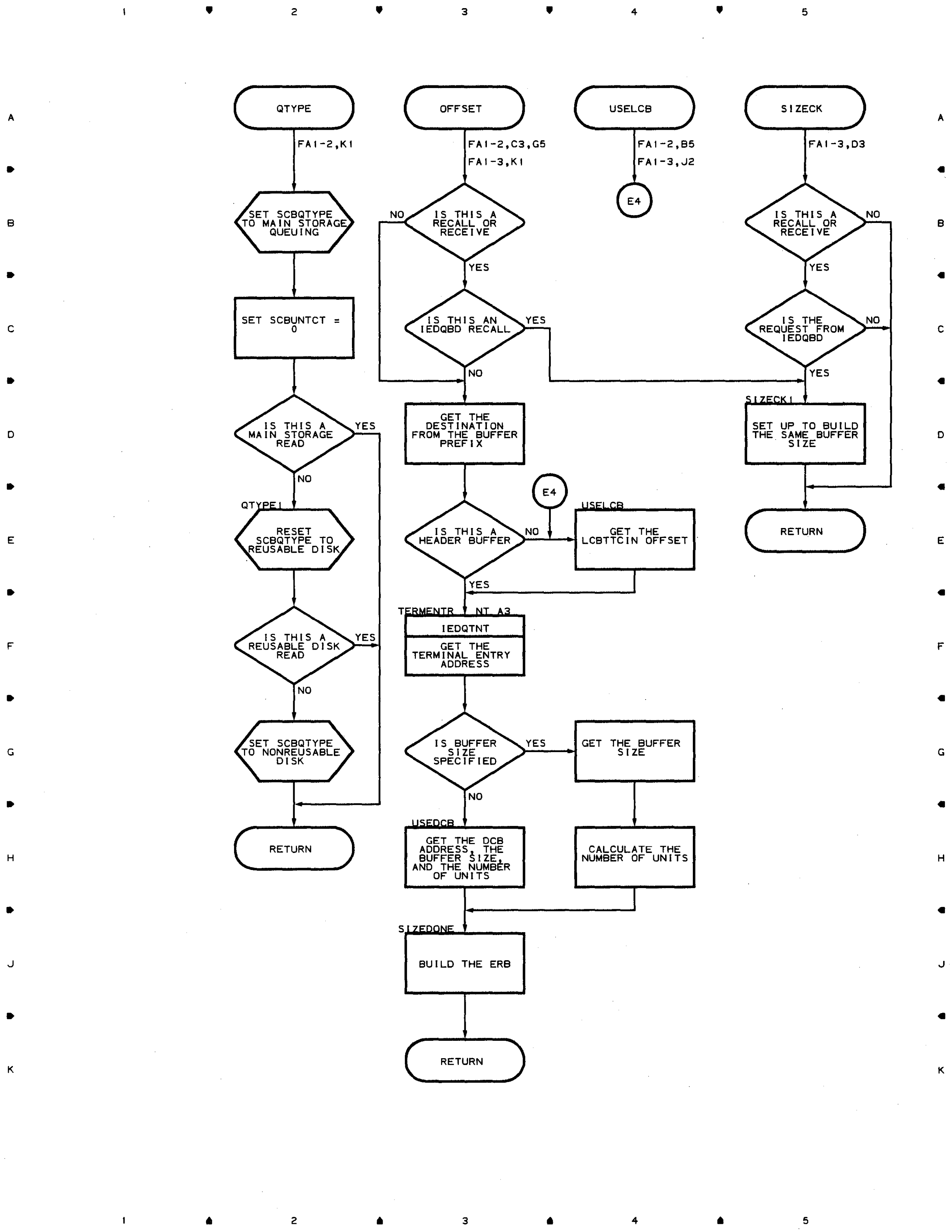

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
QTYPE
FA1-2,K1
SET SCBQTYPE TO MAIN STORAGE QUEUING
SET SCBUNTCT = 0
IS THIS A MAIN STORAGE READ
YES
NO
QTYPE1
RESET SCBQTYPE TO REUSABLE DISK
IS THIS A REUSABLE DISK READ
YES
NO
SET SCBQTYPE TO NONREUSABLE DISK
RETURN
OFFSET
FA1-2,C3,G5
FA1-3,K1
IS THIS A RECALL OR RECEIVE
NO
YES
IS THIS AN IEDQBD RECALL
YES
NO
GET THE DESTINATION FROM THE BUFFER PREFIX
IS THIS A HEADER BUFFER
NO
YES
TERMENTR
NT A3
IEDQTNT
GET THE TERMINAL ENTRY ADDRESS
IS BUFFER SIZE SPECIFIED
YES
NO
USEDCB
GET THE DCB ADDRESS, THE BUFFER SIZE, AND THE NUMBER OF UNITS
SIZEDONE
BUILD THE ERB
RETURN
USELCB
FA1-2,B5
FA1-3,J2
E4
E4
USELCB
GET THE LCBTTCIN OFFSET
GET THE BUFFER SIZE
CALCULATE THE NUMBER OF UNITS
SIZECK
FA1-3,D3
IS THIS A RECALL OR RECEIVE
NO
YES
IS THE REQUEST FROM IEDQBD
NO
YES
SIZECK1
SET UP TO BUILD THE SAME BUFFER SIZE
RETURN

Chart FA1-12 CPB INITIALIZATION - MAIN STORAGE ONLY QUEUING

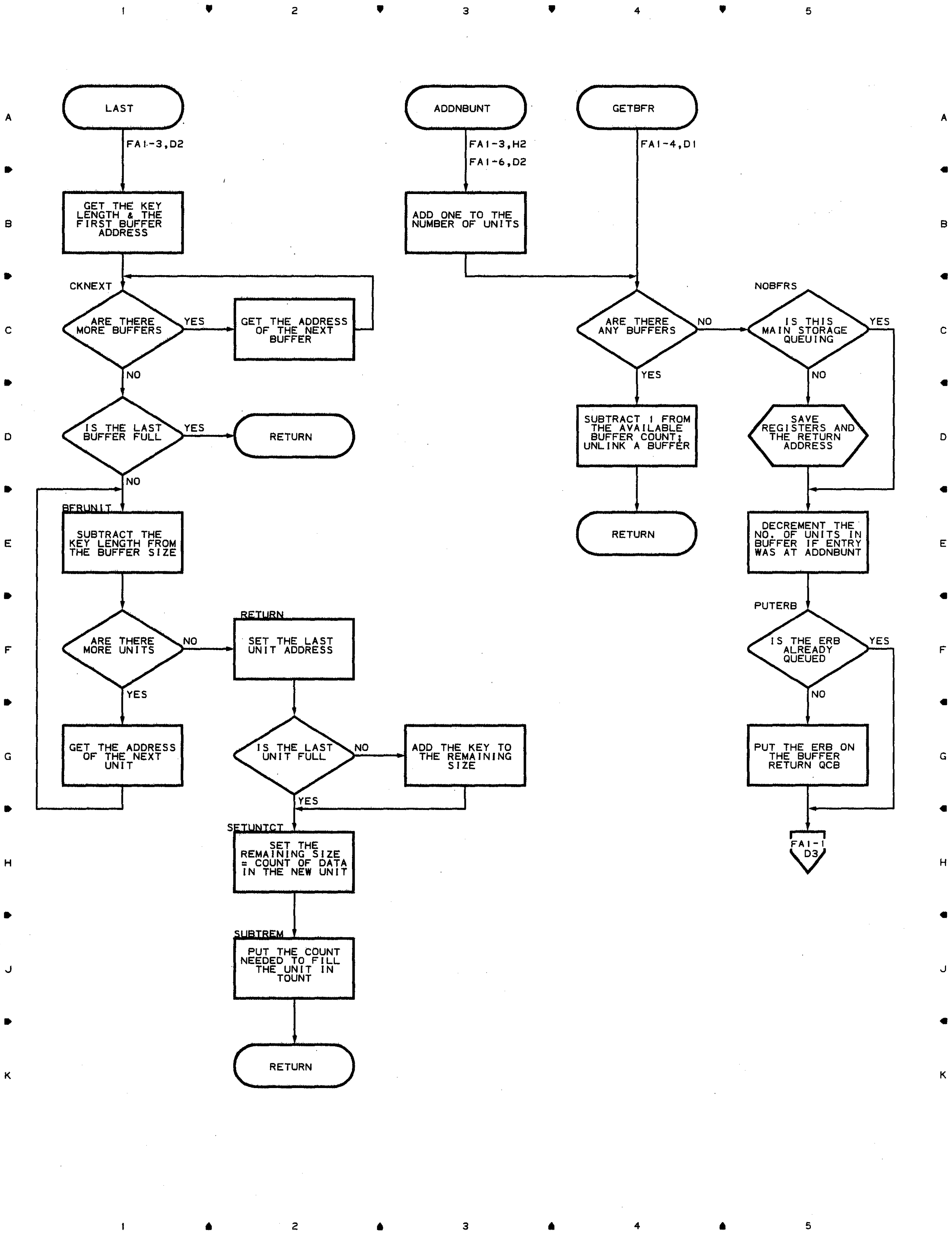
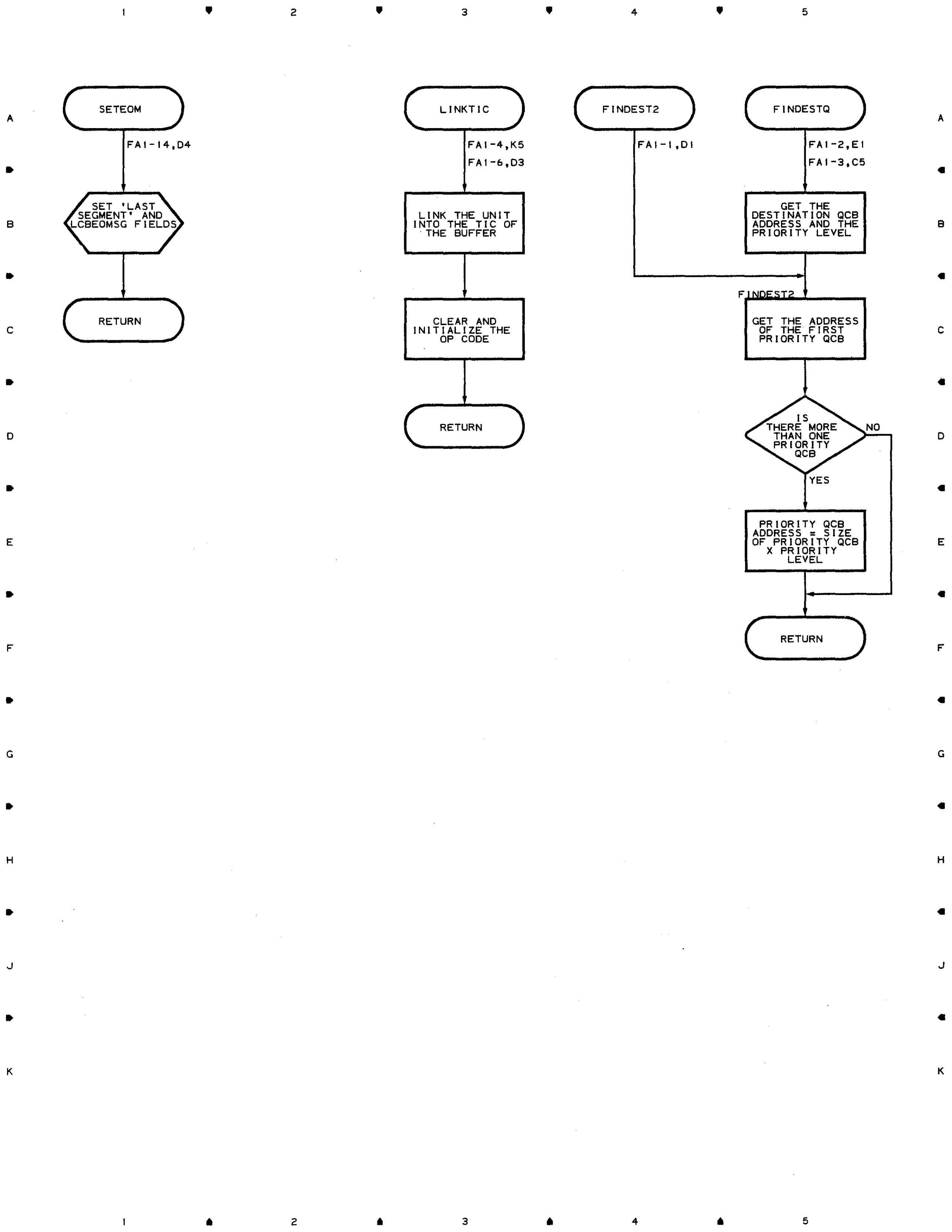

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
LAST
FA1-3,D2
GET THE KEY LENGTH & THE FIRST BUFFER ADDRESS
CKNEXT
ARE THERE MORE BUFFERS
YES
GET THE ADDRESS OF THE NEXT BUFFER
NO
IS THE LAST BUFFER FULL
YES
RETURN
NO
BFRUNIT
SUBTRACT THE KEY LENGTH FROM THE BUFFER SIZE
ARE THERE MORE UNITS
NO
RETURN
SET THE LAST UNIT ADDRESS
YES
GET THE ADDRESS OF THE NEXT UNIT
IS THE LAST UNIT FULL
NO
ADD THE KEY TO THE REMAINING SIZE
YES
SETUNTCT
SET THE REMAINING SIZE = COUNT OF DATA IN THE NEW UNIT
SUBTREM
PUT THE COUNT NEEDED TO FILL THE UNIT IN TOUNT
RETURN
ADDNBUNT
FA1-3,H2
FA1-6,D2
ADD ONE TO THE NUMBER OF UNITS
GETBFR
FA1-4,D1
ARE THERE ANY BUFFERS
NO
YES
SUBTRACT 1 FROM THE AVAILABLE BUFFER COUNT; UNLINK A BUFFER
RETURN
NOBFRS
IS THIS MAIN STORAGE QUEUING
YES
NO
SAVE REGISTERS AND THE RETURN ADDRESS
DECREMENT THE NO. OF UNITS IN BUFFER IF ENTRY WAS AT ADDNBUNT
PUTERB
IS THE ERB ALREADY QUEUED
YES
NO
PUT THE ERB ON THE BUFFER RETURN QCB
FA1-1 D3

Chart FA1-13 CPB INITIALIZATION - MAIN STORAGE ONLY QUEUING



SETEOM
FA1-14,D4
SET 'LAST SEGMENT' AND LCBEOMSG FIELDS
RETURN
LINKTIC
FA1-4,K5
FA1-6,D3
LINK THE UNIT INTO THE TIC OF THE BUFFER
CLEAR AND INITIALIZE THE OP CODE
RETURN
FINDEST2
FA1-1,D1
FINDESTQ
FA1-2,E1
FA1-3,C5
GET THE DESTINATION QCB ADDRESS AND THE PRIORITY LEVEL
FINDEST2
GET THE ADDRESS OF THE FIRST PRIORITY QCB
IS THERE MORE THAN ONE PRIORITY QCB
NO
YES
PRIORITY QCB ADDRESS = SIZE OF PRIORITY QCB X PRIORITY LEVEL
RETURN

Chart FA1-14 CPB INITIALIZATION - MAIN STORAGE ONLY QUEUING

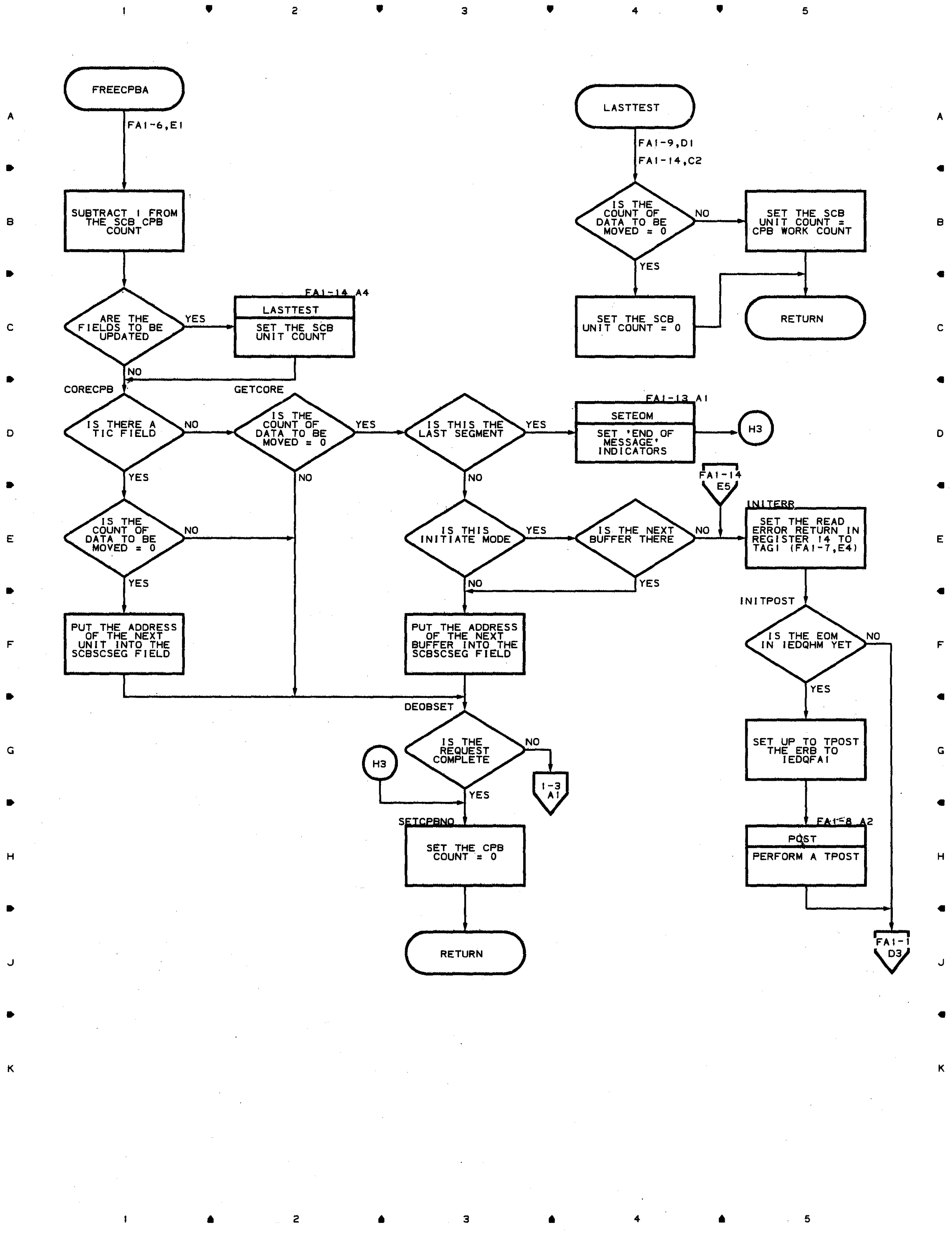
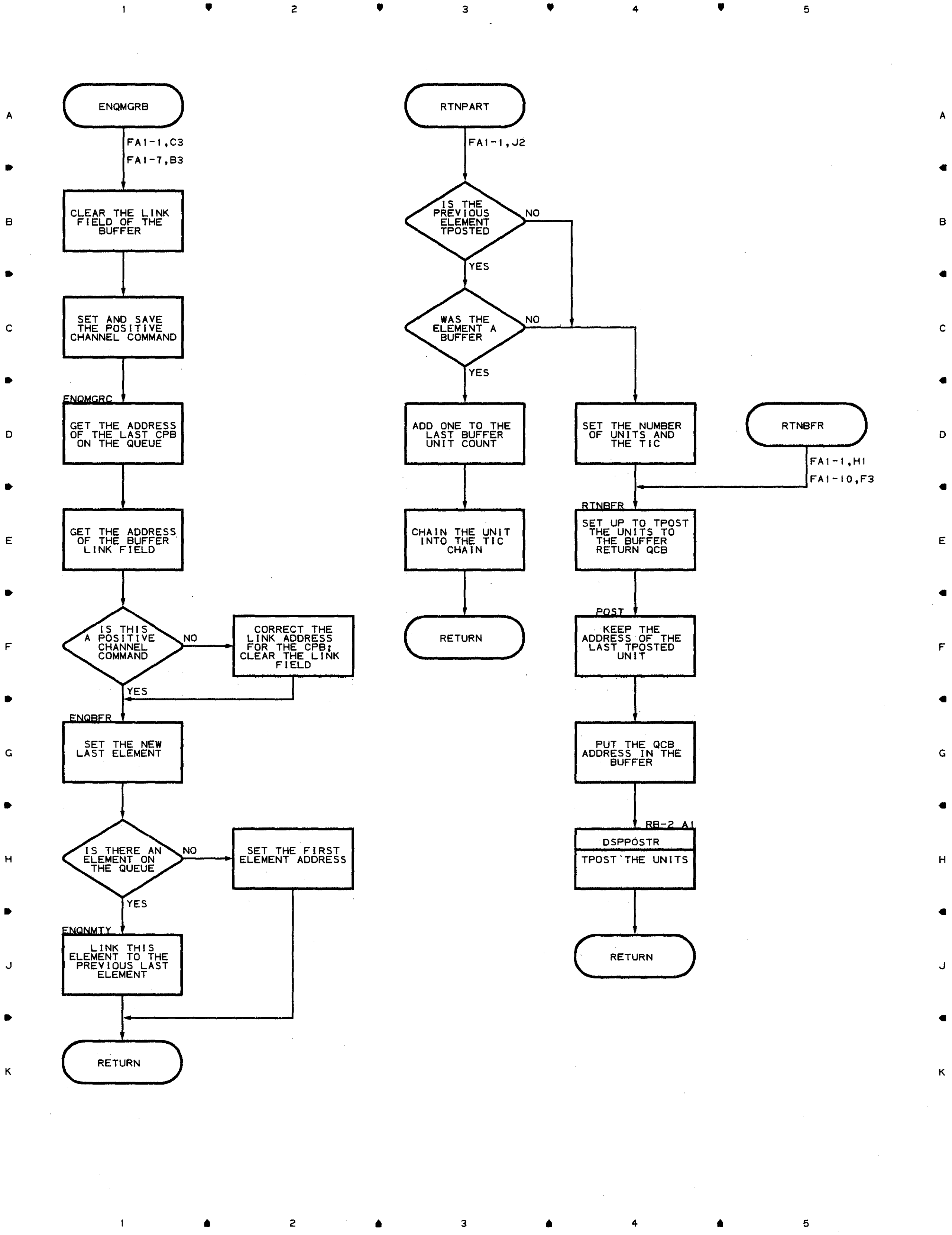
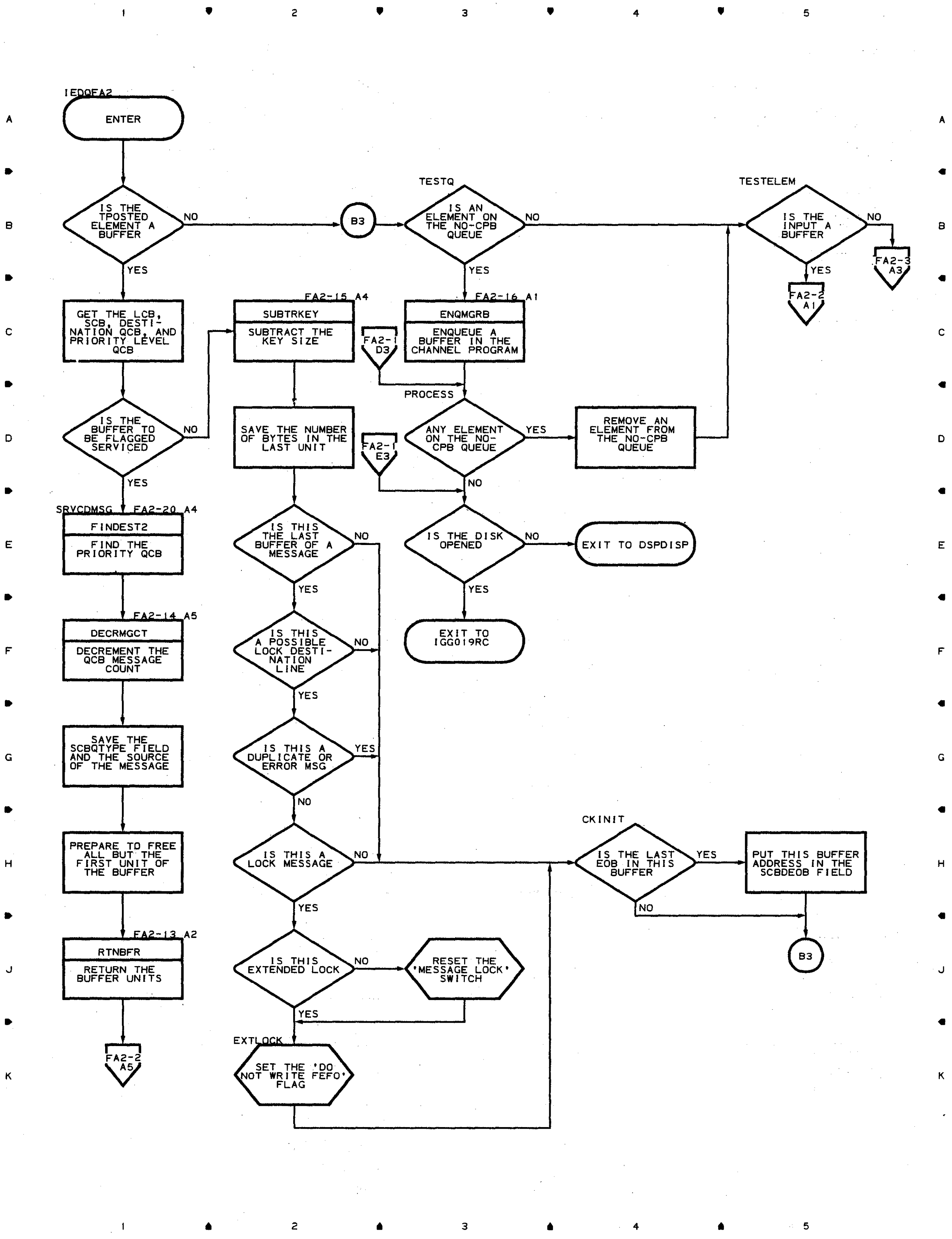

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
FREECPBA
FA1-6,E1
SUBTRACT 1 FROM THE SCB CPB COUNT
ARE THE FIELDS TO BE UPDATED
YES
FA1-14 A4
LASTTEST
SET THE SCB UNIT COUNT
NO
CORECPB
IS THERE A TIC FIELD
NO
GETCORE
IS THE COUNT OF DATA TO BE MOVED = 0
YES
IS THIS THE LAST SEGMENT
YES
FA1-13 A1
SETEOM
SET 'END OF MESSAGE' INDICATORS
H3
YES
NO
NO
IS THE COUNT OF DATA TO BE MOVED = 0
NO
IS THIS INITIATE MODE
YES
IS THE NEXT BUFFER THERE
NO
FA1-14 E5
INITERR
SET THE READ ERROR RETURN IN REGISTER 14 TO TAG1 (FA1-7,E4)
YES
NO
YES
PUT THE ADDRESS OF THE NEXT UNIT INTO THE SCBSCSEG FIELD
PUT THE ADDRESS OF THE NEXT BUFFER INTO THE SCBSCSEG FIELD
INITPOST
IS THE EOM IN IEDQHM YET
NO
YES
DEOBSET
IS THE REQUEST COMPLETE
NO
H3
1-3 A1
YES
SET UP TO TPOST THE ERB TO IEDQFA1
SETCPBNO
SET THE CPB COUNT = 0
FA1-8 A2
POST
PERFORM A TPOST
RETURN
FA1-1 D3
LASTTEST
FA1-9,D1
FA1-14,C2
IS THE COUNT OF DATA TO BE MOVED = 0
NO
SET THE SCB UNIT COUNT = CPB WORK COUNT
YES
SET THE SCB UNIT COUNT = 0
RETURN

Chart FA1-15 CPB INITIALIZATION - MAIN STORAGE ONLY QUEUING



ENQMGRB
FA1-1,C3
FA1-7,B3
CLEAR THE LINK FIELD OF THE BUFFER
SET AND SAVE THE POSITIVE CHANNEL COMMAND
ENQMGRC
GET THE ADDRESS OF THE LAST CPB ON THE QUEUE
GET THE ADDRESS OF THE BUFFER LINK FIELD
IS THIS A POSITIVE CHANNEL COMMAND
NO
CORRECT THE LINK ADDRESS FOR THE CPB; CLEAR THE LINK FIELD
YES
ENQBFR
SET THE NEW LAST ELEMENT
IS THERE AN ELEMENT ON THE QUEUE
NO
SET THE FIRST ELEMENT ADDRESS
YES
ENQNMTY
LINK THIS ELEMENT TO THE PREVIOUS LAST ELEMENT
RETURN
RTNPART
FA1-1,J2
IS THE PREVIOUS ELEMENT TPOSTED
NO
YES
WAS THE ELEMENT A BUFFER
NO
YES
ADD ONE TO THE LAST BUFFER UNIT COUNT
CHAIN THE UNIT INTO THE TIC CHAIN
RETURN
SET THE NUMBER OF UNITS AND THE TIC
RTNBFR
FA1-1,H1
FA1-10,F3
RTNBFR
SET UP TO TPOST THE UNITS TO THE BUFFER RETURN QCB
POST
KEEP THE ADDRESS OF THE LAST TPOSTED UNIT
PUT THE QCB ADDRESS IN THE BUFFER
RB-2 A1
DSPPOSTR
TPOST THE UNITS
RETURN
1 2 3 4 5
A B C D E F G H J K

Chart FA2-1 CPB INITIALIZATION - DISK ONLY QUEUING



IEDQFA2
ENTER
IS THE TPOSTED ELEMENT A BUFFER
NO
YES
B3
TESTQ
IS AN ELEMENT ON THE NO-CPB QUEUE
NO
YES
TESTELEM
IS THE INPUT A BUFFER
NO
YES
FA2-3 A3
FA2-2 A1
GET THE LCB, SCB, DESTINATION QCB, AND PRIORITY LEVEL QCB
FA2-15 A4
SUBTRKEY
SUBTRACT THE KEY SIZE
FA2-1 D3
FA2-16 A1
ENQMGRB
ENQUEUE A BUFFER IN THE CHANNEL PROGRAM
PROCESS
IS THE BUFFER TO BE FLAGGED SERVICED
NO
YES
SAVE THE NUMBER OF BYTES IN THE LAST UNIT
FA2-1 E3
ANY ELEMENT ON THE NO-CPB QUEUE
YES
NO
REMOVE AN ELEMENT FROM THE NO-CPB QUEUE
SRVCDMSG
FA2-20 A4
FINDEST2
FIND THE PRIORITY QCB
IS THIS THE LAST BUFFER OF A MESSAGE
NO
YES
IS THE DISK OPENED
NO
YES
EXIT TO DSPDISP
FA2-14 A5
DECRMGCT
DECREMENT THE QCB MESSAGE COUNT
IS THIS A POSSIBLE LOCK DESTINATION LINE
NO
YES
EXIT TO IGG019RC
SAVE THE SCBQTYPE FIELD AND THE SOURCE OF THE MESSAGE
IS THIS A DUPLICATE OR ERROR MSG
YES
NO
CKINIT
PREPARE TO FREE ALL BUT THE FIRST UNIT OF THE BUFFER
IS THIS A LOCK MESSAGE
NO
YES
IS THE LAST EOB IN THIS BUFFER
YES
NO
PUT THIS BUFFER ADDRESS IN THE SCBDEOB FIELD
FA2-13 A2
RTNBFR
RETURN THE BUFFER UNITS
IS THIS EXTENDED LOCK
NO
YES
RESET THE 'MESSAGE LOCK' SWITCH
B3
EXTLOCK
FA2-2 A5
SET THE 'DO NOT WRITE FEFO' FLAG

Chart FA2-2 CPB INITIALIZATION - DISK ONLY QUEUING

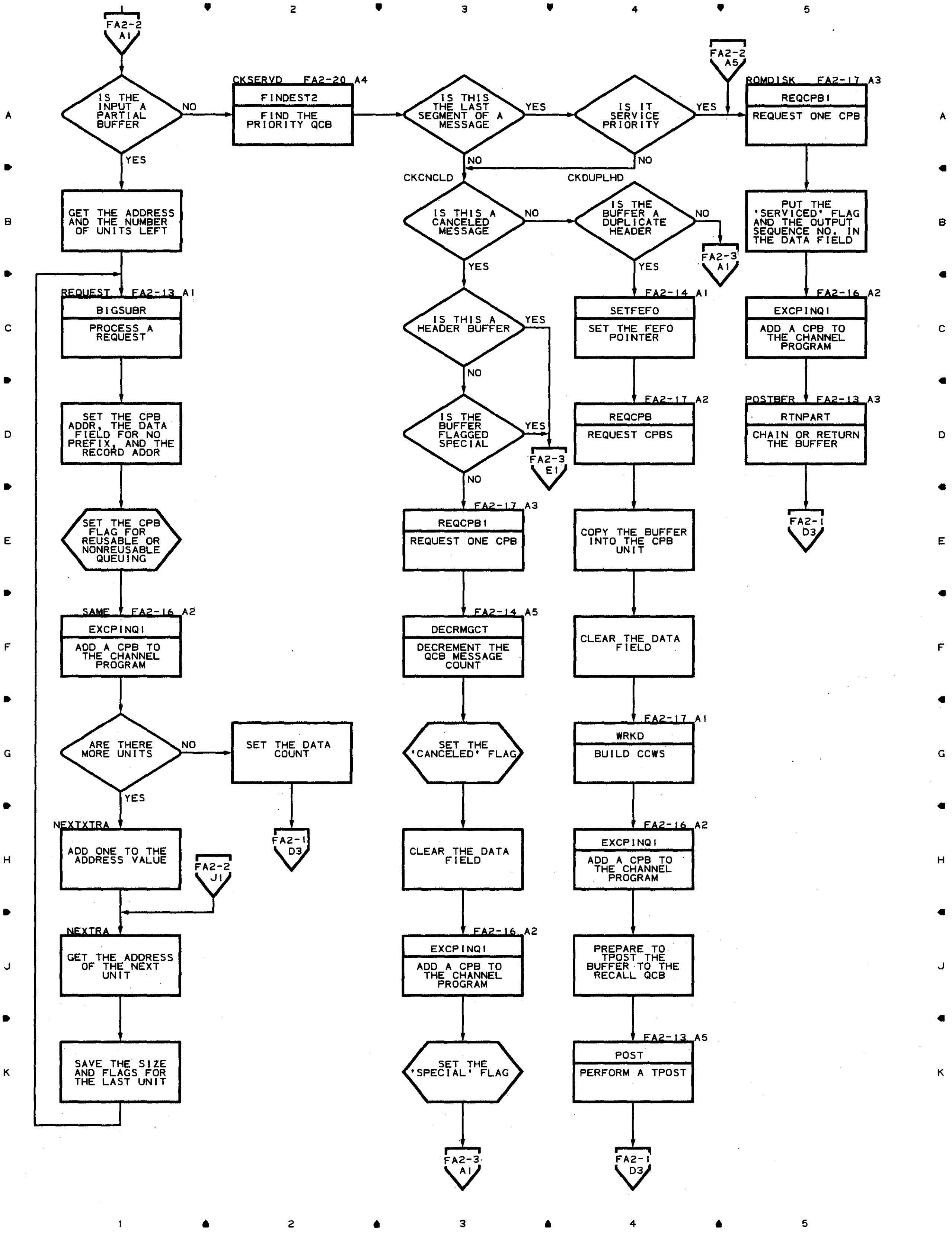

FA2-2 A1
IS THE INPUT A PARTIAL BUFFER
NO
YES
CKSERVD FA2-20 A4
FINDEST2
FIND THE PRIORITY QCB
IS THIS THE LAST SEGMENT OF A MESSAGE
YES
NO
IS IT SERVICE PRIORITY
YES
NO
FA2-2 A5
ROMDISK FA2-17 A3
REQCPB1
REQUEST ONE CPB
GET THE ADDRESS AND THE NUMBER OF UNITS LEFT
CKCNCLD
IS THIS A CANCELED MESSAGE
NO
YES
CKDUPLHD
IS THE BUFFER A DUPLICATE HEADER
NO
YES
FA2-3 A1
PUT THE 'SERVICED' FLAG AND THE OUTPUT SEQUENCE NO. IN THE DATA FIELD
REQUEST FA2-13 A1
BIGSUBR
PROCESS A REQUEST
IS THIS A HEADER BUFFER
YES
NO
FA2-14 A1
SETFEFO
SET THE FEFO POINTER
FA2-16 A2
EXCPINQ1
ADD A CPB TO THE CHANNEL PROGRAM
SET THE CPB ADDR, THE DATA FIELD FOR NO PREFIX, AND THE RECORD ADDR
IS THE BUFFER FLAGGED SPECIAL
YES
NO
FA2-3 E1
FA2-17 A2
REQCPB
REQUEST CPBS
POSTBFR FA2-13 A3
RTNPART
CHAIN OR RETURN THE BUFFER
SET THE CPB FLAG FOR REUSABLE OR NONREUSABLE QUEUING
FA2-17 A3
REQCPB1
REQUEST ONE CPB
COPY THE BUFFER INTO THE CPB UNIT
FA2-1 D3
SAME FA2-16 A2
EXCPINQ1
ADD A CPB TO THE CHANNEL PROGRAM
FA2-14 A5
DECRMGCT
DECREMENT THE QCB MESSAGE COUNT
CLEAR THE DATA FIELD
ARE THERE MORE UNITS
NO
YES
SET THE DATA COUNT
SET THE 'CANCELED' FLAG
FA2-17 A1
WRKD
BUILD CCWS
NEXTXTRA
ADD ONE TO THE ADDRESS VALUE
FA2-2 J1
FA2-1 D3
CLEAR THE DATA FIELD
FA2-16 A2
EXCPINQ1
ADD A CPB TO THE CHANNEL PROGRAM
NEXTRA
GET THE ADDRESS OF THE NEXT UNIT
FA2-16 A2
EXCPINQ1
ADD A CPB TO THE CHANNEL PROGRAM
PREPARE TO TPOST THE BUFFER TO THE RECALL QCB
SAVE THE SIZE AND FLAGS FOR THE LAST UNIT
SET THE 'SPECIAL' FLAG
FA2-13 A5
POST
PERFORM A TPOST
FA2-3 A1
FA2-1 D3
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart FA2-3 CPB INITIALIZATION - DISK ONLY QUEUING

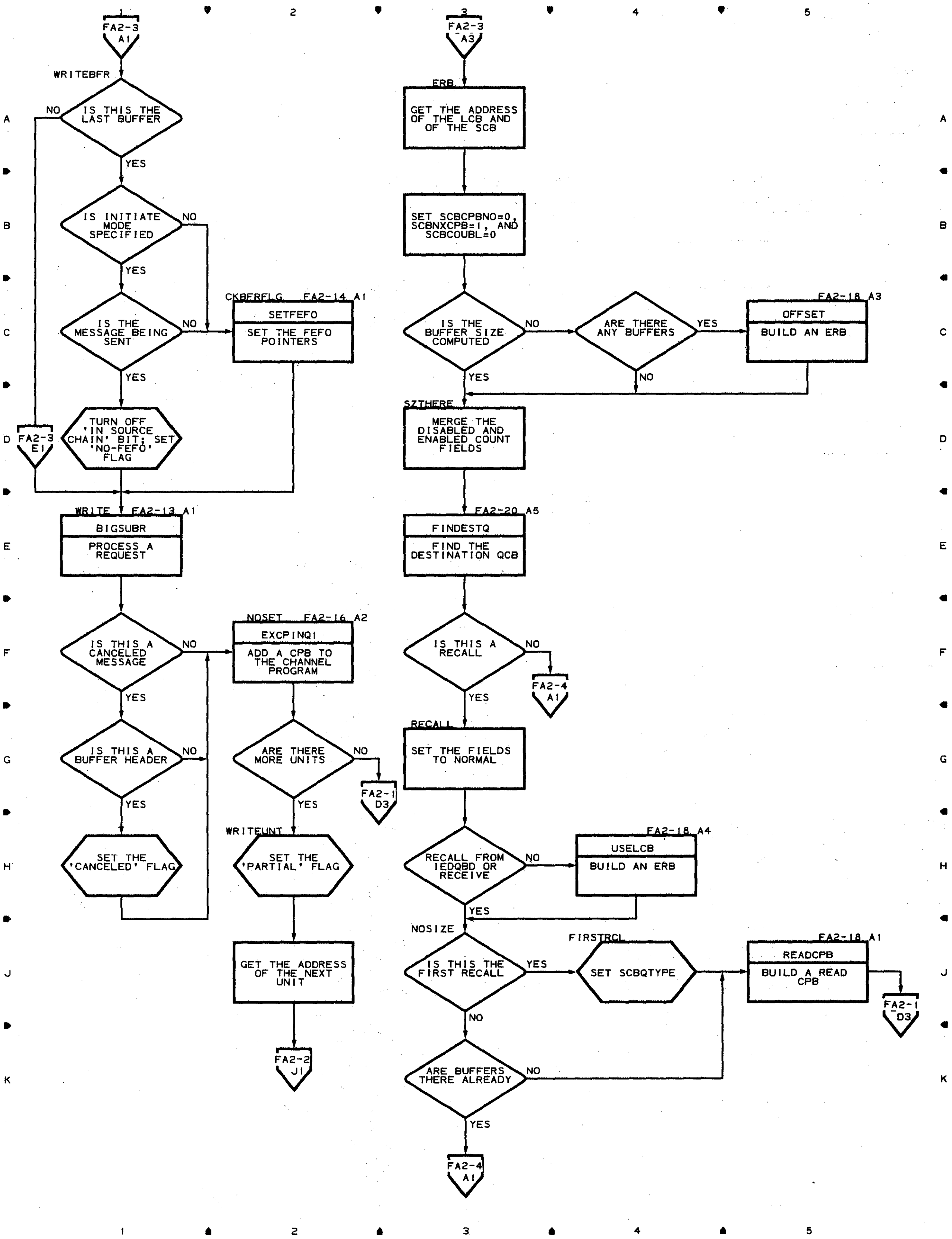
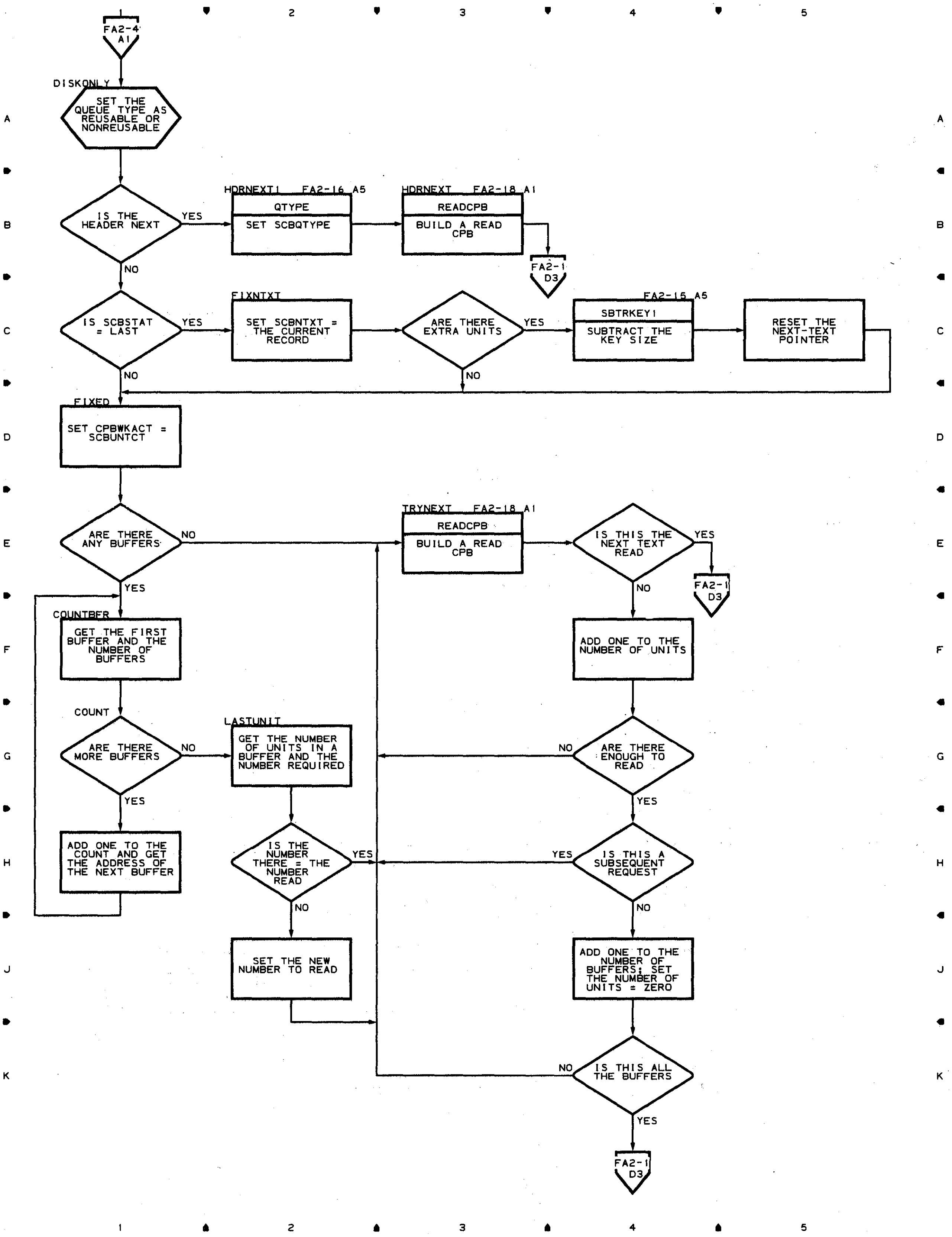


Chart FA2-4 CPB INITIALIZATION - DISK ONLY QUEUING



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
FA2-4 A1
DISKONLY
SET THE QUEUE TYPE AS REUSABLE OR NONREUSABLE
IS THE HEADER NEXT
YES
NO
HDRNEXT1 FA2-16 A5
QTYPE
SET SCBQTYPE
HDRNEXT FA2-18 A1
READCPB
BUILD A READ CPB
FA2-1 D3
IS SCBSTAT = LAST
YES
NO
FIXNTXT
SET SCBNTXT = THE CURRENT RECORD
ARE THERE EXTRA UNITS
YES
NO
FA2-15 A5
SBTRKEY1
SUBTRACT THE KEY SIZE
RESET THE NEXT-TEXT POINTER
FIXED
SET CPBWKACT = SCBUNTCT
ARE THERE ANY BUFFERS
NO
YES
TRYNEXT FA2-18 A1
READCPB
BUILD A READ CPB
IS THIS THE NEXT TEXT READ
YES
NO
FA2-1 D3
COUNTBFR
GET THE FIRST BUFFER AND THE NUMBER OF BUFFERS
ADD ONE TO THE NUMBER OF UNITS
COUNT
ARE THERE MORE BUFFERS
NO
YES
LASTUNIT
GET THE NUMBER OF UNITS IN A BUFFER AND THE NUMBER REQUIRED
ARE THERE ENOUGH TO READ
NO
YES
ADD ONE TO THE COUNT AND GET THE ADDRESS OF THE NEXT BUFFER
IS THE NUMBER THERE = THE NUMBER READ
YES
NO
IS THIS A SUBSEQUENT REQUEST
YES
NO
SET THE NEW NUMBER TO READ
ADD ONE TO THE NUMBER OF BUFFERS; SET THE NUMBER OF UNITS = ZERO
IS THIS ALL THE BUFFERS
NO
YES
FA2-1 D3

Chart FA2-5 CPB INITIALIZATION - DISK ONLY QUEUING

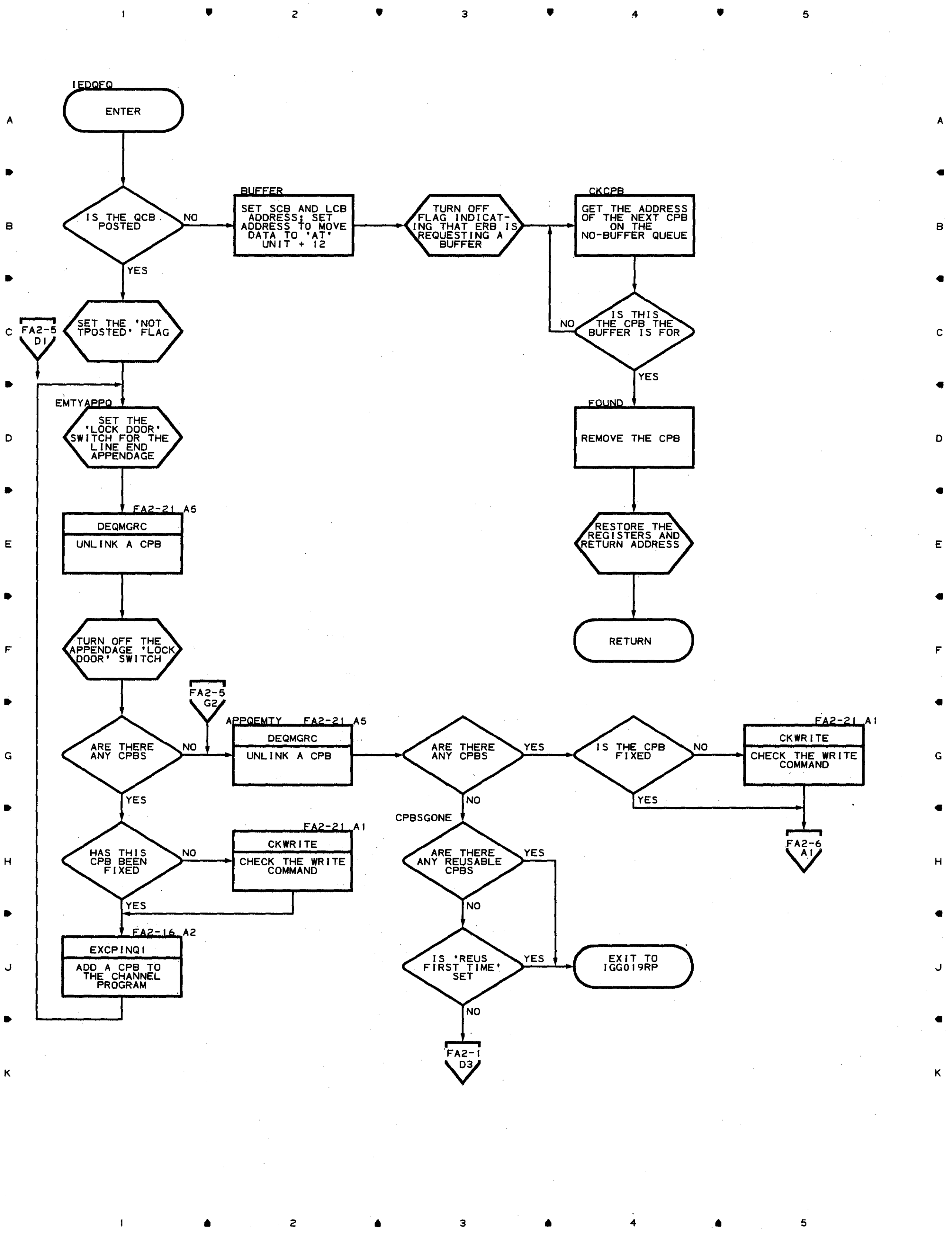


Chart FA2-6 CPB INITIALIZATION - DISK ONLY QUEUING

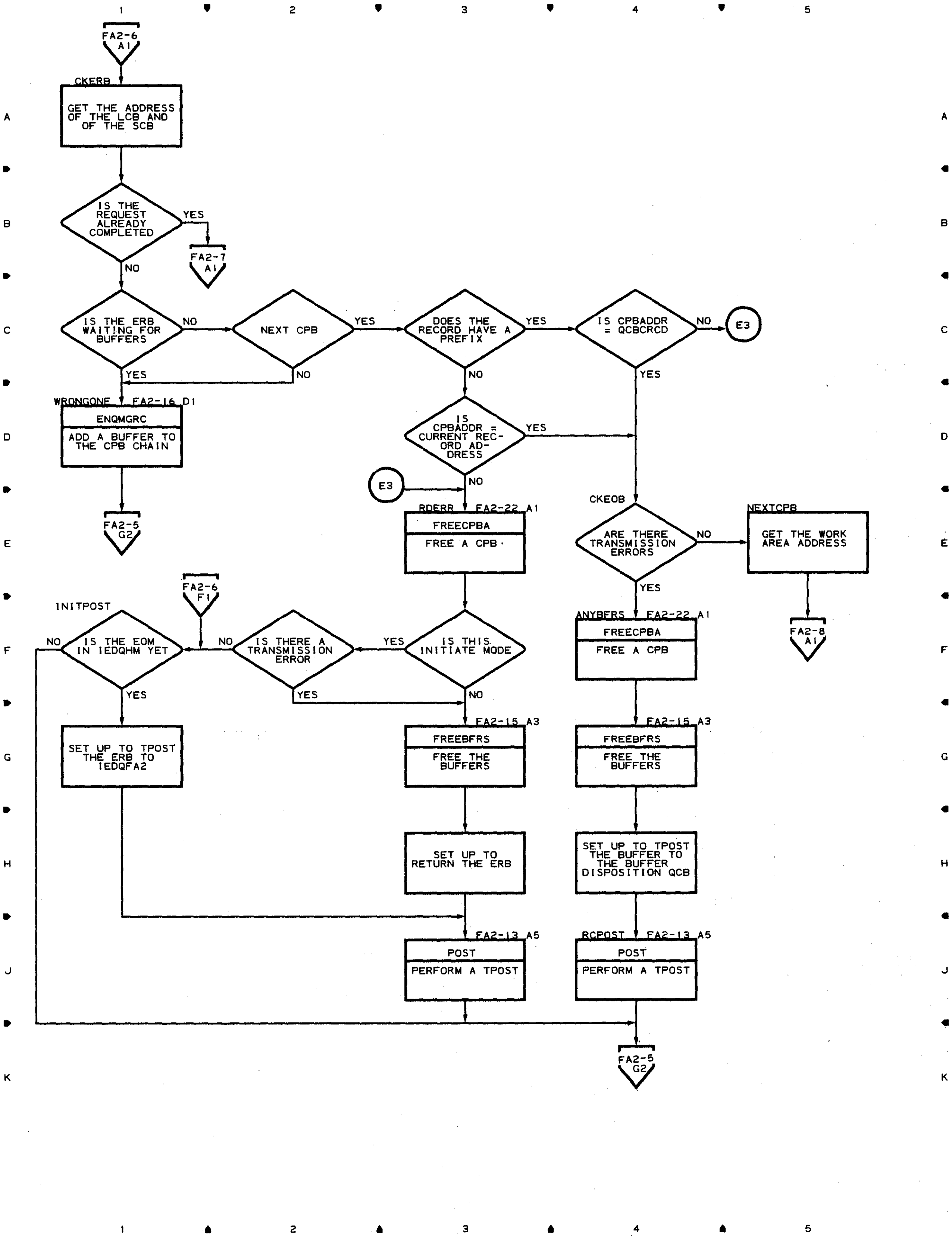


Chart FA2-7 CPB INITIALIZATION - DISK ONLY QUEUING

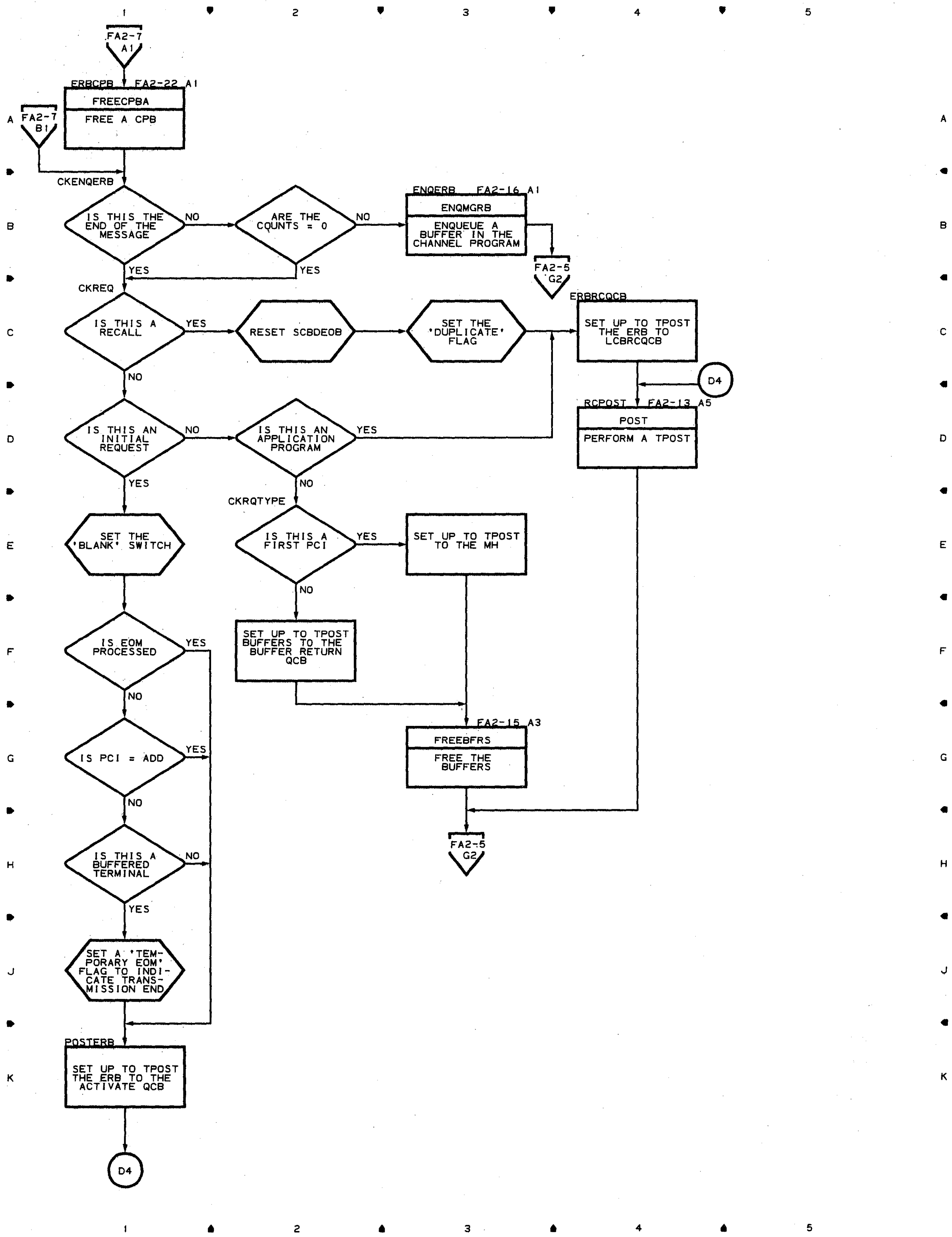


Chart FA2-8 CPB INITIALIZATION - DISK ONLY QUEUING

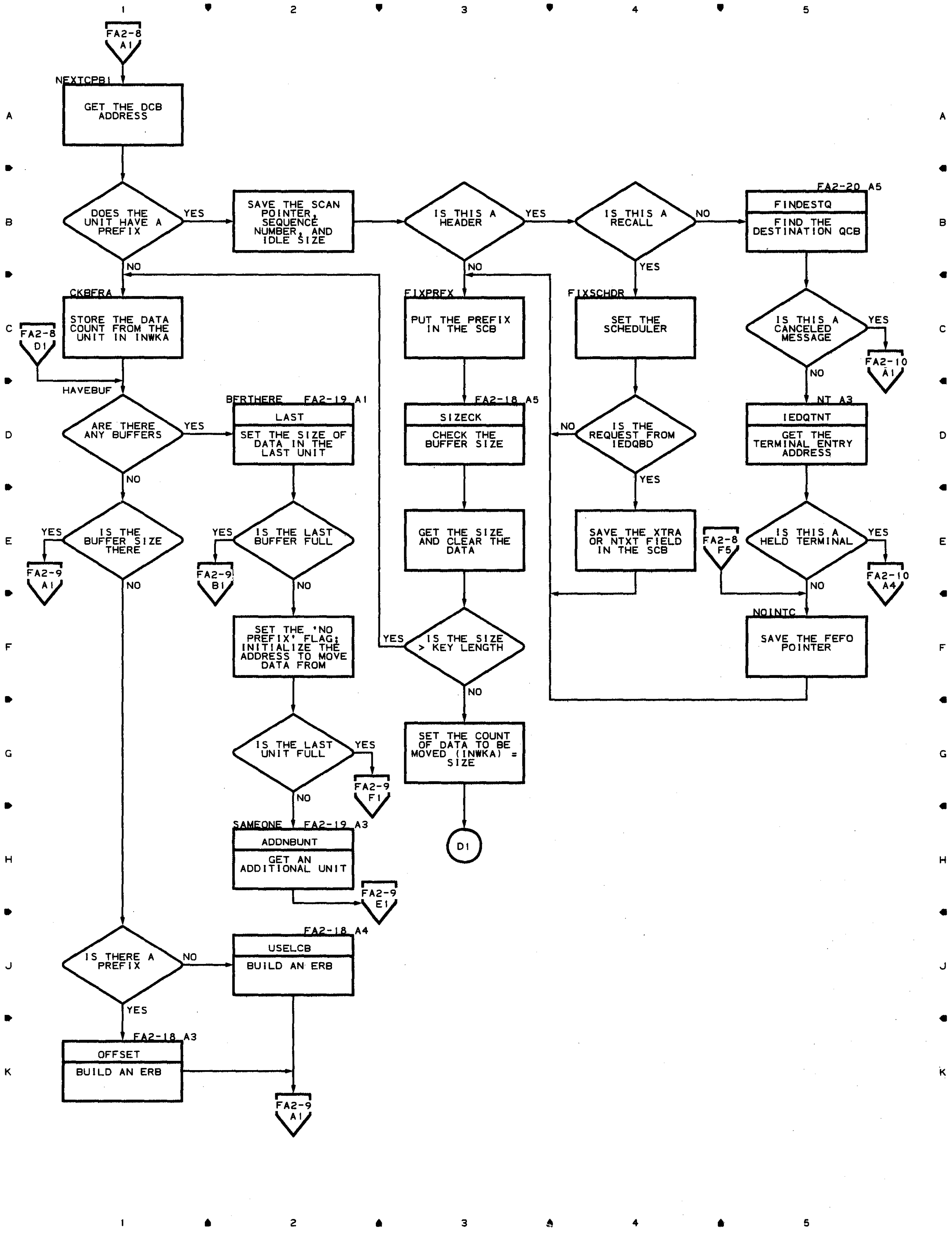


Chart FA2-9 CPB INITIALIZATION - DISK ONLY QUEUING

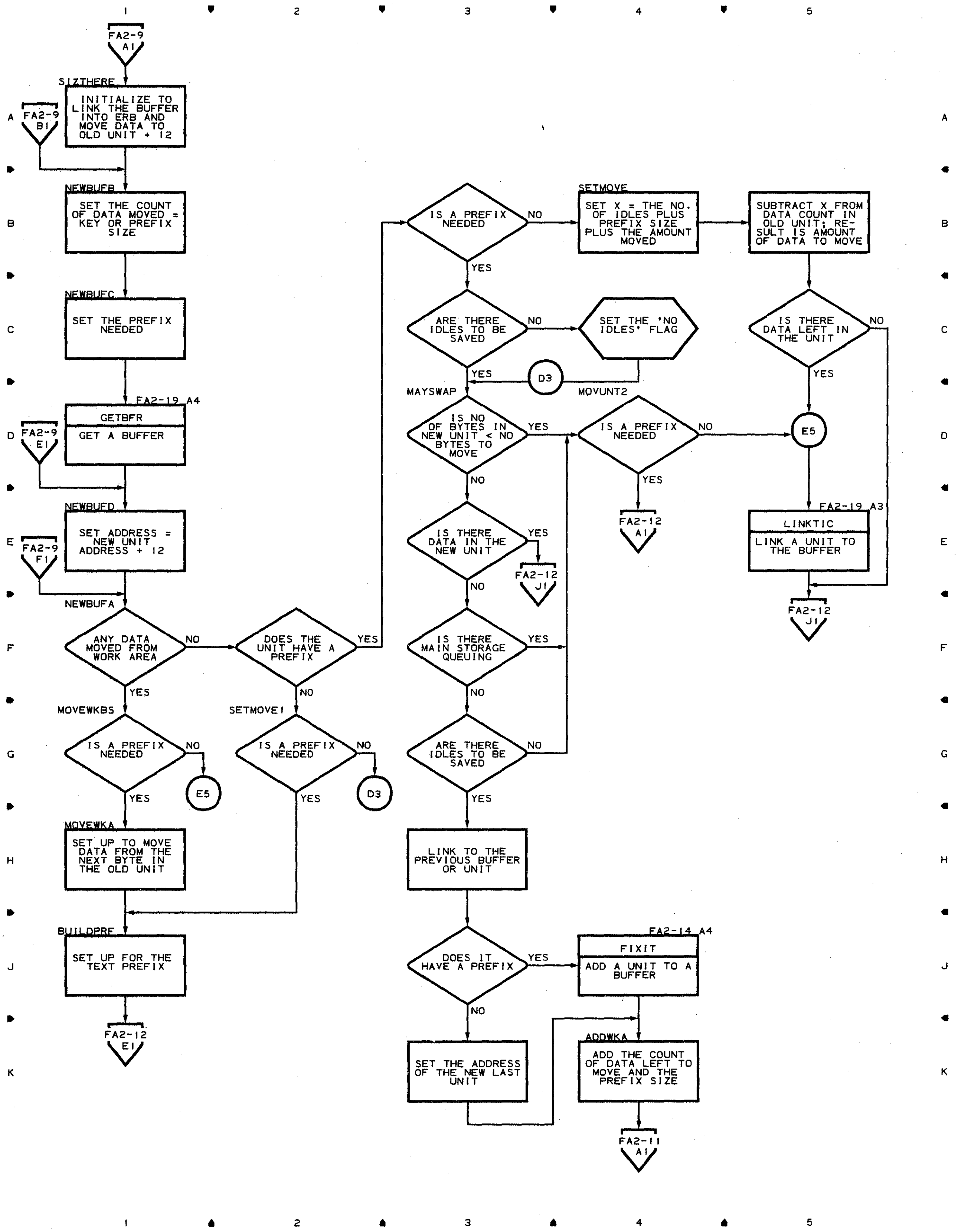

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
FA2-9 A1
SIZTHERE
INITIALIZE TO LINK THE BUFFER INTO ERB AND MOVE DATA TO OLD UNIT + 12
FA2-9 B1
NEWBUFB
SET THE COUNT OF DATA MOVED = KEY OR PREFIX SIZE
NEWBUFC
SET THE PREFIX NEEDED
FA2-19 A4
GETBFR
GET A BUFFER
FA2-9 E1
NEWBUFD
SET ADDRESS = NEW UNIT ADDRESS + 12
FA2-9 F1
NEWBUFA
ANY DATA MOVED FROM WORK AREA
NO
YES
MOVEWKBS
IS A PREFIX NEEDED
NO
E5
YES
MOVEWKA
SET UP TO MOVE DATA FROM THE NEXT BYTE IN THE OLD UNIT
BUILDPRE
SET UP FOR THE TEXT PREFIX
FA2-12 E1
DOES THE UNIT HAVE A PREFIX
YES
NO
SETMOVE1
IS A PREFIX NEEDED
NO
D3
YES
IS A PREFIX NEEDED
NO
YES
ARE THERE IDLES TO BE SAVED
NO
SET THE 'NO IDLES' FLAG
YES
D3
MAYSWAP
IS NO OF BYTES IN NEW UNIT < NO BYTES TO MOVE
YES
NO
IS THERE DATA IN THE NEW UNIT
YES
FA2-12 J1
NO
IS THERE MAIN STORAGE QUEUING
YES
NO
ARE THERE IDLES TO BE SAVED
NO
YES
LINK TO THE PREVIOUS BUFFER OR UNIT
DOES IT HAVE A PREFIX
YES
NO
SET THE ADDRESS OF THE NEW LAST UNIT
SETMOVE
SET X = THE NO. OF IDLES PLUS PREFIX SIZE PLUS THE AMOUNT MOVED
MOVUNT2
IS A PREFIX NEEDED
NO
YES
FA2-12 A1
FA2-14 A4
FIXIT
ADD A UNIT TO A BUFFER
ADDWKA
ADD THE COUNT OF DATA LEFT TO MOVE AND THE PREFIX SIZE
FA2-11 A1
SUBTRACT X FROM DATA COUNT IN OLD UNIT; RESULT IS AMOUNT OF DATA TO MOVE
IS THERE DATA LEFT IN THE UNIT
NO
YES
E5
FA2-19 A3
LINKTIC
LINK A UNIT TO THE BUFFER
FA2-12 J1

Chart FA2-10 CPB INITIALIZATION - DISK ONLY QUEUING

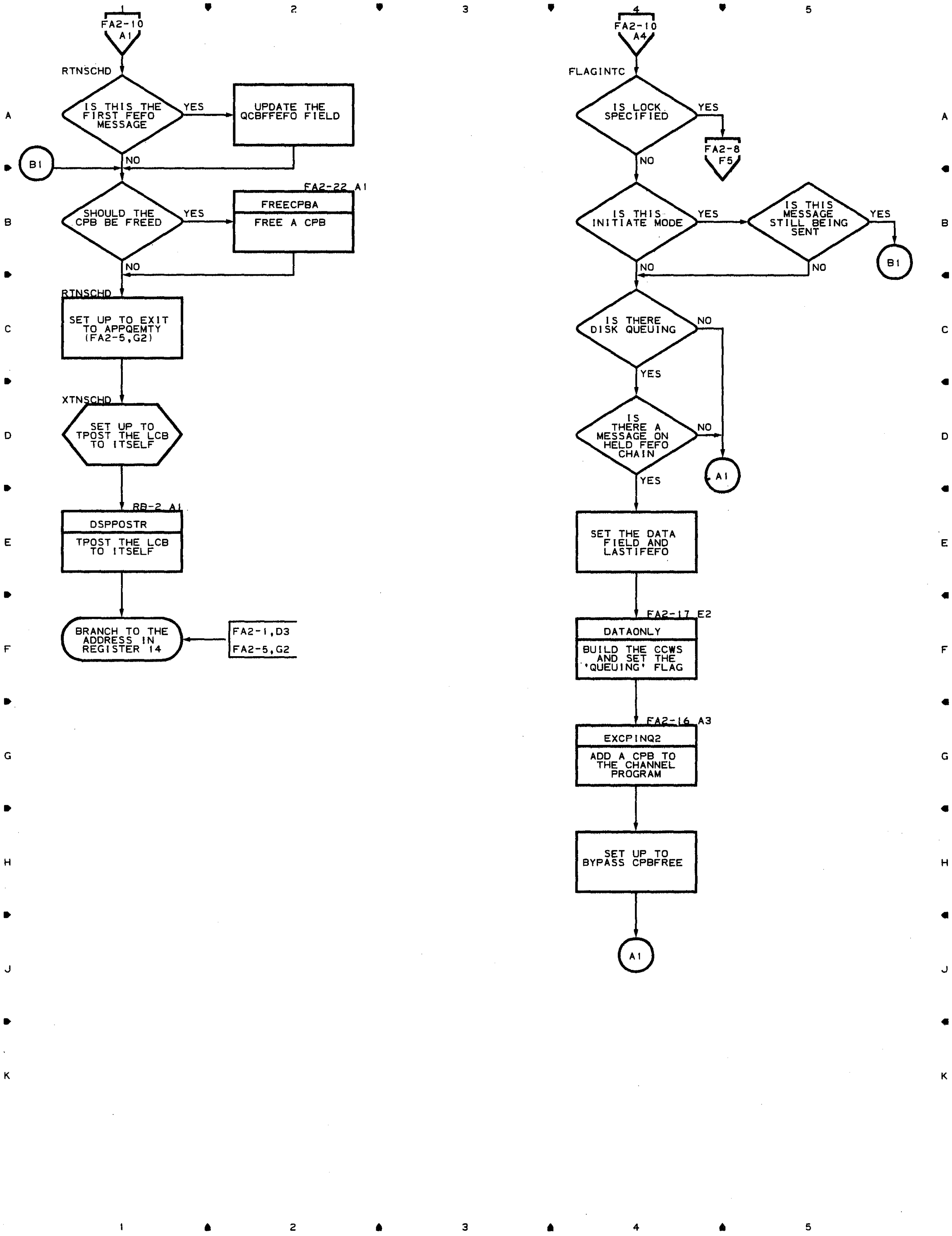

FA2-10 A1
RTNSCHD
IS THIS THE FIRST FEFO MESSAGE
YES
UPDATE THE QCBFFEFO FIELD
NO
B1
SHOULD THE CPB BE FREED
YES
FA2-22 A1
FREECPBA
FREE A CPB
NO
RTNSCHD
SET UP TO EXIT TO APPQEMTY (FA2-5,G2)
XTNSCHD
SET UP TO TPOST THE LCB TO ITSELF
RB-2 A1
DSPPOSTR
TPOST THE LCB TO ITSELF
BRANCH TO THE ADDRESS IN REGISTER 14
FA2-1,D3
FA2-5,G2
FA2-10 A4
FLAGINTC
IS LOCK SPECIFIED
YES
FA2-8 F5
NO
IS THIS INITIATE MODE
YES
IS THIS MESSAGE STILL BEING SENT
YES
B1
NO
NO
IS THERE DISK QUEUING
NO
YES
IS THERE A MESSAGE ON HELD FEFO CHAIN
NO
A1
YES
SET THE DATA FIELD AND LASTIFEFO
FA2-17 E2
DATAONLY
BUILD THE CCWS AND SET THE 'QUEUING' FLAG
FA2-16 A3
EXCPINQ2
ADD A CPB TO THE CHANNEL PROGRAM
SET UP TO BYPASS CPBFREE
A1
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart FA2-11 CPB INITIALIZATION - DISK ONLY QUEUING

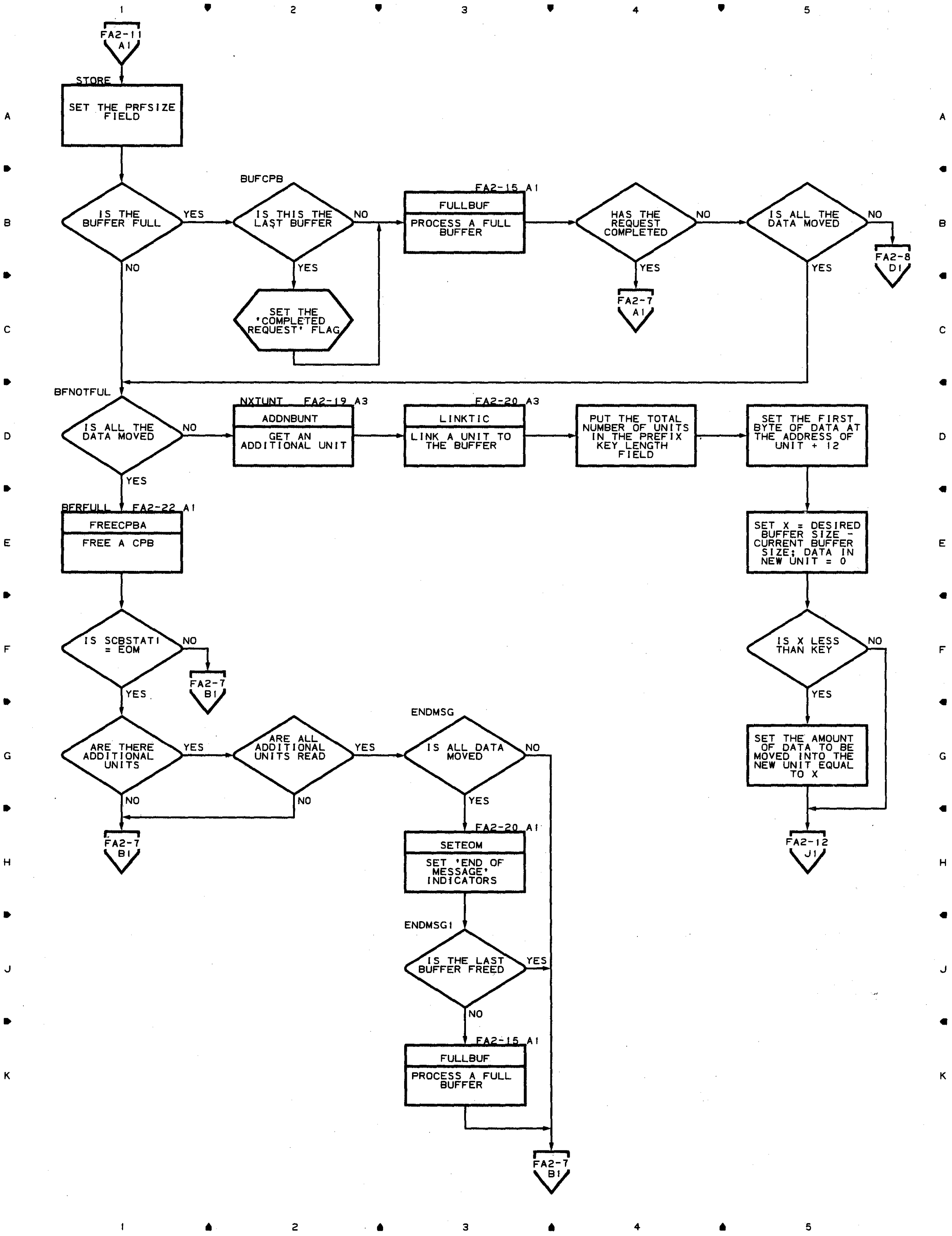


Chart FA2-12 CPB INITIALIZATION - DISK ONLY QUEUING

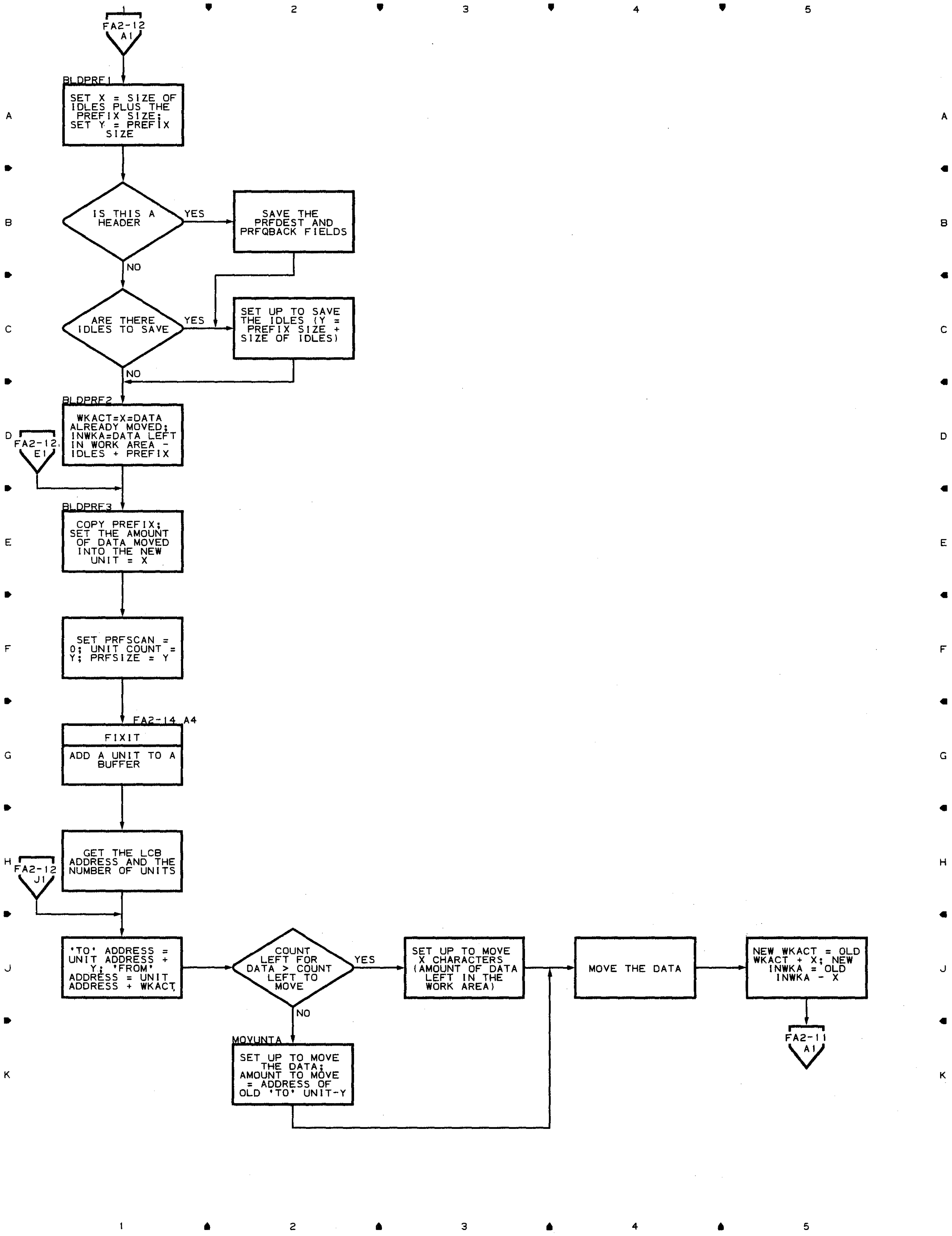

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
FA2-12 A1
BLDPREF1
SET X = SIZE OF IDLES PLUS THE PREFIX SIZE; SET Y = PREFIX SIZE
IS THIS A HEADER
YES
NO
SAVE THE PRFDEST AND PRFQBACK FIELDS
ARE THERE IDLES TO SAVE
YES
NO
SET UP TO SAVE THE IDLES (Y = PREFIX SIZE + SIZE OF IDLES)
BLDPREF2
WKACT=X=DATA ALREADY MOVED; INWKA=DATA LEFT IN WORK AREA - IDLES + PREFIX
FA2-12 E1
BLDPREF3
COPY PREFIX; SET THE AMOUNT OF DATA MOVED INTO THE NEW UNIT = X
SET PRFSCAN = 0; UNIT COUNT = Y; PRFSIZE = Y
FA2-14 A4
FIXIT
ADD A UNIT TO A BUFFER
GET THE LCB ADDRESS AND THE NUMBER OF UNITS
FA2-12 J1
'TO' ADDRESS = UNIT ADDRESS + Y; 'FROM' ADDRESS = UNIT ADDRESS + WKACT
COUNT LEFT FOR DATA > COUNT LEFT TO MOVE
YES
NO
SET UP TO MOVE X CHARACTERS (AMOUNT OF DATA LEFT IN THE WORK AREA)
MOVE THE DATA
NEW WKACT = OLD WKACT + X; NEW INWKA = OLD INWKA - X
FA2-11 A1
MOVUNTA
SET UP TO MOVE THE DATA; AMOUNT TO MOVE = ADDRESS OF OLD 'TO' UNIT-Y

Chart FA2-13 CPB INITIALIZATION - DISK ONLY QUEUING

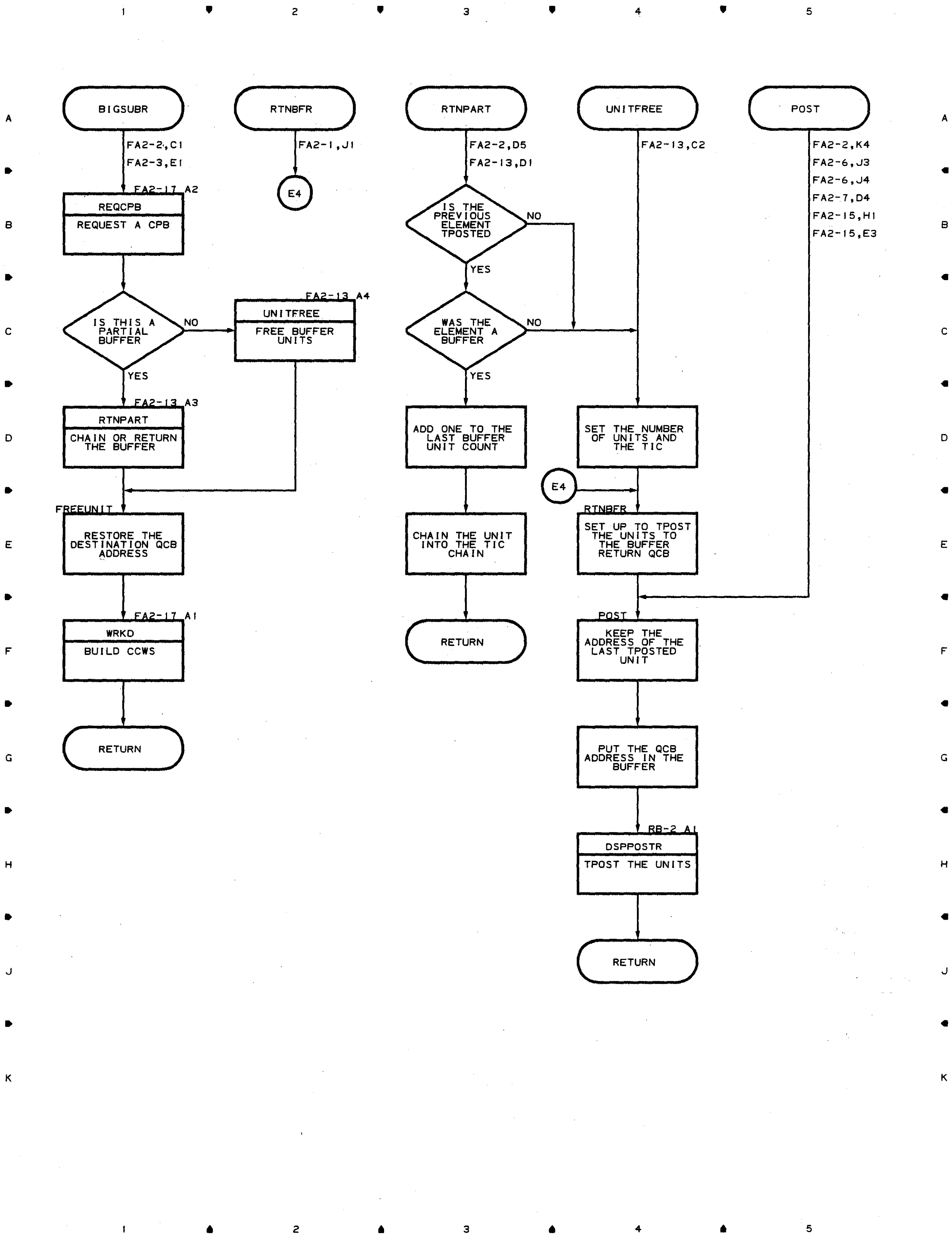


Chart FA2-14 CPB INITIALIZATION -DISK ONLY QUEUING

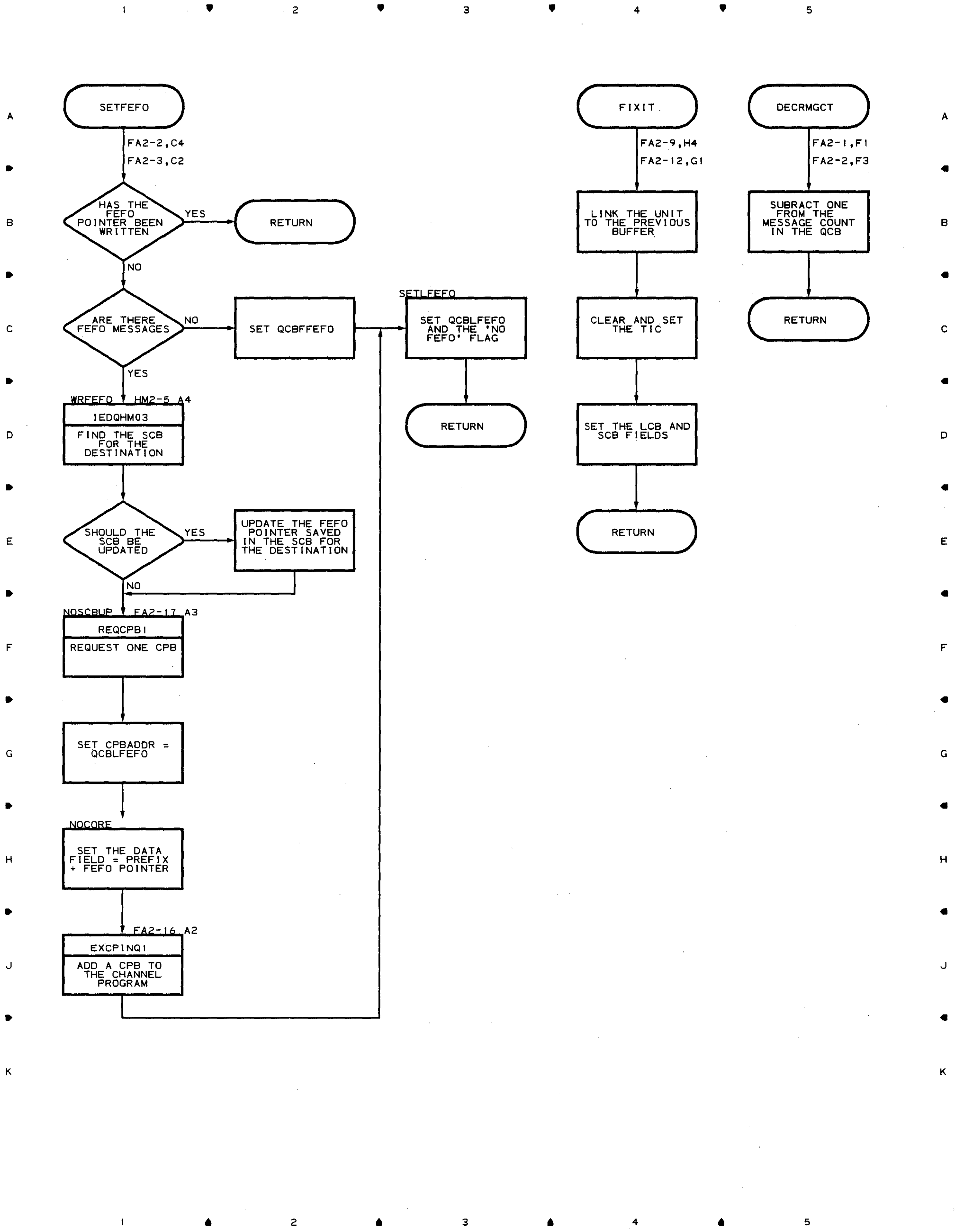

SETFEFO
FA2-2,C4
FA2-3,C2
HAS THE FEFO POINTER BEEN WRITTEN
YES
RETURN
NO
ARE THERE FEFO MESSAGES
NO
SET QCBFFEFO
SETLFEFO
SET QCBLFEFO AND THE 'NO FEFO' FLAG
RETURN
YES
WRFEFO
HM2-5 A4
IEDQHM03
FIND THE SCB FOR THE DESTINATION
SHOULD THE SCB BE UPDATED
YES
UPDATE THE FEFO POINTER SAVED IN THE SCB FOR THE DESTINATION
NO
NOSCBUP
FA2-17 A3
REQCPB1
REQUEST ONE CPB
SET CPBADDR = QCBLFEFO
NOCORE
SET THE DATA FIELD = PREFIX + FEFO POINTER
FA2-16 A2
EXCPINQ1
ADD A CPB TO THE CHANNEL PROGRAM
FIXIT
FA2-9,H4
FA2-12,G1
LINK THE UNIT TO THE PREVIOUS BUFFER
CLEAR AND SET THE TIC
SET THE LCB AND SCB FIELDS
RETURN
DECRMGCT
FA2-1,F1
FA2-2,F3
SUBRACT ONE FROM THE MESSAGE COUNT IN THE QCB
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart FA2-15 CPB INITIALIZATION - DISK ONLY QUEUING

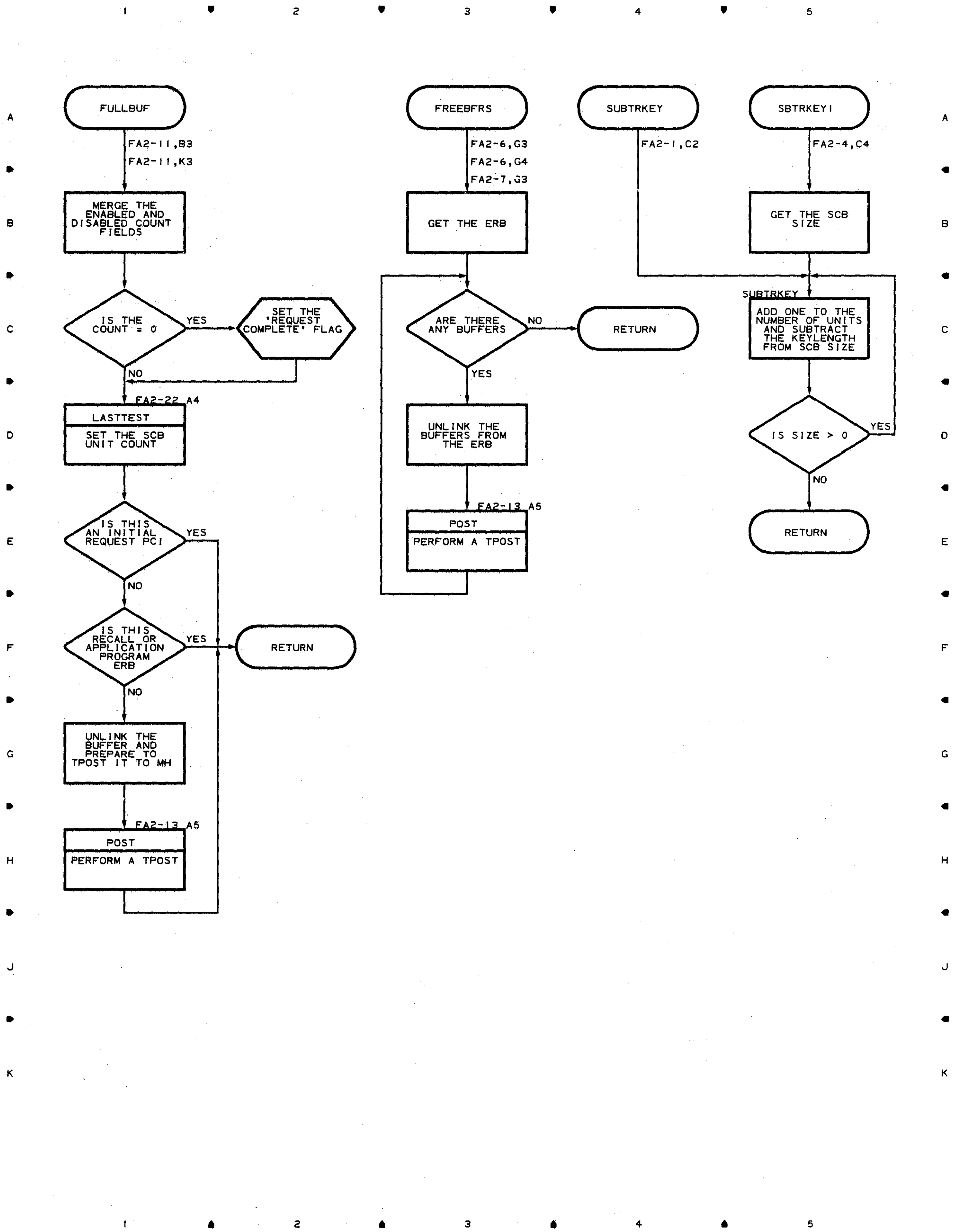


Chart FA2-16 CPB INITIALIZATION - DISK ONLY QUEUING

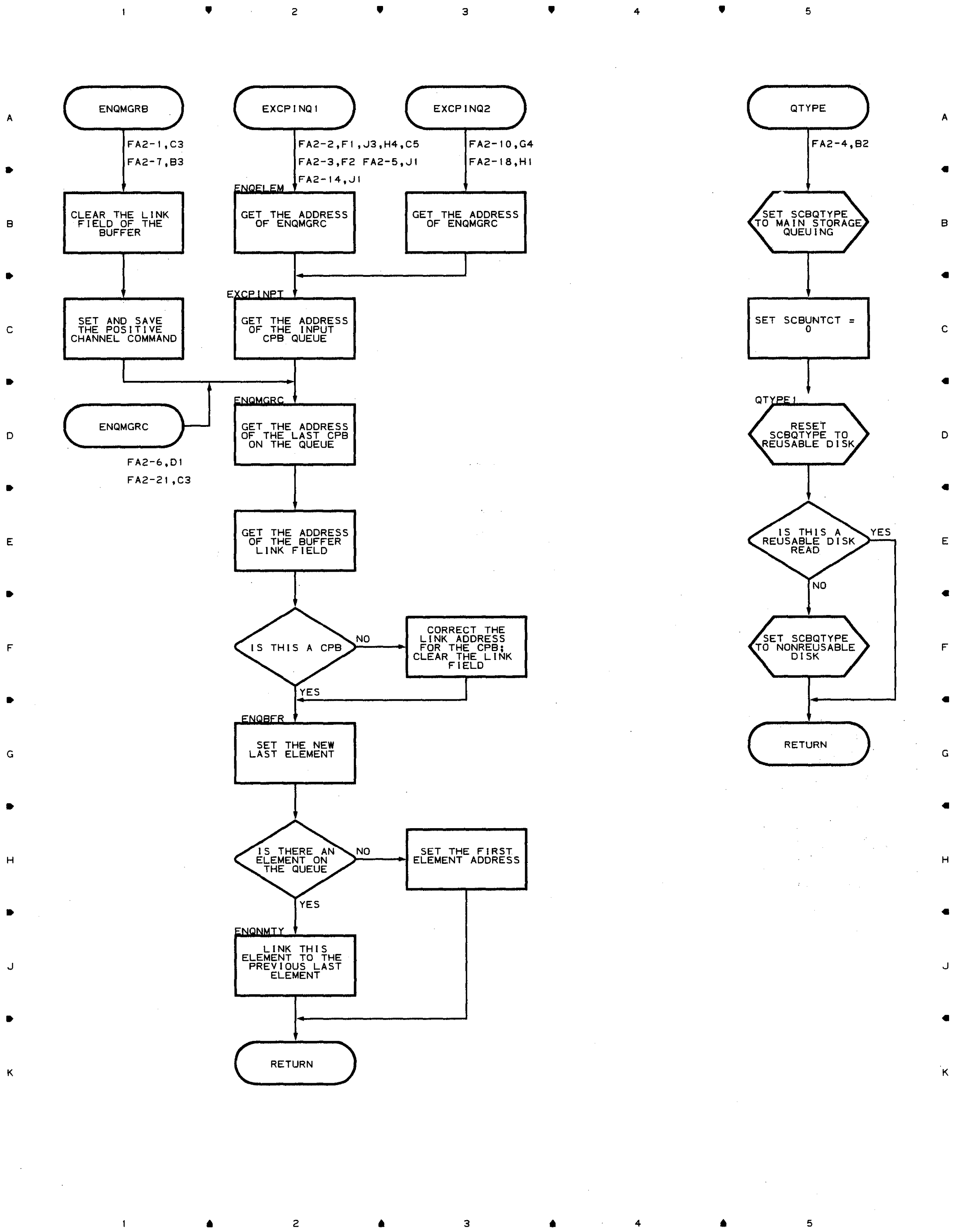
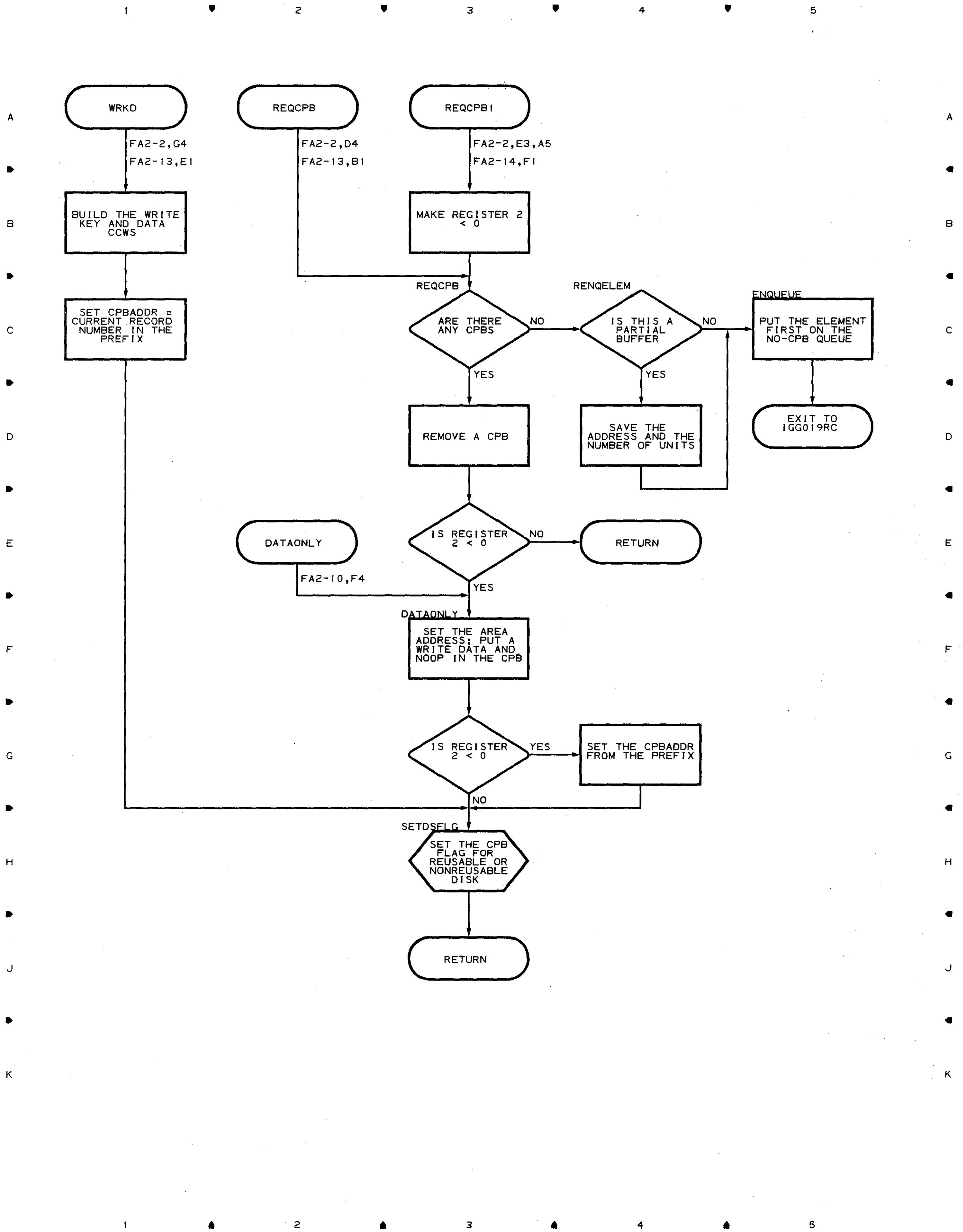
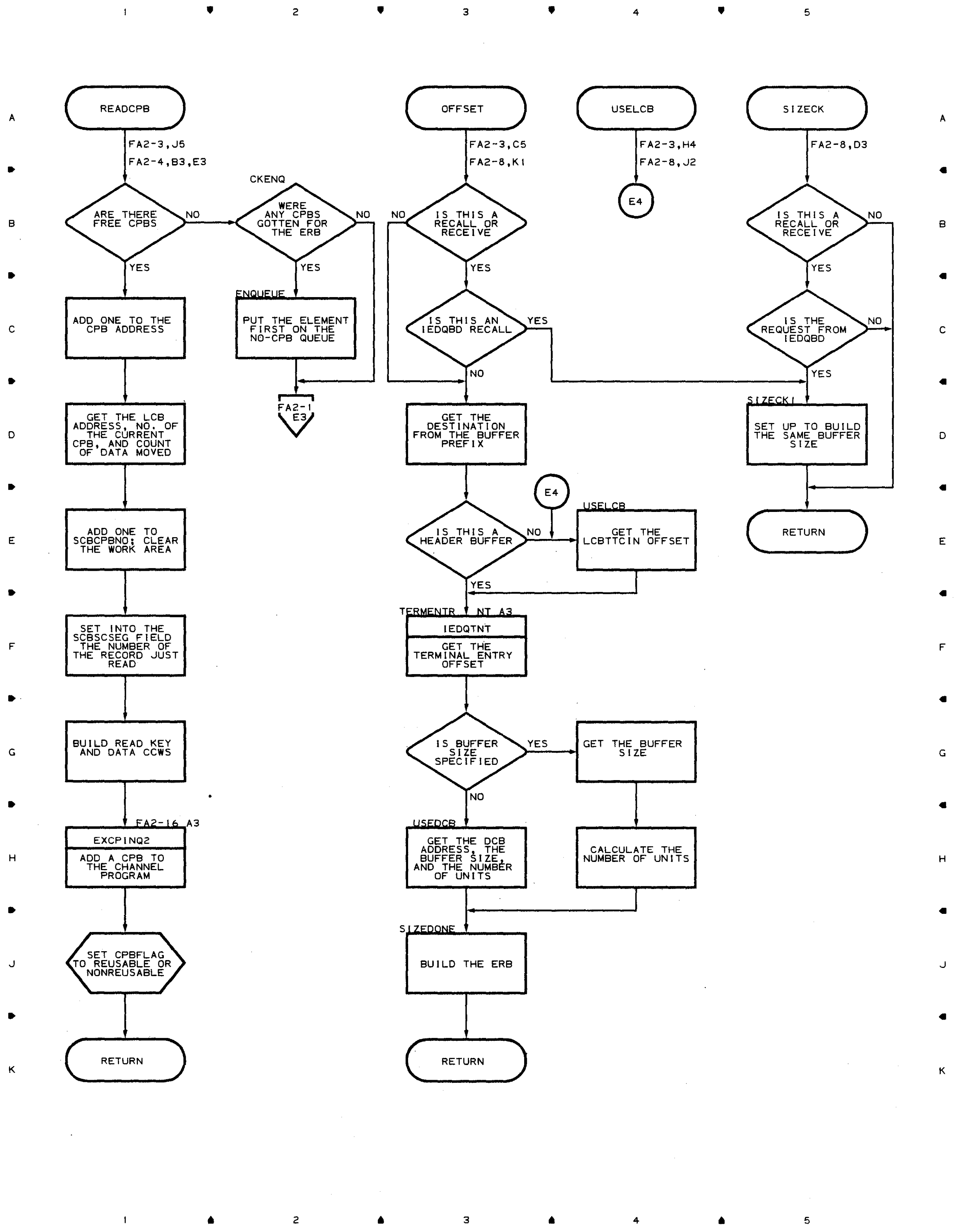

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
ENQMGRB
FA2-1,C3
FA2-7,B3
CLEAR THE LINK FIELD OF THE BUFFER
SET AND SAVE THE POSITIVE CHANNEL COMMAND
ENQMGRC
FA2-6,D1
FA2-21,C3
EXCPINQ1
FA2-2,F1,J3,H4,C5
FA2-3,F2 FA2-5,J1
FA2-14,J1
ENQELEM
GET THE ADDRESS OF ENQMGRC
EXCPINQ2
FA2-10,G4
FA2-18,H1
GET THE ADDRESS OF ENQMGRC
EXCPINPT
GET THE ADDRESS OF THE INPUT CPB QUEUE
ENQMGRC
GET THE ADDRESS OF THE LAST CPB ON THE QUEUE
GET THE ADDRESS OF THE BUFFER LINK FIELD
IS THIS A CPB
NO
CORRECT THE LINK ADDRESS FOR THE CPB; CLEAR THE LINK FIELD
YES
ENQBFR
SET THE NEW LAST ELEMENT
IS THERE AN ELEMENT ON THE QUEUE
NO
SET THE FIRST ELEMENT ADDRESS
YES
ENQNMTY
LINK THIS ELEMENT TO THE PREVIOUS LAST ELEMENT
RETURN
QTYPE
FA2-4,B2
SET SCBQTYPE TO MAIN STORAGE QUEUING
SET SCBUNTCT = 0
QTYPE1
RESET SCBQTYPE TO REUSABLE DISK
IS THIS A REUSABLE DISK READ
YES
NO
SET SCBQTYPE TO NONREUSABLE DISK
RETURN

Chart FA2-17 CPB INITIALIZATION - DISK ONLY QUEUING



WRKD
FA2-2,G4
FA2-13,E1
BUILD THE WRITE KEY AND DATA CCWS
SET CPBADDR = CURRENT RECORD NUMBER IN THE PREFIX
REQCPB
FA2-2,D4
FA2-13,B1
REQCPB1
FA2-2,E3,A5
FA2-14,F1
MAKE REGISTER 2 < 0
REQCPB
ARE THERE ANY CPBS
NO
YES
RENQELEM
IS THIS A PARTIAL BUFFER
NO
YES
ENQUEUE
PUT THE ELEMENT FIRST ON THE NO-CPB QUEUE
EXIT TO IGG019RC
REMOVE A CPB
SAVE THE ADDRESS AND THE NUMBER OF UNITS
DATAONLY
FA2-10,F4
IS REGISTER 2 < 0
NO
RETURN
YES
DATAONLY
SET THE AREA ADDRESS; PUT A WRITE DATA AND NOOP IN THE CPB
IS REGISTER 2 < 0
YES
SET THE CPBADDR FROM THE PREFIX
NO
SETDSFLG
SET THE CPB FLAG FOR REUSABLE OR NONREUSABLE DISK
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart FA2-18 CPB INITIALIZATION -DISK ONLY QUEUING



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
READCPB
FA2-3,J5
FA2-4,B3,E3
ARE THERE FREE CPBS
NO
YES
ADD ONE TO THE CPB ADDRESS
GET THE LCB ADDRESS, NO. OF THE CURRENT CPB, AND COUNT OF DATA MOVED
ADD ONE TO SCBCPBNO; CLEAR THE WORK AREA
SET INTO THE SCBSCSEG FIELD THE NUMBER OF THE RECORD JUST READ
BUILD READ KEY AND DATA CCWS
FA2-16 A3
EXCPINQ2
ADD A CPB TO THE CHANNEL PROGRAM
SET CPBFLAG TO REUSABLE OR NONREUSABLE
RETURN
CKENQ
WERE ANY CPBS GOTTEN FOR THE ERB
NO
YES
ENQUEUE
PUT THE ELEMENT FIRST ON THE NO-CPB QUEUE
FA2-1 E3
OFFSET
FA2-3,C5
FA2-8,K1
IS THIS A RECALL OR RECEIVE
NO
YES
IS THIS AN IEDQBD RECALL
YES
NO
GET THE DESTINATION FROM THE BUFFER PREFIX
IS THIS A HEADER BUFFER
NO
YES
TERMENTR
NT A3
IEDQTNT
GET THE TERMINAL ENTRY OFFSET
IS BUFFER SIZE SPECIFIED
YES
NO
USEDCB
GET THE DCB ADDRESS, THE BUFFER SIZE, AND THE NUMBER OF UNITS
SIZEDONE
BUILD THE ERB
RETURN
USELCB
FA2-3,H4
FA2-8,J2
E4
E4
USELCB
GET THE LCBTTCIN OFFSET
GET THE BUFFER SIZE
CALCULATE THE NUMBER OF UNITS
SIZECK
FA2-8,D3
IS THIS A RECALL OR RECEIVE
NO
YES
IS THE REQUEST FROM IEDQBD
NO
YES
SIZECK1
SET UP TO BUILD THE SAME BUFFER SIZE
RETURN

Chart FA2-19 CPB INITIALIZATION - DISK ONLY QUEUING

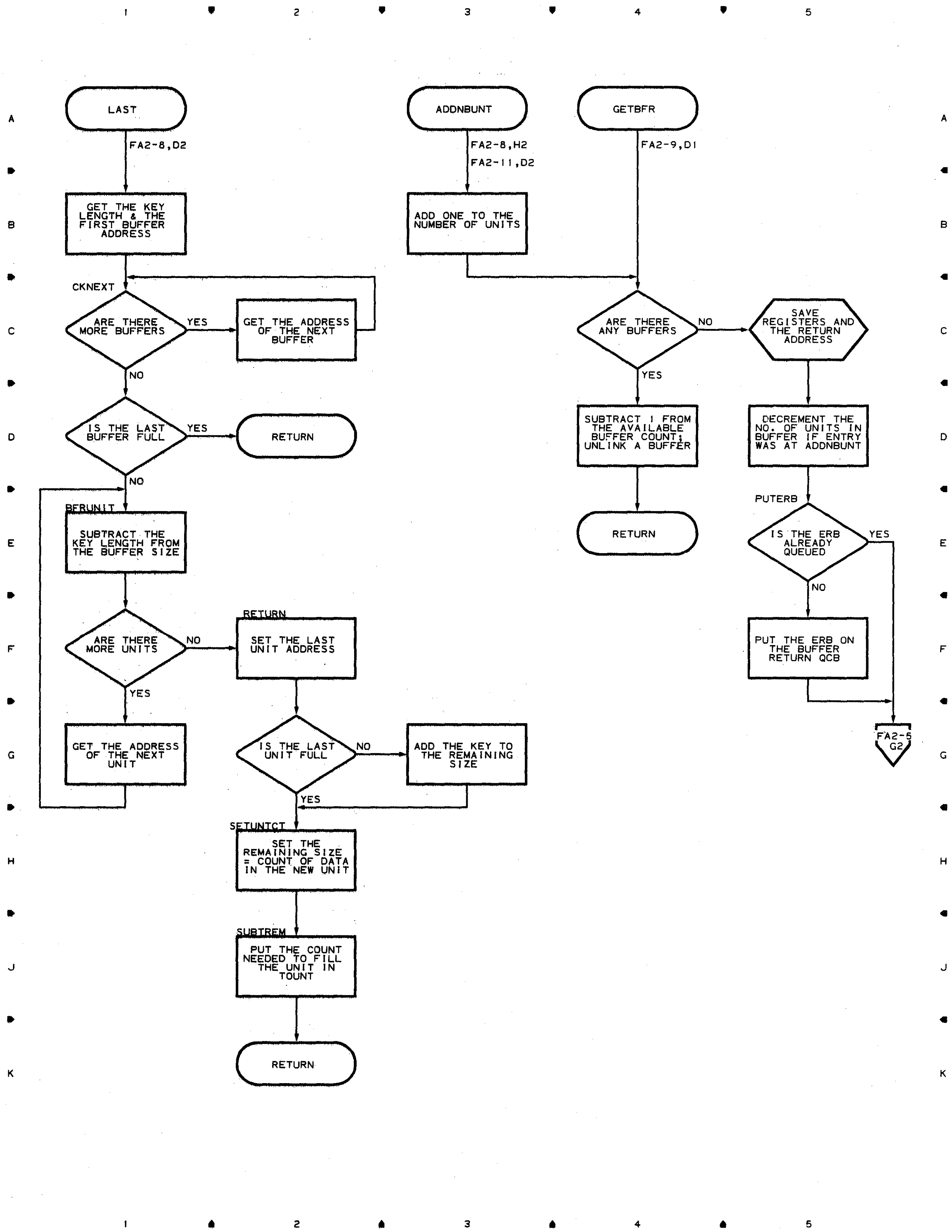
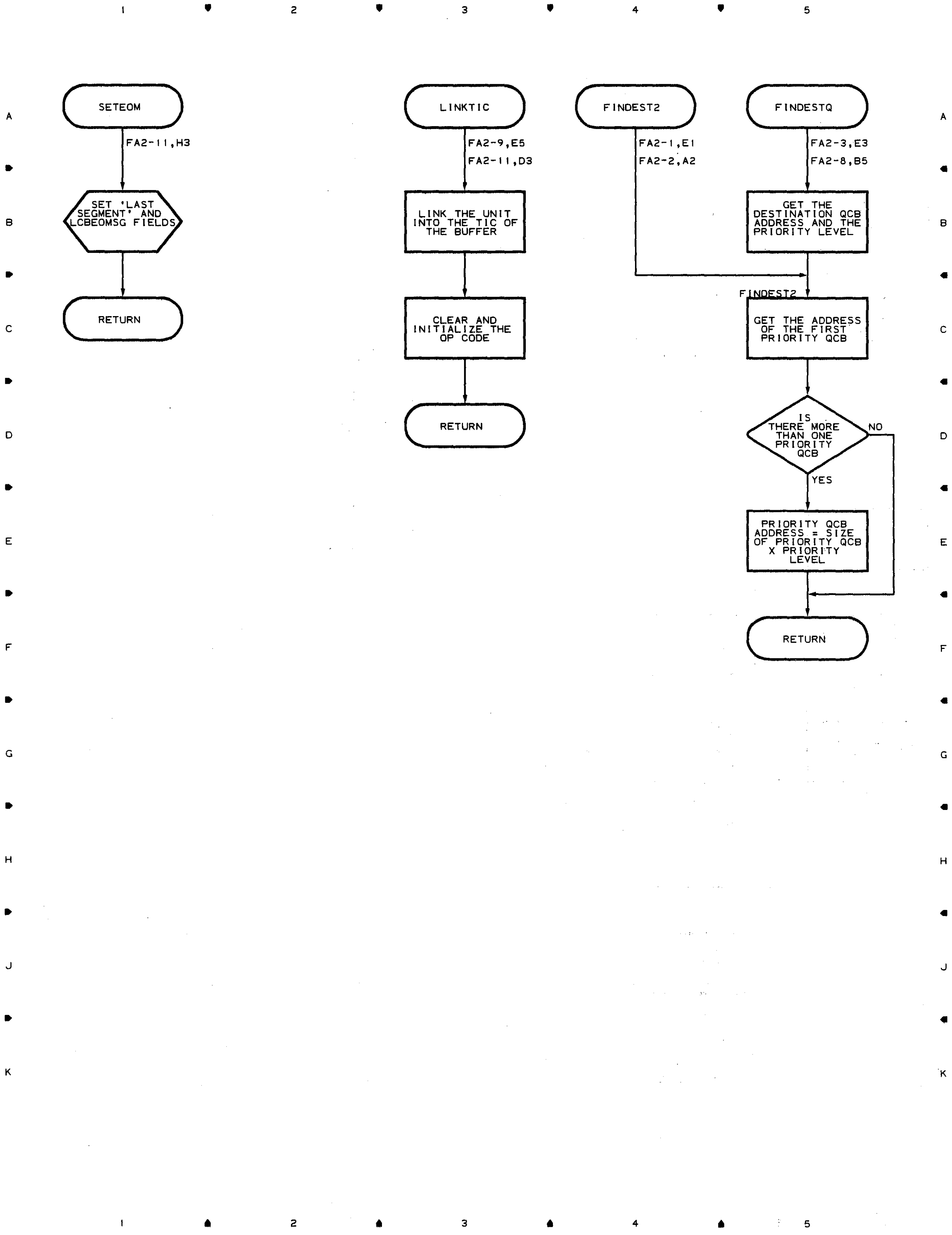
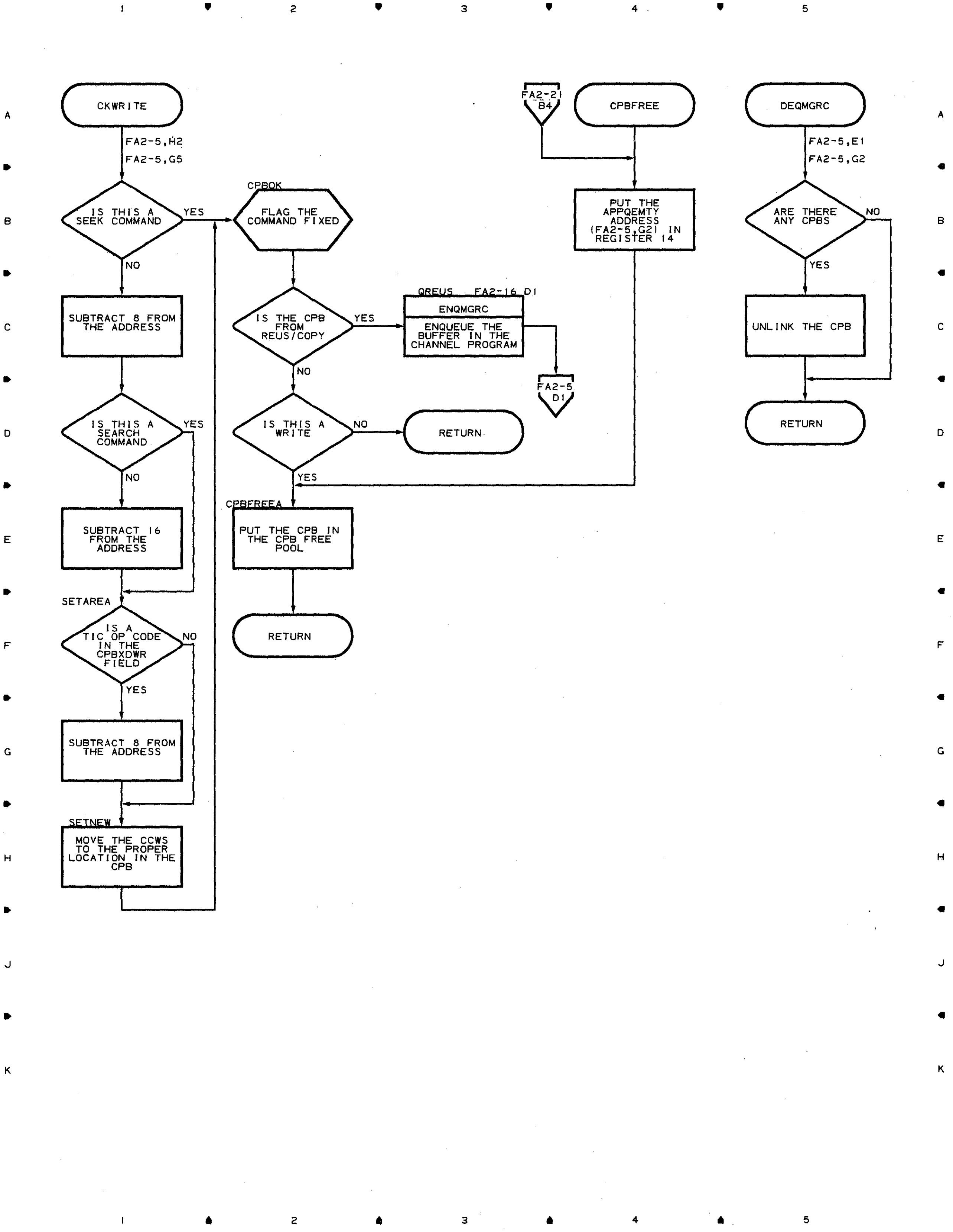

LAST
FA2-8,D2
GET THE KEY LENGTH & THE FIRST BUFFER ADDRESS
CKNEXT
ARE THERE MORE BUFFERS
YES
GET THE ADDRESS OF THE NEXT BUFFER
NO
IS THE LAST BUFFER FULL
YES
RETURN
NO
BFRUNIT
SUBTRACT THE KEY LENGTH FROM THE BUFFER SIZE
ARE THERE MORE UNITS
NO
RETURN
SET THE LAST UNIT ADDRESS
YES
GET THE ADDRESS OF THE NEXT UNIT
IS THE LAST UNIT FULL
NO
ADD THE KEY TO THE REMAINING SIZE
YES
SETUNTCT
SET THE REMAINING SIZE = COUNT OF DATA IN THE NEW UNIT
SUBTREM
PUT THE COUNT NEEDED TO FILL THE UNIT IN TOUNT
RETURN
ADDNBUNT
FA2-8,H2
FA2-11,D2
ADD ONE TO THE NUMBER OF UNITS
GETBFR
FA2-9,D1
ARE THERE ANY BUFFERS
NO
SAVE REGISTERS AND THE RETURN ADDRESS
YES
SUBTRACT 1 FROM THE AVAILABLE BUFFER COUNT; UNLINK A BUFFER
RETURN
DECREMENT THE NO. OF UNITS IN BUFFER IF ENTRY WAS AT ADDNBUNT
PUTERB
IS THE ERB ALREADY QUEUED
YES
NO
PUT THE ERB ON THE BUFFER RETURN QCB
FA2-5 G2
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart FA2-20 CPB INITIALIZATION - DISK ONLY QUEUING



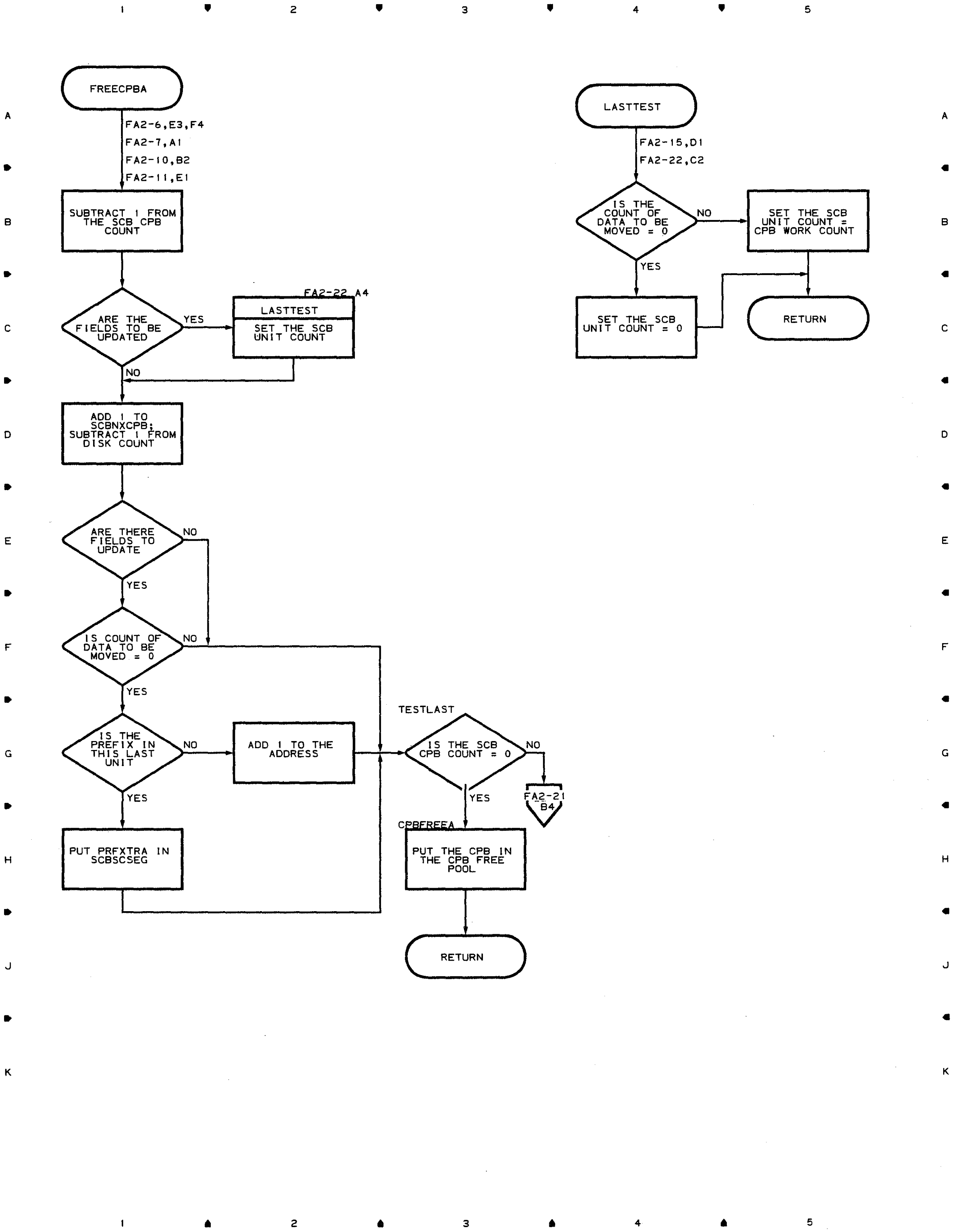
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
SETEOM
FA2-11,H3
SET 'LAST SEGMENT' AND LCBEOMSG FIELDS
RETURN
LINKTIC
FA2-9,E5
FA2-11,D3
LINK THE UNIT INTO THE TIC OF THE BUFFER
CLEAR AND INITIALIZE THE OP CODE
RETURN
FINDEST2
FA2-1,E1
FA2-2,A2
FINDESTQ
FA2-3,E3
FA2-8,B5
GET THE DESTINATION QCB ADDRESS AND THE PRIORITY LEVEL
FINDEST2
GET THE ADDRESS OF THE FIRST PRIORITY QCB
IS THERE MORE THAN ONE PRIORITY QCB
NO
YES
PRIORITY QCB ADDRESS = SIZE OF PRIORITY QCB X PRIORITY LEVEL
RETURN

Chart FA2-21 CPB INITIALIZATION - DISK ONLY QUEUING



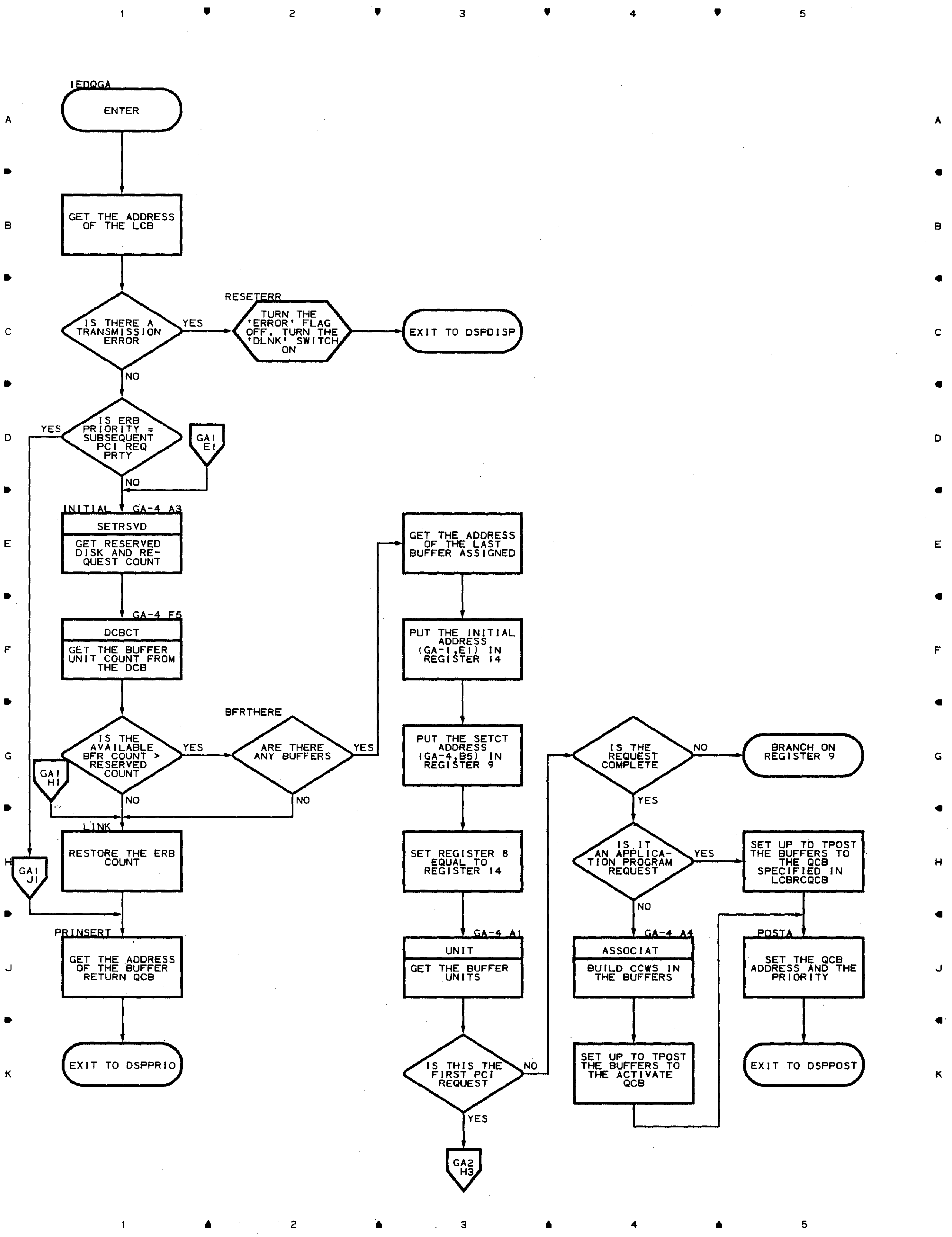
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
CKWRITE
FA2-5,H2
FA2-5,G5
IS THIS A SEEK COMMAND
YES
NO
SUBTRACT 8 FROM THE ADDRESS
IS THIS A SEARCH COMMAND
YES
NO
SUBTRACT 16 FROM THE ADDRESS
SETAREA
IS A TIC OP CODE IN THE CPBXDWR FIELD
NO
YES
SUBTRACT 8 FROM THE ADDRESS
SETNEW
MOVE THE CCWS TO THE PROPER LOCATION IN THE CPB
CPBOK
FLAG THE COMMAND FIXED
IS THE CPB FROM REUS/COPY
YES
NO
QREUS FA2-16 D1
ENQMGRC
ENQUEUE THE BUFFER IN THE CHANNEL PROGRAM
FA2-5 D1
IS THIS A WRITE
NO
YES
RETURN
CPBFREEA
PUT THE CPB IN THE CPB FREE POOL
RETURN
FA2-21 B4
CPBFREE
PUT THE APPQEMTY ADDRESS (FA2-5,G2) IN REGISTER 14
DEQMGRC
FA2-5,E1
FA2-5,G2
ARE THERE ANY CPBS
NO
YES
UNLINK THE CPB
RETURN

Chart FA2-22 CPB INITIALIZATION - DISK ONLY QUEUING



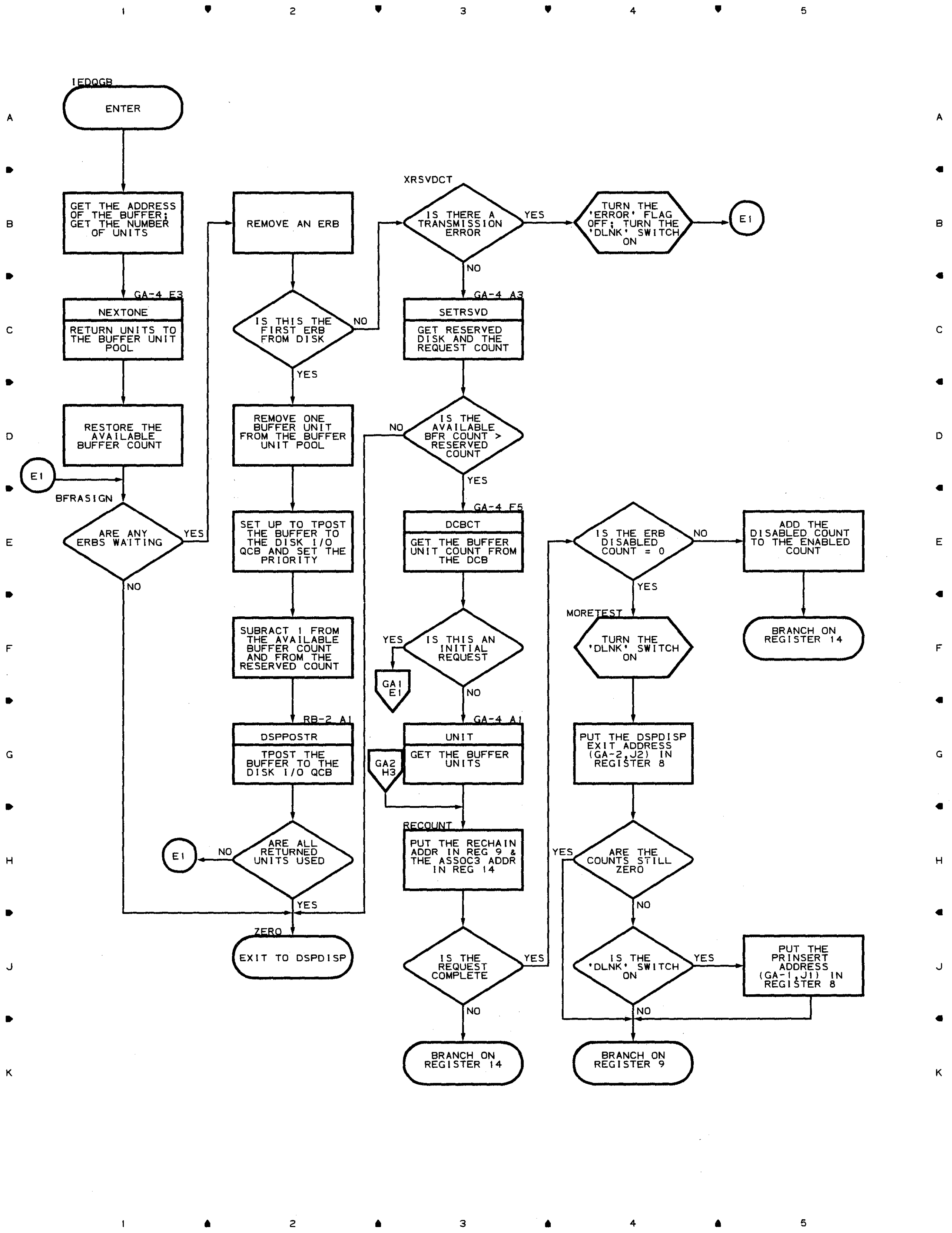
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
FREECPBA
FA2-6,E3,F4
FA2-7,A1
FA2-10,B2
FA2-11,E1
SUBTRACT 1 FROM THE SCB CPB COUNT
ARE THE FIELDS TO BE UPDATED
YES
NO
FA2-22 A4
LASTTEST
SET THE SCB UNIT COUNT
ADD 1 TO SCBNXCPB; SUBTRACT 1 FROM DISK COUNT
ARE THERE FIELDS TO UPDATE
NO
YES
IS COUNT OF DATA TO BE MOVED = 0
NO
YES
IS THE PREFIX IN THIS LAST UNIT
NO
YES
ADD 1 TO THE ADDRESS
PUT PRFXTRA IN SCBSCSEG
TESTLAST
IS THE SCB CPB COUNT = 0
NO
YES
FA2-21 B4
CPBFREEA
PUT THE CPB IN THE CPB FREE POOL
RETURN
LASTTEST
FA2-15,D1
FA2-22,C2
IS THE COUNT OF DATA TO BE MOVED = 0
NO
YES
SET THE SCB UNIT COUNT = CPB WORK COUNT
SET THE SCB UNIT COUNT = 0
RETURN

Chart GA-1 BUFFER MANAGEMENT MODULE



IEDQGA
ENTER
GET THE ADDRESS OF THE LCB
IS THERE A TRANSMISSION ERROR
YES
RESETERR
TURN THE 'ERROR' FLAG OFF. TURN THE 'DLNK' SWITCH ON
EXIT TO DSPDISP
NO
IS ERB PRIORITY = SUBSEQUENT PCI REQ PRTY
YES
GA1 E1
NO
INITIAL GA-4 A3
SETRSVD
GET RESERVED DISK AND REQUEST COUNT
GA-4 F5
DCBCT
GET THE BUFFER UNIT COUNT FROM THE DCB
IS THE AVAILABLE BFR COUNT > RESERVED COUNT
GA1 H1
YES
BFRTHERE
ARE THERE ANY BUFFERS
YES
NO
NO
LINK
RESTORE THE ERB COUNT
GA1 J1
PRINSERT
GET THE ADDRESS OF THE BUFFER RETURN QCB
EXIT TO DSPPRIO
GET THE ADDRESS OF THE LAST BUFFER ASSIGNED
PUT THE INITIAL ADDRESS (GA-1,E1) IN REGISTER 14
PUT THE SETCT ADDRESS (GA-4,B5) IN REGISTER 9
SET REGISTER 8 EQUAL TO REGISTER 14
GA-4 A1
UNIT
GET THE BUFFER UNITS
IS THIS THE FIRST PCI REQUEST
NO
YES
GA2 H3
IS THE REQUEST COMPLETE
NO
BRANCH ON REGISTER 9
YES
IS IT AN APPLICATION PROGRAM REQUEST
YES
SET UP TO TPOST THE BUFFERS TO THE QCB SPECIFIED IN LCBRCQCB
NO
GA-4 A4
ASSOCIAT
BUILD CCWS IN THE BUFFERS
SET UP TO TPOST THE BUFFERS TO THE ACTIVATE QCB
POSTA
SET THE QCB ADDRESS AND THE PRIORITY
EXIT TO DSPPOST
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart GA-2 BUFFER MANAGEMENT MODULE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQGB
ENTER
GET THE ADDRESS OF THE BUFFER; GET THE NUMBER OF UNITS
GA-4 E3
NEXTONE
RETURN UNITS TO THE BUFFER UNIT POOL
RESTORE THE AVAILABLE BUFFER COUNT
E1
BFRASIGN
ARE ANY ERBS WAITING
YES
NO
REMOVE AN ERB
IS THIS THE FIRST ERB FROM DISK
NO
YES
REMOVE ONE BUFFER UNIT FROM THE BUFFER UNIT POOL
SET UP TO TPOST THE BUFFER TO THE DISK I/O QCB AND SET THE PRIORITY
SUBRACT 1 FROM THE AVAILABLE BUFFER COUNT AND FROM THE RESERVED COUNT
RB-2 A1
DSPPOSTR
TPOST THE BUFFER TO THE DISK I/O QCB
E1
NO
ARE ALL RETURNED UNITS USED
YES
ZERO
EXIT TO DSPDISP
XRSVDCT
IS THERE A TRANSMISSION ERROR
YES
NO
TURN THE 'ERROR' FLAG OFF; TURN THE 'DLNK' SWITCH ON
E1
GA-4 A3
SETRSVD
GET RESERVED DISK AND THE REQUEST COUNT
NO
IS THE AVAILABLE BFR COUNT &gt; RESERVED COUNT
YES
GA-4 F5
DCBCT
GET THE BUFFER UNIT COUNT FROM THE DCB
YES
IS THIS AN INITIAL REQUEST
GA1 E1
NO
GA-4 A1
UNIT
GET THE BUFFER UNITS
GA2 H3
RECOUNT
PUT THE RECHAIN ADDR IN REG 9 &amp; THE ASSOC3 ADDR IN REG 14
IS THE REQUEST COMPLETE
YES
NO
BRANCH ON REGISTER 14
IS THE ERB DISABLED COUNT = 0
NO
YES
ADD THE DISABLED COUNT TO THE ENABLED COUNT
BRANCH ON REGISTER 14
MORETEST
TURN THE 'DLNK' SWITCH ON
PUT THE DSPDISP EXIT ADDRESS (GA-2,J2) IN REGISTER 8
YES
ARE THE COUNTS STILL ZERO
NO
IS THE 'DLNK' SWITCH ON
YES
NO
PUT THE PRINSERT ADDRESS (GA-1,J1) IN REGISTER 8
BRANCH ON REGISTER 9

Chart GA-3 BUFFER MANAGEMENT MODULE

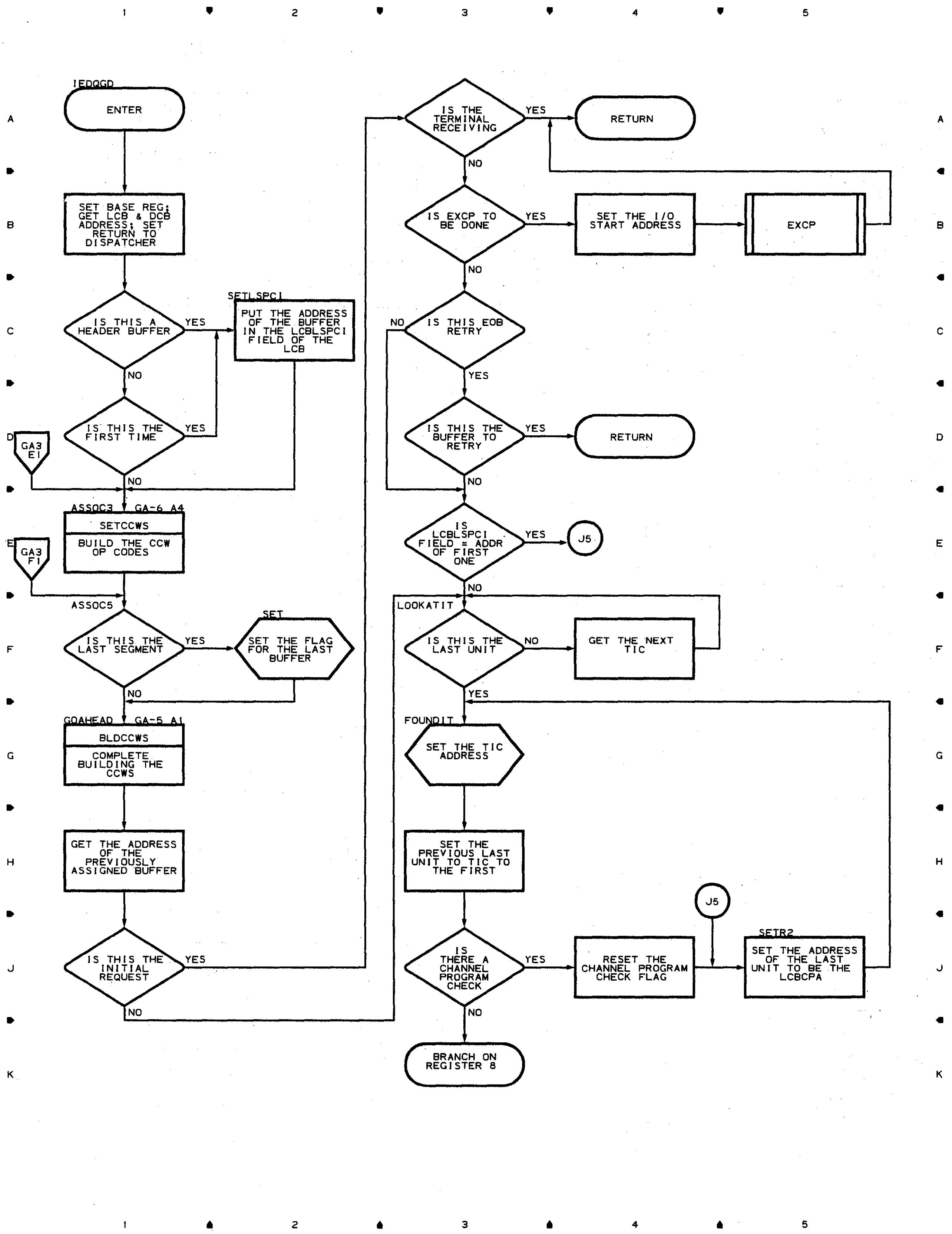

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQGD
ENTER
SET BASE REG; GET LCB & DCB ADDRESS; SET RETURN TO DISPATCHER
IS THIS A HEADER BUFFER
YES
NO
SETLSPCI
PUT THE ADDRESS OF THE BUFFER IN THE LCBLSPCI FIELD OF THE LCB
IS THIS THE FIRST TIME
YES
NO
GA3 E1
ASSOC3
GA-6 A4
SETCCWS
BUILD THE CCW OP CODES
GA3 F1
ASSOC5
IS THIS THE LAST SEGMENT
YES
NO
SET
SET THE FLAG FOR THE LAST BUFFER
GOAHEAD
GA-5 A1
BLDCCWS
COMPLETE BUILDING THE CCWS
GET THE ADDRESS OF THE PREVIOUSLY ASSIGNED BUFFER
IS THIS THE INITIAL REQUEST
YES
NO
IS THE TERMINAL RECEIVING
YES
NO
RETURN
IS EXCP TO BE DONE
YES
NO
SET THE I/O START ADDRESS
EXCP
IS THIS EOB RETRY
NO
YES
IS THIS THE BUFFER TO RETRY
YES
NO
RETURN
IS LCBLSPCI FIELD = ADDR OF FIRST ONE
YES
NO
J5
LOOKATIT
IS THIS THE LAST UNIT
NO
YES
GET THE NEXT TIC
FOUNDIT
SET THE TIC ADDRESS
SET THE PREVIOUS LAST UNIT TO TIC TO THE FIRST
IS THERE A CHANNEL PROGRAM CHECK
YES
NO
RESET THE CHANNEL PROGRAM CHECK FLAG
J5
SETR2
SET THE ADDRESS OF THE LAST UNIT TO BE THE LCBCPA
BRANCH ON REGISTER 8

Chart GA-4 BUFFER MANAGEMENT MODULE

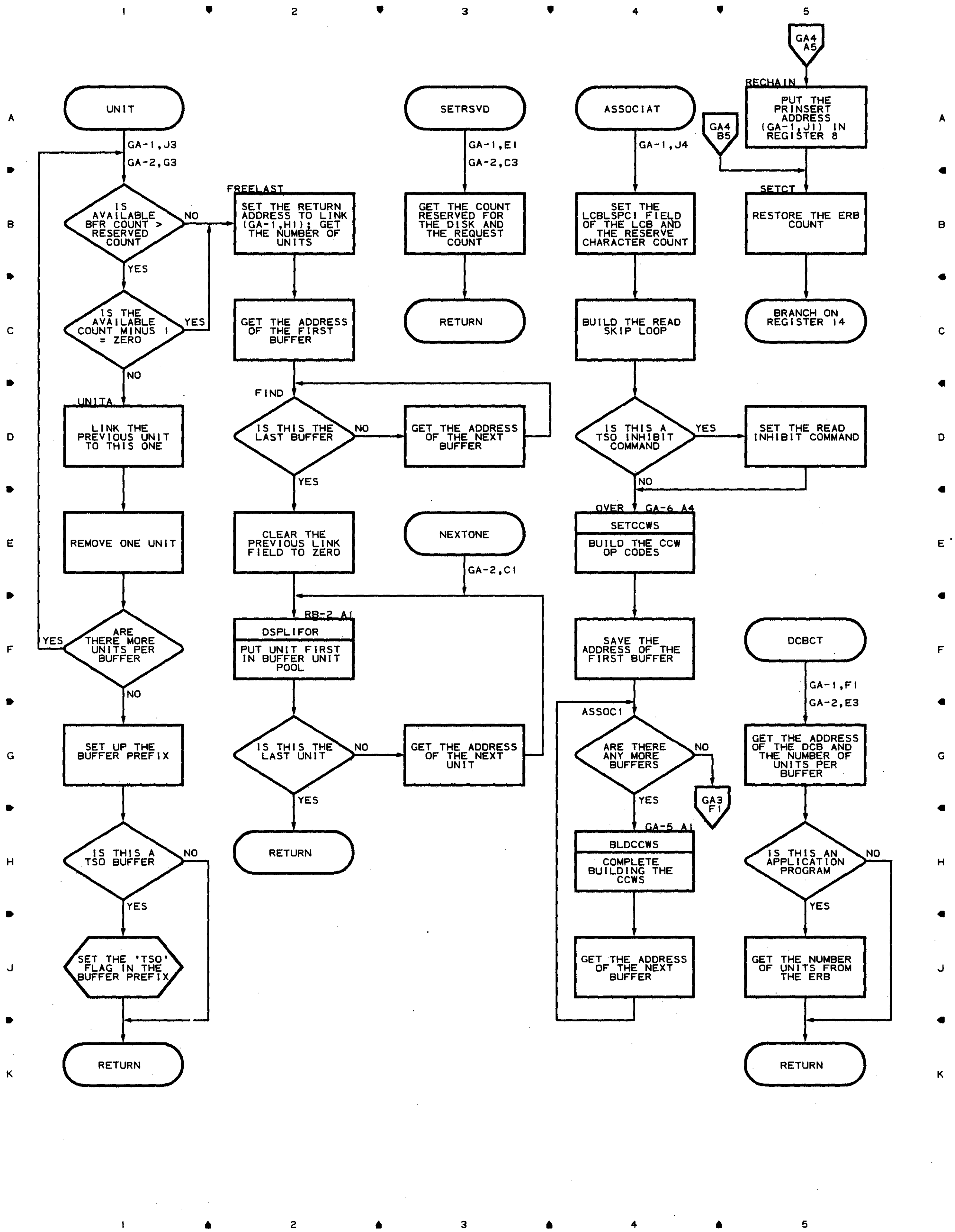
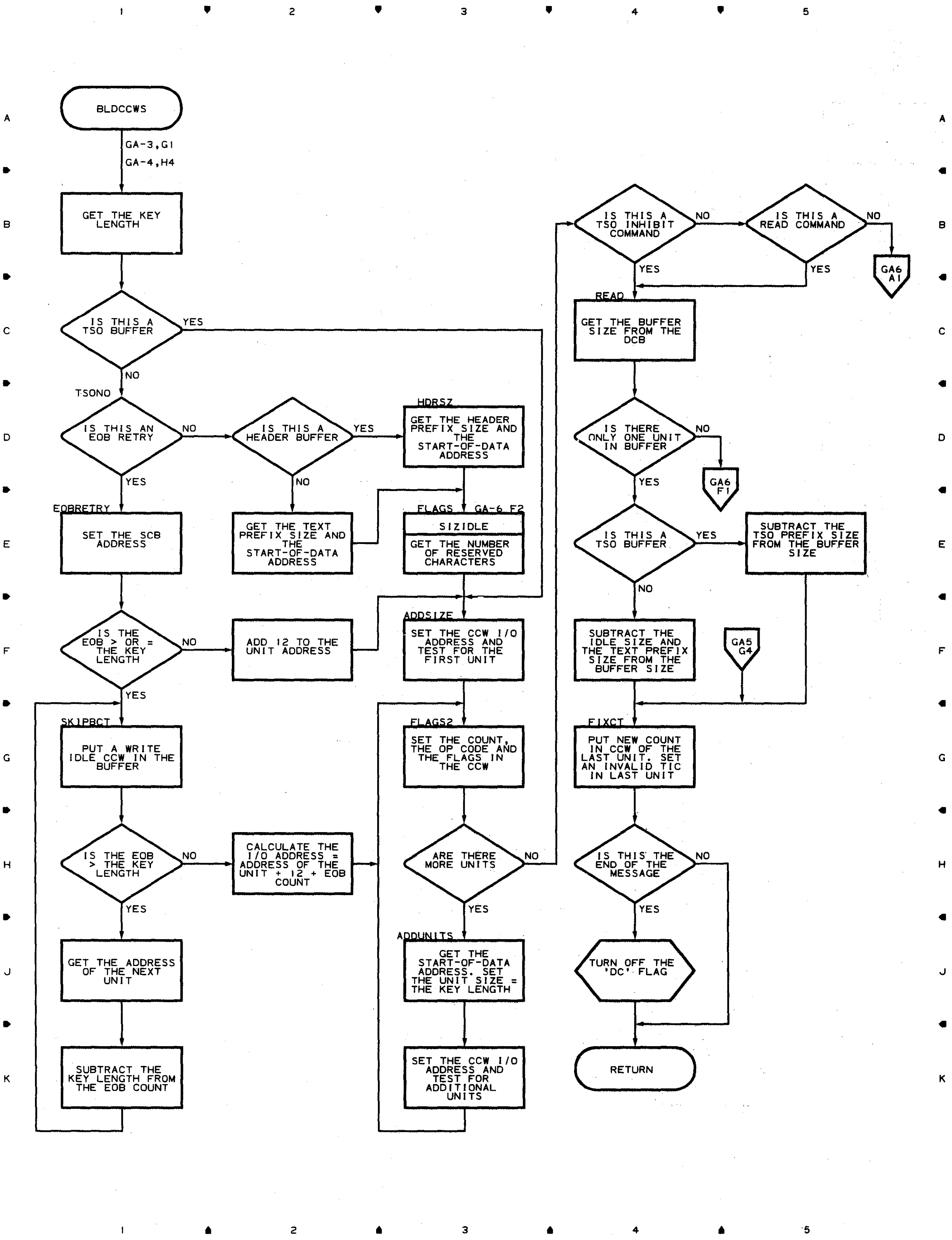
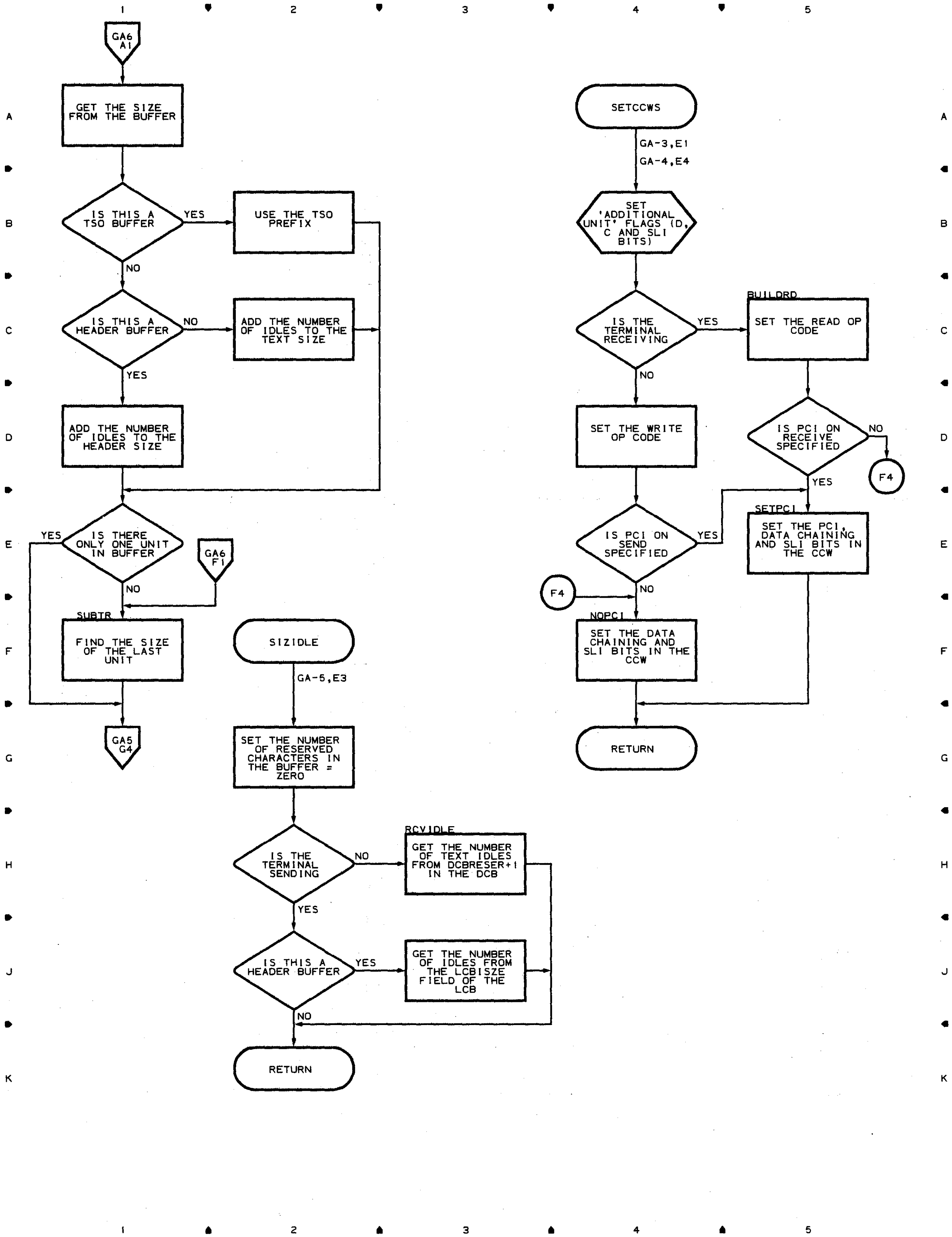

UNIT
GA-1,J3
GA-2,G3
IS AVAILABLE BFR COUNT > RESERVED COUNT
NO
YES
IS THE AVAILABLE COUNT MINUS 1 = ZERO
YES
NO
UNITA
LINK THE PREVIOUS UNIT TO THIS ONE
REMOVE ONE UNIT
ARE THERE MORE UNITS PER BUFFER
YES
NO
SET UP THE BUFFER PREFIX
IS THIS A TSO BUFFER
NO
YES
SET THE 'TSO' FLAG IN THE BUFFER PREFIX
RETURN
FREELAST
SET THE RETURN ADDRESS TO LINK (GA-1,H1); GET THE NUMBER OF UNITS
GET THE ADDRESS OF THE FIRST BUFFER
FIND
IS THIS THE LAST BUFFER
NO
YES
CLEAR THE PREVIOUS LINK FIELD TO ZERO
RB-2 A1
DSPLIFOR
PUT UNIT FIRST IN BUFFER UNIT POOL
IS THIS THE LAST UNIT
NO
YES
RETURN
SETRSVD
GA-1,E1
GA-2,C3
GET THE COUNT RESERVED FOR THE DISK AND THE REQUEST COUNT
RETURN
GET THE ADDRESS OF THE NEXT BUFFER
NEXTONE
GA-2,C1
GET THE ADDRESS OF THE NEXT UNIT
ASSOCIAT
GA-1,J4
SET THE LCBLSPCI FIELD OF THE LCB AND THE RESERVE CHARACTER COUNT
BUILD THE READ SKIP LOOP
IS THIS A TSO INHIBIT COMMAND
YES
NO
OVER
GA-6 A4
SETCCWS
BUILD THE CCW OP CODES
SAVE THE ADDRESS OF THE FIRST BUFFER
ASSOC1
ARE THERE ANY MORE BUFFERS
NO
YES
GA3 F1
GA-5 A1
BLDCCWS
COMPLETE BUILDING THE CCWS
GET THE ADDRESS OF THE NEXT BUFFER
GA4 A5
RECHAIN
PUT THE PRINSERT ADDRESS (GA-1,J1) IN REGISTER 8
GA4 B5
SETCT
RESTORE THE ERB COUNT
BRANCH ON REGISTER 14
SET THE READ INHIBIT COMMAND
DCBCT
GA-1,F1
GA-2,E3
GET THE ADDRESS OF THE DCB AND THE NUMBER OF UNITS PER BUFFER
IS THIS AN APPLICATION PROGRAM
NO
YES
GET THE NUMBER OF UNITS FROM THE ERB
RETURN
1 2 3 4 5
A B C D E F G H J K

Chart GA-5 BUFFER MANAGEMENT MODULE



BLDCCWS
GA-3,G1
GA-4,H4
GET THE KEY LENGTH
IS THIS A TSO BUFFER
YES
NO
TSONO
IS THIS AN EOB RETRY
NO
IS THIS A HEADER BUFFER
YES
HDRSZ
GET THE HEADER PREFIX SIZE AND THE START-OF-DATA ADDRESS
NO
GET THE TEXT PREFIX SIZE AND THE START-OF-DATA ADDRESS
YES
EOBRETRY
SET THE SCB ADDRESS
FLAGS GA-6 F2
SIZIDLE
GET THE NUMBER OF RESERVED CHARACTERS
IS THE EOB > OR = THE KEY LENGTH
NO
ADD 12 TO THE UNIT ADDRESS
ADDSIZE
SET THE CCW I/O ADDRESS AND TEST FOR THE FIRST UNIT
YES
SKIPBCT
PUT A WRITE IDLE CCW IN THE BUFFER
FLAGS2
SET THE COUNT, THE OP CODE AND THE FLAGS IN THE CCW
IS THE EOB > THE KEY LENGTH
NO
CALCULATE THE I/O ADDRESS = ADDRESS OF THE UNIT + 12 + EOB COUNT
ARE THERE MORE UNITS
NO
YES
GET THE ADDRESS OF THE NEXT UNIT
ADDUNITS
GET THE START-OF-DATA ADDRESS. SET THE UNIT SIZE = THE KEY LENGTH
SUBTRACT THE KEY LENGTH FROM THE EOB COUNT
SET THE CCW I/O ADDRESS AND TEST FOR ADDITIONAL UNITS
IS THIS A TSO INHIBIT COMMAND
NO
IS THIS A READ COMMAND
NO
GA6 A1
YES
YES
READ
GET THE BUFFER SIZE FROM THE DCB
IS THERE ONLY ONE UNIT IN BUFFER
NO
GA6 F1
YES
IS THIS A TSO BUFFER
YES
SUBTRACT THE TSO PREFIX SIZE FROM THE BUFFER SIZE
NO
SUBTRACT THE IDLE SIZE AND THE TEXT PREFIX SIZE FROM THE BUFFER SIZE
GA5 G4
FIXCT
PUT NEW COUNT IN CCW OF THE LAST UNIT. SET AN INVALID TIC IN LAST UNIT
IS THIS THE END OF THE MESSAGE
NO
YES
TURN OFF THE 'DC' FLAG
RETURN
1 2 3 4 5
A B C D E F G H J K

Chart GA-6 BUFFER MANAGEMENT MODULE



GA6 A1
GET THE SIZE FROM THE BUFFER
IS THIS A TSO BUFFER
YES
USE THE TSO PREFIX
NO
IS THIS A HEADER BUFFER
NO
ADD THE NUMBER OF IDLES TO THE TEXT SIZE
YES
ADD THE NUMBER OF IDLES TO THE HEADER SIZE
IS THERE ONLY ONE UNIT IN BUFFER
YES
NO
GA6 F1
SUBTR
FIND THE SIZE OF THE LAST UNIT
GA5 G4
SIZIDLE
GA-5,E3
SET THE NUMBER OF RESERVED CHARACTERS IN THE BUFFER = ZERO
IS THE TERMINAL SENDING
NO
RCVIDLE
GET THE NUMBER OF TEXT IDLES FROM DCBRESER+1 IN THE DCB
YES
IS THIS A HEADER BUFFER
YES
GET THE NUMBER OF IDLES FROM THE LCBISZE FIELD OF THE LCB
NO
RETURN
SETCCWS
GA-3,E1
GA-4,E4
SET 'ADDITIONAL UNIT' FLAGS (D, C AND SLI BITS)
IS THE TERMINAL RECEIVING
YES
BUILDRD
SET THE READ OP CODE
NO
SET THE WRITE OP CODE
IS PCI ON RECEIVE SPECIFIED
NO
F4
YES
SETPCI
SET THE PCI, DATA CHAINING AND SLI BITS IN THE CCW
IS PCI ON SEND SPECIFIED
YES
NO
F4
NOPCI
SET THE DATA CHAINING AND SLI BITS IN THE CCW
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart GT-1 TRANSPARENT TRANSMISSION CCW BUILDING ROUTINE

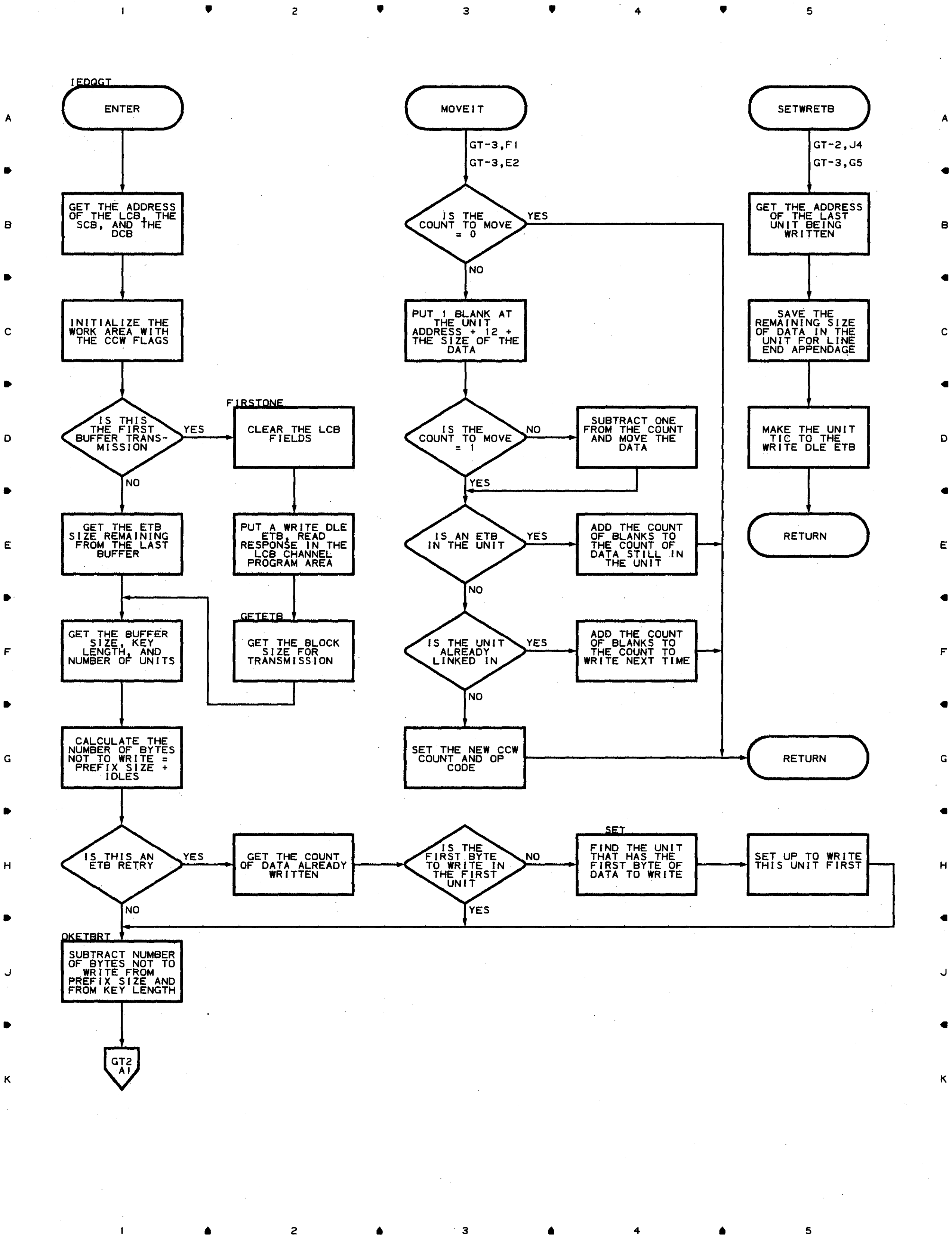

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQGT
ENTER
GET THE ADDRESS OF THE LCB, THE SCB, AND THE DCB
INITIALIZE THE WORK AREA WITH THE CCW FLAGS
IS THIS THE FIRST BUFFER TRANS-MISSION
YES
NO
FIRSTONE
CLEAR THE LCB FIELDS
PUT A WRITE DLE ETB, READ RESPONSE IN THE LCB CHANNEL PROGRAM AREA
GET THE ETB SIZE REMAINING FROM THE LAST BUFFER
GETETB
GET THE BLOCK SIZE FOR TRANSMISSION
GET THE BUFFER SIZE, KEY LENGTH, AND NUMBER OF UNITS
CALCULATE THE NUMBER OF BYTES NOT TO WRITE = PREFIX SIZE + IDLES
IS THIS AN ETB RETRY
YES
NO
GET THE COUNT OF DATA ALREADY WRITTEN
OKETBRT
SUBTRACT NUMBER OF BYTES NOT TO WRITE FROM PREFIX SIZE AND FROM KEY LENGTH
GT2 A1
MOVEIT
GT-3,F1
GT-3,E2
IS THE COUNT TO MOVE = 0
YES
NO
PUT 1 BLANK AT THE UNIT ADDRESS + 12 + THE SIZE OF THE DATA
IS THE COUNT TO MOVE = 1
NO
YES
SUBTRACT ONE FROM THE COUNT AND MOVE THE DATA
IS AN ETB IN THE UNIT
YES
NO
ADD THE COUNT OF BLANKS TO THE COUNT OF DATA STILL IN THE UNIT
IS THE UNIT ALREADY LINKED IN
YES
NO
ADD THE COUNT OF BLANKS TO THE COUNT TO WRITE NEXT TIME
SET THE NEW CCW COUNT AND OP CODE
RETURN
IS THE FIRST BYTE TO WRITE IN THE FIRST UNIT
NO
YES
SET
FIND THE UNIT THAT HAS THE FIRST BYTE OF DATA TO WRITE
SET UP TO WRITE THIS UNIT FIRST
SETWRETB
GT-2,J4
GT-3,G5
GET THE ADDRESS OF THE LAST UNIT BEING WRITTEN
SAVE THE REMAINING SIZE OF DATA IN THE UNIT FOR LINE END APPENDAGE
MAKE THE UNIT TIC TO THE WRITE DLE ETB
RETURN

Chart GT-2 TRANSPARENT TRANSMISSION CCW BUILDING ROUTINE

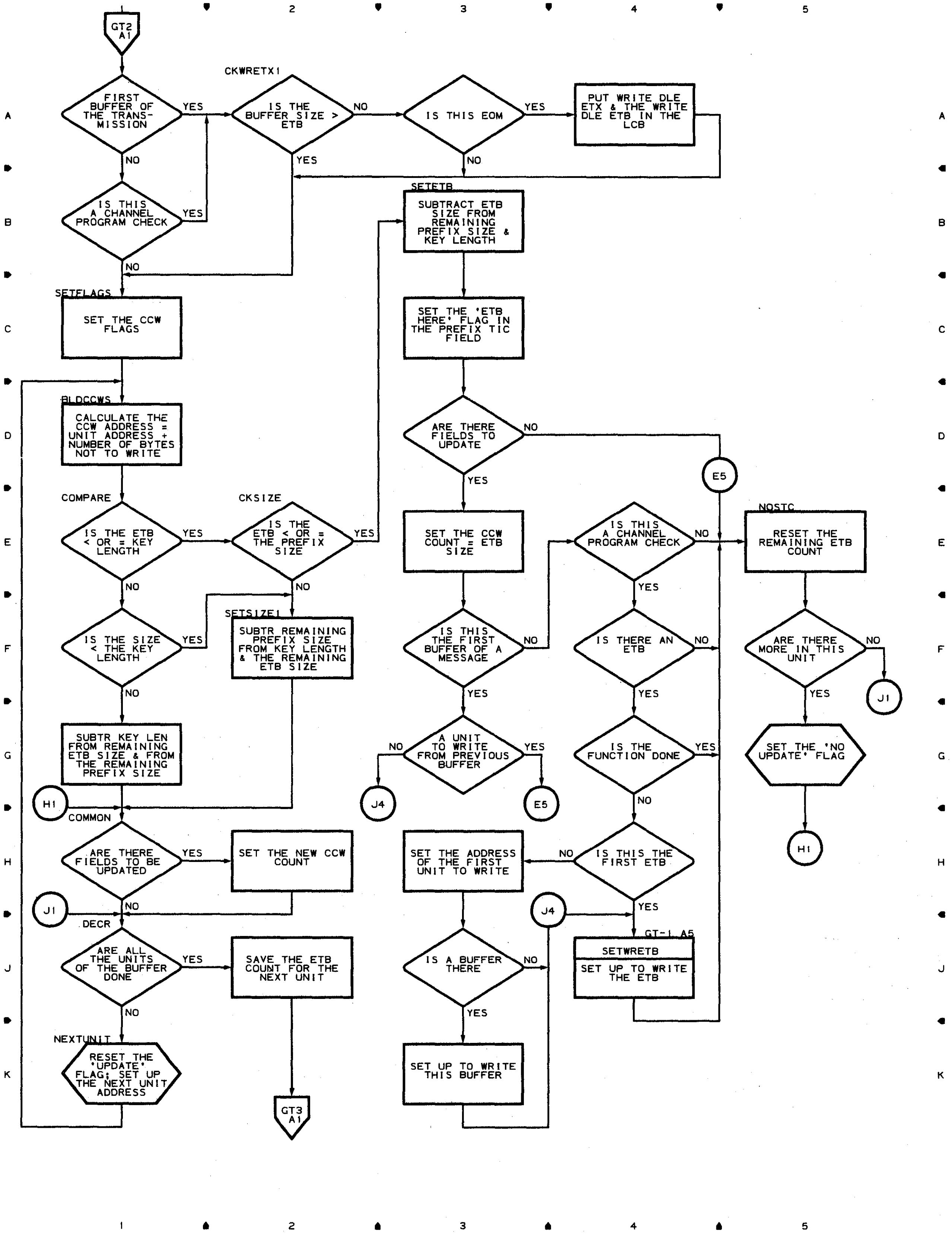

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
GT2 A1
FIRST BUFFER OF THE TRANS-MISSION
YES
NO
CKWRETX1
IS THE BUFFER SIZE > ETB
NO
YES
IS THIS EOM
YES
NO
PUT WRITE DLE ETX & THE WRITE DLE ETB IN THE LCB
IS THIS A CHANNEL PROGRAM CHECK
YES
NO
SETETB
SUBTRACT ETB SIZE FROM REMAINING PREFIX SIZE & KEY LENGTH
SETFLAGS
SET THE CCW FLAGS
SET THE 'ETB HERE' FLAG IN THE PREFIX TIC FIELD
BLDCCWS
CALCULATE THE CCW ADDRESS = UNIT ADDRESS + NUMBER OF BYTES NOT TO WRITE
ARE THERE FIELDS TO UPDATE
NO
YES
E5
COMPARE
IS THE ETB < OR = KEY LENGTH
YES
NO
CKSIZE
IS THE ETB < OR = THE PREFIX SIZE
YES
NO
SET THE CCW COUNT = ETB SIZE
IS THIS A CHANNEL PROGRAM CHECK
NO
YES
NOSTC
RESET THE REMAINING ETB COUNT
IS THE SIZE < THE KEY LENGTH
YES
NO
SETSIZE1
SUBTR REMAINING PREFIX SIZE FROM KEY LENGTH & THE REMAINING ETB SIZE
IS THIS THE FIRST BUFFER OF A MESSAGE
NO
YES
IS THERE AN ETB
NO
YES
ARE THERE MORE IN THIS UNIT
NO
YES
J1
SUBTR KEY LEN FROM REMAINING ETB SIZE & FROM THE REMAINING PREFIX SIZE
A UNIT TO WRITE FROM PREVIOUS BUFFER
NO
YES
J4
E5
IS THE FUNCTION DONE
YES
NO
SET THE 'NO UPDATE' FLAG
H1
COMMON
H1
ARE THERE FIELDS TO BE UPDATED
YES
NO
SET THE NEW CCW COUNT
SET THE ADDRESS OF THE FIRST UNIT TO WRITE
IS THIS THE FIRST ETB
NO
YES
J1
DECR
J4
GT-1 A5
SETWRETB
SET UP TO WRITE THE ETB
ARE ALL THE UNITS OF THE BUFFER DONE
YES
NO
SAVE THE ETB COUNT FOR THE NEXT UNIT
IS A BUFFER THERE
NO
YES
NEXTUNIT
RESET THE 'UPDATE' FLAG; SET UP THE NEXT UNIT ADDRESS
SET UP TO WRITE THIS BUFFER
GT3 A1

Chart GT-3 TRANSPARENT TRANSMISSION CCW BUILDING ROUTINE

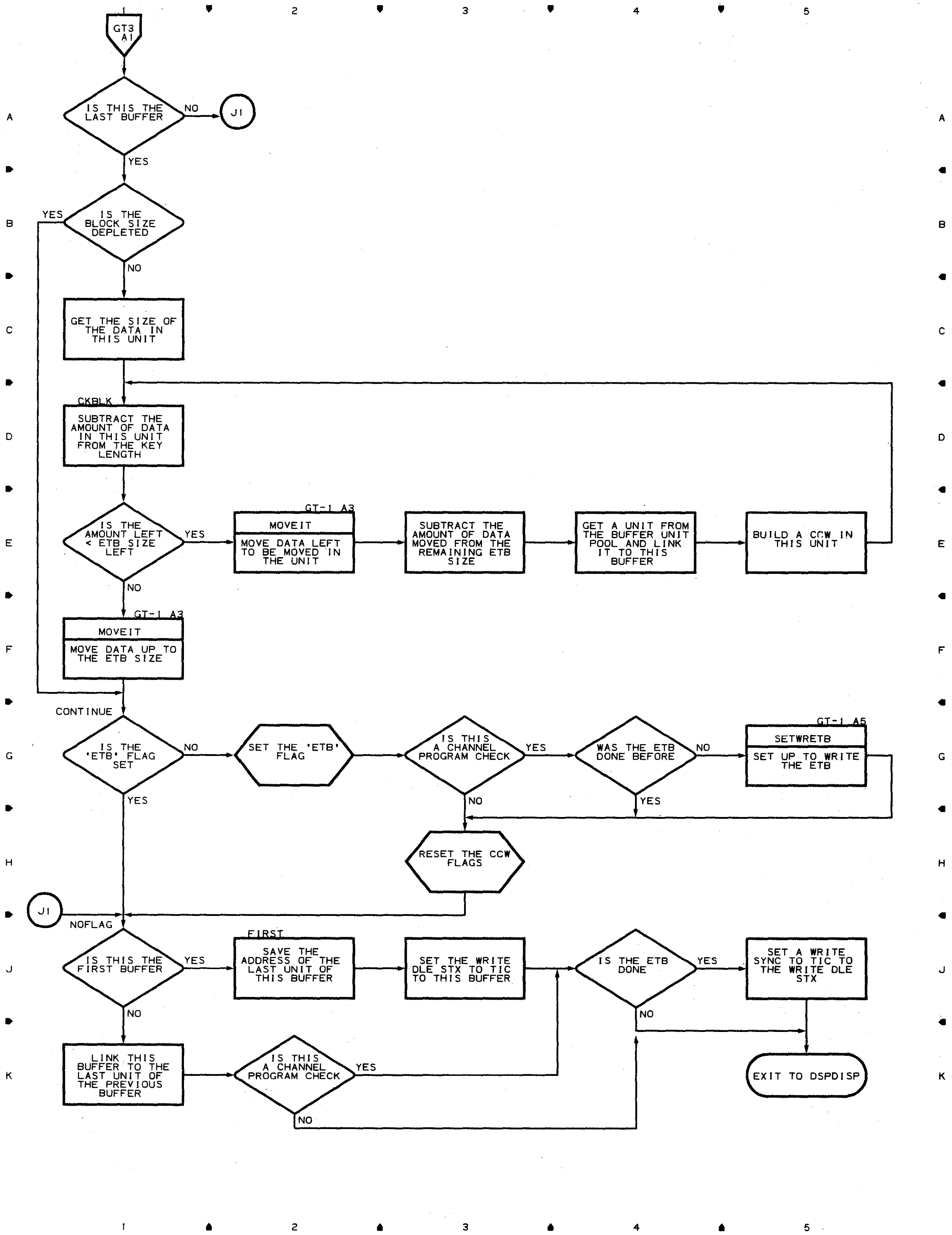

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
GT3 A1
IS THIS THE LAST BUFFER
NO
J1
YES
IS THE BLOCK SIZE DEPLETED
YES
NO
GET THE SIZE OF THE DATA IN THIS UNIT
CKBLK
SUBTRACT THE AMOUNT OF DATA IN THIS UNIT FROM THE KEY LENGTH
IS THE AMOUNT LEFT < ETB SIZE LEFT
YES
GT-1 A3
MOVEIT
MOVE DATA LEFT TO BE MOVED IN THE UNIT
SUBTRACT THE AMOUNT OF DATA MOVED FROM THE REMAINING ETB SIZE
GET A UNIT FROM THE BUFFER UNIT POOL AND LINK IT TO THIS BUFFER
BUILD A CCW IN THIS UNIT
NO
GT-1 A3
MOVEIT
MOVE DATA UP TO THE ETB SIZE
CONTINUE
IS THE 'ETB' FLAG SET
NO
SET THE 'ETB' FLAG
IS THIS A CHANNEL PROGRAM CHECK
YES
WAS THE ETB DONE BEFORE
NO
GT-1 A5
SETWRETB
SET UP TO WRITE THE ETB
YES
NO
YES
RESET THE CCW FLAGS
J1
NOFLAG
IS THIS THE FIRST BUFFER
YES
FIRST
SAVE THE ADDRESS OF THE LAST UNIT OF THIS BUFFER
SET THE WRITE DLE STX TO TIC TO THIS BUFFER
IS THE ETB DONE
YES
SET A WRITE SYNC TO TIC TO THE WRITE DLE STX
NO
NO
LINK THIS BUFFER TO THE LAST UNIT OF THE PREVIOUS BUFFER
IS THIS A CHANNEL PROGRAM CHECK
YES
NO
EXIT TO DSPDISP

Chart HG-1 TIME DELAY SUBTASK

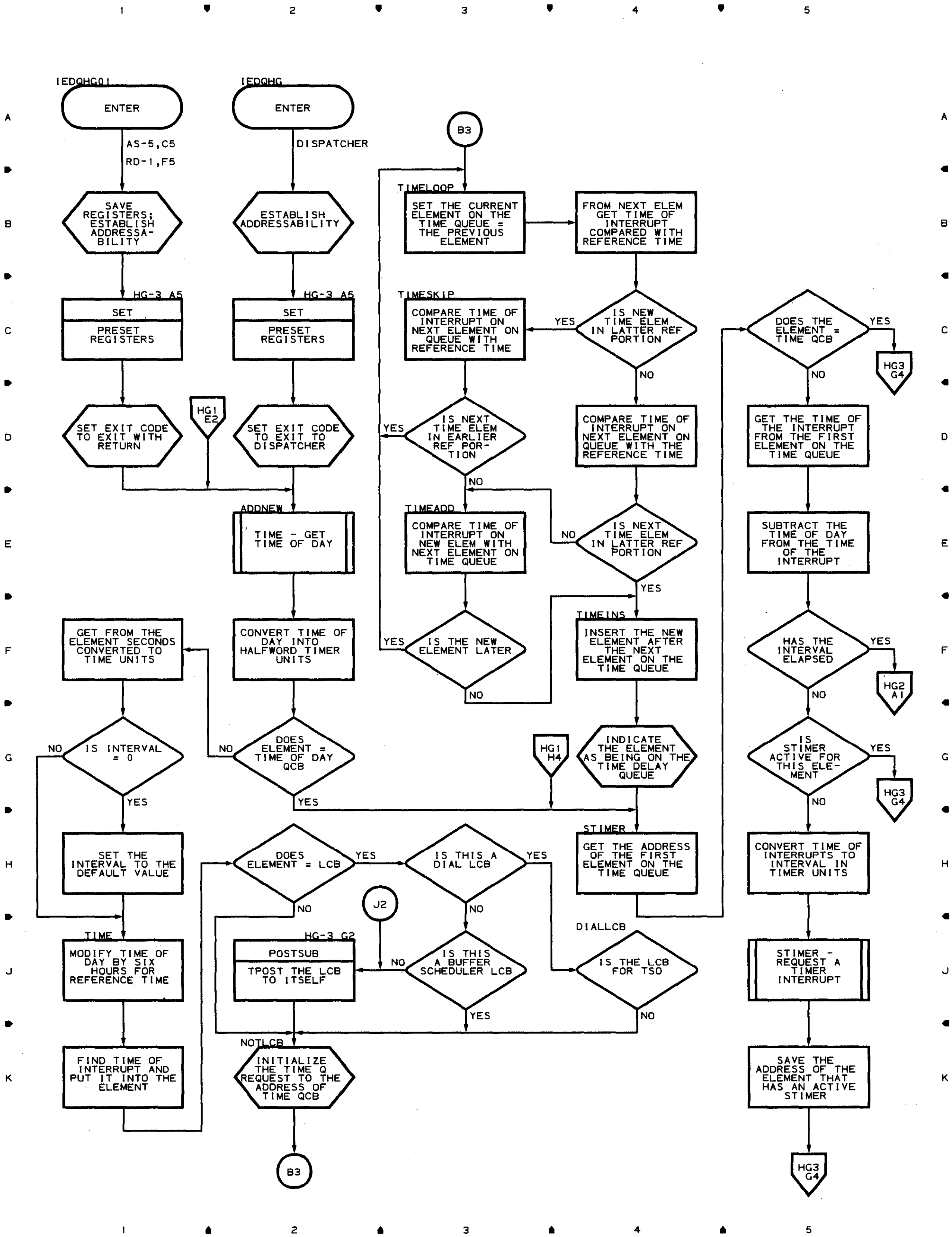

IEDQHG01
ENTER
AS-5,C5
RD-1,F5
SAVE REGISTERS; ESTABLISH ADDRESSA-BILITY
HG-3 A5
SET
PRESET REGISTERS
SET EXIT CODE TO EXIT WITH RETURN
HG1 E2
IEDQHG
ENTER
DISPATCHER
ESTABLISH ADDRESSABILITY
HG-3 A5
SET
PRESET REGISTERS
SET EXIT CODE TO EXIT TO DISPATCHER
ADDNEW
TIME - GET TIME OF DAY
CONVERT TIME OF DAY INTO HALFWORD TIMER UNITS
DOES ELEMENT = TIME OF DAY QCB
NO
YES
GET FROM THE ELEMENT SECONDS CONVERTED TO TIME UNITS
IS INTERVAL = 0
NO
YES
SET THE INTERVAL TO THE DEFAULT VALUE
TIME
MODIFY TIME OF DAY BY SIX HOURS FOR REFERENCE TIME
FIND TIME OF INTERRUPT AND PUT IT INTO THE ELEMENT
DOES ELEMENT = LCB
YES
NO
HG-3 G2
POSTSUB
TPOST THE LCB TO ITSELF
NOTLCB
INITIALIZE THE TIME Q REQUEST TO THE ADDRESS OF TIME QCB
B3
B3
TIMELOOP
SET THE CURRENT ELEMENT ON THE TIME QUEUE = THE PREVIOUS ELEMENT
TIMESKIP
COMPARE TIME OF INTERRUPT ON NEXT ELEMENT ON QUEUE WITH REFERENCE TIME
IS NEXT TIME ELEM IN EARLIER REF POR-TION
YES
NO
TIMEADD
COMPARE TIME OF INTERRUPT ON NEW ELEM WITH NEXT ELEMENT ON TIME QUEUE
IS THE NEW ELEMENT LATER
YES
NO
IS THIS A DIAL LCB
YES
NO
J2
IS THIS A BUFFER SCHEDULER LCB
NO
YES
FROM NEXT ELEM GET TIME OF INTERRUPT COMPARED WITH REFERENCE TIME
IS NEW TIME ELEM IN LATTER REF PORTION
YES
NO
COMPARE TIME OF INTERRUPT ON NEXT ELEMENT ON QUEUE WITH THE REFERENCE TIME
IS NEXT TIME ELEM IN LATTER REF PORTION
NO
YES
TIMEINS
INSERT THE NEW ELEMENT AFTER THE NEXT ELEMENT ON THE TIME QUEUE
INDICATE THE ELEMENT AS BEING ON THE TIME DELAY QUEUE
HG1 H4
STIMER
GET THE ADDRESS OF THE FIRST ELEMENT ON THE TIME QUEUE
DIALLCB
IS THE LCB FOR TSO
NO
DOES THE ELEMENT = TIME QCB
YES
NO
HG3 G4
GET THE TIME OF THE INTERRUPT FROM THE FIRST ELEMENT ON THE TIME QUEUE
SUBTRACT THE TIME OF DAY FROM THE TIME OF THE INTERRUPT
HAS THE INTERVAL ELAPSED
YES
NO
HG2 A1
IS STIMER ACTIVE FOR THIS ELE-MENT
YES
NO
HG3 G4
CONVERT TIME OF INTERRUPTS TO INTERVAL IN TIMER UNITS
STIMER - REQUEST A TIMER INTERRUPT
SAVE THE ADDRESS OF THE ELEMENT THAT HAS AN ACTIVE STIMER
HG3 G4
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart HG-2 TIME DELAY SUBTASK

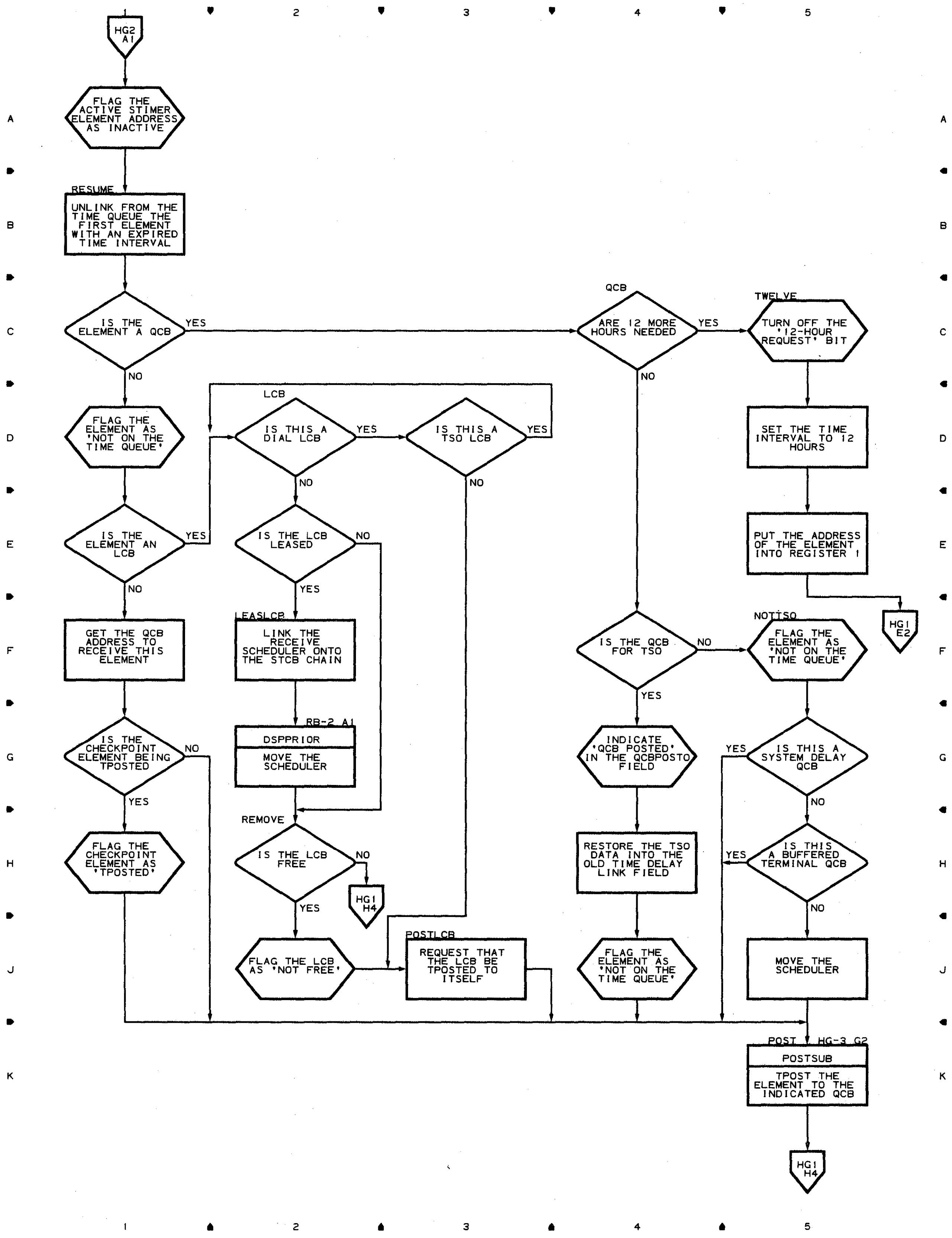
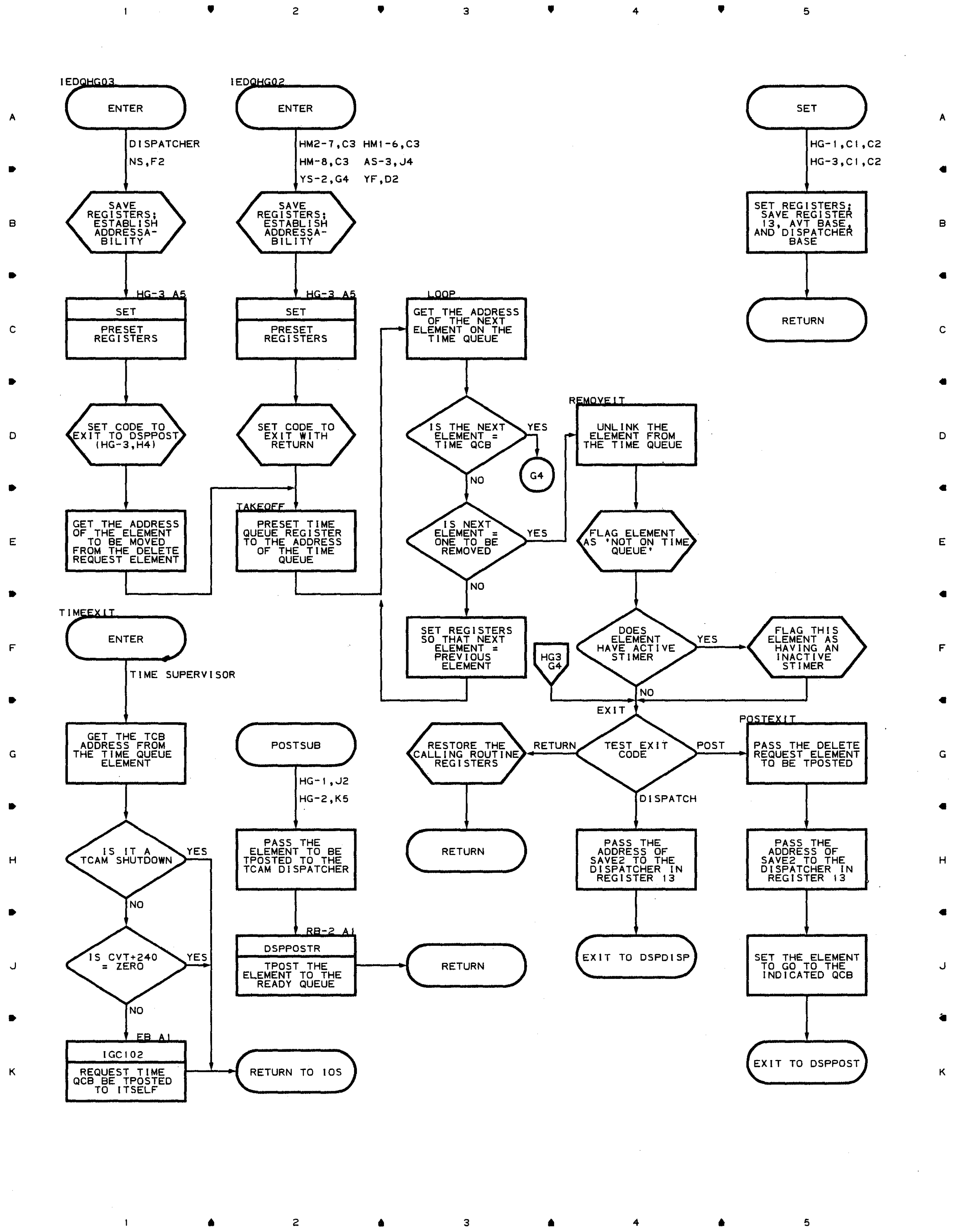

1
2
3
4
5
HG2 A1
FLAG THE ACTIVE STIMER ELEMENT ADDRESS AS INACTIVE
RESUME
UNLINK FROM THE TIME QUEUE THE FIRST ELEMENT WITH AN EXPIRED TIME INTERVAL
IS THE ELEMENT A QCB
YES
NO
QCB
ARE 12 MORE HOURS NEEDED
YES
NO
TWELVE
TURN OFF THE '12-HOUR REQUEST' BIT
SET THE TIME INTERVAL TO 12 HOURS
PUT THE ADDRESS OF THE ELEMENT INTO REGISTER 1
HG1 E2
FLAG THE ELEMENT AS 'NOT ON THE TIME QUEUE'
IS THE ELEMENT AN LCB
YES
NO
LCB
IS THIS A DIAL LCB
YES
NO
IS THIS A TSO LCB
YES
NO
IS THE LCB LEASED
NO
YES
LEASLCB
LINK THE RECEIVE SCHEDULER ONTO THE STCB CHAIN
RB-2 A1
DSPPRIOR
MOVE THE SCHEDULER
REMOVE
IS THE LCB FREE
NO
YES
HG1 H4
FLAG THE LCB AS 'NOT FREE'
POSTLCB
REQUEST THAT THE LCB BE TPOSTED TO ITSELF
GET THE QCB ADDRESS TO RECEIVE THIS ELEMENT
IS THE CHECKPOINT ELEMENT BEING TPOSTED
NO
YES
FLAG THE CHECKPOINT ELEMENT AS 'TPOSTED'
IS THE QCB FOR TSO
NO
YES
NOTTSO
FLAG THE ELEMENT AS 'NOT ON THE TIME QUEUE'
INDICATE 'QCB POSTED' IN THE QCBPOSTO FIELD
RESTORE THE TSO DATA INTO THE OLD TIME DELAY LINK FIELD
FLAG THE ELEMENT AS 'NOT ON THE TIME QUEUE'
IS THIS A SYSTEM DELAY QCB
YES
NO
IS THIS A BUFFERED TERMINAL QCB
YES
NO
MOVE THE SCHEDULER
POST
HG-3 G2
POSTSUB
TPOST THE ELEMENT TO THE INDICATED QCB
HG1 H4
A
B
C
D
E
F
G
H
J
K

Chart HG-3 TIME DELAY SUBTASK



IEDQHG03
ENTER
DISPATCHER
NS,F2
SAVE REGISTERS; ESTABLISH ADDRESSA-BILITY
HG-3 A5
SET
PRESET REGISTERS
SET CODE TO EXIT TO DSPPOST (HG-3,H4)
GET THE ADDRESS OF THE ELEMENT TO BE MOVED FROM THE DELETE REQUEST ELEMENT
TIMEEXIT
ENTER
TIME SUPERVISOR
GET THE TCB ADDRESS FROM THE TIME QUEUE ELEMENT
IS IT A TCAM SHUTDOWN
YES
NO
IS CVT+240 = ZERO
YES
NO
EB A1
IGC102
REQUEST TIME QCB BE TPOSTED TO ITSELF
IEDQHG02
ENTER
HM2-7,C3 HM1-6,C3
HM-8,C3 AS-3,J4
YS-2,G4 YF,D2
SAVE REGISTERS; ESTABLISH ADDRESSA-BILITY
HG-3 A5
SET
PRESET REGISTERS
SET CODE TO EXIT WITH RETURN
TAKEOFF
PRESET TIME QUEUE REGISTER TO THE ADDRESS OF THE TIME QUEUE
POSTSUB
HG-1,J2
HG-2,K5
PASS THE ELEMENT TO BE TPOSTED TO THE TCAM DISPATCHER
RB-2 A1
DSPPOSTR
TPOST THE ELEMENT TO THE READY QUEUE
RETURN
RETURN TO IOS
LOOP
GET THE ADDRESS OF THE NEXT ELEMENT ON THE TIME QUEUE
IS THE NEXT ELEMENT = TIME QCB
YES
G4
NO
IS NEXT ELEMENT = ONE TO BE REMOVED
YES
NO
SET REGISTERS SO THAT NEXT ELEMENT = PREVIOUS ELEMENT
HG3 G4
RESTORE THE CALLING ROUTINE REGISTERS
RETURN
RETURN
REMOVEIT
UNLINK THE ELEMENT FROM THE TIME QUEUE
FLAG ELEMENT AS 'NOT ON TIME QUEUE'
DOES ELEMENT HAVE ACTIVE STIMER
YES
NO
EXIT
TEST EXIT CODE
POST
DISPATCH
PASS THE ADDRESS OF SAVE2 TO THE DISPATCHER IN REGISTER 13
EXIT TO DSPDISP
SET
HG-1,C1,C2
HG-3,C1,C2
SET REGISTERS; SAVE REGISTER 13, AVT BASE, AND DISPATCHER BASE
RETURN
FLAG THIS ELEMENT AS HAVING AN INACTIVE STIMER
POSTEXIT
PASS THE DELETE REQUEST ELEMENT TO BE TPOSTED
PASS THE ADDRESS OF SAVE2 TO THE DISPATCHER IN REGISTER 13
SET THE ELEMENT TO GO TO THE INDICATED QCB
EXIT TO DSPPOST
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart HI-1 SYSTEM DELAY SUBTASK

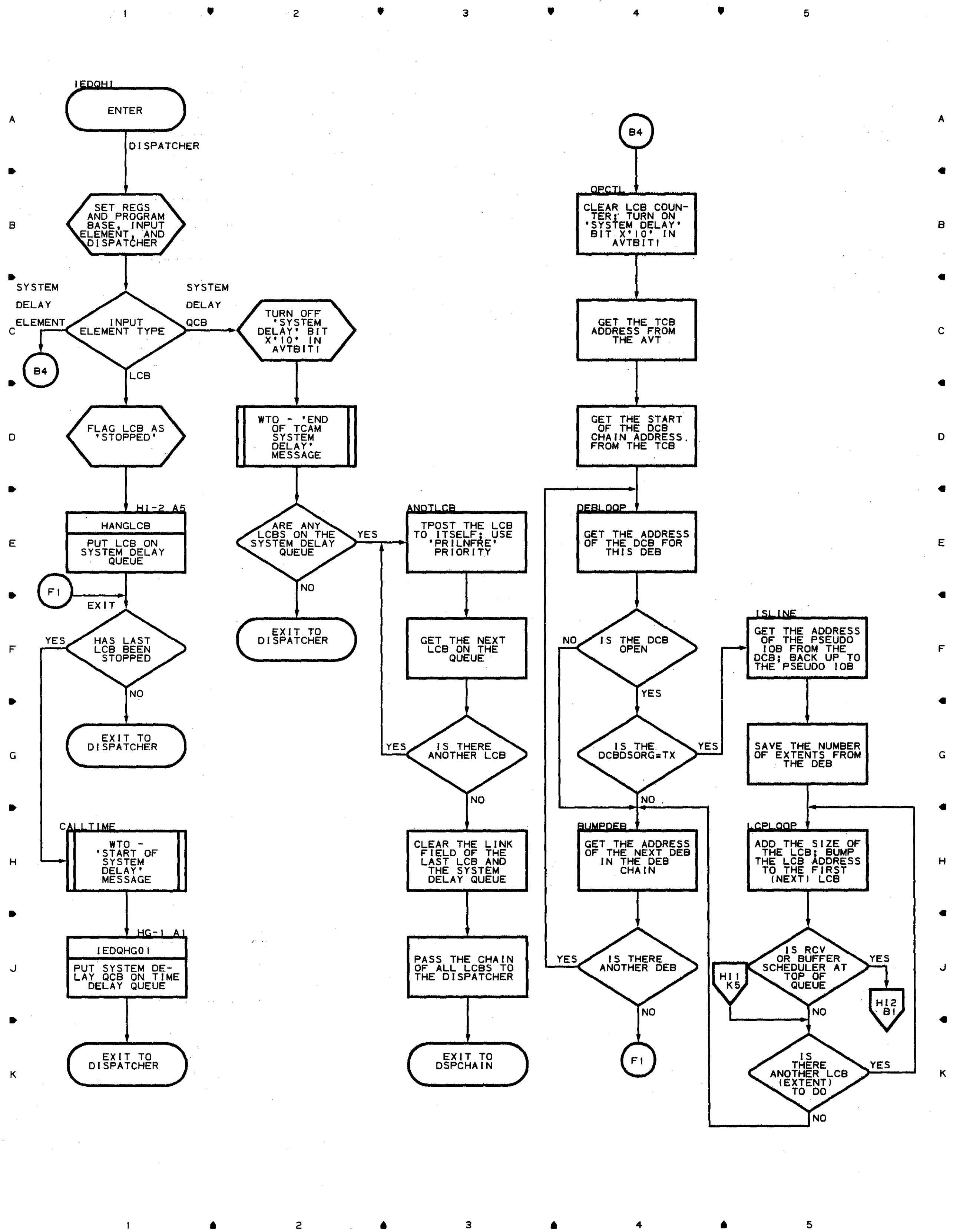

IEDQHI
ENTER
DISPATCHER
SET REGS AND PROGRAM BASE, INPUT ELEMENT, AND DISPATCHER
SYSTEM DELAY ELEMENT
INPUT ELEMENT TYPE
SYSTEM DELAY QCB
LCB
B4
TURN OFF 'SYSTEM DELAY' BIT X'10' IN AVTBIT1
FLAG LCB AS 'STOPPED'
WTO - 'END OF TCAM SYSTEM DELAY' MESSAGE
HI-2 A5
HANGLCB
PUT LCB ON SYSTEM DELAY QUEUE
ARE ANY LCBS ON THE SYSTEM DELAY QUEUE
YES
NO
EXIT TO DISPATCHER
F1
EXIT
HAS LAST LCB BEEN STOPPED
YES
NO
EXIT TO DISPATCHER
CALLTIME
WTO - 'START OF SYSTEM DELAY' MESSAGE
HG-1 A1
IEDQHG01
PUT SYSTEM DELAY QCB ON TIME DELAY QUEUE
EXIT TO DISPATCHER
ANOTLCB
TPOST THE LCB TO ITSELF; USE 'PRILNFRE' PRIORITY
GET THE NEXT LCB ON THE QUEUE
IS THERE ANOTHER LCB
YES
NO
CLEAR THE LINK FIELD OF THE LAST LCB AND THE SYSTEM DELAY QUEUE
PASS THE CHAIN OF ALL LCBS TO THE DISPATCHER
EXIT TO DSPCHAIN
B4
OPCTL
CLEAR LCB COUNTER; TURN ON 'SYSTEM DELAY' BIT X'10' IN AVTBIT1
GET THE TCB ADDRESS FROM THE AVT
GET THE START OF THE DCB CHAIN ADDRESS FROM THE TCB
DEBLOOP
GET THE ADDRESS OF THE DCB FOR THIS DEB
IS THE DCB OPEN
NO
YES
IS THE DCBDSORG=TX
YES
NO
BUMPDEB
GET THE ADDRESS OF THE NEXT DEB IN THE DEB CHAIN
IS THERE ANOTHER DEB
YES
NO
F1
ISLINE
GET THE ADDRESS OF THE PSEUDO IOB FROM THE DCB; BACK UP TO THE PSEUDO IOB
SAVE THE NUMBER OF EXTENTS FROM THE DEB
LCPLOOP
ADD THE SIZE OF THE LCB; BUMP THE LCB ADDRESS TO THE FIRST (NEXT) LCB
IS RCV OR BUFFER SCHEDULER AT TOP OF QUEUE
YES
HI2 B1
HI1 K5
NO
IS THERE ANOTHER LCB (EXTENT) TO DO
YES
NO
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart HI-2 SYSTEM DELAY SUBTASK

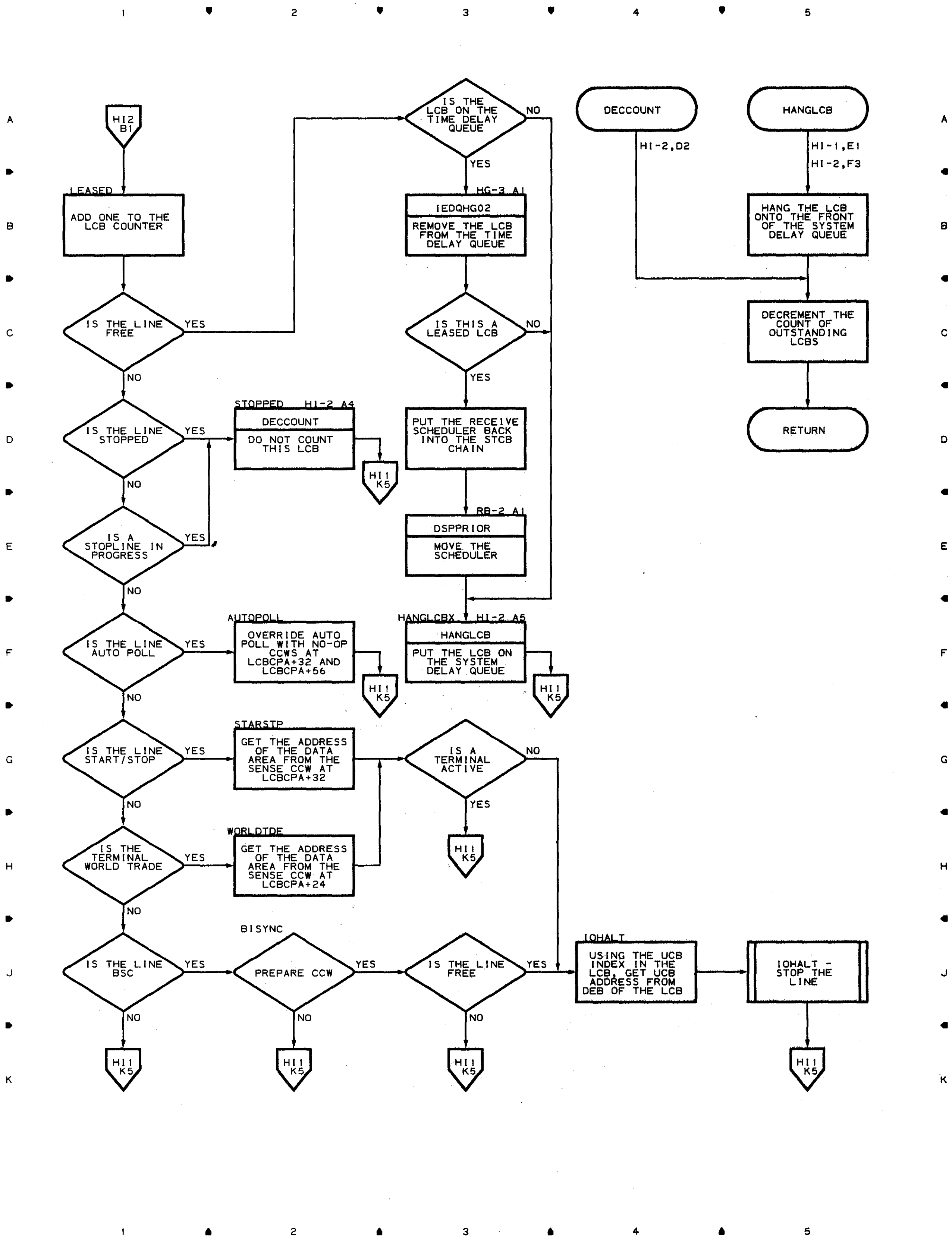
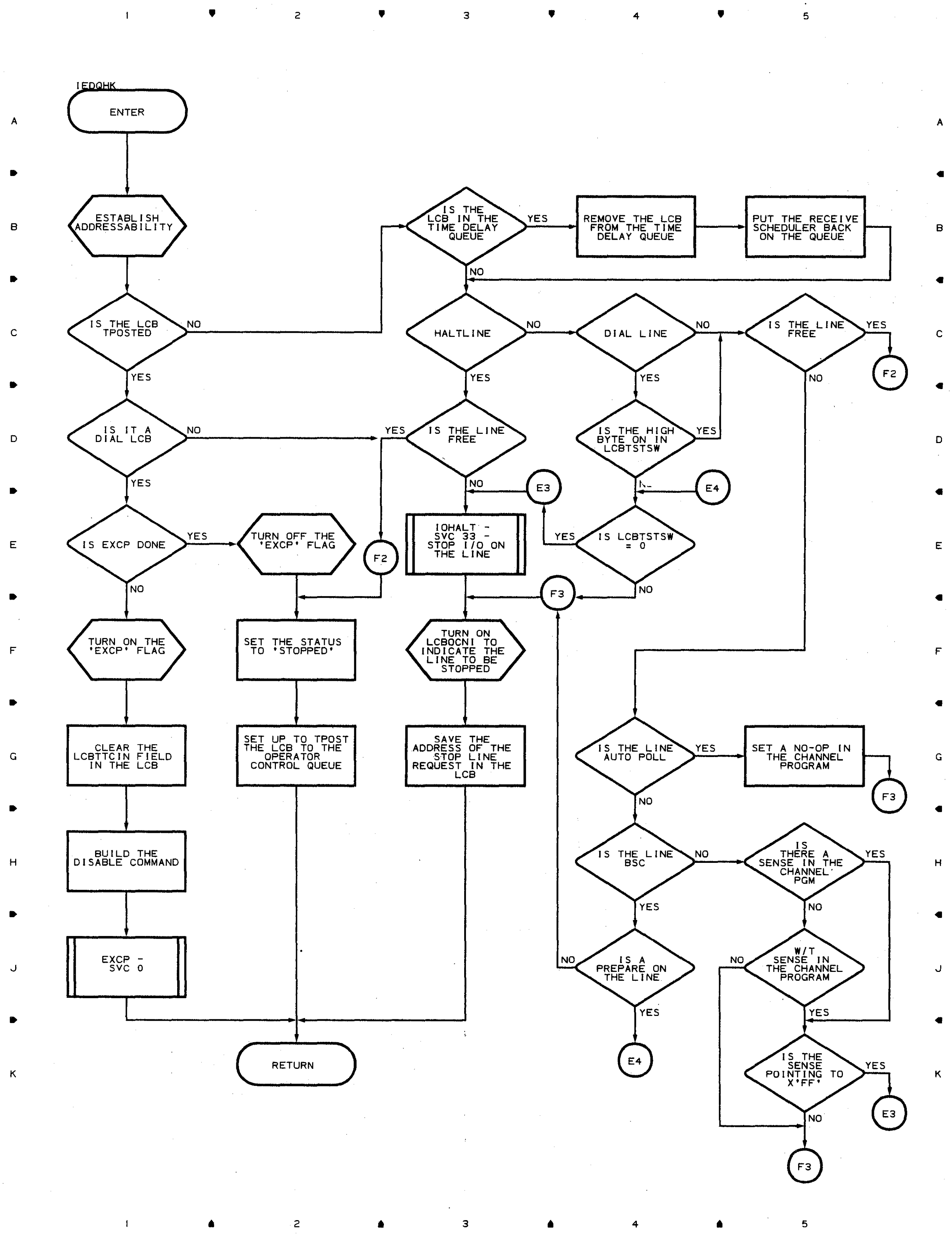

HI2 B1
LEASED
ADD ONE TO THE LCB COUNTER
IS THE LINE FREE
YES
NO
IS THE LCB ON THE TIME DELAY QUEUE
NO
YES
HG-3 A1
IEDQHG02
REMOVE THE LCB FROM THE TIME DELAY QUEUE
IS THIS A LEASED LCB
NO
YES
PUT THE RECEIVE SCHEDULER BACK INTO THE STCB CHAIN
RB-2 A1
DSPPRIOR
MOVE THE SCHEDULER
DECCOUNT
HI-2,D2
HANGLCB
HI-1,E1
HI-2,F3
HANG THE LCB ONTO THE FRONT OF THE SYSTEM DELAY QUEUE
DECREMENT THE COUNT OF OUTSTANDING LCBS
RETURN
IS THE LINE STOPPED
YES
NO
STOPPED HI-2 A4
DECCOUNT
DO NOT COUNT THIS LCB
HI1 K5
IS A STOPLINE IN PROGRESS
YES
NO
IS THE LINE AUTO POLL
YES
NO
AUTOPOLL
OVERRIDE AUTO POLL WITH NO-OP CCWS AT LCBCPA+32 AND LCBCPA+56
HI1 K5
HANGLCBX HI-2 A5
HANGLCB
PUT THE LCB ON THE SYSTEM DELAY QUEUE
HI1 K5
IS THE LINE START/STOP
YES
NO
STARSTP
GET THE ADDRESS OF THE DATA AREA FROM THE SENSE CCW AT LCBCPA+32
IS A TERMINAL ACTIVE
NO
YES
HI1 K5
IS THE TERMINAL WORLD TRADE
YES
NO
WORLDTDE
GET THE ADDRESS OF THE DATA AREA FROM THE SENSE CCW AT LCBCPA+24
IS THE LINE BSC
YES
NO
BISYNC
PREPARE CCW
YES
NO
IS THE LINE FREE
YES
NO
IOHALT
USING THE UCB INDEX IN THE LCB, GET UCB ADDRESS FROM DEB OF THE LCB
IOHALT - STOP THE LINE
HI1 K5
HI1 K5
HI1 K5
HI1 K5
HI1 K5
1 2 3 4 5
A B C D E F G H J K

Chart HK STOP LINE I/O SUBTASK



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQHK
ENTER
ESTABLISH ADDRESSABILITY
IS THE LCB IN THE TIME DELAY QUEUE
YES
REMOVE THE LCB FROM THE TIME DELAY QUEUE
PUT THE RECEIVE SCHEDULER BACK ON THE QUEUE
NO
IS THE LCB TPOSTED
NO
HALTLINE
NO
DIAL LINE
NO
IS THE LINE FREE
YES
F2
YES
YES
YES
NO
IS IT A DIAL LCB
NO
YES
IS THE LINE FREE
IS THE HIGH BYTE ON IN LCBTSTSW
YES
YES
NO
E3
E4
IS EXCP DONE
YES
TURN OFF THE 'EXCP' FLAG
F2
IOHALT - SVC 33 - STOP I/O ON THE LINE
YES
IS LCBTSTSW = 0
NO
NO
F3
TURN ON THE 'EXCP' FLAG
SET THE STATUS TO 'STOPPED'
TURN ON LCBOCNI TO INDICATE THE LINE TO BE STOPPED
CLEAR THE LCBTTCIN FIELD IN THE LCB
SET UP TO TPOST THE LCB TO THE OPERATOR CONTROL QUEUE
SAVE THE ADDRESS OF THE STOP LINE REQUEST IN THE LCB
IS THE LINE AUTO POLL
YES
SET A NO-OP IN THE CHANNEL PROGRAM
F3
NO
BUILD THE DISABLE COMMAND
IS THE LINE BSC
NO
IS THERE A SENSE IN THE CHANNEL PGM
YES
YES
NO
EXCP - SVC 0
NO
IS A PREPARE ON THE LINE
NO
W/T SENSE IN THE CHANNEL PROGRAM
YES
YES
RETURN
E4
IS THE SENSE POINTING TO X'FF'
YES
E3
NO
F3

Chart HM-1 DESTINATION SCHEDULER

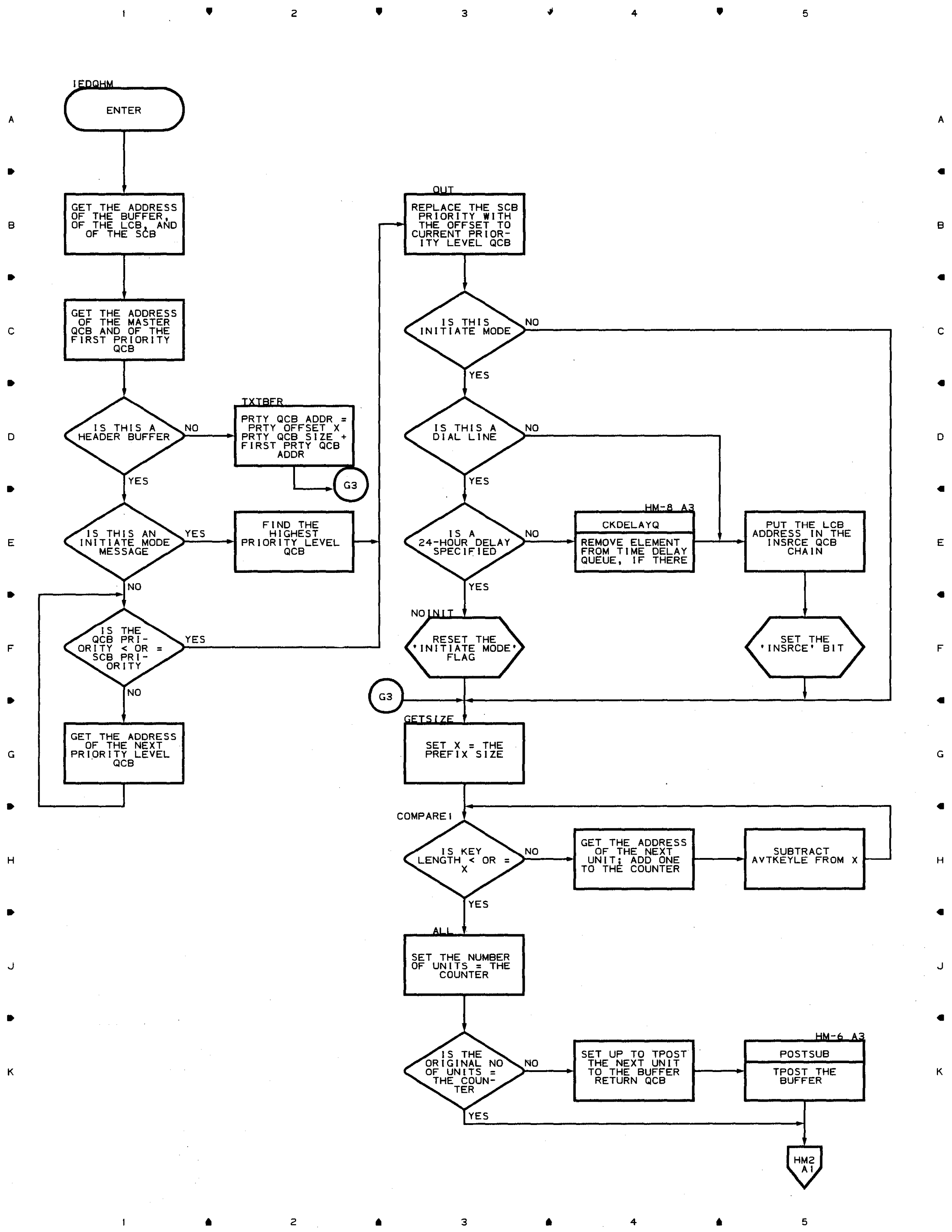

IEDQHM
ENTER
GET THE ADDRESS OF THE BUFFER, OF THE LCB, AND OF THE SCB
GET THE ADDRESS OF THE MASTER QCB AND OF THE FIRST PRIORITY QCB
IS THIS A HEADER BUFFER
NO
YES
TXTBFR
PRTY QCB ADDR = PRTY OFFSET X PRTY QCB SIZE + FIRST PRTY QCB ADDR
G3
IS THIS AN INITIATE MODE MESSAGE
YES
NO
FIND THE HIGHEST PRIORITY LEVEL QCB
IS THE QCB PRIORITY < OR = SCB PRIORITY
YES
NO
GET THE ADDRESS OF THE NEXT PRIORITY LEVEL QCB
OUT
REPLACE THE SCB PRIORITY WITH THE OFFSET TO CURRENT PRIORITY LEVEL QCB
IS THIS INITIATE MODE
NO
YES
IS THIS A DIAL LINE
NO
YES
IS A 24-HOUR DELAY SPECIFIED
NO
YES
HM-8 A3
CKDELAYQ
REMOVE ELEMENT FROM TIME DELAY QUEUE, IF THERE
PUT THE LCB ADDRESS IN THE INSRCE QCB CHAIN
SET THE 'INSRCE' BIT
NOINIT
RESET THE 'INITIATE MODE' FLAG
G3
GETSIZE
SET X = THE PREFIX SIZE
COMPARE1
IS KEY LENGTH < OR = X
NO
YES
GET THE ADDRESS OF THE NEXT UNIT; ADD ONE TO THE COUNTER
SUBTRACT AVTKEYLE FROM X
ALL
SET THE NUMBER OF UNITS = THE COUNTER
IS THE ORIGINAL NO OF UNITS = THE COUNTER
NO
YES
SET UP TO TPOST THE NEXT UNIT TO THE BUFFER RETURN QCB
HM-6 A3
POSTSUB
TPOST THE BUFFER
HM2 A1
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart HM-2 DESTINATION SCHEDULER

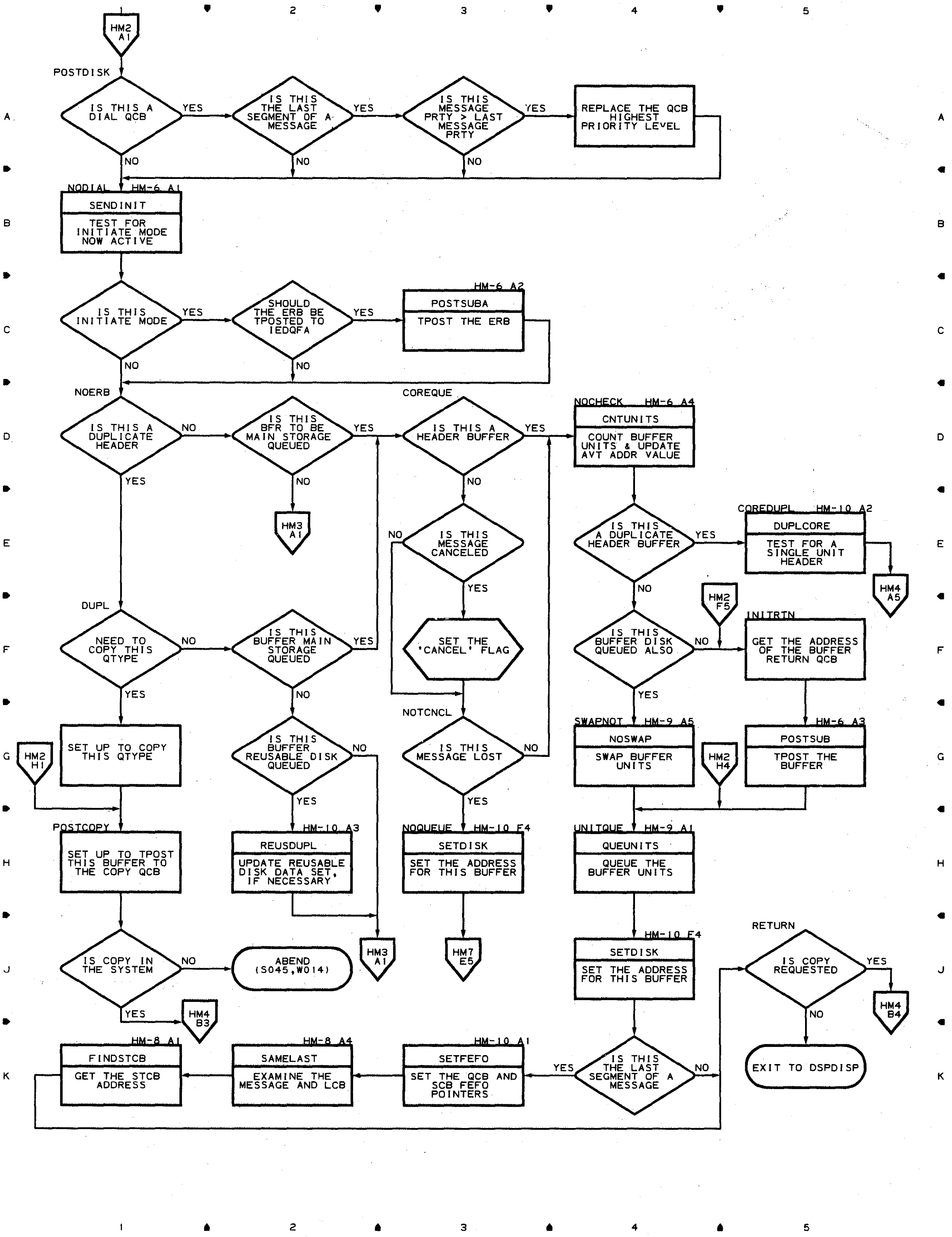


Chart HM-3 DESTINATION SCHEDULER

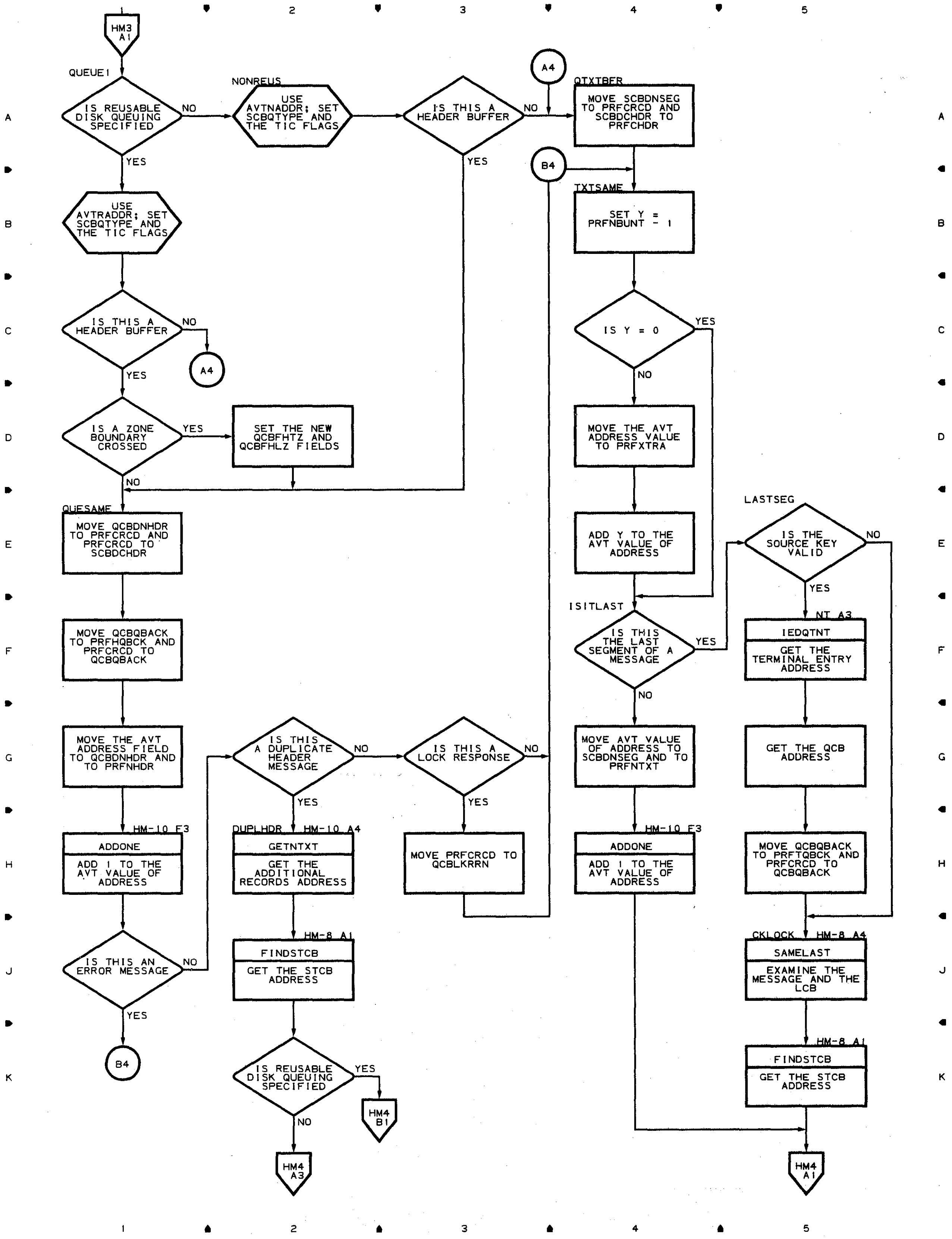
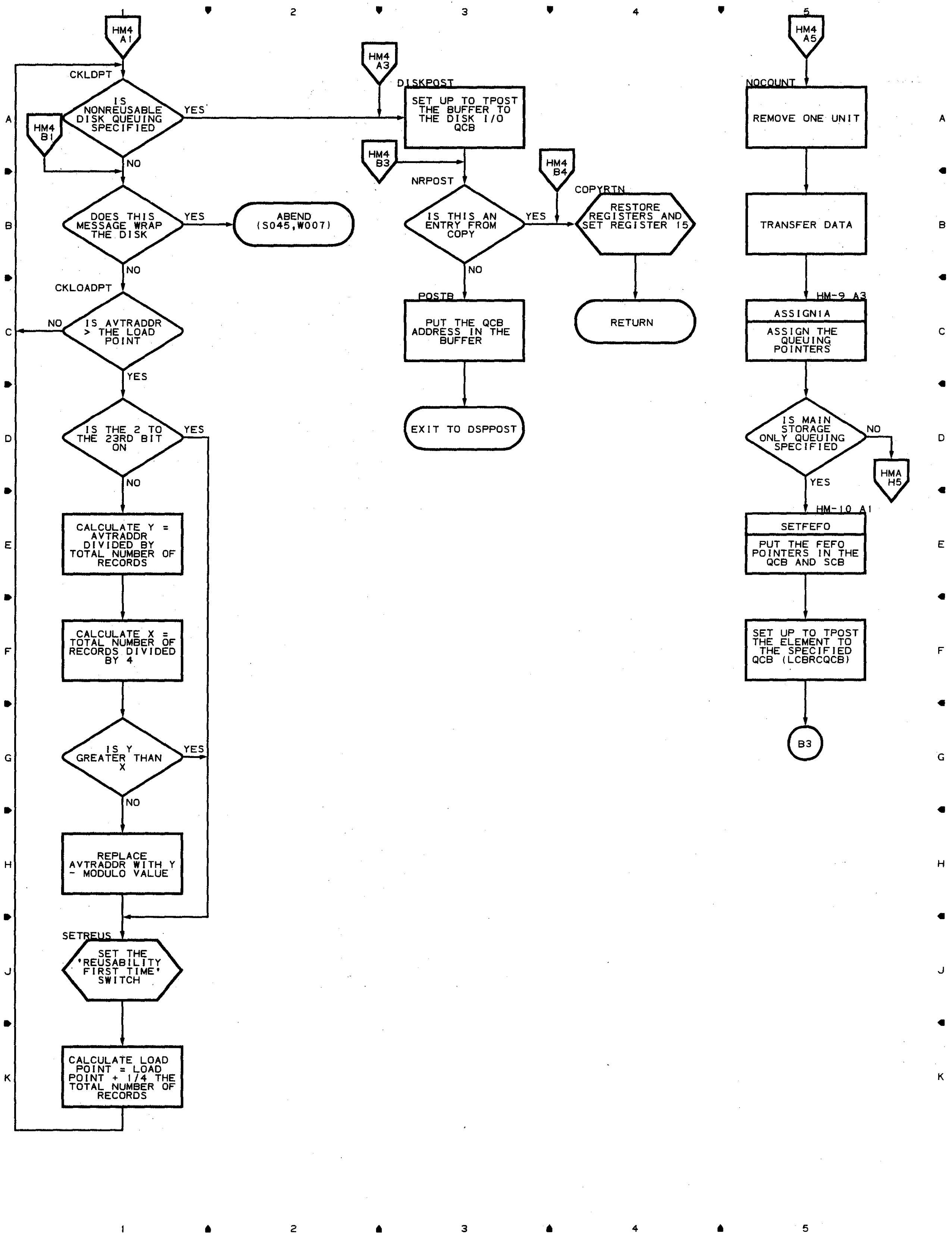

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
HM3 A1
QUEUE1
IS REUSABLE DISK QUEUING SPECIFIED
NO
YES
NONREUS
USE AVTNADDR; SET SCBQTYPE AND THE TIC FLAGS
IS THIS A HEADER BUFFER
NO
YES
A4
QTXTBFR
MOVE SCBDNSEG TO PRFCRCD AND SCBDCHDR TO PRFCHDR
B4
TXTSAME
SET Y = PRFNBUNT - 1
USE AVTRADDR; SET SCBQTYPE AND THE TIC FLAGS
IS THIS A HEADER BUFFER
NO
A4
YES
IS Y = 0
YES
NO
IS A ZONE BOUNDARY CROSSED
YES
SET THE NEW QCBFHTZ AND QCBFHLZ FIELDS
NO
MOVE THE AVT ADDRESS VALUE TO PRFXTRA
QUESAME
MOVE QCBDNHDR TO PRFCRCD AND PRFCRCD TO SCBDCHDR
ADD Y TO THE AVT VALUE OF ADDRESS
LASTSEG
IS THE SOURCE KEY VALID
NO
YES
ISITLAST
MOVE QCBQBACK TO PRFHQBCK AND PRFCRCD TO QCBQBACK
IS THIS THE LAST SEGMENT OF A MESSAGE
YES
NO
NT A3
IEDQTNT
GET THE TERMINAL ENTRY ADDRESS
MOVE THE AVT ADDRESS FIELD TO QCBDNHDR AND TO PRFNHDR
IS THIS A DUPLICATE HEADER MESSAGE
NO
YES
IS THIS A LOCK RESPONSE
NO
YES
MOVE AVT VALUE OF ADDRESS TO SCBDNSEG AND TO PRFNTXT
GET THE QCB ADDRESS
HM-10 F3
ADDONE
ADD 1 TO THE AVT VALUE OF ADDRESS
DUPLHDR
HM-10 A4
GETNTXT
GET THE ADDITIONAL RECORDS ADDRESS
MOVE PRFCRCD TO QCBLKRRN
HM-10 F3
ADDONE
ADD 1 TO THE AVT VALUE OF ADDRESS
MOVE QCBQBACK TO PRFTQBCK AND PRFCRCD TO QCBQBACK
HM-8 A1
FINDSTCB
GET THE STCB ADDRESS
CKLOCK
HM-8 A4
SAMELAST
EXAMINE THE MESSAGE AND THE LCB
IS THIS AN ERROR MESSAGE
NO
YES
B4
IS REUSABLE DISK QUEUING SPECIFIED
YES
NO
HM4 B1
HM4 A3
HM-8 A1
FINDSTCB
GET THE STCB ADDRESS
HM4 A1

Chart HM-4 DESTINATION SCHEDULER



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
HM4 A1
CKLDPT
IS NONREUSABLE DISK QUEUING SPECIFIED
YES
NO
HM4 B1
DOES THIS MESSAGE WRAP THE DISK
YES
NO
ABEND (S045,W007)
CKLOADPT
IS AVTRADDR > THE LOAD POINT
NO
YES
IS THE 2 TO THE 23RD BIT ON
YES
NO
CALCULATE Y = AVTRADDR DIVIDED BY TOTAL NUMBER OF RECORDS
CALCULATE X = TOTAL NUMBER OF RECORDS DIVIDED BY 4
IS Y GREATER THAN X
YES
NO
REPLACE AVTRADDR WITH Y - MODULO VALUE
SETREUS
SET THE 'REUSABILITY FIRST TIME' SWITCH
CALCULATE LOAD POINT = LOAD POINT + 1/4 THE TOTAL NUMBER OF RECORDS
HM4 A3
DISKPOST
SET UP TO TPOST THE BUFFER TO THE DISK I/O QCB
HM4 B3
NRPOST
IS THIS AN ENTRY FROM COPY
YES
NO
POSTB
PUT THE QCB ADDRESS IN THE BUFFER
EXIT TO DSPPOST
HM4 B4
COPYRTN
RESTORE REGISTERS AND SET REGISTER 15
RETURN
HM4 A5
NOCOUNT
REMOVE ONE UNIT
TRANSFER DATA
HM-9 A3
ASSIGNIA
ASSIGN THE QUEUING POINTERS
IS MAIN STORAGE ONLY QUEUING SPECIFIED
NO
HMA H5
YES
HM-10 A1
SETFEFO
PUT THE FEFO POINTERS IN THE QCB AND SCB
SET UP TO TPOST THE ELEMENT TO THE SPECIFIED QCB (LCBRCQCB)
B3

Chart HM-5 DESTINATION SCHEDULER

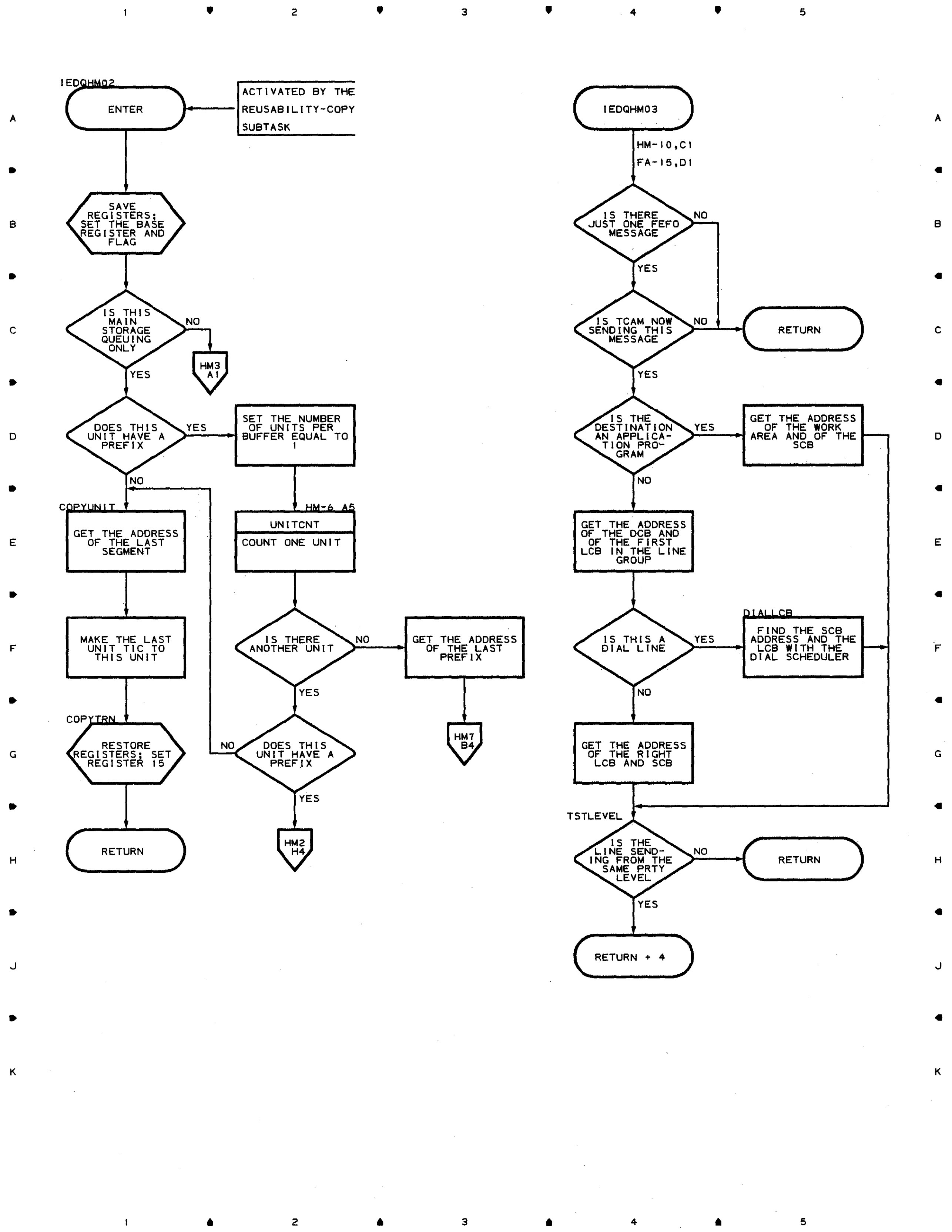

IEDQHM02
ENTER
ACTIVATED BY THE REUSABILITY-COPY SUBTASK
SAVE REGISTERS; SET THE BASE REGISTER AND FLAG
IS THIS MAIN STORAGE QUEUING ONLY
NO
HM3 A1
YES
DOES THIS UNIT HAVE A PREFIX
YES
SET THE NUMBER OF UNITS PER BUFFER EQUAL TO 1
NO
COPYUNIT
GET THE ADDRESS OF THE LAST SEGMENT
HM-6 A5
UNITCNT
COUNT ONE UNIT
MAKE THE LAST UNIT TIC TO THIS UNIT
IS THERE ANOTHER UNIT
NO
GET THE ADDRESS OF THE LAST PREFIX
YES
HM7 B4
COPYTRN
RESTORE REGISTERS; SET REGISTER 15
NO
DOES THIS UNIT HAVE A PREFIX
YES
RETURN
HM2 H4
IEDQHM03
HM-10,C1
FA-15,D1
IS THERE JUST ONE FEFO MESSAGE
NO
YES
IS TCAM NOW SENDING THIS MESSAGE
NO
RETURN
YES
IS THE DESTINATION AN APPLICATION PROGRAM
YES
GET THE ADDRESS OF THE WORK AREA AND OF THE SCB
NO
GET THE ADDRESS OF THE DCB AND OF THE FIRST LCB IN THE LINE GROUP
DIALLCB
IS THIS A DIAL LINE
YES
FIND THE SCB ADDRESS AND THE LCB WITH THE DIAL SCHEDULER
NO
GET THE ADDRESS OF THE RIGHT LCB AND SCB
TSTLEVEL
IS THE LINE SENDING FROM THE SAME PRTY LEVEL
NO
RETURN
YES
RETURN + 4
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart HM-6 DESTINATION SCHEDULER

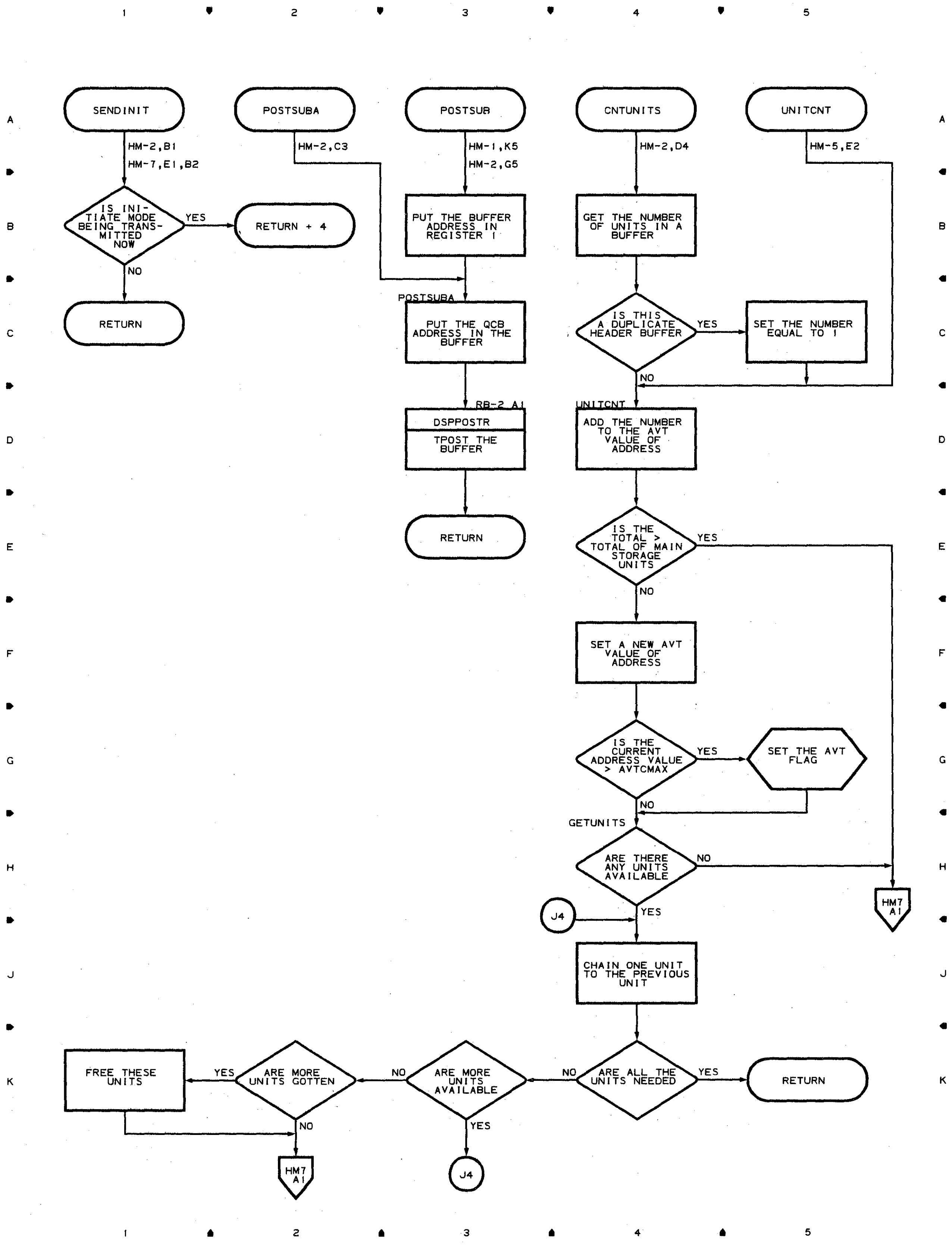

SENDINIT
HM-2,B1
HM-7,E1,B2
IS INITIATE MODE BEING TRANSMITTED NOW
YES
RETURN + 4
NO
RETURN
POSTSUBA
HM-2,C3
POSTSUB
HM-1,K5
HM-2,G5
PUT THE BUFFER ADDRESS IN REGISTER 1
POSTSUBA
PUT THE QCB ADDRESS IN THE BUFFER
RB-2 A1
DSPPOSTR
TPOST THE BUFFER
RETURN
CNTUNITS
HM-2,D4
GET THE NUMBER OF UNITS IN A BUFFER
IS THIS A DUPLICATE HEADER BUFFER
YES
SET THE NUMBER EQUAL TO 1
NO
UNITCNT
HM-5,E2
UNITCNT
ADD THE NUMBER TO THE AVT VALUE OF ADDRESS
IS THE TOTAL > TOTAL OF MAIN STORAGE UNITS
YES
NO
SET A NEW AVT VALUE OF ADDRESS
IS THE CURRENT ADDRESS VALUE > AVTCMAX
YES
SET THE AVT FLAG
NO
GETUNITS
ARE THERE ANY UNITS AVAILABLE
NO
HM7 A1
J4
YES
CHAIN ONE UNIT TO THE PREVIOUS UNIT
ARE ALL THE UNITS NEEDED
YES
RETURN
NO
ARE MORE UNITS AVAILABLE
YES
J4
NO
ARE MORE UNITS GOTTEN
YES
FREE THESE UNITS
NO
HM7 A1
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart HM-7 DESTINATION SCHEDULER

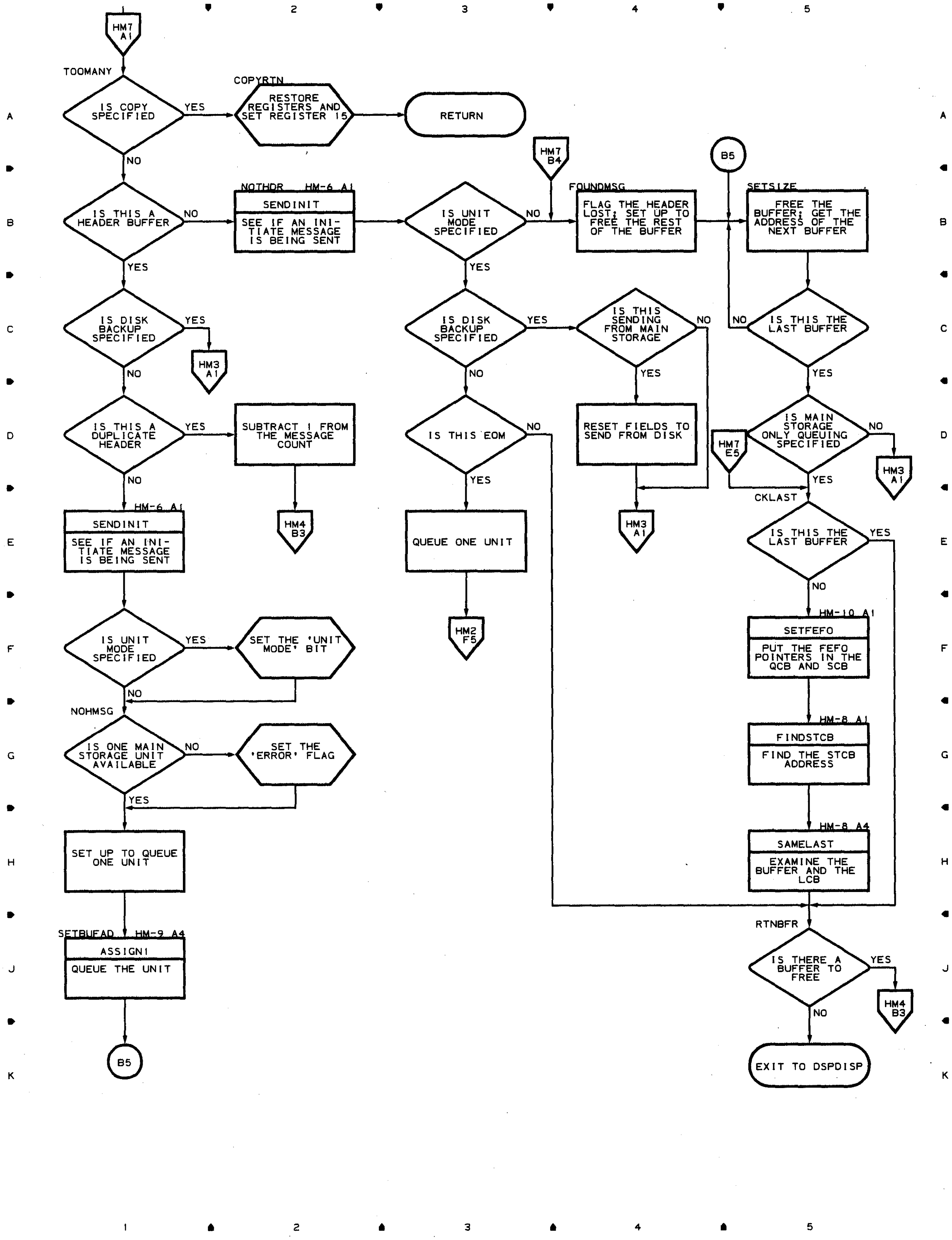
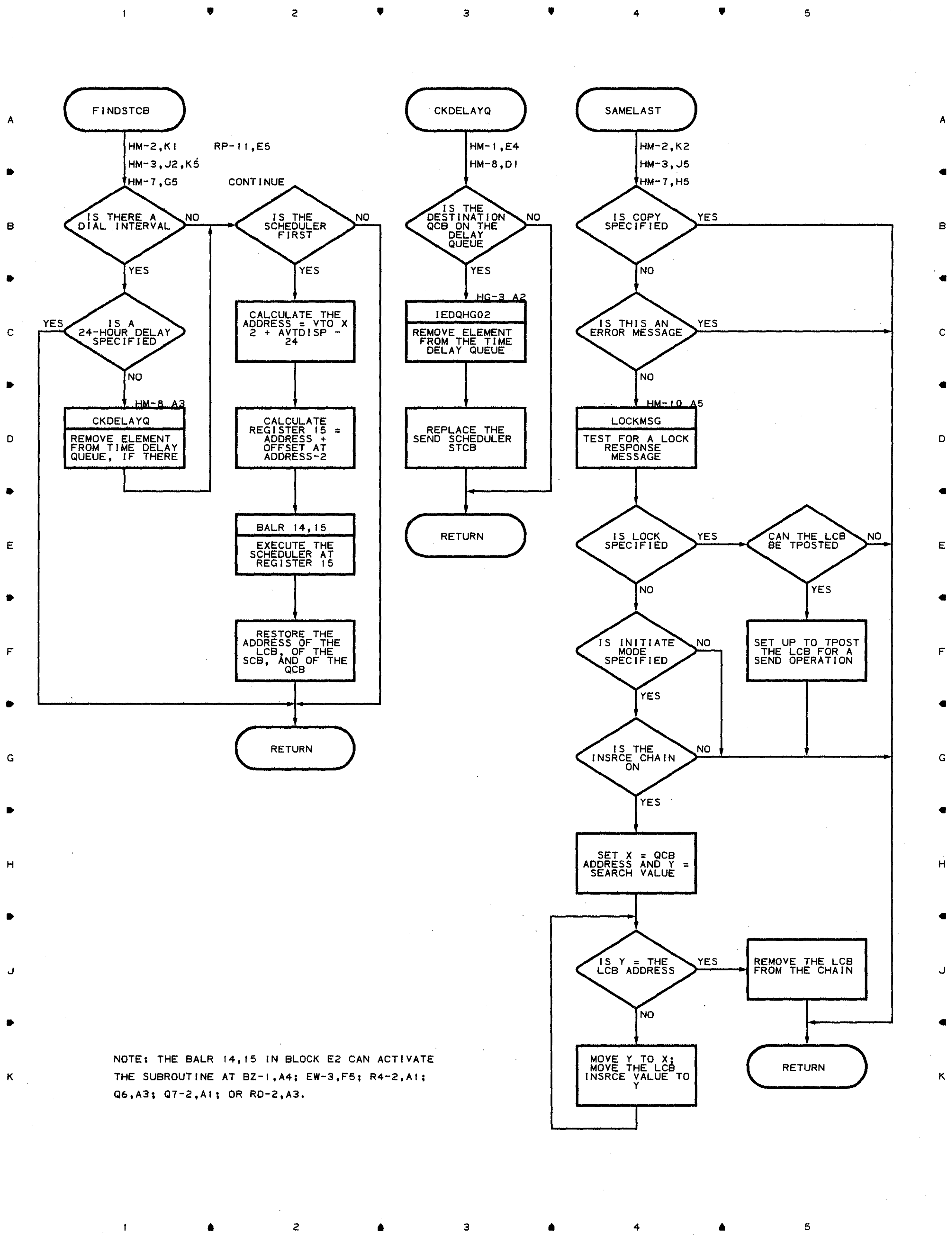


Chart HM-8 DESTINATION SCHEDULER


FINDSTCB
HM-2,K1
HM-3,J2,K5
HM-7,G5
RP-11,E5
CONTINUE
IS THERE A DIAL INTERVAL
NO
YES
IS A 24-HOUR DELAY SPECIFIED
YES
NO
HM-8 A3
CKDELAYQ
REMOVE ELEMENT FROM TIME DELAY QUEUE, IF THERE
IS THE SCHEDULER FIRST
NO
YES
CALCULATE THE ADDRESS = VTO X 2 + AVTDISP - 24
CALCULATE REGISTER 15 = ADDRESS + OFFSET AT ADDRESS-2
BALR 14,15
EXECUTE THE SCHEDULER AT REGISTER 15
RESTORE THE ADDRESS OF THE LCB, OF THE SCB, AND OF THE QCB
RETURN
CKDELAYQ
HM-1,E4
HM-8,D1
IS THE DESTINATION QCB ON THE DELAY QUEUE
NO
YES
HG-3 A2
IEDQHG02
REMOVE ELEMENT FROM THE TIME DELAY QUEUE
REPLACE THE SEND SCHEDULER STCB
RETURN
SAMELAST
HM-2,K2
HM-3,J5
HM-7,H5
IS COPY SPECIFIED
YES
NO
IS THIS AN ERROR MESSAGE
YES
NO
HM-10 A5
LOCKMSG
TEST FOR A LOCK RESPONSE MESSAGE
IS LOCK SPECIFIED
YES
NO
CAN THE LCB BE TPOSTED
NO
YES
SET UP TO TPOST THE LCB FOR A SEND OPERATION
IS INITIATE MODE SPECIFIED
NO
YES
IS THE INSRCE CHAIN ON
NO
YES
SET X = QCB ADDRESS AND Y = SEARCH VALUE
IS Y = THE LCB ADDRESS
YES
NO
REMOVE THE LCB FROM THE CHAIN
MOVE Y TO X; MOVE THE LCB INSRCE VALUE TO Y
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

NOTE: THE BALR 14,15 IN BLOCK E2 CAN ACTIVATE THE SUBROUTINE AT BZ-1,A4; EW-3,F5; R4-2,A1; Q6,A3; Q7-2,A1; OR RD-2,A3.

Chart HM-9 DESTINATION SCHEDULER

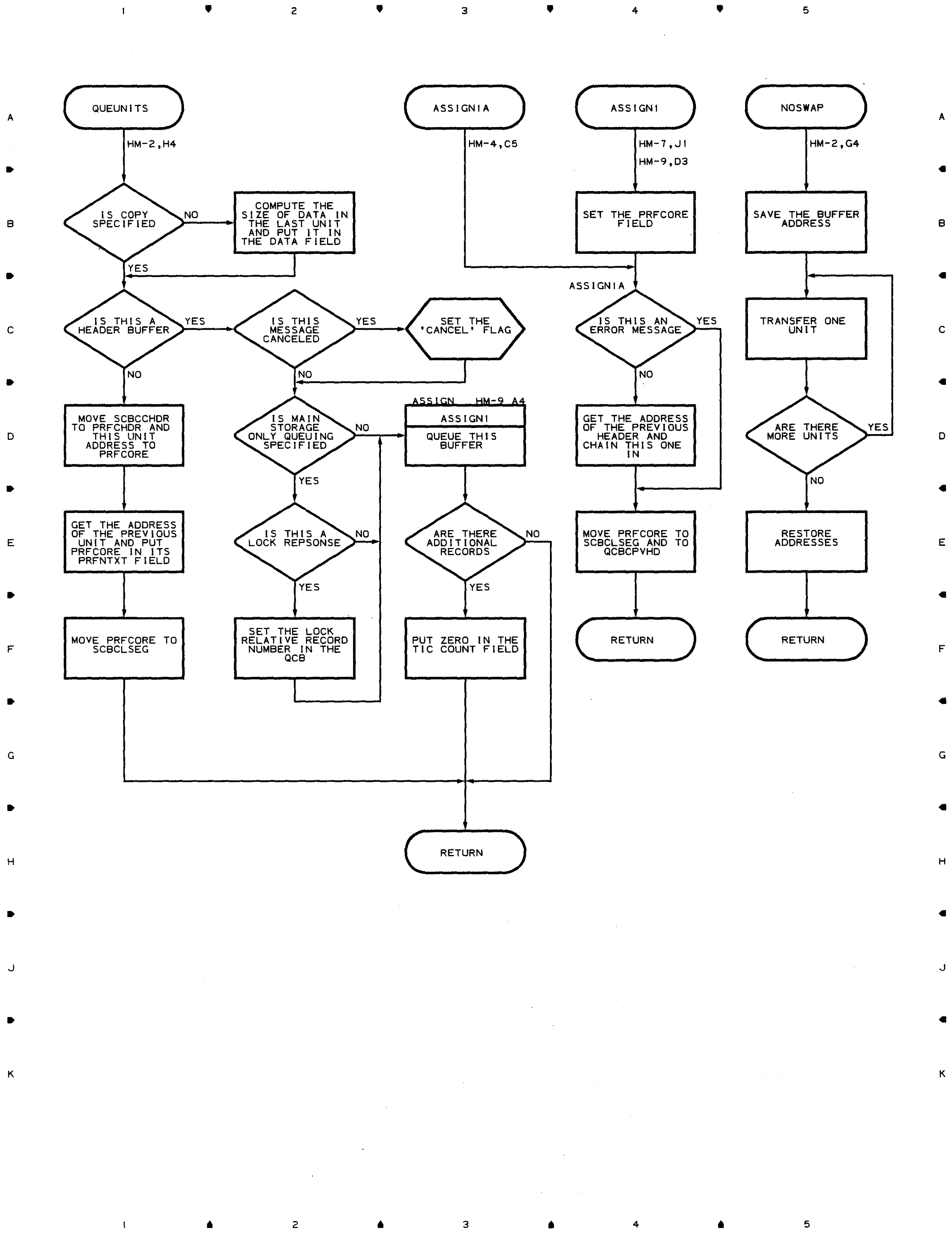

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
QUEUNITS
HM-2,H4
IS COPY SPECIFIED
NO
COMPUTE THE SIZE OF DATA IN THE LAST UNIT AND PUT IT IN THE DATA FIELD
YES
IS THIS A HEADER BUFFER
YES
IS THIS MESSAGE CANCELED
YES
SET THE 'CANCEL' FLAG
NO
NO
MOVE SCBCCHDR TO PRFCHDR AND THIS UNIT ADDRESS TO PRFCORE
IS MAIN STORAGE ONLY QUEUING SPECIFIED
NO
ASSIGN HM-9,A4
ASSIGN1
QUEUE THIS BUFFER
YES
GET THE ADDRESS OF THE PREVIOUS UNIT AND PUT PRFCORE IN ITS PRFNTXT FIELD
IS THIS A LOCK REPSONSE
NO
ARE THERE ADDITIONAL RECORDS
NO
YES
YES
MOVE PRFCORE TO SCBCLSEG
SET THE LOCK RELATIVE RECORD NUMBER IN THE QCB
PUT ZERO IN THE TIC COUNT FIELD
RETURN
ASSIGN1A
HM-4,C5
ASSIGN1
HM-7,J1
HM-9,D3
SET THE PRFCORE FIELD
ASSIGN1A
IS THIS AN ERROR MESSAGE
YES
NO
GET THE ADDRESS OF THE PREVIOUS HEADER AND CHAIN THIS ONE IN
MOVE PRFCORE TO SCBCLSEG AND TO QCBCPVHD
RETURN
NOSWAP
HM-2,G4
SAVE THE BUFFER ADDRESS
TRANSFER ONE UNIT
ARE THERE MORE UNITS
YES
NO
RESTORE ADDRESSES
RETURN

Chart HM-10 DESTINATION SCHEDULER

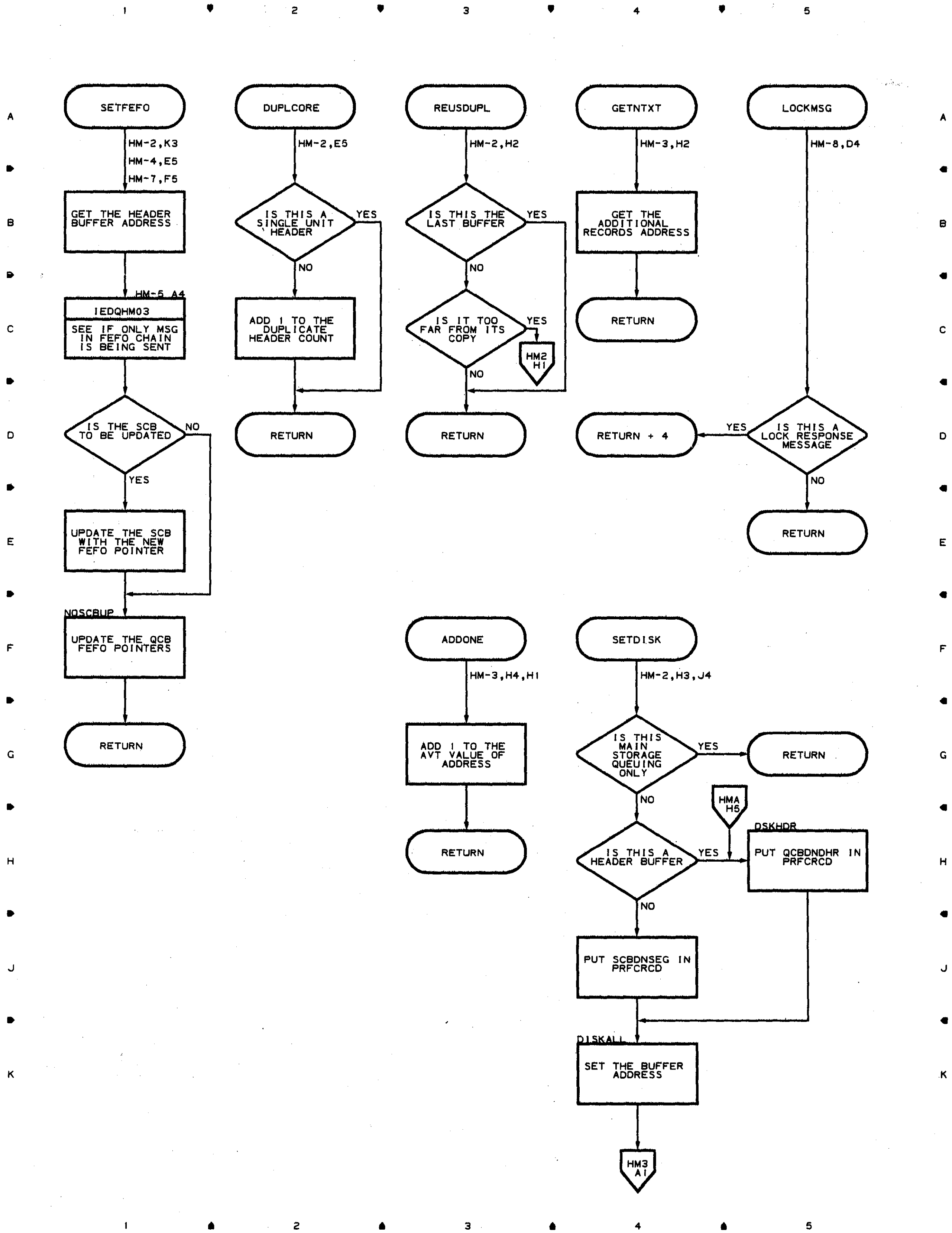
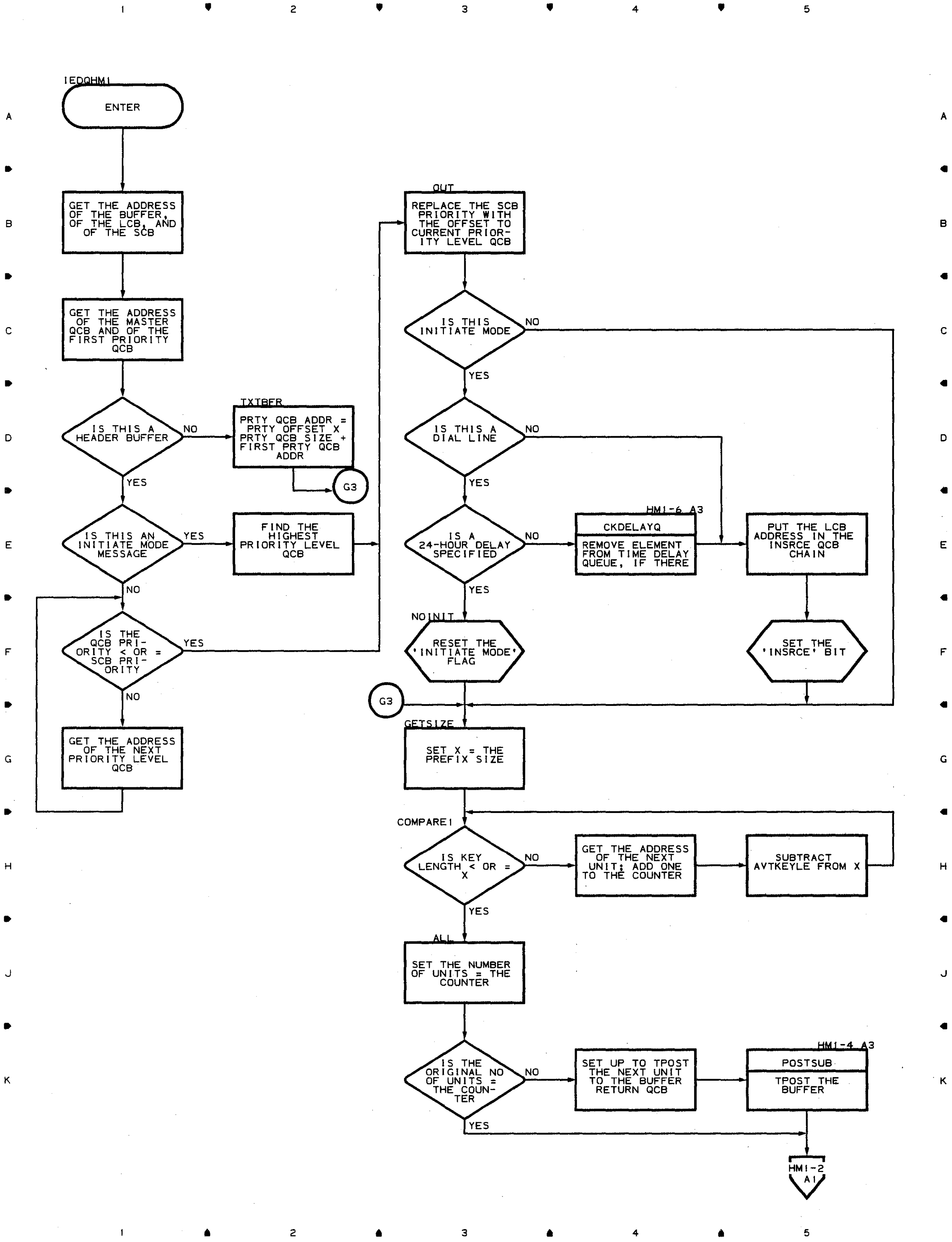


Chart HM1-1 DESTINATION SCHEDULER - MAIN STORAGE QUEUING ONLY



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQHM1
ENTER
GET THE ADDRESS OF THE BUFFER, OF THE LCB, AND OF THE SCB
GET THE ADDRESS OF THE MASTER QCB AND OF THE FIRST PRIORITY QCB
IS THIS A HEADER BUFFER
NO
YES
TXTBFR
PRTY QCB ADDR = PRTY OFFSET X PRTY QCB SIZE + FIRST PRTY QCB ADDR
G3
IS THIS AN INITIATE MODE MESSAGE
YES
NO
FIND THE HIGHEST PRIORITY LEVEL QCB
IS THE QCB PRI-ORITY < OR = SCB PRI-ORITY
YES
NO
GET THE ADDRESS OF THE NEXT PRIORITY LEVEL QCB
OUT
REPLACE THE SCB PRIORITY WITH THE OFFSET TO CURRENT PRIOR-ITY LEVEL QCB
IS THIS INITIATE MODE
NO
YES
IS THIS A DIAL LINE
NO
YES
IS A 24-HOUR DELAY SPECIFIED
NO
YES
HM1-6 A3
CKDELAYQ
REMOVE ELEMENT FROM TIME DELAY QUEUE, IF THERE
PUT THE LCB ADDRESS IN THE INSRCE QCB CHAIN
NOINIT
RESET THE 'INITIATE MODE' FLAG
SET THE 'INSRCE' BIT
G3
GETSIZE
SET X = THE PREFIX SIZE
COMPARE1
IS KEY LENGTH < OR = X
NO
YES
GET THE ADDRESS OF THE NEXT UNIT; ADD ONE TO THE COUNTER
SUBTRACT AVTKEYLE FROM X
ALL
SET THE NUMBER OF UNITS = THE COUNTER
IS THE ORIGINAL NO OF UNITS = THE COUN-TER
NO
YES
SET UP TO TPOST THE NEXT UNIT TO THE BUFFER RETURN QCB
HM1-4 A3
POSTSUB
TPOST THE BUFFER
HM1-2 A1

Chart HM1-2 DESTINATION SCHEDULER - MAIN STORAGE QUEUING ONLY

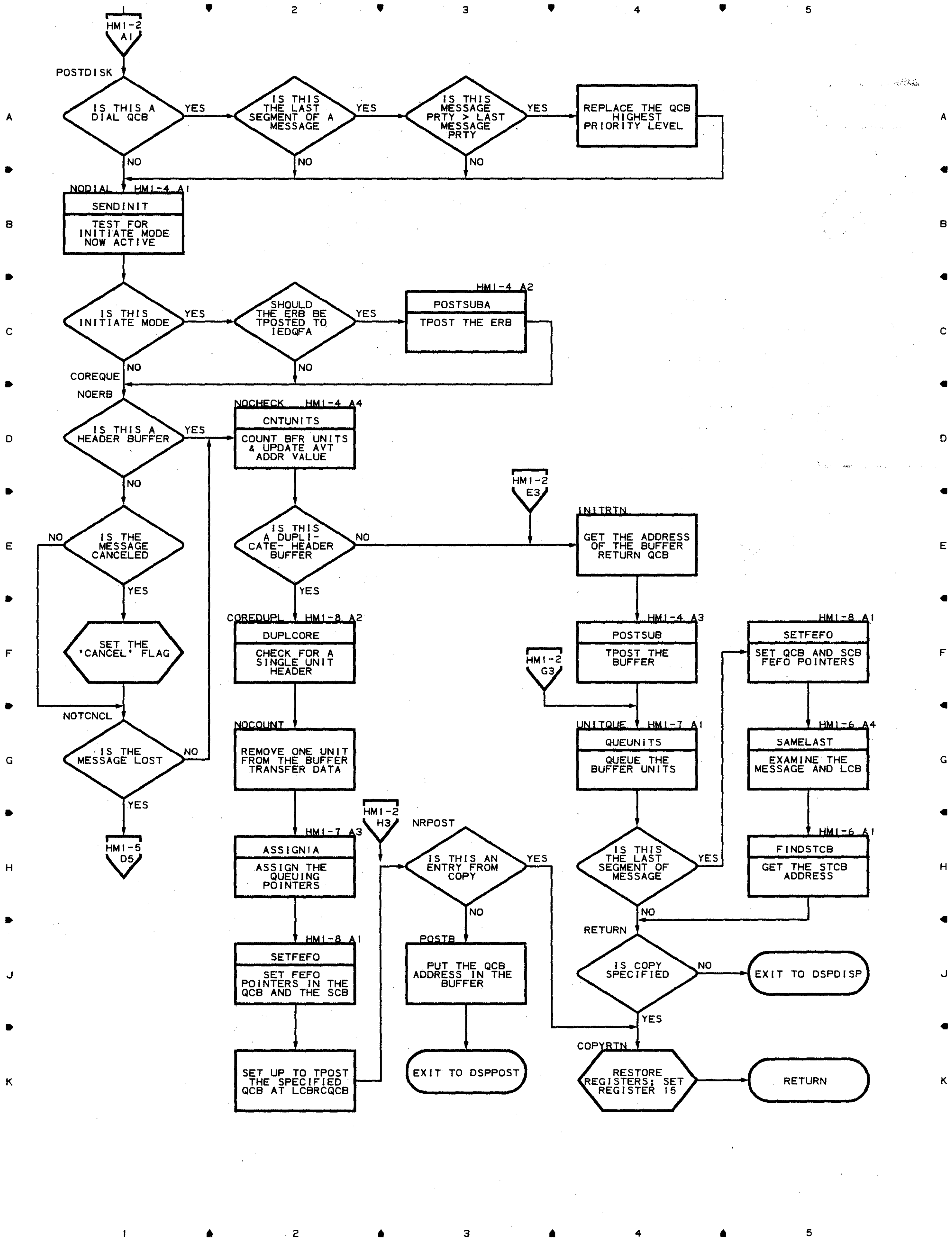
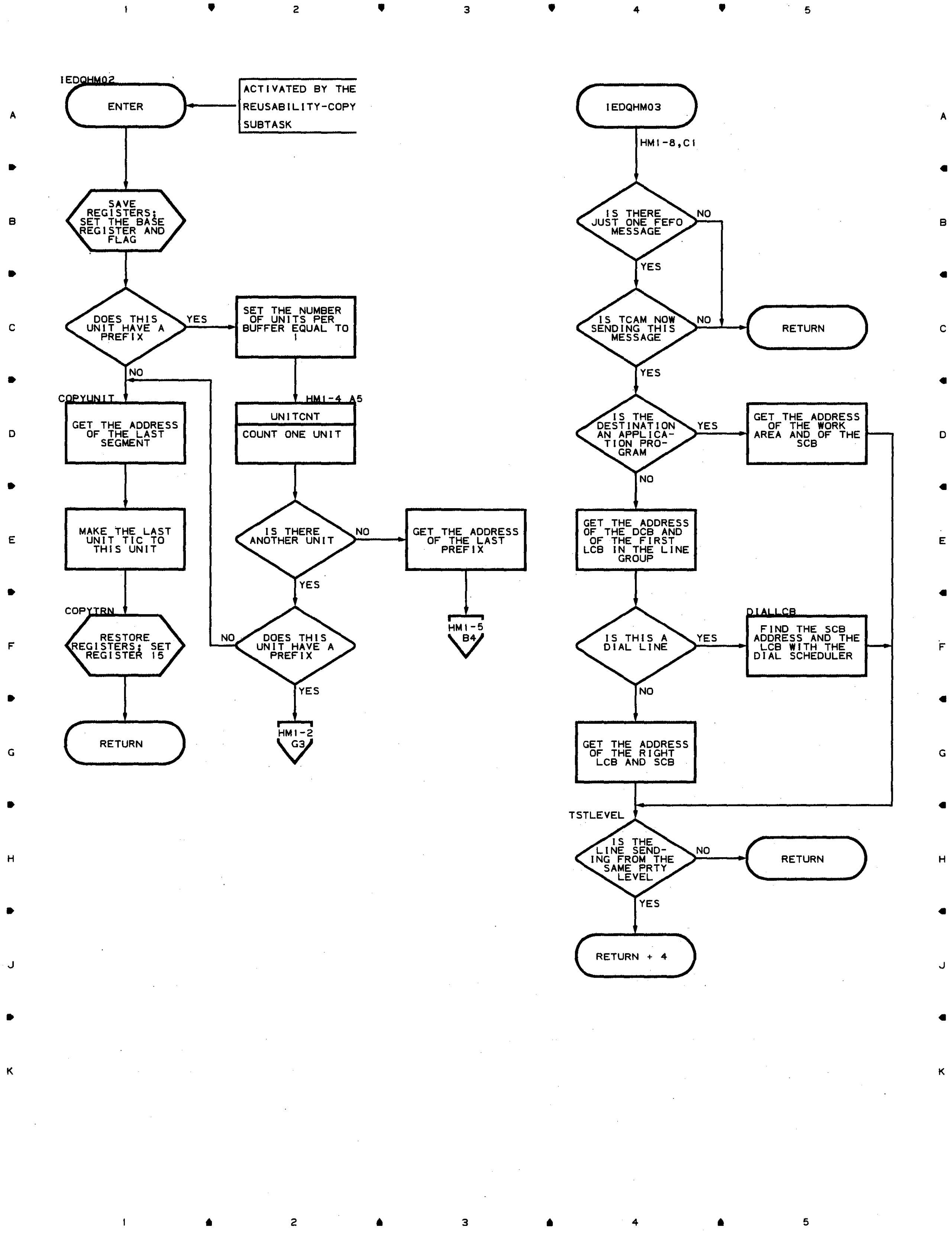

HM1-2 A1
POSTDISK
IS THIS A DIAL QCB
YES
NO
IS THIS THE LAST SEGMENT OF A MESSAGE
YES
NO
IS THIS MESSAGE PRTY > LAST MESSAGE PRTY
YES
NO
REPLACE THE QCB HIGHEST PRIORITY LEVEL
NODIAL HM1-4 A1
SENDINIT
TEST FOR INITIATE MODE NOW ACTIVE
IS THIS INITIATE MODE
YES
NO
SHOULD THE ERB BE TPOSTED TO IEDQFA
YES
NO
HM1-4 A2
POSTSUBA
TPOST THE ERB
COREQUE
NOERB
IS THIS A HEADER BUFFER
YES
NO
NOCHECK HM1-4 A4
CNTUNITS
COUNT BFR UNITS & UPDATE AVT ADDR VALUE
IS THE MESSAGE CANCELED
NO
YES
SET THE 'CANCEL' FLAG
NOTCNCL
IS THE MESSAGE LOST
NO
YES
HM1-5 D5
IS THIS A DUPLI- CATE- HEADER BUFFER
NO
YES
HM1-2 E3
INITRTN
GET THE ADDRESS OF THE BUFFER RETURN QCB
COREDUPL HM1-8 A2
DUPLCORE
CHECK FOR A SINGLE UNIT HEADER
NOCOUNT
REMOVE ONE UNIT FROM THE BUFFER TRANSFER DATA
HM1-7 A3
ASSIGNIA
ASSIGN THE QUEUING POINTERS
HM1-8 A1
SETFEFO
SET FEFO POINTERS IN THE QCB AND THE SCB
SET UP TO TPOST THE SPECIFIED QCB AT LCBRCQCB
HM1-2 H3
NRPOST
IS THIS AN ENTRY FROM COPY
YES
NO
POSTB
PUT THE QCB ADDRESS IN THE BUFFER
EXIT TO DSPPOST
HM1-4 A3
POSTSUB
TPOST THE BUFFER
HM1-2 G3
UNITQUE HM1-7 A1
QUEUNITS
QUEUE THE BUFFER UNITS
IS THIS THE LAST SEGMENT OF MESSAGE
YES
NO
RETURN
IS COPY SPECIFIED
NO
YES
EXIT TO DSPDISP
COPYRTN
RESTORE REGISTERS; SET REGISTER 15
RETURN
HM1-8 A1
SETFEFO
SET QCB AND SCB FEFO POINTERS
HM1-6 A4
SAMELAST
EXAMINE THE MESSAGE AND LCB
HM1-6 A1
FINDSTCB
GET THE STCB ADDRESS
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart HM1-3 DESTINATION SCHEDULER - MAIN STORAGE QUEUING ONLY



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQHM02
ENTER
ACTIVATED BY THE REUSABILITY-COPY SUBTASK
SAVE REGISTERS; SET THE BASE REGISTER AND FLAG
DOES THIS UNIT HAVE A PREFIX
YES
SET THE NUMBER OF UNITS PER BUFFER EQUAL TO 1
NO
COPYUNIT
GET THE ADDRESS OF THE LAST SEGMENT
HM1-4 A5
UNITCNT
COUNT ONE UNIT
MAKE THE LAST UNIT TIC TO THIS UNIT
IS THERE ANOTHER UNIT
NO
GET THE ADDRESS OF THE LAST PREFIX
YES
COPYTRN
RESTORE REGISTERS; SET REGISTER 15
NO
DOES THIS UNIT HAVE A PREFIX
HM1-5 B4
YES
RETURN
HM1-2 G3
IEDQHM03
HM1-8,C1
IS THERE JUST ONE FEFO MESSAGE
NO
YES
IS TCAM NOW SENDING THIS MESSAGE
NO
RETURN
YES
IS THE DESTINATION AN APPLICA-TION PRO-GRAM
YES
GET THE ADDRESS OF THE WORK AREA AND OF THE SCB
NO
GET THE ADDRESS OF THE DCB AND OF THE FIRST LCB IN THE LINE GROUP
DIALLCB
IS THIS A DIAL LINE
YES
FIND THE SCB ADDRESS AND THE LCB WITH THE DIAL SCHEDULER
NO
GET THE ADDRESS OF THE RIGHT LCB AND SCB
TSTLEVEL
IS THE LINE SEND-ING FROM THE SAME PRTY LEVEL
NO
RETURN
YES
RETURN + 4

Chart HM1-4 DESTINATION SCHEDULER - MAIN STORAGE QUEUING ONLY

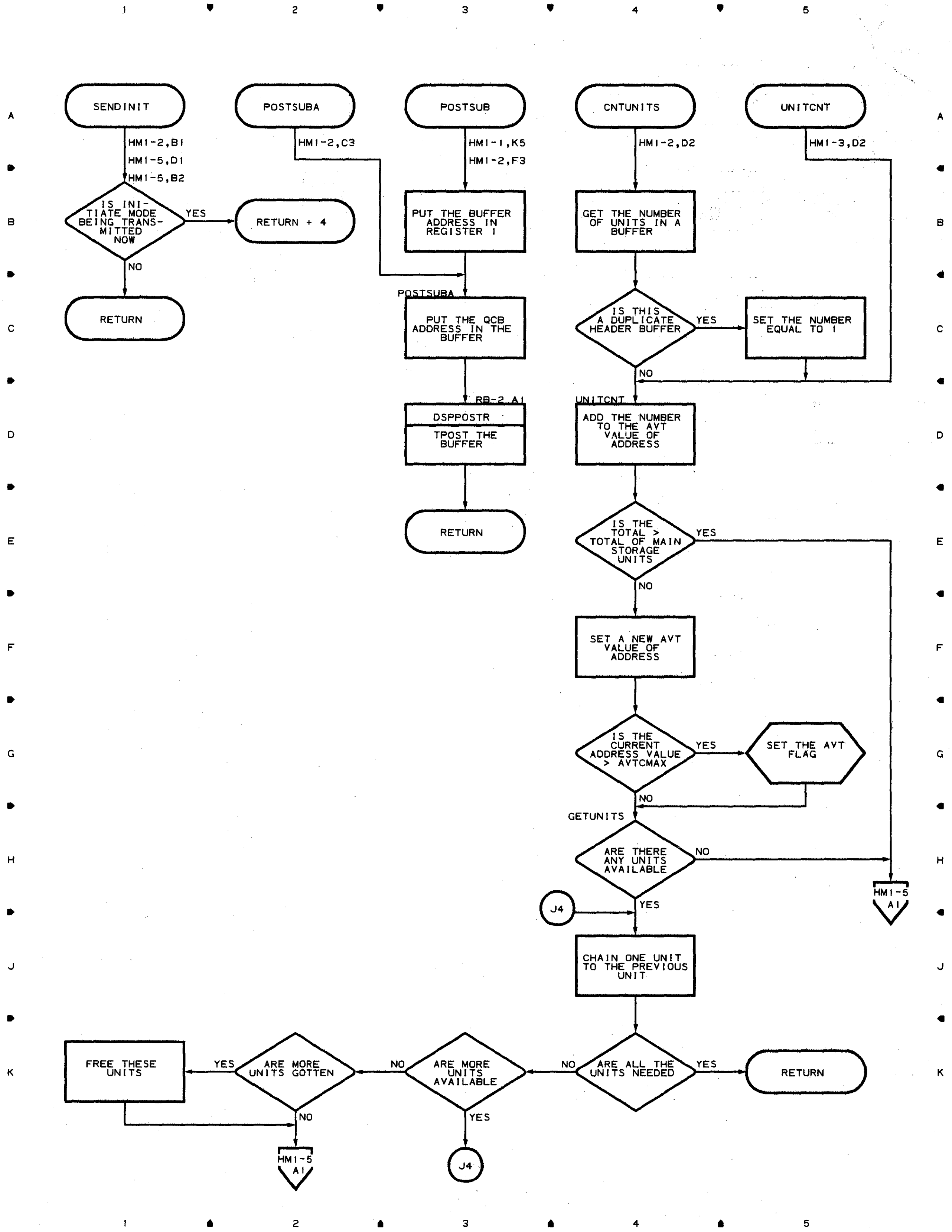

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
SENDINIT
HM1-2,B1
HM1-5,D1
HM1-5,B2
IS INITIATE MODE BEING TRANSMITTED NOW
YES
RETURN + 4
NO
RETURN
POSTSUBA
HM1-2,C3
POSTSUB
HM1-1,K5
HM1-2,F3
PUT THE BUFFER ADDRESS IN REGISTER 1
POSTSUBA
PUT THE QCB ADDRESS IN THE BUFFER
RB-2,A1
DSPPOSTR
TPOST THE BUFFER
RETURN
CNTUNITS
HM1-2,D2
GET THE NUMBER OF UNITS IN A BUFFER
IS THIS A DUPLICATE HEADER BUFFER
YES
SET THE NUMBER EQUAL TO 1
NO
UNITCNT
HM1-3,D2
UNITCNT
ADD THE NUMBER TO THE AVT VALUE OF ADDRESS
IS THE TOTAL > TOTAL OF MAIN STORAGE UNITS
YES
NO
SET A NEW AVT VALUE OF ADDRESS
IS THE CURRENT ADDRESS VALUE > AVTCMAX
YES
SET THE AVT FLAG
NO
GETUNITS
ARE THERE ANY UNITS AVAILABLE
NO
HM1-5 A1
YES
J4
CHAIN ONE UNIT TO THE PREVIOUS UNIT
ARE ALL THE UNITS NEEDED
YES
RETURN
NO
ARE MORE UNITS AVAILABLE
YES
J4
NO
ARE MORE UNITS GOTTEN
YES
FREE THESE UNITS
NO
HM1-5 A1

Chart HM1-5 DESTINATION SCHEDULER - MAIN STORAGE QUEUING ONLY

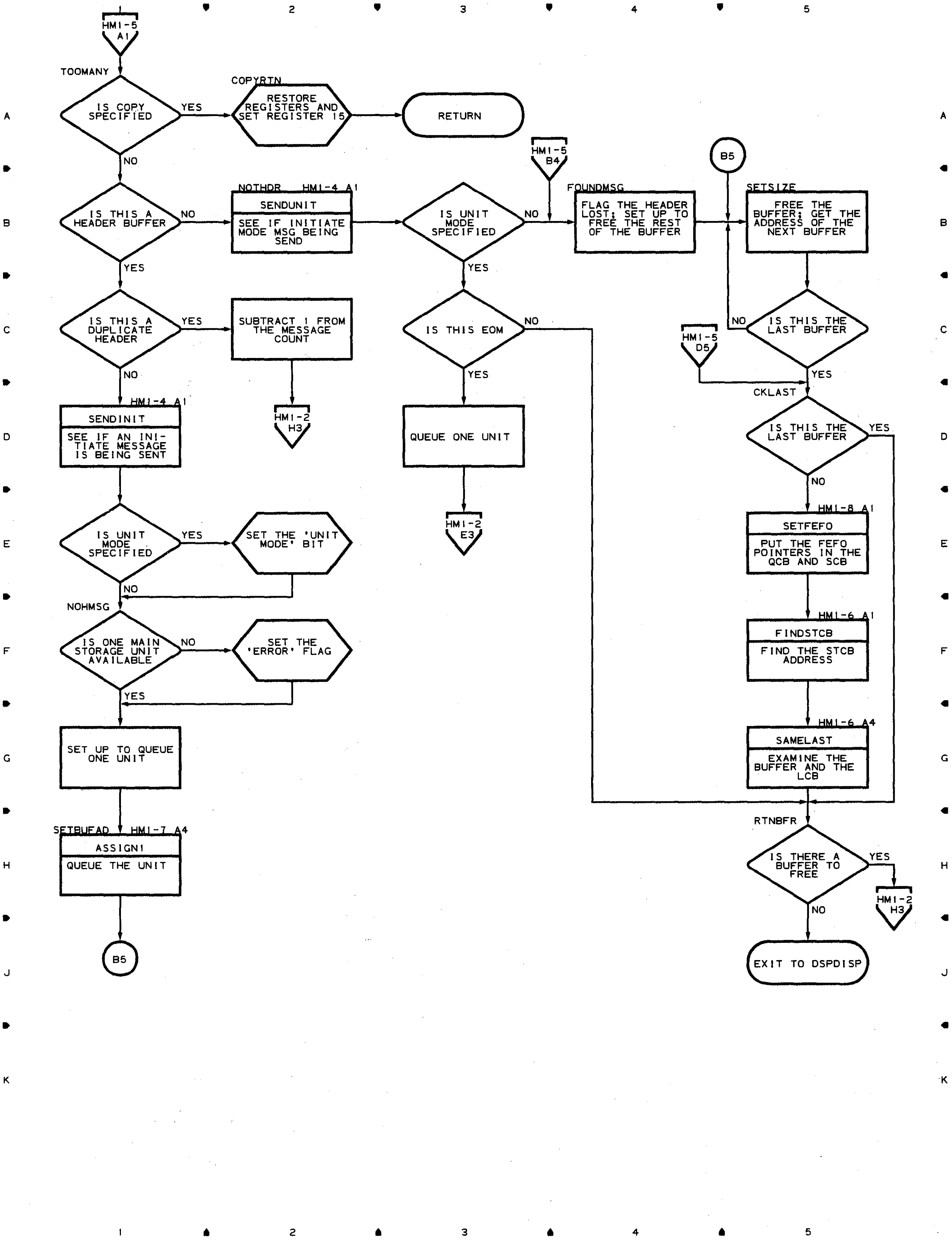
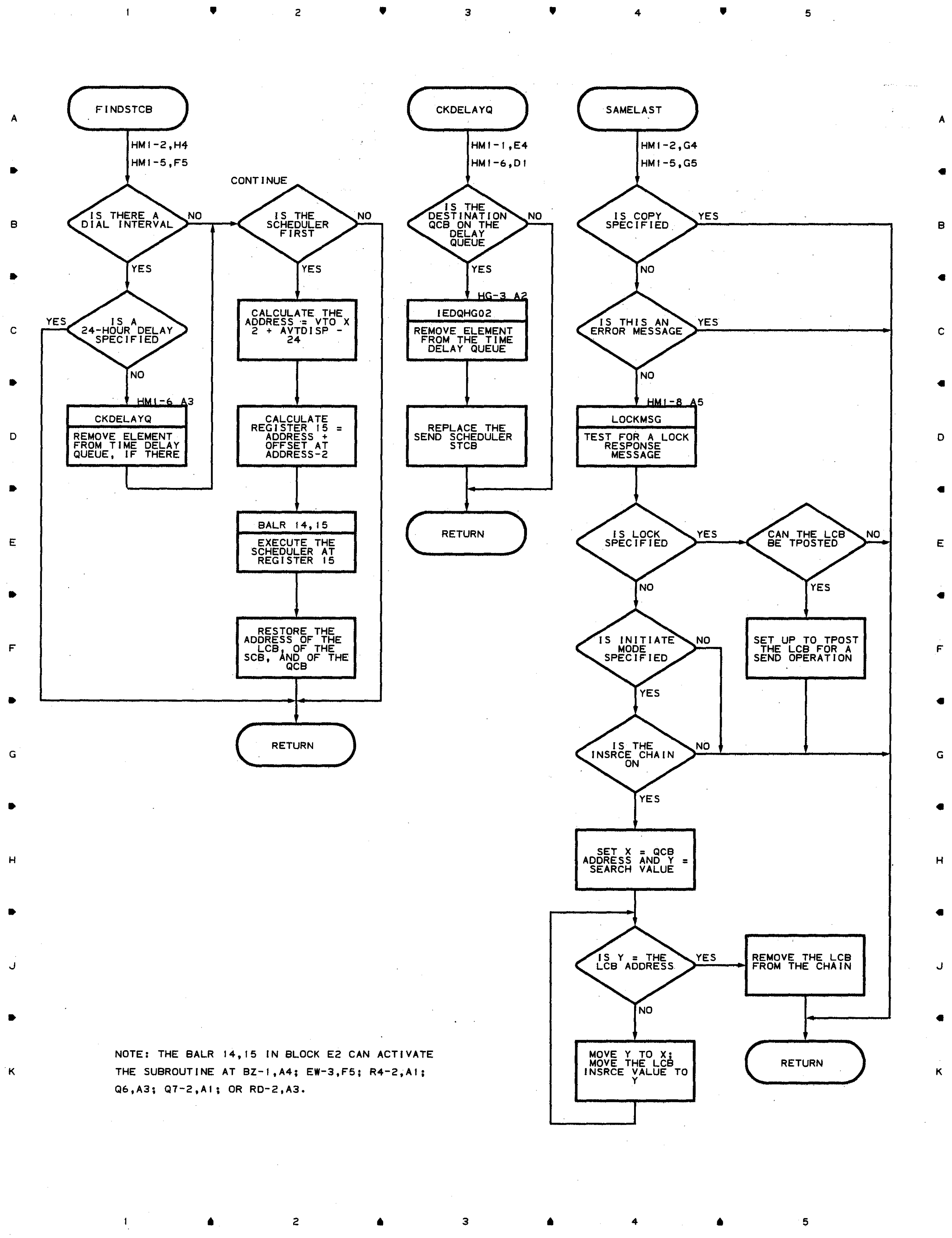


Chart HM1-6 DESTINATION SCHEDULER - MAIN STORAGE QUEUING ONLY


1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
FINDSTCB
HM1-2,H4
HM1-5,F5
IS THERE A DIAL INTERVAL
NO
YES
IS A 24-HOUR DELAY SPECIFIED
YES
NO
HM1-6 A3
CKDELAYQ
REMOVE ELEMENT FROM TIME DELAY QUEUE, IF THERE
CONTINUE
IS THE SCHEDULER FIRST
NO
YES
CALCULATE THE ADDRESS = VTO X 2 + AVTDISP - 24
CALCULATE REGISTER 15 = ADDRESS + OFFSET AT ADDRESS-2
BALR 14,15
EXECUTE THE SCHEDULER AT REGISTER 15
RESTORE THE ADDRESS OF THE LCB, OF THE SCB, AND OF THE QCB
RETURN
CKDELAYQ
HM1-1,E4
HM1-6,D1
IS THE DESTINATION QCB ON THE DELAY QUEUE
NO
YES
HG-3 A2
IEDQHG02
REMOVE ELEMENT FROM THE TIME DELAY QUEUE
REPLACE THE SEND SCHEDULER STCB
RETURN
SAMELAST
HM1-2,G4
HM1-5,G5
IS COPY SPECIFIED
YES
NO
IS THIS AN ERROR MESSAGE
YES
NO
HM1-8 A5
LOCKMSG
TEST FOR A LOCK RESPONSE MESSAGE
IS LOCK SPECIFIED
YES
NO
CAN THE LCB BE TPOSTED
NO
YES
SET UP TO TPOST THE LCB FOR A SEND OPERATION
IS INITIATE MODE SPECIFIED
NO
YES
IS THE INSRCE CHAIN ON
NO
YES
SET X = QCB ADDRESS AND Y = SEARCH VALUE
IS Y = THE LCB ADDRESS
YES
NO
REMOVE THE LCB FROM THE CHAIN
MOVE Y TO X; MOVE THE LCB INSRCE VALUE TO Y
RETURN

NOTE: THE BALR 14,15 IN BLOCK E2 CAN ACTIVATE THE SUBROUTINE AT BZ-1,A4; EW-3,F5; R4-2,A1; Q6,A3; Q7-2,A1; OR RD-2,A3.

Chart HM1-7 DESTINATION SCHEDULER - MAIN STORAGE QUEUING ONLY

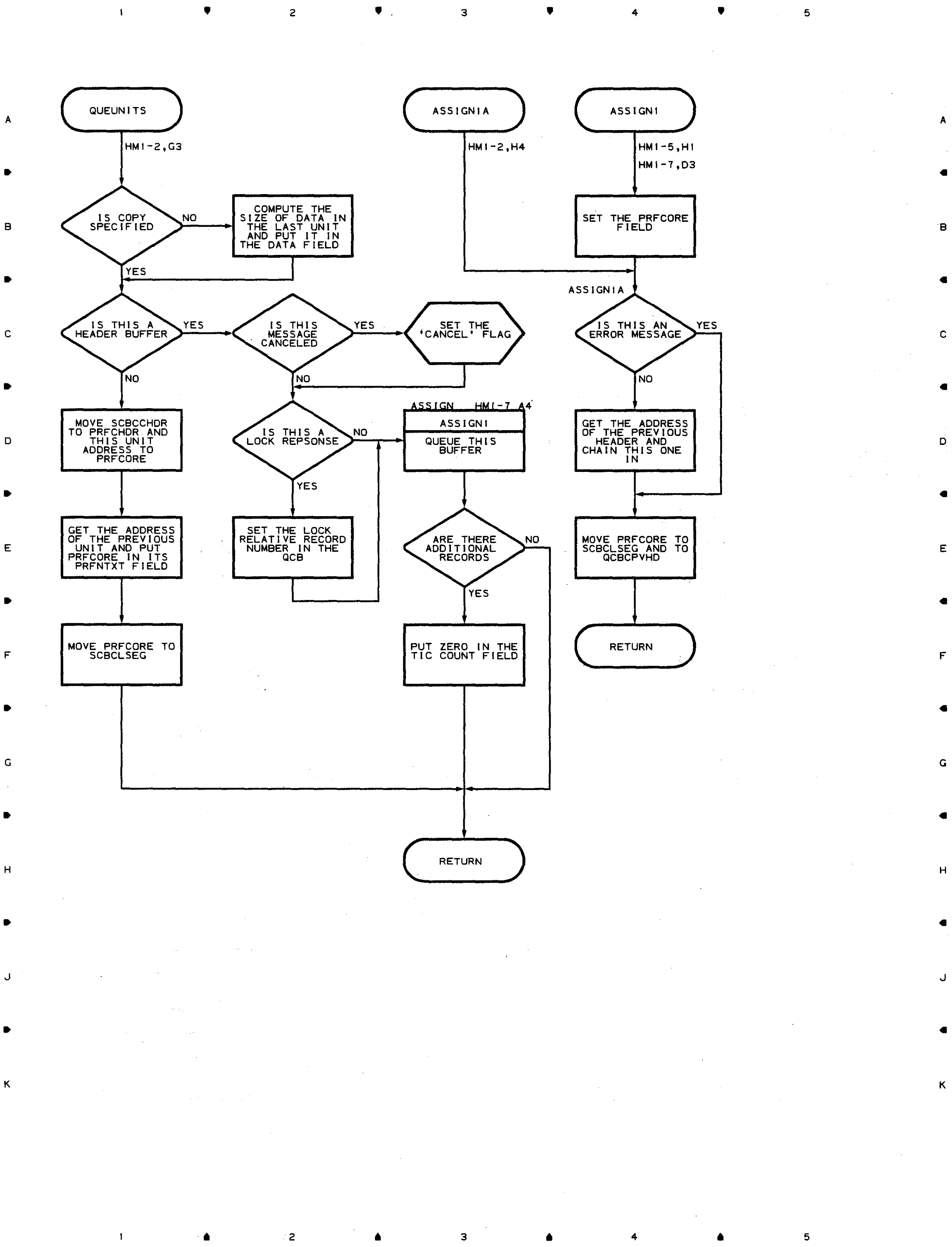
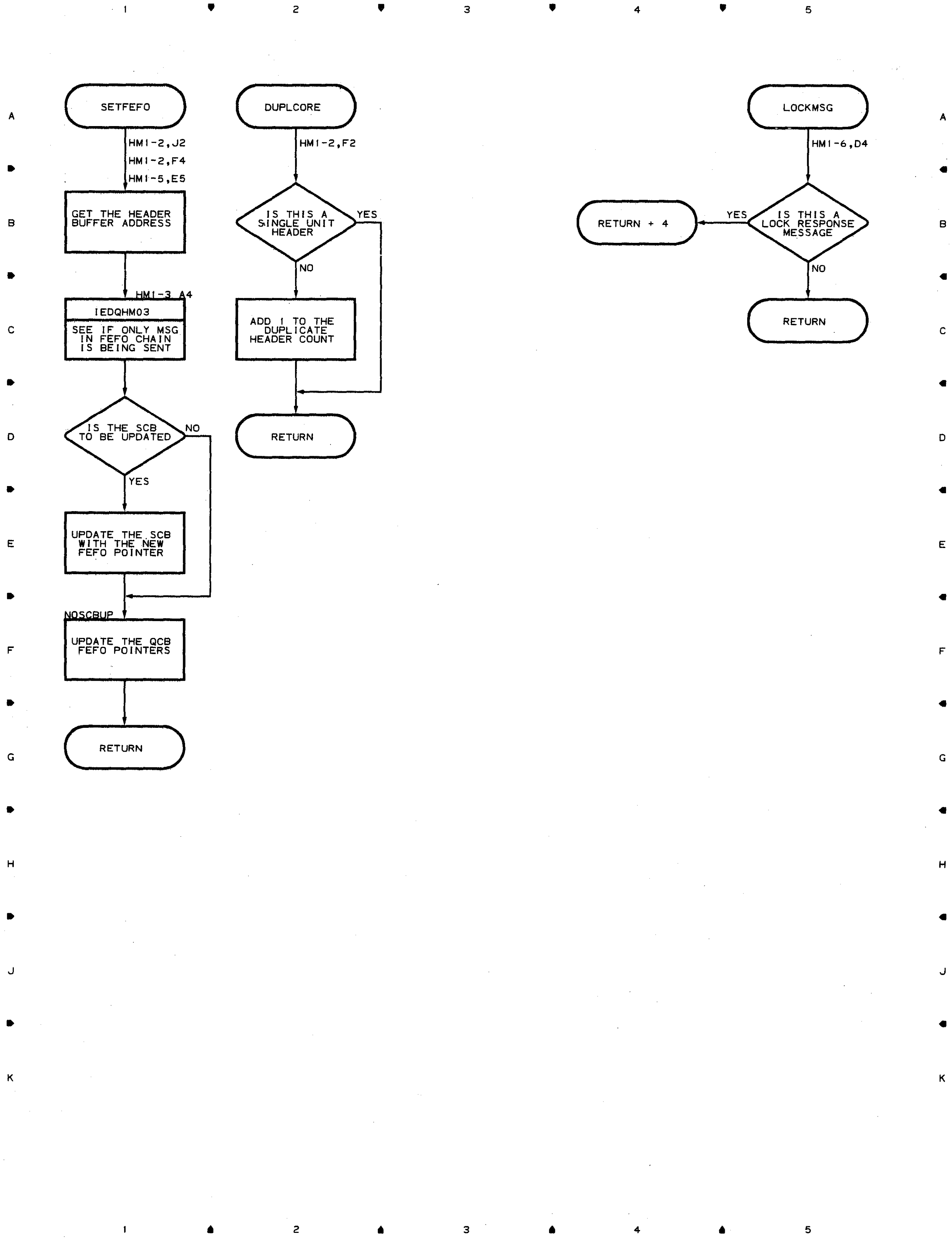

QUEUNITS
HM1-2,G3
IS COPY SPECIFIED
NO
COMPUTE THE SIZE OF DATA IN THE LAST UNIT AND PUT IT IN THE DATA FIELD
YES
IS THIS A HEADER BUFFER
YES
IS THIS MESSAGE CANCELED
YES
SET THE 'CANCEL' FLAG
NO
NO
MOVE SCBCCHDR TO PRFCHDR AND THIS UNIT ADDRESS TO PRFCORE
IS THIS A LOCK REPSONSE
NO
YES
GET THE ADDRESS OF THE PREVIOUS UNIT AND PUT PRFCORE IN ITS PRFNTXT FIELD
SET THE LOCK RELATIVE RECORD NUMBER IN THE QCB
MOVE PRFCORE TO SCBCLSEG
ASSIGNIA
HM1-2,H4
ASSIGN HM1-7 A4
ASSIGNI
QUEUE THIS BUFFER
ARE THERE ADDITIONAL RECORDS
NO
YES
PUT ZERO IN THE TIC COUNT FIELD
RETURN
ASSIGNI
HM1-5,H1
HM1-7,D3
SET THE PRFCORE FIELD
ASSIGNIA
IS THIS AN ERROR MESSAGE
YES
NO
GET THE ADDRESS OF THE PREVIOUS HEADER AND CHAIN THIS ONE IN
MOVE PRFCORE TO SCBCLSEG AND TO QCBCPVHD
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart HM1-8 DESTINATION SCHEDULER - MAIN STORAGE QUEUING ONLY



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
SETFEFO
HM1-2,J2
HM1-2,F4
HM1-5,E5
GET THE HEADER BUFFER ADDRESS
HM1-3 A4
IEDQHM03
SEE IF ONLY MSG IN FEFO CHAIN IS BEING SENT
IS THE SCB TO BE UPDATED
NO
YES
UPDATE THE SCB WITH THE NEW FEFO POINTER
NOSCBUP
UPDATE THE QCB FEFO POINTERS
RETURN
DUPLCORE
HM1-2,F2
IS THIS A SINGLE UNIT HEADER
YES
NO
ADD 1 TO THE DUPLICATE HEADER COUNT
RETURN
LOCKMSG
HM1-6,D4
IS THIS A LOCK RESPONSE MESSAGE
YES
RETURN + 4
NO
RETURN

Chart HM2-1 DESTINATION SCHEDULER - DISK QUEUING ONLY

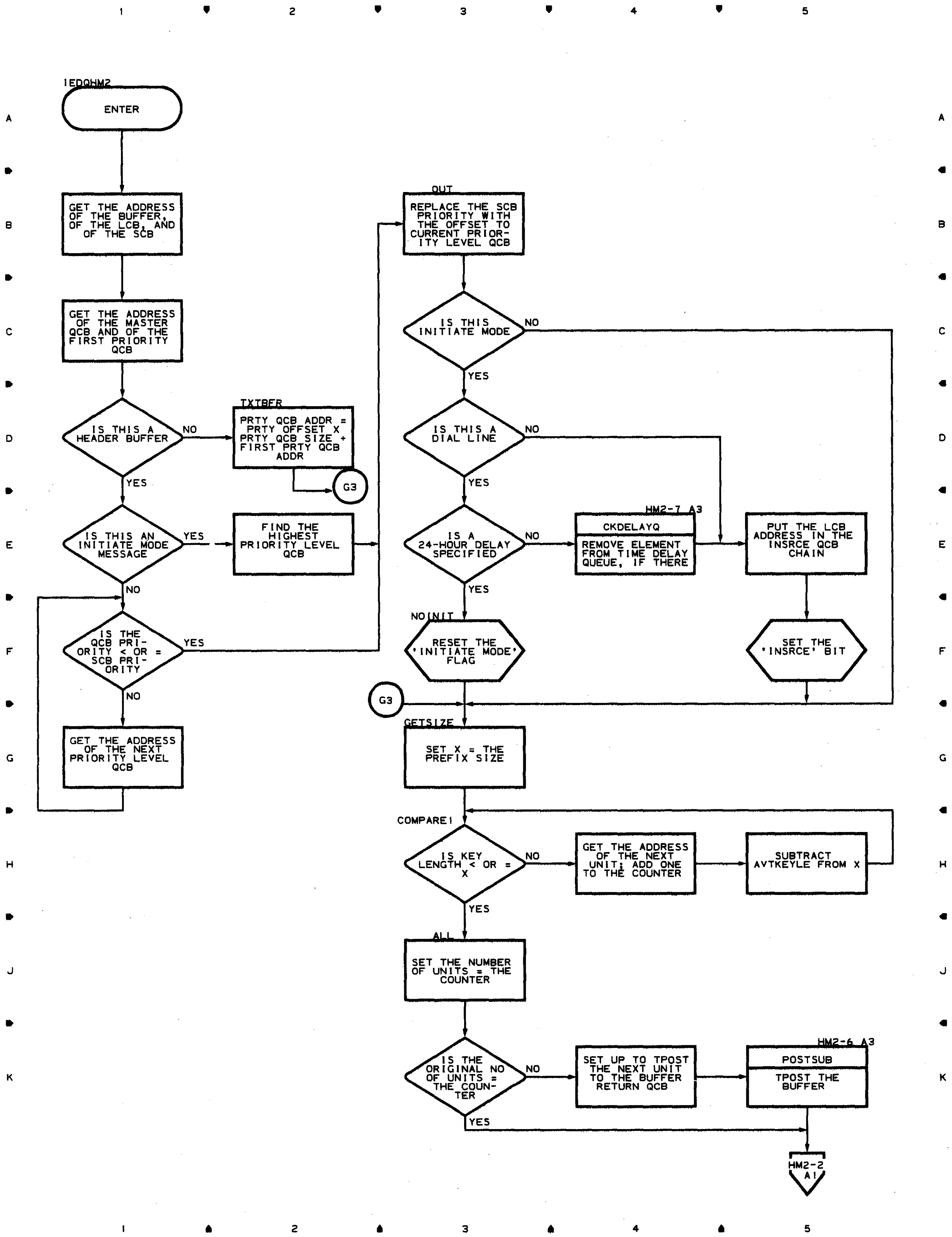
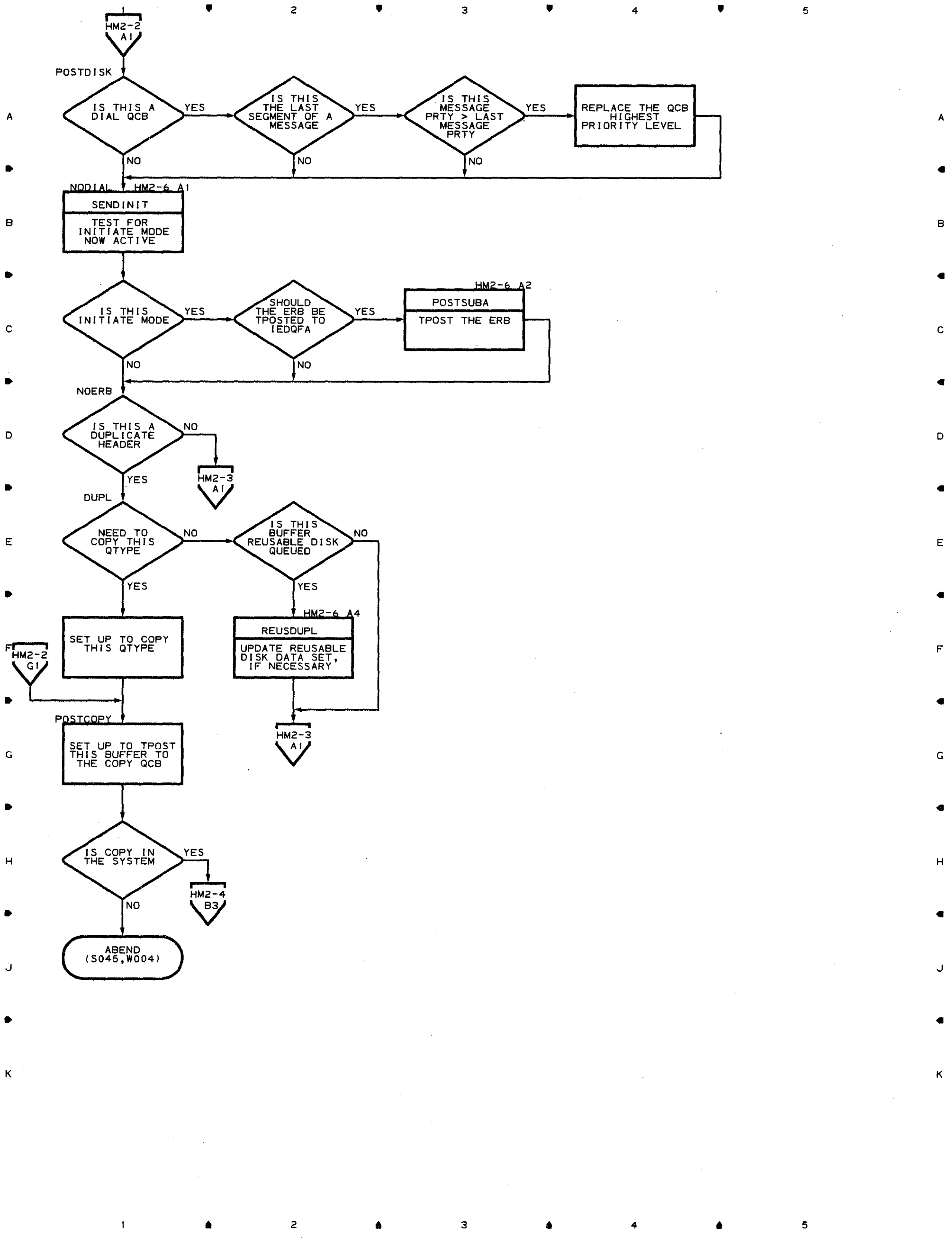

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQHM2
ENTER
GET THE ADDRESS OF THE BUFFER, OF THE LCB, AND OF THE SCB
GET THE ADDRESS OF THE MASTER QCB AND OF THE FIRST PRIORITY QCB
IS THIS A HEADER BUFFER
NO
YES
TXTBFR
PRTY QCB ADDR = PRTY OFFSET X PRTY QCB SIZE + FIRST PRTY QCB ADDR
G3
IS THIS AN INITIATE MODE MESSAGE
YES
NO
FIND THE HIGHEST PRIORITY LEVEL QCB
IS THE QCB PRIORITY < OR = SCB PRIORITY
YES
NO
GET THE ADDRESS OF THE NEXT PRIORITY LEVEL QCB
OUT
REPLACE THE SCB PRIORITY WITH THE OFFSET TO CURRENT PRIORITY LEVEL QCB
IS THIS INITIATE MODE
NO
YES
IS THIS A DIAL LINE
NO
YES
IS A 24-HOUR DELAY SPECIFIED
NO
YES
HM2-7 A3
CKDELAYQ
REMOVE ELEMENT FROM TIME DELAY QUEUE, IF THERE
PUT THE LCB ADDRESS IN THE INSRCE QCB CHAIN
NOINIT
RESET THE 'INITIATE MODE' FLAG
SET THE 'INSRCE' BIT
G3
GETSIZE
SET X = THE PREFIX SIZE
COMPARE1
IS KEY LENGTH < OR = X
NO
YES
GET THE ADDRESS OF THE NEXT UNIT; ADD ONE TO THE COUNTER
SUBTRACT AVTKEYLE FROM X
ALL
SET THE NUMBER OF UNITS = THE COUNTER
IS THE ORIGINAL NO OF UNITS = THE COUNTER
NO
YES
SET UP TO TPOST THE NEXT UNIT TO THE BUFFER RETURN QCB
HM2-6 A3
POSTSUB
TPOST THE BUFFER
HM2-2 A1

Chart HM2-2 DESTINATION SCHEDULER - DISK QUEUING ONLY



HM2-2 A1
POSTDISK
IS THIS A DIAL QCB
YES
IS THIS THE LAST SEGMENT OF A MESSAGE
YES
IS THIS MESSAGE PRTY > LAST MESSAGE PRTY
YES
REPLACE THE QCB HIGHEST PRIORITY LEVEL
NO
NO
NO
NODIAL
HM2-6 A1
SENDINIT
TEST FOR INITIATE MODE NOW ACTIVE
IS THIS INITIATE MODE
YES
SHOULD THE ERB BE TPOSTED TO IEDQFA
YES
HM2-6 A2
POSTSUBA
TPOST THE ERB
NO
NO
NOERB
IS THIS A DUPLICATE HEADER
NO
HM2-3 A1
YES
DUPL
NEED TO COPY THIS QTYPE
NO
IS THIS BUFFER REUSABLE DISK QUEUED
NO
YES
YES
SET UP TO COPY THIS QTYPE
HM2-6 A4
REUSDUPL
UPDATE REUSABLE DISK DATA SET, IF NECESSARY
HM2-2 G1
POSTCOPY
SET UP TO TPOST THIS BUFFER TO THE COPY QCB
HM2-3 A1
IS COPY IN THE SYSTEM
YES
HM2-4 B3
NO
ABEND (S045,W004)
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart HM2-3 DESTINATION SCHEDULER - DISK QUEUING ONLY

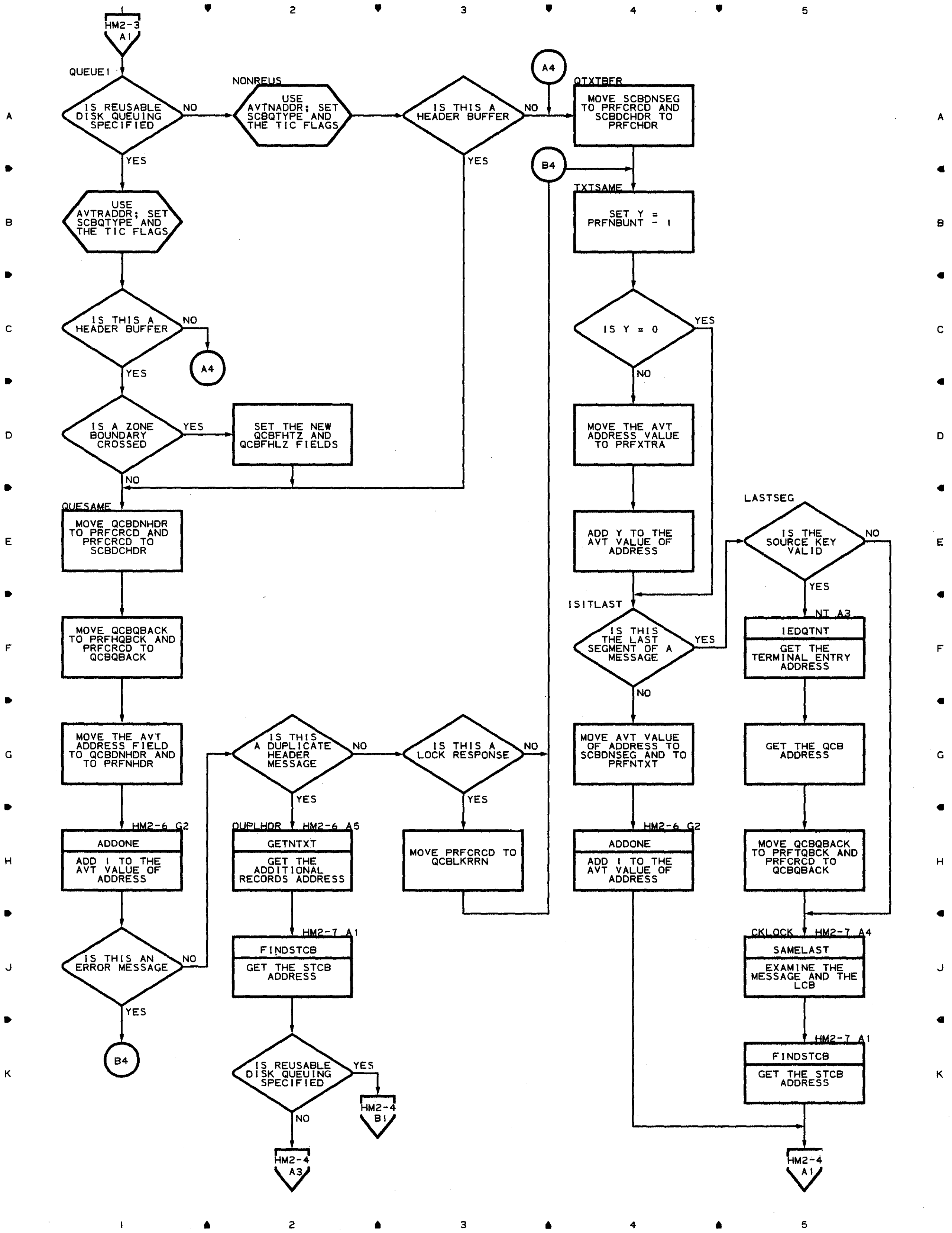
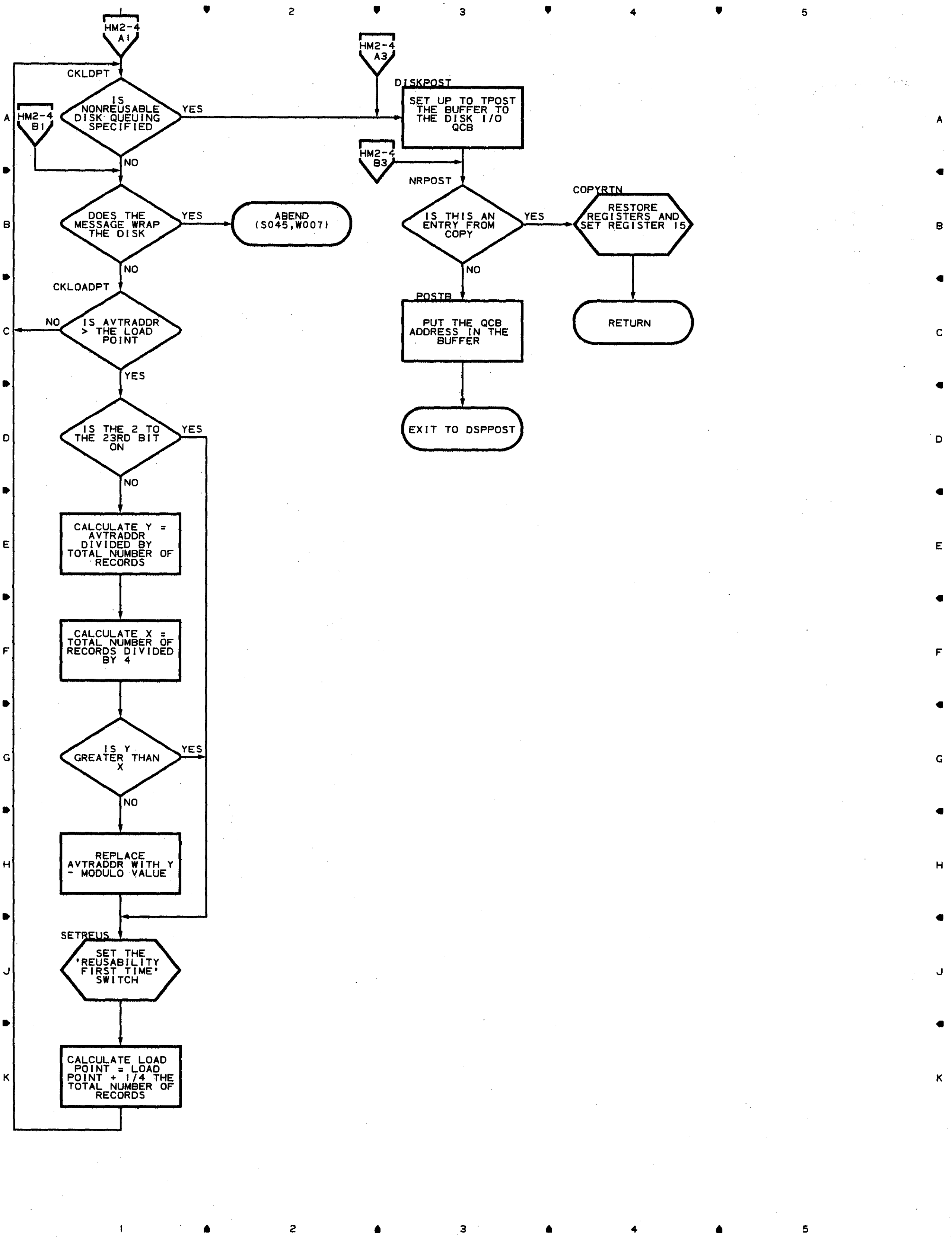

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
HM2-3 A1
QUEUE1
IS REUSABLE DISK QUEUING SPECIFIED
NO
YES
NONREUS
USE AVTNADDR; SET SCBQTYPE AND THE TIC FLAGS
IS THIS A HEADER BUFFER
NO
YES
A4
QTXTBFR
MOVE SCBDNSEG TO PRFCRCD AND SCBDCHDR TO PRFCHDR
B4
TXTSAME
SET Y = PRFNBUNT - 1
USE AVTRADDR; SET SCBQTYPE AND THE TIC FLAGS
IS THIS A HEADER BUFFER
NO
A4
YES
IS Y = 0
YES
NO
IS A ZONE BOUNDARY CROSSED
YES
SET THE NEW QCBFHTZ AND QCBFHLZ FIELDS
MOVE THE AVT ADDRESS VALUE TO PRFXTRA
NO
QUESAME
MOVE QCBDNHDR TO PRFCRCD AND PRFCRCD TO SCBDCHDR
ADD Y TO THE AVT VALUE OF ADDRESS
LASTSEG
IS THE SOURCE KEY VALID
NO
YES
ISITLAST
MOVE QCBQBACK TO PRFHQBCK AND PRFCRCD TO QCBQBACK
IS THIS THE LAST SEGMENT OF A MESSAGE
YES
NO
NT A3
IEDQTNT
GET THE TERMINAL ENTRY ADDRESS
MOVE THE AVT ADDRESS FIELD TO QCBDNHDR AND TO PRFNHDR
IS THIS A DUPLICATE HEADER MESSAGE
NO
YES
IS THIS A LOCK RESPONSE
NO
YES
MOVE AVT VALUE OF ADDRESS TO SCBDNSEG AND TO PRFNTXT
GET THE QCB ADDRESS
HM2-6 G2
ADDONE
ADD 1 TO THE AVT VALUE OF ADDRESS
DUPLHDR
HM2-6 A5
GETNTXT
GET THE ADDITIONAL RECORDS ADDRESS
MOVE PRFCRCD TO QCBLKRRN
HM2-6 G2
ADDONE
ADD 1 TO THE AVT VALUE OF ADDRESS
MOVE QCBQBACK TO PRFTQBCK AND PRFCRCD TO QCBQBACK
HM2-7 A1
FINDSTCB
GET THE STCB ADDRESS
CKLOCK
HM2-7 A4
SAMELAST
EXAMINE THE MESSAGE AND THE LCB
IS THIS AN ERROR MESSAGE
NO
YES
B4
IS REUSABLE DISK QUEUING SPECIFIED
YES
NO
HM2-4 B1
HM2-4 A3
HM2-7 A1
FINDSTCB
GET THE STCB ADDRESS
HM2-4 A1

Chart HM2-4 DESTINATION SCHEDULER - DISK QUEUING ONLY



1
2
3
4
5
HM2-4 A1
CKLDPT
IS NONREUSABLE DISK QUEUING SPECIFIED
YES
NO
HM2-4 B1
DOES THE MESSAGE WRAP THE DISK
YES
ABEND (S045,W007)
NO
CKLOADPT
IS AVTRADDR > THE LOAD POINT
NO
YES
IS THE 2 TO THE 23RD BIT ON
YES
NO
CALCULATE Y = AVTRADDR DIVIDED BY TOTAL NUMBER OF RECORDS
CALCULATE X = TOTAL NUMBER OF RECORDS DIVIDED BY 4
IS Y GREATER THAN X
YES
NO
REPLACE AVTRADDR WITH Y - MODULO VALUE
SETREUS
SET THE 'REUSABILITY FIRST TIME' SWITCH
CALCULATE LOAD POINT = LOAD POINT + 1/4 THE TOTAL NUMBER OF RECORDS
HM2-4 A3
DISKPOST
SET UP TO TPOST THE BUFFER TO THE DISK I/O QCB
HM2-4 B3
NRPOST
IS THIS AN ENTRY FROM COPY
YES
NO
COPYRTN
RESTORE REGISTERS AND SET REGISTER 15
RETURN
POSTB
PUT THE QCB ADDRESS IN THE BUFFER
EXIT TO DSPPOST
A
B
C
D
E
F
G
H
J
K

Chart HM2-5 DESTINATION SCHEDULER - DISK QUEUING ONLY

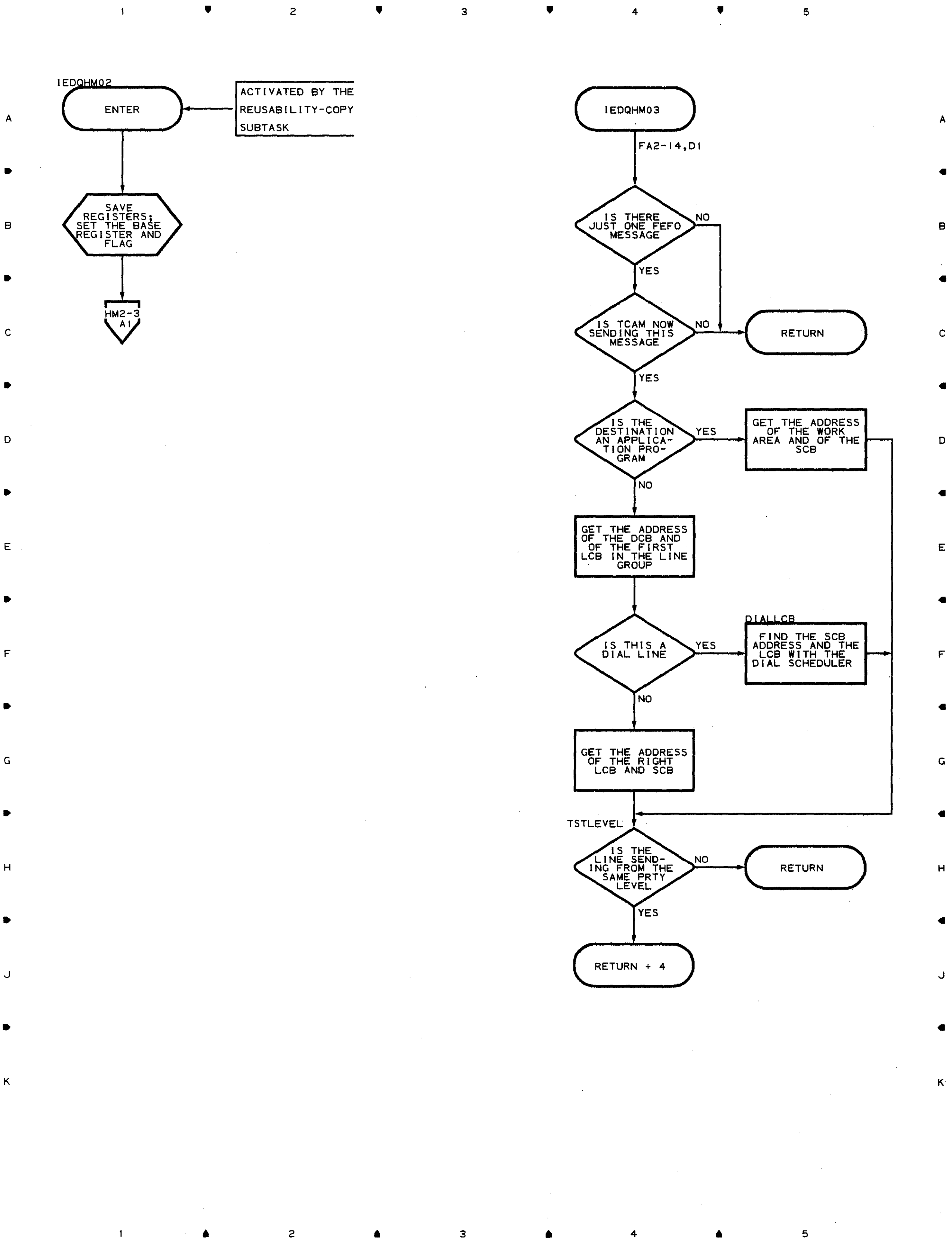

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQHM02
ENTER
ACTIVATED BY THE REUSABILITY-COPY SUBTASK
SAVE REGISTERS; SET THE BASE REGISTER AND FLAG
HM2-3 A1
IEDQHM03
FA2-14,D1
IS THERE JUST ONE FEFO MESSAGE
NO
YES
IS TCAM NOW SENDING THIS MESSAGE
NO
RETURN
YES
IS THE DESTINATION AN APPLICATION PROGRAM
YES
GET THE ADDRESS OF THE WORK AREA AND OF THE SCB
NO
GET THE ADDRESS OF THE DCB AND OF THE FIRST LCB IN THE LINE GROUP
DIALLCB
IS THIS A DIAL LINE
YES
FIND THE SCB ADDRESS AND THE LCB WITH THE DIAL SCHEDULER
NO
GET THE ADDRESS OF THE RIGHT LCB AND SCB
TSTLEVEL
IS THE LINE SENDING FROM THE SAME PRTY LEVEL
NO
RETURN
YES
RETURN + 4

Chart HM2-6 DESTINATION SCHEDULER - DISK QUEUING ONLY

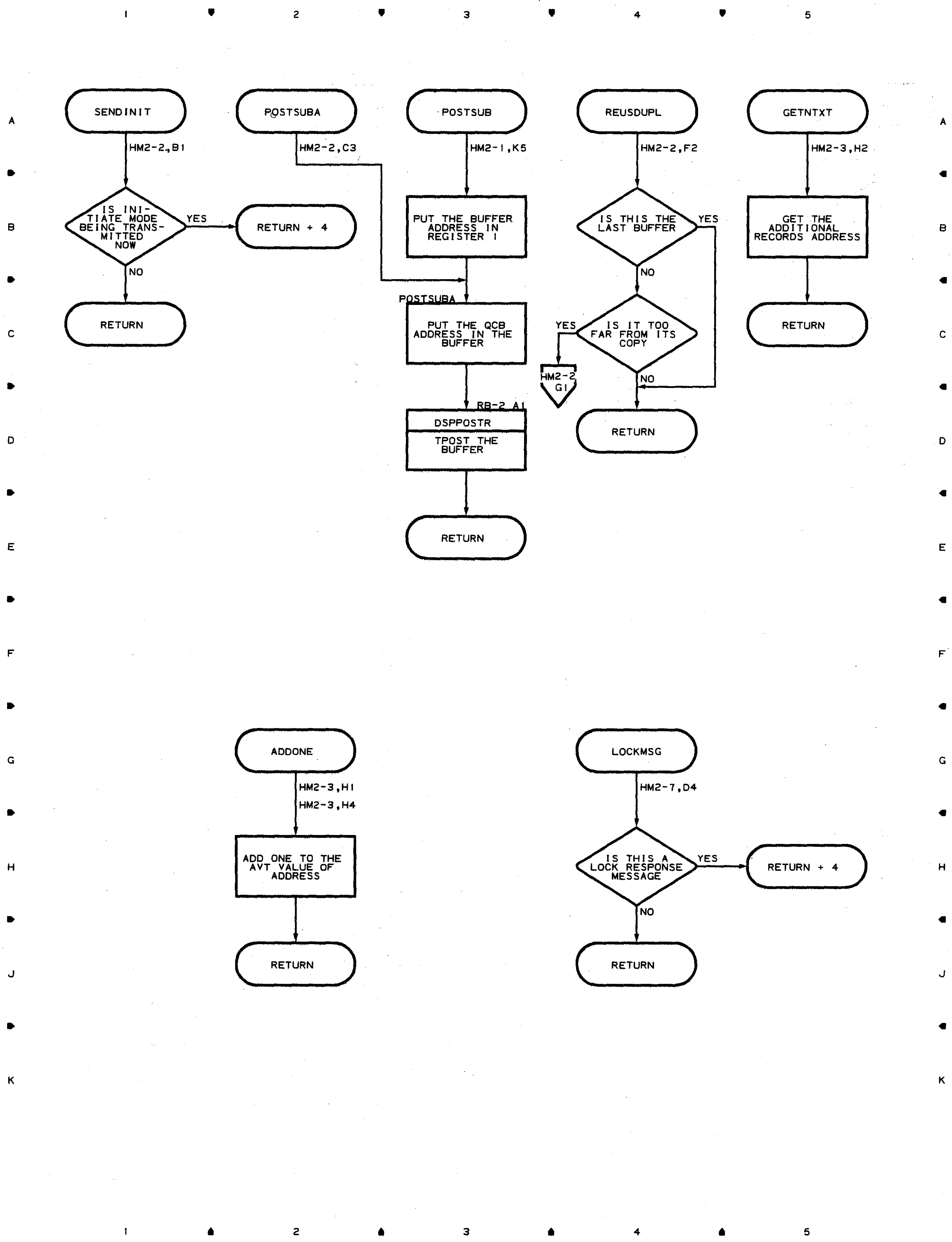
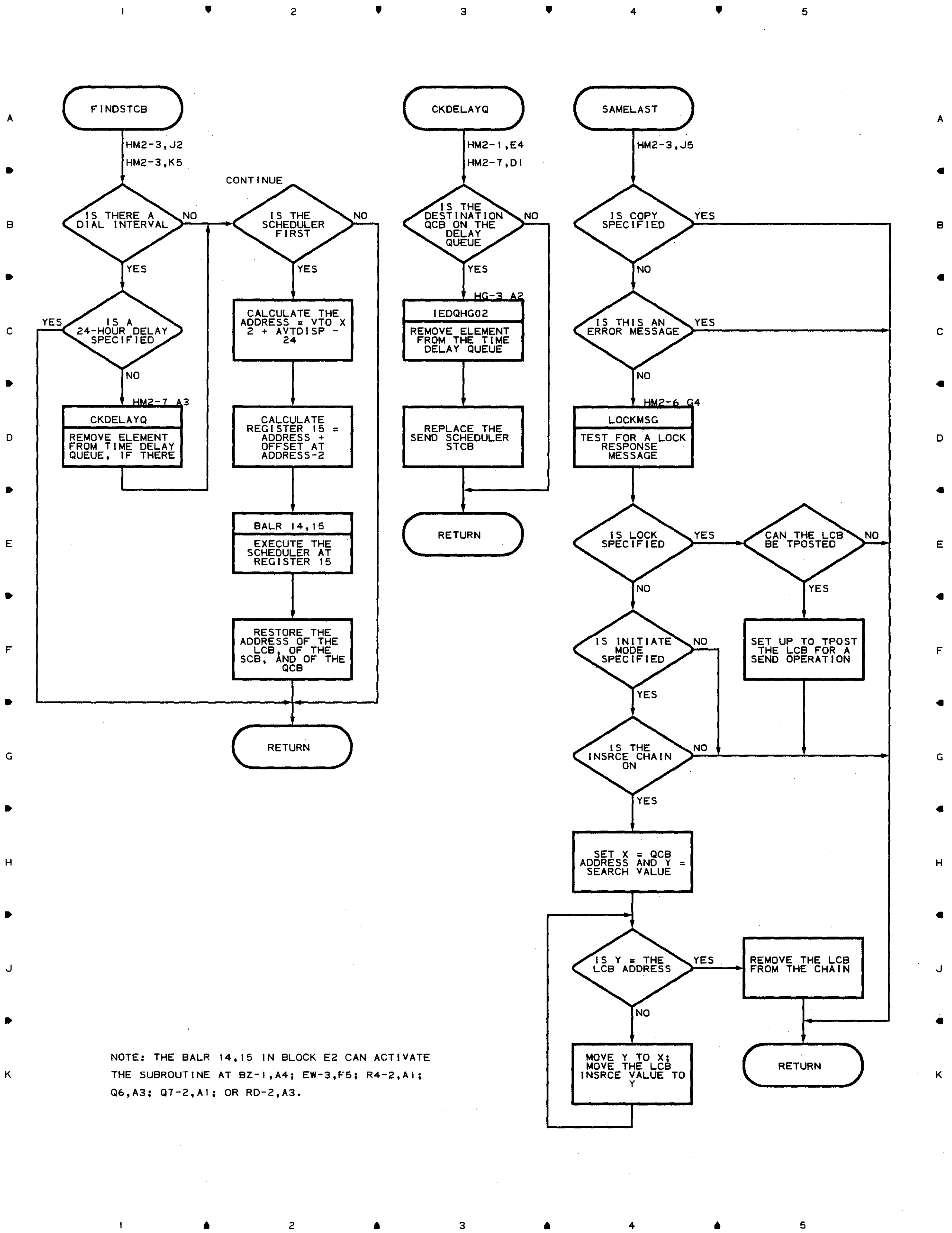


Chart HM2-7 DESTINATION SCHEDULER - DISK QUEUING ONLY


FINDSTCB
HM2-3,J2
HM2-3,K5
IS THERE A DIAL INTERVAL
NO
YES
IS A 24-HOUR DELAY SPECIFIED
YES
NO
HM2-7 A3
CKDELAYQ
REMOVE ELEMENT FROM TIME DELAY QUEUE, IF THERE
CONTINUE
IS THE SCHEDULER FIRST
NO
YES
CALCULATE THE ADDRESS = VTO X 2 + AVTDISP - 24
CALCULATE REGISTER 15 = ADDRESS + OFFSET AT ADDRESS-2
BALR 14,15
EXECUTE THE SCHEDULER AT REGISTER 15
RESTORE THE ADDRESS OF THE LCB, OF THE SCB, AND OF THE QCB
RETURN
CKDELAYQ
HM2-1,E4
HM2-7,D1
IS THE DESTINATION QCB ON THE DELAY QUEUE
NO
YES
HG-3 A2
IEDQHG02
REMOVE ELEMENT FROM THE TIME DELAY QUEUE
REPLACE THE SEND SCHEDULER STCB
RETURN
SAMELAST
HM2-3,J5
IS COPY SPECIFIED
YES
NO
IS THIS AN ERROR MESSAGE
YES
NO
HM2-6 G4
LOCKMSG
TEST FOR A LOCK RESPONSE MESSAGE
IS LOCK SPECIFIED
YES
NO
CAN THE LCB BE TPOSTED
NO
YES
SET UP TO TPOST THE LCB FOR A SEND OPERATION
IS INITIATE MODE SPECIFIED
NO
YES
IS THE INSRCE CHAIN ON
NO
YES
SET X = QCB ADDRESS AND Y = SEARCH VALUE
IS Y = THE LCB ADDRESS
YES
NO
REMOVE THE LCB FROM THE CHAIN
MOVE Y TO X; MOVE THE LCB INSRCE VALUE TO Y
RETURN
1 2 3 4 5
A B C D E F G H J K

NOTE: THE BALR 14,15 IN BLOCK E2 CAN ACTIVATE THE SUBROUTINE AT BZ-1,A4; EW-3,F5; R4-2,A1; Q6,A3; Q7-2,A1; OR RD-2,A3.

Chart JC-1 START/STOP ERP CONTROL MODULE

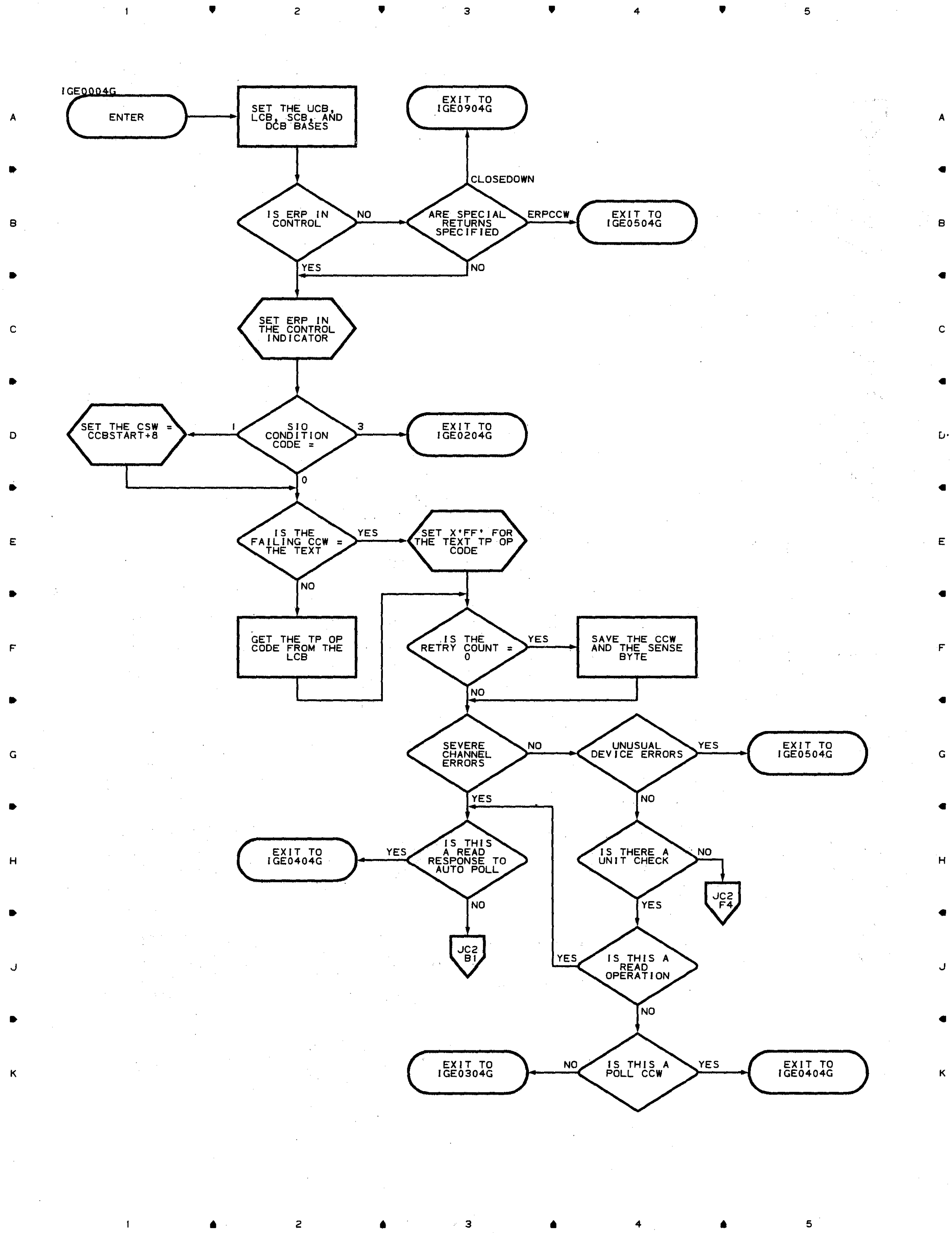


Chart JC-2 START/STOP ERP CONTROL MODULE

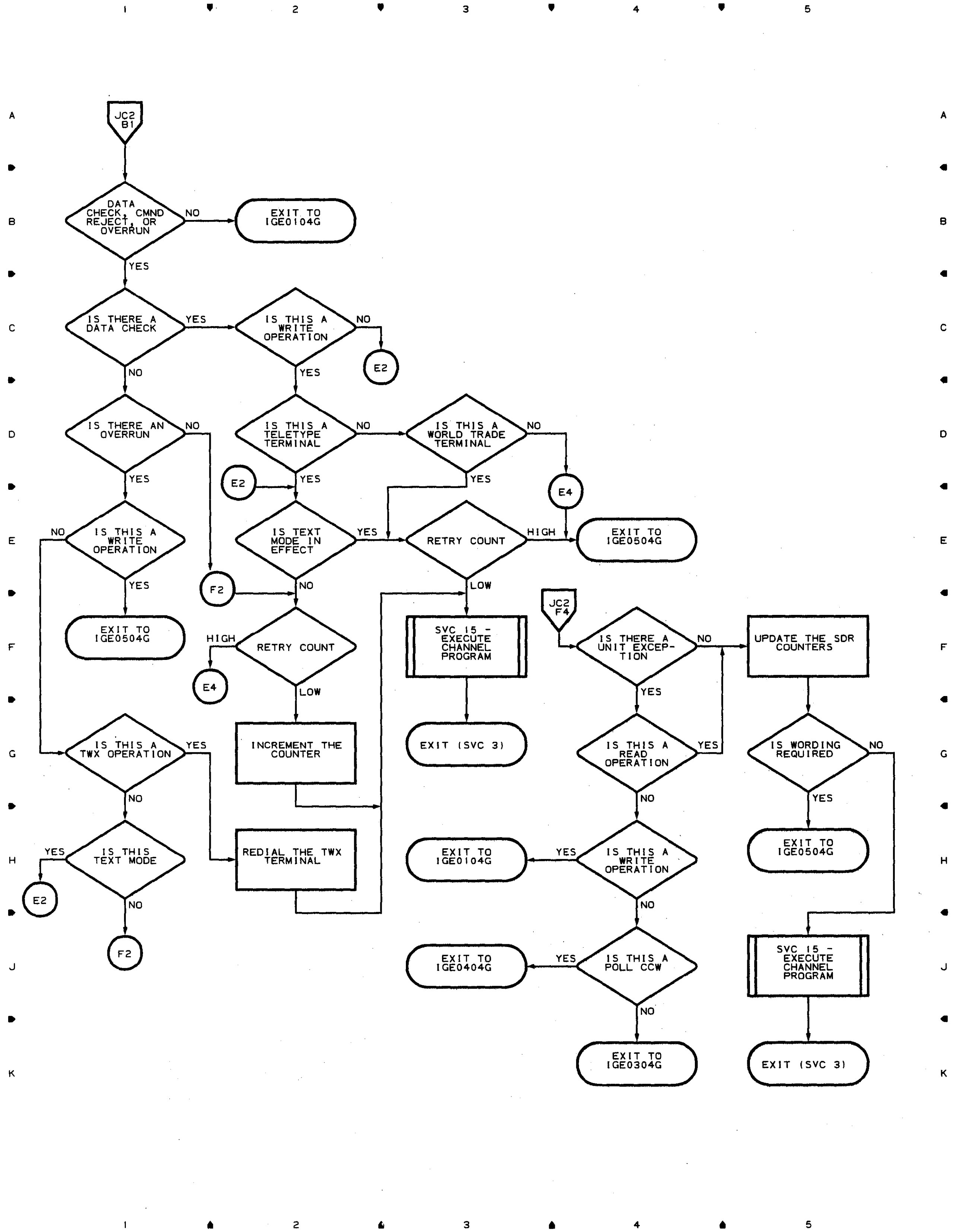
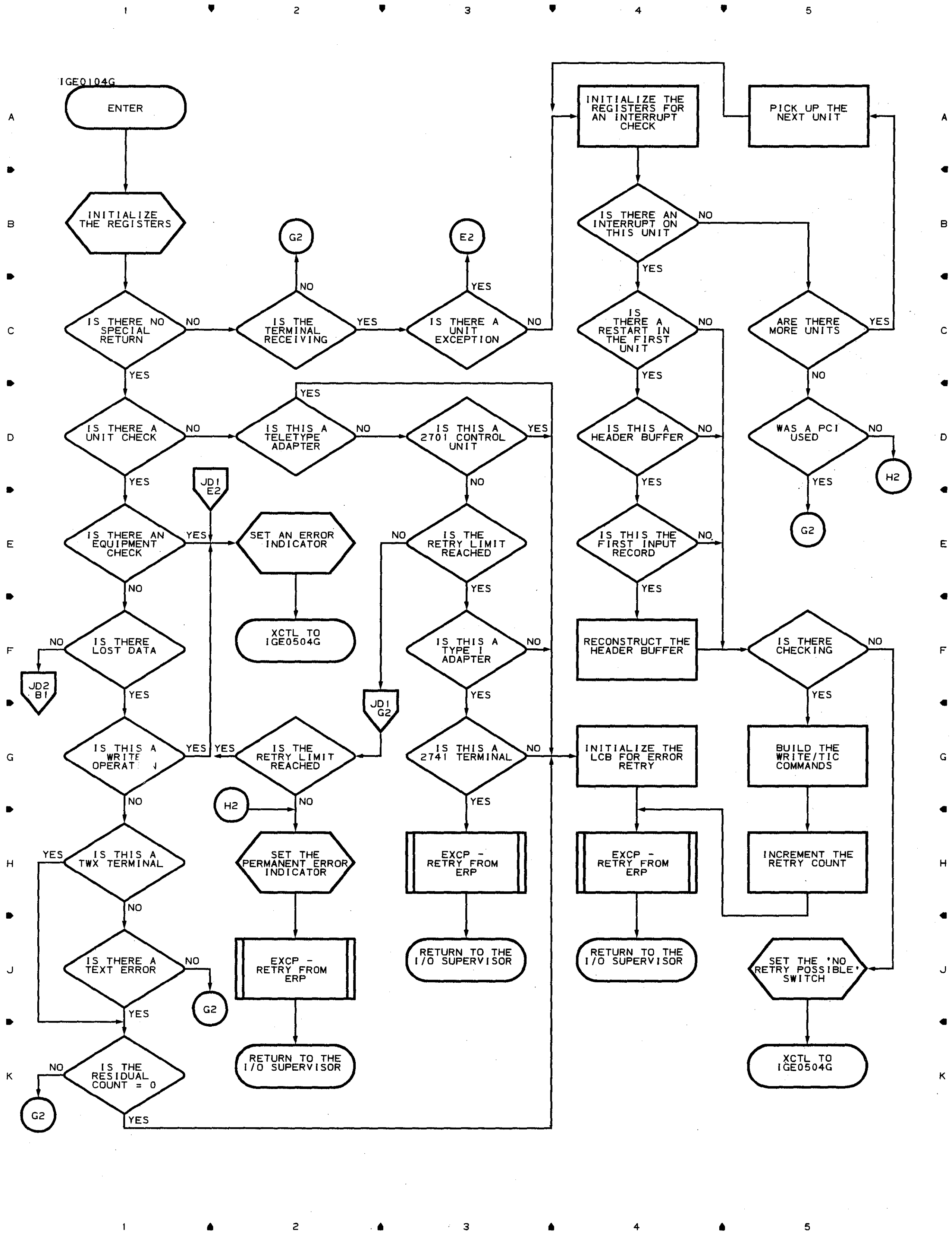


Chart JD-1 READ/WRITE UNIT CHECK AND UNIT EXCEPTION ERP MODULE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGE0104G
ENTER
INITIALIZE THE REGISTERS
IS THERE NO SPECIAL RETURN
NO
YES
IS THE TERMINAL RECEIVING
NO
G2
YES
IS THERE A UNIT EXCEPTION
YES
E2
NO
INITIALIZE THE REGISTERS FOR AN INTERRUPT CHECK
PICK UP THE NEXT UNIT
IS THERE AN INTERRUPT ON THIS UNIT
NO
YES
IS THERE A RESTART IN THE FIRST UNIT
NO
YES
ARE THERE MORE UNITS
YES
NO
IS THERE A UNIT CHECK
NO
YES
IS THIS A TELETYPE ADAPTER
YES
NO
IS THIS A 2701 CONTROL UNIT
YES
NO
IS THIS A HEADER BUFFER
NO
YES
WAS A PCI USED
NO
H2
YES
G2
JD1 E2
IS THERE AN EQUIPMENT CHECK
YES
NO
SET AN ERROR INDICATOR
XCTL TO IGE0504G
IS THE RETRY LIMIT REACHED
NO
YES
IS THIS THE FIRST INPUT RECORD
NO
YES
RECONSTRUCT THE HEADER BUFFER
IS THERE LOST DATA
NO
JD2 B1
YES
IS THIS A TYPE I ADAPTER
NO
YES
IS THERE CHECKING
NO
YES
JD1 G2
IS THIS A WRITE OPERATION
YES
NO
IS THE RETRY LIMIT REACHED
YES
NO
IS THIS A 2741 TERMINAL
NO
YES
INITIALIZE THE LCB FOR ERROR RETRY
BUILD THE WRITE/TIC COMMANDS
H2
SET THE PERMANENT ERROR INDICATOR
EXCP - RETRY FROM ERP
EXCP - RETRY FROM ERP
INCREMENT THE RETRY COUNT
IS THIS A TWX TERMINAL
YES
NO
IS THERE A TEXT ERROR
NO
G2
YES
EXCP - RETRY FROM ERP
RETURN TO THE I/O SUPERVISOR
RETURN TO THE I/O SUPERVISOR
SET THE 'NO RETRY POSSIBLE' SWITCH
IS THE RESIDUAL COUNT = 0
NO
G2
YES
RETURN TO THE I/O SUPERVISOR
XCTL TO IGE0504G

Chart JD-2 READ/WRITE UNIT CHECK AND UNIT EXCEPTION ERP MODULE

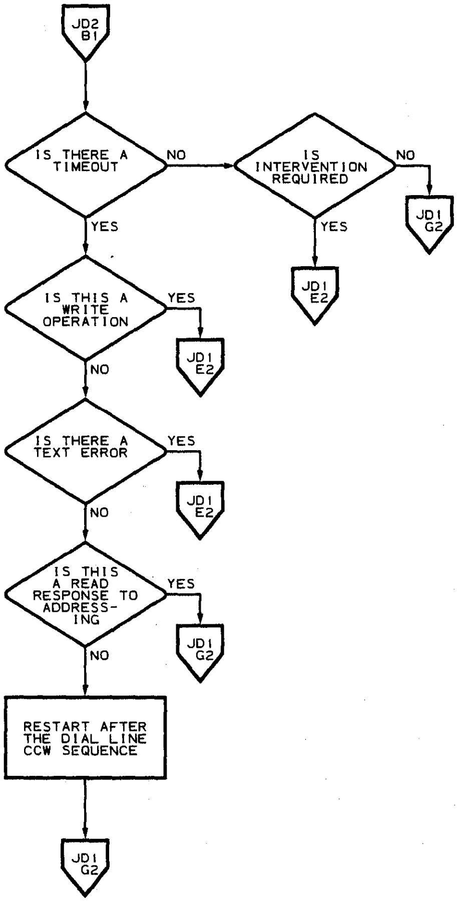
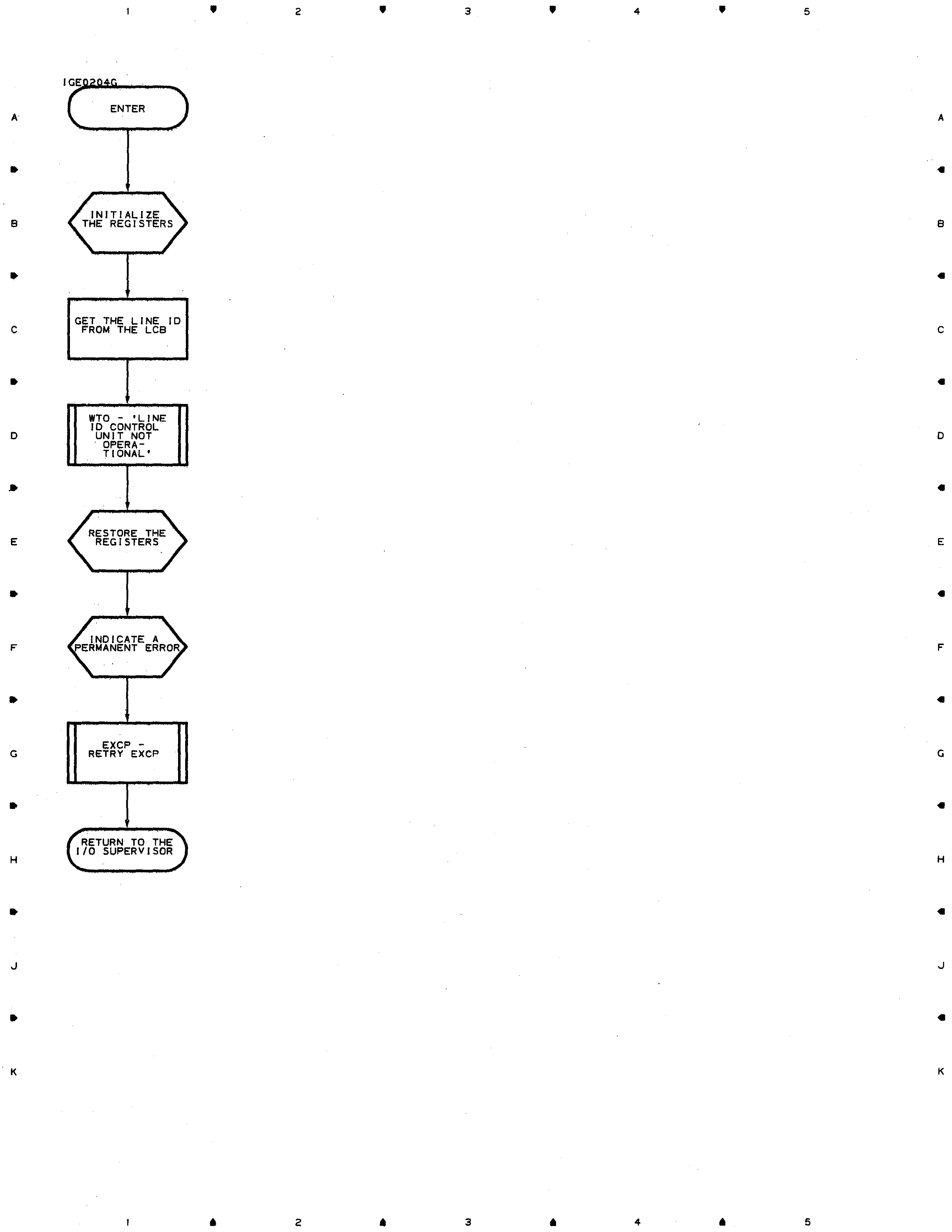
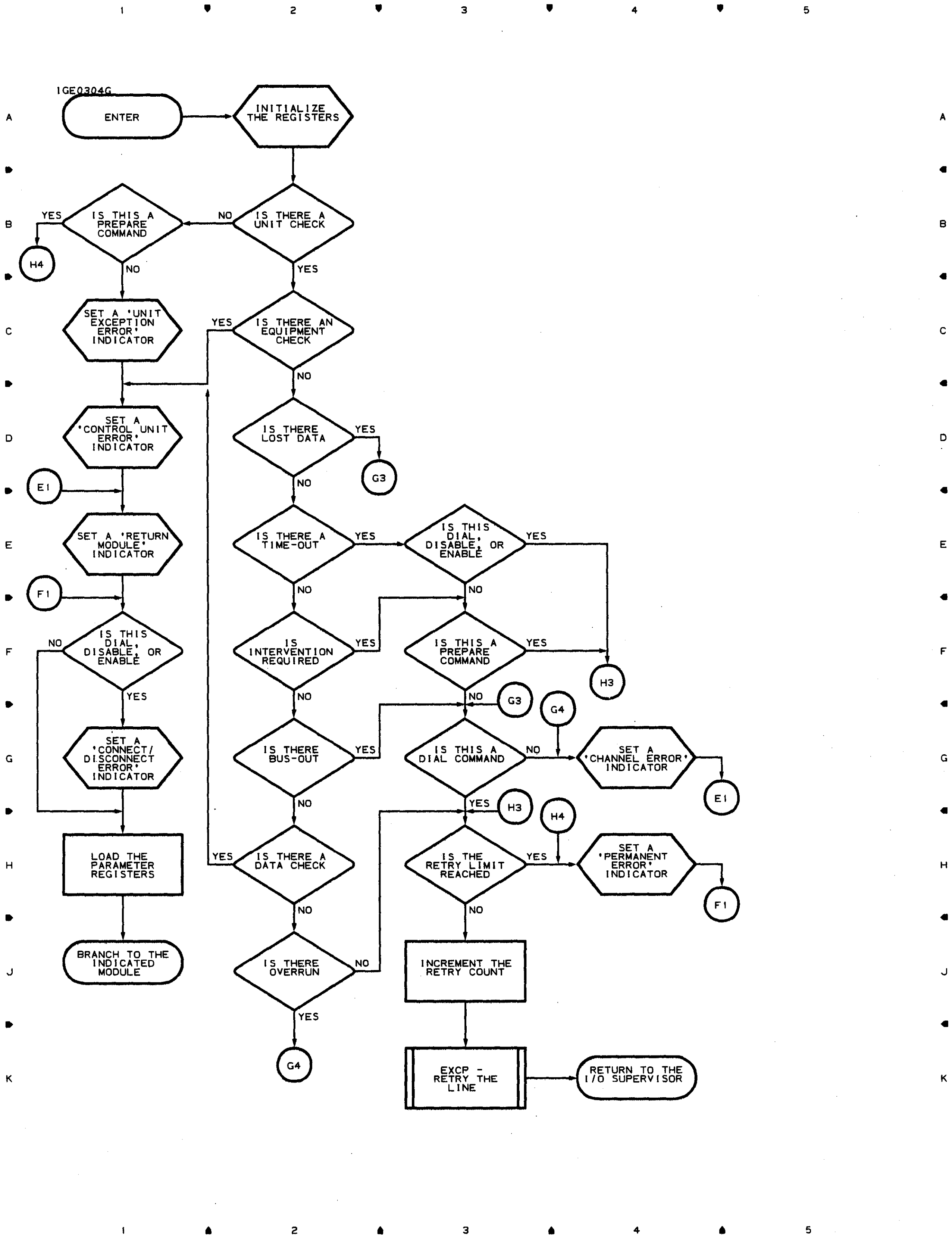


Chart JE NON-OPERATIONAL CONTROL UNIT ERP MODULE



1
2
3
4
5
IGE0204G
ENTER
INITIALIZE THE REGISTERS
GET THE LINE ID FROM THE LCB
WTO - 'LINE ID CONTROL UNIT NOT OPERA- TIONAL'
RESTORE THE REGISTERS
INDICATE A PERMANENT ERROR
EXCP - RETRY EXCP
RETURN TO THE I/O SUPERVISOR
A
B
C
D
E
F
G
H
J
K

Chart JF UNIT CHECK FOR NON-READ, NON-WRITE, AND NON-POLL CCW'S ERP MODULE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGE0304G
ENTER
INITIALIZE THE REGISTERS
IS THERE A UNIT CHECK
NO
YES
IS THIS A PREPARE COMMAND
YES
H4
NO
SET A 'UNIT EXCEPTION ERROR' INDICATOR
IS THERE AN EQUIPMENT CHECK
YES
NO
SET A 'CONTROL UNIT ERROR' INDICATOR
IS THERE LOST DATA
YES
G3
NO
E1
SET A 'RETURN MODULE' INDICATOR
IS THERE A TIME-OUT
YES
IS THIS DIAL, DISABLE, OR ENABLE
YES
NO
NO
F1
IS THIS DIAL, DISABLE, OR ENABLE
NO
YES
IS INTERVENTION REQUIRED
YES
IS THIS A PREPARE COMMAND
YES
H3
NO
NO
G3
G4
SET A 'CONNECT/ DISCONNECT ERROR' INDICATOR
IS THERE BUS-OUT
YES
IS THIS A DIAL COMMAND
NO
SET A 'CHANNEL ERROR' INDICATOR
E1
NO
YES
H3
H4
LOAD THE PARAMETER REGISTERS
IS THERE A DATA CHECK
YES
IS THE RETRY LIMIT REACHED
YES
SET A 'PERMANENT ERROR' INDICATOR
F1
NO
NO
BRANCH TO THE INDICATED MODULE
IS THERE OVERRUN
NO
INCREMENT THE RETRY COUNT
YES
G4
EXCP - RETRY THE LINE
RETURN TO THE I/O SUPERVISOR

Chart JG AUTO-POLL AND READ-RESPONSE-TO-POLL UNIT CHECK AND UNIT EXCEPTION ERP MODULE

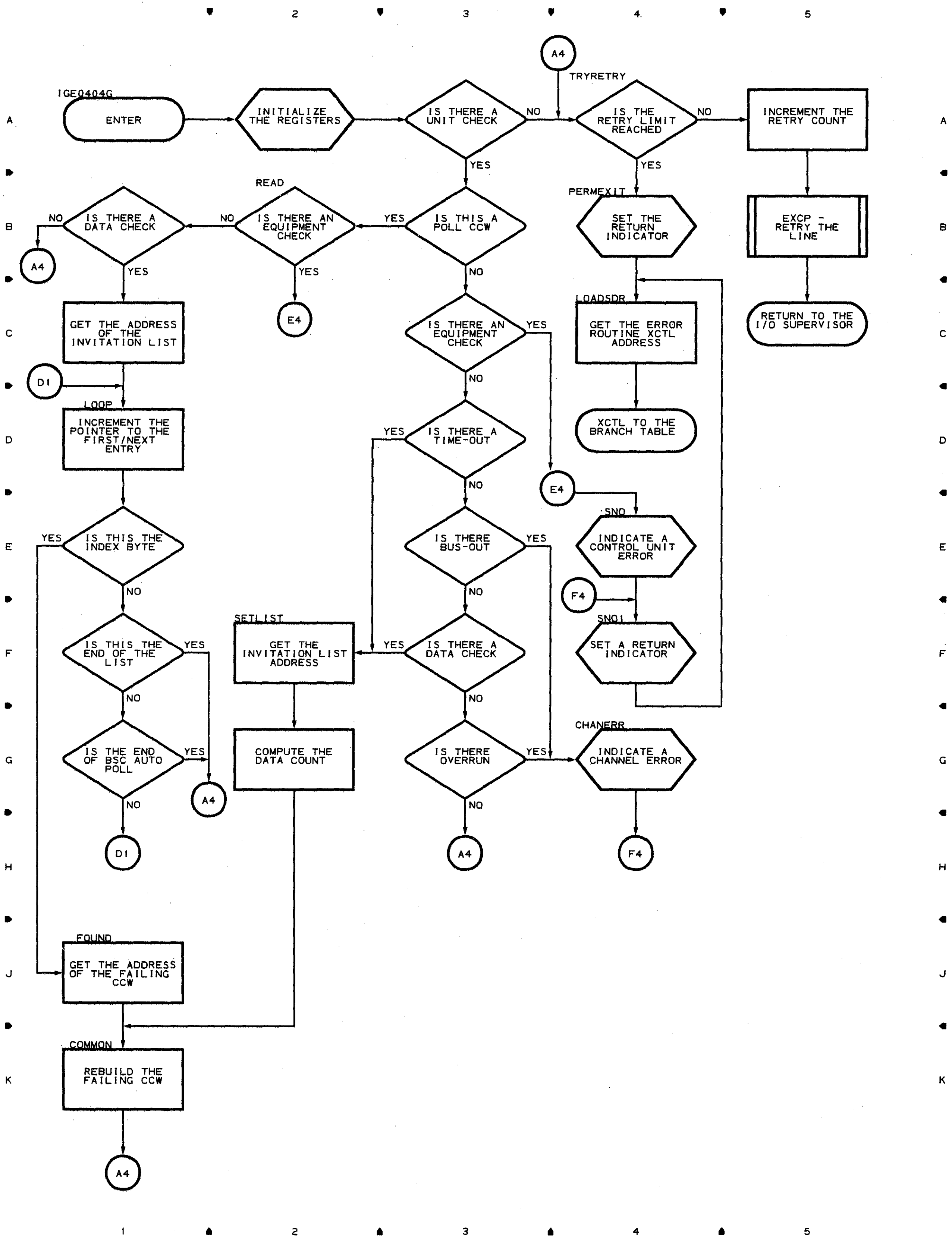


Chart JH ERROR POST AND SECOND LEVEL CCW RETURN MODULE

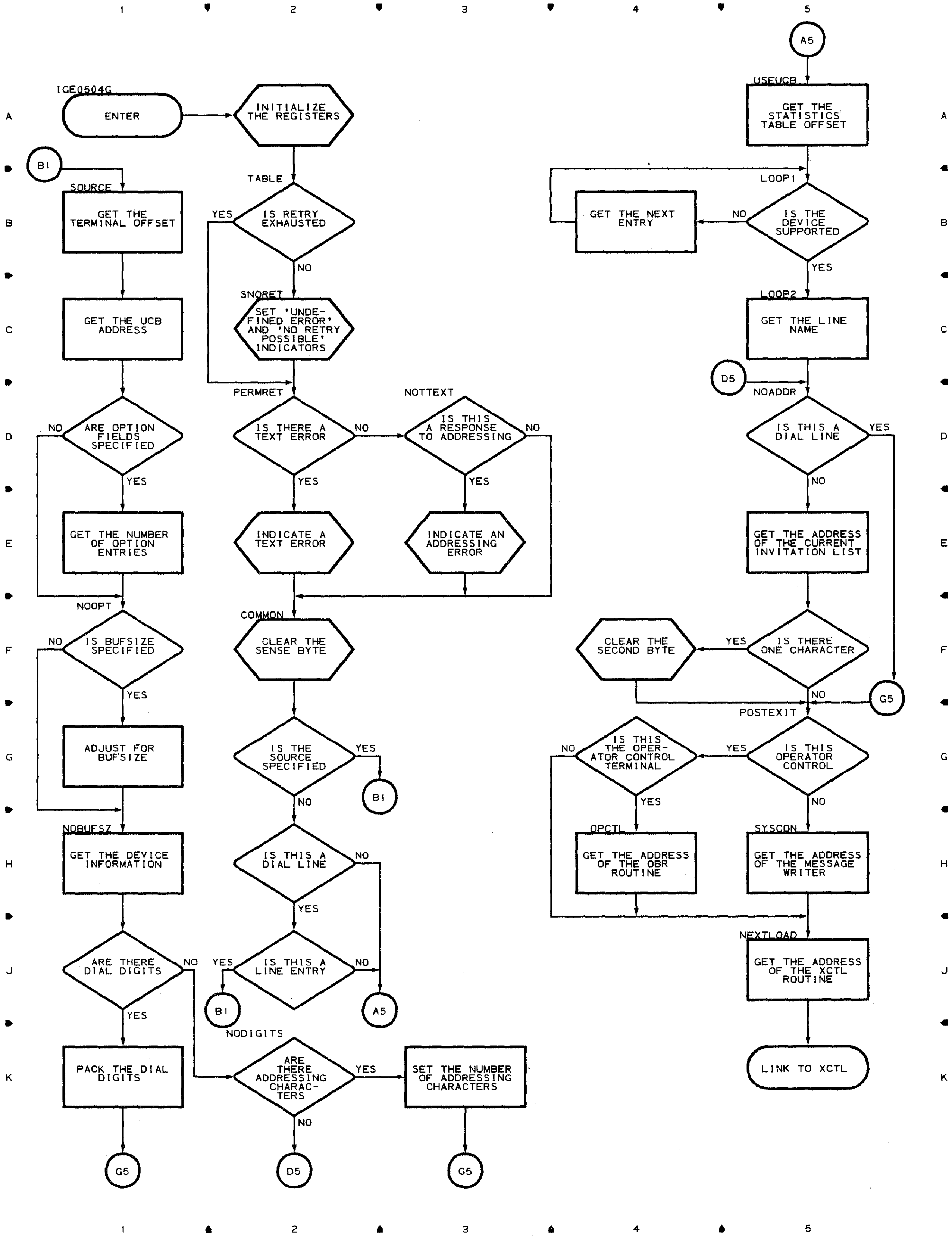


Chart JI UNIT CHECK AND UNIT EXCEPTION ON READ/WRITE CCW'S FOR AUDIO AND 2260 LOCAL DEVICES ERP MODULE

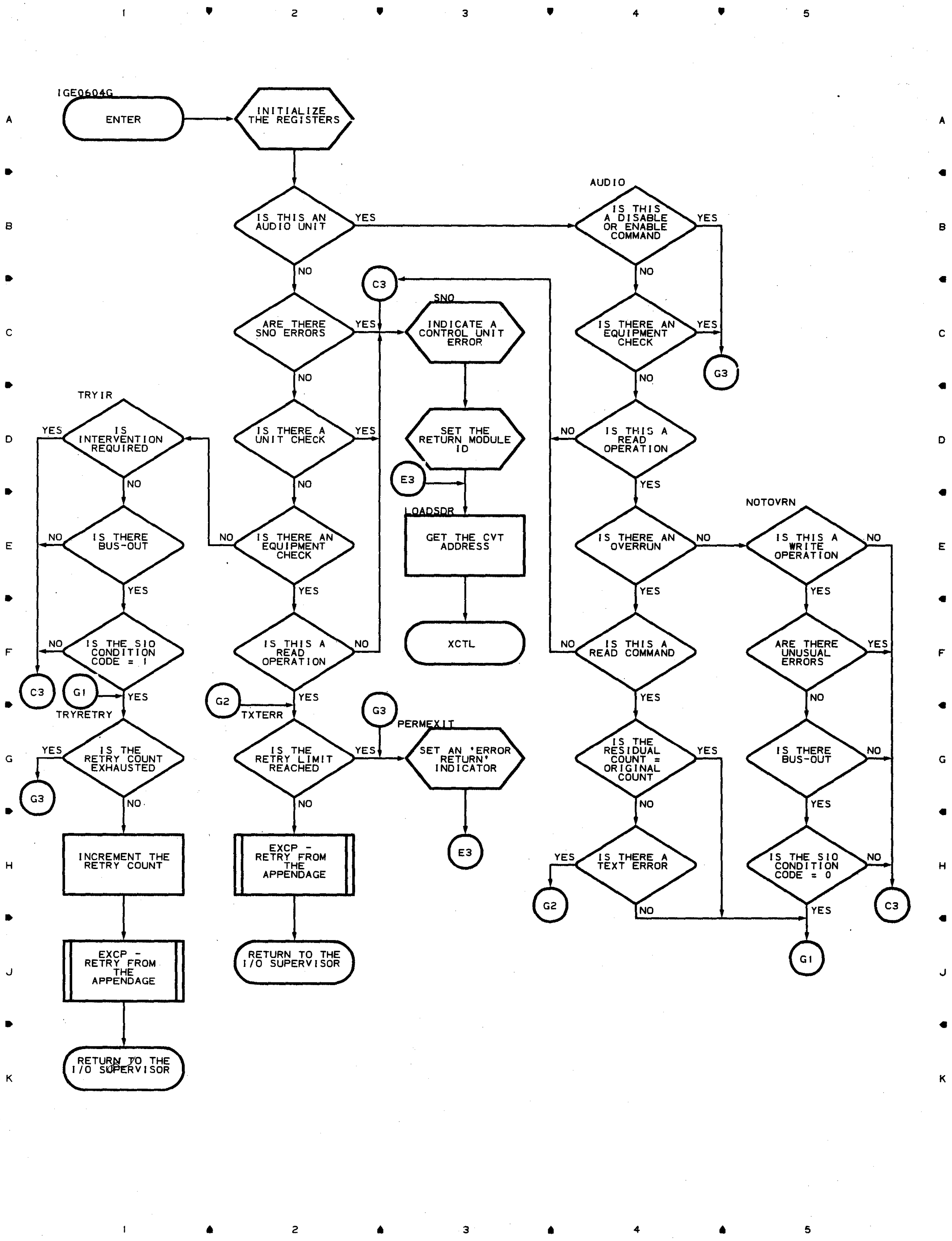


Chart JJ START/STOP CHANNEL CHECK ERP MODULE

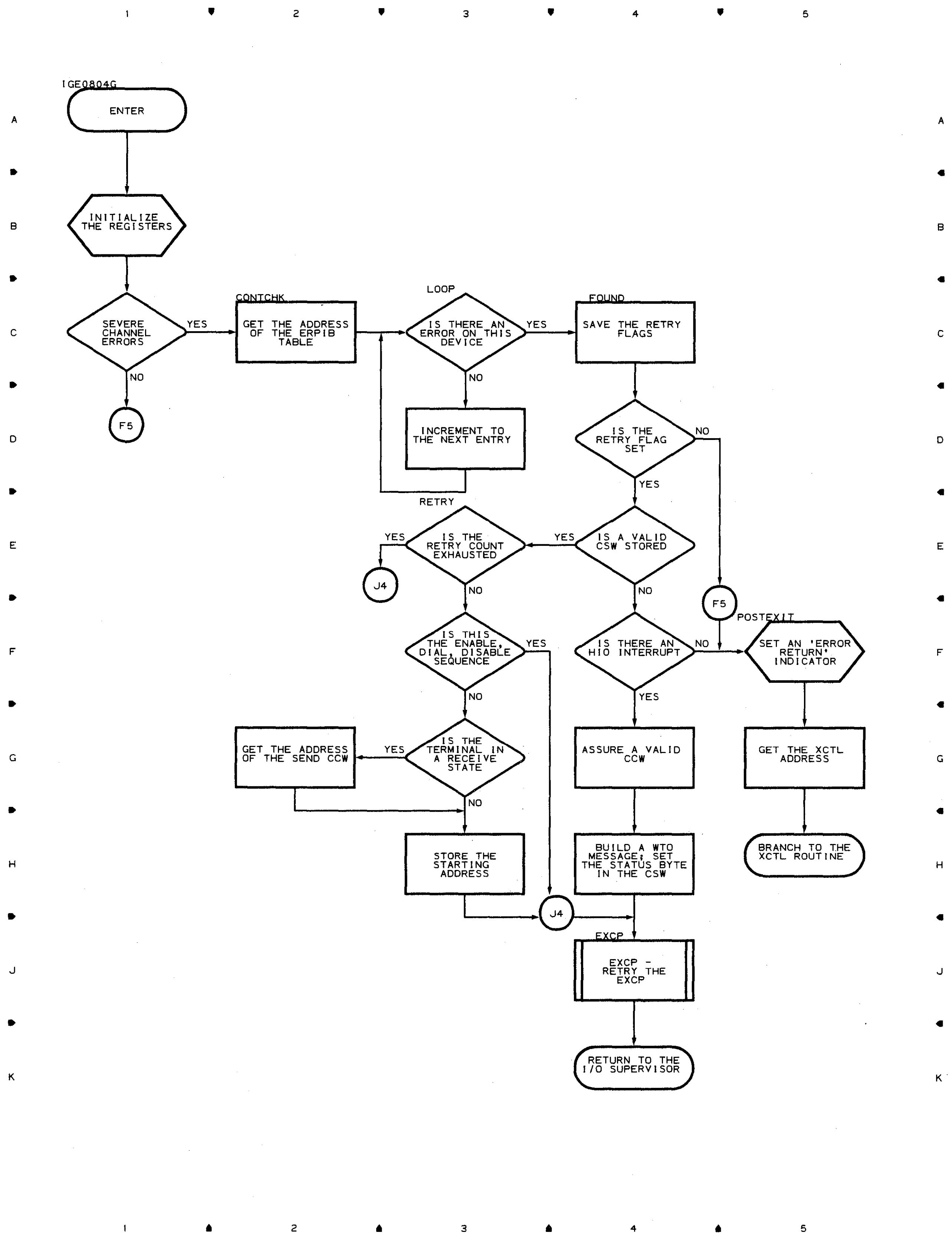

IGE0804G
ENTER
INITIALIZE THE REGISTERS
SEVERE CHANNEL ERRORS
YES
NO
F5
CONTCHK
GET THE ADDRESS OF THE ERPIB TABLE
LOOP
IS THERE AN ERROR ON THIS DEVICE
YES
NO
INCREMENT TO THE NEXT ENTRY
FOUND
SAVE THE RETRY FLAGS
IS THE RETRY FLAG SET
NO
YES
RETRY
IS THE RETRY COUNT EXHAUSTED
YES
J4
NO
IS A VALID CSW STORED
YES
NO
F5
POSTEXIT
IS THIS THE ENABLE, DIAL, DISABLE SEQUENCE
YES
NO
IS THERE AN HIO INTERRUPT
NO
YES
SET AN 'ERROR RETURN' INDICATOR
GET THE ADDRESS OF THE SEND CCW
YES
IS THE TERMINAL IN A RECEIVE STATE
NO
ASSURE A VALID CCW
GET THE XCTL ADDRESS
STORE THE STARTING ADDRESS
BUILD A WTO MESSAGE; SET THE STATUS BYTE IN THE CSW
BRANCH TO THE XCTL ROUTINE
J4
EXCP
EXCP - RETRY THE EXCP
RETURN TO THE I/O SUPERVISOR
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart JK CLOSEDOWN TERMINAL STATISTICS RECORDING MODULE

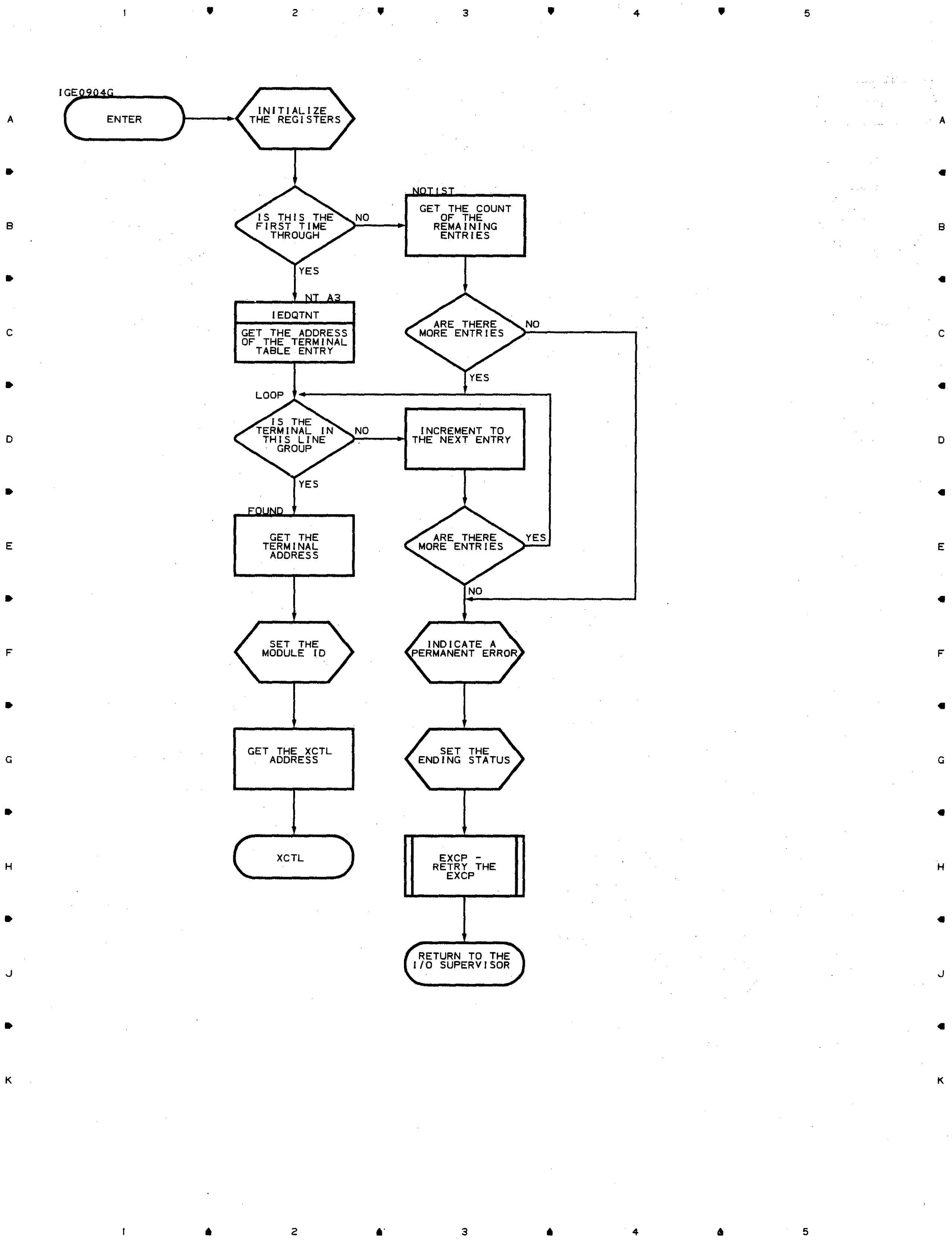
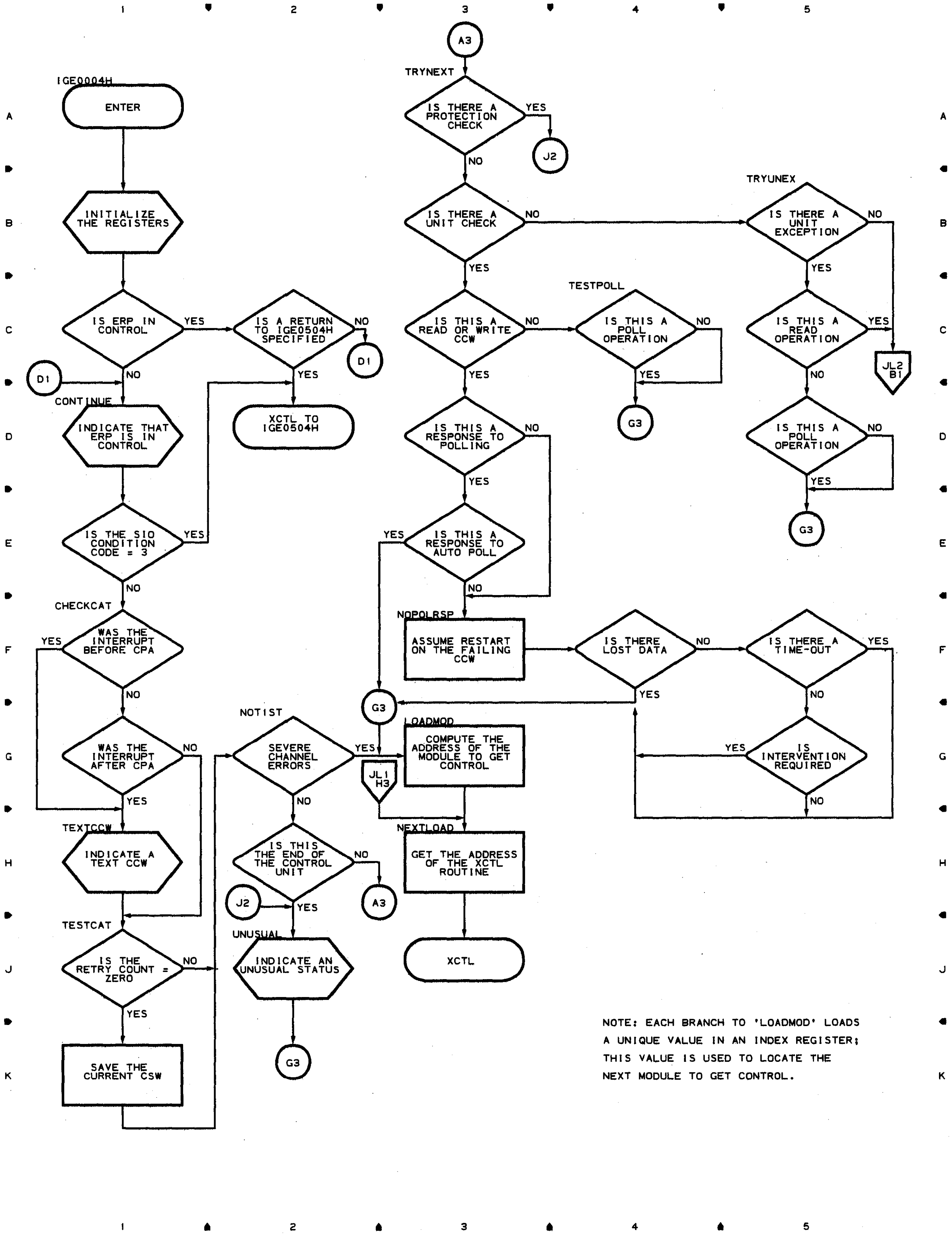


Chart JL-1 BSC ERP CONTROL MODULE



NOTE: EACH BRANCH TO 'LOADMOD' LOADS A UNIQUE VALUE IN AN INDEX REGISTER; THIS VALUE IS USED TO LOCATE THE NEXT MODULE TO GET CONTROL.

Chart JL-2 BSC ERP CONTROL MODULE

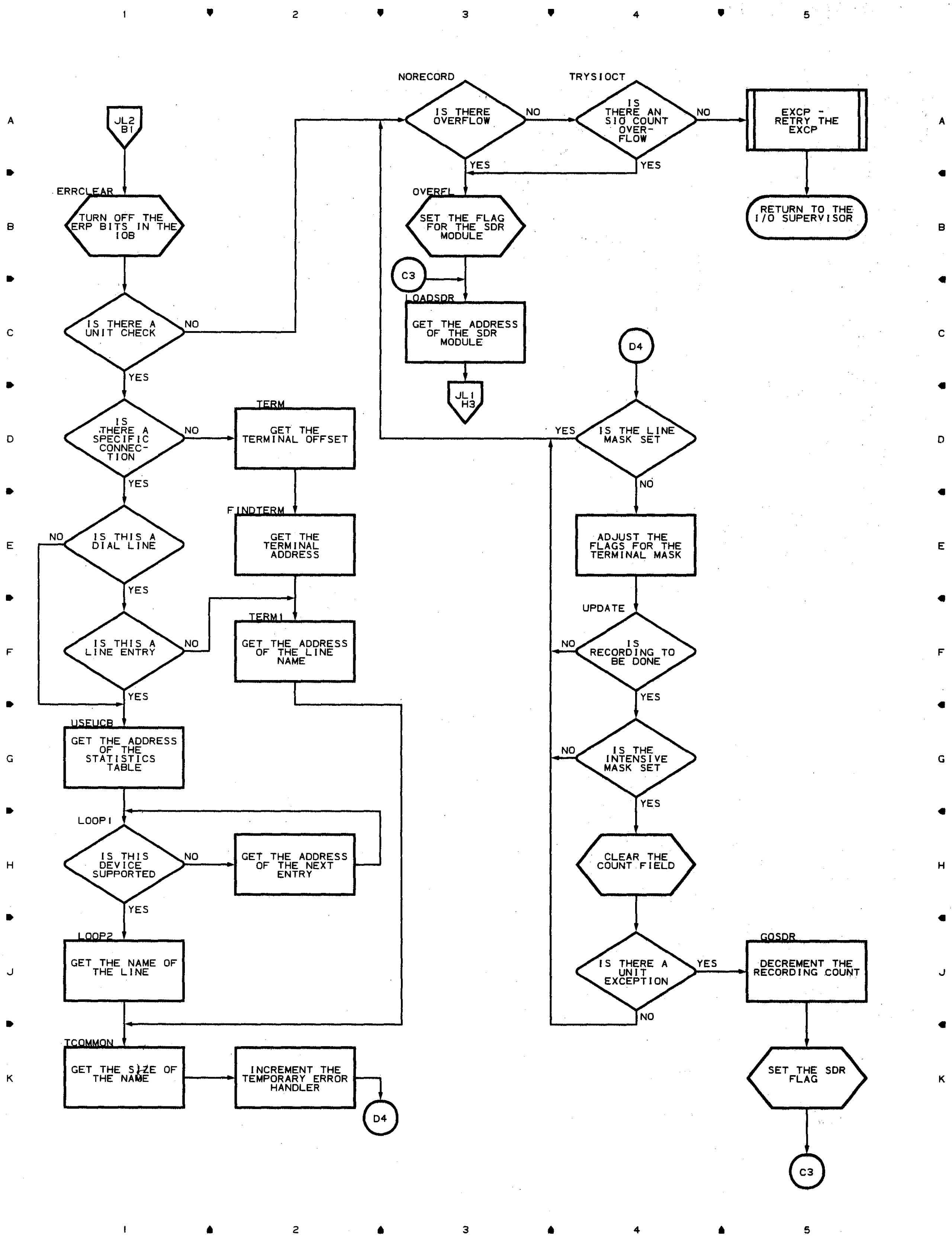


Chart JM-1 BSC READ/WRITE EQUIPMENT CHECK, LOST DATA, INTERVENTION REQUIRED, AND UNIT EXCEPTION ERP MODULE

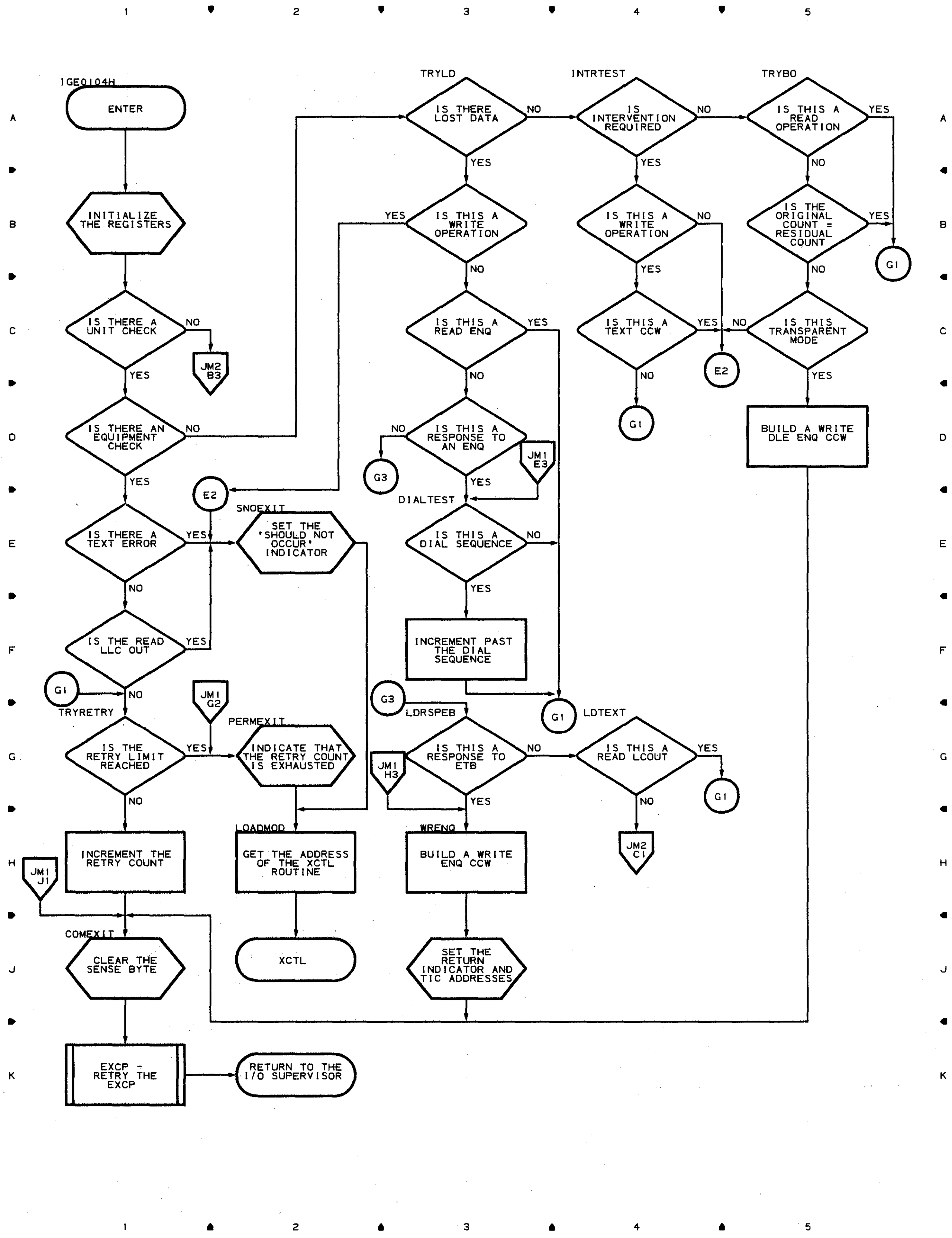


Chart JM-2 BSC READ/WRITE EQUIPMENT CHECK, LOST DATA, INTERVENTION REQUIRED, AND UNIT EXCEPTION ERP MODULE

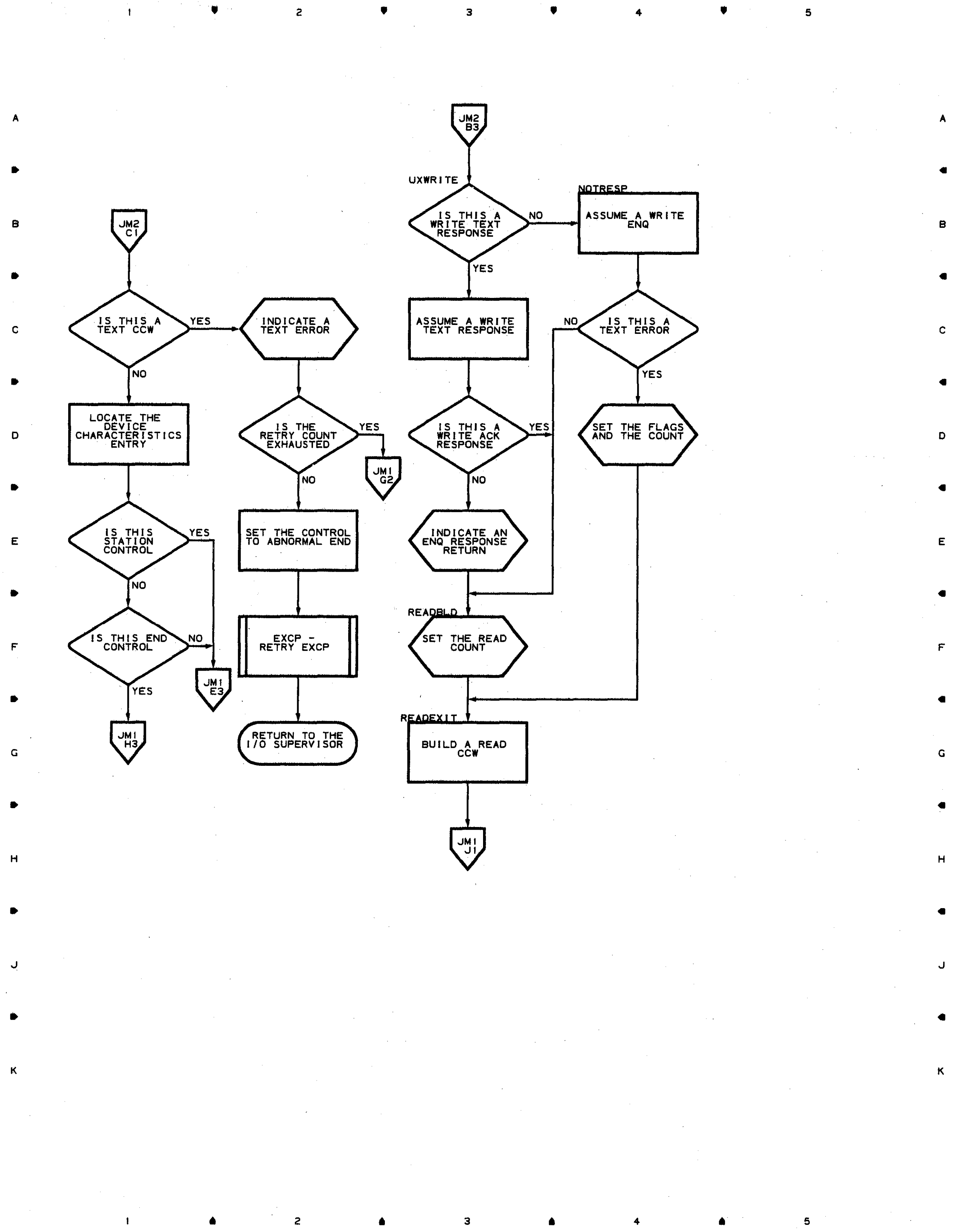
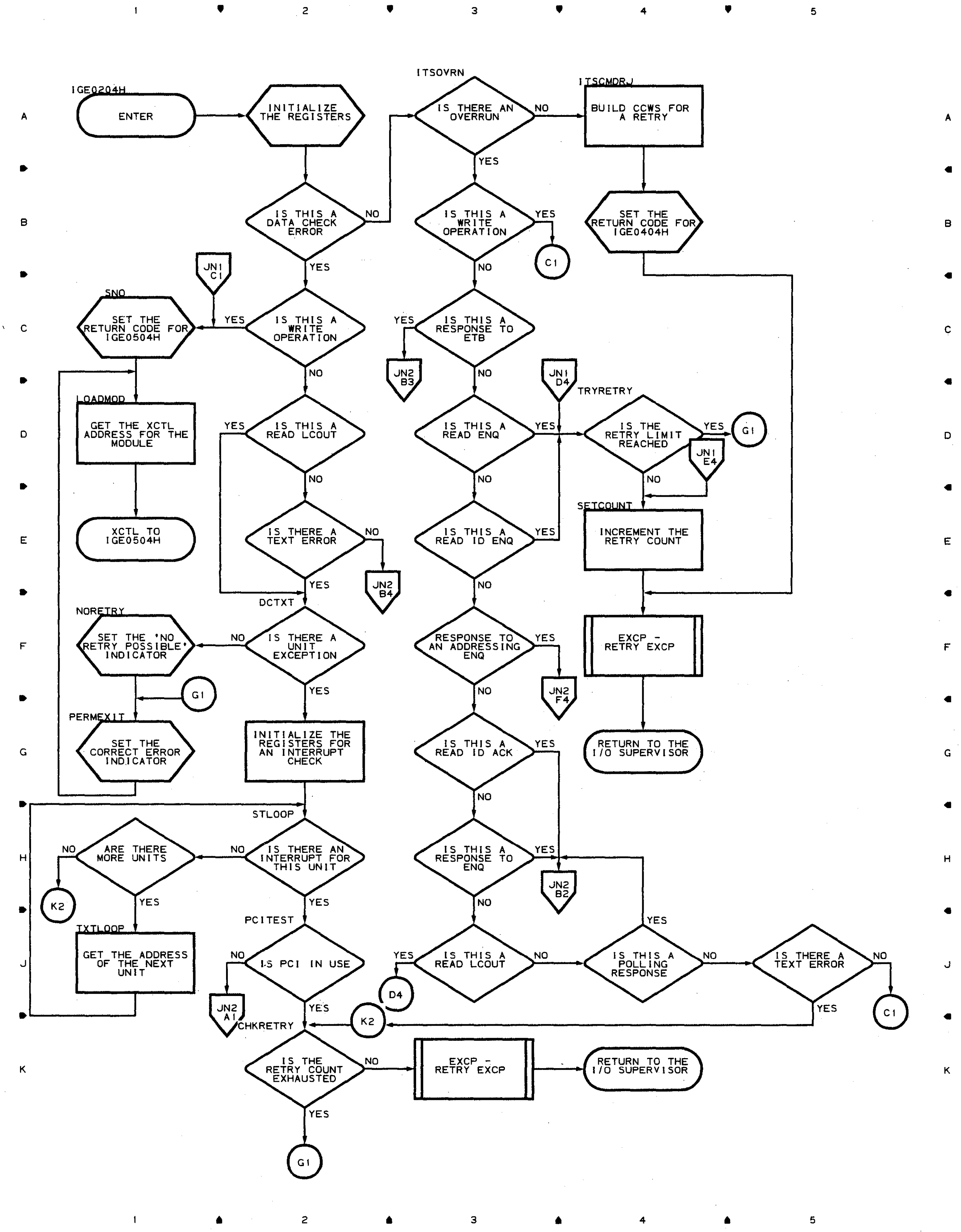


Chart JN-1 BSC READ/WRITE DATA CHECK, OVERRUN, AND COMMAND REJECT ERP MODULE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGE0204H
ENTER
INITIALIZE THE REGISTERS
ITSOVRN
IS THERE AN OVERRUN
NO
YES
ITSCMDRJ
BUILD CCWS FOR A RETRY
IS THIS A DATA CHECK ERROR
NO
YES
IS THIS A WRITE OPERATION
YES
NO
C1
SET THE RETURN CODE FOR IGE0404H
JN1 C1
SNO
SET THE RETURN CODE FOR IGE0504H
YES
IS THIS A WRITE OPERATION
NO
YES
IS THIS A RESPONSE TO ETB
NO
JN2 B3
JN1 D4
TRYRETRY
LOADMOD
GET THE XCTL ADDRESS FOR THE MODULE
YES
IS THIS A READ LCOUT
NO
IS THIS A READ ENQ
YES
NO
IS THE RETRY LIMIT REACHED
YES
G1
JN1 E4
NO
SETCOUNT
XCTL TO IGE0504H
IS THERE A TEXT ERROR
NO
YES
JN2 B4
IS THIS A READ ID ENQ
YES
NO
INCREMENT THE RETRY COUNT
DCTXT
NORETRY
SET THE 'NO RETRY POSSIBLE' INDICATOR
NO
IS THERE A UNIT EXCEPTION
YES
RESPONSE TO AN ADDRESSING ENQ
YES
NO
JN2 F4
EXCP - RETRY EXCP
G1
PERMEXIT
SET THE CORRECT ERROR INDICATOR
INITIALIZE THE REGISTERS FOR AN INTERRUPT CHECK
IS THIS A READ ID ACK
YES
NO
RETURN TO THE I/O SUPERVISOR
STLOOP
NO
ARE THERE MORE UNITS
YES
NO
IS THERE AN INTERRUPT FOR THIS UNIT
YES
IS THIS A RESPONSE TO ENQ
YES
NO
JN2 B2
K2
TXTLOOP
GET THE ADDRESS OF THE NEXT UNIT
PCITEST
NO
IS PCI IN USE
YES
JN2 A1
YES
IS THIS A READ LCOUT
NO
D4
IS THIS A POLLING RESPONSE
YES
NO
IS THERE A TEXT ERROR
NO
YES
C1
K2
CHKRETRY
IS THE RETRY COUNT EXHAUSTED
NO
YES
EXCP - RETRY EXCP
RETURN TO THE I/O SUPERVISOR
G1

Chart JN-2 BSC READ/WRITE DATA CHECK, OVERRUN, AND COMMAND REJECT ERP MODULE

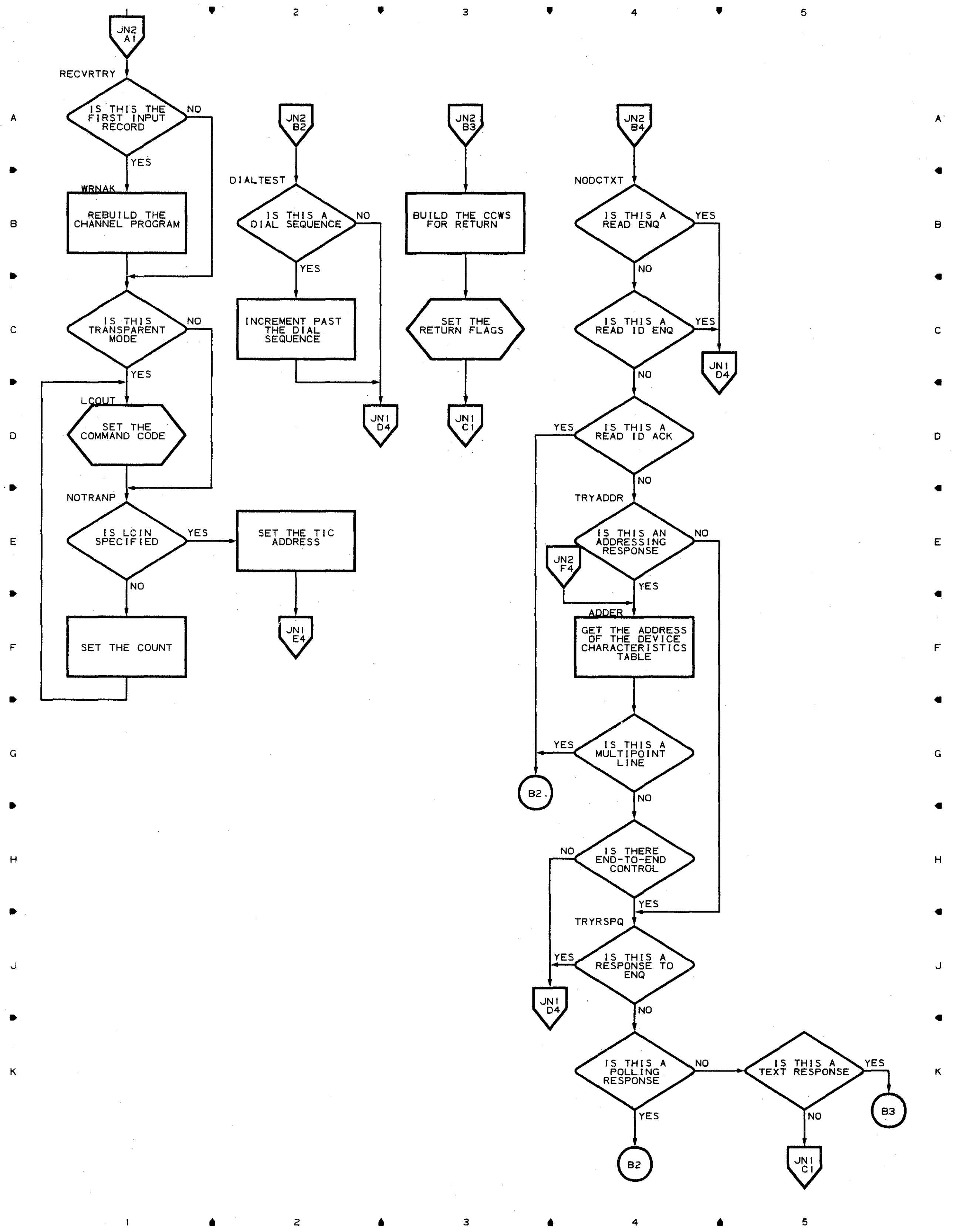


Chart JO BSC SECOND LEVEL CCW RETURN MODULE

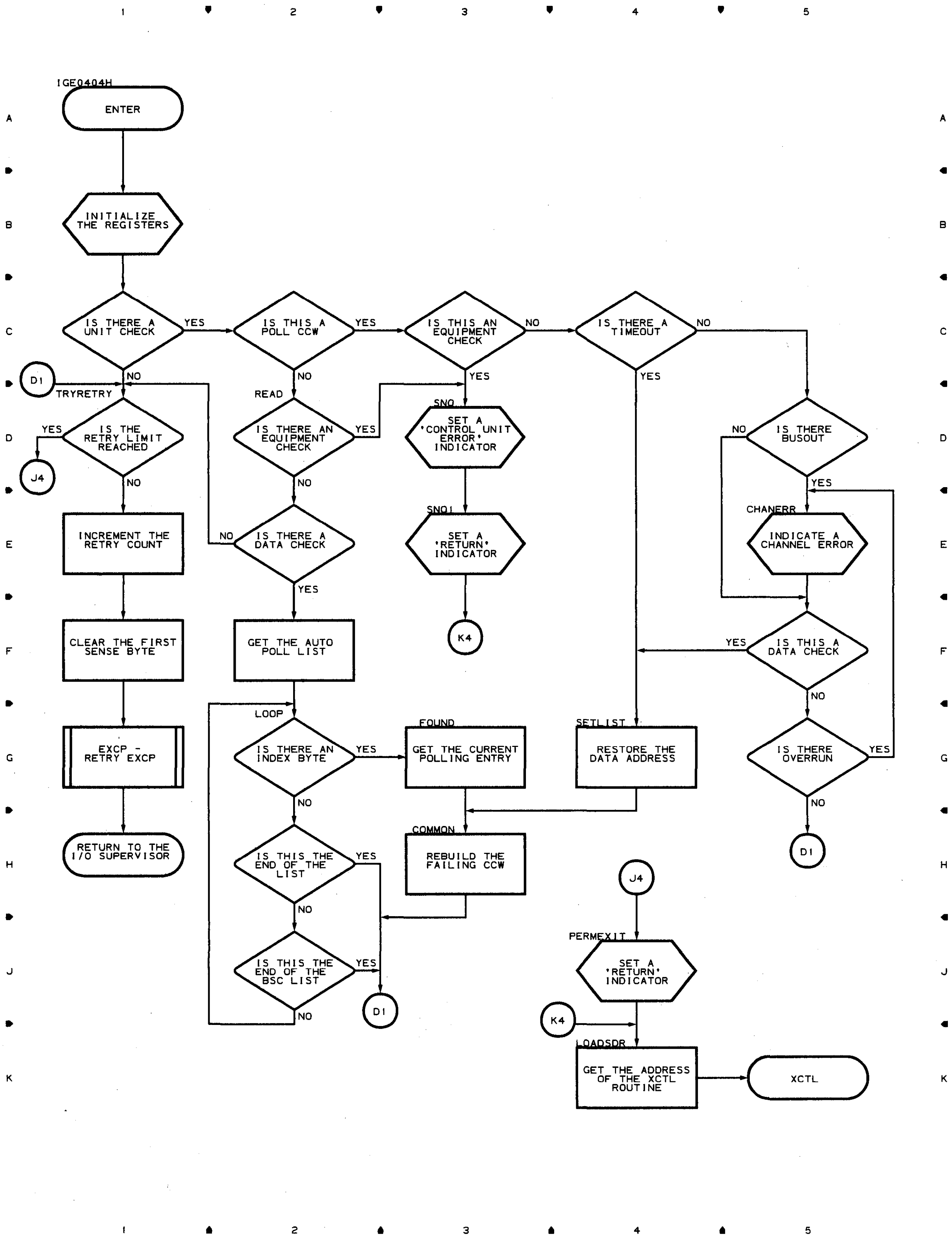
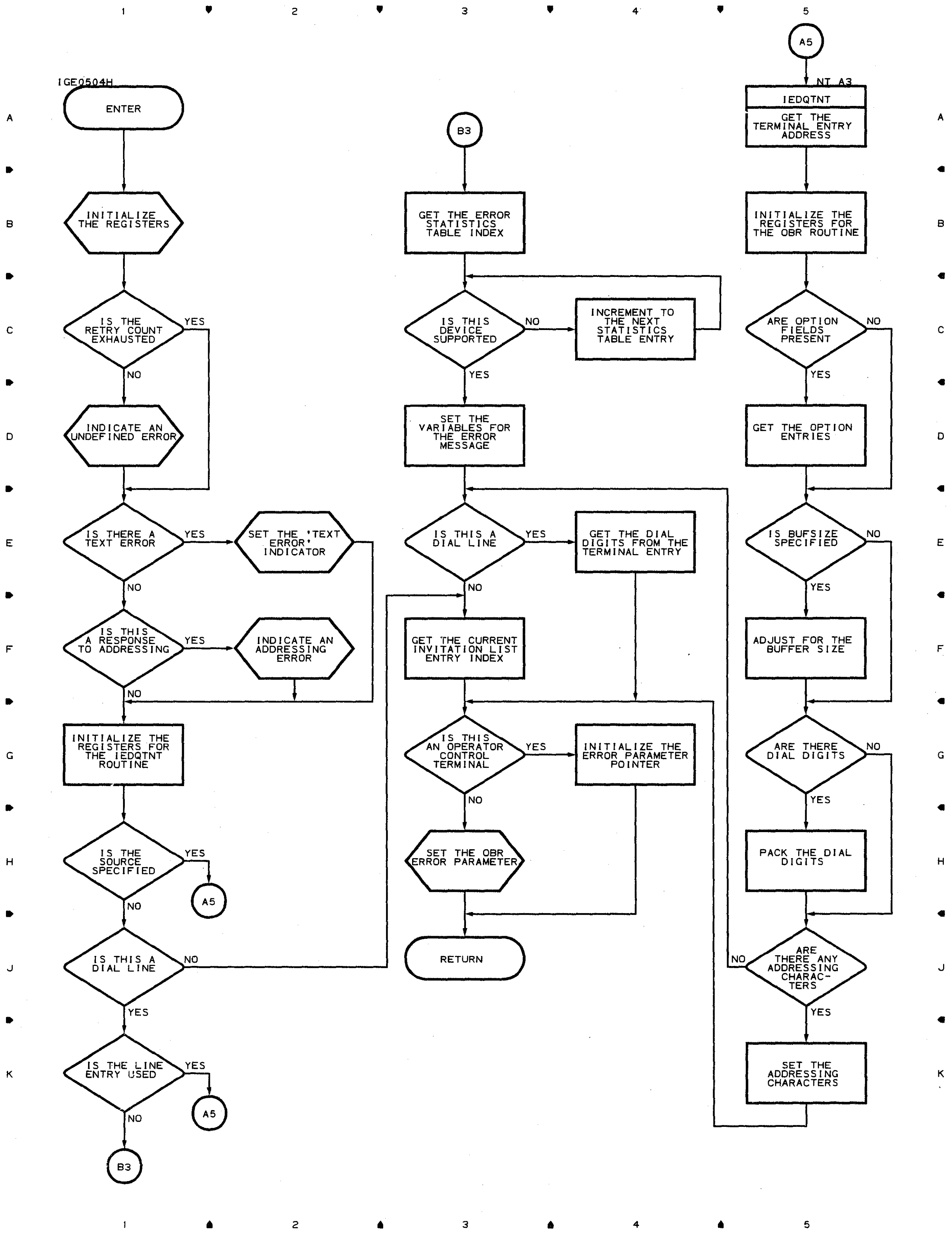


Chart JP BSC ERROR POST MODULE



IGE0504H
ENTER
INITIALIZE THE REGISTERS
IS THE RETRY COUNT EXHAUSTED
YES
NO
INDICATE AN UNDEFINED ERROR
IS THERE A TEXT ERROR
YES
SET THE 'TEXT ERROR' INDICATOR
NO
IS THIS A RESPONSE TO ADDRESSING
YES
INDICATE AN ADDRESSING ERROR
NO
INITIALIZE THE REGISTERS FOR THE IEDQTNT ROUTINE
IS THE SOURCE SPECIFIED
YES
A5
NO
IS THIS A DIAL LINE
NO
YES
IS THE LINE ENTRY USED
YES
A5
NO
B3
B3
GET THE ERROR STATISTICS TABLE INDEX
IS THIS DEVICE SUPPORTED
NO
INCREMENT TO THE NEXT STATISTICS TABLE ENTRY
YES
SET THE VARIABLES FOR THE ERROR MESSAGE
IS THIS A DIAL LINE
YES
GET THE DIAL DIGITS FROM THE TERMINAL ENTRY
NO
GET THE CURRENT INVITATION LIST ENTRY INDEX
IS THIS AN OPERATOR CONTROL TERMINAL
YES
INITIALIZE THE ERROR PARAMETER POINTER
NO
SET THE OBR ERROR PARAMETER
RETURN
A5
NT A3
IEDQTNT
GET THE TERMINAL ENTRY ADDRESS
INITIALIZE THE REGISTERS FOR THE OBR ROUTINE
ARE OPTION FIELDS PRESENT
NO
YES
GET THE OPTION ENTRIES
IS BUFSIZE SPECIFIED
NO
YES
ADJUST FOR THE BUFFER SIZE
ARE THERE DIAL DIGITS
NO
YES
PACK THE DIAL DIGITS
ARE THERE ANY ADDRESSING CHARAC- TERS
NO
YES
SET THE ADDRESSING CHARACTERS
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart JQ BSC CHANNEL CHECK ERP MODULE

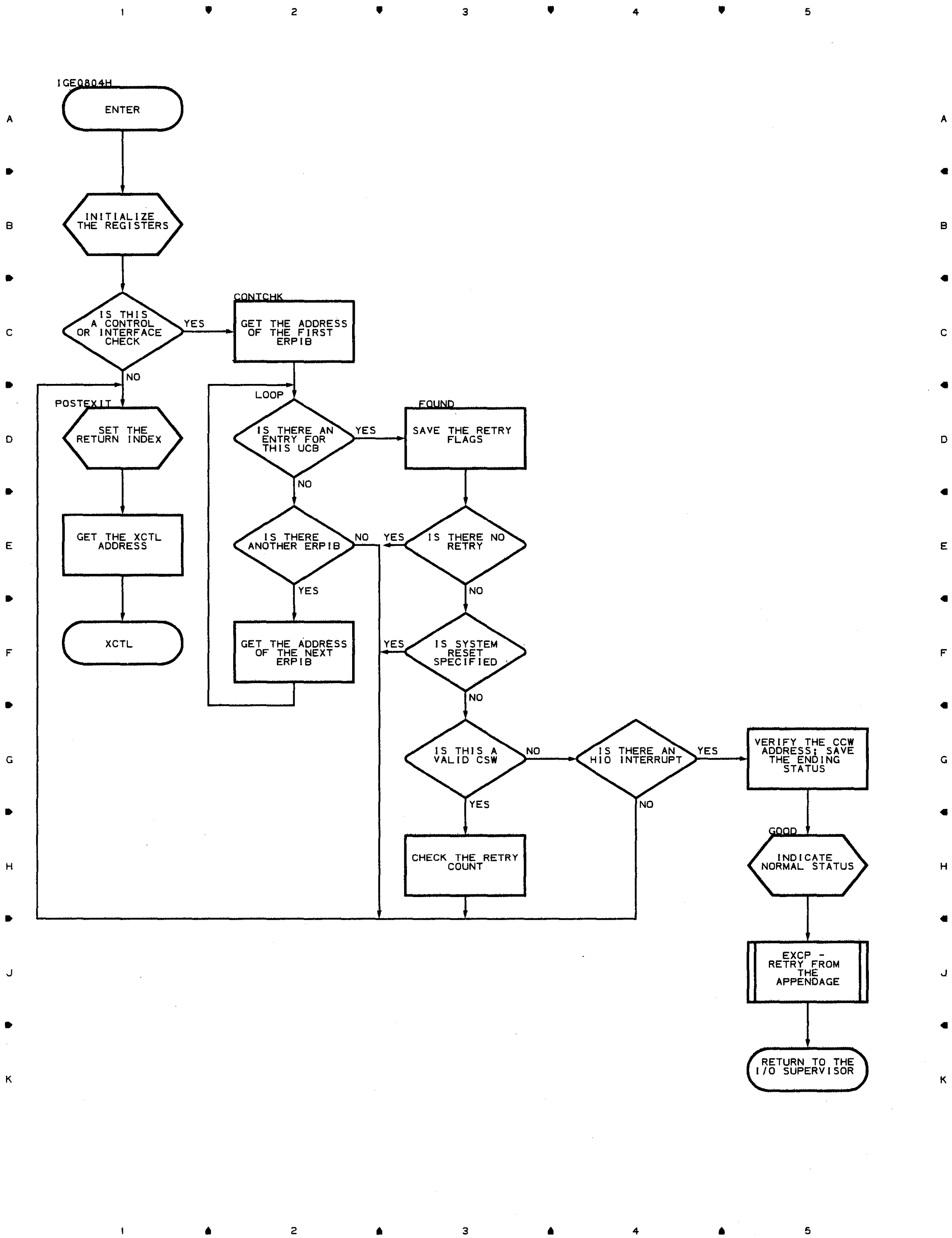
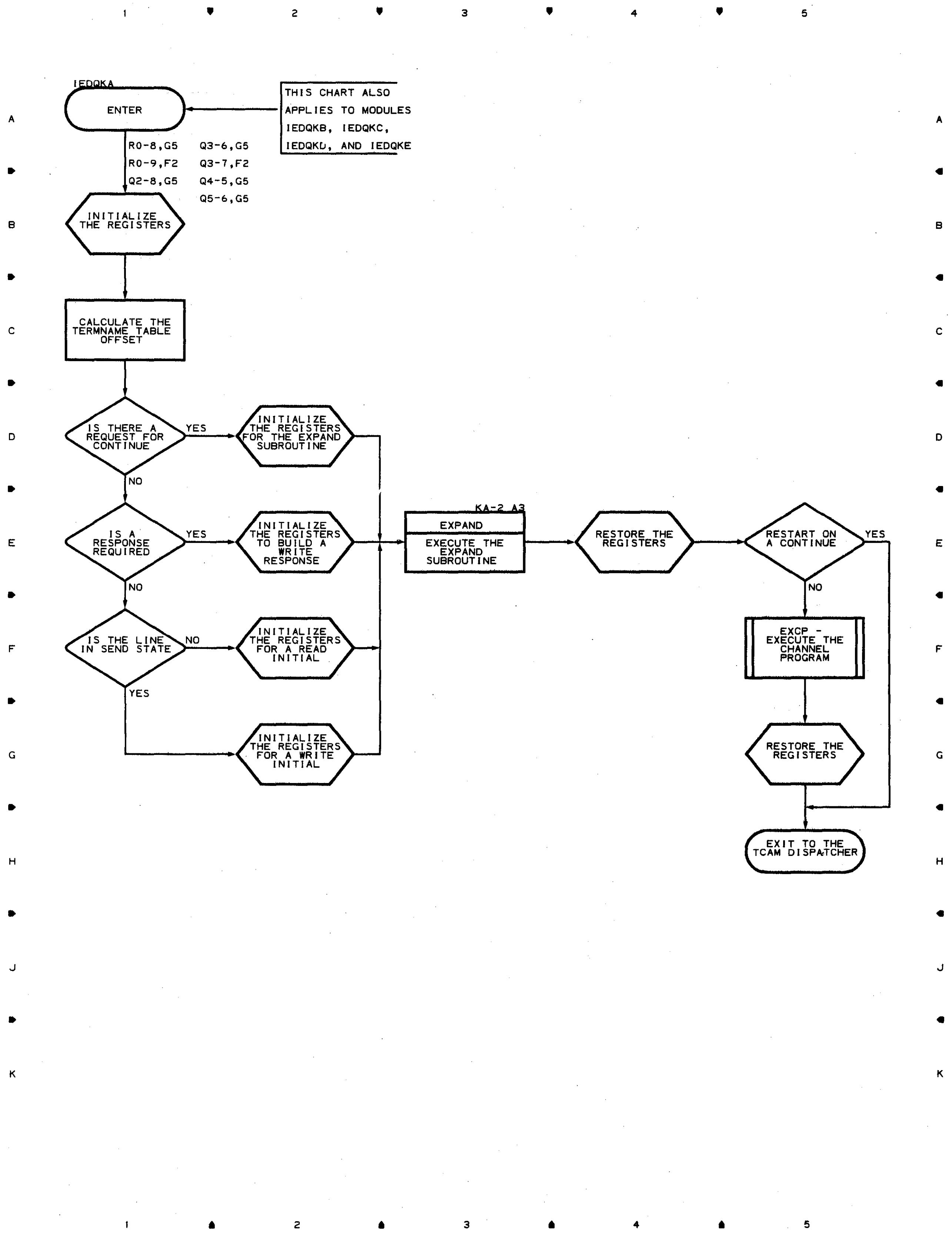

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGE0804H
ENTER
INITIALIZE THE REGISTERS
IS THIS A CONTROL OR INTERFACE CHECK
YES
NO
CONTCHK
GET THE ADDRESS OF THE FIRST ERPIB
LOOP
IS THERE AN ENTRY FOR THIS UCB
YES
NO
FOUND
SAVE THE RETRY FLAGS
POSTEXIT
SET THE RETURN INDEX
GET THE XCTL ADDRESS
XCTL
IS THERE ANOTHER ERPIB
NO
YES
YES
IS THERE NO RETRY
NO
GET THE ADDRESS OF THE NEXT ERPIB
YES
IS SYSTEM RESET SPECIFIED
NO
IS THIS A VALID CSW
NO
YES
IS THERE AN HIO INTERRUPT
YES
NO
VERIFY THE CCW ADDRESS; SAVE THE ENDING STATUS
CHECK THE RETRY COUNT
GOOD
INDICATE NORMAL STATUS
EXCP - RETRY FROM THE APPENDAGE
RETURN TO THE I/O SUPERVISOR

Chart KA-1 ACTIVATE-I/O GENERATOR SUBTASK



IEDQKA
ENTER
THIS CHART ALSO APPLIES TO MODULES IEDQKB, IEDQKC, IEDQKD, AND IEDQKE
R0-8,G5
R0-9,F2
Q2-8,G5
Q3-6,G5
Q3-7,F2
Q4-5,G5
Q5-6,G5
INITIALIZE THE REGISTERS
CALCULATE THE TERMNAME TABLE OFFSET
IS THERE A REQUEST FOR CONTINUE
YES
NO
INITIALIZE THE REGISTERS FOR THE EXPAND SUBROUTINE
IS A RESPONSE REQUIRED
YES
NO
INITIALIZE THE REGISTERS TO BUILD A WRITE RESPONSE
IS THE LINE IN SEND STATE
NO
YES
INITIALIZE THE REGISTERS FOR A READ INITIAL
INITIALIZE THE REGISTERS FOR A WRITE INITIAL
KA-2 A3
EXPAND
EXECUTE THE EXPAND SUBROUTINE
RESTORE THE REGISTERS
RESTART ON A CONTINUE
YES
NO
EXCP - EXECUTE THE CHANNEL PROGRAM
RESTORE THE REGISTERS
EXIT TO THE TCAM DISPATCHER
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart KA-2 ACTIVATE-I/O GENERATOR SUBTASK

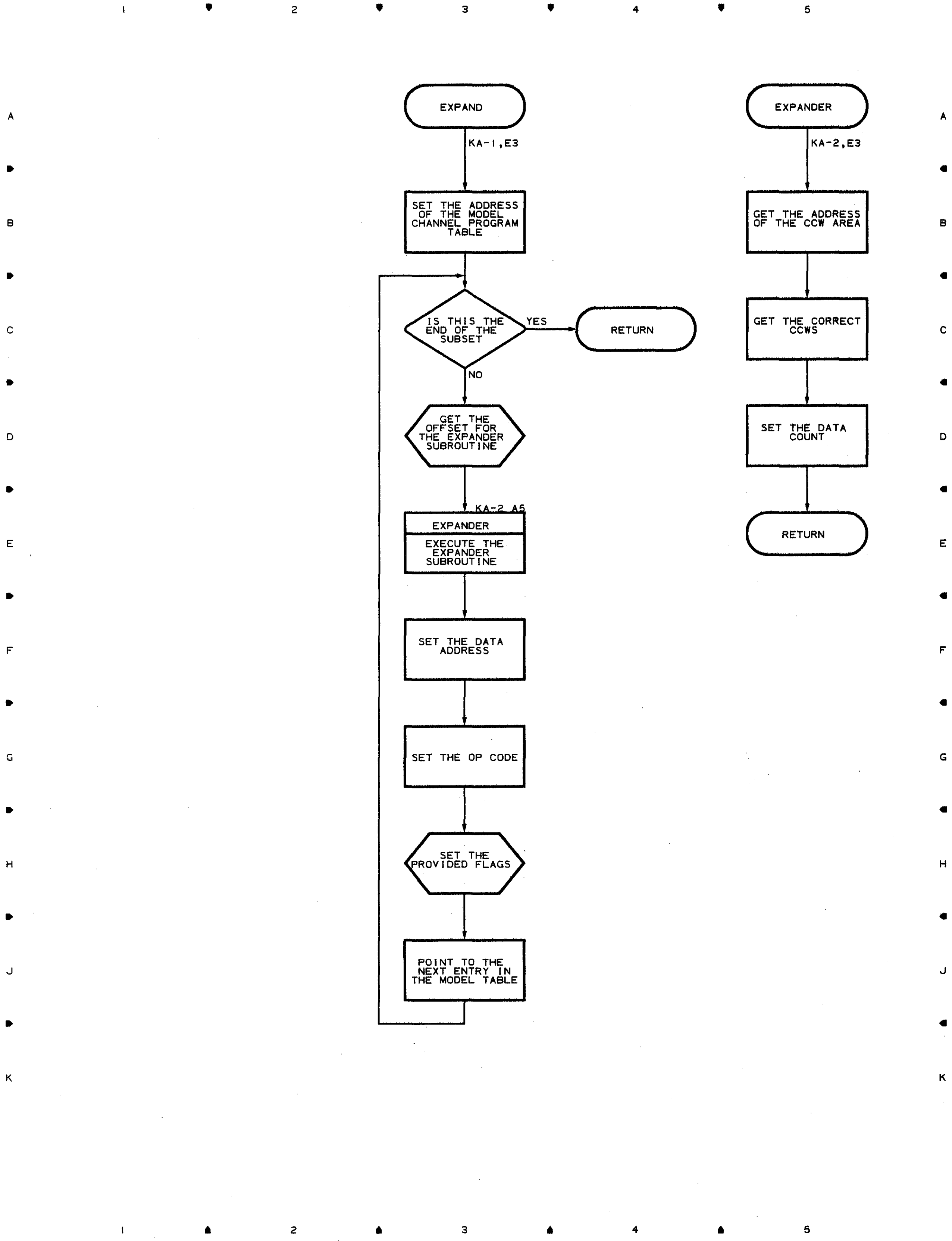

EXPAND
KA-1,E3
SET THE ADDRESS OF THE MODEL CHANNEL PROGRAM TABLE
IS THIS THE END OF THE SUBSET
YES
RETURN
NO
GET THE OFFSET FOR THE EXPANDER SUBROUTINE
KA-2 A5
EXPANDER
EXECUTE THE EXPANDER SUBROUTINE
SET THE DATA ADDRESS
SET THE OP CODE
SET THE PROVIDED FLAGS
POINT TO THE NEXT ENTRY IN THE MODEL TABLE
EXPANDER
KA-2,E3
GET THE ADDRESS OF THE CCW AREA
GET THE CORRECT CCWS
SET THE DATA COUNT
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart LA OPEN ERROR HANDLER

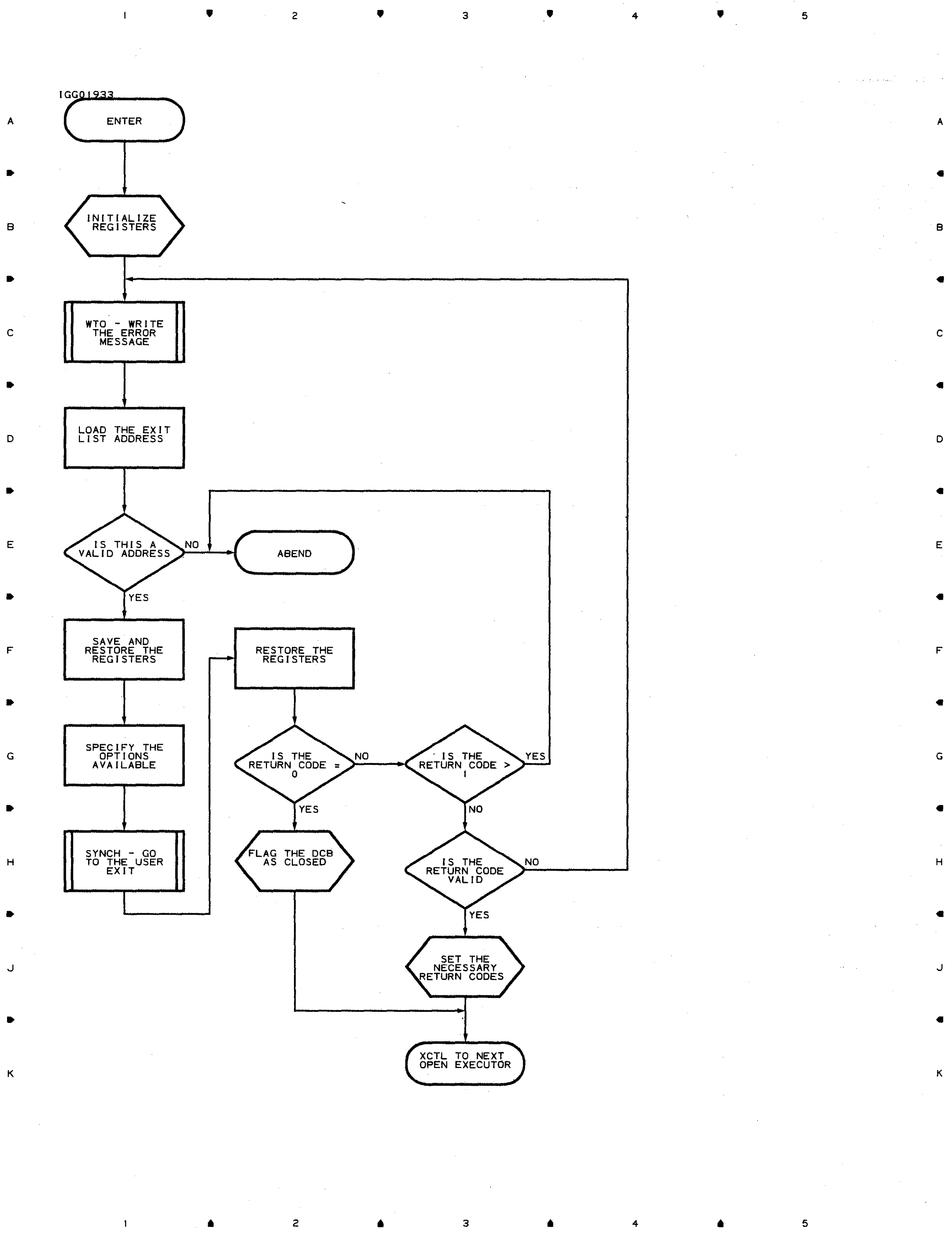


Chart LB DISK MESSAGE QUEUES OPEN ROUTINE - LOAD 1

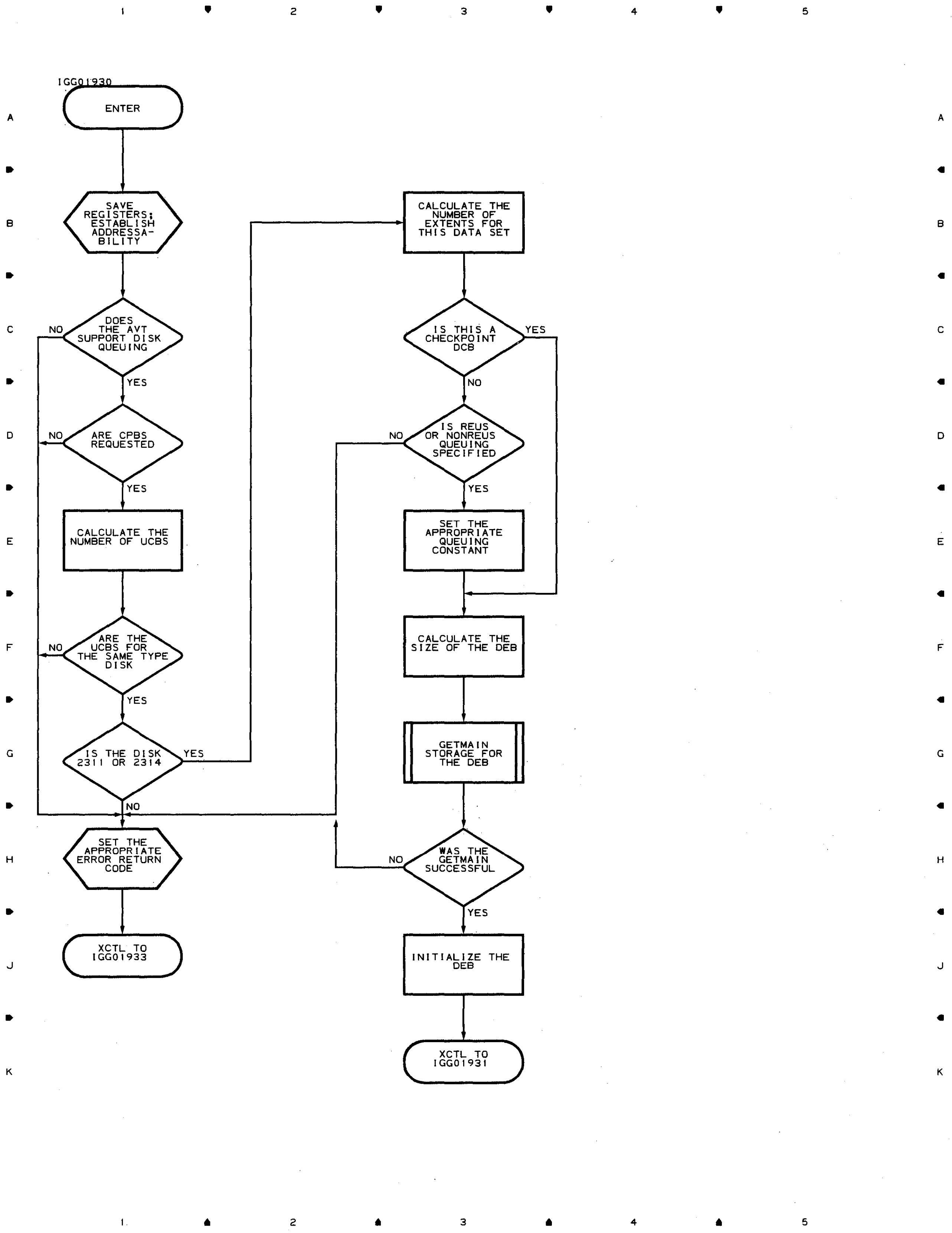

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG01930
ENTER
SAVE REGISTERS; ESTABLISH ADDRESSA-BILITY
DOES THE AVT SUPPORT DISK QUEUING
NO
YES
ARE CPBS REQUESTED
NO
YES
CALCULATE THE NUMBER OF UCBS
ARE THE UCBS FOR THE SAME TYPE DISK
NO
YES
IS THE DISK 2311 OR 2314
YES
NO
SET THE APPROPRIATE ERROR RETURN CODE
XCTL TO IGG01933
CALCULATE THE NUMBER OF EXTENTS FOR THIS DATA SET
IS THIS A CHECKPOINT DCB
YES
NO
IS REUS OR NONREUS QUEUING SPECIFIED
NO
YES
SET THE APPROPRIATE QUEUING CONSTANT
CALCULATE THE SIZE OF THE DEB
GETMAIN STORAGE FOR THE DEB
WAS THE GETMAIN SUCCESSFUL
NO
YES
INITIALIZE THE DEB
XCTL TO IGG01931

Chart LC DISK MESSAGE QUEUES OPEN ROUTINE - LOAD 2

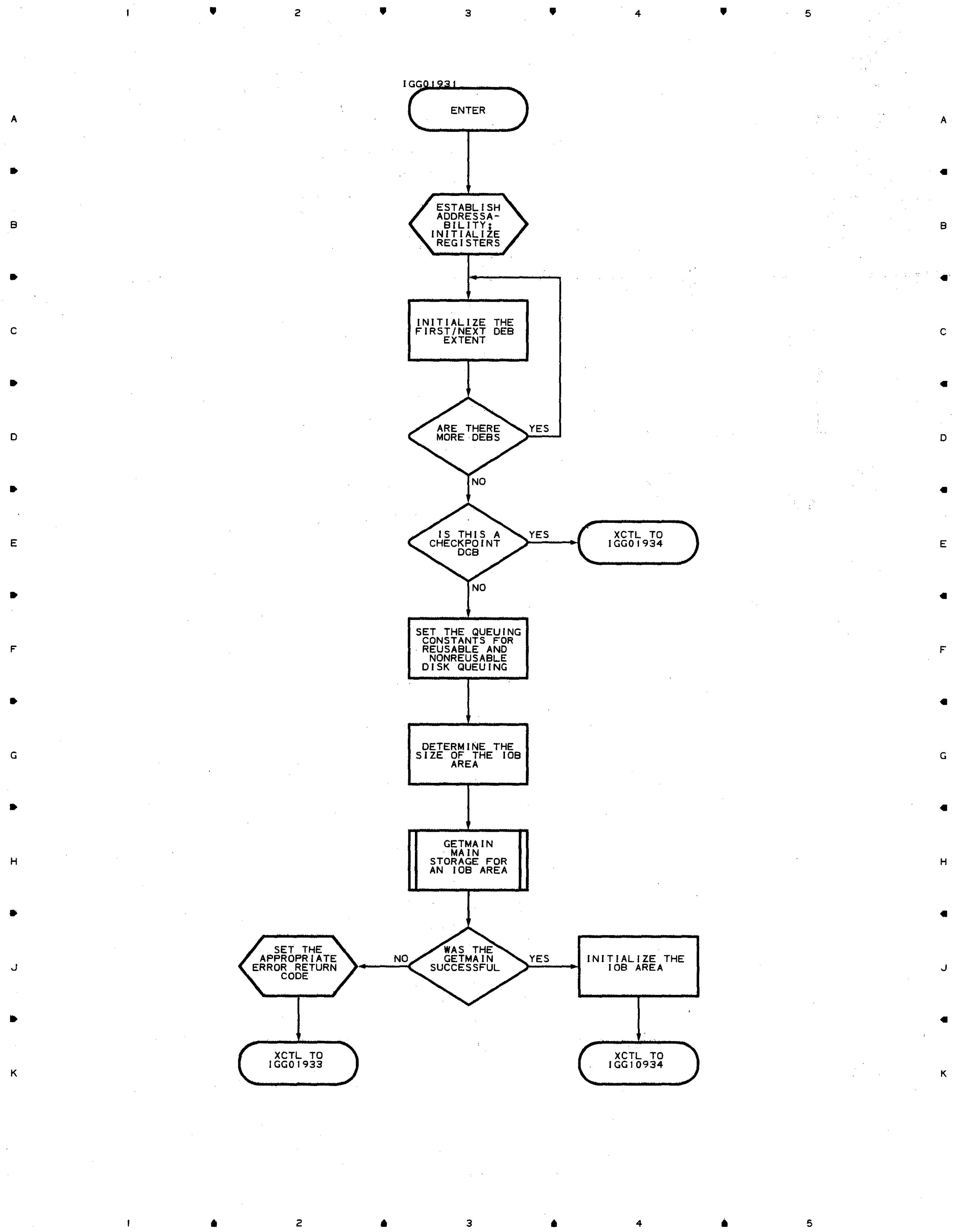

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG01931
ENTER
ESTABLISH ADDRESSABILITY; INITIALIZE REGISTERS
INITIALIZE THE FIRST/NEXT DEB EXTENT
ARE THERE MORE DEBS
YES
NO
IS THIS A CHECKPOINT DCB
YES
XCTL TO IGG01934
NO
SET THE QUEUING CONSTANTS FOR REUSABLE AND NONREUSABLE DISK QUEUING
DETERMINE THE SIZE OF THE IOB AREA
GETMAIN MAIN STORAGE FOR AN IOB AREA
WAS THE GETMAIN SUCCESSFUL
NO
SET THE APPROPRIATE ERROR RETURN CODE
YES
INITIALIZE THE IOB AREA
XCTL TO IGG01933
XCTL TO IGG10934

Chart LD DISK MESSAGE QUEUES OPEN ROUTINE - LOAD 3

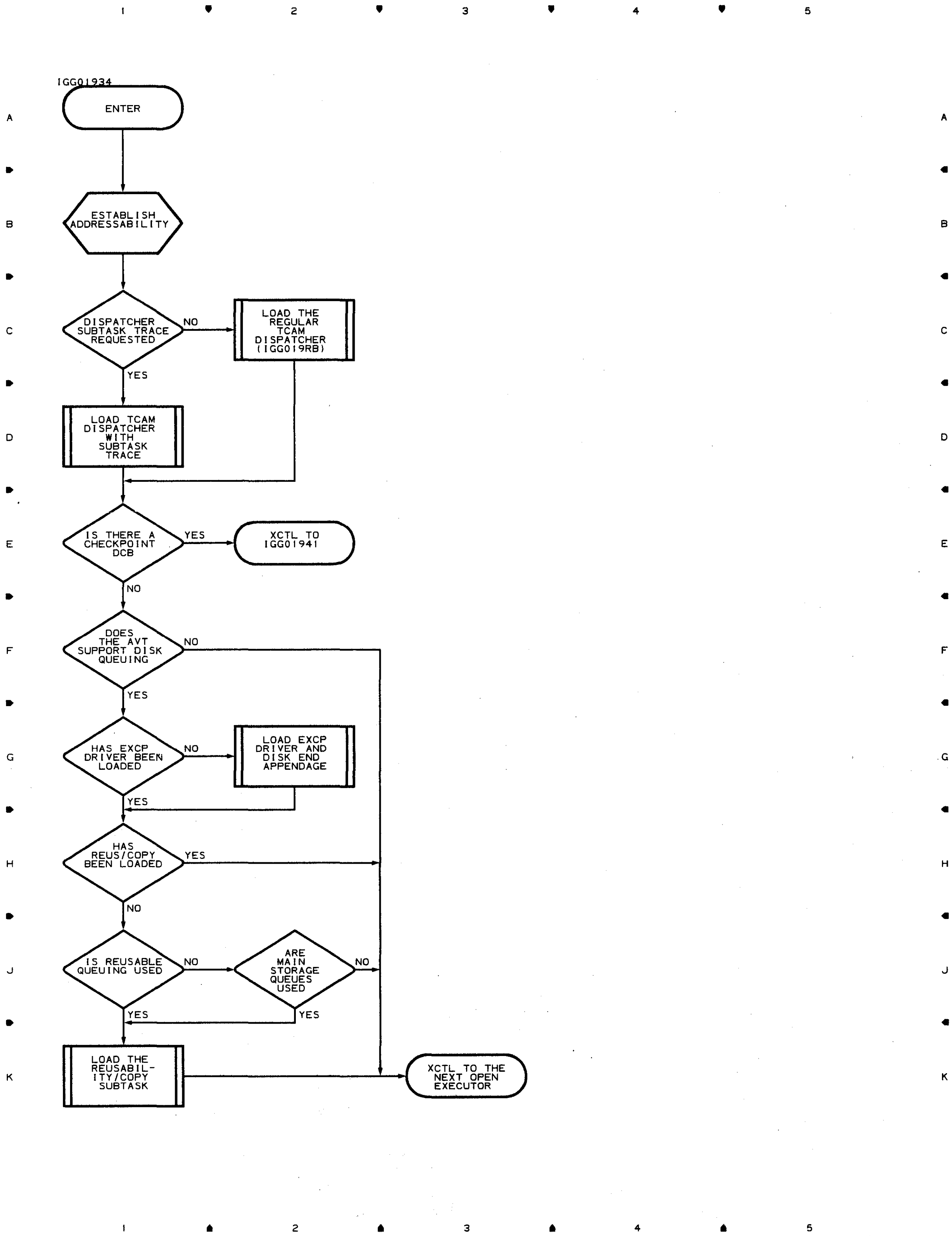

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG01934
ENTER
ESTABLISH ADDRESSABILITY
DISPATCHER SUBTASK TRACE REQUESTED
NO
LOAD THE REGULAR TCAM DISPATCHER (IGG019RB)
YES
LOAD TCAM DISPATCHER WITH SUBTASK TRACE
IS THERE A CHECKPOINT DCB
YES
XCTL TO IGG01941
NO
DOES THE AVT SUPPORT DISK QUEUING
NO
YES
HAS EXCP DRIVER BEEN LOADED
NO
LOAD EXCP DRIVER AND DISK END APPENDAGE
YES
HAS REUS/COPY BEEN LOADED
YES
NO
IS REUSABLE QUEUING USED
NO
ARE MAIN STORAGE QUEUES USED
NO
YES
YES
LOAD THE REUSABIL-ITY/COPY SUBTASK
XCTL TO THE NEXT OPEN EXECUTOR

Chart LE LINE GROUP OPEN ROUTINE - LOAD 1

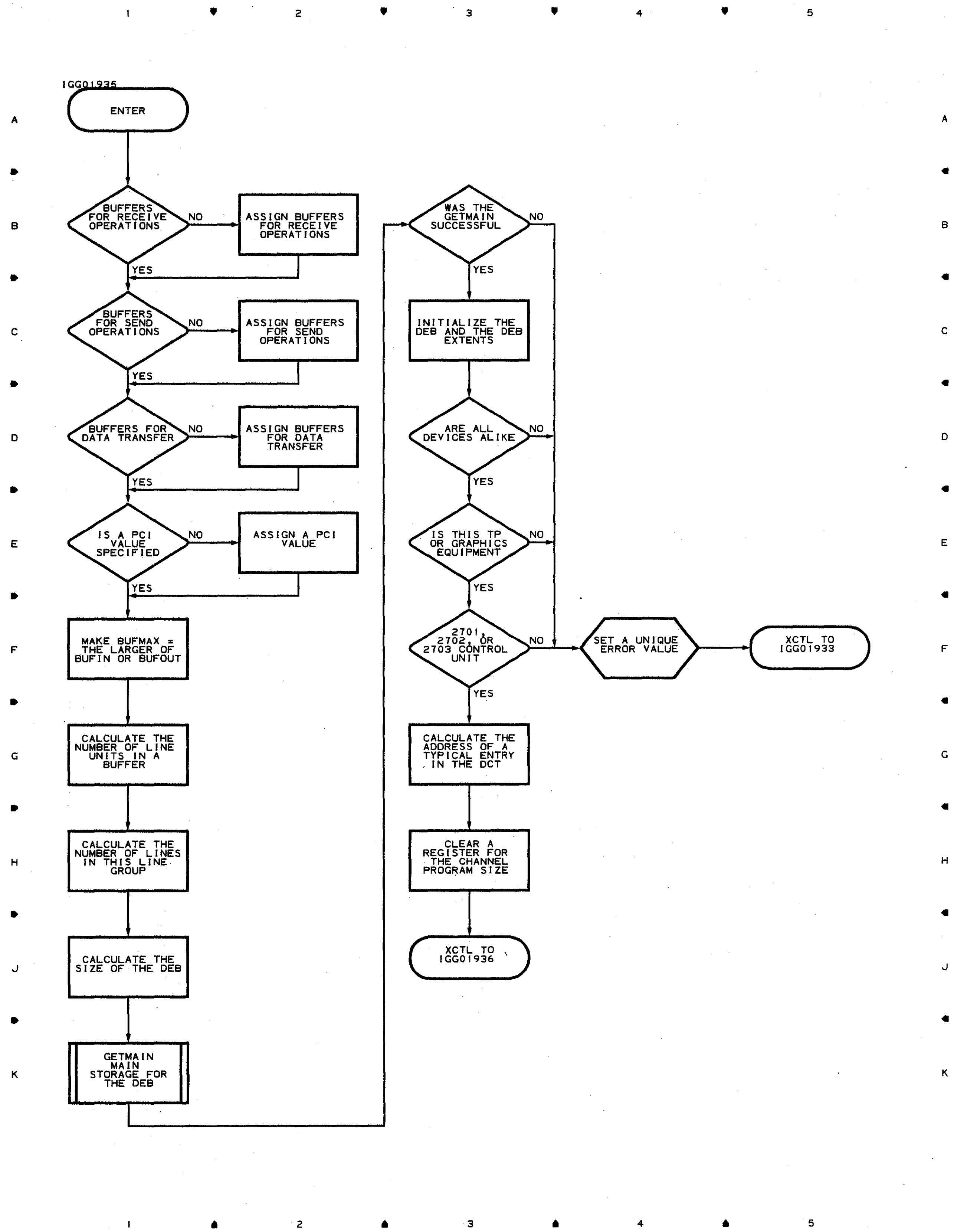
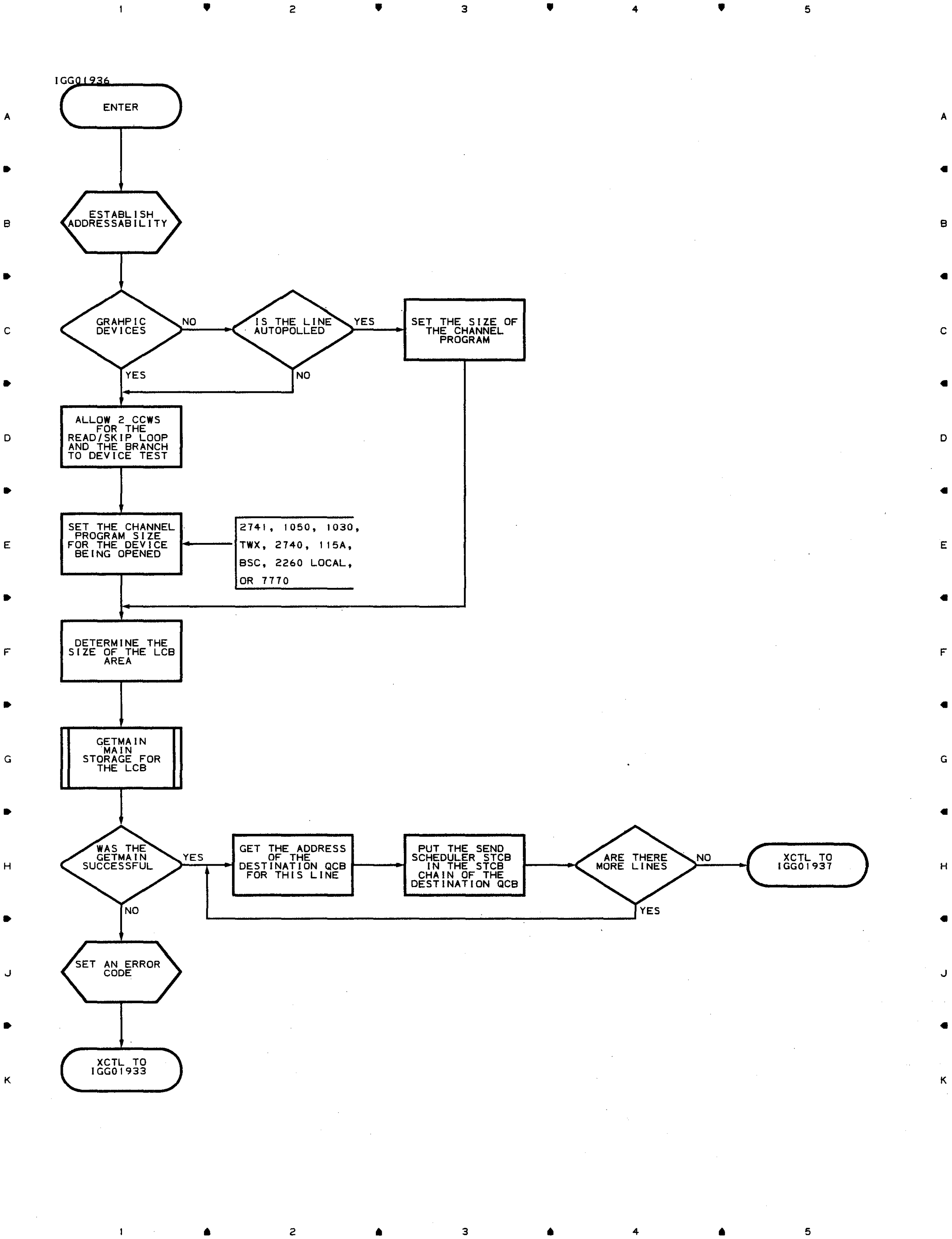

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG01935
ENTER
BUFFERS FOR RECEIVE OPERATIONS
NO
YES
ASSIGN BUFFERS FOR RECEIVE OPERATIONS
BUFFERS FOR SEND OPERATIONS
NO
YES
ASSIGN BUFFERS FOR SEND OPERATIONS
BUFFERS FOR DATA TRANSFER
NO
YES
ASSIGN BUFFERS FOR DATA TRANSFER
IS A PCI VALUE SPECIFIED
NO
YES
ASSIGN A PCI VALUE
MAKE BUFMAX = THE LARGER OF BUFIN OR BUFOUT
CALCULATE THE NUMBER OF LINE UNITS IN A BUFFER
CALCULATE THE NUMBER OF LINES IN THIS LINE GROUP
CALCULATE THE SIZE OF THE DEB
GETMAIN MAIN STORAGE FOR THE DEB
WAS THE GETMAIN SUCCESSFUL
NO
YES
INITIALIZE THE DEB AND THE DEB EXTENTS
ARE ALL DEVICES ALIKE
NO
YES
IS THIS TP OR GRAPHICS EQUIPMENT
NO
YES
2701, 2702, OR 2703 CONTROL UNIT
NO
YES
SET A UNIQUE ERROR VALUE
XCTL TO IGG01933
CALCULATE THE ADDRESS OF A TYPICAL ENTRY IN THE DCT
CLEAR A REGISTER FOR THE CHANNEL PROGRAM SIZE
XCTL TO IGG01936

Chart LF LINE GROUP OPEN ROUTINE - LOAD 2


1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG01936
ENTER
ESTABLISH ADDRESSABILITY
GRAHPIC DEVICES
NO
IS THE LINE AUTOPOLLED
YES
SET THE SIZE OF THE CHANNEL PROGRAM
YES
NO
ALLOW 2 CCWS FOR THE READ/SKIP LOOP AND THE BRANCH TO DEVICE TEST
SET THE CHANNEL PROGRAM SIZE FOR THE DEVICE BEING OPENED
2741, 1050, 1030, TWX, 2740, 115A, BSC, 2260 LOCAL, OR 7770
DETERMINE THE SIZE OF THE LCB AREA
GETMAIN MAIN STORAGE FOR THE LCB
WAS THE GETMAIN SUCCESSFUL
YES
GET THE ADDRESS OF THE DESTINATION QCB FOR THIS LINE
PUT THE SEND SCHEDULER STCB IN THE STCB CHAIN OF THE DESTINATION QCB
ARE THERE MORE LINES
NO
XCTL TO IGG01937
YES
NO
SET AN ERROR CODE
XCTL TO IGG01933

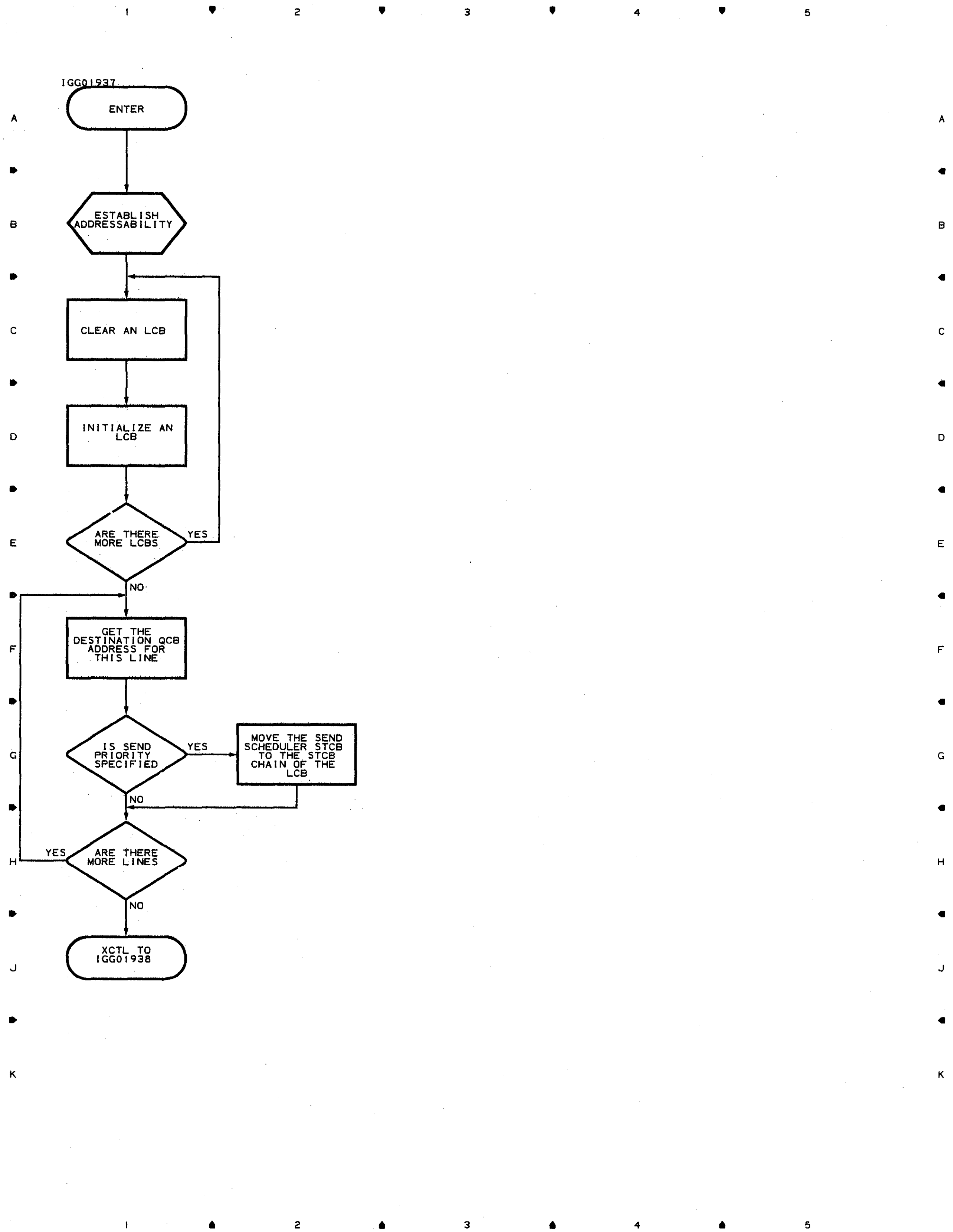

1
2
3
4
5
IGG01937
ENTER
ESTABLISH ADDRESSABILITY
CLEAR AN LCB
INITIALIZE AN LCB
ARE THERE MORE LCBS
YES
NO
GET THE DESTINATION QCB ADDRESS FOR THIS LINE
IS SEND PRIORITY SPECIFIED
YES
MOVE THE SEND SCHEDULER STCB TO THE STCB CHAIN OF THE LCB
NO
ARE THERE MORE LINES
YES
NO
XCTL TO IGG01938
A
B
C
D
E
F
G
H
J
K

Chart LH LINE GROUP OPEN ROUTINE - LOAD 4

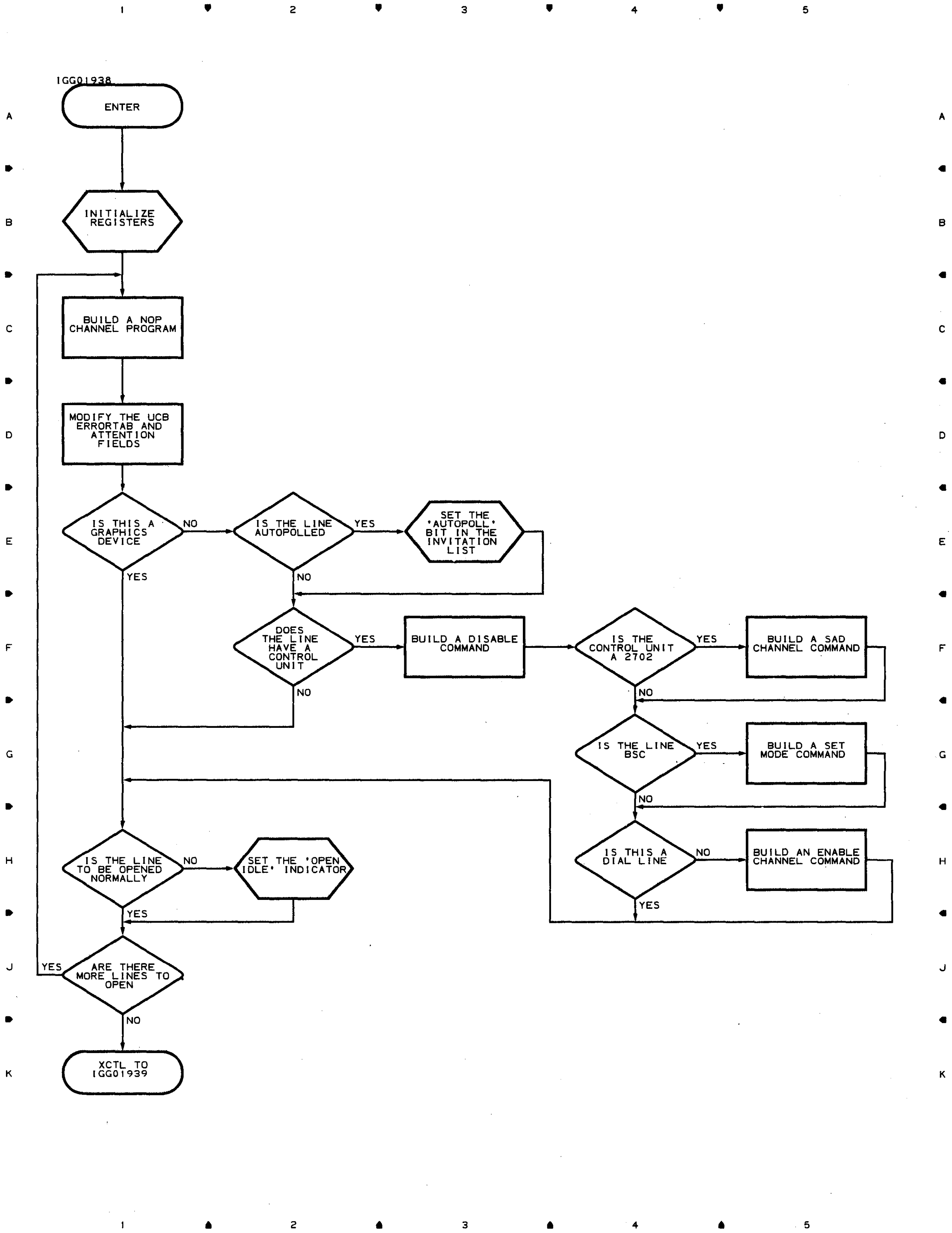

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG01938
ENTER
INITIALIZE REGISTERS
BUILD A NOP CHANNEL PROGRAM
MODIFY THE UCB ERRORTAB AND ATTENTION FIELDS
IS THIS A GRAPHICS DEVICE
NO
YES
IS THE LINE AUTOPOLLED
YES
NO
SET THE 'AUTOPOLL' BIT IN THE INVITATION LIST
DOES THE LINE HAVE A CONTROL UNIT
YES
NO
BUILD A DISABLE COMMAND
IS THE CONTROL UNIT A 2702
YES
NO
BUILD A SAD CHANNEL COMMAND
IS THE LINE BSC
YES
NO
BUILD A SET MODE COMMAND
IS THIS A DIAL LINE
NO
YES
BUILD AN ENABLE CHANNEL COMMAND
IS THE LINE TO BE OPENED NORMALLY
NO
YES
SET THE 'OPEN IDLE' INDICATOR
ARE THERE MORE LINES TO OPEN
YES
NO
XCTL TO IGG01939

Chart LI LINE GROUP OPEN ROUTINE - LOAD 5

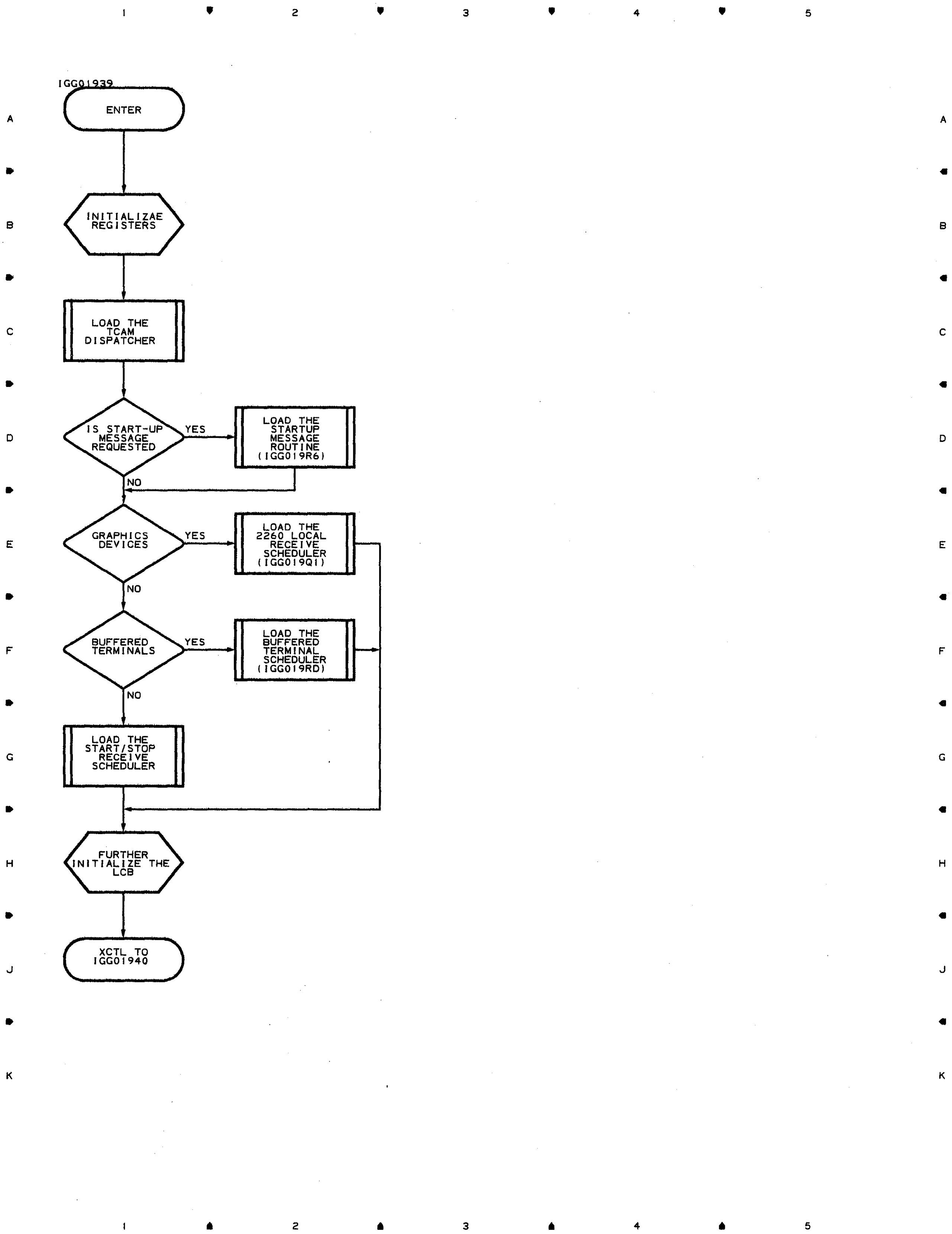

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG01939
ENTER
INITIALIZAE REGISTERS
LOAD THE TCAM DISPATCHER
IS START-UP MESSAGE REQUESTED
YES
LOAD THE STARTUP MESSAGE ROUTINE (IGG019R6)
NO
GRAPHICS DEVICES
YES
LOAD THE 2260 LOCAL RECEIVE SCHEDULER (IGG019Q1)
NO
BUFFERED TERMINALS
YES
LOAD THE BUFFERED TERMINAL SCHEDULER (IGG019RD)
NO
LOAD THE START/STOP RECEIVE SCHEDULER
FURTHER INITIALIZE THE LCB
XCTL TO IGG01940

Chart LJ LINE GROUP OPEN ROUTINE - LOAD 6

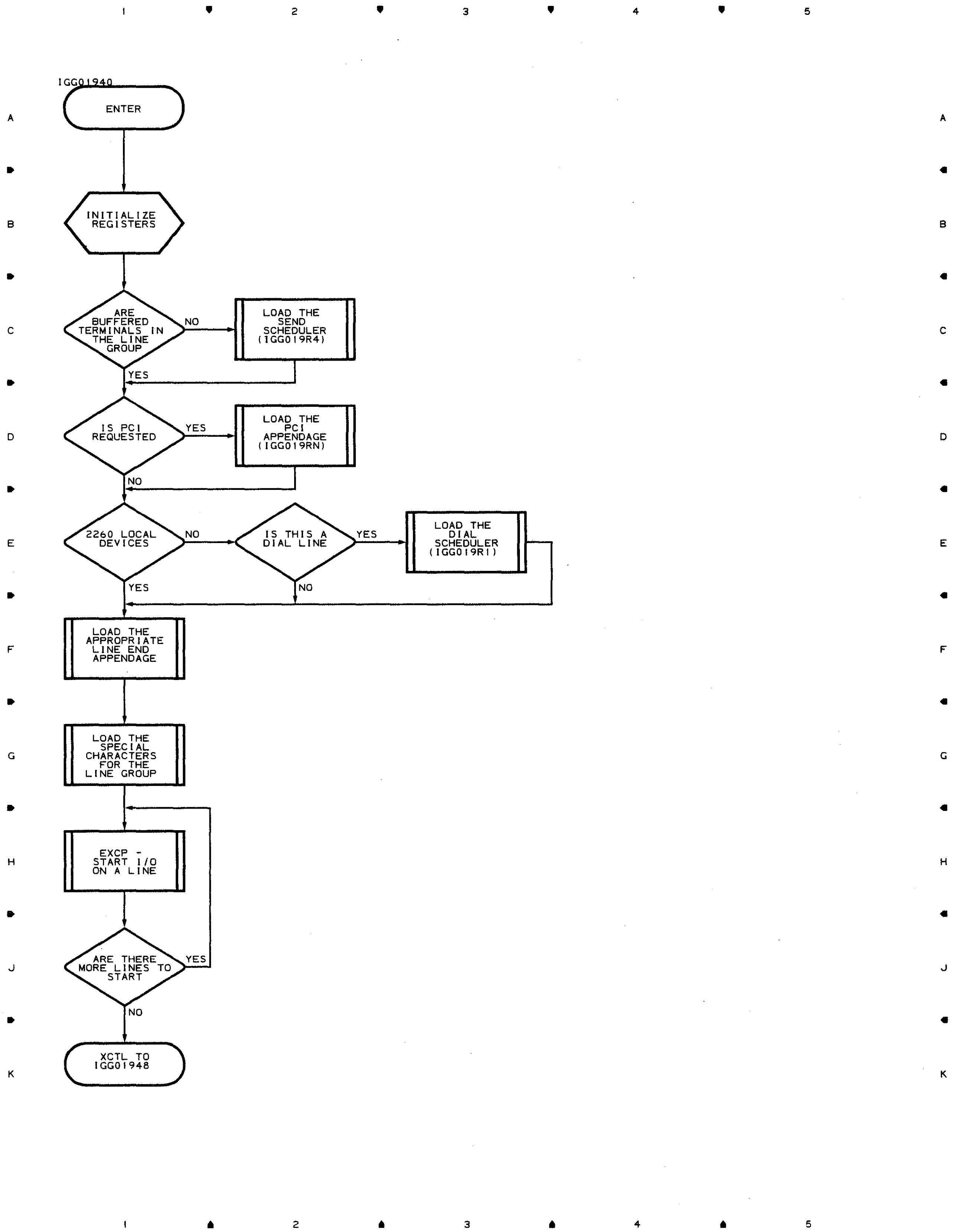

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG01940
ENTER
INITIALIZE REGISTERS
ARE BUFFERED TERMINALS IN THE LINE GROUP
NO
LOAD THE SEND SCHEDULER (IGG019R4)
YES
IS PCI REQUESTED
YES
LOAD THE PCI APPENDAGE (IGG019RN)
NO
2260 LOCAL DEVICES
NO
IS THIS A DIAL LINE
YES
LOAD THE DIAL SCHEDULER (IGG019RI)
YES
NO
LOAD THE APPROPRIATE LINE END APPENDAGE
LOAD THE SPECIAL CHARACTERS FOR THE LINE GROUP
EXCP - START I/O ON A LINE
ARE THERE MORE LINES TO START
YES
NO
XCTL TO IGG01948

Chart LK. LINE GROUP OPEN ROUTINE - LOAD 7

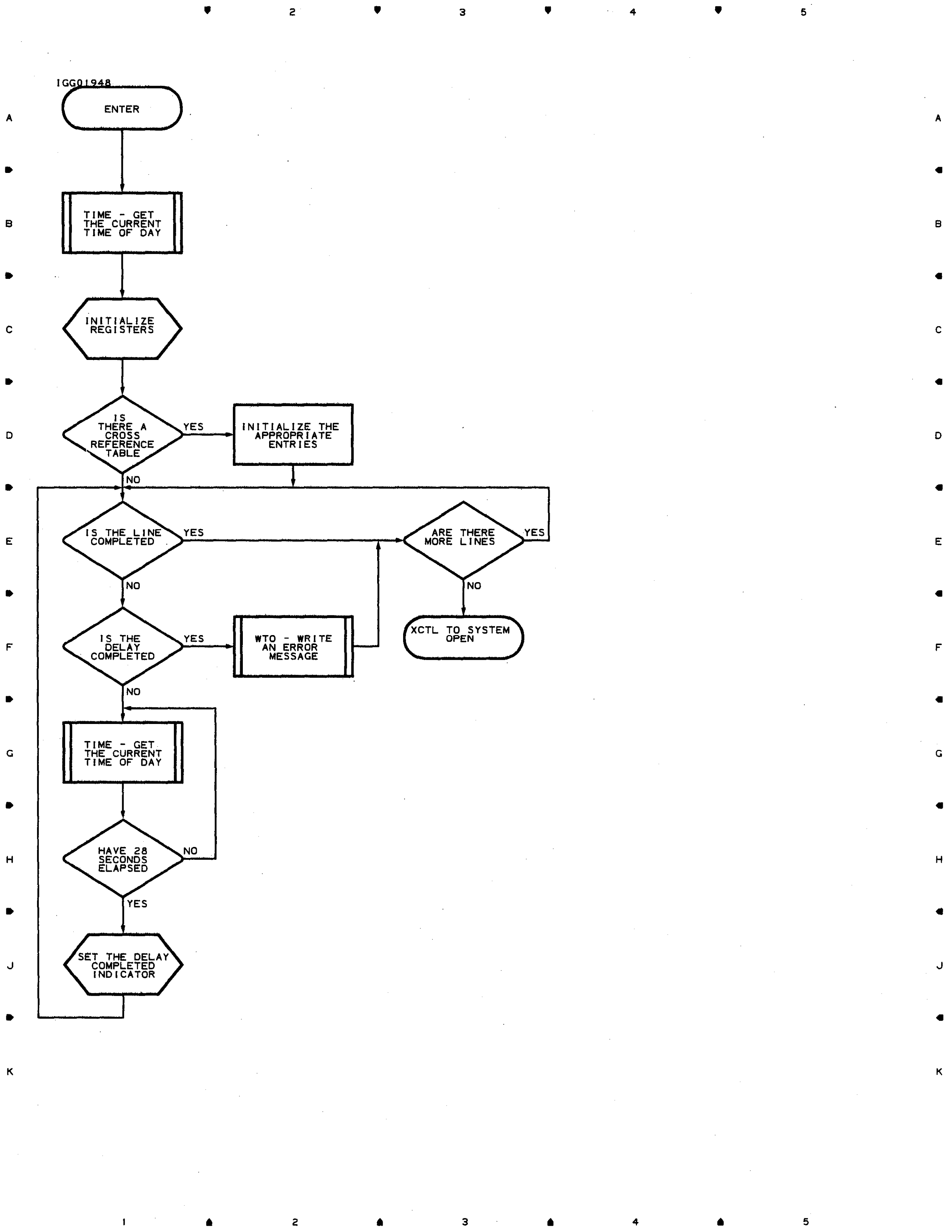

IGG01948
ENTER
TIME - GET THE CURRENT TIME OF DAY
INITIALIZE REGISTERS
IS THERE A CROSS REFERENCE TABLE
YES
INITIALIZE THE APPROPRIATE ENTRIES
NO
IS THE LINE COMPLETED
YES
ARE THERE MORE LINES
YES
NO
XCTL TO SYSTEM OPEN
NO
IS THE DELAY COMPLETED
YES
WTO - WRITE AN ERROR MESSAGE
NO
TIME - GET THE CURRENT TIME OF DAY
HAVE 28 SECONDS ELAPSED
NO
YES
SET THE DELAY COMPLETED INDICATOR
A
B
C
D
E
F
G
H
J
K
1
2
3
4
5

Chart LM RETURN INTERFACE ROUTINE

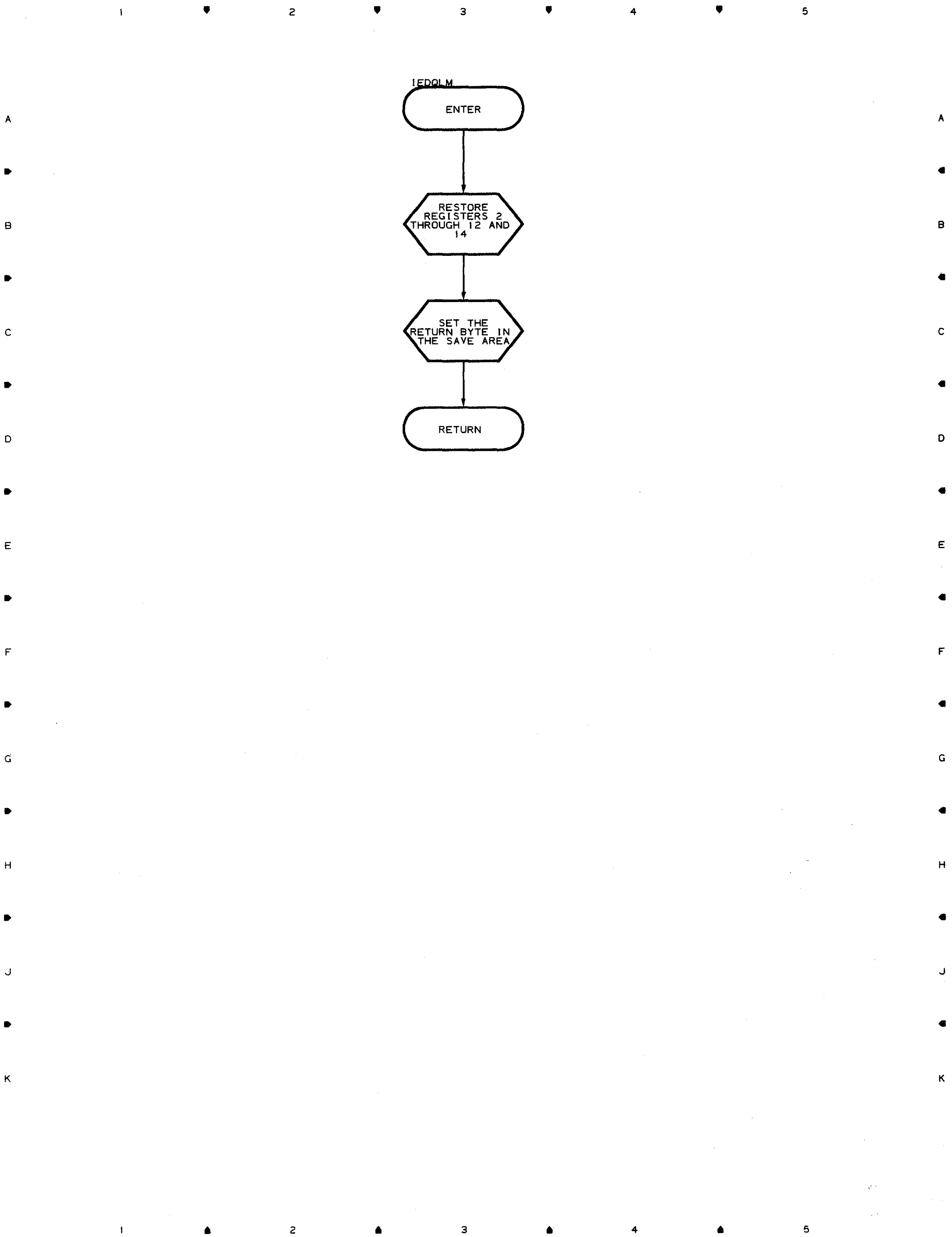

1 2 3 4 5
A B C D E F G H J K
IEDQLM
ENTER
RESTORE REGISTERS 2 THROUGH 12 AND 14
SET THE RETURN BYTE IN THE SAVE AREA
RETURN

Chart L1 DISK MESSAGE QUEUES CLOSE ROUTINE

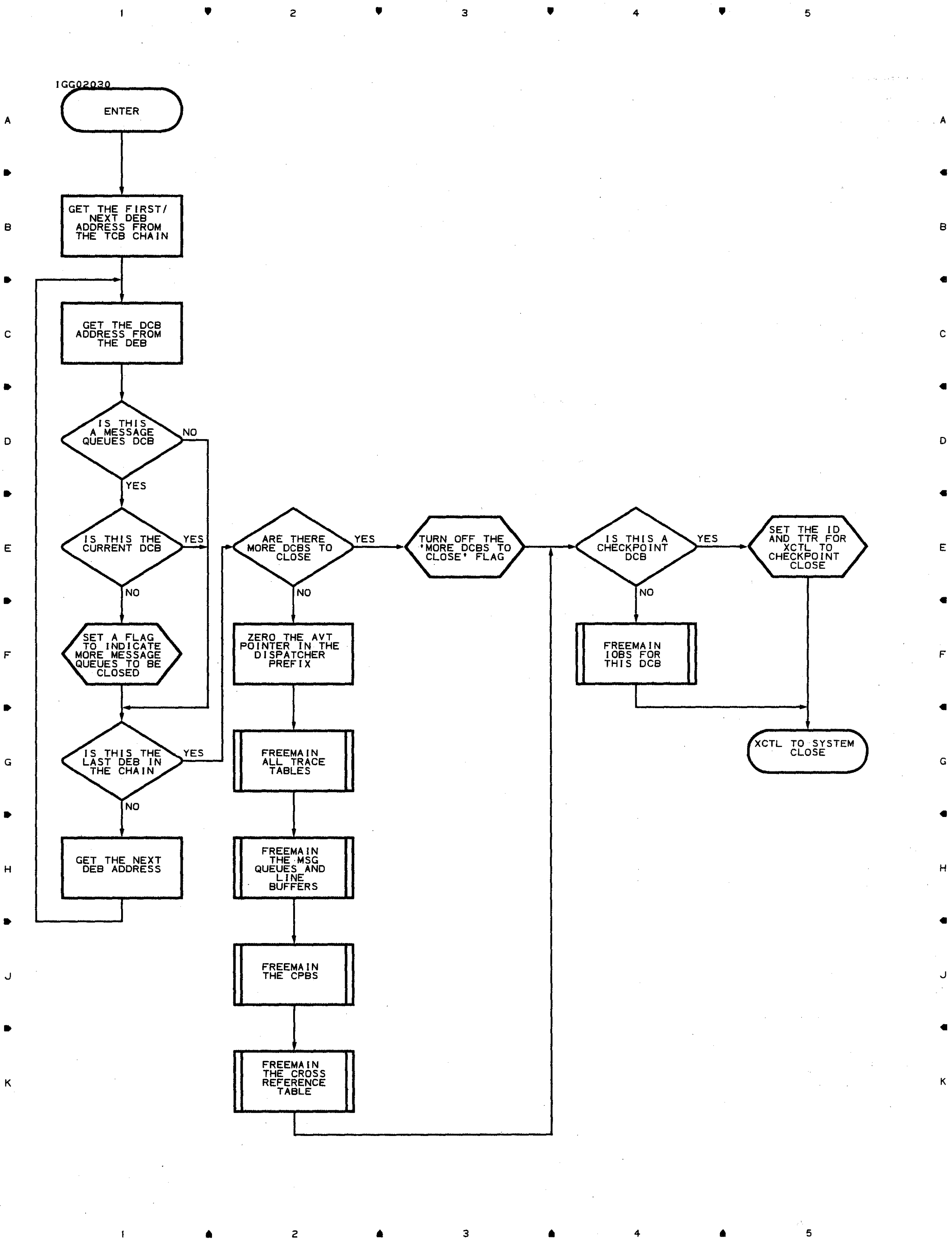

1 2 3 4 5
A B C D E F G H J K
IGG02030
ENTER
GET THE FIRST/ NEXT DEB ADDRESS FROM THE TCB CHAIN
GET THE DCB ADDRESS FROM THE DEB
IS THIS A MESSAGE QUEUES DCB
NO
YES
IS THIS THE CURRENT DCB
YES
NO
SET A FLAG TO INDICATE MORE MESSAGE QUEUES TO BE CLOSED
IS THIS THE LAST DEB IN THE CHAIN
YES
NO
GET THE NEXT DEB ADDRESS
ARE THERE MORE DCBS TO CLOSE
YES
NO
ZERO THE AVT POINTER IN THE DISPATCHER PREFIX
FREEMAIN ALL TRACE TABLES
FREEMAIN THE MSG QUEUES AND LINE BUFFERS
FREEMAIN THE CPBS
FREEMAIN THE CROSS REFERENCE TABLE
TURN OFF THE 'MORE DCBS TO CLOSE' FLAG
IS THIS A CHECKPOINT DCB
YES
NO
FREEMAIN IOBS FOR THIS DCB
SET THE ID AND TTR FOR XCTL TO CHECKPOINT CLOSE
XCTL TO SYSTEM CLOSE

Chart L4 LINE GROUP CLOSE ROUTINE - LOAD 1

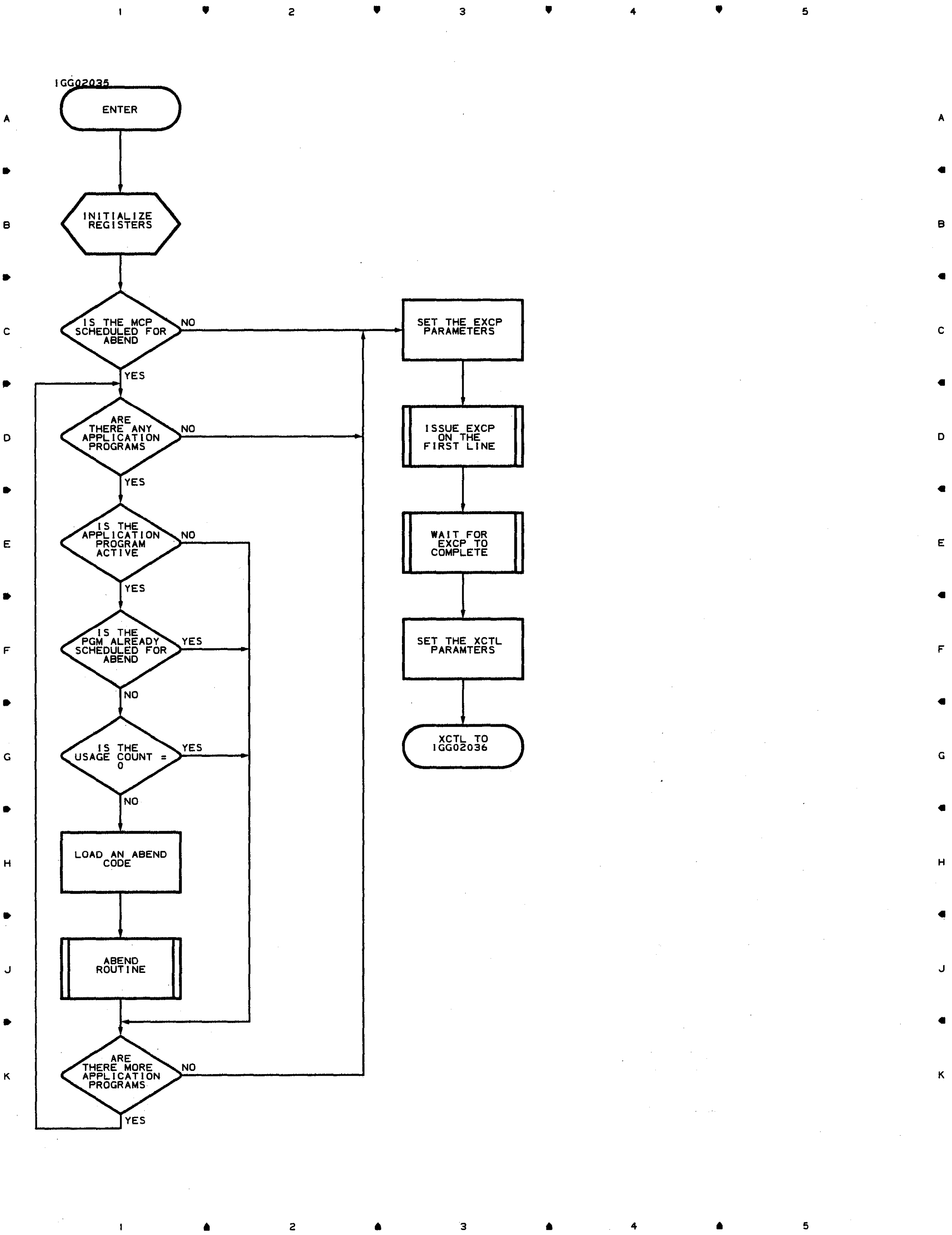

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG02035
ENTER
INITIALIZE REGISTERS
IS THE MCP SCHEDULED FOR ABEND
NO
YES
ARE THERE ANY APPLICATION PROGRAMS
NO
YES
IS THE APPLICATION PROGRAM ACTIVE
NO
YES
IS THE PGM ALREADY SCHEDULED FOR ABEND
YES
NO
IS THE USAGE COUNT = 0
YES
NO
LOAD AN ABEND CODE
ABEND ROUTINE
ARE THERE MORE APPLICATION PROGRAMS
NO
YES
SET THE EXCP PARAMETERS
ISSUE EXCP ON THE FIRST LINE
WAIT FOR EXCP TO COMPLETE
SET THE XCTL PARAMTERS
XCTL TO IGG02036

Chart L5 LINE GROUP CLOSE ROUTINE - LOAD 2

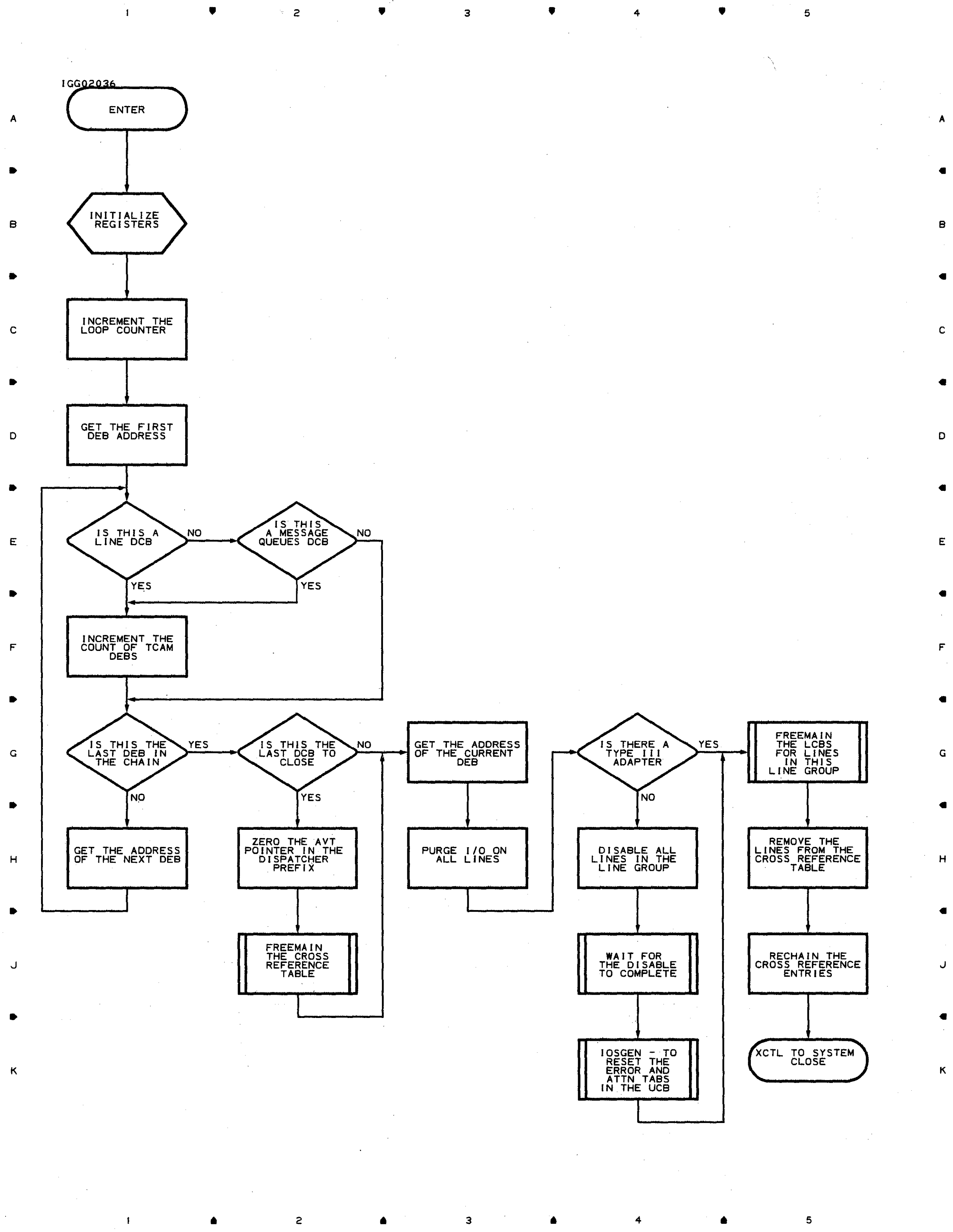
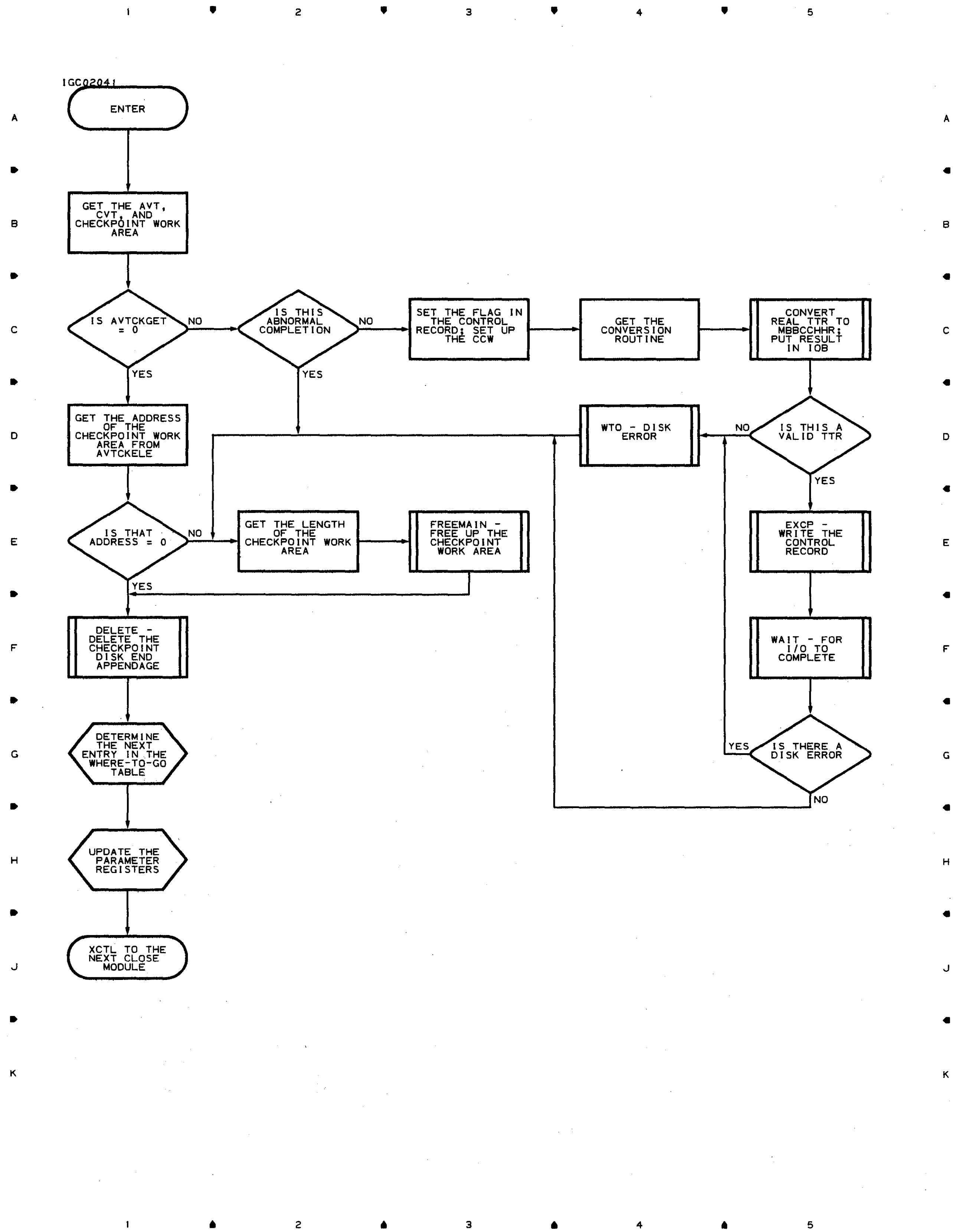

IGG02036
ENTER
INITIALIZE REGISTERS
INCREMENT THE LOOP COUNTER
GET THE FIRST DEB ADDRESS
IS THIS A LINE DCB
NO
YES
IS THIS A MESSAGE QUEUES DCB
NO
YES
INCREMENT THE COUNT OF TCAM DEBS
IS THIS THE LAST DEB IN THE CHAIN
YES
NO
GET THE ADDRESS OF THE NEXT DEB
IS THIS THE LAST DCB TO CLOSE
NO
YES
ZERO THE AVT POINTER IN THE DISPATCHER PREFIX
FREEMAIN THE CROSS REFERENCE TABLE
GET THE ADDRESS OF THE CURRENT DEB
PURGE I/O ON ALL LINES
IS THERE A TYPE III ADAPTER
YES
NO
DISABLE ALL LINES IN THE LINE GROUP
WAIT FOR THE DISABLE TO COMPLETE
IOSGEN - TO RESET THE ERROR AND ATTN TABS IN THE UCB
FREEMAIN THE LCBS FOR LINES IN THIS LINE GROUP
REMOVE THE LINES FROM THE CROSS REFERENCE TABLE
RECHAIN THE CROSS REFERENCE ENTRIES
XCTL TO SYSTEM CLOSE
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart L6 CHECKPOINT CLOSE ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGC0204I
ENTER
GET THE AVT, CVT, AND CHECKPOINT WORK AREA
IS AVTCKGET = 0
NO
YES
IS THIS ABNORMAL COMPLETION
NO
YES
SET THE FLAG IN THE CONTROL RECORD; SET UP THE CCW
GET THE CONVERSION ROUTINE
CONVERT REAL TTR TO MBBCCHHR; PUT RESULT IN IOB
GET THE ADDRESS OF THE CHECKPOINT WORK AREA FROM AVTCKELE
WTO - DISK ERROR
NO
IS THIS A VALID TTR
YES
IS THAT ADDRESS = 0
NO
YES
GET THE LENGTH OF THE CHECKPOINT WORK AREA
FREEMAIN - FREE UP THE CHECKPOINT WORK AREA
EXCP - WRITE THE CONTROL RECORD
DELETE - DELETE THE CHECKPOINT DISK END APPENDAGE
WAIT - FOR I/O TO COMPLETE
DETERMINE THE NEXT ENTRY IN THE WHERE-TO-GO TABLE
YES
IS THERE A DISK ERROR
NO
UPDATE THE PARAMETER REGISTERS
XCTL TO THE NEXT CLOSE MODULE

Chart L7 GET/PUT AND READ/WRITE OPEN EXECUTOR - LOAD 1

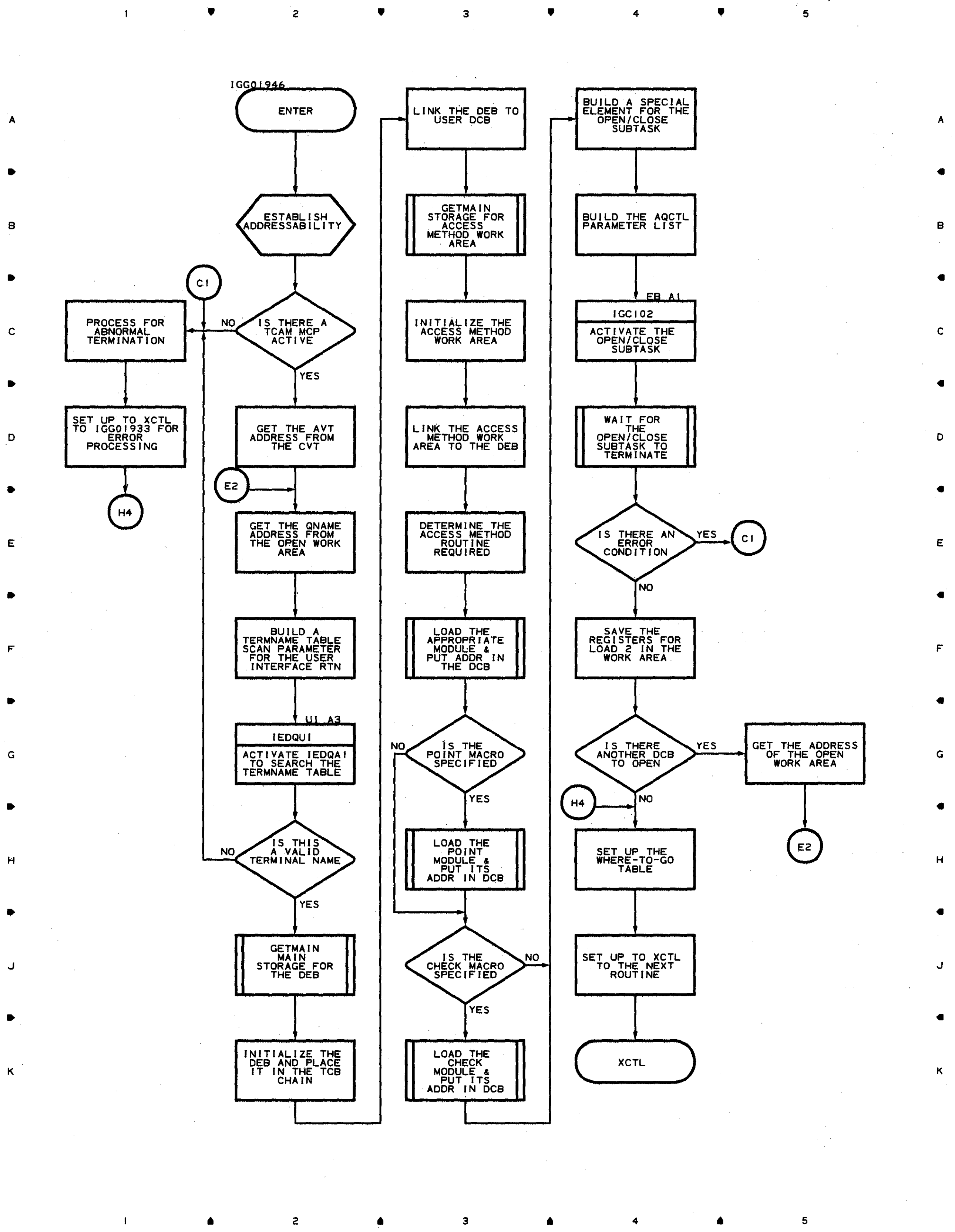

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG01946
ENTER
ESTABLISH ADDRESSABILITY
C1
IS THERE A TCAM MCP ACTIVE
NO
YES
PROCESS FOR ABNORMAL TERMINATION
SET UP TO XCTL TO IGG01933 FOR ERROR PROCESSING
H4
GET THE AVT ADDRESS FROM THE CVT
E2
GET THE QNAME ADDRESS FROM THE OPEN WORK AREA
BUILD A TERMNAME TABLE SCAN PARAMETER FOR THE USER INTERFACE RTN
U1 A3
IEDQUI
ACTIVATE IEDQAI TO SEARCH THE TERMNAME TABLE
IS THIS A VALID TERMINAL NAME
NO
YES
GETMAIN MAIN STORAGE FOR THE DEB
INITIALIZE THE DEB AND PLACE IT IN THE TCB CHAIN
LINK THE DEB TO USER DCB
GETMAIN STORAGE FOR ACCESS METHOD WORK AREA
INITIALIZE THE ACCESS METHOD WORK AREA
LINK THE ACCESS METHOD WORK AREA TO THE DEB
DETERMINE THE ACCESS METHOD ROUTINE REQUIRED
LOAD THE APPROPRIATE MODULE & PUT ADDR IN THE DCB
IS THE POINT MACRO SPECIFIED
NO
YES
LOAD THE POINT MODULE & PUT ITS ADDR IN DCB
IS THE CHECK MACRO SPECIFIED
NO
YES
LOAD THE CHECK MODULE & PUT ITS ADDR IN DCB
BUILD A SPECIAL ELEMENT FOR THE OPEN/CLOSE SUBTASK
BUILD THE AQCTL PARAMETER LIST
EB A1
IGC102
ACTIVATE THE OPEN/CLOSE SUBTASK
WAIT FOR THE OPEN/CLOSE SUBTASK TO TERMINATE
IS THERE AN ERROR CONDITION
YES
C1
NO
SAVE THE REGISTERS FOR LOAD 2 IN THE WORK AREA
IS THERE ANOTHER DCB TO OPEN
YES
GET THE ADDRESS OF THE OPEN WORK AREA
E2
NO
H4
SET UP THE WHERE-TO-GO TABLE
SET UP TO XCTL TO THE NEXT ROUTINE
XCTL

Chart L8 GET/PUT AND READ/WRITE OPEN EXECUTOR - LOAD 2

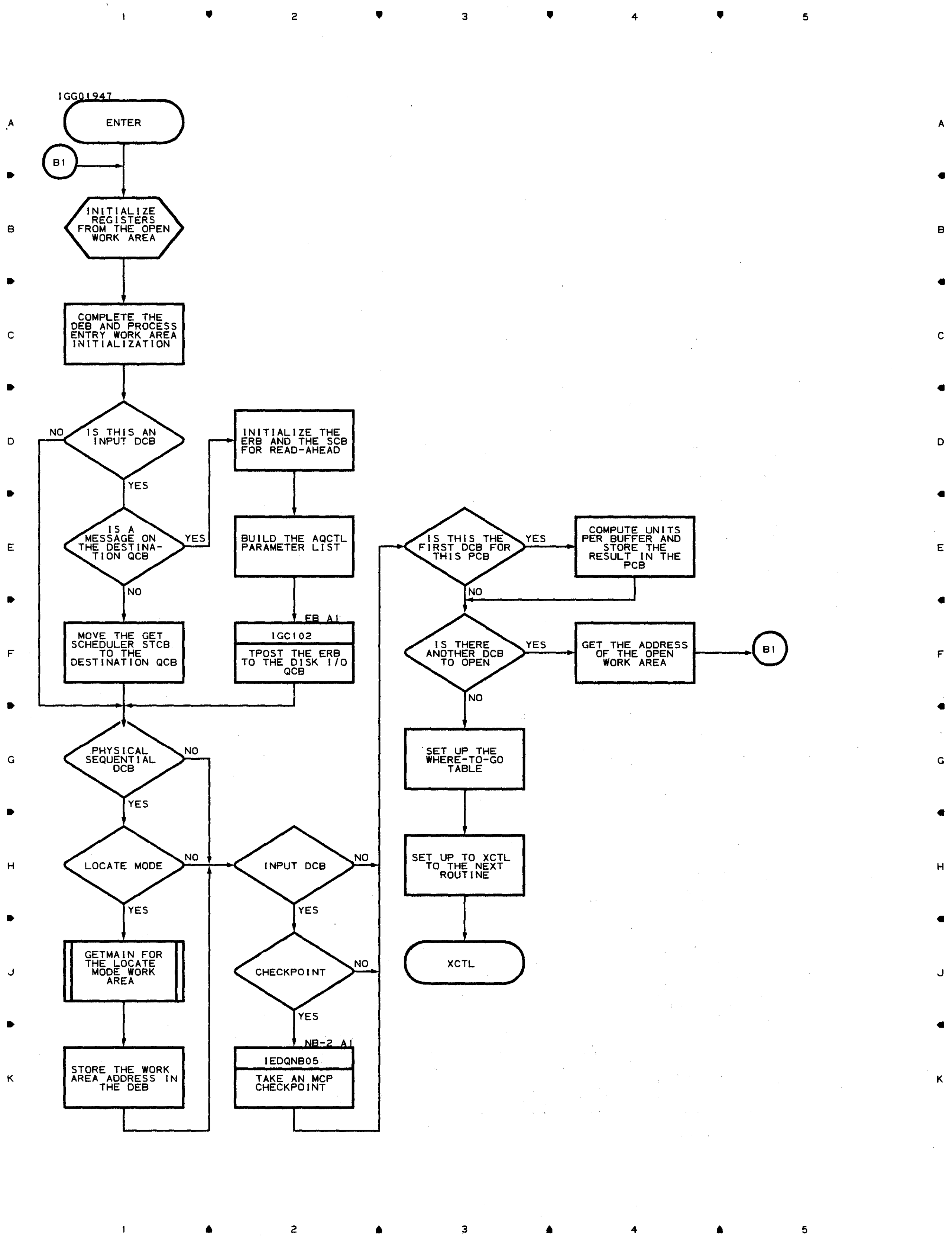

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG01947
ENTER
B1
INITIALIZE REGISTERS FROM THE OPEN WORK AREA
COMPLETE THE DEB AND PROCESS ENTRY WORK AREA INITIALIZATION
IS THIS AN INPUT DCB
NO
YES
INITIALIZE THE ERB AND THE SCB FOR READ-AHEAD
IS A MESSAGE ON THE DESTINA-TION QCB
YES
NO
BUILD THE AQCTL PARAMETER LIST
EB A1
IGC102
TPOST THE ERB TO THE DISK I/O QCB
MOVE THE GET SCHEDULER STCB TO THE DESTINATION QCB
IS THIS THE FIRST DCB FOR THIS PCB
YES
NO
COMPUTE UNITS PER BUFFER AND STORE THE RESULT IN THE PCB
IS THERE ANOTHER DCB TO OPEN
YES
NO
GET THE ADDRESS OF THE OPEN WORK AREA
B1
PHYSICAL SEQUENTIAL DCB
NO
YES
SET UP THE WHERE-TO-GO TABLE
LOCATE MODE
NO
YES
INPUT DCB
NO
YES
SET UP TO XCTL TO THE NEXT ROUTINE
GETMAIN FOR THE LOCATE MODE WORK AREA
CHECKPOINT
NO
YES
XCTL
NB-2 A1
IEDQNB05
TAKE AN MCP CHECKPOINT
STORE THE WORK AREA ADDRESS IN THE DEB

Chart L9 GET/PUT AND READ/WRITE CLOSE EXECUTOR - LOAD 1

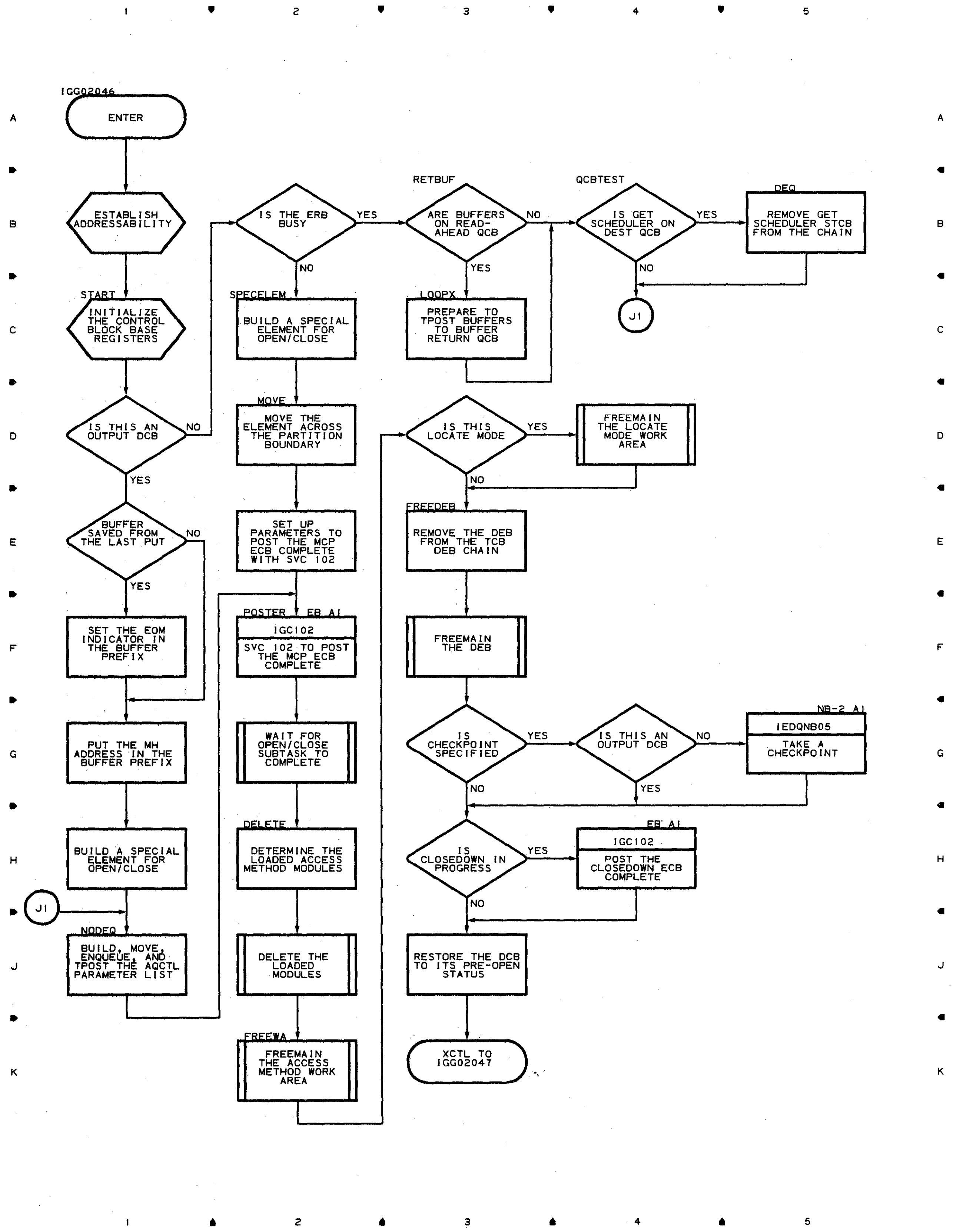
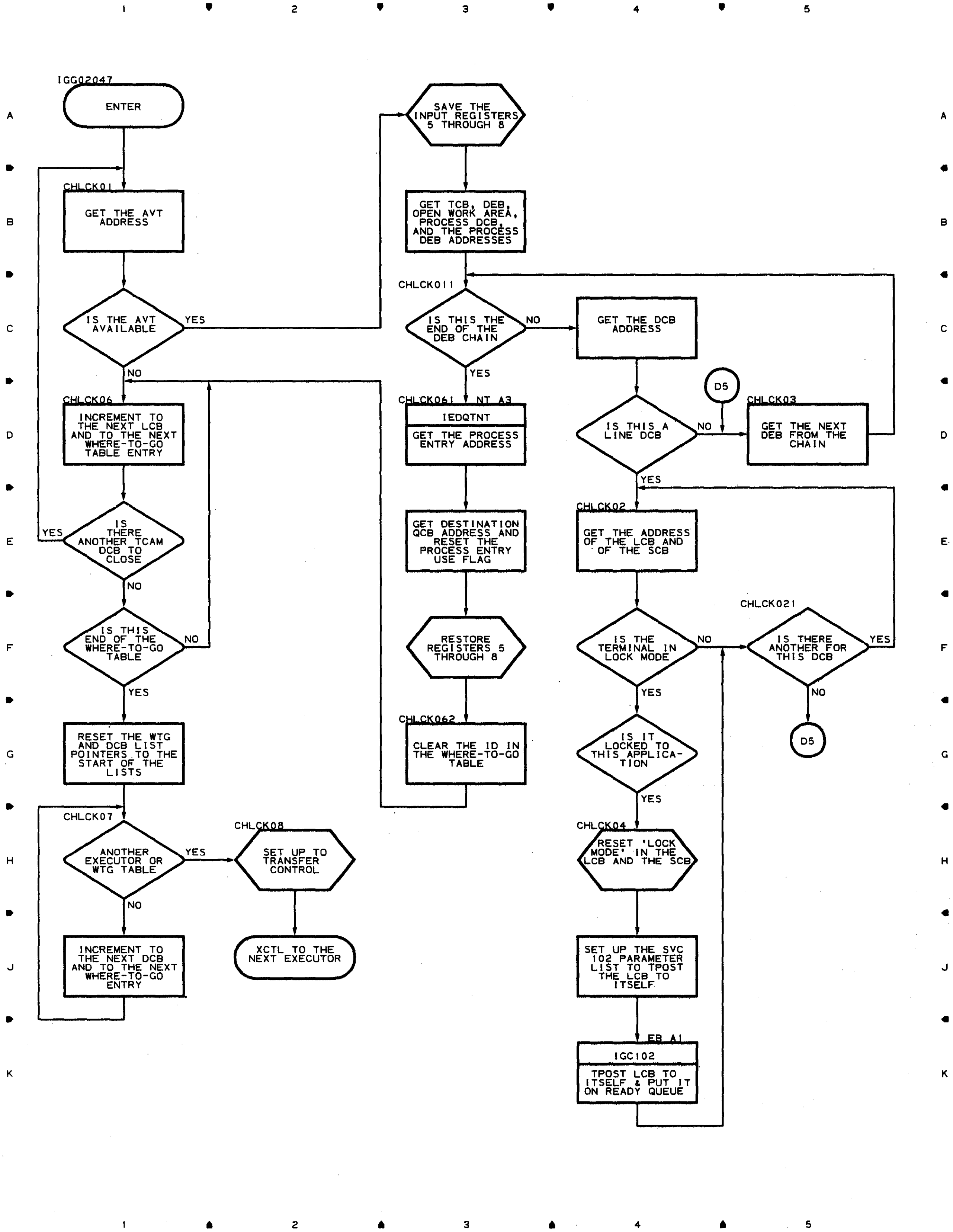

IGG02046
ENTER
ESTABLISH ADDRESSABILITY
START
INITIALIZE THE CONTROL BLOCK BASE REGISTERS
IS THIS AN OUTPUT DCB
NO
YES
BUFFER SAVED FROM THE LAST PUT
NO
YES
SET THE EOM INDICATOR IN THE BUFFER PREFIX
PUT THE MH ADDRESS IN THE BUFFER PREFIX
BUILD A SPECIAL ELEMENT FOR OPEN/CLOSE
J1
NODEQ
BUILD, MOVE, ENQUEUE, AND TPOST THE AQCTL PARAMETER LIST
IS THE ERB BUSY
YES
NO
SPECELEM
BUILD A SPECIAL ELEMENT FOR OPEN/CLOSE
MOVE
MOVE THE ELEMENT ACROSS THE PARTITION BOUNDARY
SET UP PARAMETERS TO POST THE MCP ECB COMPLETE WITH SVC 102
POSTER
EB A1
IGC102
SVC 102 TO POST THE MCP ECB COMPLETE
WAIT FOR OPEN/CLOSE SUBTASK TO COMPLETE
DELETE
DETERMINE THE LOADED ACCESS METHOD MODULES
DELETE THE LOADED MODULES
FREEWA
FREEMAIN THE ACCESS METHOD WORK AREA
RETBUF
ARE BUFFERS ON READ-AHEAD QCB
NO
YES
LOOPX
PREPARE TO TPOST BUFFERS TO BUFFER RETURN QCB
IS THIS LOCATE MODE
YES
NO
FREEMAIN THE LOCATE MODE WORK AREA
FREEDEB
REMOVE THE DEB FROM THE TCB DEB CHAIN
FREEMAIN THE DEB
IS CHECKPOINT SPECIFIED
YES
NO
IS THIS AN OUTPUT DCB
NO
YES
NB-2 A1
IEDQNB05
TAKE A CHECKPOINT
IS CLOSEDOWN IN PROGRESS
YES
NO
EB A1
IGC102
POST THE CLOSEDOWN ECB COMPLETE
RESTORE THE DCB TO ITS PRE-OPEN STATUS
XCTL TO IGG02047
QCBTEST
IS GET SCHEDULER ON DEST QCB
YES
NO
DEQ
REMOVE GET SCHEDULER STCB FROM THE CHAIN
J1
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart L10 GET/PUT AND READ/WRITE CLOSE EXECUTOR - LOAD 2



IGG02047
ENTER
CHLCK01
GET THE AVT ADDRESS
IS THE AVT AVAILABLE
YES
NO
CHLCK06
INCREMENT TO THE NEXT LCB AND TO THE NEXT WHERE-TO-GO TABLE ENTRY
IS THERE ANOTHER TCAM DCB TO CLOSE
YES
NO
IS THIS END OF THE WHERE-TO-GO TABLE
NO
YES
RESET THE WTG AND DCB LIST POINTERS TO THE START OF THE LISTS
CHLCK07
ANOTHER EXECUTOR OR WTG TABLE
YES
NO
INCREMENT TO THE NEXT DCB AND TO THE NEXT WHERE-TO-GO ENTRY
CHLCK08
SET UP TO TRANSFER CONTROL
XCTL TO THE NEXT EXECUTOR
SAVE THE INPUT REGISTERS 5 THROUGH 8
GET TCB, DEB, OPEN WORK AREA, PROCESS DCB, AND THE PROCESS DEB ADDRESSES
CHLCK011
IS THIS THE END OF THE DEB CHAIN
NO
YES
CHLCK061
NT A3
IEDQTNT
GET THE PROCESS ENTRY ADDRESS
GET DESTINATION QCB ADDRESS AND RESET THE PROCESS ENTRY USE FLAG
RESTORE REGISTERS 5 THROUGH 8
CHLCK062
CLEAR THE ID IN THE WHERE-TO-GO TABLE
GET THE DCB ADDRESS
IS THIS A LINE DCB
NO
YES
D5
CHLCK03
GET THE NEXT DEB FROM THE CHAIN
CHLCK02
GET THE ADDRESS OF THE LCB AND OF THE SCB
IS THE TERMINAL IN LOCK MODE
NO
YES
CHLCK021
IS THERE ANOTHER FOR THIS DCB
YES
NO
D5
IS IT LOCKED TO THIS APPLICATION
YES
CHLCK04
RESET 'LOCK MODE' IN THE LCB AND THE SCB
SET UP THE SVC 102 PARAMETER LIST TO TPOST THE LCB TO ITSELF
EB A1
IGC102
TPOST LCB TO ITSELF & PUT IT ON READY QUEUE
A
B
C
D
E
F
G
H
J
K
1
2
3
4
5

Chart MA CHECKPOINT OPEN ROUTINE

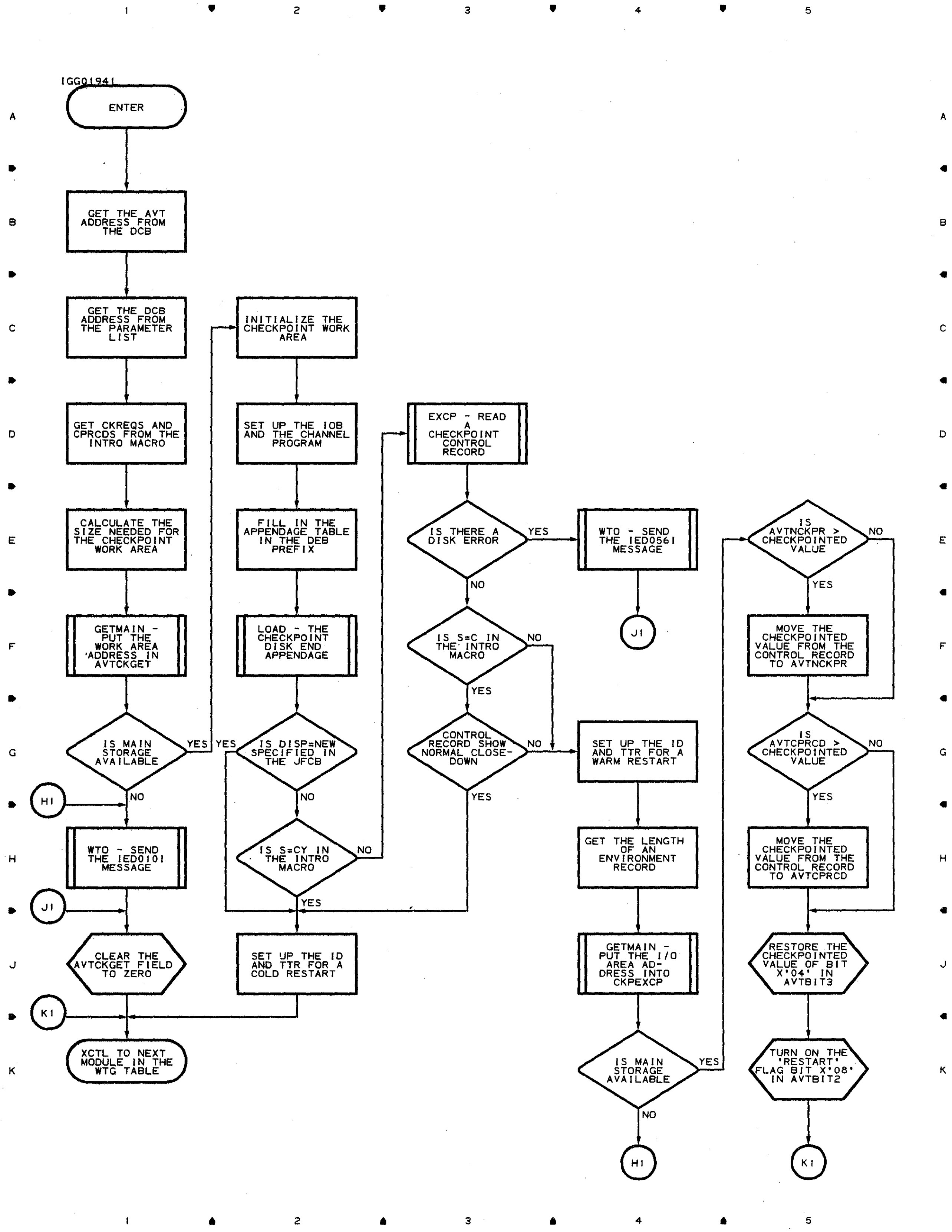

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG01941
ENTER
GET THE AVT ADDRESS FROM THE DCB
GET THE DCB ADDRESS FROM THE PARAMETER LIST
GET CKREQS AND CPRCDS FROM THE INTRO MACRO
CALCULATE THE SIZE NEEDED FOR THE CHECKPOINT WORK AREA
GETMAIN - PUT THE WORK AREA ADDRESS IN AVTCKGET
IS MAIN STORAGE AVAILABLE
YES
NO
H1
WTO - SEND THE IED010I MESSAGE
J1
CLEAR THE AVTCKGET FIELD TO ZERO
K1
XCTL TO NEXT MODULE IN THE WTG TABLE
INITIALIZE THE CHECKPOINT WORK AREA
SET UP THE IOB AND THE CHANNEL PROGRAM
FILL IN THE APPENDAGE TABLE IN THE DEB PREFIX
LOAD - THE CHECKPOINT DISK END APPENDAGE
IS DISP=NEW SPECIFIED IN THE JFCB
YES
NO
IS S=CY IN THE INTRO MACRO
NO
YES
SET UP THE ID AND TTR FOR A COLD RESTART
EXCP - READ A CHECKPOINT CONTROL RECORD
IS THERE A DISK ERROR
YES
NO
IS S=C IN THE INTRO MACRO
NO
YES
CONTROL RECORD SHOW NORMAL CLOSE-DOWN
NO
YES
WTO - SEND THE IED056I MESSAGE
J1
SET UP THE ID AND TTR FOR A WARM RESTART
GET THE LENGTH OF AN ENVIRONMENT RECORD
GETMAIN - PUT THE I/O AREA ADDRESS INTO CKPEXCP
IS MAIN STORAGE AVAILABLE
YES
NO
H1
IS AVTNCKPR > CHECKPOINTED VALUE
NO
YES
MOVE THE CHECKPOINTED VALUE FROM THE CONTROL RECORD TO AVTNCKPR
IS AVTCPRCD > CHECKPOINTED VALUE
NO
YES
MOVE THE CHECKPOINTED VALUE FROM THE CONTROL RECORD TO AVTCPRCD
RESTORE THE CHECKPOINTED VALUE OF BIT X'04' IN AVTBIT3
TURN ON THE 'RESTART' FLAG BIT X'08' IN AVTBIT2
K1

Chart MB-1 CHECKPOINT DISK INITIALIZATION ROUTINE

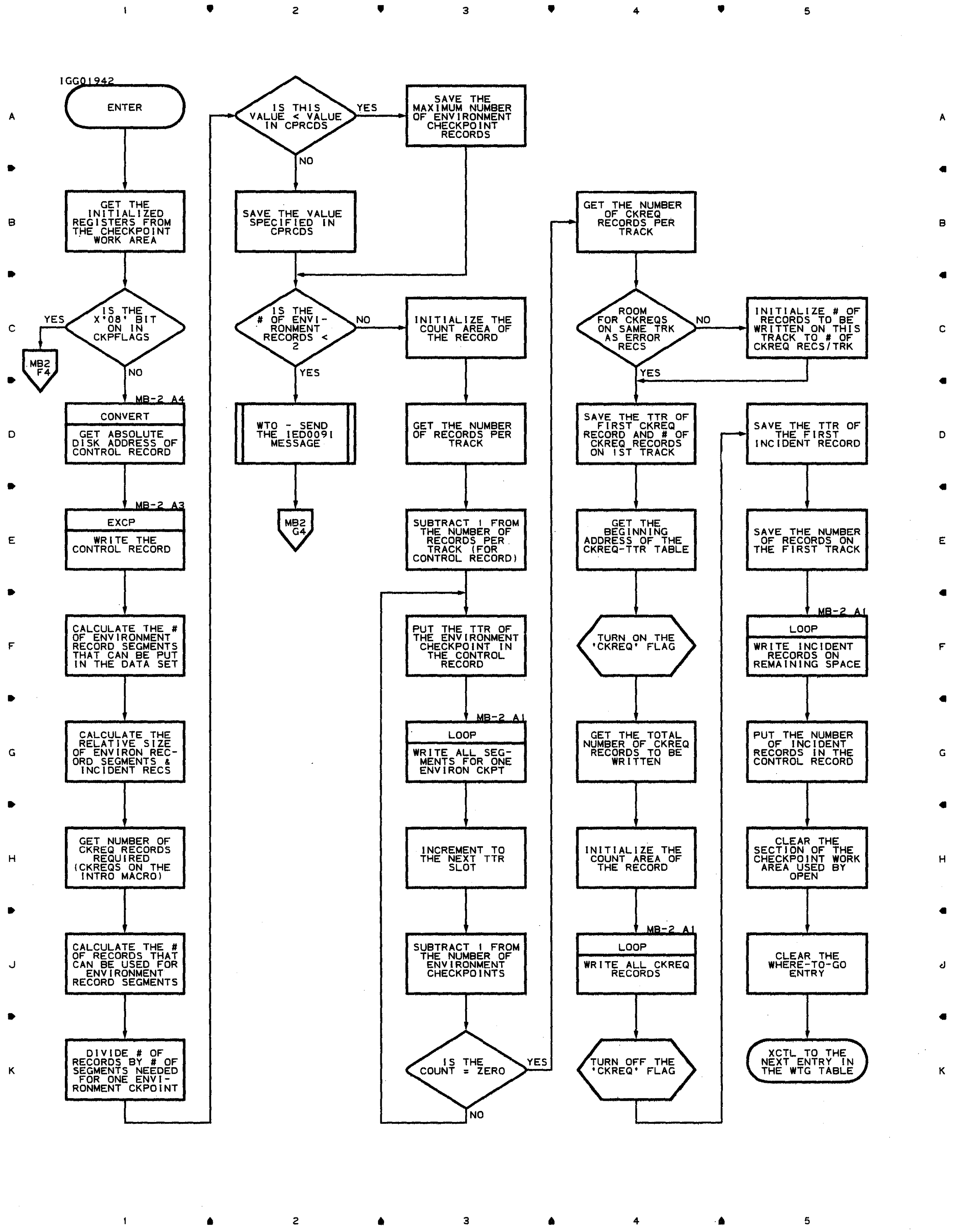
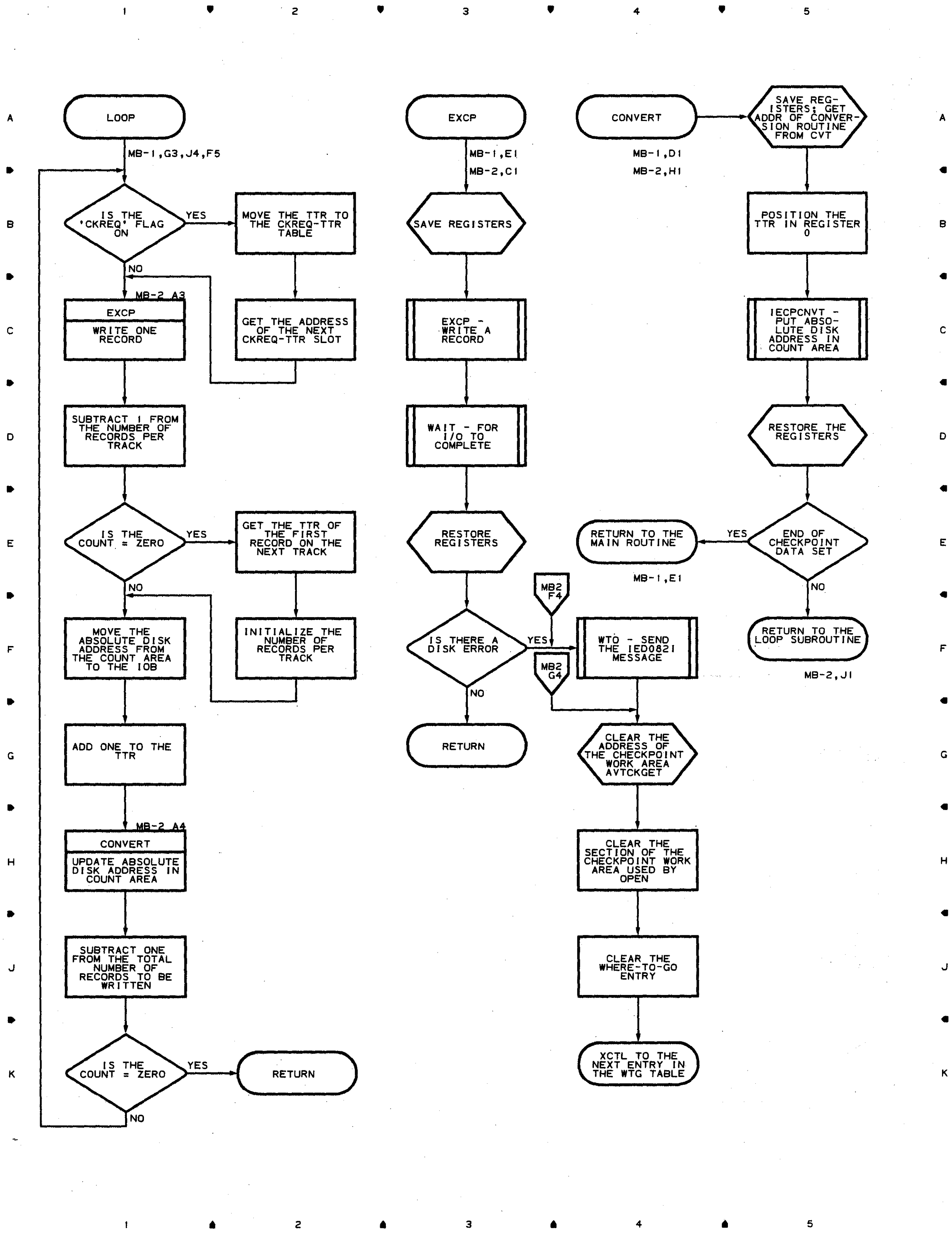
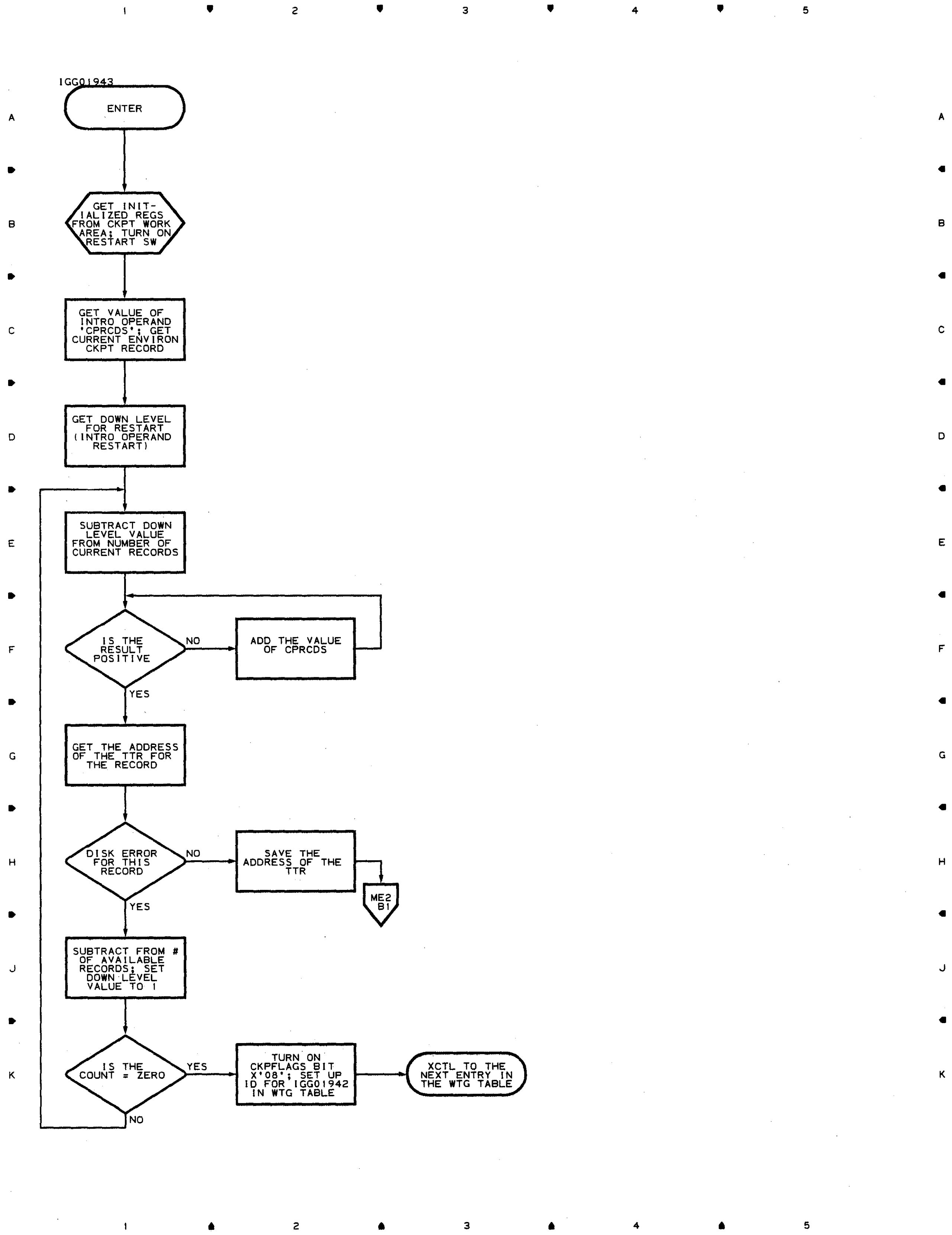

IGG01942
ENTER
GET THE INITIALIZED REGISTERS FROM THE CHECKPOINT WORK AREA
IS THE X'08' BIT ON IN CKPFLAGS
YES
MB2 F4
NO
MB-2 A4
CONVERT
GET ABSOLUTE DISK ADDRESS OF CONTROL RECORD
MB-2 A3
EXCP
WRITE THE CONTROL RECORD
CALCULATE THE # OF ENVIRONMENT RECORD SEGMENTS THAT CAN BE PUT IN THE DATA SET
CALCULATE THE RELATIVE SIZE OF ENVIRON REC- ORD SEGMENTS & INCIDENT RECS
GET NUMBER OF CKREQ RECORDS REQUIRED (CKREQS ON THE INTRO MACRO)
CALCULATE THE # OF RECORDS THAT CAN BE USED FOR ENVIRONMENT RECORD SEGMENTS
DIVIDE # OF RECORDS BY # OF SEGMENTS NEEDED FOR ONE ENVI- RONMENT CKPOINT
IS THIS VALUE < VALUE IN CPRCDS
YES
NO
SAVE THE MAXIMUM NUMBER OF ENVIRONMENT CHECKPOINT RECORDS
SAVE THE VALUE SPECIFIED IN CPRCDS
IS THE # OF ENVI- RONMENT RECORDS < 2
NO
YES
WTO - SEND THE IED009I MESSAGE
MB2 G4
INITIALIZE THE COUNT AREA OF THE RECORD
GET THE NUMBER OF RECORDS PER TRACK
SUBTRACT 1 FROM THE NUMBER OF RECORDS PER TRACK (FOR CONTROL RECORD)
PUT THE TTR OF THE ENVIRONMENT CHECKPOINT IN THE CONTROL RECORD
MB-2 A1
LOOP
WRITE ALL SEG- MENTS FOR ONE ENVIRON CKPT
INCREMENT TO THE NEXT TTR SLOT
SUBTRACT 1 FROM THE NUMBER OF ENVIRONMENT CHECKPOINTS
IS THE COUNT = ZERO
YES
NO
GET THE NUMBER OF CKREQ RECORDS PER TRACK
ROOM FOR CKREQS ON SAME TRK AS ERROR RECS
NO
YES
INITIALIZE # OF RECORDS TO BE WRITTEN ON THIS TRACK TO # OF CKREQ RECS/TRK
SAVE THE TTR OF FIRST CKREQ RECORD AND # OF CKREQ RECORDS ON 1ST TRACK
GET THE BEGINNING ADDRESS OF THE CKREQ-TTR TABLE
TURN ON THE 'CKREQ' FLAG
GET THE TOTAL NUMBER OF CKREQ RECORDS TO BE WRITTEN
INITIALIZE THE COUNT AREA OF THE RECORD
MB-2 A1
LOOP
WRITE ALL CKREQ RECORDS
TURN OFF THE 'CKREQ' FLAG
SAVE THE TTR OF THE FIRST INCIDENT RECORD
SAVE THE NUMBER OF RECORDS ON THE FIRST TRACK
MB-2 A1
LOOP
WRITE INCIDENT RECORDS ON REMAINING SPACE
PUT THE NUMBER OF INCIDENT RECORDS IN THE CONTROL RECORD
CLEAR THE SECTION OF THE CHECKPOINT WORK AREA USED BY OPEN
CLEAR THE WHERE-TO-GO ENTRY
XCTL TO THE NEXT ENTRY IN THE WTG TABLE
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart MB-2 CHECKPOINT DISK INITIALIZATION ROUTINE



1
2
3
4
5
A
LOOP
EXCP
CONVERT
SAVE REGISTERS; GET ADDR OF CONVERSION ROUTINE FROM CVT
MB-1,G3,J4,F5
MB-1,E1
MB-2,C1
MB-1,D1
MB-2,H1
B
IS THE 'CKREQ' FLAG ON
YES
MOVE THE TTR TO THE CKREQ-TTR TABLE
SAVE REGISTERS
POSITION THE TTR IN REGISTER 0
NO
MB-2 A3
C
EXCP
WRITE ONE RECORD
GET THE ADDRESS OF THE NEXT CKREQ-TTR SLOT
EXCP - WRITE A RECORD
IECPCNVT - PUT ABSOLUTE DISK ADDRESS IN COUNT AREA
D
SUBTRACT 1 FROM THE NUMBER OF RECORDS PER TRACK
WAIT - FOR I/O TO COMPLETE
RESTORE THE REGISTERS
E
IS THE COUNT = ZERO
YES
GET THE TTR OF THE FIRST RECORD ON THE NEXT TRACK
RESTORE REGISTERS
RETURN TO THE MAIN ROUTINE
YES
END OF CHECKPOINT DATA SET
MB-1,E1
NO
MB2 F4
NO
F
MOVE THE ABSOLUTE DISK ADDRESS FROM THE COUNT AREA TO THE IOB
INITIALIZE THE NUMBER OF RECORDS PER TRACK
IS THERE A DISK ERROR
YES
WTO - SEND THE IED082I MESSAGE
RETURN TO THE LOOP SUBROUTINE
MB2 G4
MB-2,J1
NO
G
ADD ONE TO THE TTR
RETURN
CLEAR THE ADDRESS OF THE CHECKPOINT WORK AREA AVTCKGET
MB-2 A4
H
CONVERT
UPDATE ABSOLUTE DISK ADDRESS IN COUNT AREA
CLEAR THE SECTION OF THE CHECKPOINT WORK AREA USED BY OPEN
J
SUBTRACT ONE FROM THE TOTAL NUMBER OF RECORDS TO BE WRITTEN
CLEAR THE WHERE-TO-GO ENTRY
K
IS THE COUNT = ZERO
YES
RETURN
XCTL TO THE NEXT ENTRY IN THE WTG TABLE
NO

Chart ME-1 CHECKPOINT/RESTART FROM ENVIRONMENT RECORD ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG01943
ENTER
GET INIT-IALIZED REGS FROM CKPT WORK AREA; TURN ON RESTART SW
GET VALUE OF INTRO OPERAND 'CPRCDS'; GET CURRENT ENVIRON CKPT RECORD
GET DOWN LEVEL FOR RESTART (INTRO OPERAND RESTART)
SUBTRACT DOWN LEVEL VALUE FROM NUMBER OF CURRENT RECORDS
IS THE RESULT POSITIVE
NO
ADD THE VALUE OF CPRCDS
YES
GET THE ADDRESS OF THE TTR FOR THE RECORD
DISK ERROR FOR THIS RECORD
NO
SAVE THE ADDRESS OF THE TTR
ME2 B1
YES
SUBTRACT FROM # OF AVAILABLE RECORDS; SET DOWN LEVEL VALUE TO 1
IS THE COUNT = ZERO
YES
TURN ON CKPFLAGS BIT X'08'; SET UP ID FOR IGG01942 IN WTG TABLE
XCTL TO THE NEXT ENTRY IN THE WTG TABLE
NO

Chart ME-2 CHECKPOINT/RESTART FROM ENVIRONMENT RECORD ROUTINE

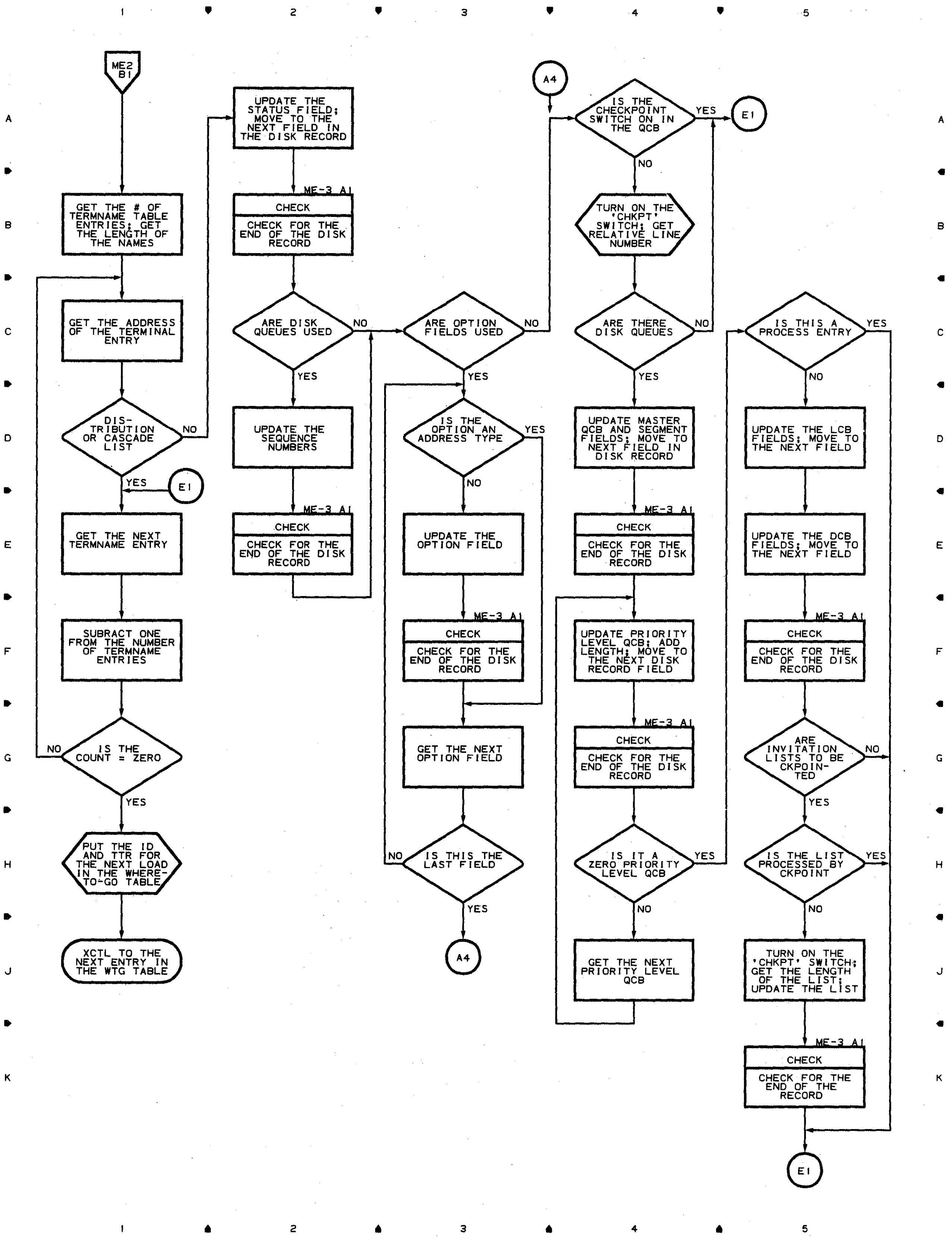

ME2 B1
GET THE # OF TERMNAME TABLE ENTRIES; GET THE LENGTH OF THE NAMES
GET THE ADDRESS OF THE TERMINAL ENTRY
DIS-TRIBUTION OR CASCADE LIST
NO
YES
E1
GET THE NEXT TERMNAME ENTRY
SUBRACT ONE FROM THE NUMBER OF TERMNAME ENTRIES
IS THE COUNT = ZERO
NO
YES
PUT THE ID AND TTR FOR THE NEXT LOAD IN THE WHERE-TO-GO TABLE
XCTL TO THE NEXT ENTRY IN THE WTG TABLE
UPDATE THE STATUS FIELD; MOVE TO THE NEXT FIELD IN THE DISK RECORD
ME-3 A1
CHECK
CHECK FOR THE END OF THE DISK RECORD
ARE DISK QUEUES USED
NO
YES
UPDATE THE SEQUENCE NUMBERS
ME-3 A1
CHECK
CHECK FOR THE END OF THE DISK RECORD
ARE OPTION FIELDS USED
NO
YES
IS THE OPTION AN ADDRESS TYPE
YES
NO
UPDATE THE OPTION FIELD
ME-3 A1
CHECK
CHECK FOR THE END OF THE DISK RECORD
GET THE NEXT OPTION FIELD
IS THIS THE LAST FIELD
NO
YES
A4
A4
IS THE CHECKPOINT SWITCH ON IN THE QCB
YES
E1
NO
TURN ON THE 'CHKPT' SWITCH; GET RELATIVE LINE NUMBER
ARE THERE DISK QUEUES
NO
YES
UPDATE MASTER QCB AND SEGMENT FIELDS; MOVE TO NEXT FIELD IN DISK RECORD
ME-3 A1
CHECK
CHECK FOR THE END OF THE DISK RECORD
UPDATE PRIORITY LEVEL QCB; ADD LENGTH; MOVE TO THE NEXT DISK RECORD FIELD
ME-3 A1
CHECK
CHECK FOR THE END OF THE DISK RECORD
IS IT A ZERO PRIORITY LEVEL QCB
YES
NO
GET THE NEXT PRIORITY LEVEL QCB
IS THIS A PROCESS ENTRY
YES
NO
UPDATE THE LCB FIELDS; MOVE TO THE NEXT FIELD
UPDATE THE DCB FIELDS; MOVE TO THE NEXT FIELD
ME-3 A1
CHECK
CHECK FOR THE END OF THE DISK RECORD
ARE INVITATION LISTS TO BE CKPOIN-TED
NO
YES
IS THE LIST PROCESSED BY CKPOINT
YES
NO
TURN ON THE 'CHKPT' SWITCH; GET THE LENGTH OF THE LIST; UPDATE THE LIST
ME-3 A1
CHECK
CHECK FOR THE END OF THE RECORD
E1
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart ME-3 CHECKPOINT/RESTART FROM ENVIRONMENT RECORD ROUTINE

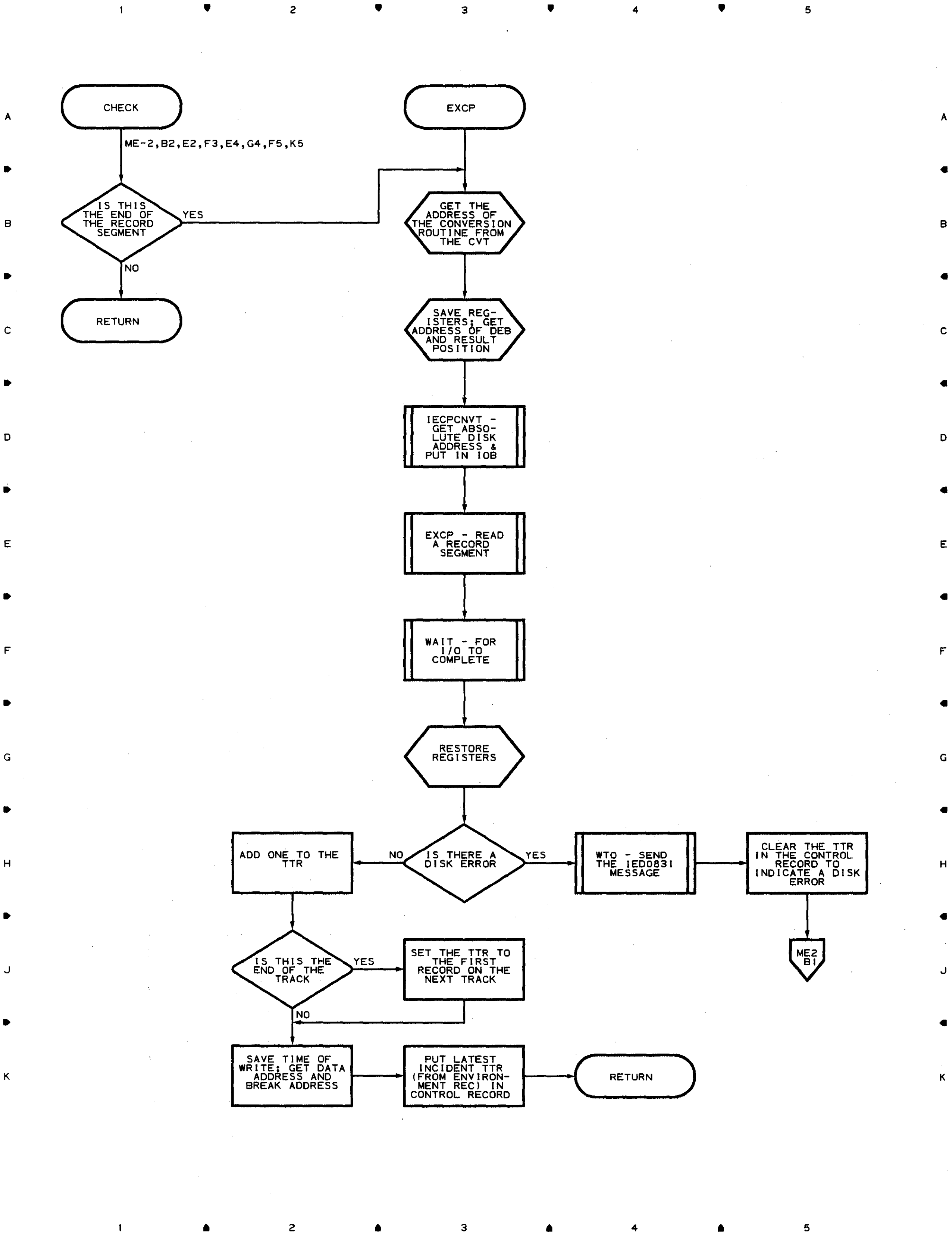

CHECK
ME-2,B2,E2,F3,E4,G4,F5,K5
IS THIS THE END OF THE RECORD SEGMENT
YES
NO
RETURN
EXCP
GET THE ADDRESS OF THE CONVERSION ROUTINE FROM THE CVT
SAVE REGISTERS; GET ADDRESS OF DEB AND RESULT POSITION
IECPCNVT - GET ABSOLUTE DISK ADDRESS & PUT IN IOB
EXCP - READ A RECORD SEGMENT
WAIT - FOR I/O TO COMPLETE
RESTORE REGISTERS
IS THERE A DISK ERROR
NO
YES
ADD ONE TO THE TTR
WTO - SEND THE IED083I MESSAGE
CLEAR THE TTR IN THE CONTROL RECORD TO INDICATE A DISK ERROR
ME2 B1
IS THIS THE END OF THE TRACK
YES
NO
SET THE TTR TO THE FIRST RECORD ON THE NEXT TRACK
SAVE TIME OF WRITE; GET DATA ADDRESS AND BREAK ADDRESS
PUT LATEST INCIDENT TTR (FROM ENVIRONMENT REC) IN CONTROL RECORD
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart MG-1 CHECKPOINT/RESTART FROM INCIDENT AND CKREQ RECORDS ROUTINE

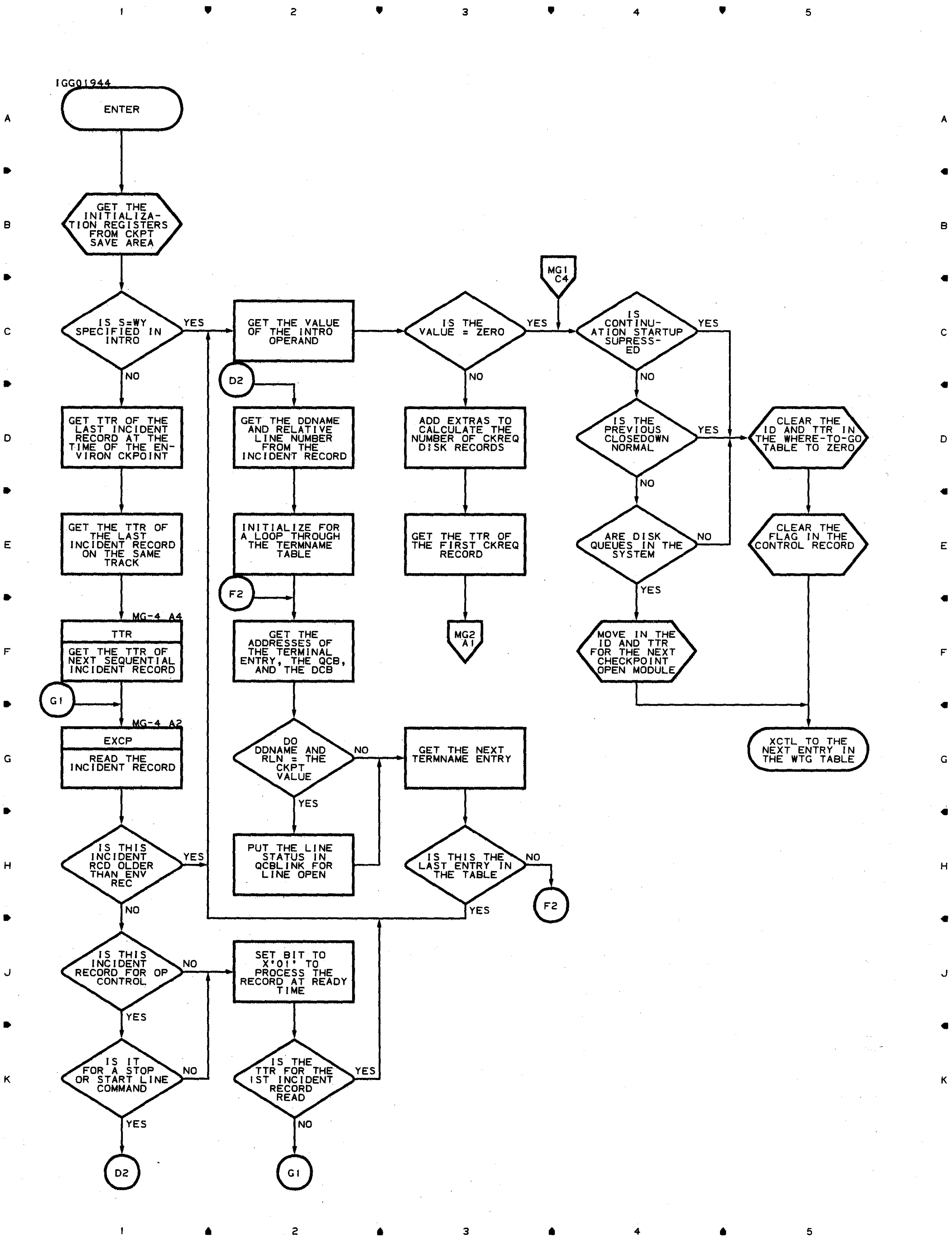

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG01944
ENTER
GET THE INITIALIZATION REGISTERS FROM CKPT SAVE AREA
IS S=WY SPECIFIED IN INTRO
YES
NO
GET TTR OF THE LAST INCIDENT RECORD AT THE TIME OF THE ENVIRON CKPOINT
GET THE TTR OF THE LAST INCIDENT RECORD ON THE SAME TRACK
MG-4 A4
TTR
GET THE TTR OF NEXT SEQUENTIAL INCIDENT RECORD
G1
MG-4 A2
EXCP
READ THE INCIDENT RECORD
IS THIS INCIDENT RCD OLDER THAN ENV REC
YES
NO
IS THIS INCIDENT RECORD FOR OP CONTROL
NO
YES
IS IT FOR A STOP OR START LINE COMMAND
NO
YES
D2
GET THE VALUE OF THE INTRO OPERAND
D2
GET THE DDNAME AND RELATIVE LINE NUMBER FROM THE INCIDENT RECORD
INITIALIZE FOR A LOOP THROUGH THE TERMNAME TABLE
F2
GET THE ADDRESSES OF THE TERMINAL ENTRY, THE QCB, AND THE DCB
DO DDNAME AND RLN = THE CKPT VALUE
NO
YES
PUT THE LINE STATUS IN QCBLINK FOR LINE OPEN
SET BIT TO X'01' TO PROCESS THE RECORD AT READY TIME
IS THE TTR FOR THE 1ST INCIDENT RECORD READ
YES
NO
G1
IS THE VALUE = ZERO
YES
NO
ADD EXTRAS TO CALCULATE THE NUMBER OF CKREQ DISK RECORDS
GET THE TTR OF THE FIRST CKREQ RECORD
MG2 A1
GET THE NEXT TERMNAME ENTRY
IS THIS THE LAST ENTRY IN THE TABLE
NO
YES
F2
MG1 C4
IS CONTINUATION STARTUP SUPRESSED
YES
NO
IS THE PREVIOUS CLOSEDOWN NORMAL
YES
NO
ARE DISK QUEUES IN THE SYSTEM
NO
YES
MOVE IN THE ID AND TTR FOR THE NEXT CHECKPOINT OPEN MODULE
CLEAR THE ID AND TTR IN THE WHERE-TO-GO TABLE TO ZERO
CLEAR THE FLAG IN THE CONTROL RECORD
XCTL TO THE NEXT ENTRY IN THE WTG TABLE

Chart MG-2 CHECKPOINT/RESTART FROM INCIDENT AND CKREQ RECORDS ROUTINE

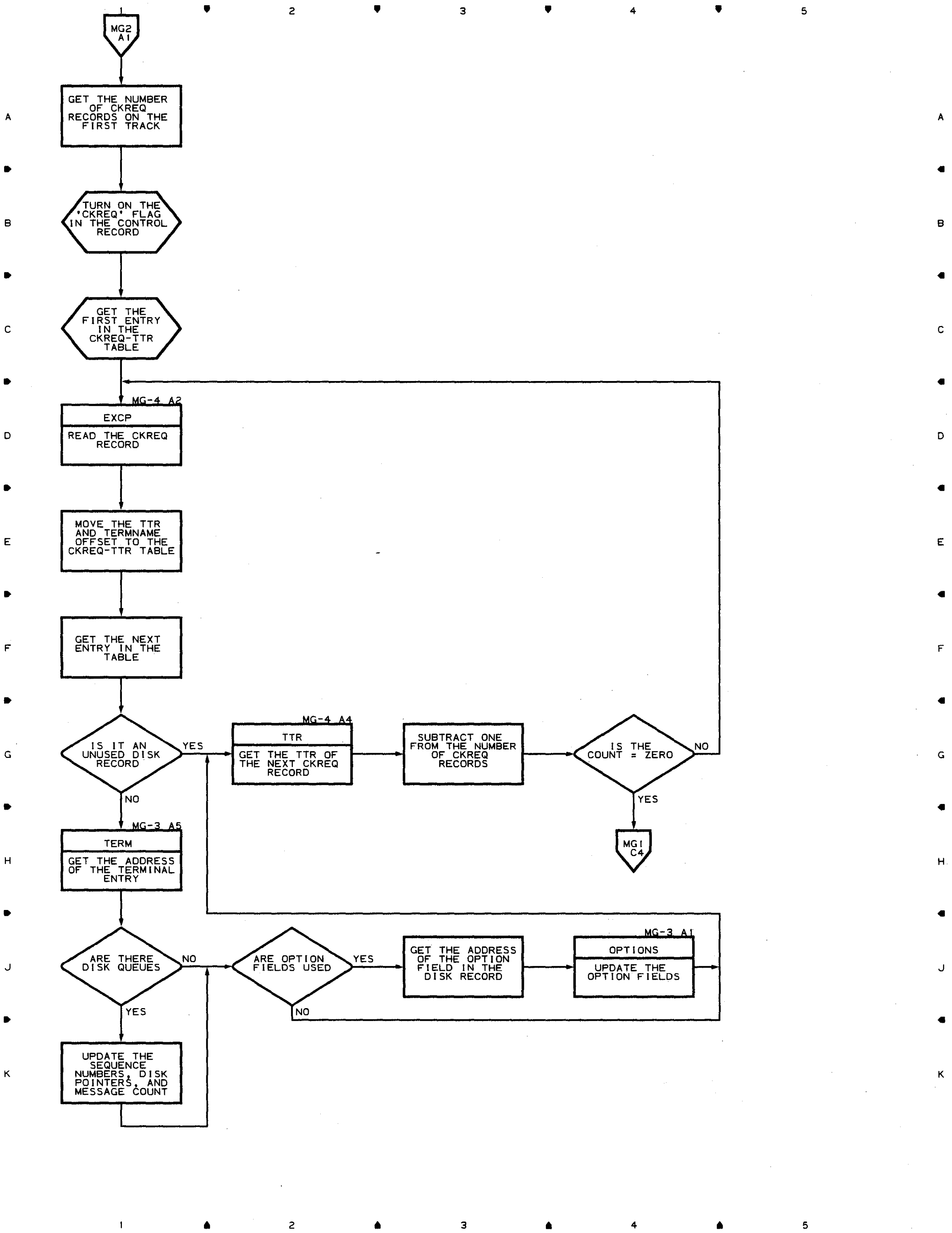

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
MG2 A1
GET THE NUMBER OF CKREQ RECORDS ON THE FIRST TRACK
TURN ON THE 'CKREQ' FLAG IN THE CONTROL RECORD
GET THE FIRST ENTRY IN THE CKREQ-TTR TABLE
MG-4 A2
EXCP
READ THE CKREQ RECORD
MOVE THE TTR AND TERMNAME OFFSET TO THE CKREQ-TTR TABLE
GET THE NEXT ENTRY IN THE TABLE
IS IT AN UNUSED DISK RECORD
YES
MG-4 A4
TTR
GET THE TTR OF THE NEXT CKREQ RECORD
SUBTRACT ONE FROM THE NUMBER OF CKREQ RECORDS
IS THE COUNT = ZERO
NO
YES
MG1 C4
NO
MG-3 A5
TERM
GET THE ADDRESS OF THE TERMINAL ENTRY
ARE THERE DISK QUEUES
NO
ARE OPTION FIELDS USED
YES
GET THE ADDRESS OF THE OPTION FIELD IN THE DISK RECORD
MG-3 A1
OPTIONS
UPDATE THE OPTION FIELDS
NO
YES
UPDATE THE SEQUENCE NUMBERS, DISK POINTERS, AND MESSAGE COUNT

Chart MG-3 CHECKPOINT/RESTART FROM INCIDENT AND CKREQ RECORDS ROUTINE

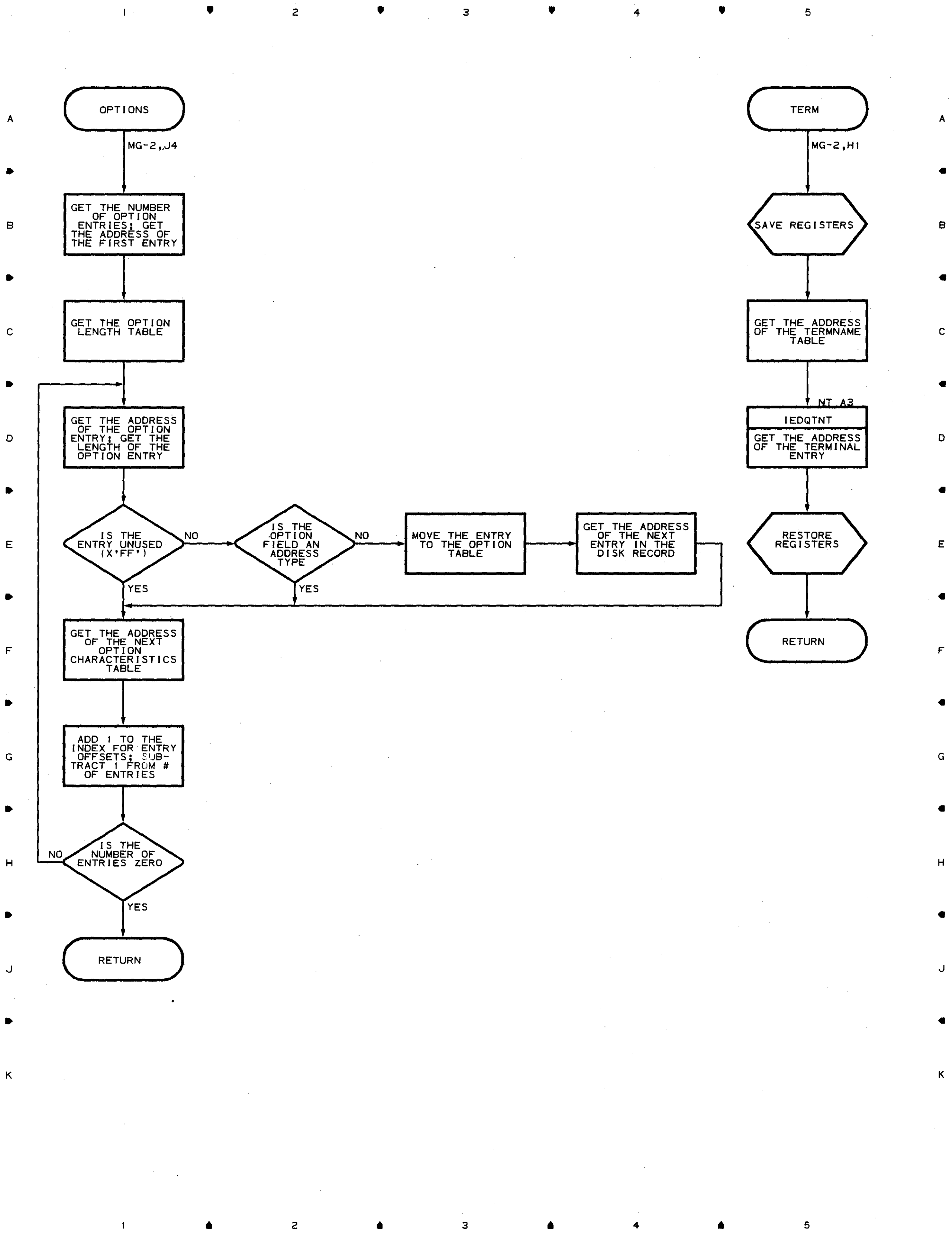

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
OPTIONS
MG-2,J4
GET THE NUMBER OF OPTION ENTRIES; GET THE ADDRESS OF THE FIRST ENTRY
GET THE OPTION LENGTH TABLE
GET THE ADDRESS OF THE OPTION ENTRY; GET THE LENGTH OF THE OPTION ENTRY
IS THE ENTRY UNUSED (X'FF')
NO
YES
IS THE OPTION FIELD AN ADDRESS TYPE
NO
YES
MOVE THE ENTRY TO THE OPTION TABLE
GET THE ADDRESS OF THE NEXT ENTRY IN THE DISK RECORD
GET THE ADDRESS OF THE NEXT OPTION CHARACTERISTICS TABLE
ADD 1 TO THE INDEX FOR ENTRY OFFSETS; SUB-TRACT 1 FROM # OF ENTRIES
IS THE NUMBER OF ENTRIES ZERO
NO
YES
RETURN
TERM
MG-2,H1
SAVE REGISTERS
GET THE ADDRESS OF THE TERMNAME TABLE
NT A3
IEDQTNT
GET THE ADDRESS OF THE TERMINAL ENTRY
RESTORE REGISTERS
RETURN

Chart MG-4 CHECKPOINT/RESTART FROM INCIDENT AND CKREQ RECORDS ROUTINE

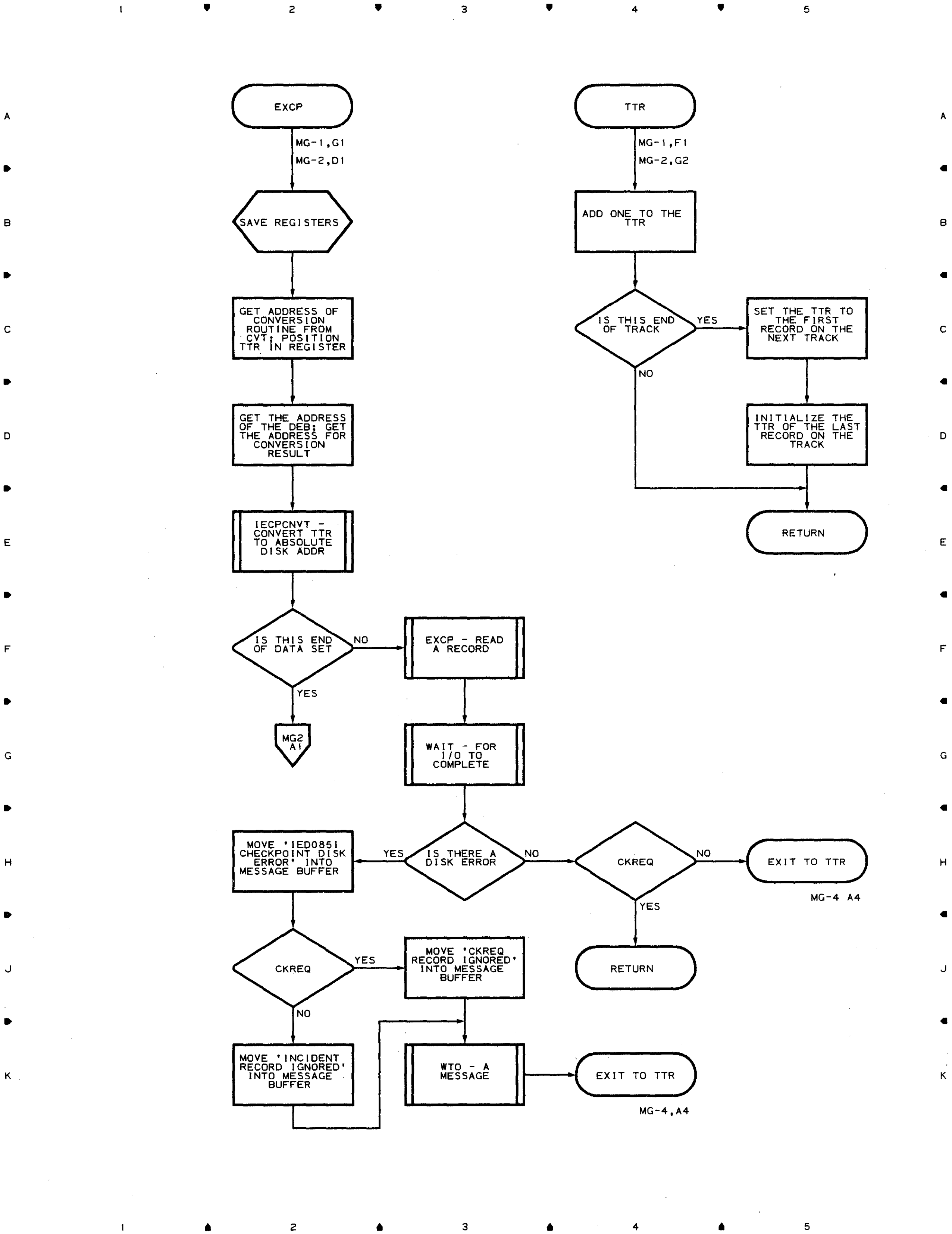
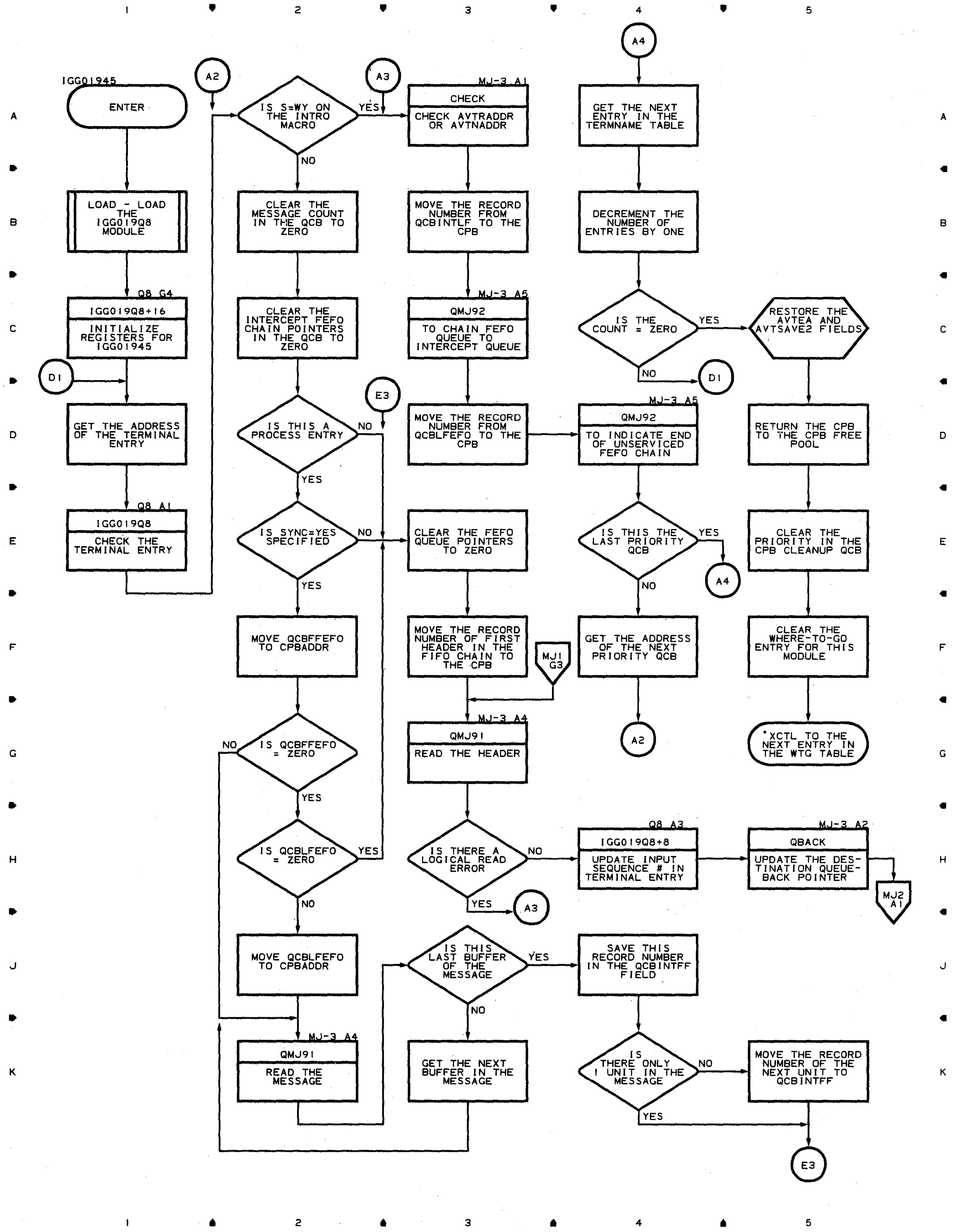
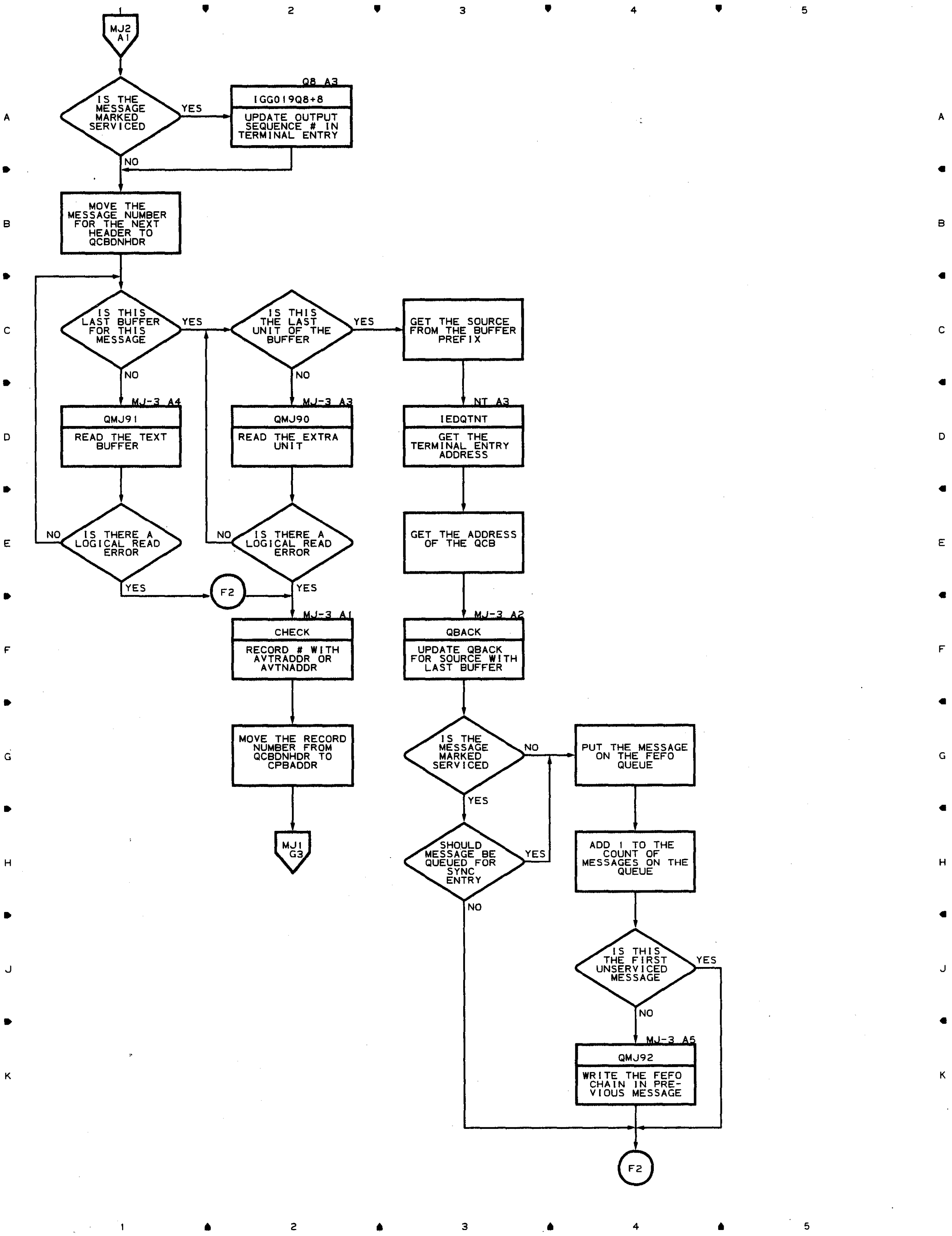

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
EXCP
MG-1,G1
MG-2,D1
SAVE REGISTERS
GET ADDRESS OF CONVERSION ROUTINE FROM CVT; POSITION TTR IN REGISTER
GET THE ADDRESS OF THE DEB; GET THE ADDRESS FOR CONVERSION RESULT
IECPCNVT - CONVERT TTR TO ABSOLUTE DISK ADDR
IS THIS END OF DATA SET
NO
YES
MG2 A1
EXCP - READ A RECORD
WAIT - FOR I/O TO COMPLETE
IS THERE A DISK ERROR
YES
NO
MOVE 'IED0851 CHECKPOINT DISK ERROR' INTO MESSAGE BUFFER
CKREQ
YES
NO
MOVE 'CKREQ RECORD IGNORED' INTO MESSAGE BUFFER
MOVE 'INCIDENT RECORD IGNORED' INTO MESSAGE BUFFER
WTO - A MESSAGE
EXIT TO TTR
MG-4,A4
CKREQ
NO
YES
EXIT TO TTR
MG-4 A4
RETURN
TTR
MG-1,F1
MG-2,G2
ADD ONE TO THE TTR
IS THIS END OF TRACK
YES
NO
SET THE TTR TO THE FIRST RECORD ON THE NEXT TRACK
INITIALIZE THE TTR OF THE LAST RECORD ON THE TRACK
RETURN

Chart MJ-1 CHECKPOINT CONTINUATION RESTART ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG01945
ENTER
LOAD - LOAD THE IGG019Q8 MODULE
Q8 G4
IGG019Q8+16
INITIALIZE REGISTERS FOR IGG01945
D1
GET THE ADDRESS OF THE TERMINAL ENTRY
Q8 A1
IGG019Q8
CHECK THE TERMINAL ENTRY
A2
IS S=WY ON THE INTRO MACRO
YES
NO
CLEAR THE MESSAGE COUNT IN THE QCB TO ZERO
CLEAR THE INTERCEPT FEFO CHAIN POINTERS IN THE QCB TO ZERO
IS THIS A PROCESS ENTRY
NO
YES
IS SYNC=YES SPECIFIED
NO
YES
MOVE QCBFFEFO TO CPBADDR
IS QCBFFEFO = ZERO
NO
YES
IS QCBLFEFO = ZERO
YES
NO
MOVE QCBLFEFO TO CPBADDR
MJ-3 A4
QMJ91
READ THE MESSAGE
A3
MJ-3 A1
CHECK
CHECK AVTRADDR OR AVTNADDR
MOVE THE RECORD NUMBER FROM QCBINTLF TO THE CPB
MJ-3 A5
QMJ92
TO CHAIN FEFO QUEUE TO INTERCEPT QUEUE
E3
MOVE THE RECORD NUMBER FROM QCBLFEFO TO THE CPB
CLEAR THE FEFO QUEUE POINTERS TO ZERO
MOVE THE RECORD NUMBER OF FIRST HEADER IN THE FIFO CHAIN TO THE CPB
MJ1 G3
MJ-3 A4
QMJ91
READ THE HEADER
IS THERE A LOGICAL READ ERROR
NO
YES
A3
IS THIS LAST BUFFER OF THE MESSAGE
YES
NO
GET THE NEXT BUFFER IN THE MESSAGE
A4
GET THE NEXT ENTRY IN THE TERMNAME TABLE
DECREMENT THE NUMBER OF ENTRIES BY ONE
IS THE COUNT = ZERO
YES
NO
D1
MJ-3 A5
QMJ92
TO INDICATE END OF UNSERVICED FEFO CHAIN
IS THIS THE LAST PRIORITY QCB
YES
NO
A4
GET THE ADDRESS OF THE NEXT PRIORITY QCB
A2
Q8 A3
IGG019Q8+8
UPDATE INPUT SEQUENCE # IN TERMINAL ENTRY
SAVE THIS RECORD NUMBER IN THE QCBINTFF FIELD
IS THERE ONLY 1 UNIT IN THE MESSAGE
NO
YES
RESTORE THE AVTEA AND AVTSAVE2 FIELDS
RETURN THE CPB TO THE CPB FREE POOL
CLEAR THE PRIORITY IN THE CPB CLEANUP QCB
CLEAR THE WHERE-TO-GO ENTRY FOR THIS MODULE
XCTL TO THE NEXT ENTRY IN THE WTG TABLE
MJ-3 A2
QBACK
UPDATE THE DESTINATION QUEUE-BACK POINTER
MJ2 A1
MOVE THE RECORD NUMBER OF THE NEXT UNIT TO QCBINTFF
E3

Chart MJ-2 CHECKPOINT CONTINUATION RESTART ROUTINE



1
2
3
4
5
MJ2 A1
IS THE MESSAGE MARKED SERVICED
YES
NO
Q8 A3
IGG019Q8+8
UPDATE OUTPUT SEQUENCE # IN TERMINAL ENTRY
MOVE THE MESSAGE NUMBER FOR THE NEXT HEADER TO QCBDNHDR
IS THIS LAST BUFFER FOR THIS MESSAGE
YES
NO
IS THIS THE LAST UNIT OF THE BUFFER
YES
NO
GET THE SOURCE FROM THE BUFFER PREFIX
MJ-3 A4
QMJ91
READ THE TEXT BUFFER
MJ-3 A3
QMJ90
READ THE EXTRA UNIT
NT A3
IEDQTNT
GET THE TERMINAL ENTRY ADDRESS
NO
IS THERE A LOGICAL READ ERROR
YES
NO
IS THERE A LOGICAL READ ERROR
YES
F2
GET THE ADDRESS OF THE QCB
MJ-3 A1
CHECK
RECORD # WITH AVTRADDR OR AVTNADDR
MJ-3 A2
QBACK
UPDATE QBACK FOR SOURCE WITH LAST BUFFER
MOVE THE RECORD NUMBER FROM QCBDNHDR TO CPBADDR
IS THE MESSAGE MARKED SERVICED
NO
YES
PUT THE MESSAGE ON THE FEFO QUEUE
MJ1 G3
SHOULD MESSAGE BE QUEUED FOR SYNC ENTRY
YES
NO
ADD 1 TO THE COUNT OF MESSAGES ON THE QUEUE
IS THIS THE FIRST UNSERVICED MESSAGE
YES
NO
MJ-3 A5
QMJ92
WRITE THE FEFO CHAIN IN PRE-VIOUS MESSAGE
F2
A
B
C
D
E
F
G
H
J
K

Chart MJ-3 CHECKPOINT CONTINUATION RESTART ROUTINE

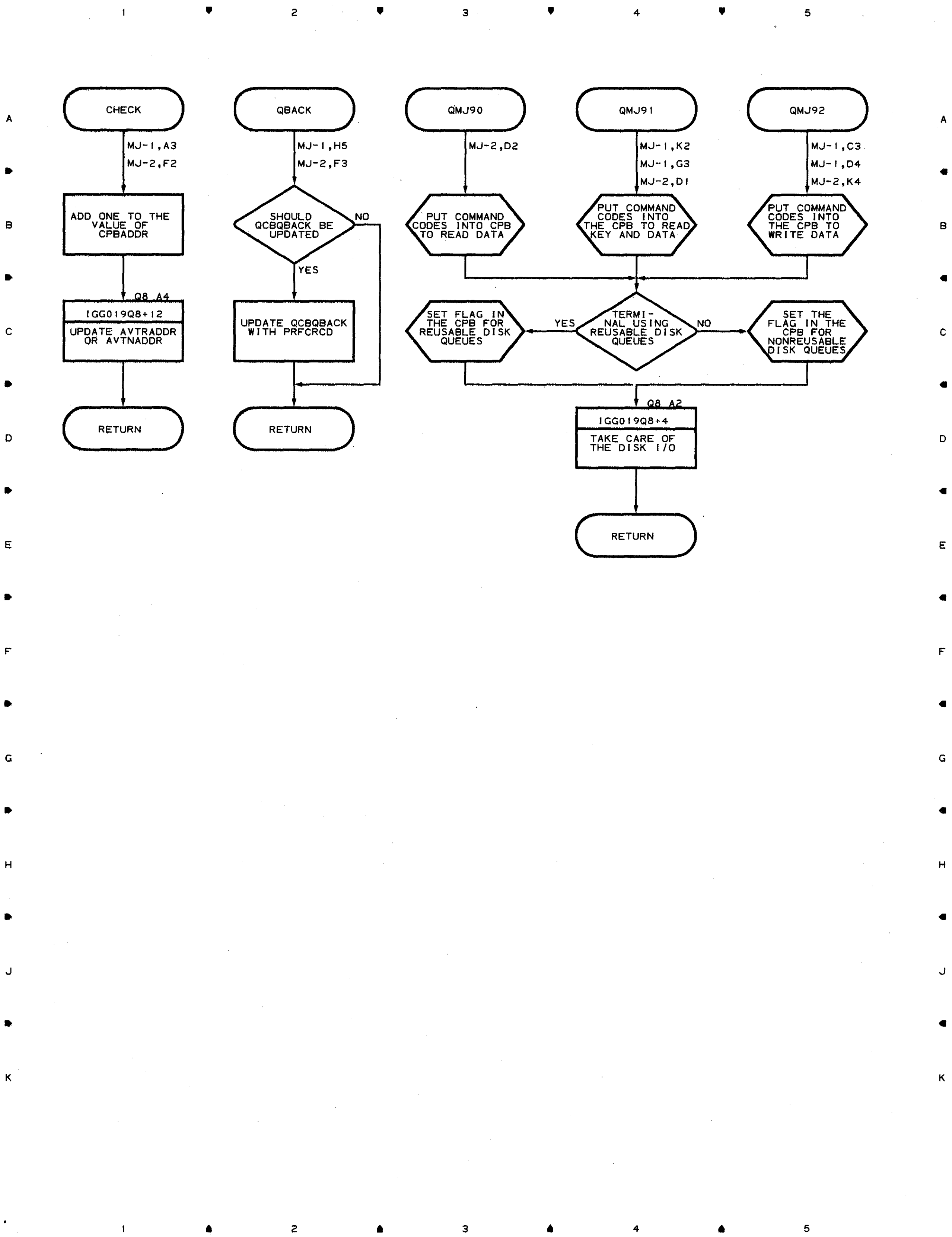

CHECK
MJ-1,A3
MJ-2,F2
ADD ONE TO THE VALUE OF CPBADDR
Q8 A4
IGG019Q8+12
UPDATE AVTRADDR OR AVTNADDR
RETURN
QBACK
MJ-1,H5
MJ-2,F3
SHOULD QCBQBACK BE UPDATED
NO
YES
UPDATE QCBQBACK WITH PRFCRCD
RETURN
QMJ90
MJ-2,D2
PUT COMMAND CODES INTO CPB TO READ DATA
QMJ91
MJ-1,K2
MJ-1,G3
MJ-2,D1
PUT COMMAND CODES INTO THE CPB TO READ KEY AND DATA
QMJ92
MJ-1,C3
MJ-1,D4
MJ-2,K4
PUT COMMAND CODES INTO THE CPB TO WRITE DATA
SET FLAG IN THE CPB FOR REUSABLE DISK QUEUES
YES
TERMINAL USING REUSABLE DISK QUEUES
NO
SET THE FLAG IN THE CPB FOR NONREUSABLE DISK QUEUES
Q8 A2
IGG019Q8+4
TAKE CARE OF THE DISK I/O
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart MM-1 CHECKPOINT DISK ALLOCATION ROUTINE

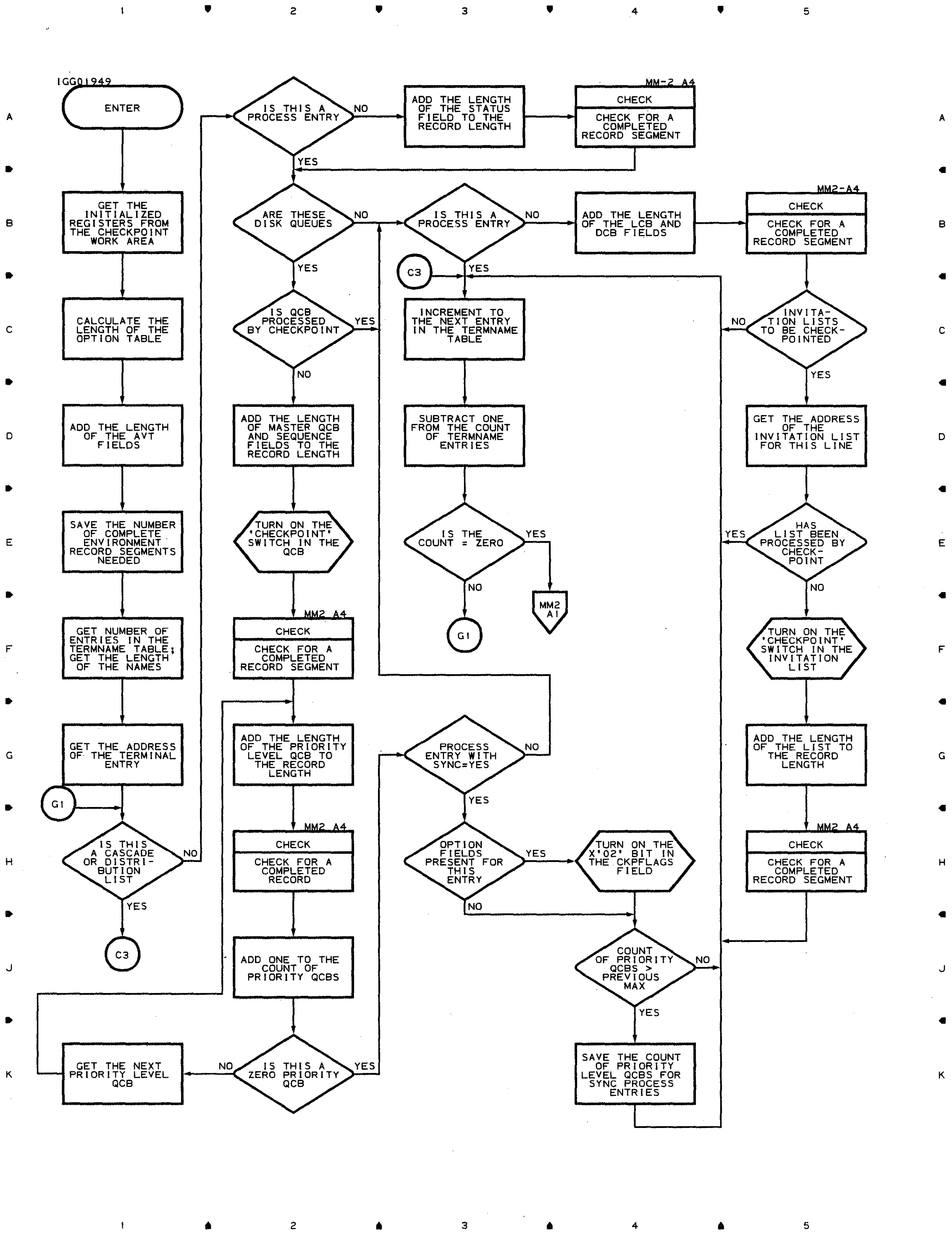
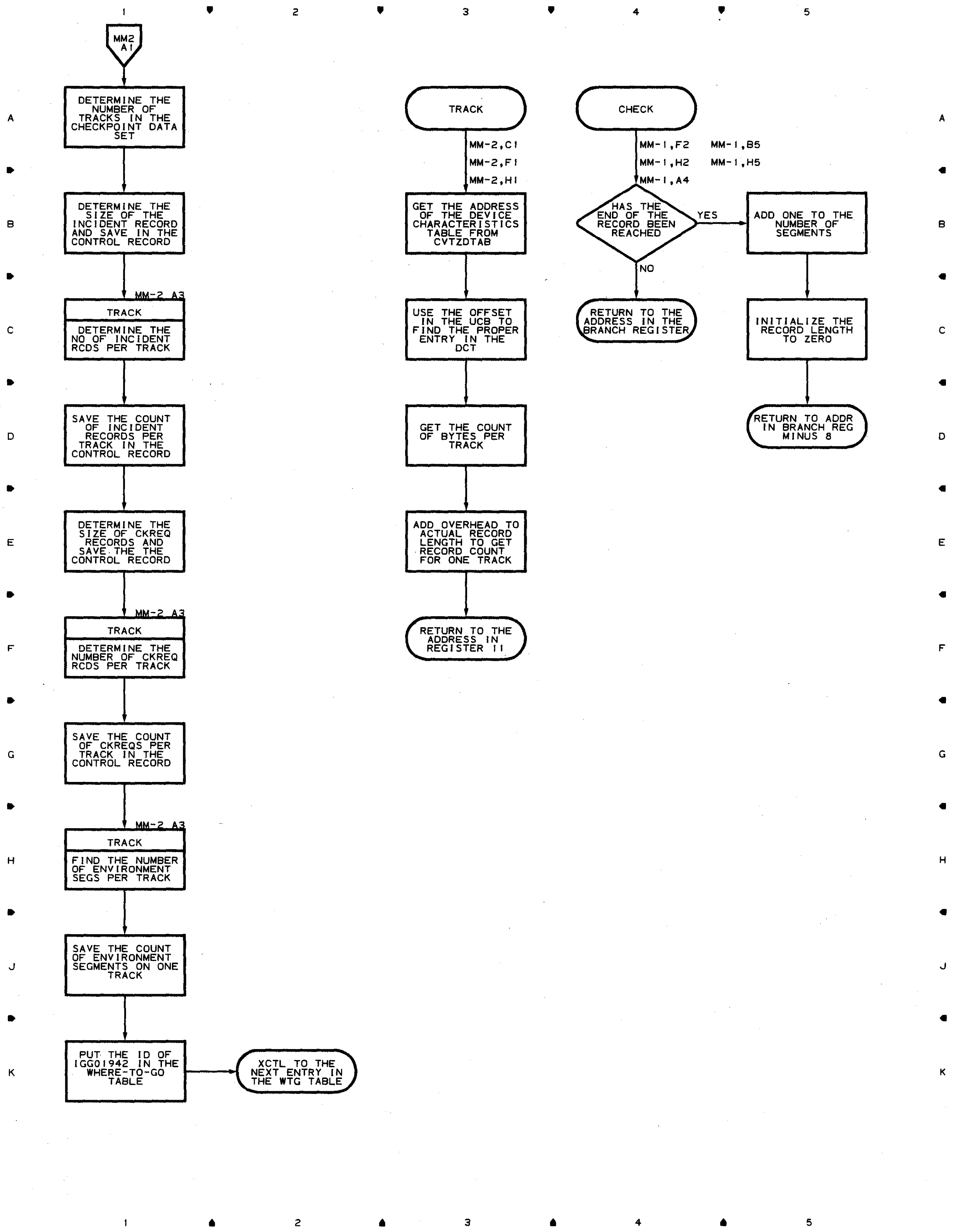
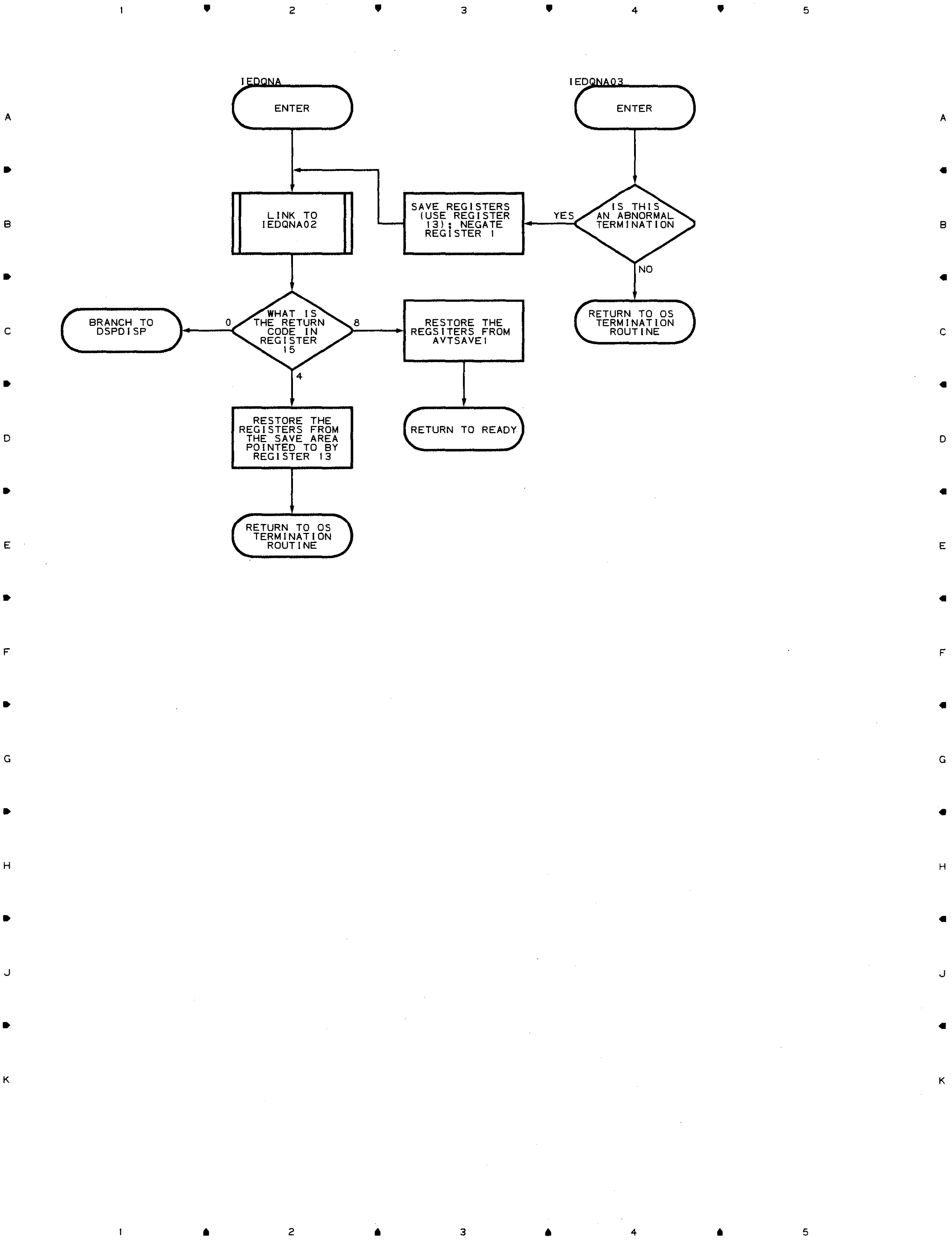

IGG01949
ENTER
GET THE INITIALIZED REGISTERS FROM THE CHECKPOINT WORK AREA
CALCULATE THE LENGTH OF THE OPTION TABLE
ADD THE LENGTH OF THE AVT FIELDS
SAVE THE NUMBER OF COMPLETE ENVIRONMENT RECORD SEGMENTS NEEDED
GET NUMBER OF ENTRIES IN THE TERMNAME TABLE; GET THE LENGTH OF THE NAMES
GET THE ADDRESS OF THE TERMINAL ENTRY
G1
IS THIS A CASCADE OR DISTRIBUTION LIST
NO
YES
C3
GET THE NEXT PRIORITY LEVEL QCB
IS THIS A PROCESS ENTRY
NO
YES
ARE THESE DISK QUEUES
NO
YES
IS QCB PROCESSED BY CHECKPOINT
YES
NO
ADD THE LENGTH OF MASTER QCB AND SEQUENCE FIELDS TO THE RECORD LENGTH
TURN ON THE 'CHECKPOINT' SWITCH IN THE QCB
MM2 A4
CHECK
CHECK FOR A COMPLETED RECORD SEGMENT
ADD THE LENGTH OF THE PRIORITY LEVEL QCB TO THE RECORD LENGTH
MM2 A4
CHECK
CHECK FOR A COMPLETED RECORD
ADD ONE TO THE COUNT OF PRIORITY QCBS
IS THIS A ZERO PRIORITY QCB
NO
YES
ADD THE LENGTH OF THE STATUS FIELD TO THE RECORD LENGTH
IS THIS A PROCESS ENTRY
NO
YES
C3
INCREMENT TO THE NEXT ENTRY IN THE TERMNAME TABLE
SUBTRACT ONE FROM THE COUNT OF TERMNAME ENTRIES
IS THE COUNT = ZERO
YES
NO
G1
MM2 A1
PROCESS ENTRY WITH SYNC=YES
NO
YES
OPTION FIELDS PRESENT FOR THIS ENTRY
YES
NO
MM-2 A4
CHECK
CHECK FOR A COMPLETED RECORD SEGMENT
ADD THE LENGTH OF THE LCB AND DCB FIELDS
TURN ON THE X'02' BIT IN THE CKPFLAGS FIELD
COUNT OF PRIORITY QCBS > PREVIOUS MAX
NO
YES
SAVE THE COUNT OF PRIORITY LEVEL QCBS FOR SYNC PROCESS ENTRIES
MM2-A4
CHECK
CHECK FOR A COMPLETED RECORD SEGMENT
INVITATION LISTS TO BE CHECKPOINTED
NO
YES
GET THE ADDRESS OF THE INVITATION LIST FOR THIS LINE
HAS LIST BEEN PROCESSED BY CHECKPOINT
YES
NO
TURN ON THE 'CHECKPOINT' SWITCH IN THE INVITATION LIST
ADD THE LENGTH OF THE LIST TO THE RECORD LENGTH
MM2 A4
CHECK
CHECK FOR A COMPLETED RECORD SEGMENT
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart MM-2 CHECKPOINT DISK ALLOCATION ROUTINE



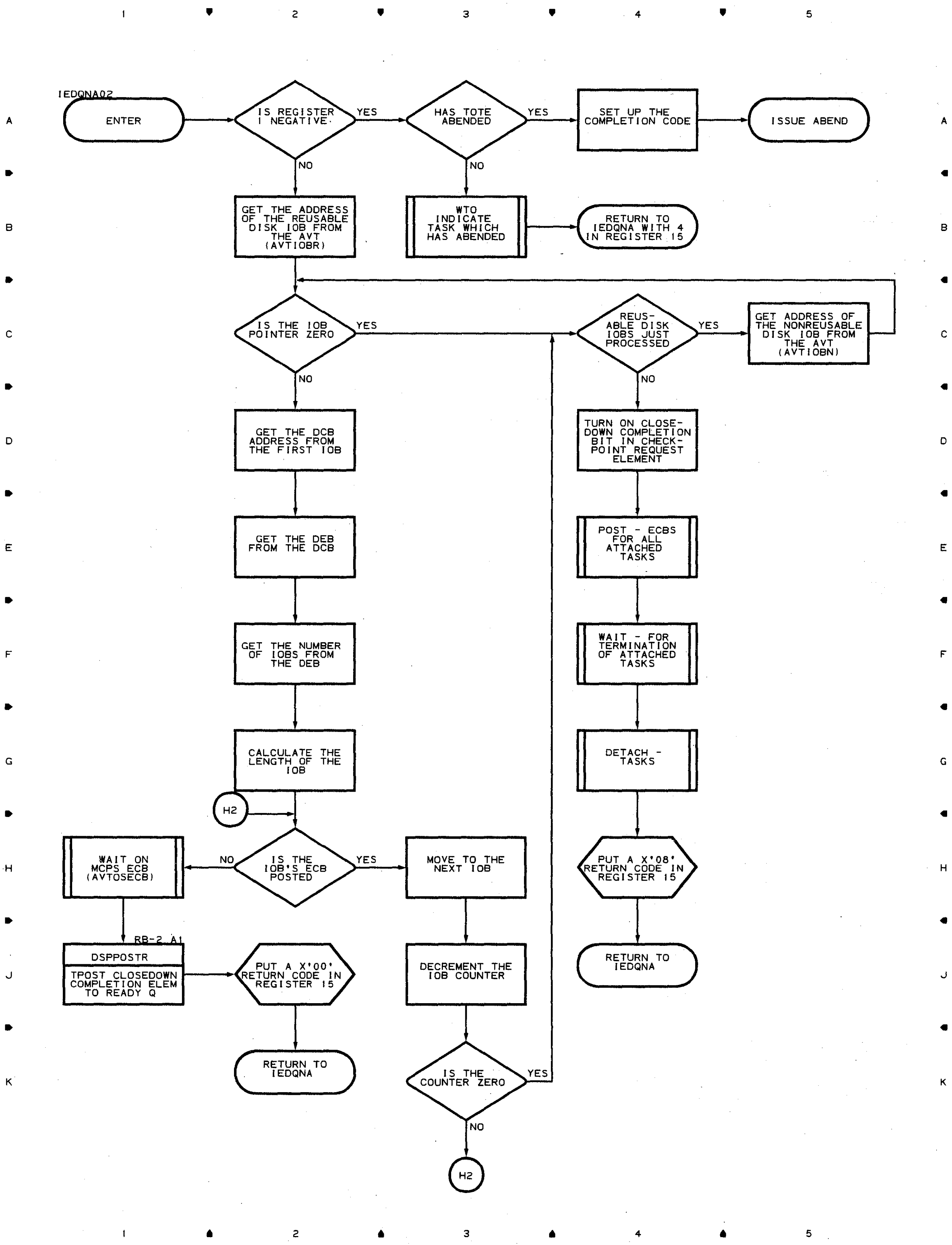
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
MM2 A1
DETERMINE THE NUMBER OF TRACKS IN THE CHECKPOINT DATA SET
DETERMINE THE SIZE OF THE INCIDENT RECORD AND SAVE IN THE CONTROL RECORD
MM-2 A3
TRACK
DETERMINE THE NO OF INCIDENT RCDS PER TRACK
SAVE THE COUNT OF INCIDENT RECORDS PER TRACK IN THE CONTROL RECORD
DETERMINE THE SIZE OF CKREQ RECORDS AND SAVE THE THE CONTROL RECORD
MM-2 A3
TRACK
DETERMINE THE NUMBER OF CKREQ RCDS PER TRACK
SAVE THE COUNT OF CKREQS PER TRACK IN THE CONTROL RECORD
MM-2 A3
TRACK
FIND THE NUMBER OF ENVIRONMENT SEGS PER TRACK
SAVE THE COUNT OF ENVIRONMENT SEGMENTS ON ONE TRACK
PUT THE ID OF IGG01942 IN THE WHERE-TO-GO TABLE
XCTL TO THE NEXT ENTRY IN THE WTG TABLE
TRACK
MM-2,C1
MM-2,F1
MM-2,H1
GET THE ADDRESS OF THE DEVICE CHARACTERISTICS TABLE FROM CVTZDTAB
USE THE OFFSET IN THE UCB TO FIND THE PROPER ENTRY IN THE DCT
GET THE COUNT OF BYTES PER TRACK
ADD OVERHEAD TO ACTUAL RECORD LENGTH TO GET RECORD COUNT FOR ONE TRACK
RETURN TO THE ADDRESS IN REGISTER 11
CHECK
MM-1,F2
MM-1,H2
MM-1,A4
MM-1,B5
MM-1,H5
HAS THE END OF THE RECORD BEEN REACHED
YES
NO
RETURN TO THE ADDRESS IN THE BRANCH REGISTER
ADD ONE TO THE NUMBER OF SEGMENTS
INITIALIZE THE RECORD LENGTH TO ZERO
RETURN TO ADDR IN BRANCH REG MINUS 8

Chart NA RESIDENT CLOSEDOWN COMPLETION ROUTINE



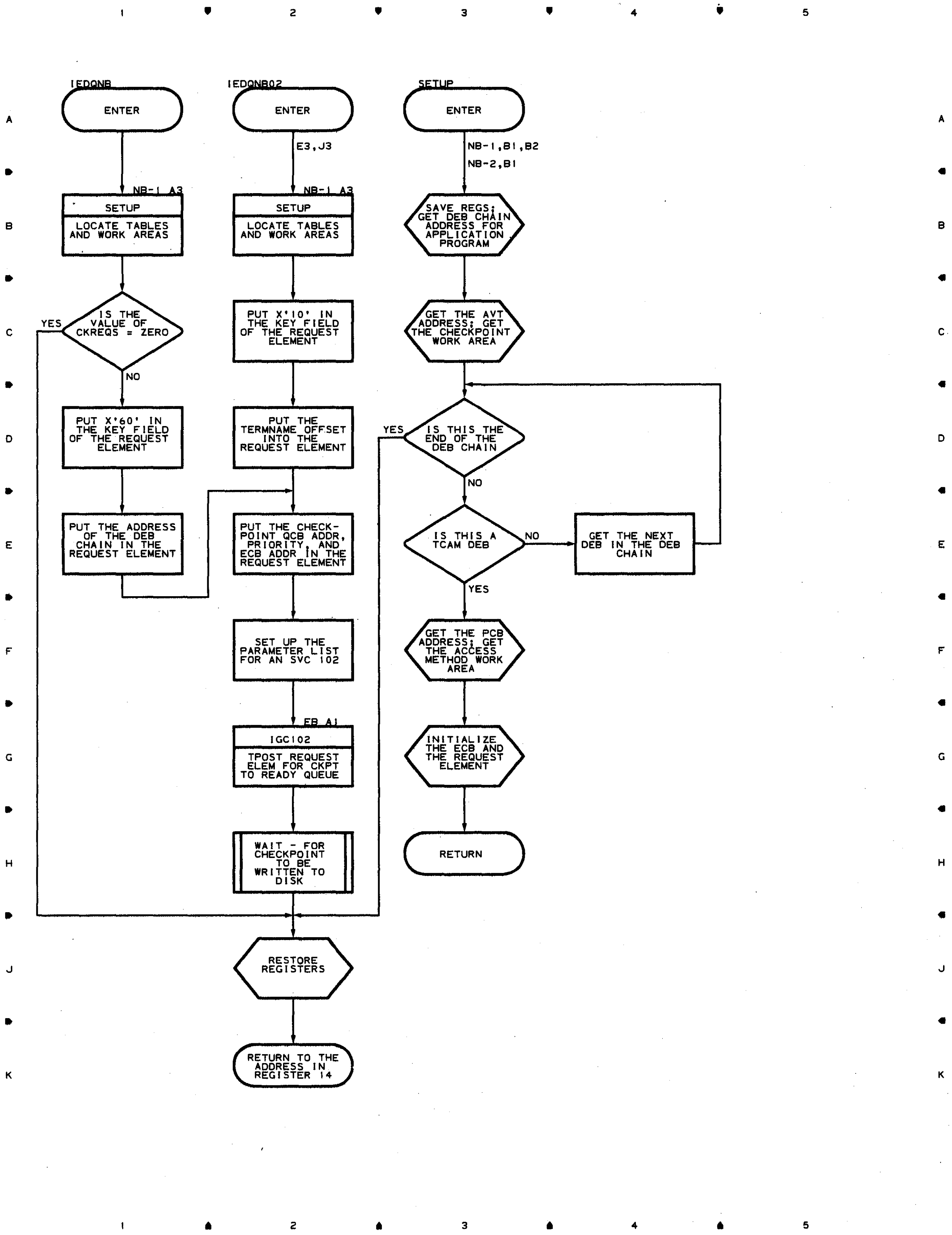
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQNA
ENTER
IEDQNA03
ENTER
LINK TO IEDQNA02
SAVE REGISTERS (USE REGISTER 13); NEGATE REGISTER 1
YES
IS THIS AN ABNORMAL TERMINATION
NO
BRANCH TO DSPDISP
0
WHAT IS THE RETURN CODE IN REGISTER 15
8
RESTORE THE REGSITERS FROM AVTSAVE1
RETURN TO OS TERMINATION ROUTINE
4
RESTORE THE REGISTERS FROM THE SAVE AREA POINTED TO BY REGISTER 13
RETURN TO READY
RETURN TO OS TERMINATION ROUTINE

Chart NA2 NONRESIDENT CLOSEDOWN COMPLETION ROUTINE



1
2
3
4
5
IEDQNA02
ENTER
IS REGISTER 1 NEGATIVE
YES
NO
HAS TOTE ABENDED
YES
NO
SET UP THE COMPLETION CODE
ISSUE ABEND
GET THE ADDRESS OF THE REUSABLE DISK IOB FROM THE AVT (AVTIOBR)
WTO INDICATE TASK WHICH HAS ABENDED
RETURN TO IEDQNA WITH 4 IN REGISTER 15
IS THE IOB POINTER ZERO
YES
NO
REUSABLE DISK IOBS JUST PROCESSED
YES
NO
GET ADDRESS OF THE NONREUSABLE DISK IOB FROM THE AVT (AVTIOBN)
GET THE DCB ADDRESS FROM THE FIRST IOB
TURN ON CLOSEDOWN COMPLETION BIT IN CHECKPOINT REQUEST ELEMENT
GET THE DEB FROM THE DCB
POST - ECBS FOR ALL ATTACHED TASKS
GET THE NUMBER OF IOBS FROM THE DEB
WAIT - FOR TERMINATION OF ATTACHED TASKS
CALCULATE THE LENGTH OF THE IOB
DETACH - TASKS
H2
WAIT ON MCPS ECB (AVTOSECB)
NO
IS THE IOB'S ECB POSTED
YES
MOVE TO THE NEXT IOB
PUT A X'08' RETURN CODE IN REGISTER 15
RB-2 A1
DSPPOSTR
TPOST CLOSEDOWN COMPLETION ELEM TO READY Q
PUT A X'00' RETURN CODE IN REGISTER 15
DECREMENT THE IOB COUNTER
RETURN TO IEDQNA
RETURN TO IEDQNA
IS THE COUNTER ZERO
YES
NO
H2
A
B
C
D
E
F
G
H
J
K

Chart NB-1 APPLICATION PROGRAM/CHECKPOINT INTERFACE ROUTINE



IEDQNB
ENTER
NB-1 A3
SETUP
LOCATE TABLES AND WORK AREAS
IS THE VALUE OF CKREQS = ZERO
YES
NO
PUT X'60' IN THE KEY FIELD OF THE REQUEST ELEMENT
PUT THE ADDRESS OF THE DEB CHAIN IN THE REQUEST ELEMENT
IEDQNB02
ENTER
E3,J3
NB-1 A3
SETUP
LOCATE TABLES AND WORK AREAS
PUT X'10' IN THE KEY FIELD OF THE REQUEST ELEMENT
PUT THE TERMNAME OFFSET INTO THE REQUEST ELEMENT
PUT THE CHECK-POINT QCB ADDR, PRIORITY, AND ECB ADDR IN THE REQUEST ELEMENT
SET UP THE PARAMETER LIST FOR AN SVC 102
FB A1
IGC102
TPOST REQUEST ELEM FOR CKPT TO READY QUEUE
WAIT - FOR CHECKPOINT TO BE WRITTEN TO DISK
RESTORE REGISTERS
RETURN TO THE ADDRESS IN REGISTER 14
SETUP
ENTER
NB-1,B1,B2
NB-2,B1
SAVE REGS; GET DEB CHAIN ADDRESS FOR APPLICATION PROGRAM
GET THE AVT ADDRESS; GET THE CHECKPOINT WORK AREA
IS THIS THE END OF THE DEB CHAIN
YES
NO
IS THIS A TCAM DEB
NO
YES
GET THE NEXT DEB IN THE DEB CHAIN
GET THE PCB ADDRESS; GET THE ACCESS METHOD WORK AREA
INITIALIZE THE ECB AND THE REQUEST ELEMENT
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart NB-2 APPLICATION PROGRAM/CHECKPOINT INTERFACE ROUTINE

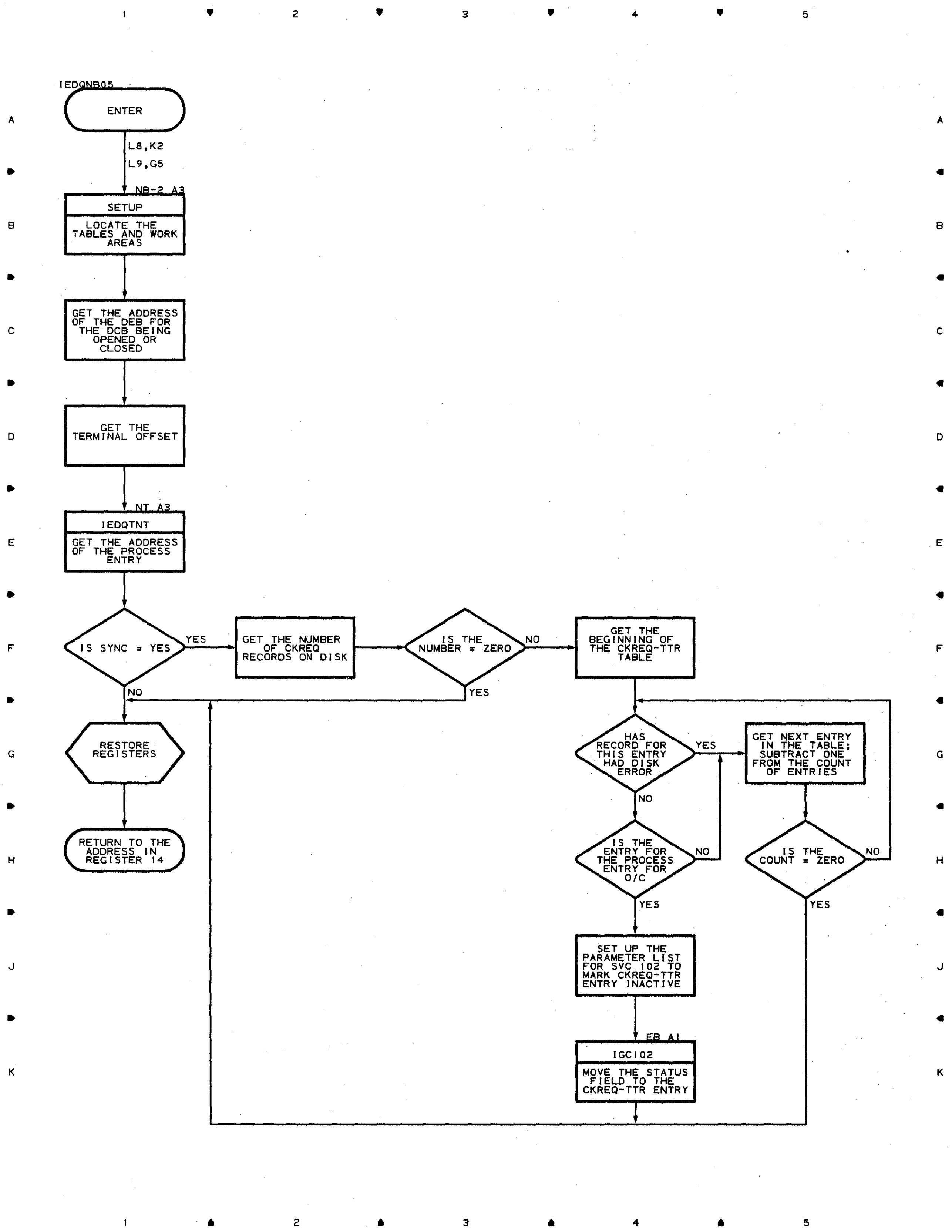

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQNB05
ENTER
L8,K2
L9,G5
NB-2 A3
SETUP
LOCATE THE TABLES AND WORK AREAS
GET THE ADDRESS OF THE DEB FOR THE DCB BEING OPENED OR CLOSED
GET THE TERMINAL OFFSET
NT A3
IEDQTNT
GET THE ADDRESS OF THE PROCESS ENTRY
IS SYNC = YES
YES
NO
GET THE NUMBER OF CKREQ RECORDS ON DISK
IS THE NUMBER = ZERO
NO
YES
GET THE BEGINNING OF THE CKREQ-TTR TABLE
RESTORE REGISTERS
RETURN TO THE ADDRESS IN REGISTER 14
HAS RECORD FOR THIS ENTRY HAD DISK ERROR
YES
NO
GET NEXT ENTRY IN THE TABLE; SUBTRACT ONE FROM THE COUNT OF ENTRIES
IS THE ENTRY FOR THE PROCESS ENTRY FOR O/C
NO
YES
IS THE COUNT = ZERO
NO
YES
SET UP THE PARAMETER LIST FOR SVC 102 TO MARK CKREQ-TTR ENTRY INACTIVE
EB A1
IGC102
MOVE THE STATUS FIELD TO THE CKREQ-TTR ENTRY

Chart ND-1 READY ROUTINE

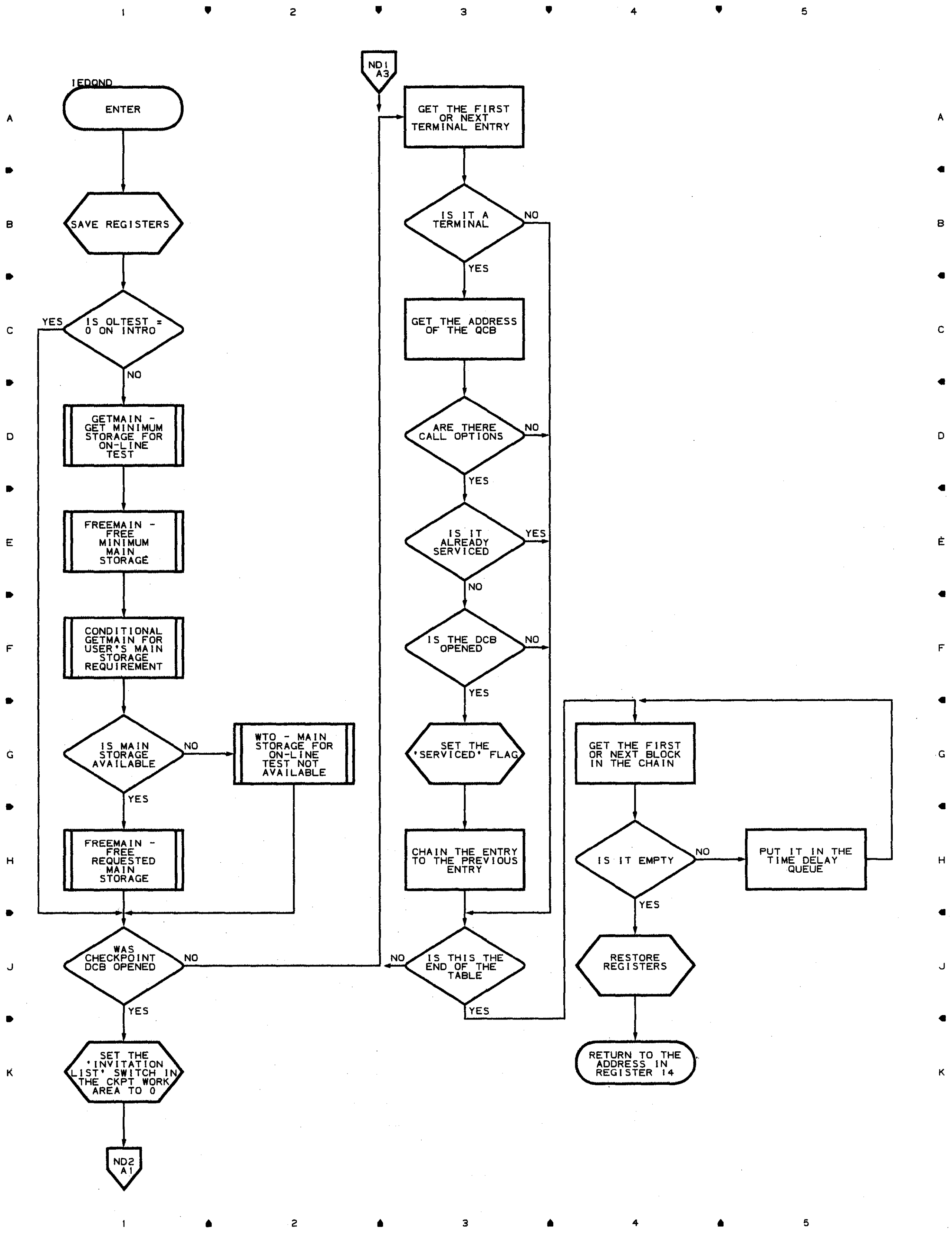

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQND
ENTER
SAVE REGISTERS
IS OLTEST = 0 ON INTRO
YES
NO
GETMAIN - GET MINIMUM STORAGE FOR ON-LINE TEST
FREEMAIN - FREE MINIMUM MAIN STORAGE
CONDITIONAL GETMAIN FOR USER'S MAIN STORAGE REQUIREMENT
IS MAIN STORAGE AVAILABLE
NO
WTO - MAIN STORAGE FOR ON-LINE TEST NOT AVAILABLE
YES
FREEMAIN - FREE REQUESTED MAIN STORAGE
WAS CHECKPOINT DCB OPENED
NO
YES
SET THE 'INVITATION LIST' SWITCH IN THE CKPT WORK AREA TO 0
ND2 A1
ND1 A3
GET THE FIRST OR NEXT TERMINAL ENTRY
IS IT A TERMINAL
NO
YES
GET THE ADDRESS OF THE QCB
ARE THERE CALL OPTIONS
NO
YES
IS IT ALREADY SERVICED
YES
NO
IS THE DCB OPENED
NO
YES
SET THE 'SERVICED' FLAG
CHAIN THE ENTRY TO THE PREVIOUS ENTRY
IS THIS THE END OF THE TABLE
NO
YES
GET THE FIRST OR NEXT BLOCK IN THE CHAIN
IS IT EMPTY
NO
PUT IT IN THE TIME DELAY QUEUE
YES
RESTORE REGISTERS
RETURN TO THE ADDRESS IN REGISTER 14

Chart ND-2 READY ROUTINE

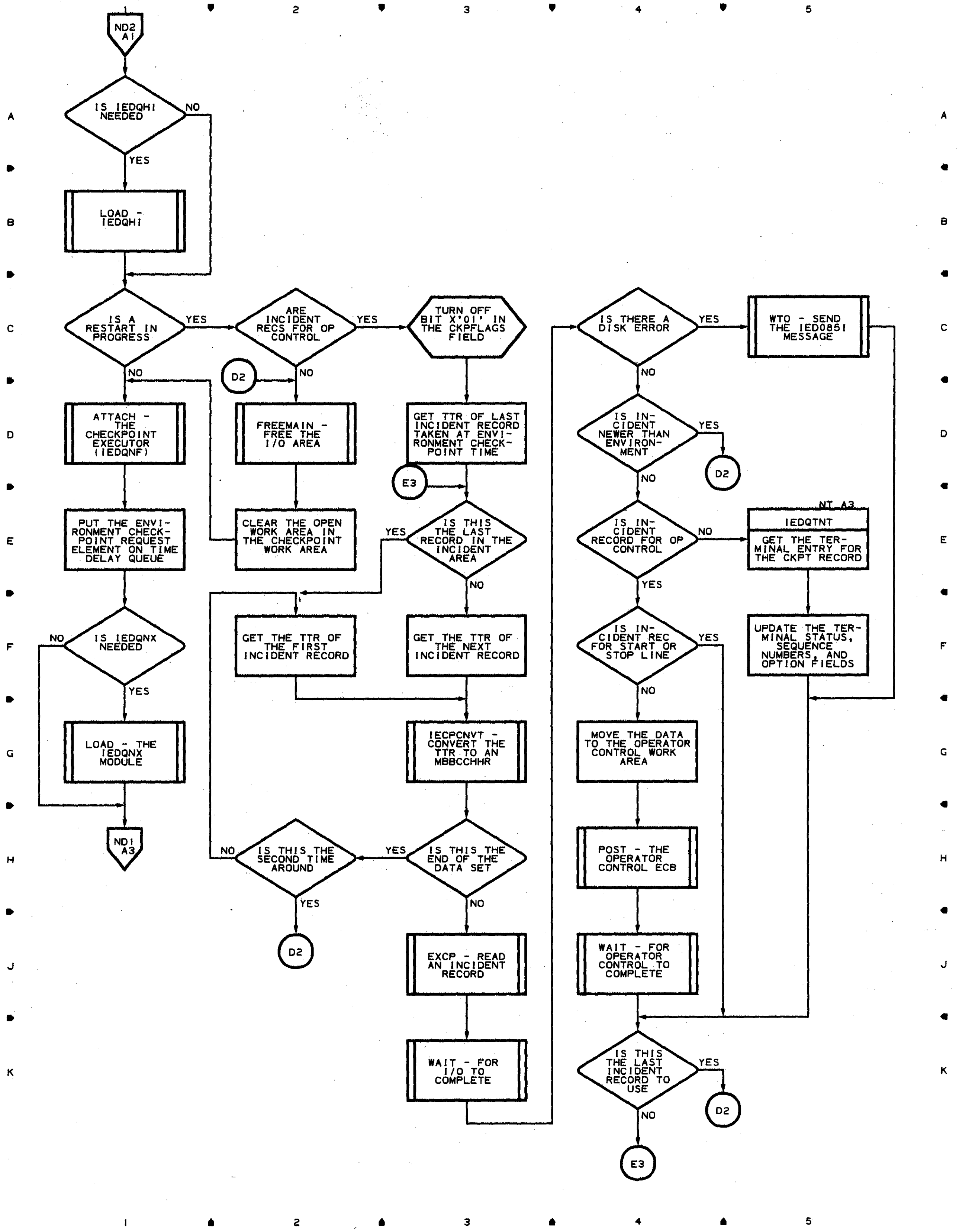
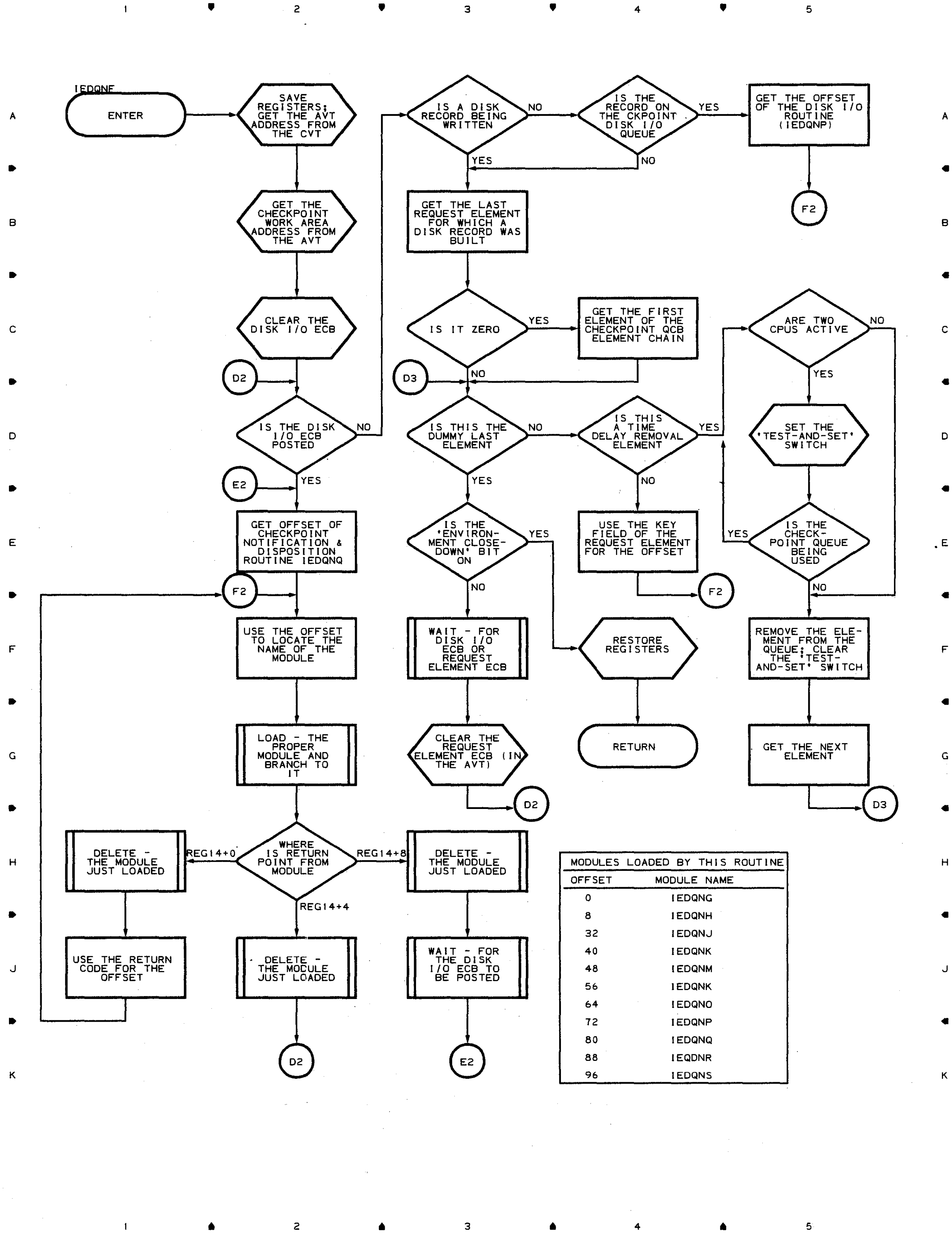

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
ND2 A1
IS IEDQHI NEEDED
NO
YES
LOAD - IEDQHI
IS A RESTART IN PROGRESS
YES
NO
ARE INCIDENT RECS FOR OP CONTROL
YES
NO
TURN OFF BIT X'01' IN THE CKPFLAGS FIELD
IS THERE A DISK ERROR
YES
NO
WTO - SEND THE IED0851 MESSAGE
D2
ATTACH - THE CHECKPOINT EXECUTOR (IEDQNF)
FREEMAIN - FREE THE I/O AREA
GET TTR OF LAST INCIDENT RECORD TAKEN AT ENVIRONMENT CHECKPOINT TIME
IS INCIDENT NEWER THAN ENVIRONMENT
YES
D2
NO
E3
PUT THE ENVIRONMENT CHECKPOINT REQUEST ELEMENT ON TIME DELAY QUEUE
CLEAR THE OPEN WORK AREA IN THE CHECKPOINT WORK AREA
YES
IS THIS THE LAST RECORD IN THE INCIDENT AREA
NO
IS INCIDENT RECORD FOR OP CONTROL
NO
YES
NT A3
IEDQTNT
GET THE TERMINAL ENTRY FOR THE CKPT RECORD
NO
IS IEDQNX NEEDED
YES
GET THE TTR OF THE FIRST INCIDENT RECORD
GET THE TTR OF THE NEXT INCIDENT RECORD
IS INCIDENT REC FOR START OR STOP LINE
YES
NO
UPDATE THE TERMINAL STATUS, SEQUENCE NUMBERS, AND OPTION FIELDS
LOAD - THE IEDQNX MODULE
IECPCNVT - CONVERT THE TTR TO AN MBBCCHHR
MOVE THE DATA TO THE OPERATOR CONTROL WORK AREA
ND1 A3
NO
IS THIS THE SECOND TIME AROUND
YES
IS THIS THE END OF THE DATA SET
YES
NO
POST - THE OPERATOR CONTROL ECB
D2
EXCP - READ AN INCIDENT RECORD
WAIT - FOR OPERATOR CONTROL TO COMPLETE
WAIT - FOR I/O TO COMPLETE
IS THIS THE LAST INCIDENT RECORD TO USE
YES
D2
NO
E3

Chart NF CHECKPOINT EXECUTOR ROUTINE


1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQNF
ENTER
SAVE REGISTERS; GET THE AVT ADDRESS FROM THE CVT
IS A DISK RECORD BEING WRITTEN
NO
YES
IS THE RECORD ON THE CKPOINT DISK I/O QUEUE
YES
NO
GET THE OFFSET OF THE DISK I/O ROUTINE (IEDQNP)
F2
GET THE CHECKPOINT WORK AREA ADDRESS FROM THE AVT
GET THE LAST REQUEST ELEMENT FOR WHICH A DISK RECORD WAS BUILT
CLEAR THE DISK I/O ECB
IS IT ZERO
YES
NO
GET THE FIRST ELEMENT OF THE CHECKPOINT QCB ELEMENT CHAIN
ARE TWO CPUS ACTIVE
NO
YES
D2
D3
IS THE DISK I/O ECB POSTED
NO
YES
IS THIS THE DUMMY LAST ELEMENT
NO
YES
IS THIS A TIME DELAY REMOVAL ELEMENT
YES
NO
SET THE 'TEST-AND-SET' SWITCH
E2
GET OFFSET OF CHECKPOINT NOTIFICATION & DISPOSITION ROUTINE IEDQNQ
IS THE 'ENVIRON-MENT CLOSE-DOWN' BIT ON
YES
NO
USE THE KEY FIELD OF THE REQUEST ELEMENT FOR THE OFFSET
F2
IS THE CHECK-POINT QUEUE BEING USED
YES
NO
F2
USE THE OFFSET TO LOCATE THE NAME OF THE MODULE
WAIT - FOR DISK I/O ECB OR REQUEST ELEMENT ECB
RESTORE REGISTERS
REMOVE THE ELE-MENT FROM THE QUEUE; CLEAR THE 'TEST-AND-SET' SWITCH
LOAD - THE PROPER MODULE AND BRANCH TO IT
CLEAR THE REQUEST ELEMENT ECB (IN THE AVT)
RETURN
GET THE NEXT ELEMENT
D2
D3
DELETE - THE MODULE JUST LOADED
REG14+0
WHERE IS RETURN POINT FROM MODULE
REG14+8
DELETE - THE MODULE JUST LOADED
REG14+4
USE THE RETURN CODE FOR THE OFFSET
DELETE - THE MODULE JUST LOADED
WAIT - FOR THE DISK I/O ECB TO BE POSTED
D2
E2

MODULES LOADED BY THIS ROUTINE	
OFFSET	MODULE NAME
0	IEDQNG
8	IEDQNH
32	IEDQNJ
40	IEDQNK
48	IEDQNM
56	IEDQNK
64	IEDQNO
72	IEDQNP
80	IEDQNQ
88	IEQDNR
96	IEDQNS

BUILD INCIDENT RECORD FOR MH ROUTINE

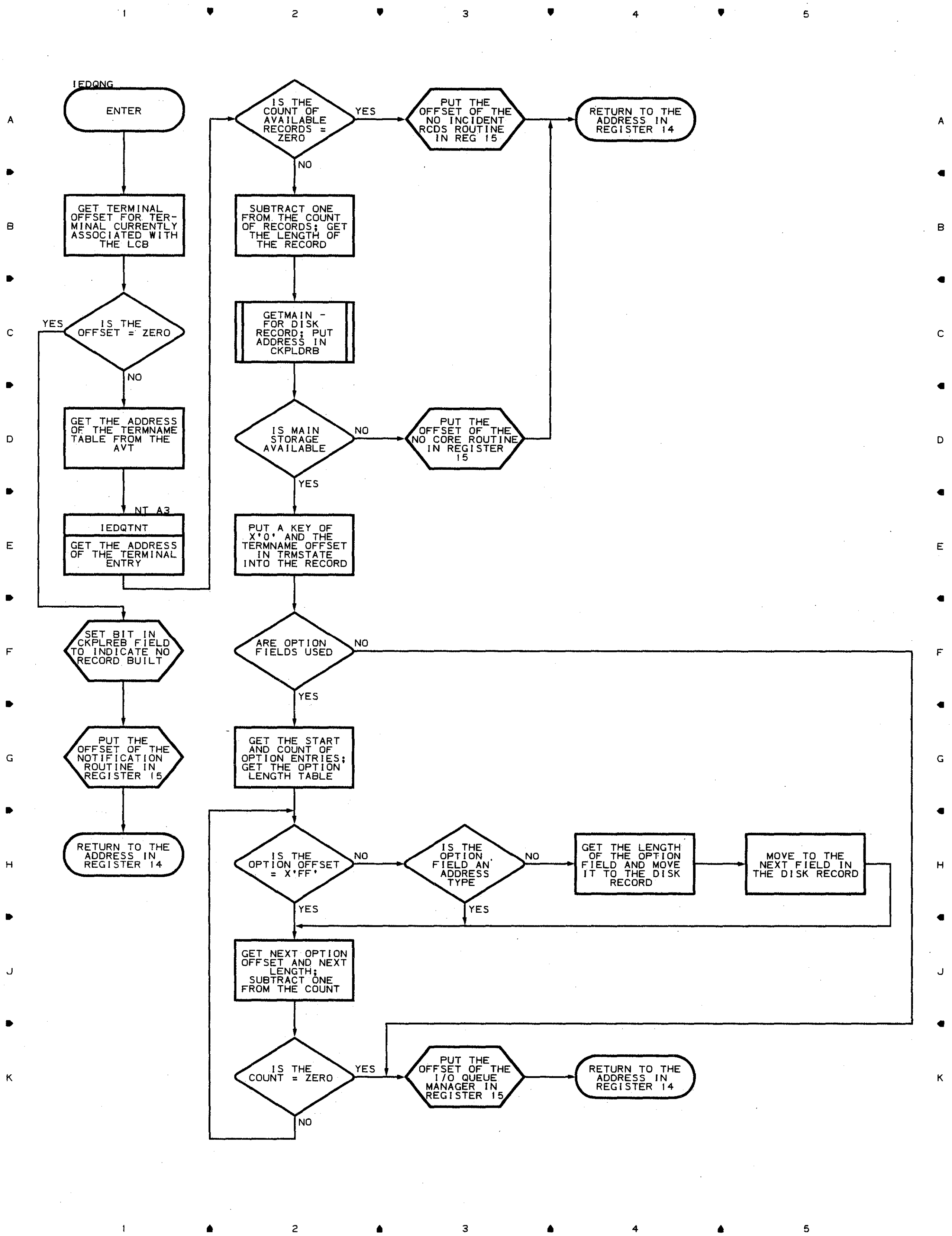
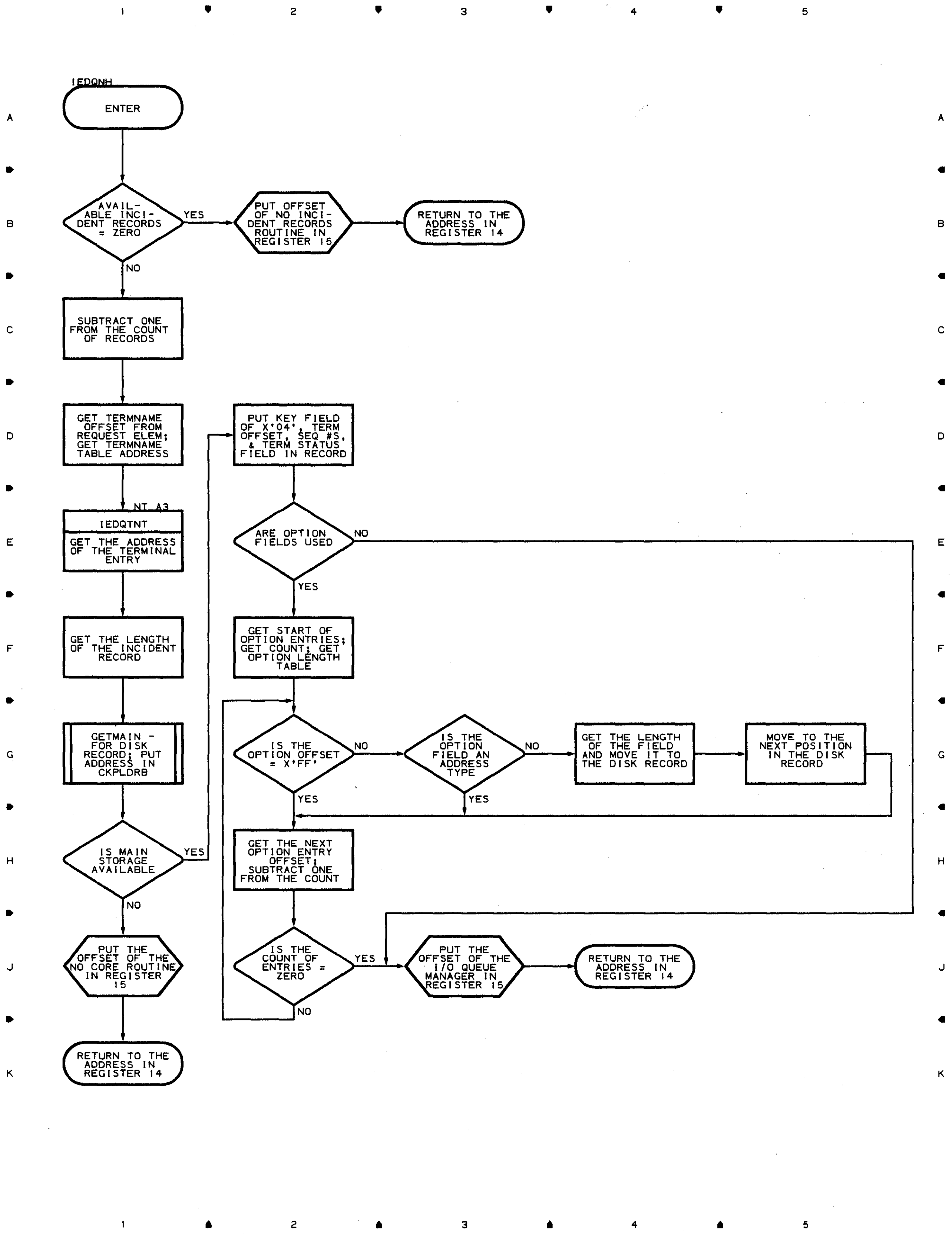
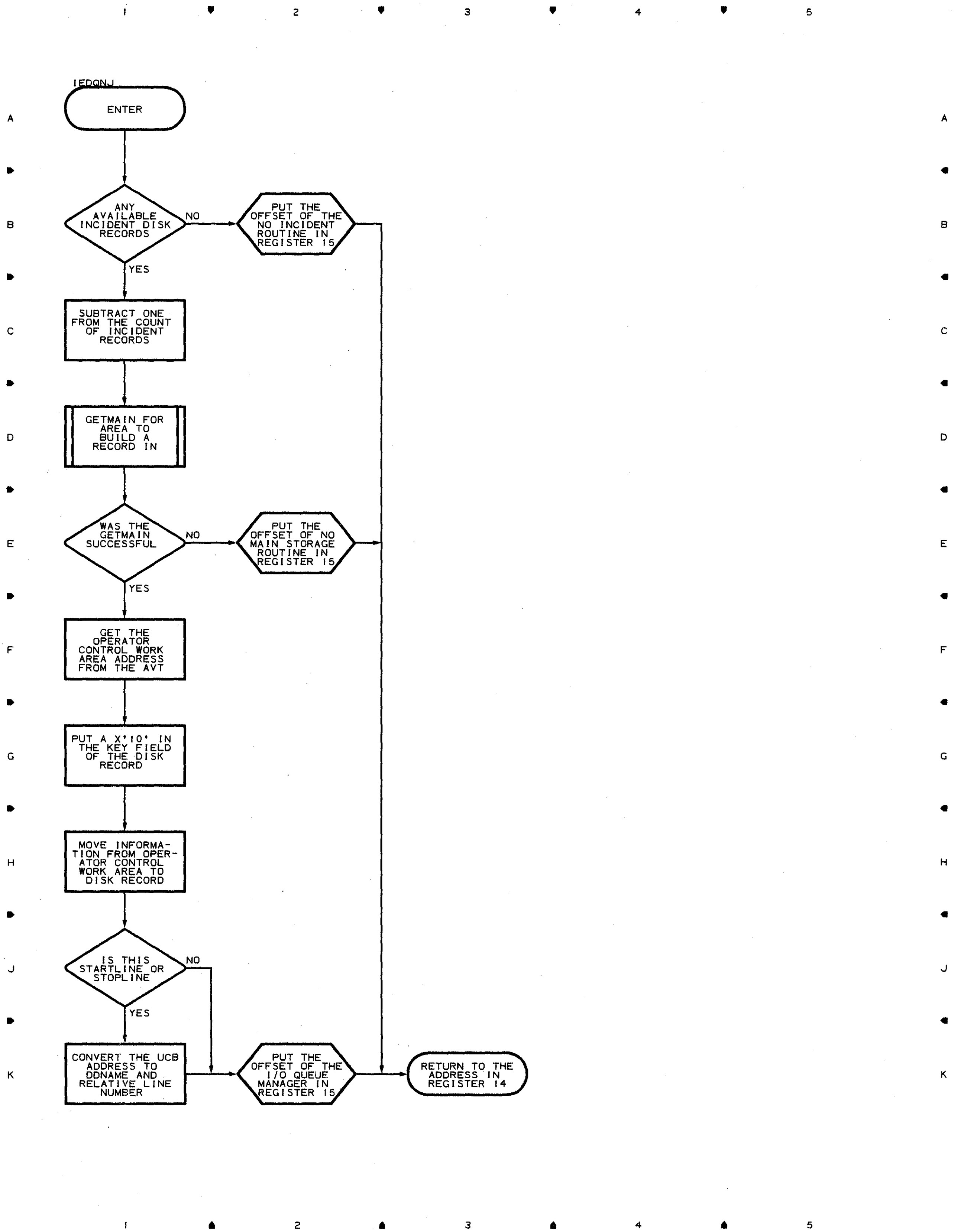

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQNG
ENTER
GET TERMINAL OFFSET FOR TERMINAL CURRENTLY ASSOCIATED WITH THE LCB
IS THE OFFSET = ZERO
YES
NO
GET THE ADDRESS OF THE TERMNAME TABLE FROM THE AVT
NT A3
IEDQTNT
GET THE ADDRESS OF THE TERMINAL ENTRY
SET BIT IN CKPLREB FIELD TO INDICATE NO RECORD BUILT
PUT THE OFFSET OF THE NOTIFICATION ROUTINE IN REGISTER 15
RETURN TO THE ADDRESS IN REGISTER 14
IS THE COUNT OF AVAILABLE RECORDS = ZERO
YES
NO
PUT THE OFFSET OF THE NO INCIDENT RCDS ROUTINE IN REG 15
RETURN TO THE ADDRESS IN REGISTER 14
SUBTRACT ONE FROM THE COUNT OF RECORDS; GET THE LENGTH OF THE RECORD
GETMAIN - FOR DISK RECORD; PUT ADDRESS IN CKPLDRB
IS MAIN STORAGE AVAILABLE
NO
YES
PUT THE OFFSET OF THE NO CORE ROUTINE IN REGISTER 15
PUT A KEY OF X'0' AND THE TERMNAME OFFSET IN TRMSTATE INTO THE RECORD
ARE OPTION FIELDS USED
NO
YES
GET THE START AND COUNT OF OPTION ENTRIES; GET THE OPTION LENGTH TABLE
IS THE OPTION OFFSET = X'FF'
NO
YES
IS THE OPTION FIELD AN ADDRESS TYPE
NO
YES
GET THE LENGTH OF THE OPTION FIELD AND MOVE IT TO THE DISK RECORD
MOVE TO THE NEXT FIELD IN THE DISK RECORD
GET NEXT OPTION OFFSET AND NEXT LENGTH; SUBTRACT ONE FROM THE COUNT
IS THE COUNT = ZERO
YES
NO
PUT THE OFFSET OF THE I/O QUEUE MANAGER IN REGISTER 15
RETURN TO THE ADDRESS IN REGISTER 14

Chart NH BUILD INCIDENT RECORD FOR TCHNG ROUTINE



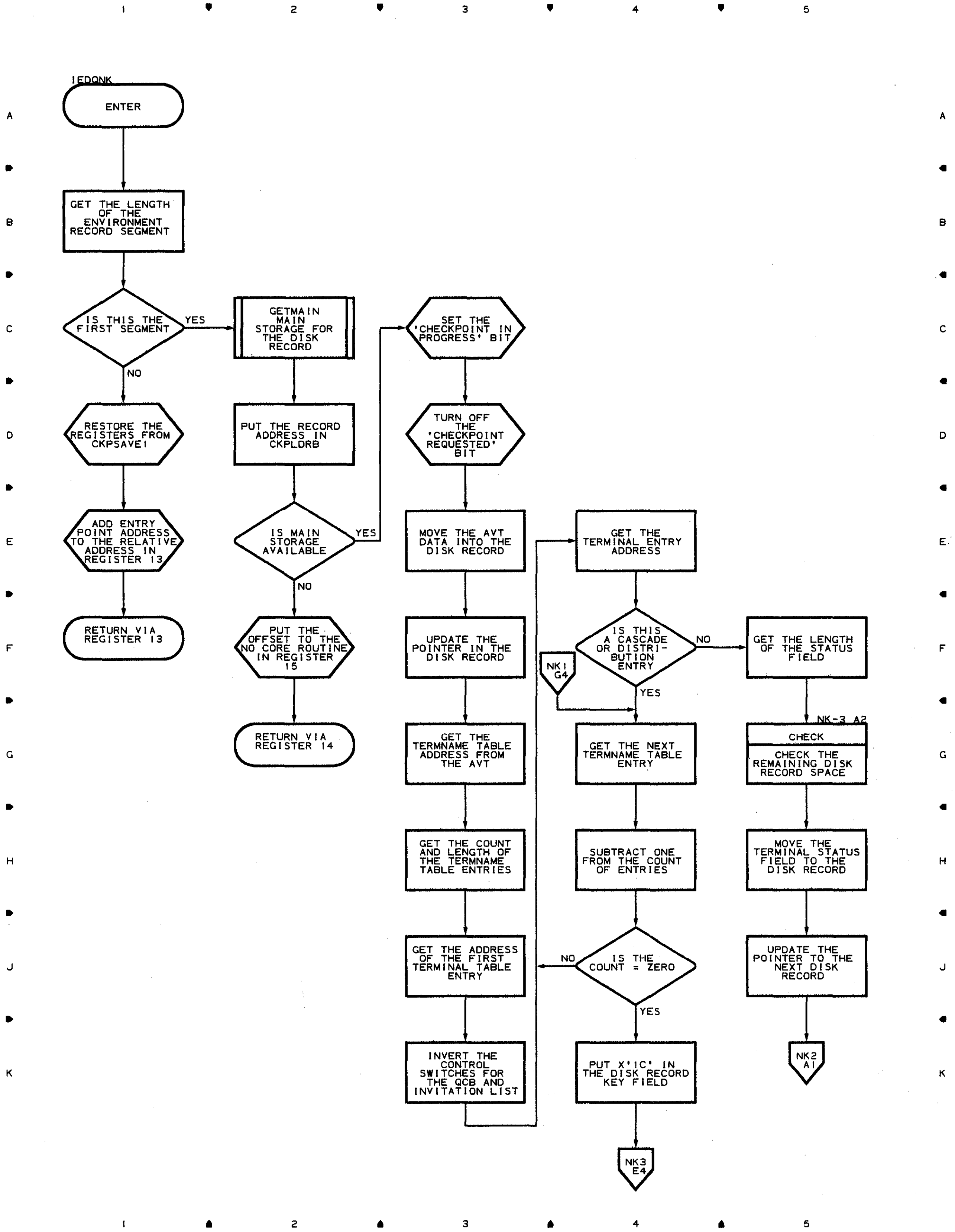
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQNH
ENTER
AVAIL-ABLE INCI-DENT RECORDS = ZERO
YES
NO
PUT OFFSET OF NO INCI-DENT RECORDS ROUTINE IN REGISTER 15
RETURN TO THE ADDRESS IN REGISTER 14
SUBTRACT ONE FROM THE COUNT OF RECORDS
GET TERMNAME OFFSET FROM REQUEST ELEM; GET TERMNAME TABLE ADDRESS
NT A3
IEDQTNT
GET THE ADDRESS OF THE TERMINAL ENTRY
GET THE LENGTH OF THE INCIDENT RECORD
GETMAIN - FOR DISK RECORD; PUT ADDRESS IN CKPLDRB
IS MAIN STORAGE AVAILABLE
YES
NO
PUT THE OFFSET OF THE NO CORE ROUTINE IN REGISTER 15
RETURN TO THE ADDRESS IN REGISTER 14
PUT KEY FIELD OF X'04', TERM OFFSET, SEQ #S, & TERM STATUS FIELD IN RECORD
ARE OPTION FIELDS USED
NO
YES
GET START OF OPTION ENTRIES; GET COUNT; GET OPTION LENGTH TABLE
IS THE OPTION OFFSET = X'FF'
NO
YES
IS THE OPTION FIELD AN ADDRESS TYPE
NO
YES
GET THE LENGTH OF THE FIELD AND MOVE IT TO THE DISK RECORD
MOVE TO THE NEXT POSITION IN THE DISK RECORD
GET THE NEXT OPTION ENTRY OFFSET; SUBTRACT ONE FROM THE COUNT
IS THE COUNT OF ENTRIES = ZERO
YES
NO
PUT THE OFFSET OF THE I/O QUEUE MANAGER IN REGISTER 15
RETURN TO THE ADDRESS IN REGISTER 14

Chart NJ INCIDENT CHECKPOINT FOR OPERATOR CONTROL ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQNJ
ENTER
ANY AVAILABLE INCIDENT DISK RECORDS
NO
YES
PUT THE OFFSET OF THE NO INCIDENT ROUTINE IN REGISTER 15
SUBTRACT ONE FROM THE COUNT OF INCIDENT RECORDS
GETMAIN FOR AREA TO BUILD A RECORD IN
WAS THE GETMAIN SUCCESSFUL
NO
YES
PUT THE OFFSET OF NO MAIN STORAGE ROUTINE IN REGISTER 15
GET THE OPERATOR CONTROL WORK AREA ADDRESS FROM THE AVT
PUT A X'10' IN THE KEY FIELD OF THE DISK RECORD
MOVE INFORMATION FROM OPERATOR CONTROL WORK AREA TO DISK RECORD
IS THIS STARTLINE OR STOPLINE
NO
YES
CONVERT THE UCB ADDRESS TO DDNAME AND RELATIVE LINE NUMBER
PUT THE OFFSET OF THE I/O QUEUE MANAGER IN REGISTER 15
RETURN TO THE ADDRESS IN REGISTER 14

Chart NK-1 ENVIRONMENT CHECKPOINT ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQNK
ENTER
GET THE LENGTH OF THE ENVIRONMENT RECORD SEGMENT
IS THIS THE FIRST SEGMENT
YES
NO
GETMAIN MAIN STORAGE FOR THE DISK RECORD
SET THE 'CHECKPOINT IN PROGRESS' BIT
RESTORE THE REGISTERS FROM CKPSAVE1
PUT THE RECORD ADDRESS IN CKPLDRB
TURN OFF THE 'CHECKPOINT REQUESTED' BIT
ADD ENTRY POINT ADDRESS TO THE RELATIVE ADDRESS IN REGISTER 13
IS MAIN STORAGE AVAILABLE
YES
NO
MOVE THE AVT DATA INTO THE DISK RECORD
GET THE TERMINAL ENTRY ADDRESS
RETURN VIA REGISTER 13
PUT THE OFFSET TO THE NO CORE ROUTINE IN REGISTER 15
UPDATE THE POINTER IN THE DISK RECORD
NK1 G4
IS THIS A CASCADE OR DISTRIBUTION ENTRY
NO
YES
GET THE LENGTH OF THE STATUS FIELD
RETURN VIA REGISTER 14
GET THE TERMNAME TABLE ADDRESS FROM THE AVT
GET THE NEXT TERMNAME TABLE ENTRY
NK-3 A2
CHECK
CHECK THE REMAINING DISK RECORD SPACE
GET THE COUNT AND LENGTH OF THE TERMNAME TABLE ENTRIES
SUBTRACT ONE FROM THE COUNT OF ENTRIES
MOVE THE TERMINAL STATUS FIELD TO THE DISK RECORD
GET THE ADDRESS OF THE FIRST TERMINAL TABLE ENTRY
NO
IS THE COUNT = ZERO
YES
UPDATE THE POINTER TO THE NEXT DISK RECORD
INVERT THE CONTROL SWITCHES FOR THE QCB AND INVITATION LIST
PUT X'1C' IN THE DISK RECORD KEY FIELD
NK2 A1
NK3 E4

Chart NK-2 ENVIRONMENT CHECKPOINT ROUTINE

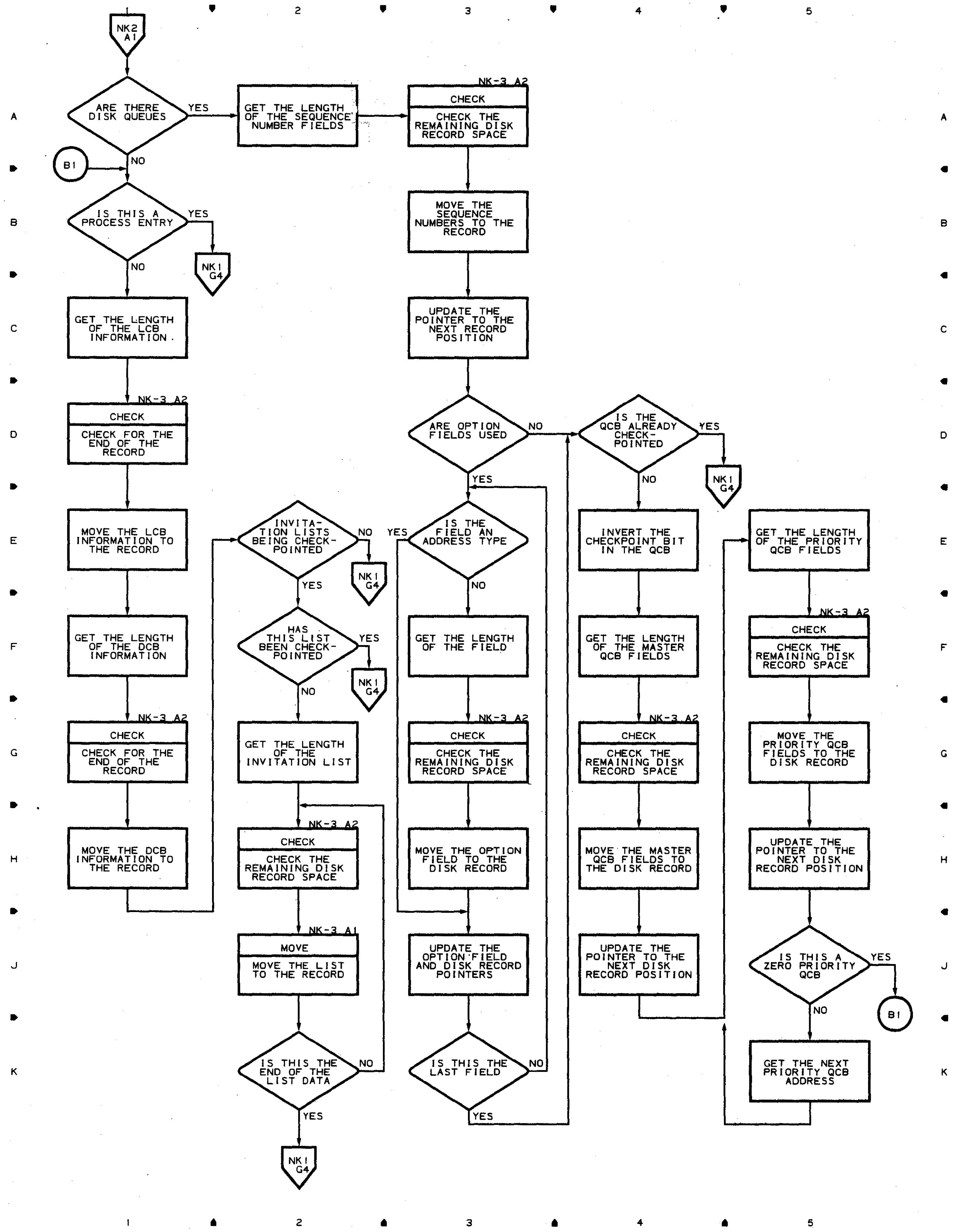

NK2 A1
ARE THERE DISK QUEUES
YES
NO
GET THE LENGTH OF THE SEQUENCE NUMBER FIELDS
NK-3 A2
CHECK
CHECK THE REMAINING DISK RECORD SPACE
B1
IS THIS A PROCESS ENTRY
YES
NO
NK1 G4
MOVE THE SEQUENCE NUMBERS TO THE RECORD
GET THE LENGTH OF THE LCB INFORMATION
UPDATE THE POINTER TO THE NEXT RECORD POSITION
NK-3 A2
CHECK
CHECK FOR THE END OF THE RECORD
ARE OPTION FIELDS USED
NO
YES
IS THE QCB ALREADY CHECK-POINTED
YES
NO
NK1 G4
MOVE THE LCB INFORMATION TO THE RECORD
INVITATION LISTS BEING CHECK-POINTED
NO
YES
NK1 G4
YES
IS THE FIELD AN ADDRESS TYPE
NO
INVERT THE CHECKPOINT BIT IN THE QCB
GET THE LENGTH OF THE PRIORITY QCB FIELDS
GET THE LENGTH OF THE DCB INFORMATION
HAS THIS LIST BEEN CHECK-POINTED
YES
NO
NK1 G4
GET THE LENGTH OF THE FIELD
GET THE LENGTH OF THE MASTER QCB FIELDS
NK-3 A2
CHECK
CHECK THE REMAINING DISK RECORD SPACE
NK-3 A2
CHECK
CHECK FOR THE END OF THE RECORD
GET THE LENGTH OF THE INVITATION LIST
NK-3 A2
CHECK
CHECK THE REMAINING DISK RECORD SPACE
NK-3 A2
CHECK
CHECK THE REMAINING DISK RECORD SPACE
MOVE THE PRIORITY QCB FIELDS TO THE DISK RECORD
MOVE THE DCB INFORMATION TO THE RECORD
NK-3 A2
CHECK
CHECK THE REMAINING DISK RECORD SPACE
MOVE THE OPTION FIELD TO THE DISK RECORD
MOVE THE MASTER QCB FIELDS TO THE DISK RECORD
UPDATE THE POINTER TO THE NEXT DISK RECORD POSITION
NK-3 A1
MOVE
MOVE THE LIST TO THE RECORD
UPDATE THE OPTION FIELD AND DISK RECORD POINTERS
UPDATE THE POINTER TO THE NEXT DISK RECORD POSITION
IS THIS A ZERO PRIORITY QCB
YES
NO
B1
IS THIS THE END OF THE LIST DATA
NO
YES
IS THIS THE LAST FIELD
NO
YES
GET THE NEXT PRIORITY QCB ADDRESS
NK1 G4
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart NK-3 ENVIRONMENT CHECKPOINT ROUTINE

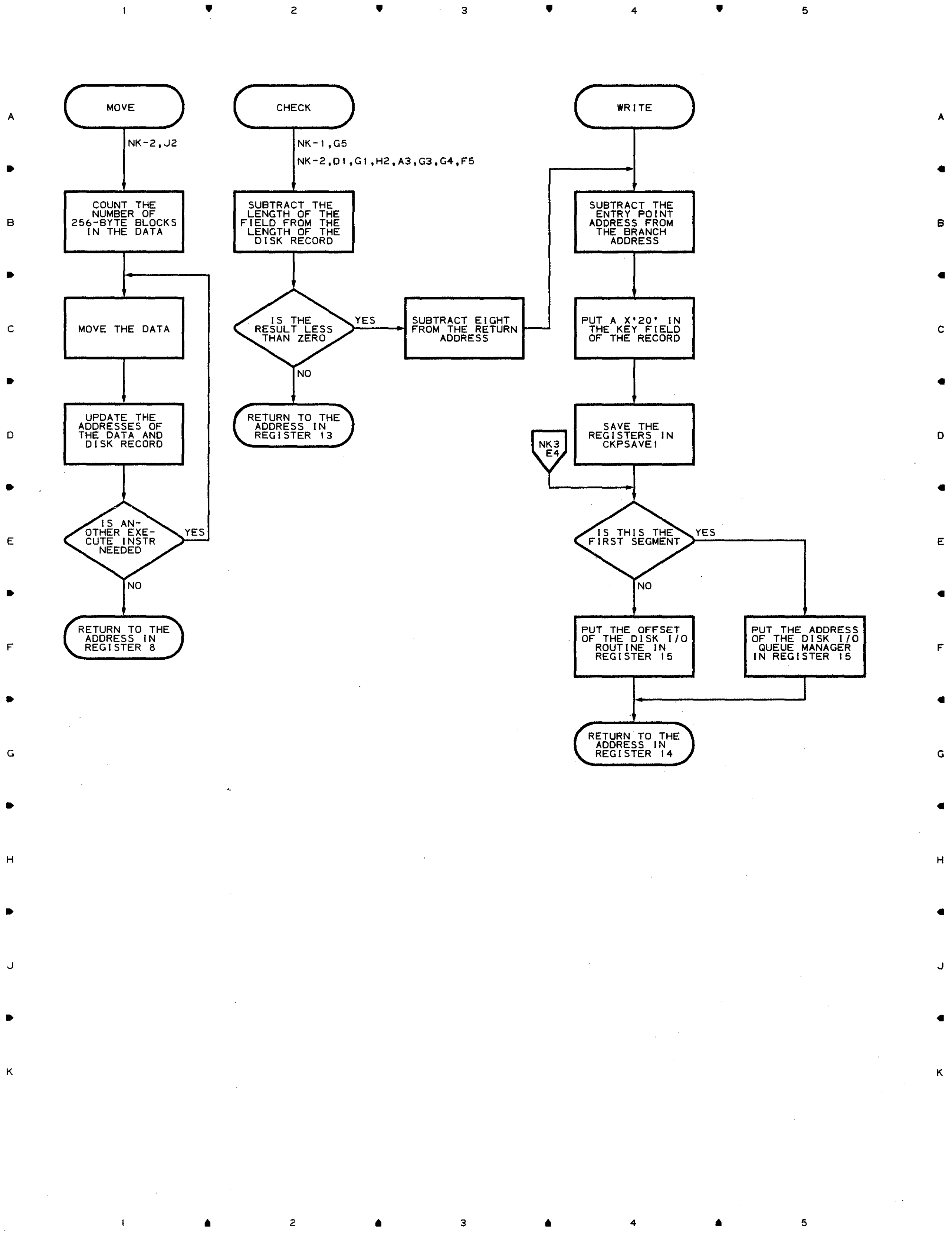

MOVE
NK-2,J2
COUNT THE NUMBER OF 256-BYTE BLOCKS IN THE DATA
MOVE THE DATA
UPDATE THE ADDRESSES OF THE DATA AND DISK RECORD
IS ANOTHER EXECUTE INSTR NEEDED
YES
NO
RETURN TO THE ADDRESS IN REGISTER 8
CHECK
NK-1,G5
NK-2,D1,G1,H2,A3,G3,G4,F5
SUBTRACT THE LENGTH OF THE FIELD FROM THE LENGTH OF THE DISK RECORD
IS THE RESULT LESS THAN ZERO
YES
NO
SUBTRACT EIGHT FROM THE RETURN ADDRESS
RETURN TO THE ADDRESS IN REGISTER 13
WRITE
SUBTRACT THE ENTRY POINT ADDRESS FROM THE BRANCH ADDRESS
PUT A X'20' IN THE KEY FIELD OF THE RECORD
SAVE THE REGISTERS IN CKPSAVE1
NK3 E4
IS THIS THE FIRST SEGMENT
YES
NO
PUT THE OFFSET OF THE DISK I/O ROUTINE IN REGISTER 15
PUT THE ADDRESS OF THE DISK I/O QUEUE MANAGER IN REGISTER 15
RETURN TO THE ADDRESS IN REGISTER 14
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart NM BUILD CKREQ DISK RECORD ROUTINE

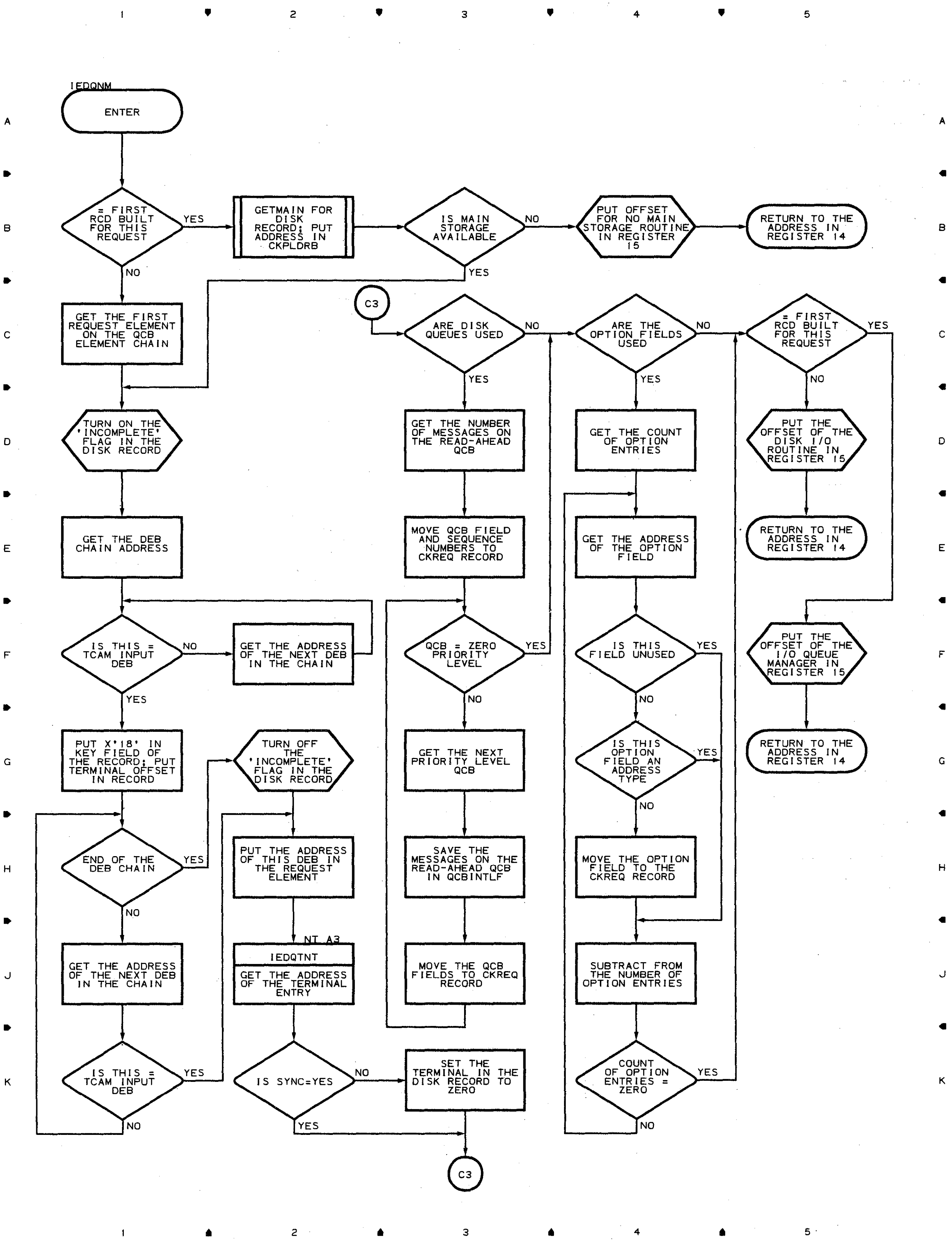

IEDQNM
ENTER
= FIRST RCD BUILT FOR THIS REQUEST
YES
NO
GETMAIN FOR DISK RECORD; PUT ADDRESS IN CKPLDRB
IS MAIN STORAGE AVAILABLE
NO
YES
PUT OFFSET FOR NO MAIN STORAGE ROUTINE IN REGISTER 15
RETURN TO THE ADDRESS IN REGISTER 14
GET THE FIRST REQUEST ELEMENT ON THE QCB ELEMENT CHAIN
C3
ARE DISK QUEUES USED
NO
YES
ARE THE OPTION FIELDS USED
NO
YES
= FIRST RCD BUILT FOR THIS REQUEST
YES
NO
TURN ON THE 'INCOMPLETE' FLAG IN THE DISK RECORD
GET THE NUMBER OF MESSAGES ON THE READ-AHEAD QCB
GET THE COUNT OF OPTION ENTRIES
PUT THE OFFSET OF THE DISK I/O ROUTINE IN REGISTER 15
GET THE DEB CHAIN ADDRESS
MOVE QCB FIELD AND SEQUENCE NUMBERS TO CKREQ RECORD
GET THE ADDRESS OF THE OPTION FIELD
RETURN TO THE ADDRESS IN REGISTER 14
IS THIS = TCAM INPUT DEB
NO
YES
GET THE ADDRESS OF THE NEXT DEB IN THE CHAIN
QCB = ZERO PRIORITY LEVEL
YES
NO
IS THIS FIELD UNUSED
YES
NO
PUT THE OFFSET OF THE I/O QUEUE MANAGER IN REGISTER 15
PUT X'18' IN KEY FIELD OF THE RECORD; PUT TERMINAL OFFSET IN RECORD
TURN OFF THE 'INCOMPLETE' FLAG IN THE DISK RECORD
GET THE NEXT PRIORITY LEVEL QCB
IS THIS OPTION FIELD AN ADDRESS TYPE
YES
NO
RETURN TO THE ADDRESS IN REGISTER 14
END OF THE DEB CHAIN
YES
NO
PUT THE ADDRESS OF THIS DEB IN THE REQUEST ELEMENT
SAVE THE MESSAGES ON THE READ-AHEAD QCB IN QCBINTLF
MOVE THE OPTION FIELD TO THE CKREQ RECORD
NT A3
IEDQTNT
GET THE ADDRESS OF THE NEXT DEB IN THE CHAIN
GET THE ADDRESS OF THE TERMINAL ENTRY
MOVE THE QCB FIELDS TO CKREQ RECORD
SUBTRACT FROM THE NUMBER OF OPTION ENTRIES
IS THIS = TCAM INPUT DEB
YES
NO
IS SYNC=YES
NO
YES
SET THE TERMINAL IN THE DISK RECORD TO ZERO
COUNT OF OPTION ENTRIES = ZERO
YES
NO
C3
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart NO CHECKPOINT QUEUE MANAGER ROUTINE

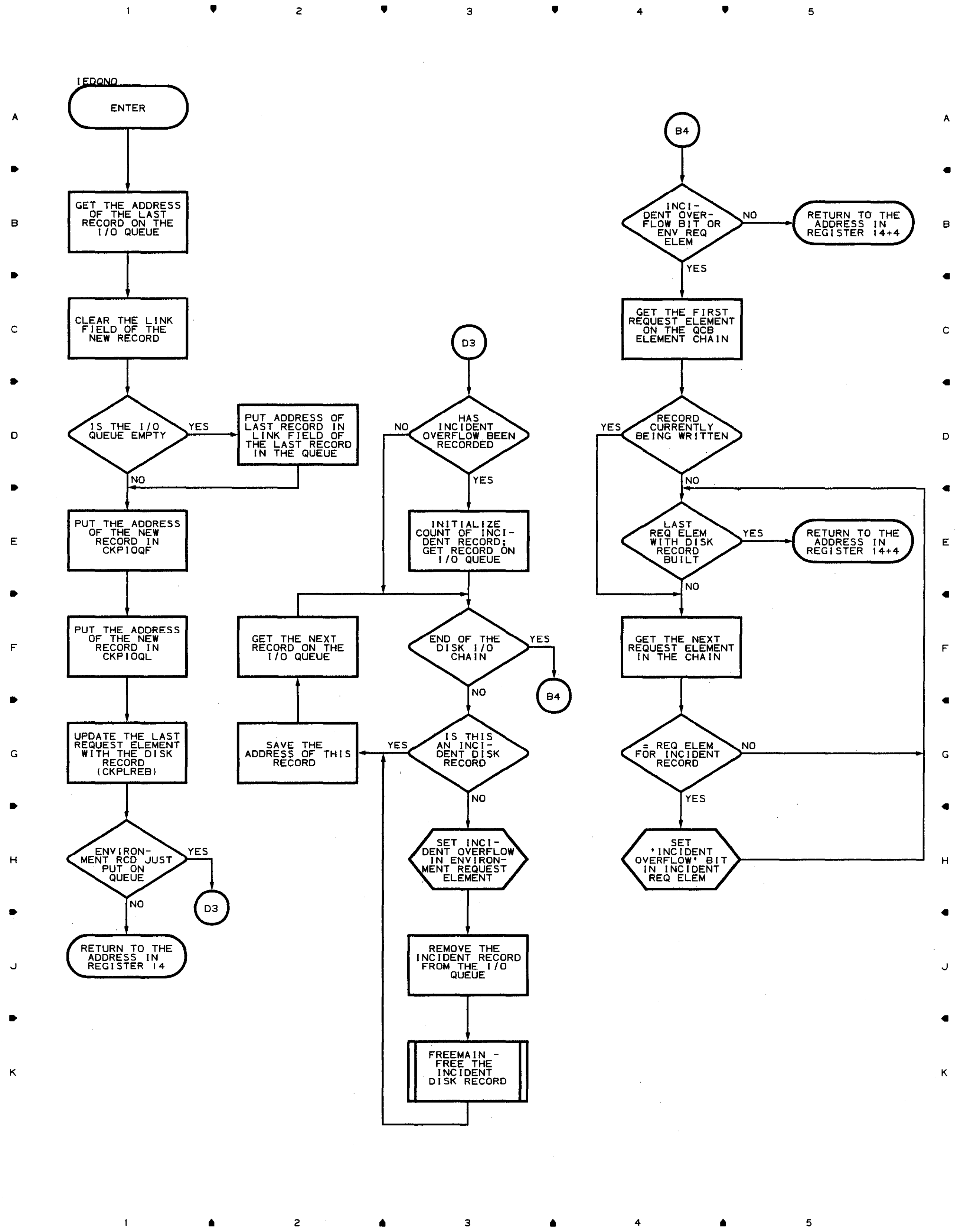

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQNO
ENTER
GET THE ADDRESS OF THE LAST RECORD ON THE I/O QUEUE
CLEAR THE LINK FIELD OF THE NEW RECORD
IS THE I/O QUEUE EMPTY
YES
PUT ADDRESS OF LAST RECORD IN LINK FIELD OF THE LAST RECORD IN THE QUEUE
NO
PUT THE ADDRESS OF THE NEW RECORD IN CKPIOQF
PUT THE ADDRESS OF THE NEW RECORD IN CKPIOQL
UPDATE THE LAST REQUEST ELEMENT WITH THE DISK RECORD (CKPLREB)
ENVIRON-MENT RCD JUST PUT ON QUEUE
YES
D3
NO
RETURN TO THE ADDRESS IN REGISTER 14
D3
HAS INCIDENT OVERFLOW BEEN RECORDED
NO
YES
INITIALIZE COUNT OF INCI-DENT RECORD; GET RECORD ON I/O QUEUE
GET THE NEXT RECORD ON THE I/O QUEUE
END OF THE DISK I/O CHAIN
YES
B4
NO
SAVE THE ADDRESS OF THIS RECORD
YES
IS THIS AN INCI-DENT DISK RECORD
NO
SET INCI-DENT OVERFLOW IN ENVIRON-MENT REQUEST ELEMENT
REMOVE THE INCIDENT RECORD FROM THE I/O QUEUE
FREEMAIN - FREE THE INCIDENT DISK RECORD
B4
INCI-DENT OVER-FLOW BIT OR ENV REQ ELEM
NO
RETURN TO THE ADDRESS IN REGISTER 14+4
YES
GET THE FIRST REQUEST ELEMENT ON THE QCB ELEMENT CHAIN
RECORD CURRENTLY BEING WRITTEN
YES
NO
LAST REQ ELEM WITH DISK RECORD BUILT
YES
RETURN TO THE ADDRESS IN REGISTER 14+4
NO
GET THE NEXT REQUEST ELEMENT IN THE CHAIN
= REQ ELEM FOR INCIDENT RECORD
NO
YES
SET 'INCIDENT OVERFLOW' BIT IN INCIDENT REQ ELEM

Chart NP-1 CHECKPOINT DISK I/O ROUTINE

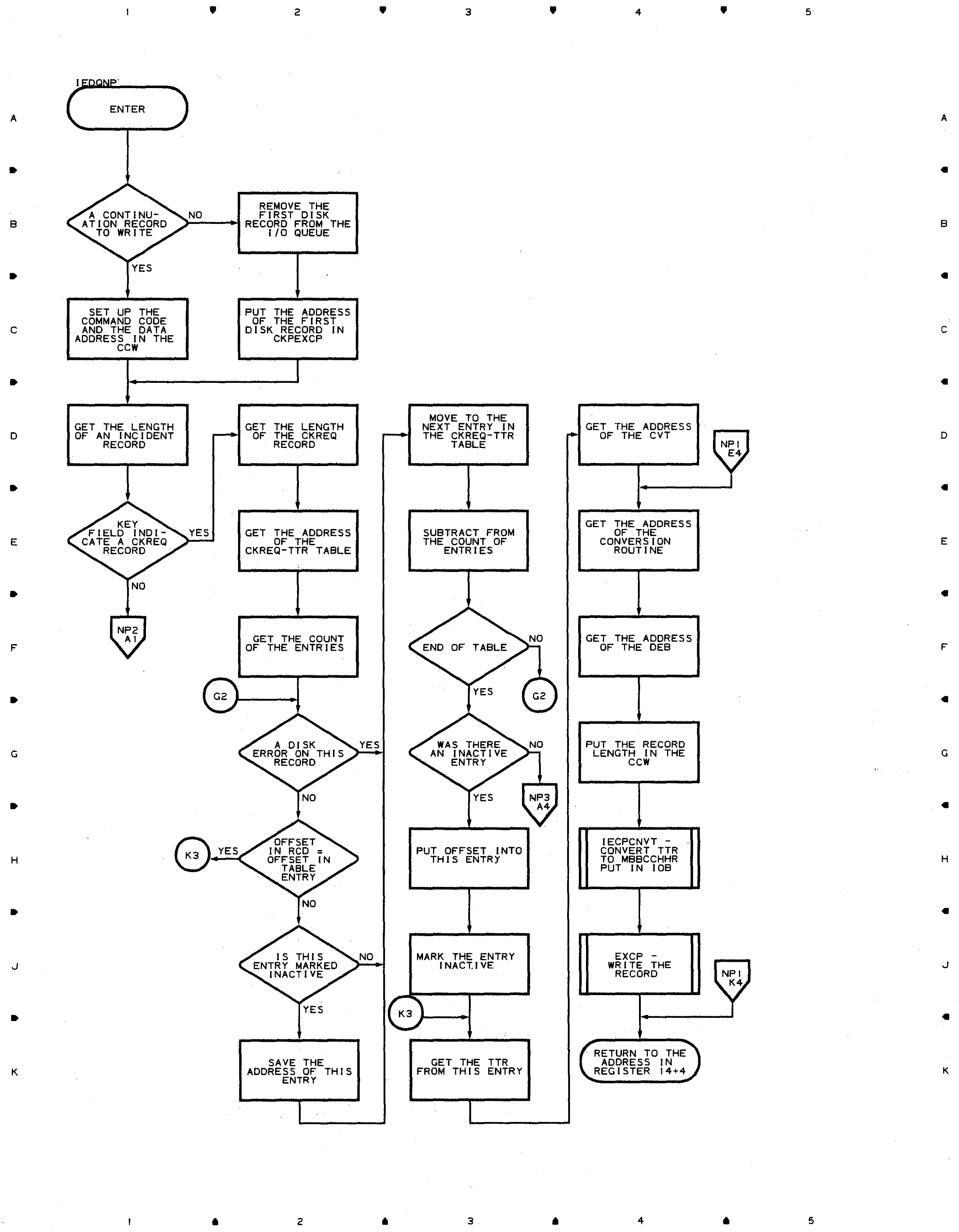

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQNP
ENTER
A CONTINUATION RECORD TO WRITE
NO
YES
REMOVE THE FIRST DISK RECORD FROM THE I/O QUEUE
SET UP THE COMMAND CODE AND THE DATA ADDRESS IN THE CCW
PUT THE ADDRESS OF THE FIRST DISK RECORD IN CKPEXCP
GET THE LENGTH OF AN INCIDENT RECORD
KEY FIELD INDICATE A CKREQ RECORD
YES
NO
NP2 A1
GET THE LENGTH OF THE CKREQ RECORD
GET THE ADDRESS OF THE CKREQ-TTR TABLE
GET THE COUNT OF THE ENTRIES
G2
A DISK ERROR ON THIS RECORD
YES
NO
OFFSET IN RCD = OFFSET IN TABLE ENTRY
YES
K3
NO
IS THIS ENTRY MARKED INACTIVE
NO
YES
SAVE THE ADDRESS OF THIS ENTRY
MOVE TO THE NEXT ENTRY IN THE CKREQ-TTR TABLE
SUBTRACT FROM THE COUNT OF ENTRIES
END OF TABLE
NO
G2
YES
WAS THERE AN INACTIVE ENTRY
NO
NP3 A4
YES
PUT OFFSET INTO THIS ENTRY
MARK THE ENTRY INACTIVE
K3
GET THE TTR FROM THIS ENTRY
GET THE ADDRESS OF THE CVT
NP1 E4
GET THE ADDRESS OF THE CONVERSION ROUTINE
GET THE ADDRESS OF THE DEB
PUT THE RECORD LENGTH IN THE CCW
IECPCNVT - CONVERT TTR TO MBBCCHHR PUT IN IOB
EXCP - WRITE THE RECORD
NP1 K4
RETURN TO THE ADDRESS IN REGISTER 14+4

Chart NP-2 CHECKPOINT DISK I/O ROUTINE

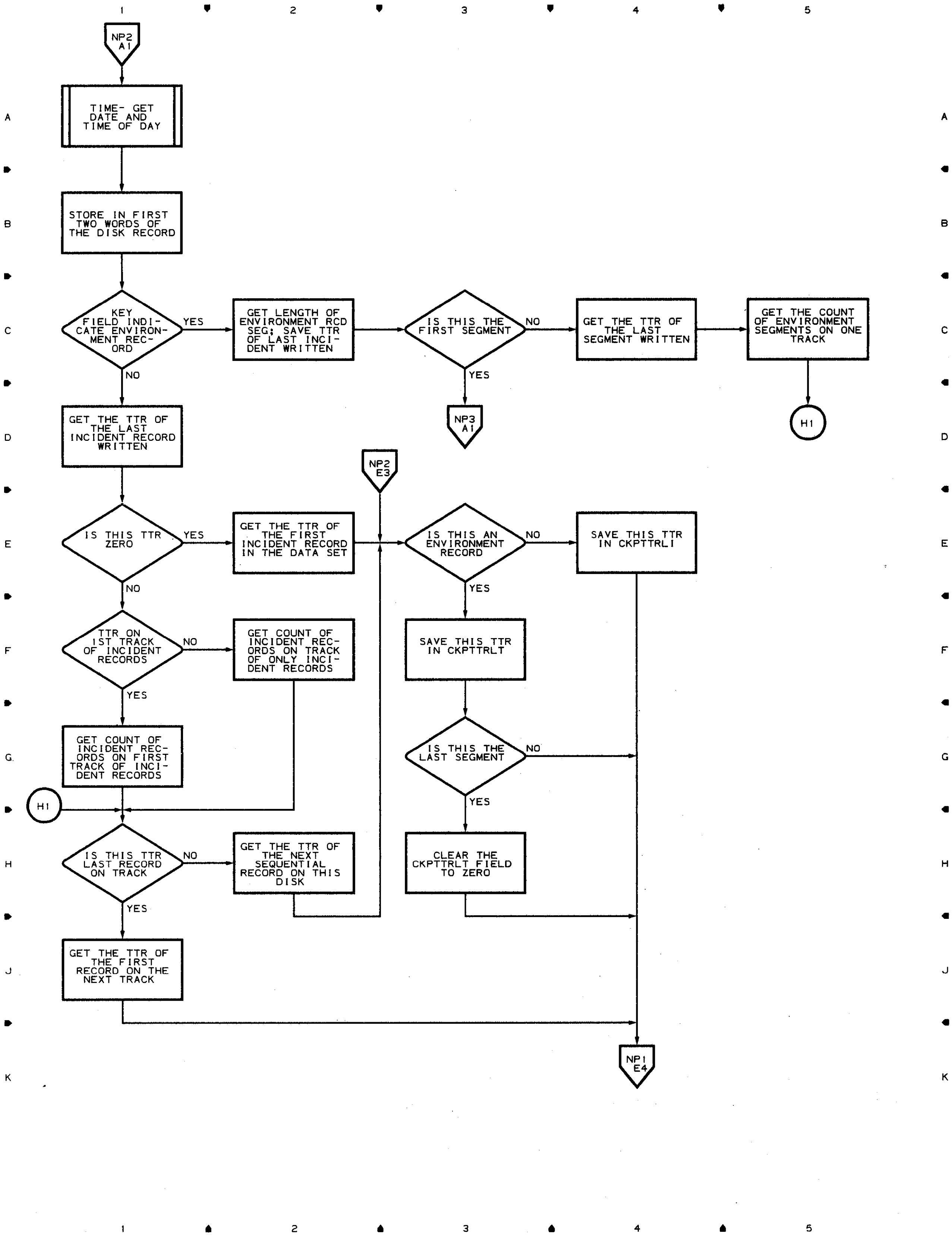
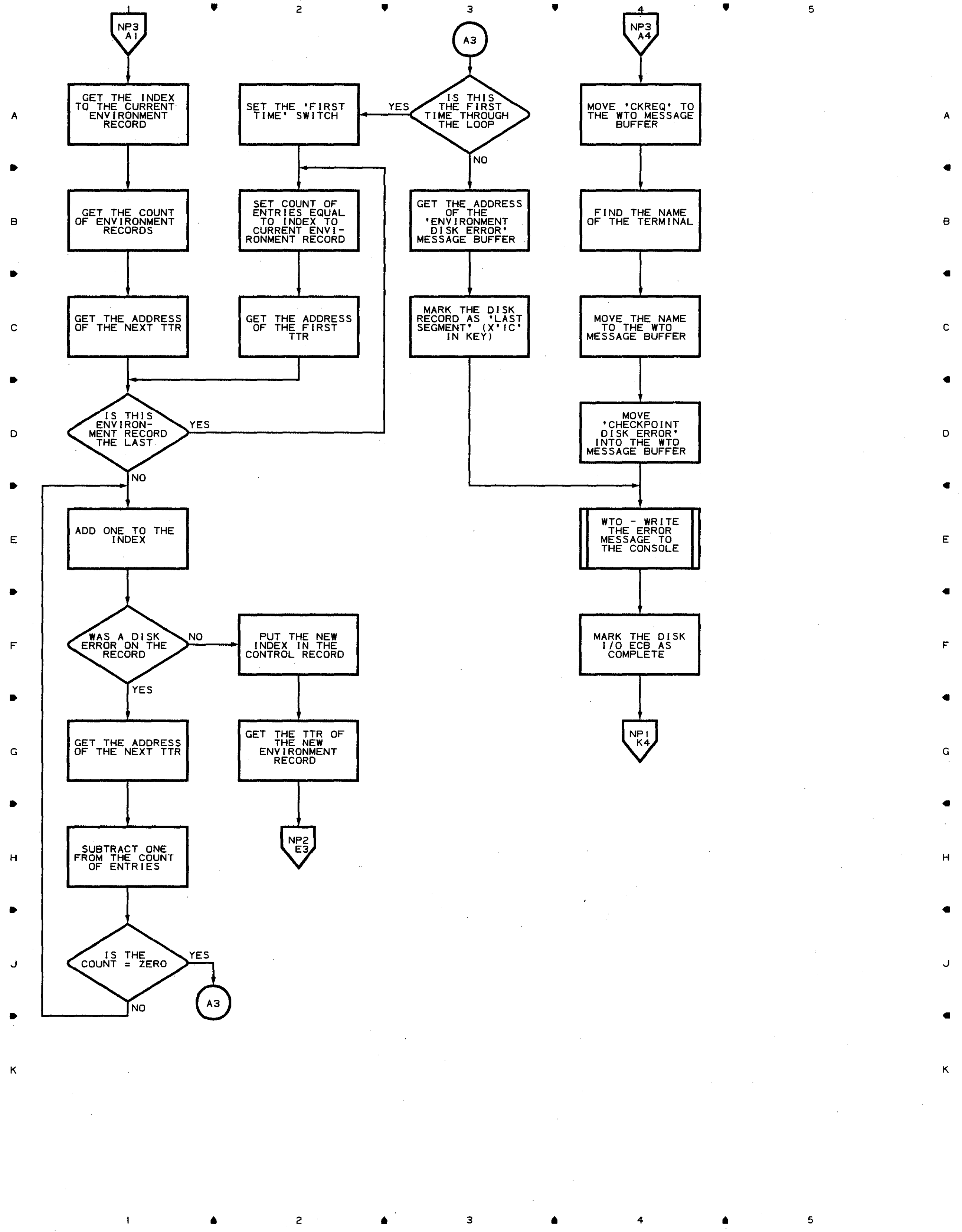
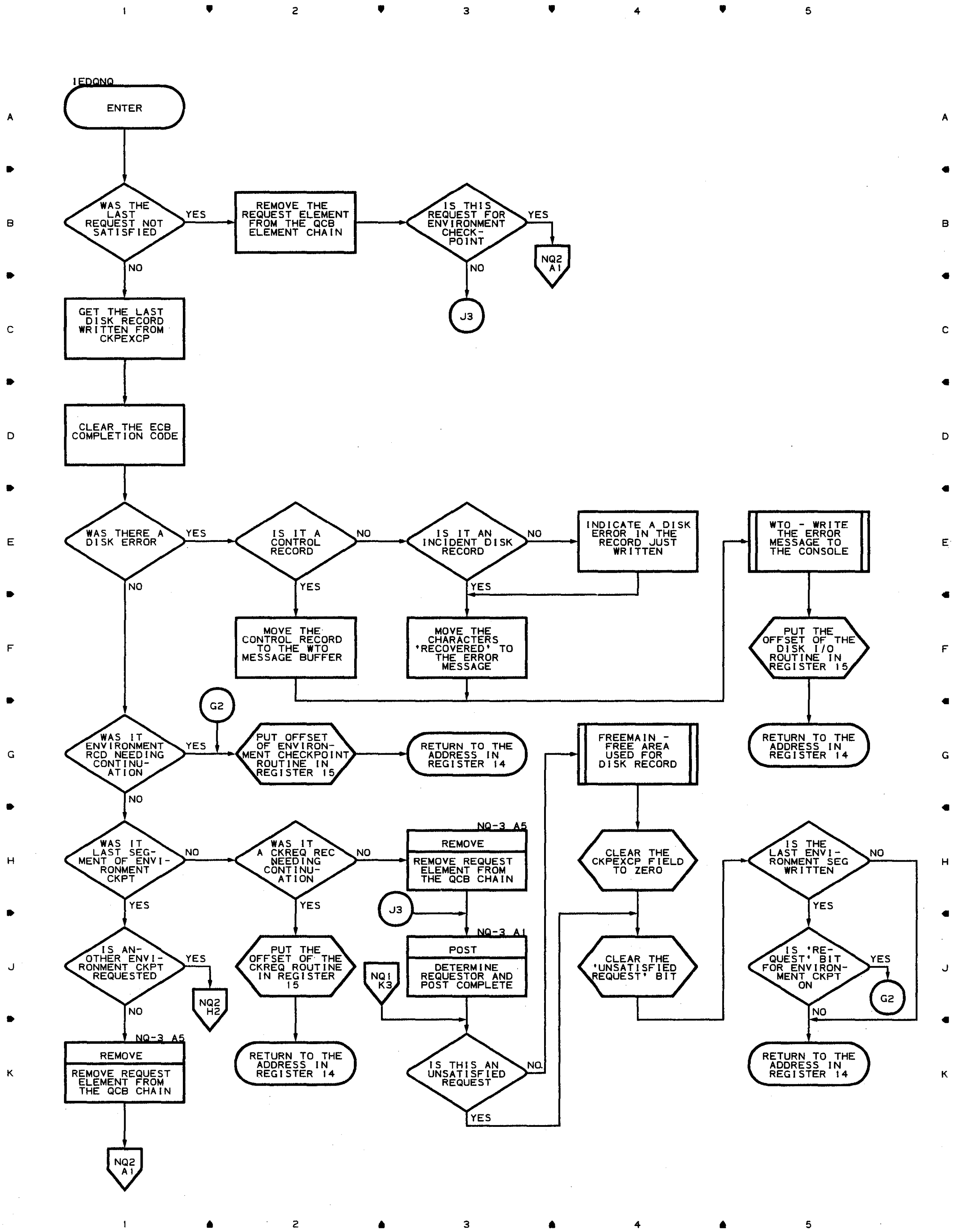

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
NP2 A1
TIME- GET DATE AND TIME OF DAY
STORE IN FIRST TWO WORDS OF THE DISK RECORD
KEY FIELD INDICATE ENVIRONMENT RECORD
YES
NO
GET LENGTH OF ENVIRONMENT RCD SEG; SAVE TTR OF LAST INCIDENT WRITTEN
IS THIS THE FIRST SEGMENT
NO
YES
NP3 A1
GET THE TTR OF THE LAST SEGMENT WRITTEN
GET THE COUNT OF ENVIRONMENT SEGMENTS ON ONE TRACK
H1
GET THE TTR OF THE LAST INCIDENT RECORD WRITTEN
NP2 E3
IS THIS TTR ZERO
YES
NO
GET THE TTR OF THE FIRST INCIDENT RECORD IN THE DATA SET
IS THIS AN ENVIRONMENT RECORD
NO
YES
SAVE THIS TTR IN CKPTTRLI
TTR ON 1ST TRACK OF INCIDENT RECORDS
NO
YES
GET COUNT OF INCIDENT RECORDS ON TRACK OF ONLY INCIDENT RECORDS
SAVE THIS TTR IN CKPTTRLT
GET COUNT OF INCIDENT RECORDS ON FIRST TRACK OF INCIDENT RECORDS
IS THIS THE LAST SEGMENT
NO
YES
H1
IS THIS TTR LAST RECORD ON TRACK
NO
YES
GET THE TTR OF THE NEXT SEQUENTIAL RECORD ON THIS DISK
CLEAR THE CKPTTRLT FIELD TO ZERO
GET THE TTR OF THE FIRST RECORD ON THE NEXT TRACK
NP1 E4

Chart NP-3 CHECKPOINT DISK I/O ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
NP3 A1
GET THE INDEX TO THE CURRENT ENVIRONMENT RECORD
GET THE COUNT OF ENVIRONMENT RECORDS
GET THE ADDRESS OF THE NEXT TTR
IS THIS ENVIRON-MENT RECORD THE LAST
YES
NO
ADD ONE TO THE INDEX
WAS A DISK ERROR ON THE RECORD
NO
YES
GET THE ADDRESS OF THE NEXT TTR
SUBTRACT ONE FROM THE COUNT OF ENTRIES
IS THE COUNT = ZERO
YES
NO
A3
SET THE 'FIRST TIME' SWITCH
SET COUNT OF ENTRIES EQUAL TO INDEX TO CURRENT ENVI-RONMENT RECORD
GET THE ADDRESS OF THE FIRST TTR
PUT THE NEW INDEX IN THE CONTROL RECORD
GET THE TTR OF THE NEW ENVIRONMENT RECORD
NP2 E3
A3
IS THIS THE FIRST TIME THROUGH THE LOOP
YES
NO
GET THE ADDRESS OF THE 'ENVIRONMENT DISK ERROR' MESSAGE BUFFER
MARK THE DISK RECORD AS 'LAST SEGMENT' (X'1C' IN KEY)
NP3 A4
MOVE 'CKREQ' TO THE WTO MESSAGE BUFFER
FIND THE NAME OF THE TERMINAL
MOVE THE NAME TO THE WTO MESSAGE BUFFER
MOVE 'CHECKPOINT DISK ERROR' INTO THE WTO MESSAGE BUFFER
WTO - WRITE THE ERROR MESSAGE TO THE CONSOLE
MARK THE DISK I/O ECB AS COMPLETE
NP1 K4

Chart NQ-1 CHECKPOINT NOTIFICATION AND DISPOSITION ROUTINE



IEDQNQ
ENTER
WAS THE LAST REQUEST NOT SATISFIED
YES
NO
REMOVE THE REQUEST ELEMENT FROM THE QCB ELEMENT CHAIN
IS THIS REQUEST FOR ENVIRONMENT CHECK-POINT
YES
NO
NQ2 A1
J3
GET THE LAST DISK RECORD WRITTEN FROM CKPEXCP
CLEAR THE ECB COMPLETION CODE
WAS THERE A DISK ERROR
YES
NO
IS IT A CONTROL RECORD
NO
YES
IS IT AN INCIDENT DISK RECORD
NO
YES
INDICATE A DISK ERROR IN THE RECORD JUST WRITTEN
WTO - WRITE THE ERROR MESSAGE TO THE CONSOLE
MOVE THE CONTROL RECORD TO THE WTO MESSAGE BUFFER
MOVE THE CHARACTERS 'RECOVERED' TO THE ERROR MESSAGE
PUT THE OFFSET OF THE DISK I/O ROUTINE IN REGISTER 15
G2
WAS IT ENVIRONMENT RCD NEEDING CONTINU-ATION
YES
NO
PUT OFFSET OF ENVIRON-MENT CHECKPOINT ROUTINE IN REGISTER 15
RETURN TO THE ADDRESS IN REGISTER 14
FREEMAIN - FREE AREA USED FOR DISK RECORD
RETURN TO THE ADDRESS IN REGISTER 14
WAS IT LAST SEG-MENT OF ENVI-RONMENT CKPT
NO
YES
WAS IT A CKREQ REC NEEDING CONTINU-ATION
NO
YES
NQ-3 A5
REMOVE
REMOVE REQUEST ELEMENT FROM THE QCB CHAIN
CLEAR THE CKPEXCP FIELD TO ZERO
IS THE LAST ENVI-RONMENT SEG WRITTEN
NO
YES
J3
NQ-3 A1
POST
DETERMINE REQUESTOR AND POST COMPLETE
IS AN-OTHER ENVI-RONMENT CKPT REQUESTED
YES
NO
NQ2 H2
PUT THE OFFSET OF THE CKREQ ROUTINE IN REGISTER 15
NQ1 K3
CLEAR THE 'UNSATISFIED REQUEST' BIT
IS 'RE-QUEST' BIT FOR ENVIRON-MENT CKPT ON
YES
NO
G2
NQ-3 A5
REMOVE
REMOVE REQUEST ELEMENT FROM THE QCB CHAIN
RETURN TO THE ADDRESS IN REGISTER 14
IS THIS AN UNSATISFIED REQUEST
NO
YES
RETURN TO THE ADDRESS IN REGISTER 14
NQ2 A1
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart NQ-2 CHECKPOINT NOTIFICATION AND DISPOSITION ROUTINE

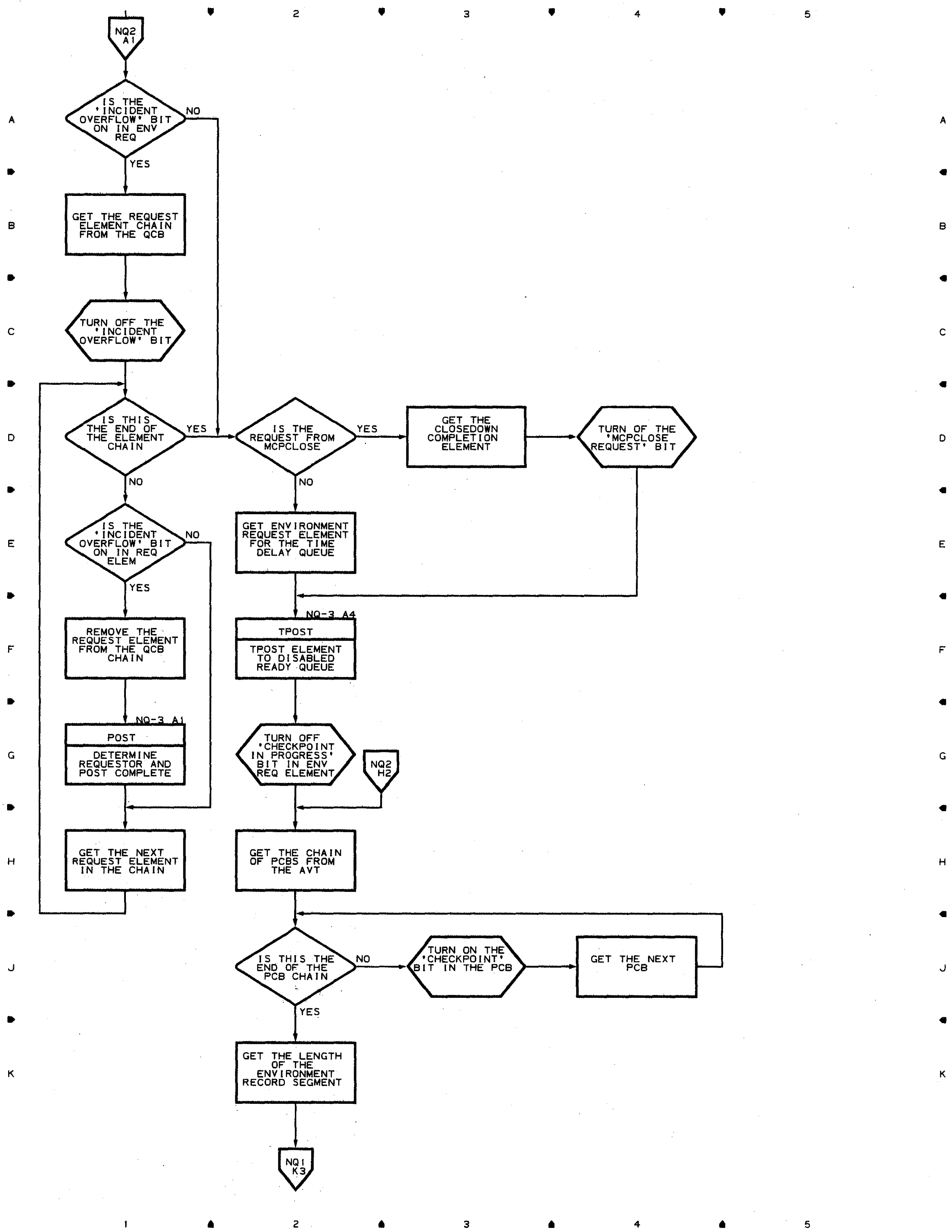

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
NQ2 A1
IS THE 'INCIDENT OVERFLOW' BIT ON IN ENV REQ
NO
YES
GET THE REQUEST ELEMENT CHAIN FROM THE QCB
TURN OFF THE 'INCIDENT OVERFLOW' BIT
IS THIS THE END OF THE ELEMENT CHAIN
YES
NO
IS THE REQUEST FROM MCPCLOSE
YES
NO
GET THE CLOSEDOWN COMPLETION ELEMENT
TURN OF THE 'MCPCLOSE REQUEST' BIT
IS THE 'INCIDENT OVERFLOW' BIT ON IN REQ ELEM
NO
YES
GET ENVIRONMENT REQUEST ELEMENT FOR THE TIME DELAY QUEUE
NQ-3 A4
TPOST
TPOST ELEMENT TO DISABLED READY QUEUE
REMOVE THE REQUEST ELEMENT FROM THE QCB CHAIN
NQ-3 A1
POST
DETERMINE REQUESTOR AND POST COMPLETE
TURN OFF 'CHECKPOINT IN PROGRESS' BIT IN ENV REQ ELEMENT
NQ2 H2
GET THE NEXT REQUEST ELEMENT IN THE CHAIN
GET THE CHAIN OF PCBS FROM THE AVT
IS THIS THE END OF THE PCB CHAIN
NO
YES
TURN ON THE 'CHECKPOINT' BIT IN THE PCB
GET THE NEXT PCB
GET THE LENGTH OF THE ENVIRONMENT RECORD SEGMENT
NQ1 K3

Chart NQ-3 CHECKPOINT NOTIFICATION AND DISPOSITION ROUTINE

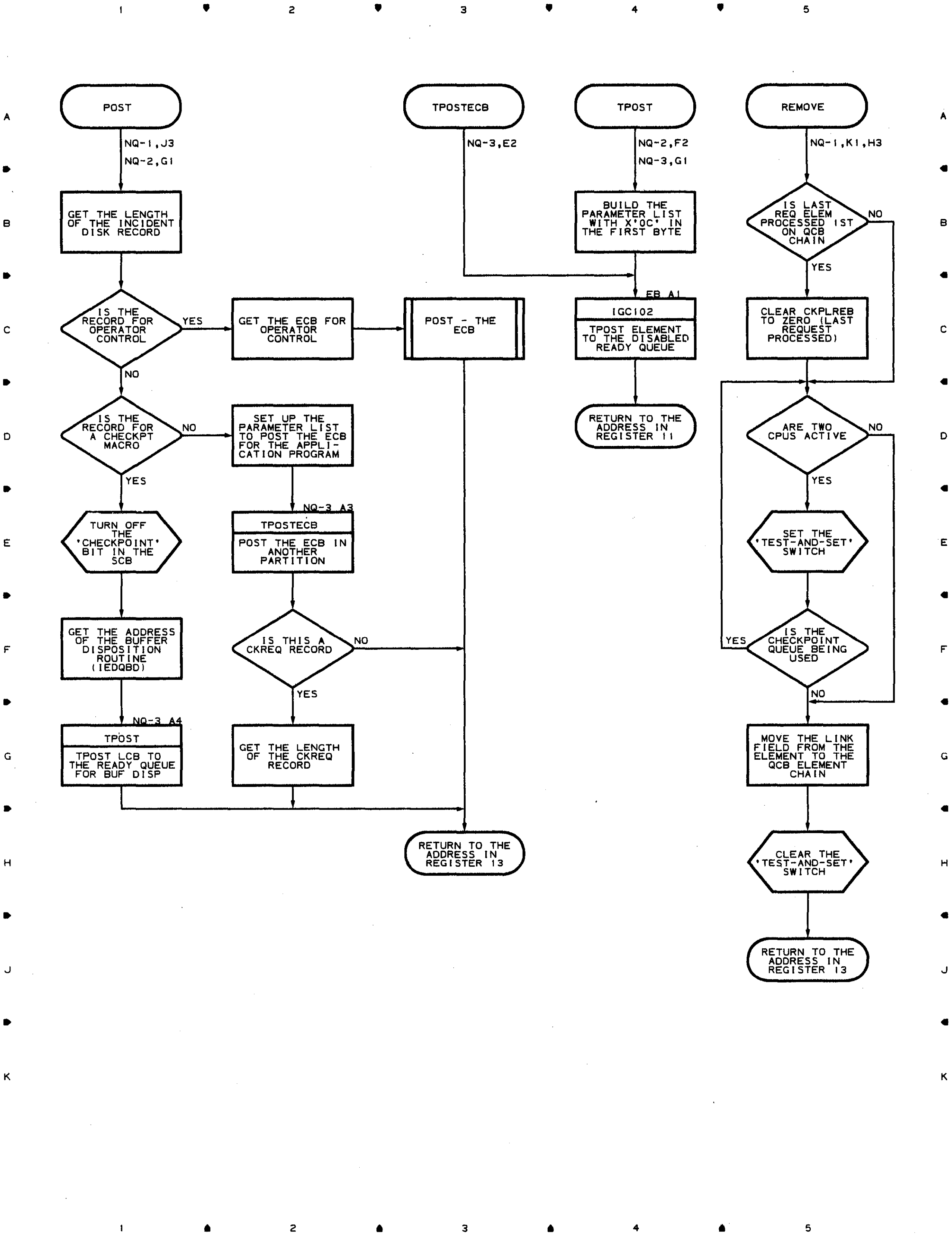
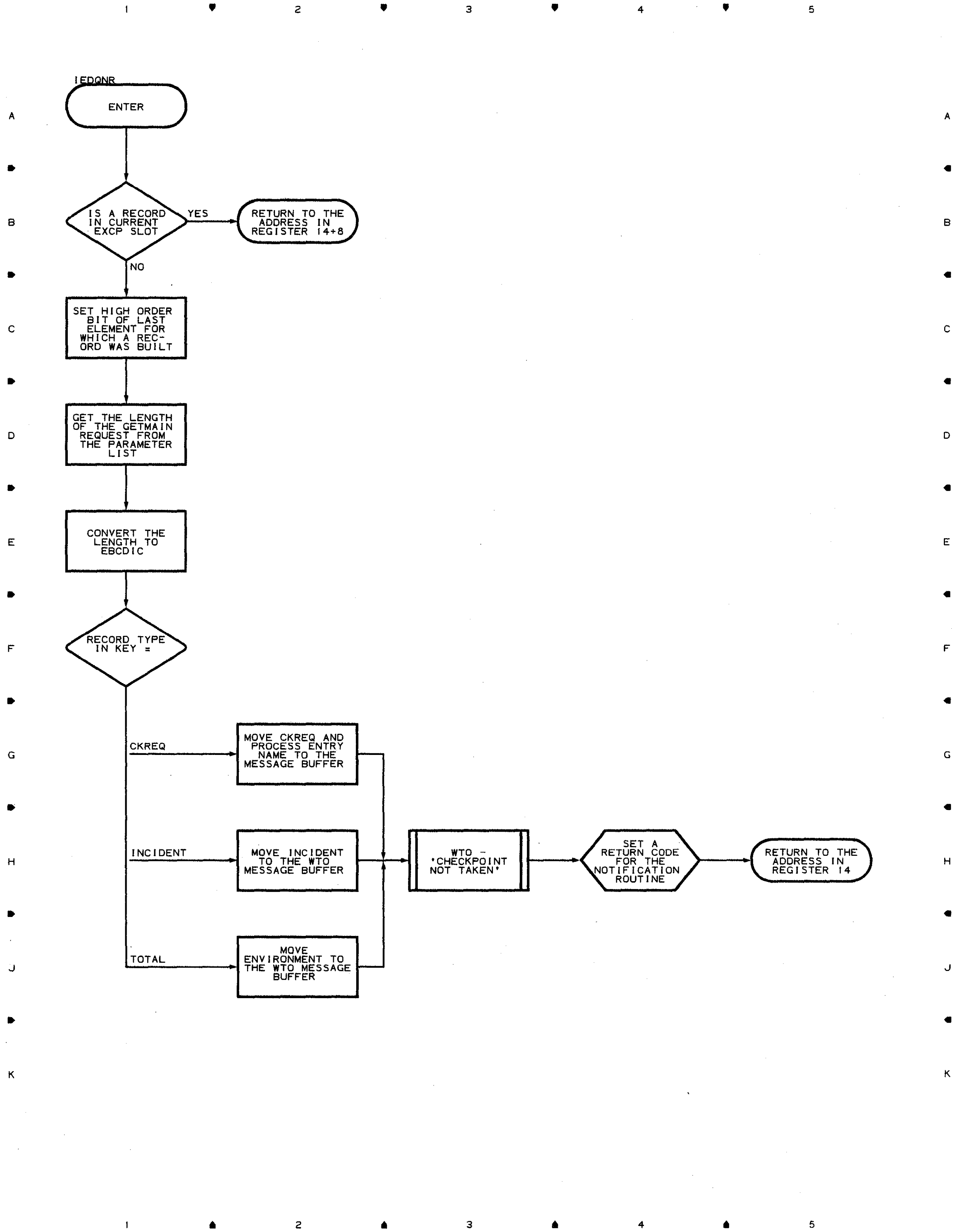
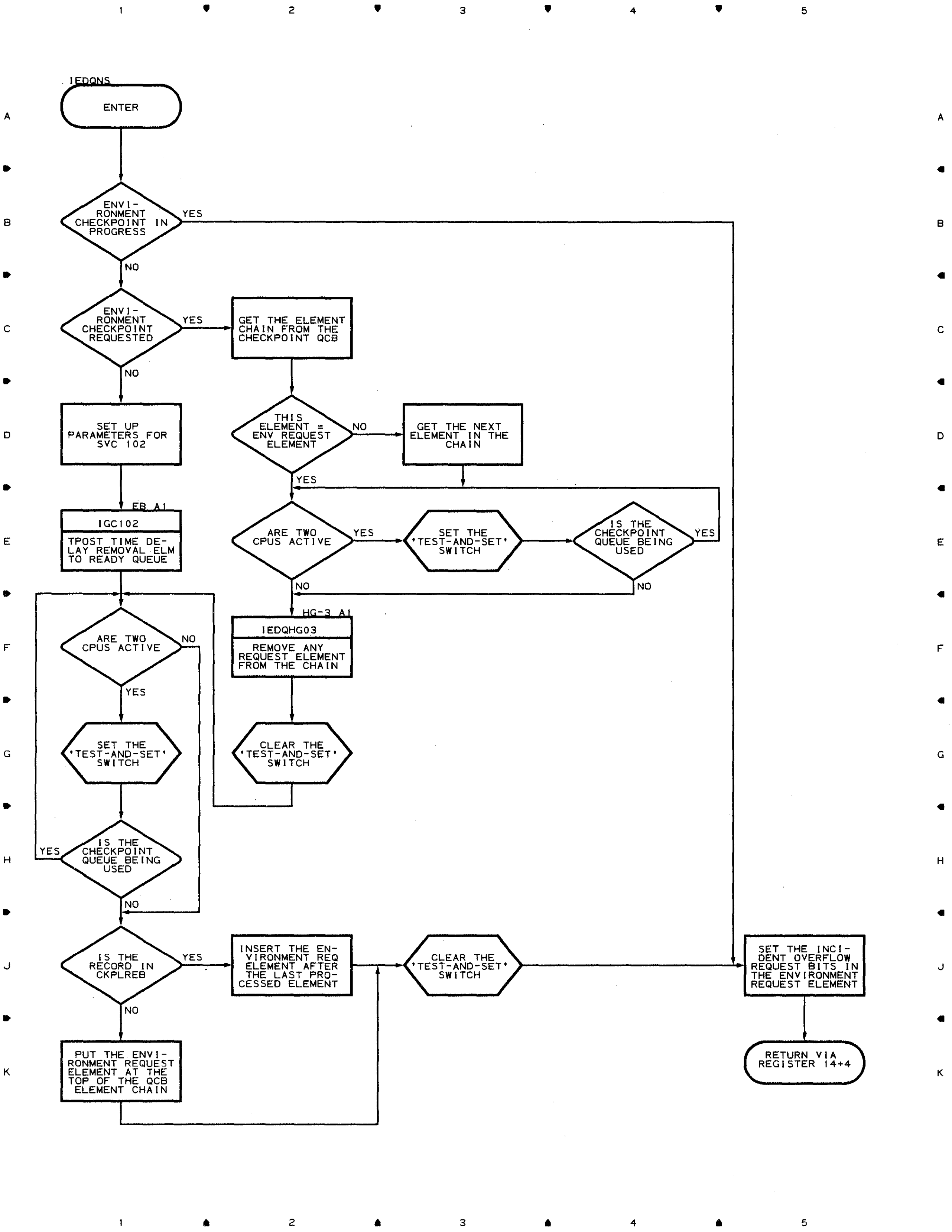

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
POST
NQ-1,J3
NQ-2,G1
GET THE LENGTH OF THE INCIDENT DISK RECORD
IS THE RECORD FOR OPERATOR CONTROL
YES
NO
GET THE ECB FOR OPERATOR CONTROL
POST - THE ECB
IS THE RECORD FOR A CHECKPT MACRO
NO
YES
SET UP THE PARAMETER LIST TO POST THE ECB FOR THE APPLI-CATION PROGRAM
NQ-3 A3
TPOSTECB
POST THE ECB IN ANOTHER PARTITION
TURN OFF THE 'CHECKPOINT' BIT IN THE SCB
GET THE ADDRESS OF THE BUFFER DISPOSITION ROUTINE (IEDQBD)
IS THIS A CKREQ RECORD
NO
YES
NQ-3 A4
TPOST
TPOST LCB TO THE READY QUEUE FOR BUF DISP
GET THE LENGTH OF THE CKREQ RECORD
RETURN TO THE ADDRESS IN REGISTER 13
TPOSTECB
NQ-3,E2
TPOST
NQ-2,F2
NQ-3,G1
BUILD THE PARAMETER LIST WITH X'0C' IN THE FIRST BYTE
EB A1
IGC102
TPOST ELEMENT TO THE DISABLED READY QUEUE
RETURN TO THE ADDRESS IN REGISTER 11
REMOVE
NQ-1,K1,H3
IS LAST REQ ELEM PROCESSED 1ST ON QCB CHAIN
NO
YES
CLEAR CKPLREB TO ZERO (LAST REQUEST PROCESSED)
ARE TWO CPUS ACTIVE
NO
YES
SET THE 'TEST-AND-SET' SWITCH
IS THE CHECKPOINT QUEUE BEING USED
YES
NO
MOVE THE LINK FIELD FROM THE ELEMENT TO THE QCB ELEMENT CHAIN
CLEAR THE 'TEST-AND-SET' SWITCH
RETURN TO THE ADDRESS IN REGISTER 13

Chart NR   CHECKPOINT - NO AVAILABLE CORE ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQNR
ENTER
IS A RECORD IN CURRENT EXCP SLOT
YES
RETURN TO THE ADDRESS IN REGISTER 14+8
NO
SET HIGH ORDER BIT OF LAST ELEMENT FOR WHICH A RECORD WAS BUILT
GET THE LENGTH OF THE GETMAIN REQUEST FROM THE PARAMETER LIST
CONVERT THE LENGTH TO EBCDIC
RECORD TYPE IN KEY =
CKREQ
MOVE CKREQ AND PROCESS ENTRY NAME TO THE MESSAGE BUFFER
INCIDENT
MOVE INCIDENT TO THE WTO MESSAGE BUFFER
TOTAL
MOVE ENVIRONMENT TO THE WTO MESSAGE BUFFER
WTO - 'CHECKPOINT NOT TAKEN'
SET A RETURN CODE FOR THE NOTIFICATION ROUTINE
RETURN TO THE ADDRESS IN REGISTER 14

Chart NS CHECKPOINT - NO INCIDENT RECORDS ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQNS
ENTER
ENVIRONMENT CHECKPOINT IN PROGRESS
YES
NO
ENVIRONMENT CHECKPOINT REQUESTED
YES
NO
GET THE ELEMENT CHAIN FROM THE CHECKPOINT QCB
SET UP PARAMETERS FOR SVC 102
THIS ELEMENT = ENV REQUEST ELEMENT
NO
GET THE NEXT ELEMENT IN THE CHAIN
YES
FB A1
IGC102
TPOST TIME DELAY REMOVAL ELM TO READY QUEUE
ARE TWO CPUS ACTIVE
YES
SET THE 'TEST-AND-SET' SWITCH
IS THE CHECKPOINT QUEUE BEING USED
YES
NO
NO
HG-3 A1
IEDQHG03
REMOVE ANY REQUEST ELEMENT FROM THE CHAIN
ARE TWO CPUS ACTIVE
NO
YES
SET THE 'TEST-AND-SET' SWITCH
CLEAR THE 'TEST-AND-SET' SWITCH
IS THE CHECKPOINT QUEUE BEING USED
YES
NO
IS THE RECORD IN CKPLREB
YES
NO
INSERT THE ENVIRONMENT REQ ELEMENT AFTER THE LAST PROCESSED ELEMENT
CLEAR THE 'TEST-AND-SET' SWITCH
SET THE INCIDENT OVERFLOW REQUEST BITS IN THE ENVIRONMENT REQUEST ELEMENT
RETURN VIA REGISTER 14+4
PUT THE ENVIRONMENT REQUEST ELEMENT AT THE TOP OF THE QCB ELEMENT CHAIN

Chart NT TERMNAME TABLE CODE

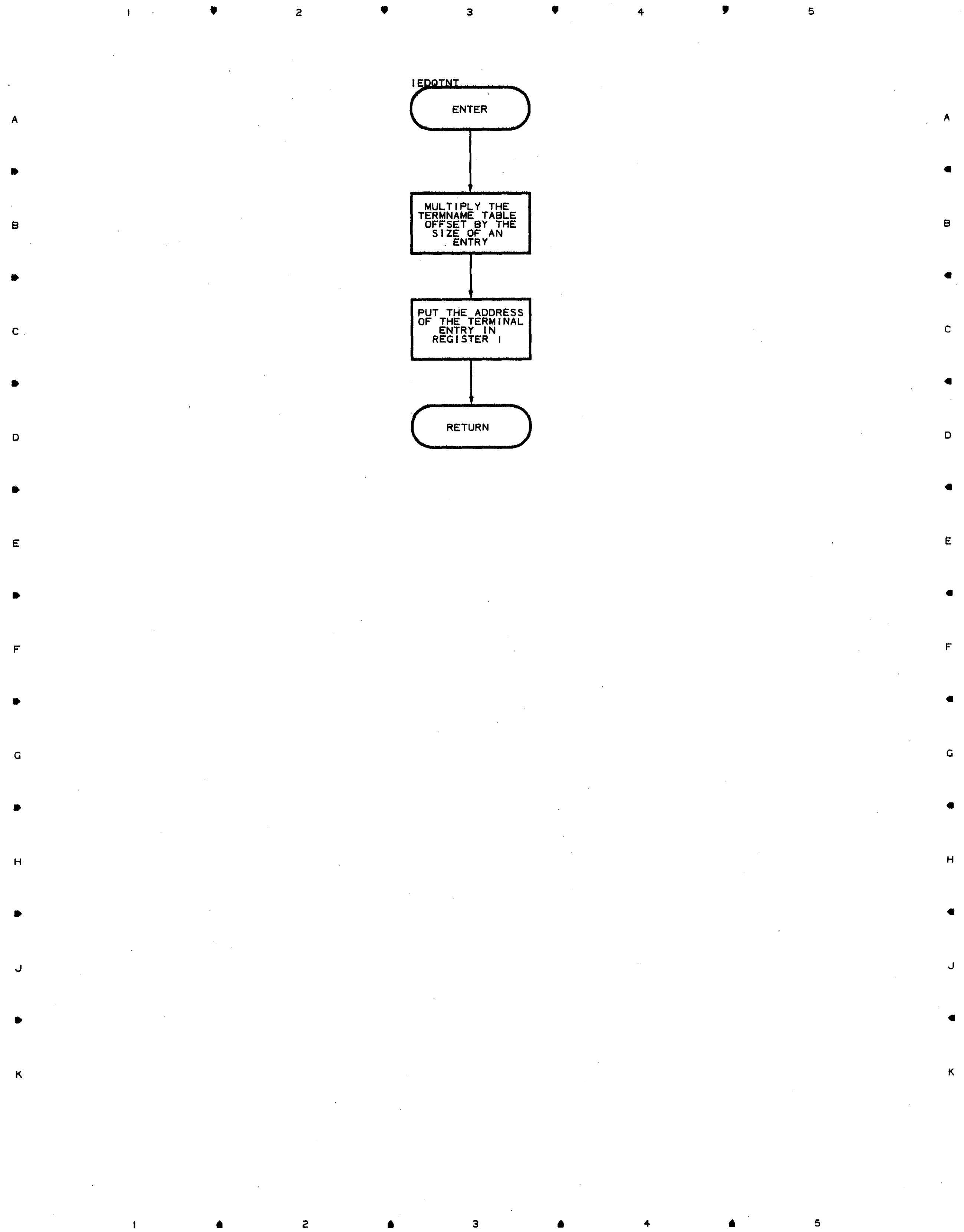

IEDQTNT
ENTER
MULTIPLY THE TERMNAME TABLE OFFSET BY THE SIZE OF AN ENTRY
PUT THE ADDRESS OF THE TERMINAL ENTRY IN REGISTER 1
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart NX-1 OPERATOR AWARENESS MESSAGE ROUTER

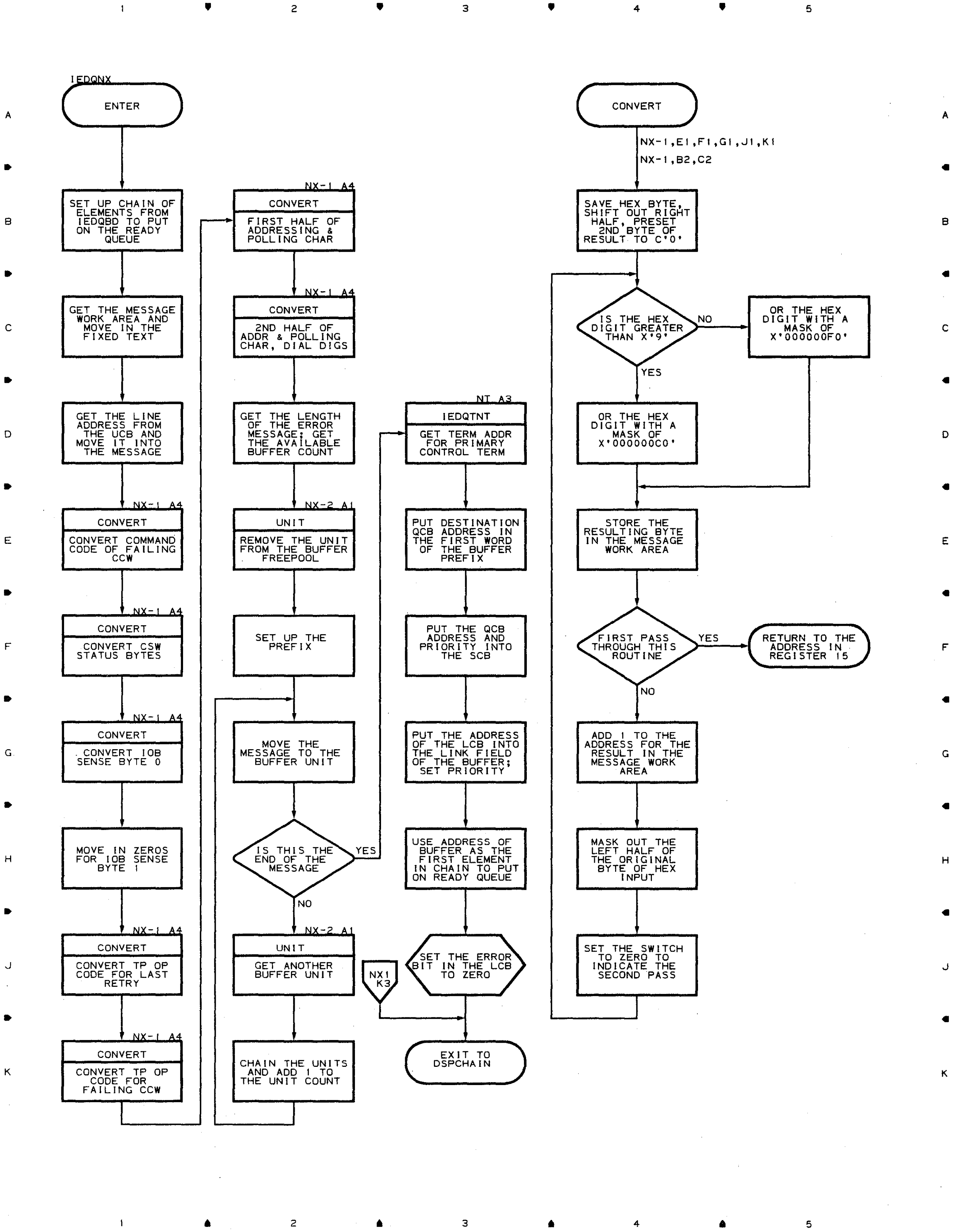
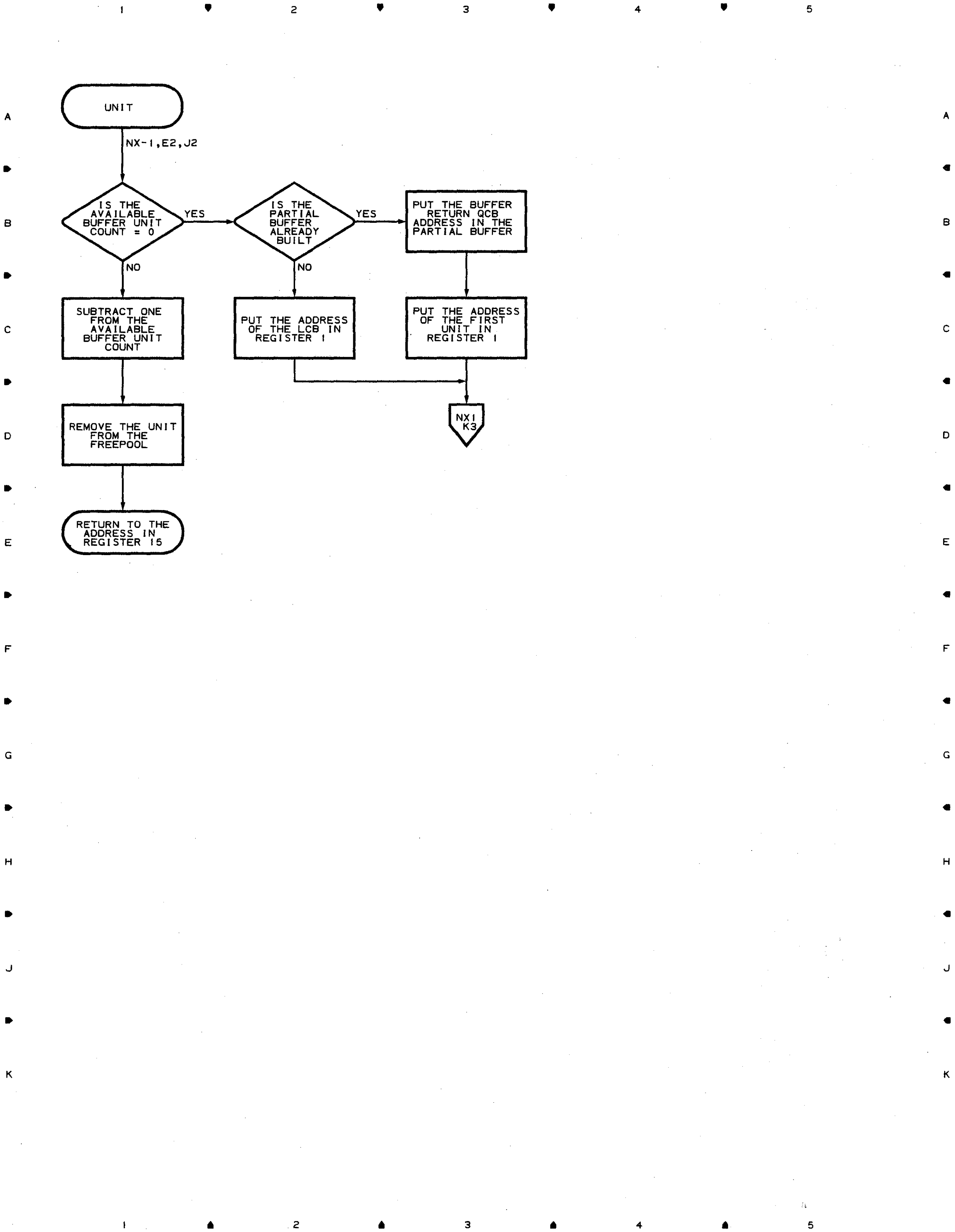

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQNX
ENTER
SET UP CHAIN OF ELEMENTS FROM IEDQBD TO PUT ON THE READY QUEUE
GET THE MESSAGE WORK AREA AND MOVE IN THE FIXED TEXT
GET THE LINE ADDRESS FROM THE UCB AND MOVE IT INTO THE MESSAGE
NX-1 A4
CONVERT
CONVERT COMMAND CODE OF FAILING CCW
NX-1 A4
CONVERT
CONVERT CSW STATUS BYTES
NX-1 A4
CONVERT
CONVERT IOB SENSE BYTE 0
MOVE IN ZEROS FOR IOB SENSE BYTE 1
NX-1 A4
CONVERT
CONVERT TP OP CODE FOR LAST RETRY
NX-1 A4
CONVERT
CONVERT TP OP CODE FOR FAILING CCW
NX-1 A4
CONVERT
FIRST HALF OF ADDRESSING & POLLING CHAR
NX-1 A4
CONVERT
2ND HALF OF ADDR & POLLING CHAR, DIAL DIGS
GET THE LENGTH OF THE ERROR MESSAGE; GET THE AVAILABLE BUFFER COUNT
NX-2 A1
UNIT
REMOVE THE UNIT FROM THE BUFFER FREEPOOL
SET UP THE PREFIX
MOVE THE MESSAGE TO THE BUFFER UNIT
IS THIS THE END OF THE MESSAGE
YES
NO
NX-2 A1
UNIT
GET ANOTHER BUFFER UNIT
CHAIN THE UNITS AND ADD 1 TO THE UNIT COUNT
NT A3
IEDQTNT
GET TERM ADDR FOR PRIMARY CONTROL TERM
PUT DESTINATION QCB ADDRESS IN THE FIRST WORD OF THE BUFFER PREFIX
PUT THE QCB ADDRESS AND PRIORITY INTO THE SCB
PUT THE ADDRESS OF THE LCB INTO THE LINK FIELD OF THE BUFFER; SET PRIORITY
USE ADDRESS OF BUFFER AS THE FIRST ELEMENT IN CHAIN TO PUT ON READY QUEUE
SET THE ERROR BIT IN THE LCB TO ZERO
NX1 K3
EXIT TO DSPCHAIN
CONVERT
NX-1,E1,F1,G1,J1,K1
NX-1,B2,C2
SAVE HEX BYTE, SHIFT OUT RIGHT HALF, PRESET 2ND BYTE OF RESULT TO C'0'
IS THE HEX DIGIT GREATER THAN X'9'
NO
YES
OR THE HEX DIGIT WITH A MASK OF X'000000F0'
OR THE HEX DIGIT WITH A MASK OF X'000000C0'
STORE THE RESULTING BYTE IN THE MESSAGE WORK AREA
FIRST PASS THROUGH THIS ROUTINE
YES
NO
RETURN TO THE ADDRESS IN REGISTER 15
ADD 1 TO THE ADDRESS FOR THE RESULT IN THE MESSAGE WORK AREA
MASK OUT THE LEFT HALF OF THE ORIGINAL BYTE OF HEX INPUT
SET THE SWITCH TO ZERO TO INDICATE THE SECOND PASS

Chart NX-2 OPERATOR AWARENESS MESSAGE ROUTER



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
UNIT
NX-1,E2,J2
IS THE AVAILABLE BUFFER UNIT COUNT = 0
YES
NO
IS THE PARTIAL BUFFER ALREADY BUILT
YES
NO
PUT THE BUFFER RETURN QCB ADDRESS IN THE PARTIAL BUFFER
SUBTRACT ONE FROM THE AVAILABLE BUFFER UNIT COUNT
PUT THE ADDRESS OF THE LCB IN REGISTER 1
PUT THE ADDRESS OF THE FIRST UNIT IN REGISTER 1
REMOVE THE UNIT FROM THE FREEPOOL
NX1 K3
RETURN TO THE ADDRESS IN REGISTER 15

Chart NZ TCAM COMMAND SCHEDULER - SVC 34

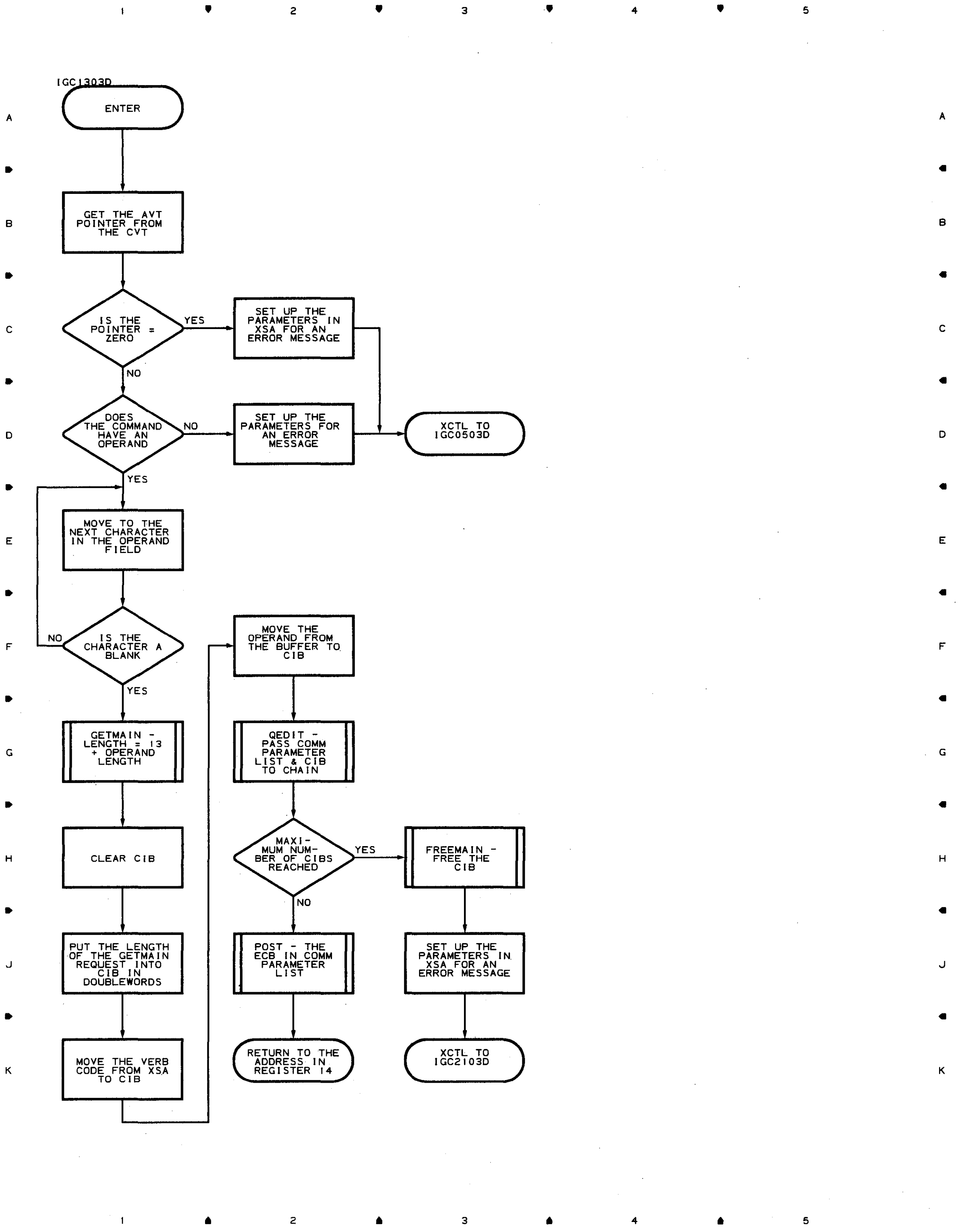

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGC1303D
ENTER
GET THE AVT POINTER FROM THE CVT
IS THE POINTER = ZERO
YES
SET UP THE PARAMETERS IN XSA FOR AN ERROR MESSAGE
NO
DOES THE COMMAND HAVE AN OPERAND
NO
SET UP THE PARAMETERS FOR AN ERROR MESSAGE
XCTL TO IGC0503D
YES
MOVE TO THE NEXT CHARACTER IN THE OPERAND FIELD
NO
IS THE CHARACTER A BLANK
YES
MOVE THE OPERAND FROM THE BUFFER TO CIB
GETMAIN - LENGTH = 13 + OPERAND LENGTH
QEDIT - PASS COMM PARAMETER LIST & CIB TO CHAIN
CLEAR CIB
MAXIMUM NUMBER OF CIBS REACHED
YES
FREEMAIN - FREE THE CIB
NO
PUT THE LENGTH OF THE GETMAIN REQUEST INTO CIB IN DOUBLEWORDS
POST - THE ECB IN COMM PARAMETER LIST
SET UP THE PARAMETERS IN XSA FOR AN ERROR MESSAGE
MOVE THE VERB CODE FROM XSA TO CIB
RETURN TO THE ADDRESS IN REGISTER 14
XCTL TO IGC2103D

Chart OA LINK ROUTINE

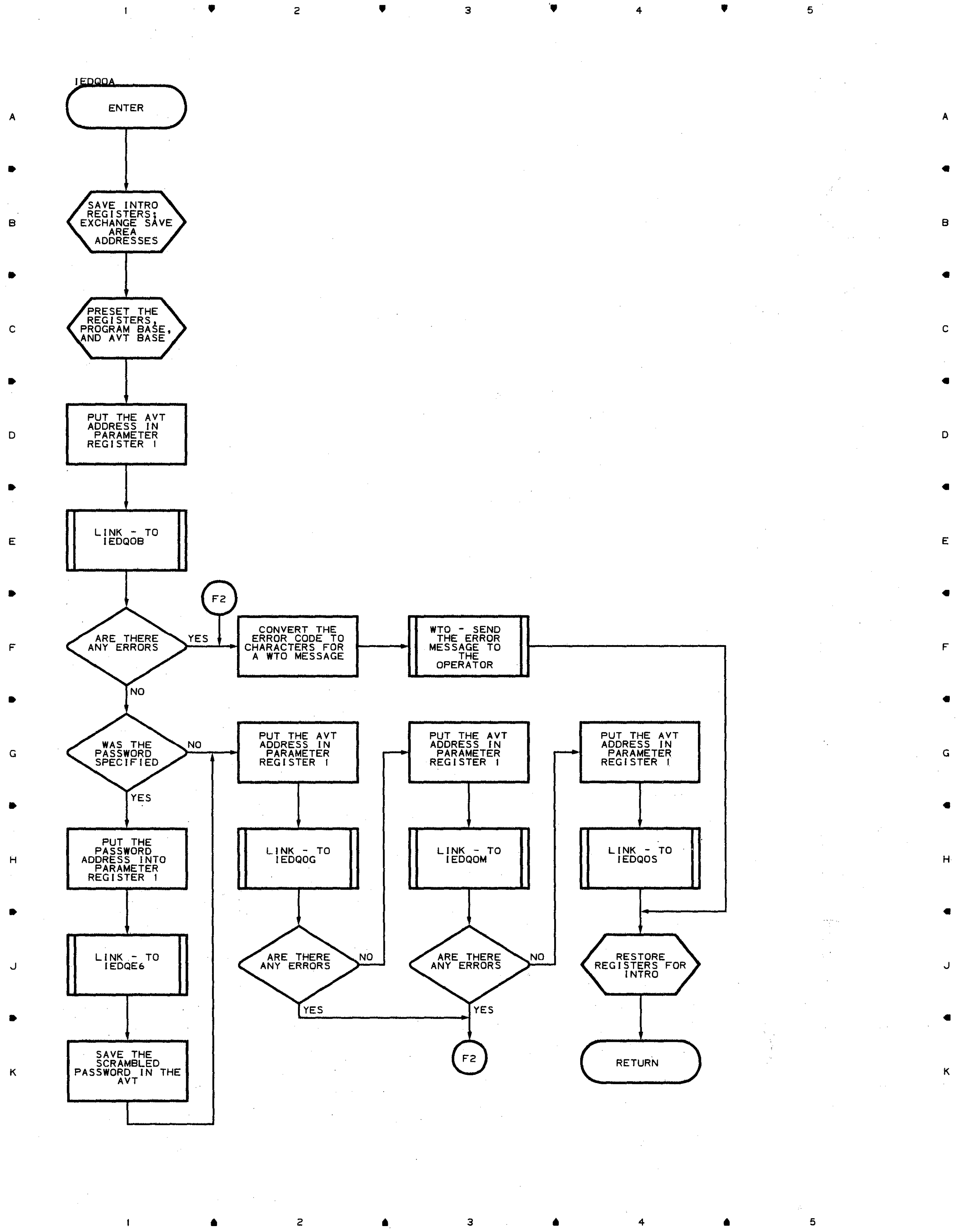

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQOA
ENTER
SAVE INTRO REGISTERS; EXCHANGE SAVE AREA ADDRESSES
PRESET THE REGISTERS, PROGRAM BASE, AND AVT BASE
PUT THE AVT ADDRESS IN PARAMETER REGISTER 1
LINK - TO IEDQOB
F2
ARE THERE ANY ERRORS
YES
NO
CONVERT THE ERROR CODE TO CHARACTERS FOR A WTO MESSAGE
WTO - SEND THE ERROR MESSAGE TO THE OPERATOR
WAS THE PASSWORD SPECIFIED
NO
YES
PUT THE AVT ADDRESS IN PARAMETER REGISTER 1
PUT THE AVT ADDRESS IN PARAMETER REGISTER 1
PUT THE AVT ADDRESS IN PARAMETER REGISTER 1
PUT THE PASSWORD ADDRESS INTO PARAMETER REGISTER 1
LINK - TO IEDQOG
LINK - TO IEDQOM
LINK - TO IEDQOS
LINK - TO IEDQE6
ARE THERE ANY ERRORS
NO
YES
ARE THERE ANY ERRORS
NO
YES
RESTORE REGISTERS FOR INTRO
SAVE THE SCRAMBLED PASSWORD IN THE AVT
F2
RETURN

Chart OB-1 WTOR INTERPRETER ROUTINE

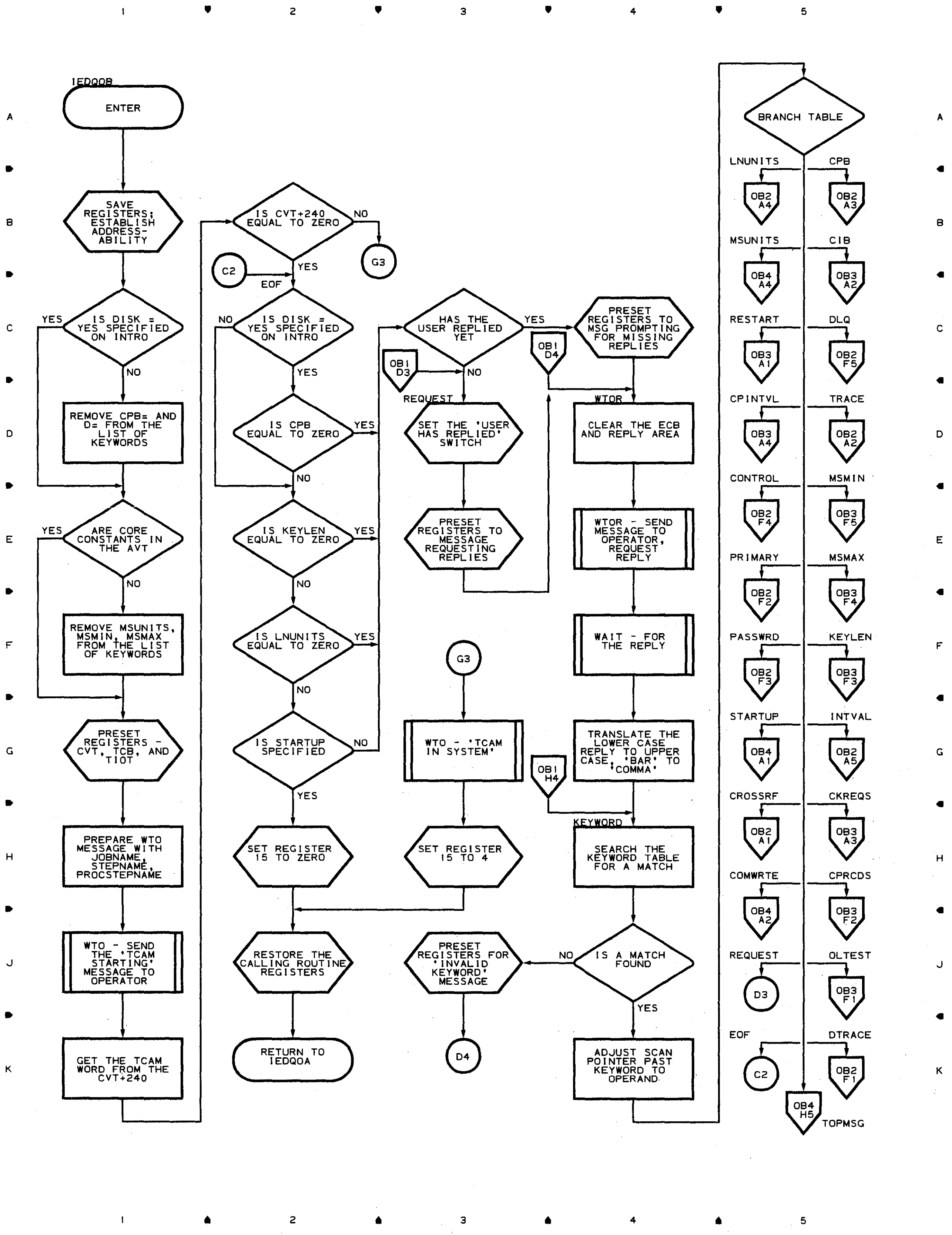

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQOB
ENTER
SAVE REGISTERS; ESTABLISH ADDRESS-ABILITY
IS DISK = YES SPECIFIED ON INTRO
YES
NO
REMOVE CPB= AND D= FROM THE LIST OF KEYWORDS
ARE CORE CONSTANTS IN THE AVT
YES
NO
REMOVE MSUNITS, MSMIN, MSMAX FROM THE LIST OF KEYWORDS
PRESET REGISTERS - CVT, TCB, AND TIOT
PREPARE WTO MESSAGE WITH JOBNAME, STEPNAME, PROCSTEPNAME
WTO - SEND THE 'TCAM STARTING' MESSAGE TO OPERATOR
GET THE TCAM WORD FROM THE CVT+240
IS CVT+240 EQUAL TO ZERO
NO
G3
YES
C2
EOF
IS DISK = YES SPECIFIED ON INTRO
NO
YES
IS CPB EQUAL TO ZERO
YES
NO
IS KEYLEN EQUAL TO ZERO
YES
NO
IS LNUNITS EQUAL TO ZERO
YES
NO
IS STARTUP SPECIFIED
NO
YES
SET REGISTER 15 TO ZERO
RESTORE THE CALLING ROUTINE REGISTERS
RETURN TO IEDQOA
HAS THE USER REPLIED YET
YES
NO
OB1 D3
REQUEST
SET THE 'USER HAS REPLIED' SWITCH
PRESET REGISTERS TO MESSAGE REQUESTING REPLIES
G3
WTO - 'TCAM IN SYSTEM'
SET REGISTER 15 TO 4
PRESET REGISTERS FOR 'INVALID KEYWORD' MESSAGE
D4
PRESET REGISTERS TO MSG PROMPTING FOR MISSING REPLIES
OB1 D4
WTOR
CLEAR THE ECB AND REPLY AREA
WTOR - SEND MESSAGE TO OPERATOR, REQUEST REPLY
WAIT - FOR THE REPLY
TRANSLATE THE LOWER CASE REPLY TO UPPER CASE, 'BAR' TO 'COMMA'
OB1 H4
KEYWORD
SEARCH THE KEYWORD TABLE FOR A MATCH
IS A MATCH FOUND
NO
YES
ADJUST SCAN POINTER PAST KEYWORD TO OPERAND
BRANCH TABLE
LNUNITS
OB2 A4
CPB
OB2 A3
MSUNITS
OB4 A4
CIB
OB3 A2
RESTART
OB3 A1
DLQ
OB2 F5
CPINTVL
OB3 A4
TRACE
OB2 A2
CONTROL
OB2 F4
MSMIN
OB3 F5
PRIMARY
OB2 F2
MSMAX
OB3 F4
PASSWRD
OB2 F3
KEYLEN
OB3 F3
STARTUP
OB4 A1
INTVAL
OB2 A5
CROSSRF
OB2 A1
CKREQS
OB3 A3
COMWRTE
OB4 A2
CPRCDS
OB3 F2
REQUEST
D3
OLTEST
OB3 F1
EOF
C2
DTRACE
OB2 F1
OB4 H5
TOPMSG

Chart OB-2 WTOR INTERPRETER ROUTINE

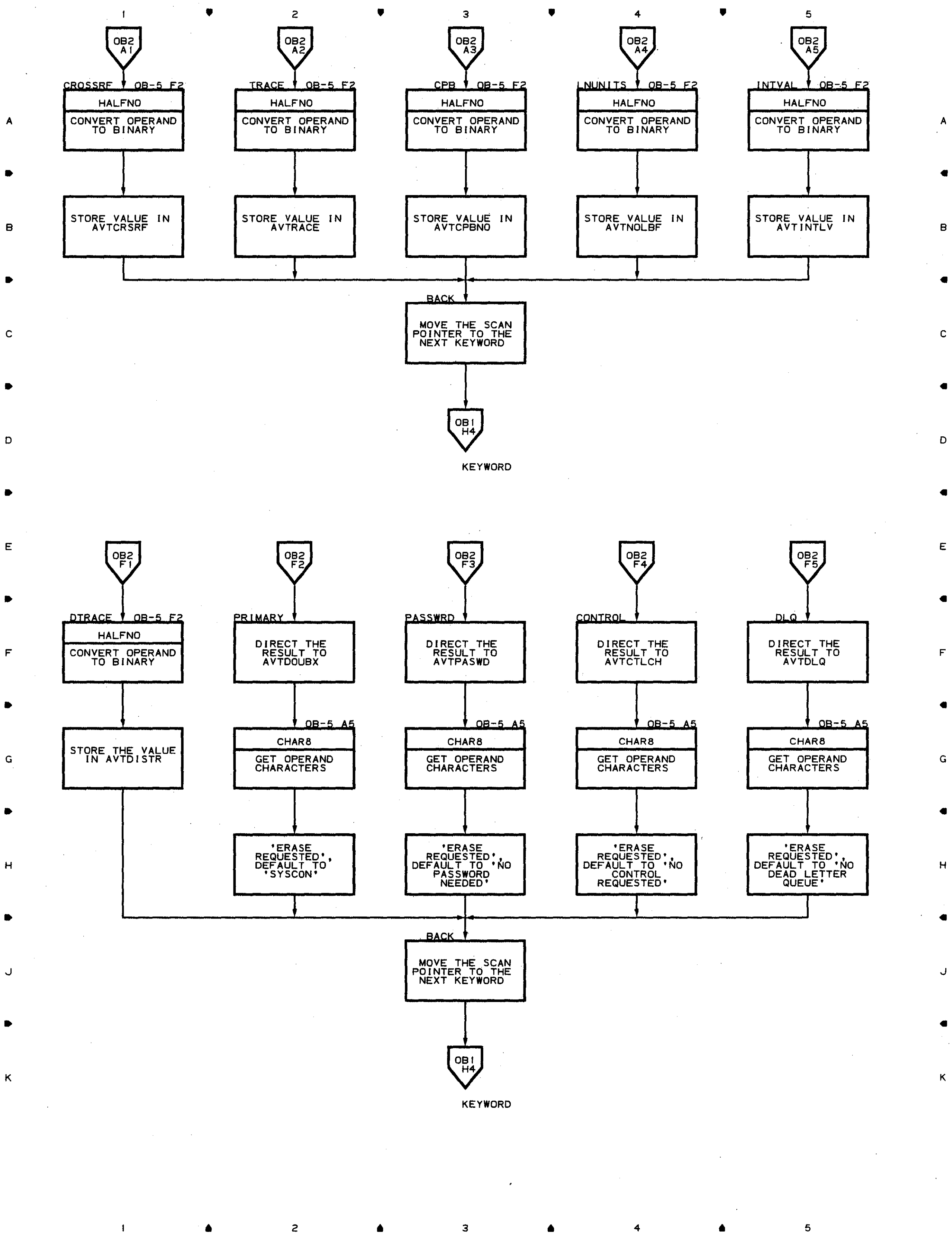

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
OB2 A1
CROSSRF OB-5 F2
HALFNO
CONVERT OPERAND TO BINARY
STORE VALUE IN AVTCRSRF
OB2 A2
TRACE OB-5 F2
HALFNO
CONVERT OPERAND TO BINARY
STORE VALUE IN AVTRACE
OB2 A3
CPB OB-5 F2
HALFNO
CONVERT OPERAND TO BINARY
STORE VALUE IN AVTCPBNO
OB2 A4
LNUNITS OB-5 F2
HALFNO
CONVERT OPERAND TO BINARY
STORE VALUE IN AVTNOLBF
OB2 A5
INTVAL OB-5 F2
HALFNO
CONVERT OPERAND TO BINARY
STORE VALUE IN AVTINTLV
BACK
MOVE THE SCAN POINTER TO THE NEXT KEYWORD
OB1 H4
KEYWORD
OB2 F1
DTRACE OB-5 F2
HALFNO
CONVERT OPERAND TO BINARY
STORE THE VALUE IN AVTDISTR
OB2 F2
PRIMARY
DIRECT THE RESULT TO AVTDOUBX
OB-5 A5
CHAR8
GET OPERAND CHARACTERS
'ERASE REQUESTED', DEFAULT TO 'SYSCON'
OB2 F3
PASSWRD
DIRECT THE RESULT TO AVTPASWD
OB-5 A5
CHAR8
GET OPERAND CHARACTERS
'ERASE REQUESTED', DEFAULT TO 'NO PASSWORD NEEDED'
OB2 F4
CONTROL
DIRECT THE RESULT TO AVTCTLCH
OB-5 A5
CHAR8
GET OPERAND CHARACTERS
'ERASE REQUESTED', DEFAULT TO 'NO CONTROL REQUESTED'
OB2 F5
DLQ
DIRECT THE RESULT TO AVTDLQ
OB-5 A5
CHAR8
GET OPERAND CHARACTERS
'ERASE REQUESTED', DEFAULT TO 'NO DEAD LETTER QUEUE'
BACK
MOVE THE SCAN POINTER TO THE NEXT KEYWORD
OB1 H4
KEYWORD

Chart OB-3 WTOR INTERPRETER ROUTINE

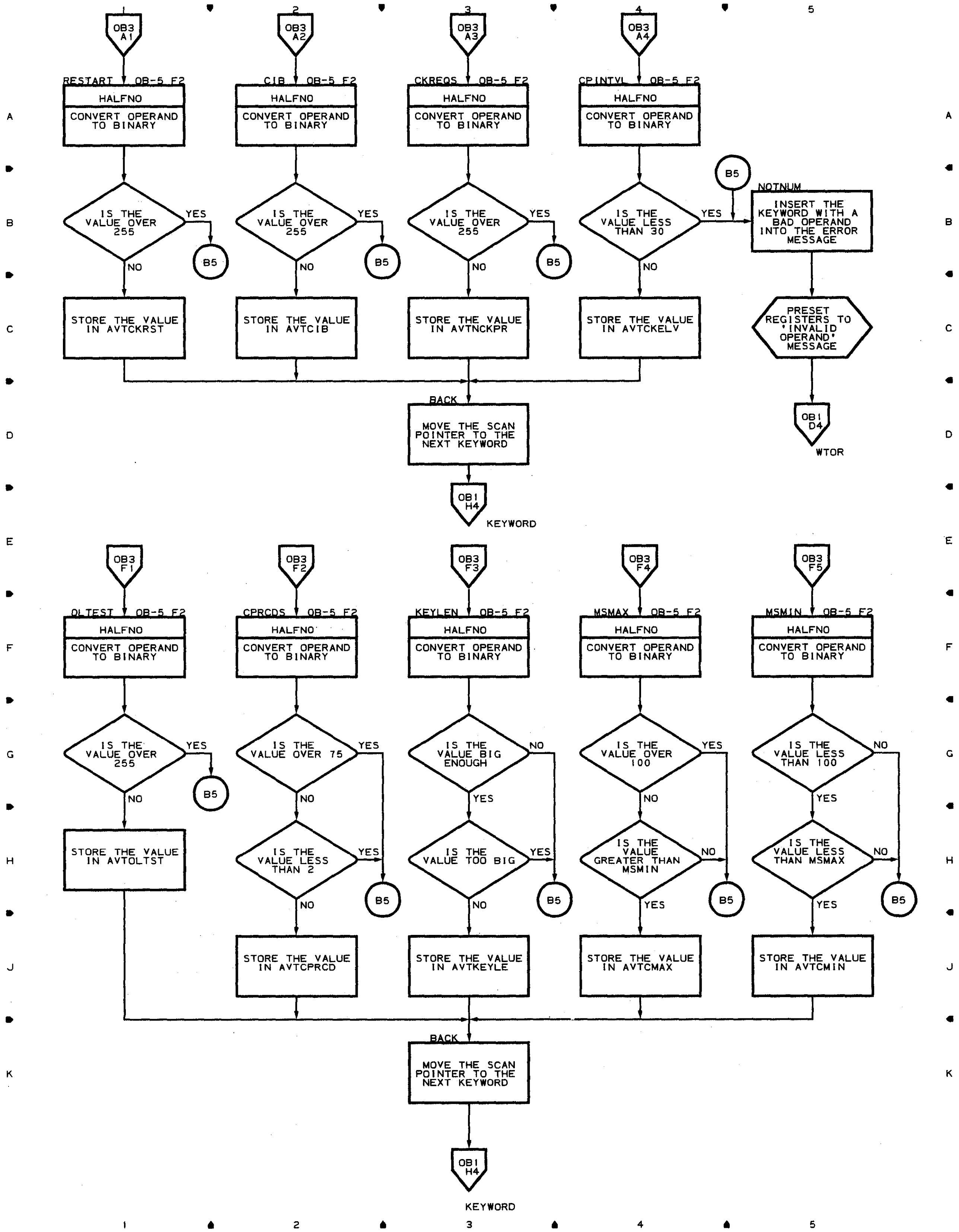

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
OB3 A1
RESTART OB-5 F2
HALFNO
CONVERT OPERAND TO BINARY
IS THE VALUE OVER 255
YES
B5
NO
STORE THE VALUE IN AVTCKRST
OB3 A2
CIB OB-5 F2
HALFNO
CONVERT OPERAND TO BINARY
IS THE VALUE OVER 255
YES
B5
NO
STORE THE VALUE IN AVTCIB
OB3 A3
CKREQS OB-5 F2
HALFNO
CONVERT OPERAND TO BINARY
IS THE VALUE OVER 255
YES
B5
NO
STORE THE VALUE IN AVTNCKPR
OB3 A4
CPINTVL OB-5 F2
HALFNO
CONVERT OPERAND TO BINARY
IS THE VALUE LESS THAN 30
YES
NO
STORE THE VALUE IN AVTCKELV
B5
NOTNUM
INSERT THE KEYWORD WITH A BAD OPERAND INTO THE ERROR MESSAGE
PRESET REGISTERS TO 'INVALID OPERAND' MESSAGE
OB1 D4
WTOR
BACK
MOVE THE SCAN POINTER TO THE NEXT KEYWORD
OB1 H4
KEYWORD
OB3 F1
OLTEST OB-5 F2
HALFNO
CONVERT OPERAND TO BINARY
IS THE VALUE OVER 255
YES
B5
NO
STORE THE VALUE IN AVTOLTST
OB3 F2
CPRCDS OB-5 F2
HALFNO
CONVERT OPERAND TO BINARY
IS THE VALUE OVER 75
YES
NO
IS THE VALUE LESS THAN 2
YES
B5
NO
STORE THE VALUE IN AVTCPRCD
OB3 F3
KEYLEN OB-5 F2
HALFNO
CONVERT OPERAND TO BINARY
IS THE VALUE BIG ENOUGH
NO
YES
IS THE VALUE TOO BIG
YES
B5
NO
STORE THE VALUE IN AVTKEYLE
OB3 F4
MSMAX OB-5 F2
HALFNO
CONVERT OPERAND TO BINARY
IS THE VALUE OVER 100
YES
NO
IS THE VALUE GREATER THAN MSMIN
NO
B5
YES
STORE THE VALUE IN AVTCMAX
OB3 F5
MSMIN OB-5 F2
HALFNO
CONVERT OPERAND TO BINARY
IS THE VALUE LESS THAN 100
NO
YES
IS THE VALUE LESS THAN MSMAX
NO
B5
YES
STORE THE VALUE IN AVTCMIN
BACK
MOVE THE SCAN POINTER TO THE NEXT KEYWORD
OB1 H4
KEYWORD

Chart OB-4 WTOR INTERPRETER ROUTINE

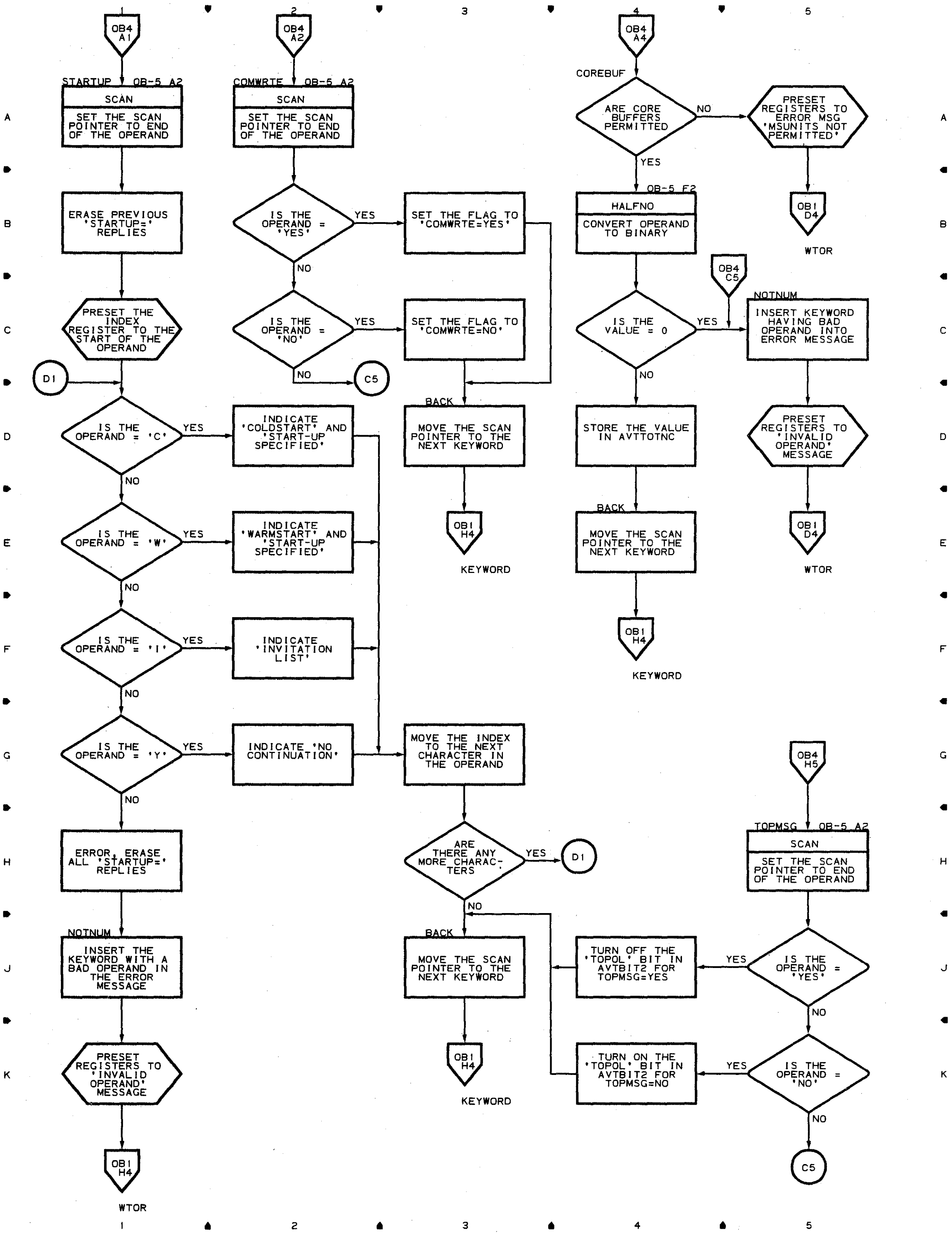

1
2
3
4
5
OB4 A1
OB4 A2
OB4 A4
STARTUP OB-5 A2
SCAN
SET THE SCAN POINTER TO END OF THE OPERAND
COMWRTE OB-5 A2
SCAN
SET THE SCAN POINTER TO END OF THE OPERAND
COREBUF
ARE CORE BUFFERS PERMITTED
NO
PRESET REGISTERS TO ERROR MSG 'MSUNITS NOT PERMITTED'
A
YES
ERASE PREVIOUS 'STARTUP=' REPLIES
IS THE OPERAND = 'YES'
YES
SET THE FLAG TO 'COMWRTE=YES'
OB-5 F2
HALFNO
CONVERT OPERAND TO BINARY
OB1 D4
WTOR
B
NO
OB4 C5
NOTNUM
PRESET THE INDEX REGISTER TO THE START OF THE OPERAND
IS THE OPERAND = 'NO'
YES
SET THE FLAG TO 'COMWRTE=NO'
IS THE VALUE = 0
YES
INSERT KEYWORD HAVING BAD OPERAND INTO ERROR MESSAGE
C
D1
NO
C5
NO
BACK
IS THE OPERAND = 'C'
YES
INDICATE 'COLDSTART' AND 'START-UP SPECIFIED'
MOVE THE SCAN POINTER TO THE NEXT KEYWORD
STORE THE VALUE IN AVTTOTNC
PRESET REGISTERS TO 'INVALID OPERAND' MESSAGE
D
NO
BACK
IS THE OPERAND = 'W'
YES
INDICATE 'WARMSTART' AND 'START-UP SPECIFIED'
OB1 H4
MOVE THE SCAN POINTER TO THE NEXT KEYWORD
OB1 D4
E
KEYWORD
WTOR
NO
IS THE OPERAND = 'I'
YES
INDICATE 'INVITATION LIST'
OB1 H4
F
KEYWORD
NO
IS THE OPERAND = 'Y'
YES
INDICATE 'NO CONTINUATION'
MOVE THE INDEX TO THE NEXT CHARACTER IN THE OPERAND
OB4 H5
G
NO
TOPMSG OB-5 A2
ERROR, ERASE ALL 'STARTUP=' REPLIES
ARE THERE ANY MORE CHARACTERS
YES
D1
SCAN
SET THE SCAN POINTER TO END OF THE OPERAND
H
NO
NOTNUM
BACK
INSERT THE KEYWORD WITH A BAD OPERAND IN THE ERROR MESSAGE
MOVE THE SCAN POINTER TO THE NEXT KEYWORD
TURN OFF THE 'TOPOL' BIT IN AVTBIT2 FOR TOPMSG=YES
YES
IS THE OPERAND = 'YES'
J
NO
PRESET REGISTERS TO 'INVALID OPERAND' MESSAGE
OB1 H4
TURN ON THE 'TOPOL' BIT IN AVTBIT2 FOR TOPMSG=NO
YES
IS THE OPERAND = 'NO'
K
KEYWORD
NO
OB1 H4
C5
WTOR
1
2
3
4
5

Chart OB-5 WTOR INTERPRETER ROUTINE

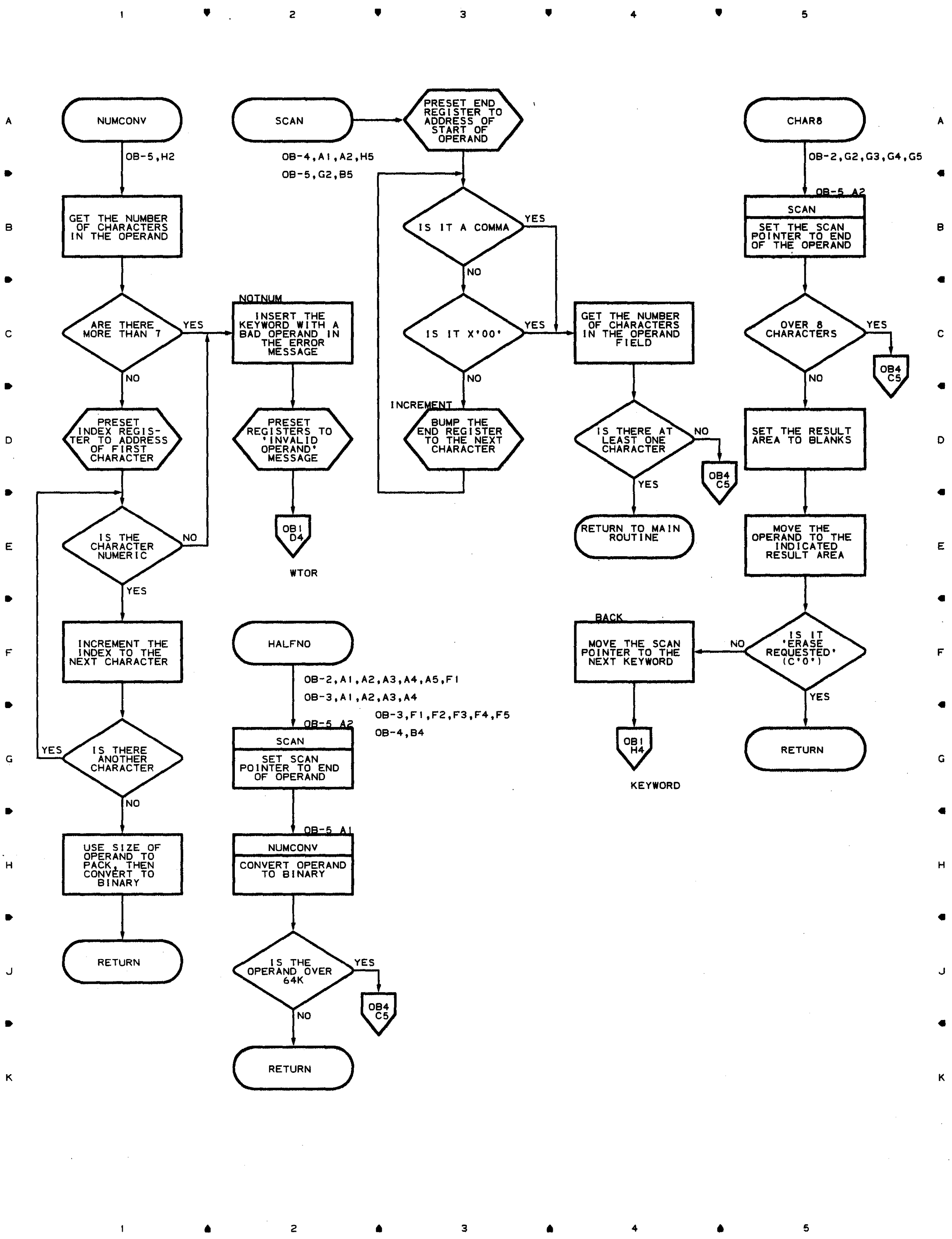

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
NUMCONV
OB-5,H2
GET THE NUMBER OF CHARACTERS IN THE OPERAND
ARE THERE MORE THAN 7
YES
NO
PRESET INDEX REGIS-TER TO ADDRESS OF FIRST CHARACTER
IS THE CHARACTER NUMERIC
NO
YES
INCREMENT THE INDEX TO THE NEXT CHARACTER
YES
IS THERE ANOTHER CHARACTER
NO
USE SIZE OF OPERAND TO PACK, THEN CONVERT TO BINARY
RETURN
SCAN
OB-4,A1,A2,H5
OB-5,G2,B5
NOTNUM
INSERT THE KEYWORD WITH A BAD OPERAND IN THE ERROR MESSAGE
PRESET REGISTERS TO 'INVALID OPERAND' MESSAGE
OB1 D4
WTOR
HALFNO
OB-2,A1,A2,A3,A4,A5,F1
OB-3,A1,A2,A3,A4
OB-3,F1,F2,F3,F4,F5
OB-4,B4
OB-5 A2
SCAN
SET SCAN POINTER TO END OF OPERAND
OB-5 A1
NUMCONV
CONVERT OPERAND TO BINARY
IS THE OPERAND OVER 64K
YES
OB4 C5
NO
RETURN
PRESET END REGISTER TO ADDRESS OF START OF OPERAND
IS IT A COMMA
YES
NO
IS IT X'00'
YES
NO
INCREMENT
BUMP THE END REGISTER TO THE NEXT CHARACTER
GET THE NUMBER OF CHARACTERS IN THE OPERAND FIELD
IS THERE AT LEAST ONE CHARACTER
NO
OB4 C5
YES
RETURN TO MAIN ROUTINE
BACK
MOVE THE SCAN POINTER TO THE NEXT KEYWORD
OB1 H4
KEYWORD
CHAR8
OB-2,G2,G3,G4,G5
OB-5 A2
SCAN
SET THE SCAN POINTER TO END OF THE OPERAND
OVER 8 CHARACTERS
YES
OB4 C5
NO
SET THE RESULT AREA TO BLANKS
MOVE THE OPERAND TO THE INDICATED RESULT AREA
IS IT 'ERASE REQUESTED' (C'0')
NO
YES
RETURN

Chart OG INTRO GETMAIN ROUTINE

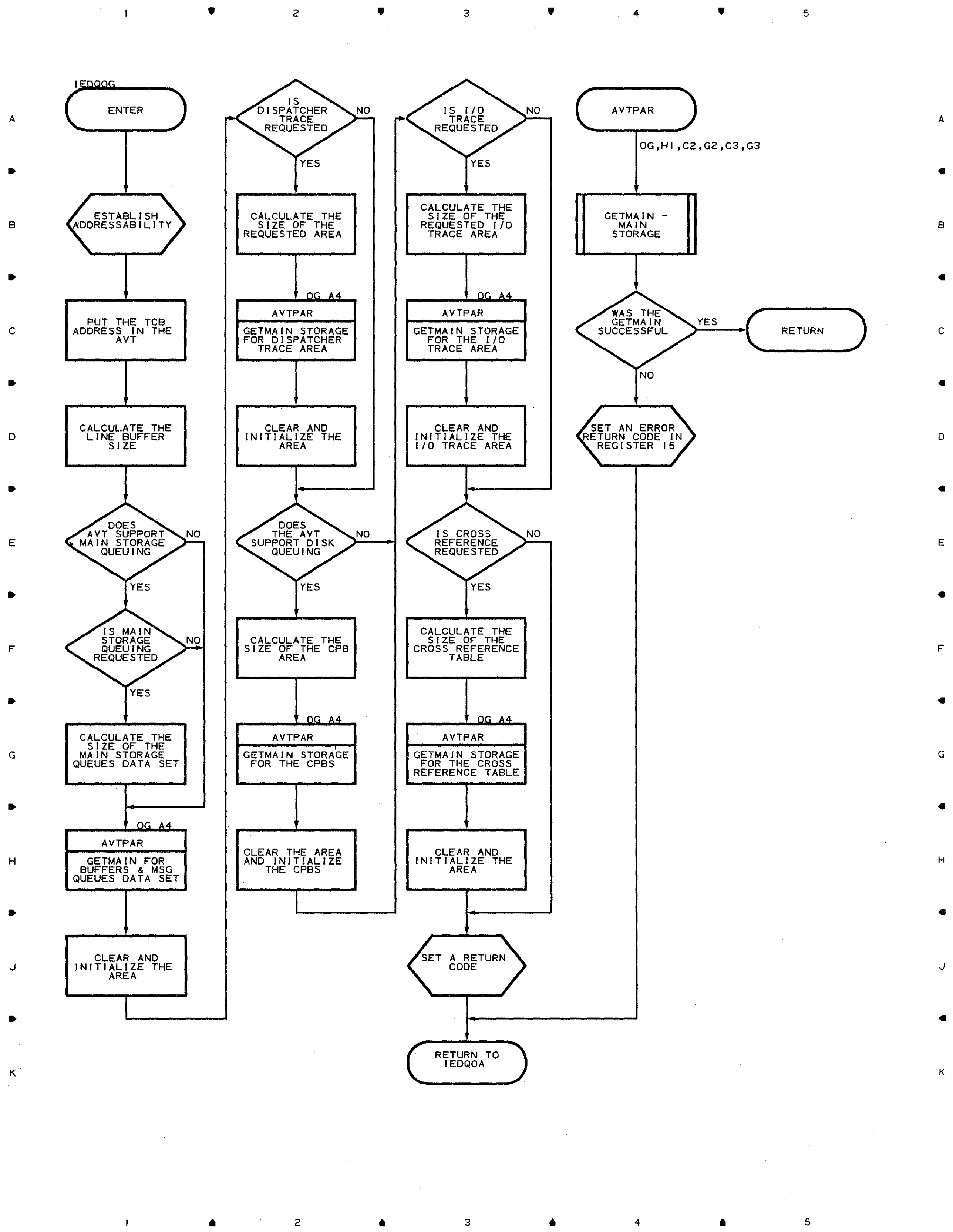

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQOG
ENTER
ESTABLISH ADDRESSABILITY
PUT THE TCB ADDRESS IN THE AVT
CALCULATE THE LINE BUFFER SIZE
DOES AVT SUPPORT MAIN STORAGE QUEUING
NO
YES
IS MAIN STORAGE QUEUING REQUESTED
NO
YES
CALCULATE THE SIZE OF THE MAIN STORAGE QUEUES DATA SET
OG A4
AVTPAR
GETMAIN FOR BUFFERS & MSG QUEUES DATA SET
CLEAR AND INITIALIZE THE AREA
IS DISPATCHER TRACE REQUESTED
NO
YES
CALCULATE THE SIZE OF THE REQUESTED AREA
OG A4
AVTPAR
GETMAIN STORAGE FOR DISPATCHER TRACE AREA
CLEAR AND INITIALIZE THE AREA
DOES THE AVT SUPPORT DISK QUEUING
NO
YES
CALCULATE THE SIZE OF THE CPB AREA
OG A4
AVTPAR
GETMAIN STORAGE FOR THE CPBS
CLEAR THE AREA AND INITIALIZE THE CPBS
IS I/O TRACE REQUESTED
NO
YES
CALCULATE THE SIZE OF THE REQUESTED I/O TRACE AREA
OG A4
AVTPAR
GETMAIN STORAGE FOR THE I/O TRACE AREA
CLEAR AND INITIALIZE THE I/O TRACE AREA
IS CROSS REFERENCE REQUESTED
NO
YES
CALCULATE THE SIZE OF THE CROSS REFERENCE TABLE
OG A4
AVTPAR
GETMAIN STORAGE FOR THE CROSS REFERENCE TABLE
CLEAR AND INITIALIZE THE AREA
SET A RETURN CODE
RETURN TO IEDQOA
AVTPAR
OG,H1,C2,G2,C3,G3
GETMAIN - MAIN STORAGE
WAS THE GETMAIN SUCCESSFUL
YES
RETURN
NO
SET AN ERROR RETURN CODE IN REGISTER 15

Chart OM TERMNAME TABLE SORT ROUTINE

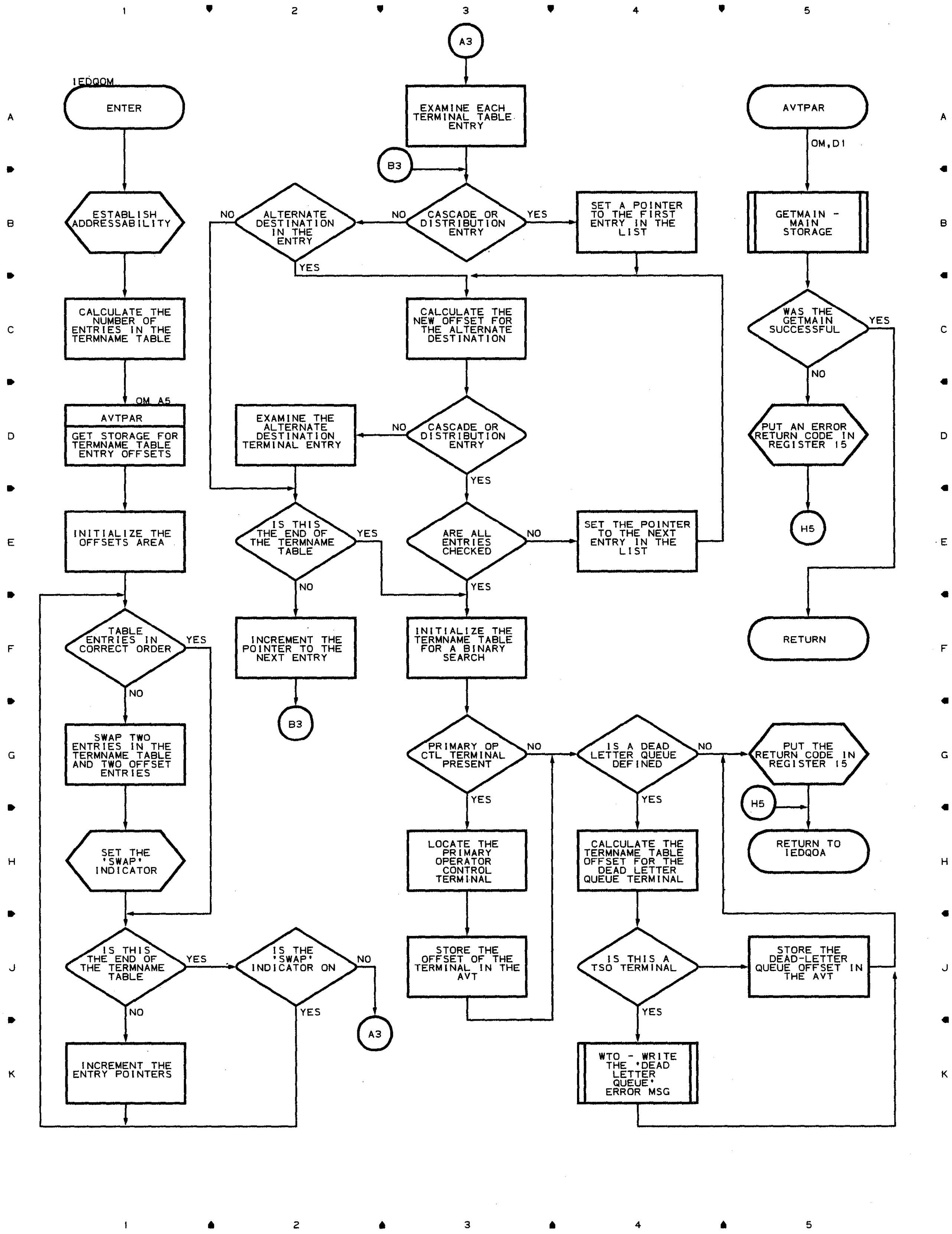
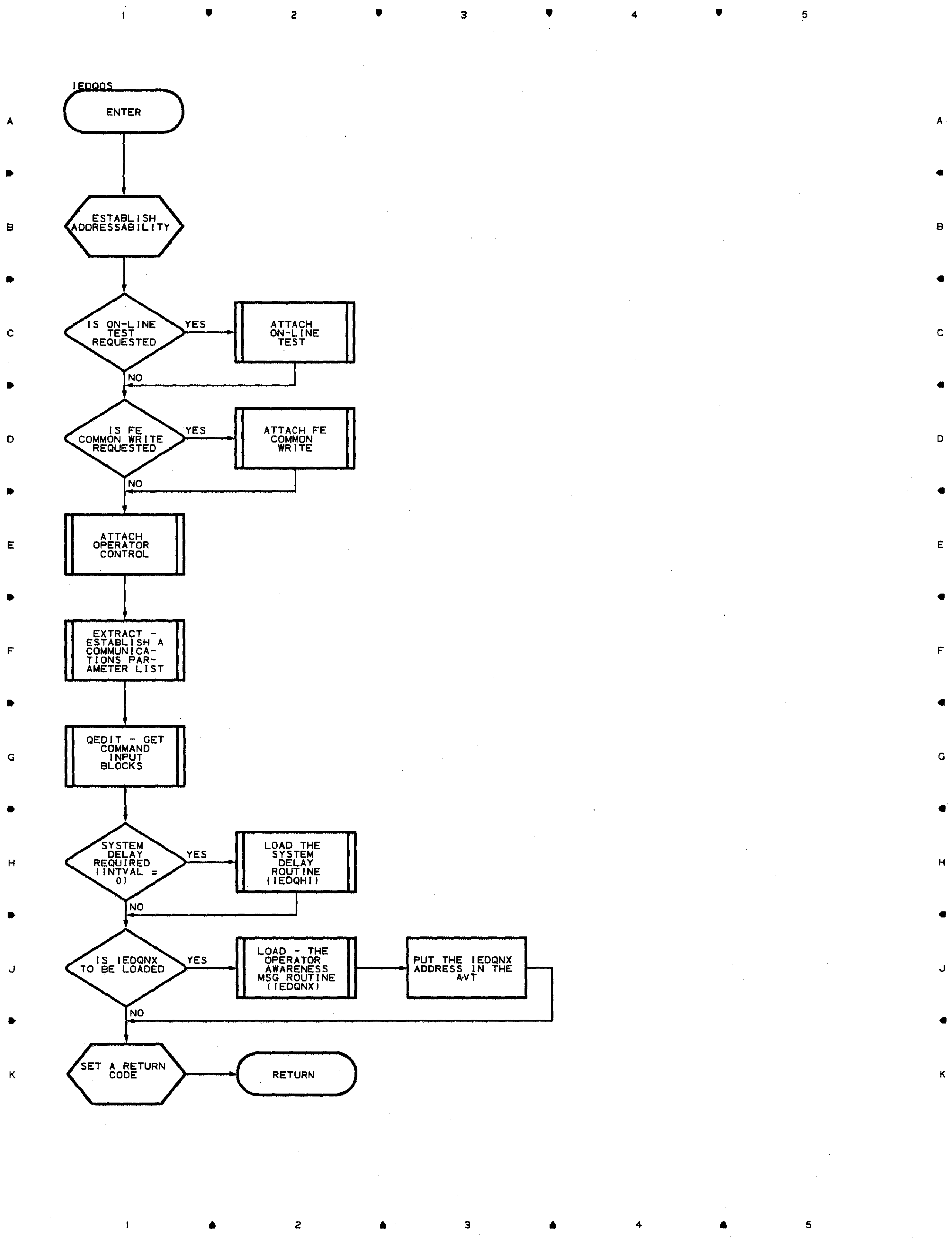

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQOM
ENTER
ESTABLISH ADDRESSABILITY
CALCULATE THE NUMBER OF ENTRIES IN THE TERMNAME TABLE
OM,A5
AVTPAR
GET STORAGE FOR TERMNAME TABLE ENTRY OFFSETS
INITIALIZE THE OFFSETS AREA
TABLE ENTRIES IN CORRECT ORDER
YES
NO
SWAP TWO ENTRIES IN THE TERMNAME TABLE AND TWO OFFSET ENTRIES
SET THE 'SWAP' INDICATOR
IS THIS THE END OF THE TERMNAME TABLE
YES
NO
INCREMENT THE ENTRY POINTERS
IS THE 'SWAP' INDICATOR ON
NO
YES
A3
A3
EXAMINE EACH TERMINAL TABLE ENTRY
B3
CASCADE OR DISTRIBUTION ENTRY
NO
YES
ALTERNATE DESTINATION IN THE ENTRY
NO
YES
SET A POINTER TO THE FIRST ENTRY IN THE LIST
CALCULATE THE NEW OFFSET FOR THE ALTERNATE DESTINATION
CASCADE OR DISTRIBUTION ENTRY
NO
YES
EXAMINE THE ALTERNATE DESTINATION TERMINAL ENTRY
IS THIS THE END OF THE TERMNAME TABLE
YES
NO
INCREMENT THE POINTER TO THE NEXT ENTRY
B3
ARE ALL ENTRIES CHECKED
NO
YES
SET THE POINTER TO THE NEXT ENTRY IN THE LIST
INITIALIZE THE TERMNAME TABLE FOR A BINARY SEARCH
PRIMARY OP CTL TERMINAL PRESENT
NO
YES
LOCATE THE PRIMARY OPERATOR CONTROL TERMINAL
STORE THE OFFSET OF THE TERMINAL IN THE AVT
IS A DEAD LETTER QUEUE DEFINED
NO
YES
CALCULATE THE TERMNAME TABLE OFFSET FOR THE DEAD LETTER QUEUE TERMINAL
IS THIS A TSO TERMINAL
YES
WTO - WRITE THE 'DEAD LETTER QUEUE' ERROR MSG
STORE THE DEAD-LETTER QUEUE OFFSET IN THE AVT
AVTPAR
OM,D1
GETMAIN - MAIN STORAGE
WAS THE GETMAIN SUCCESSFUL
YES
NO
PUT AN ERROR RETURN CODE IN REGISTER 15
H5
RETURN
PUT THE RETURN CODE IN REGISTER 15
H5
RETURN TO IEDQOA

Chart OS ATTACH ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQOS
ENTER
ESTABLISH ADDRESSABILITY
IS ON-LINE TEST REQUESTED
YES
ATTACH ON-LINE TEST
NO
IS FE COMMON WRITE REQUESTED
YES
ATTACH FE COMMON WRITE
NO
ATTACH OPERATOR CONTROL
EXTRACT - ESTABLISH A COMMUNICA-TIONS PAR-AMETER LIST
QEDIT - GET COMMAND INPUT BLOCKS
SYSTEM DELAY REQUIRED (INTVAL = 0)
YES
LOAD THE SYSTEM DELAY ROUTINE (IEDQHI)
NO
IS IEDQNX TO BE LOADED
YES
LOAD - THE OPERATOR AWARENESS MSG ROUTINE (IEDQNX)
PUT THE IEDQNX ADDRESS IN THE AVT
NO
SET A RETURN CODE
RETURN

Chart Q0 LINE I/O INTERRUPT TRACE ROUTINE

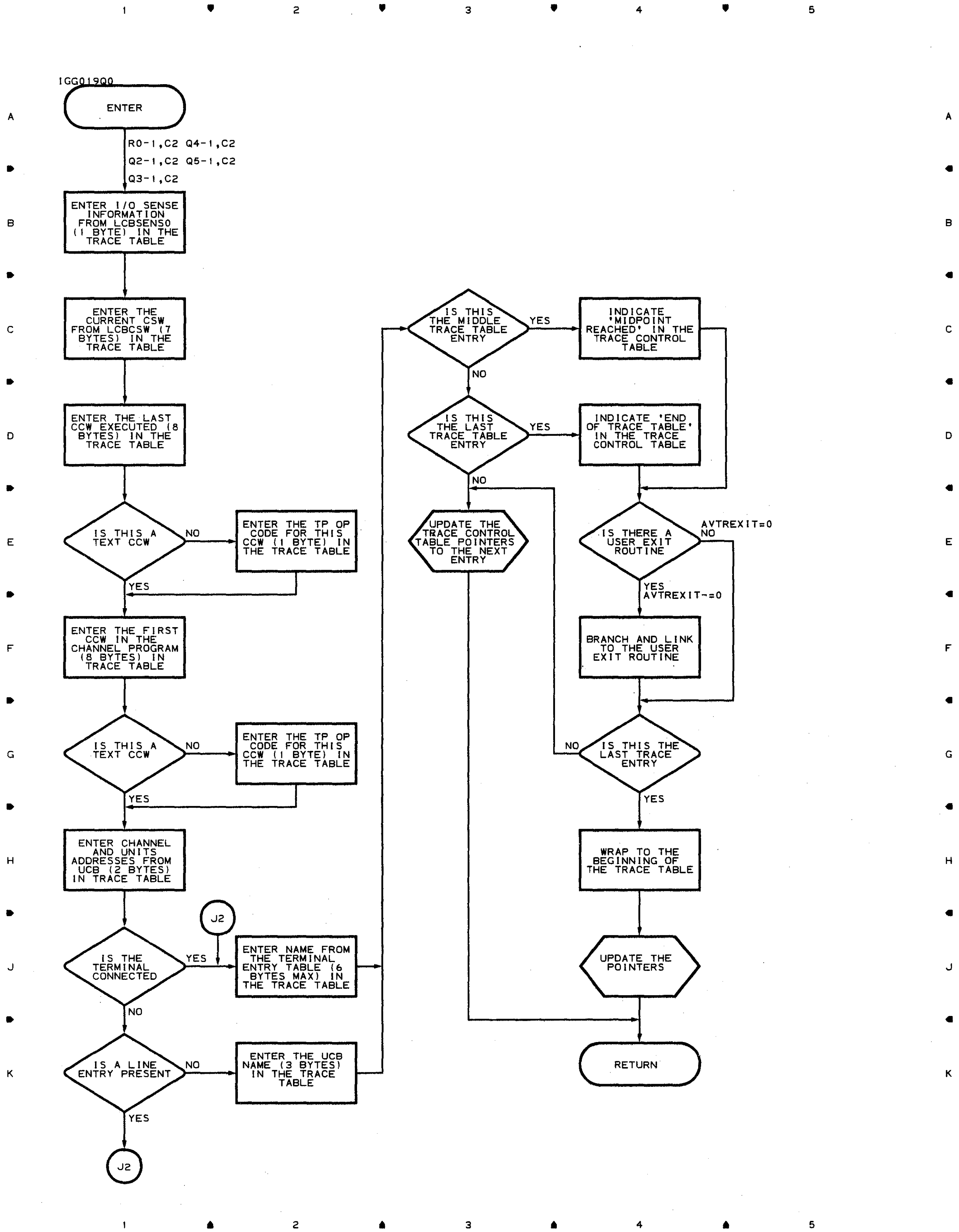

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG019Q0
ENTER
R0-1,C2 Q4-1,C2
Q2-1,C2 Q5-1,C2
Q3-1,C2
ENTER I/O SENSE INFORMATION FROM LCBSENS0 (1 BYTE) IN THE TRACE TABLE
ENTER THE CURRENT CSW FROM LCBCSW (7 BYTES) IN THE TRACE TABLE
ENTER THE LAST CCW EXECUTED (8 BYTES) IN THE TRACE TABLE
IS THIS A TEXT CCW
NO
YES
ENTER THE TP OP CODE FOR THIS CCW (1 BYTE) IN THE TRACE TABLE
ENTER THE FIRST CCW IN THE CHANNEL PROGRAM (8 BYTES) IN TRACE TABLE
IS THIS A TEXT CCW
NO
YES
ENTER THE TP OP CODE FOR THIS CCW (1 BYTE) IN THE TRACE TABLE
ENTER CHANNEL AND UNITS ADDRESSES FROM UCB (2 BYTES) IN TRACE TABLE
J2
IS THE TERMINAL CONNECTED
YES
NO
ENTER NAME FROM THE TERMINAL ENTRY TABLE (6 BYTES MAX) IN THE TRACE TABLE
IS A LINE ENTRY PRESENT
NO
YES
ENTER THE UCB NAME (3 BYTES) IN THE TRACE TABLE
J2
IS THIS THE MIDDLE TRACE TABLE ENTRY
YES
NO
INDICATE 'MIDPOINT REACHED' IN THE TRACE CONTROL TABLE
IS THIS THE LAST TRACE TABLE ENTRY
YES
NO
INDICATE 'END OF TRACE TABLE' IN THE TRACE CONTROL TABLE
UPDATE THE TRACE CONTROL TABLE POINTERS TO THE NEXT ENTRY
IS THERE A USER EXIT ROUTINE
AVTREXIT=0
NO
YES
AVTREXIT¬=0
BRANCH AND LINK TO THE USER EXIT ROUTINE
IS THIS THE LAST TRACE ENTRY
NO
YES
WRAP TO THE BEGINNING OF THE TRACE TABLE
UPDATE THE POINTERS
RETURN

Chart Q1 LOCAL RECEIVE SCHEDULER

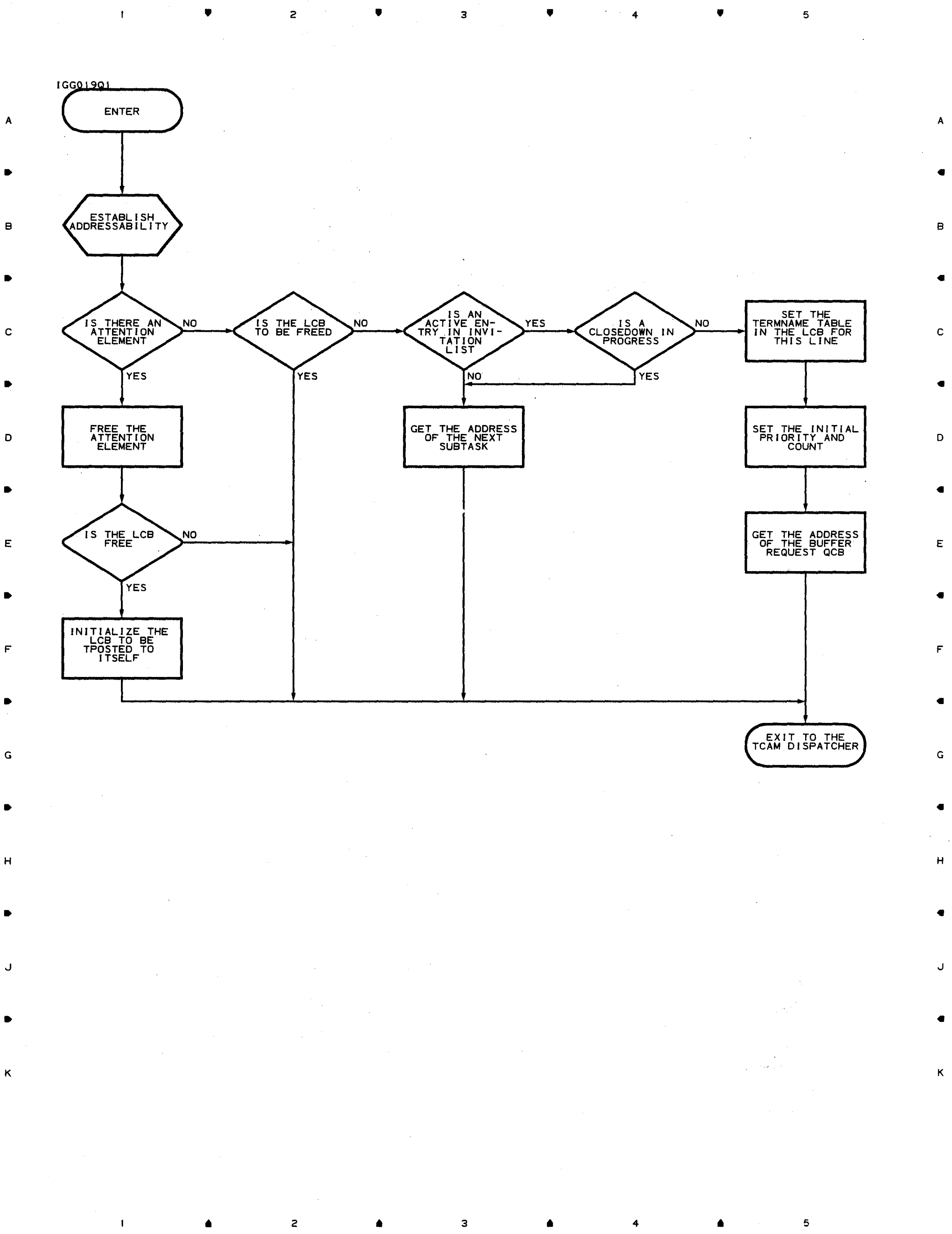

IGG019Q1
ENTER
ESTABLISH ADDRESSABILITY
IS THERE AN ATTENTION ELEMENT
NO
YES
IS THE LCB TO BE FREED
NO
YES
IS AN ACTIVE ENTRY IN INVITATION LIST
YES
NO
IS A CLOSEDOWN IN PROGRESS
NO
YES
SET THE TERMNAME TABLE IN THE LCB FOR THIS LINE
FREE THE ATTENTION ELEMENT
GET THE ADDRESS OF THE NEXT SUBTASK
SET THE INITIAL PRIORITY AND COUNT
IS THE LCB FREE
NO
YES
GET THE ADDRESS OF THE BUFFER REQUEST QCB
INITIALIZE THE LCB TO BE TPOSTED TO ITSELF
EXIT TO THE TCAM DISPATCHER
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart Q2-1 LINE END APPENDAGE FOR BSC LINES

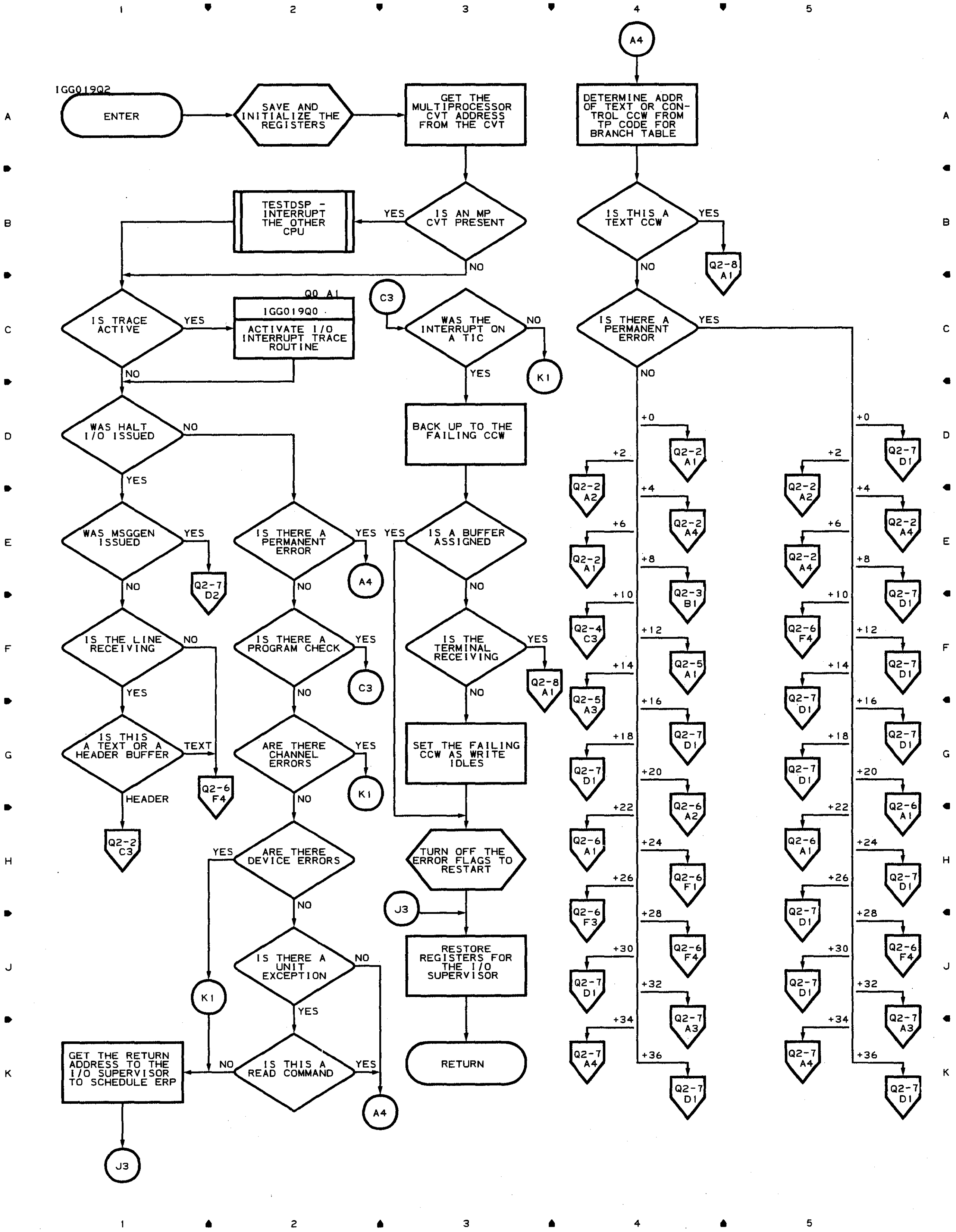
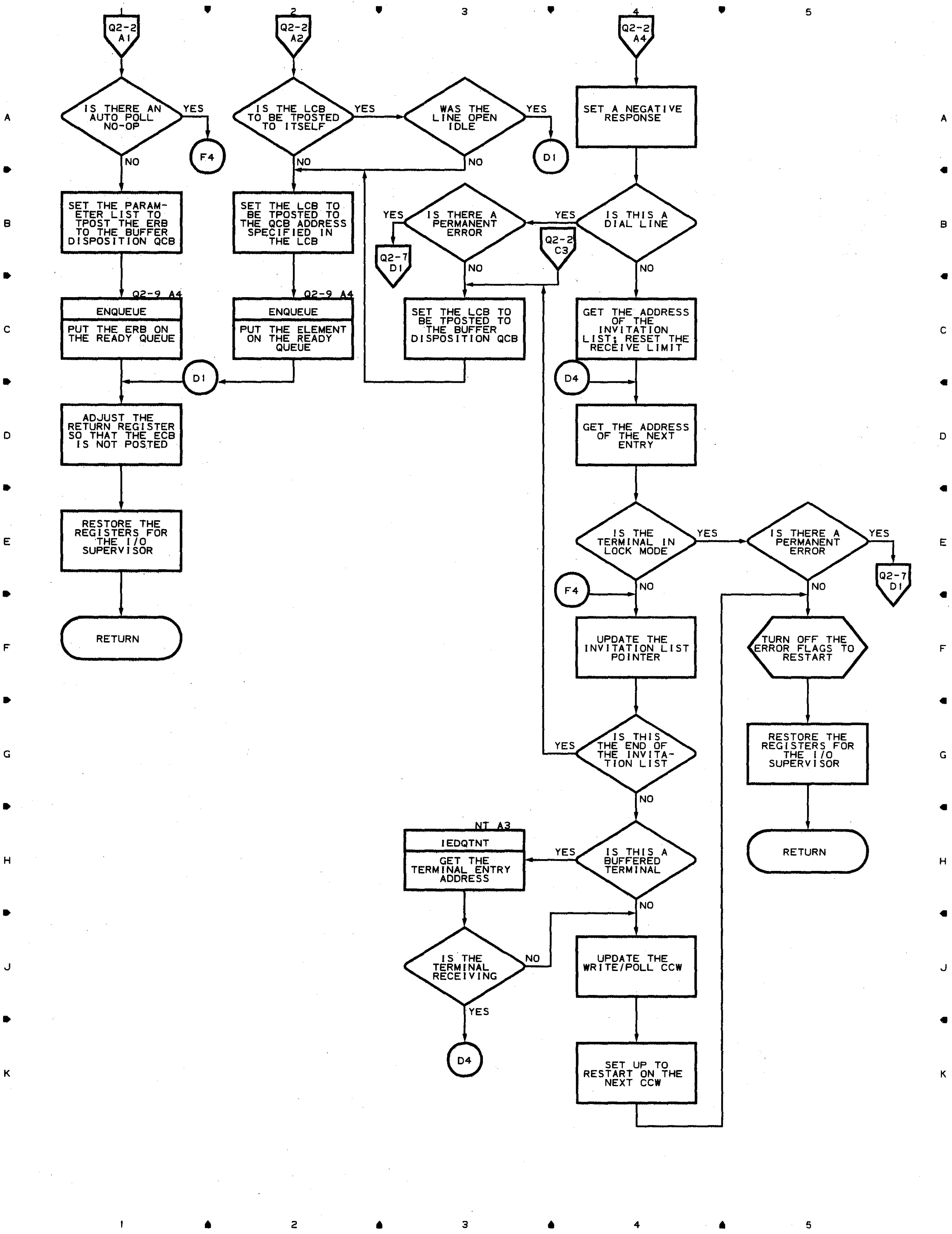


Chart Q2-2 LINE END APPENDAGE FOR BSC LINES



1
2
3
4
5
Q2-2 A1
Q2-2 A2
Q2-2 A4
IS THERE AN AUTO POLL NO-OP
YES
F4
NO
IS THE LCB TO BE TPOSTED TO ITSELF
YES
WAS THE LINE OPEN IDLE
YES
D1
NO
NO
SET A NEGATIVE RESPONSE
SET THE PARAMETER LIST TO TPOST THE ERB TO THE BUFFER DISPOSITION QCB
SET THE LCB TO BE TPOSTED TO THE QCB ADDRESS SPECIFIED IN THE LCB
YES
IS THERE A PERMANENT ERROR
YES
IS THIS A DIAL LINE
Q2-7 D1
NO
Q2-2 C3
NO
Q2-9 A4
ENQUEUE
PUT THE ERB ON THE READY QUEUE
Q2-9 A4
ENQUEUE
PUT THE ELEMENT ON THE READY QUEUE
SET THE LCB TO BE TPOSTED TO THE BUFFER DISPOSITION QCB
GET THE ADDRESS OF THE INVITATION LIST; RESET THE RECEIVE LIMIT
D1
D4
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
GET THE ADDRESS OF THE NEXT ENTRY
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
IS THE TERMINAL IN LOCK MODE
YES
IS THERE A PERMANENT ERROR
YES
Q2-7 D1
F4
NO
NO
RETURN
UPDATE THE INVITATION LIST POINTER
TURN OFF THE ERROR FLAGS TO RESTART
YES
IS THIS THE END OF THE INVITATION LIST
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
NO
NT A3
IEDQTNT
GET THE TERMINAL ENTRY ADDRESS
YES
IS THIS A BUFFERED TERMINAL
RETURN
NO
IS THE TERMINAL RECEIVING
NO
UPDATE THE WRITE/POLL CCW
YES
D4
SET UP TO RESTART ON THE NEXT CCW
A
B
C
D
E
F
G
H
J
K

Chart Q2-3 LINE END APPENDAGE FOR BSC LINES

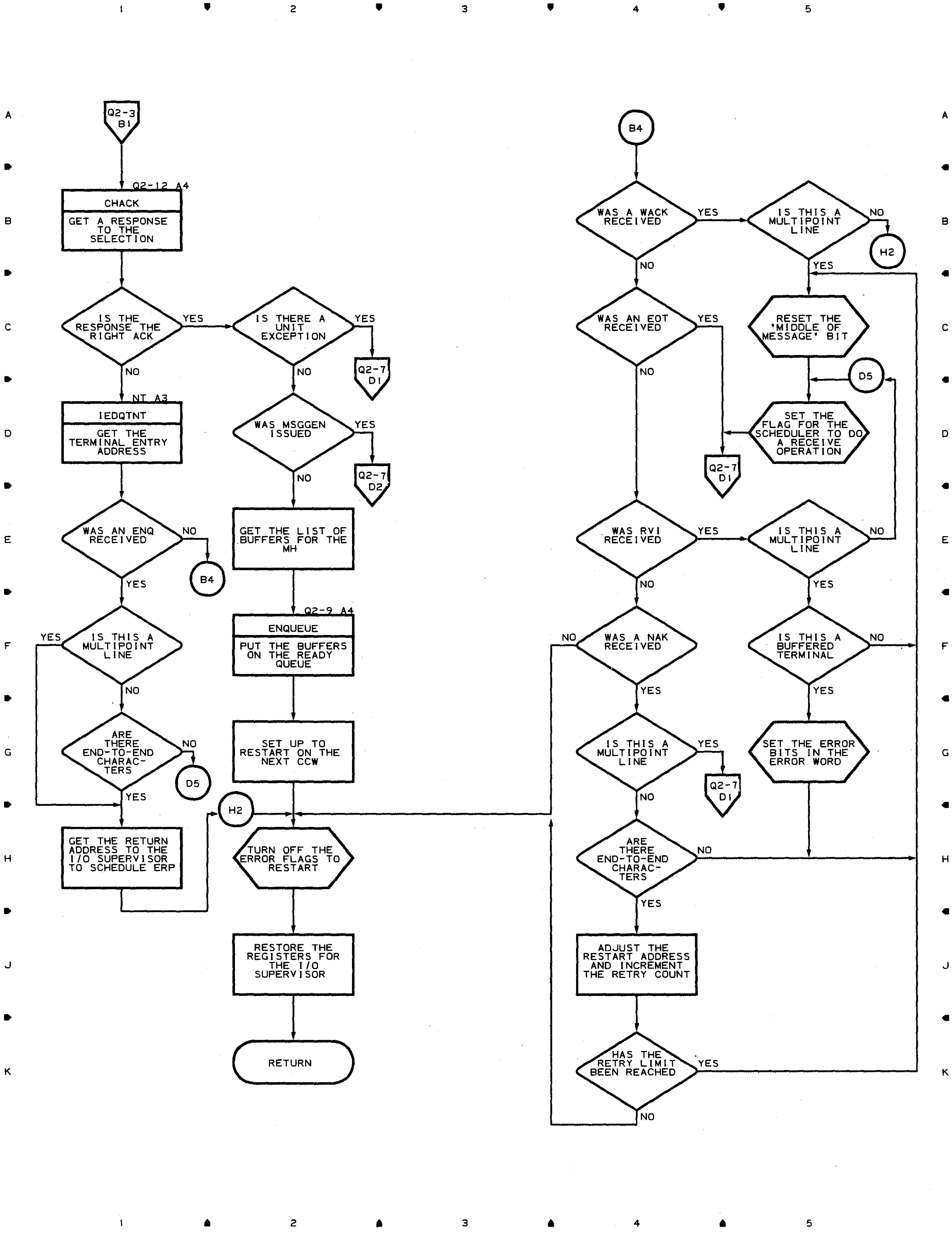

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
Q2-3 B1
Q2-12 A4
CHACK
GET A RESPONSE TO THE SELECTION
IS THE RESPONSE THE RIGHT ACK
YES
NO
IS THERE A UNIT EXCEPTION
YES
NO
Q2-7 D1
NT A3
IEDQTNT
GET THE TERMINAL ENTRY ADDRESS
WAS MSGGEN ISSUED
YES
NO
Q2-7 D2
WAS AN ENQ RECEIVED
NO
B4
YES
GET THE LIST OF BUFFERS FOR THE MH
Q2-9 A4
ENQUEUE
PUT THE BUFFERS ON THE READY QUEUE
IS THIS A MULTIPOINT LINE
YES
NO
ARE THERE END-TO-END CHARAC-TERS
NO
D5
YES
SET UP TO RESTART ON THE NEXT CCW
H2
GET THE RETURN ADDRESS TO THE I/O SUPERVISOR TO SCHEDULE ERP
TURN OFF THE ERROR FLAGS TO RESTART
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN
B4
WAS A WACK RECEIVED
YES
NO
IS THIS A MULTIPOINT LINE
NO
H2
YES
WAS AN EOT RECEIVED
YES
NO
RESET THE 'MIDDLE OF MESSAGE' BIT
D5
SET THE FLAG FOR THE SCHEDULER TO DO A RECEIVE OPERATION
Q2-7 D1
WAS RVI RECEIVED
YES
NO
IS THIS A MULTIPOINT LINE
NO
YES
WAS A NAK RECEIVED
NO
YES
IS THIS A BUFFERED TERMINAL
NO
YES
IS THIS A MULTIPOINT LINE
YES
NO
Q2-7 D1
SET THE ERROR BITS IN THE ERROR WORD
ARE THERE END-TO-END CHARAC-TERS
NO
YES
ADJUST THE RESTART ADDRESS AND INCREMENT THE RETRY COUNT
HAS THE RETRY LIMIT BEEN REACHED
YES
NO

Chart Q2-4 LINE END APPENDAGE FOR BSC LINES

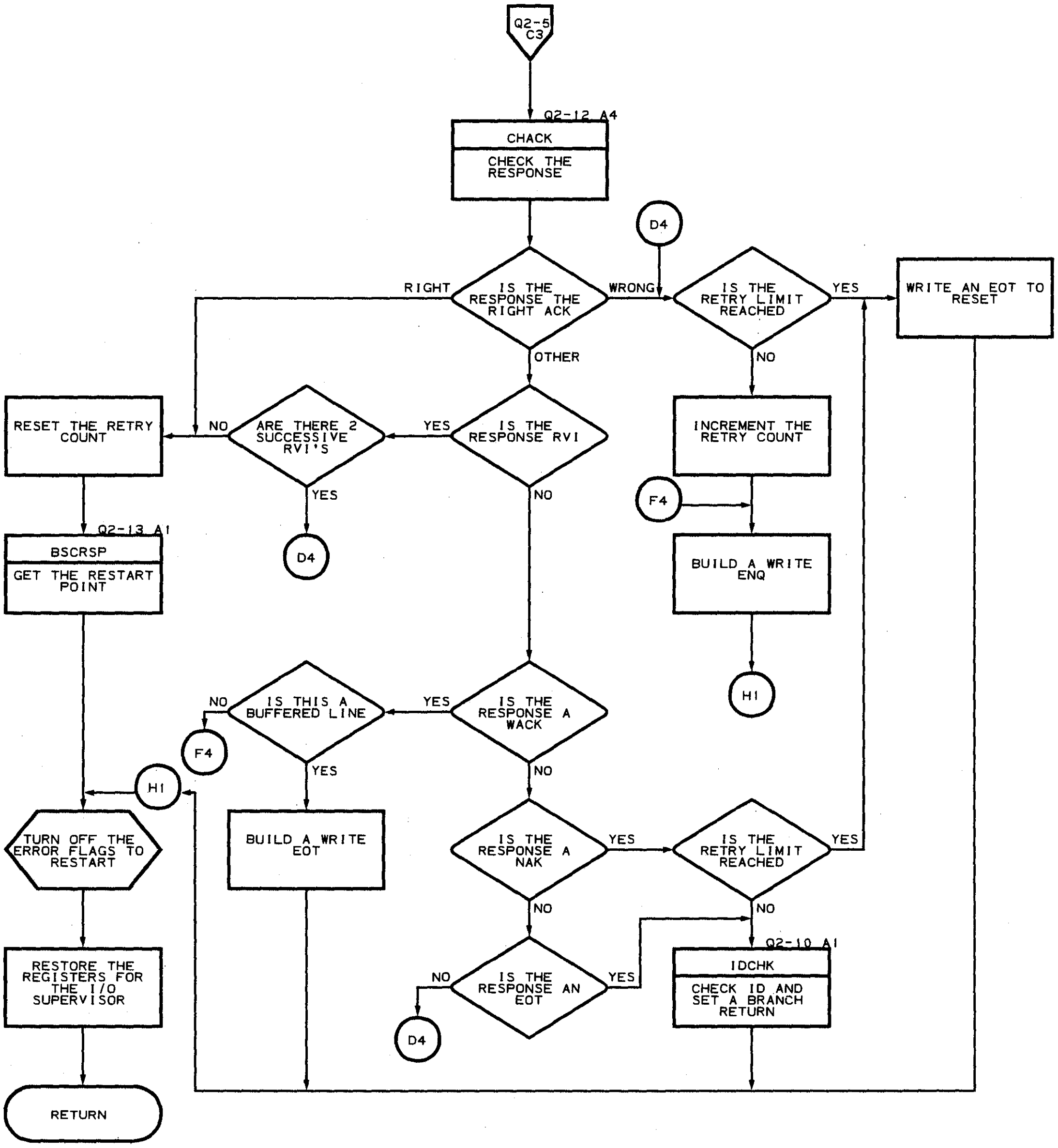


Chart Q2-5 LINE END APPENDAGE FOR BSC LINES

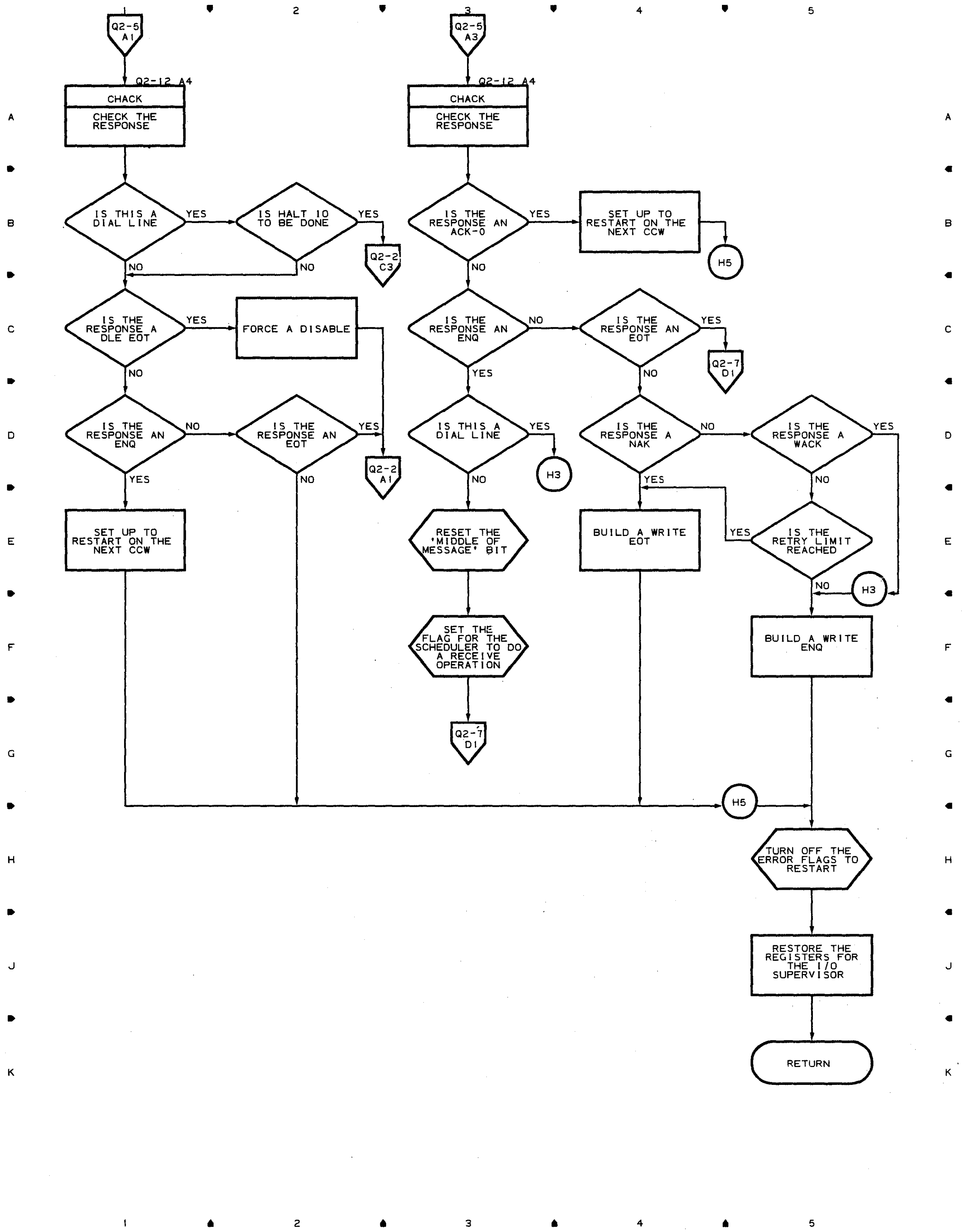


Chart Q2-6 LINE END APPENDAGE FOR BSC LINES

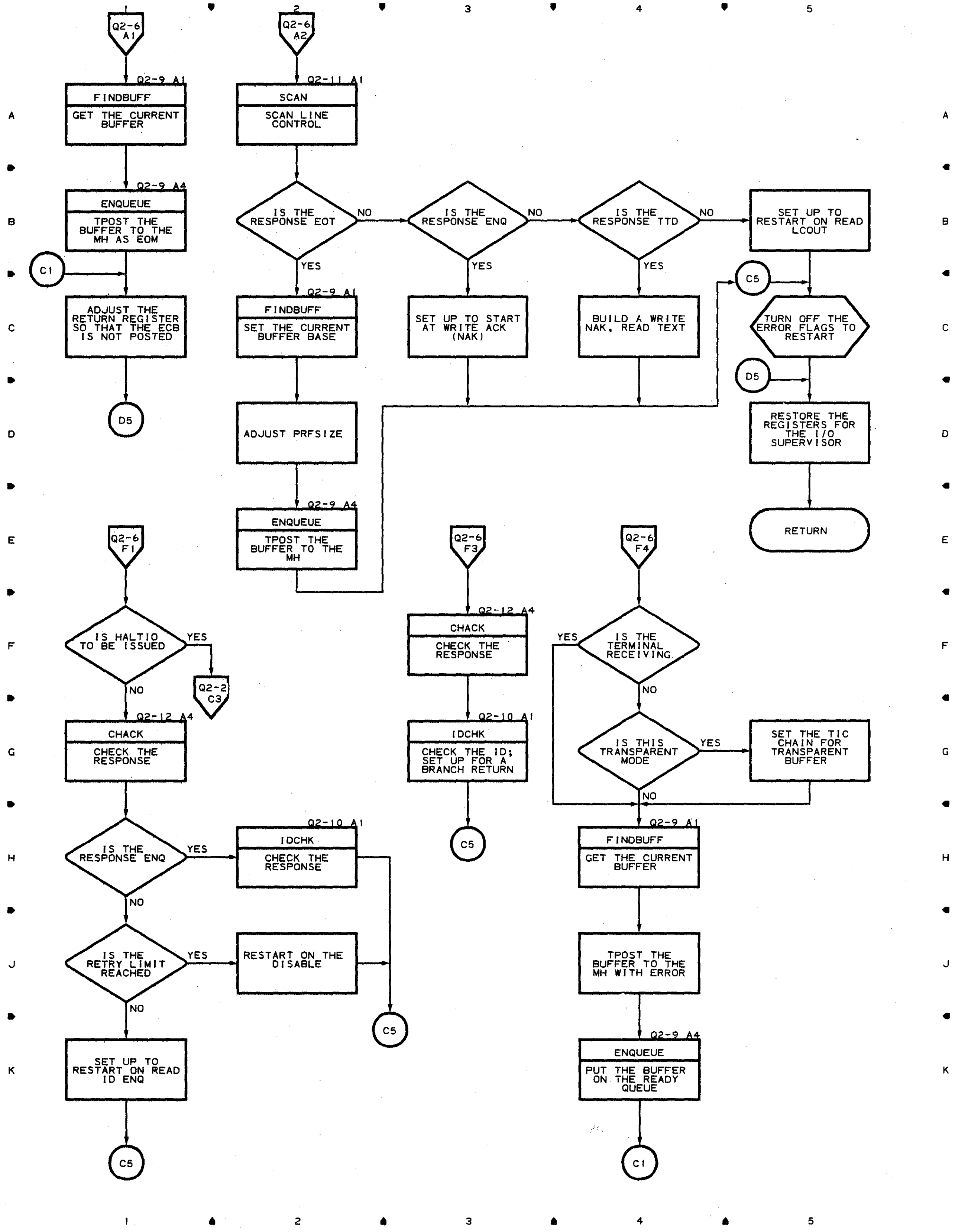
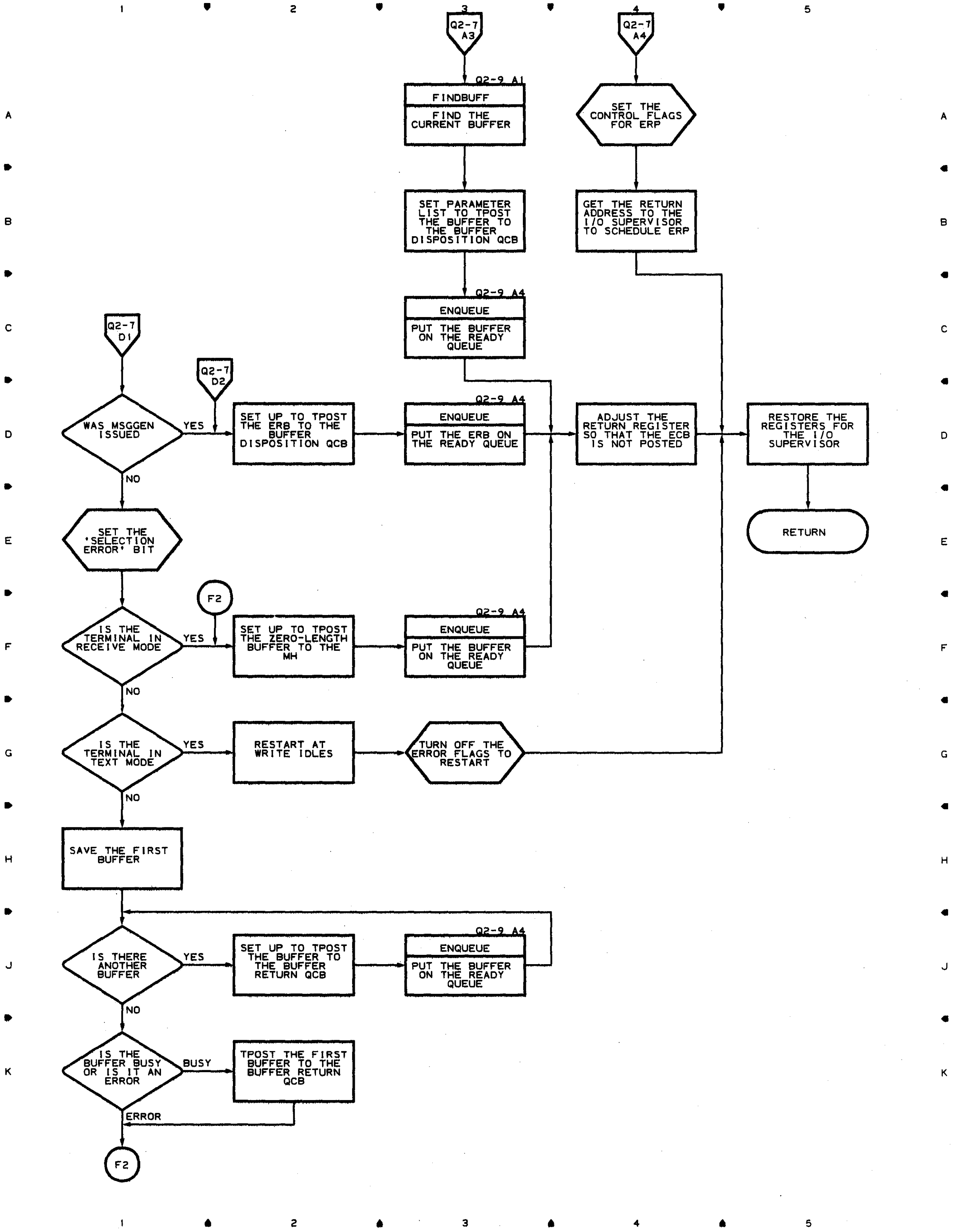

Q2-6 A1
Q2-9 A1
FINDBUFF
GET THE CURRENT BUFFER
Q2-9 A4
ENQUEUE
TPOST THE BUFFER TO THE MH AS EOM
C1
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
D5
Q2-6 A2
Q2-11 A1
SCAN
SCAN LINE CONTROL
IS THE RESPONSE EOT
NO
YES
Q2-9 A1
FINDBUFF
SET THE CURRENT BUFFER BASE
ADJUST PRFSIZE
Q2-9 A4
ENQUEUE
TPOST THE BUFFER TO THE MH
IS THE RESPONSE ENQ
NO
YES
SET UP TO START AT WRITE ACK (NAK)
IS THE RESPONSE TTD
NO
YES
BUILD A WRITE NAK, READ TEXT
SET UP TO RESTART ON READ LCOUT
C5
TURN OFF THE ERROR FLAGS TO RESTART
D5
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN
Q2-6 F1
IS HALTIO TO BE ISSUED
YES
Q2-2 C3
NO
Q2-12 A4
CHACK
CHECK THE RESPONSE
IS THE RESPONSE ENQ
YES
Q2-10 A1
IDCHK
CHECK THE RESPONSE
NO
IS THE RETRY LIMIT REACHED
YES
RESTART ON THE DISABLE
C5
NO
SET UP TO RESTART ON READ ID ENQ
C5
Q2-6 F3
Q2-12 A4
CHACK
CHECK THE RESPONSE
Q2-10 A1
IDCHK
CHECK THE ID; SET UP FOR A BRANCH RETURN
C5
Q2-6 F4
IS THE TERMINAL RECEIVING
YES
NO
IS THIS TRANSPARENT MODE
YES
SET THE TIC CHAIN FOR TRANSPARENT BUFFER
NO
Q2-9 A1
FINDBUFF
GET THE CURRENT BUFFER
TPOST THE BUFFER TO THE MH WITH ERROR
Q2-9 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
C1
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart Q2-7 LINE END APPENDAGE FOR BSC LINES



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
Q2-7 A3
Q2-7 A4
Q2-9 A1
FINDBUFF
FIND THE CURRENT BUFFER
SET THE CONTROL FLAGS FOR ERP
SET PARAMETER LIST TO TPOST THE BUFFER TO THE BUFFER DISPOSITION QCB
GET THE RETURN ADDRESS TO THE I/O SUPERVISOR TO SCHEDULE ERP
Q2-9 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
Q2-7 D1
Q2-7 D2
WAS MSGGEN ISSUED
YES
SET UP TO TPOST THE ERB TO THE BUFFER DISPOSITION QCB
Q2-9 A4
ENQUEUE
PUT THE ERB ON THE READY QUEUE
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
NO
SET THE 'SELECTION ERROR' BIT
RETURN
F2
IS THE TERMINAL IN RECEIVE MODE
YES
SET UP TO TPOST THE ZERO-LENGTH BUFFER TO THE MH
Q2-9 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
NO
IS THE TERMINAL IN TEXT MODE
YES
RESTART AT WRITE IDLES
TURN OFF THE ERROR FLAGS TO RESTART
NO
SAVE THE FIRST BUFFER
IS THERE ANOTHER BUFFER
YES
SET UP TO TPOST THE BUFFER TO THE BUFFER RETURN QCB
Q2-9 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
NO
IS THE BUFFER BUSY OR IS IT AN ERROR
BUSY
TPOST THE FIRST BUFFER TO THE BUFFER RETURN QCB
ERROR
F2

Chart Q2-8 LINE END APPENDAGE FOR BSC LINES

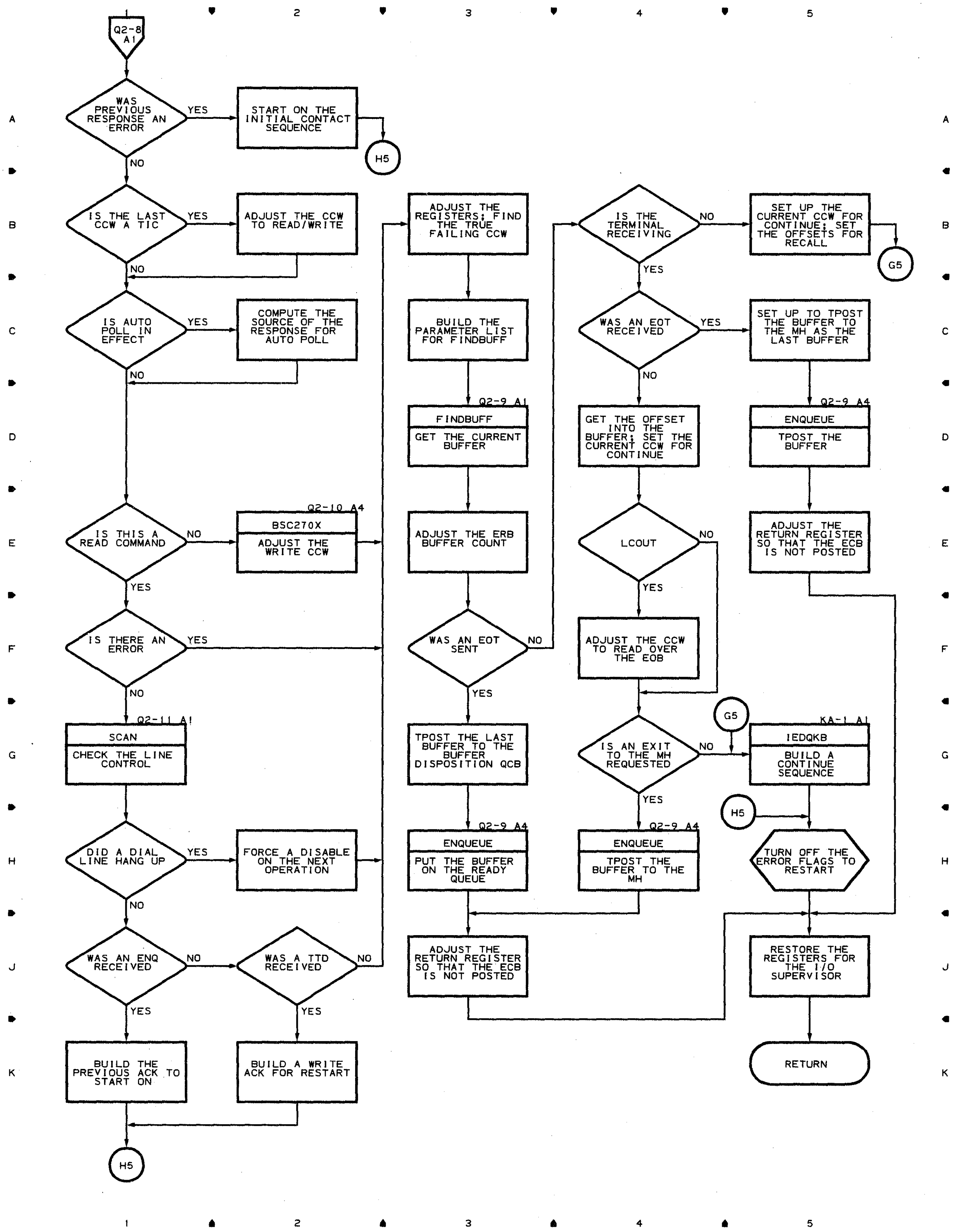

1
2
3
4
5
Q2-8 A1
WAS PREVIOUS RESPONSE AN ERROR
YES
START ON THE INITIAL CONTACT SEQUENCE
H5
NO
IS THE LAST CCW A TIC
YES
ADJUST THE CCW TO READ/WRITE
NO
IS AUTO POLL IN EFFECT
YES
COMPUTE THE SOURCE OF THE RESPONSE FOR AUTO POLL
NO
IS THIS A READ COMMAND
NO
Q2-10 A4
BSC270X
ADJUST THE WRITE CCW
YES
IS THERE AN ERROR
YES
NO
Q2-11 A1
SCAN
CHECK THE LINE CONTROL
DID A DIAL LINE HANG UP
YES
FORCE A DISABLE ON THE NEXT OPERATION
NO
WAS AN ENQ RECEIVED
NO
WAS A TTD RECEIVED
NO
YES
YES
BUILD THE PREVIOUS ACK TO START ON
BUILD A WRITE ACK FOR RESTART
H5
ADJUST THE REGISTERS; FIND THE TRUE FAILING CCW
BUILD THE PARAMETER LIST FOR FINDBUFF
Q2-9 A1
FINDBUFF
GET THE CURRENT BUFFER
ADJUST THE ERB BUFFER COUNT
WAS AN EOT SENT
NO
YES
TPOST THE LAST BUFFER TO THE BUFFER DISPOSITION QCB
Q2-9 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
IS THE TERMINAL RECEIVING
NO
YES
WAS AN EOT RECEIVED
YES
NO
GET THE OFFSET INTO THE BUFFER; SET THE CURRENT CCW FOR CONTINUE
LCOUT
NO
YES
ADJUST THE CCW TO READ OVER THE EOB
IS AN EXIT TO THE MH REQUESTED
NO
YES
Q2-9 A4
ENQUEUE
TPOST THE BUFFER TO THE MH
SET UP THE CURRENT CCW FOR CONTINUE; SET THE OFFSETS FOR RECALL
G5
SET UP TO TPOST THE BUFFER TO THE MH AS THE LAST BUFFER
Q2-9 A4
ENQUEUE
TPOST THE BUFFER
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
G5
KA-1 A1
IEDQKB
BUILD A CONTINUE SEQUENCE
H5
TURN OFF THE ERROR FLAGS TO RESTART
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN
A
B
C
D
E
F
G
H
J
K

Chart Q2-9 LINE END APPENDAGE FOR BSC LINES

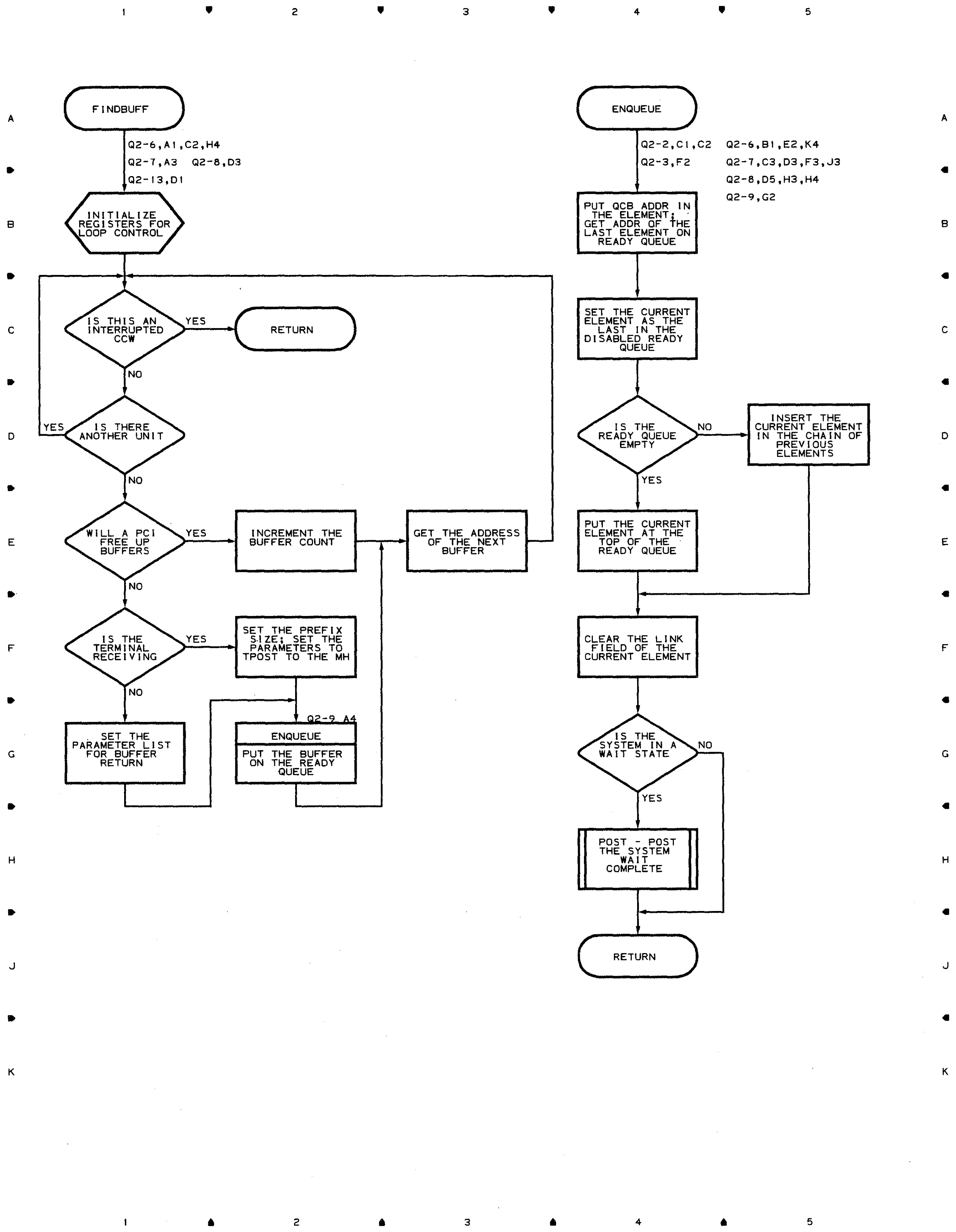
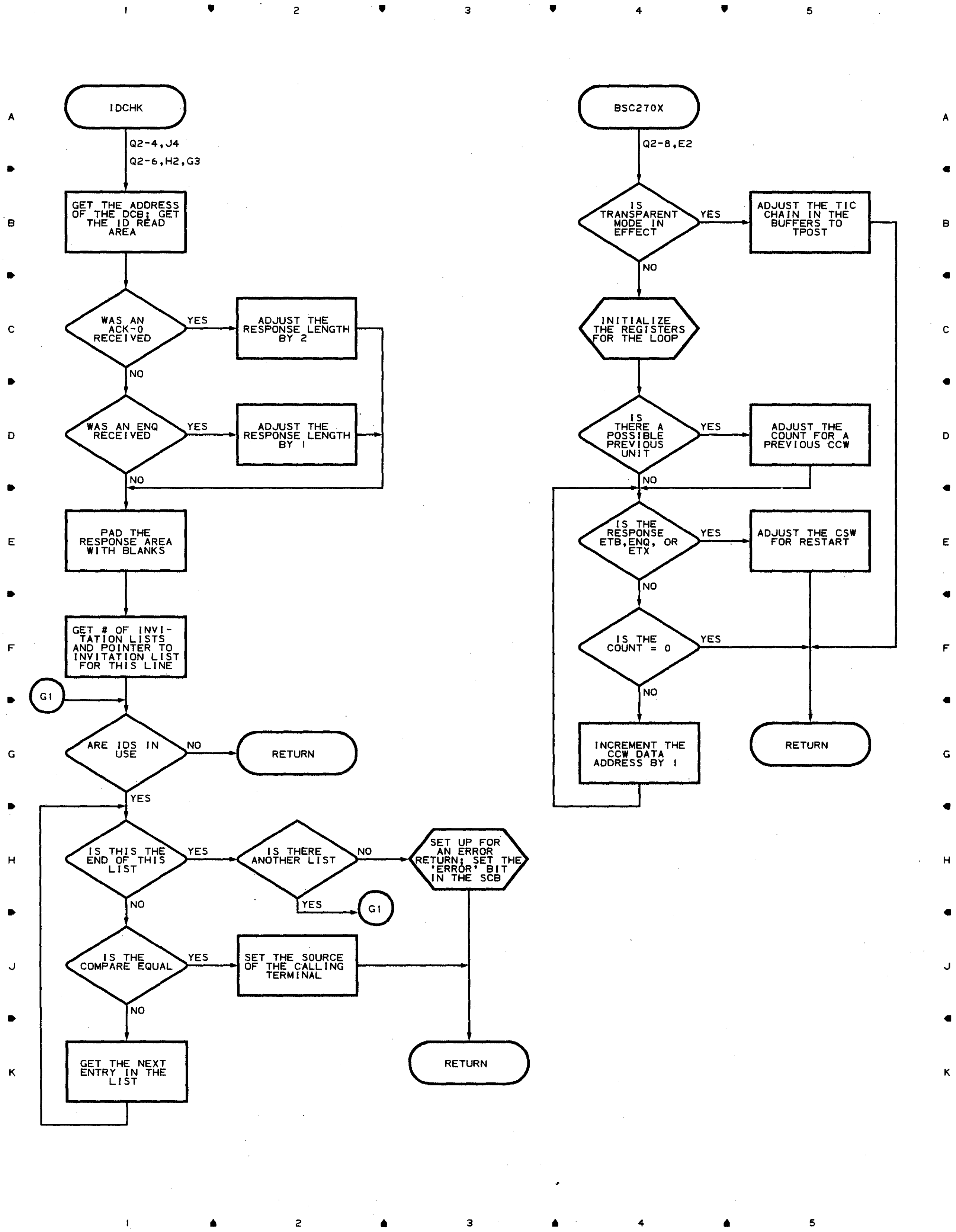
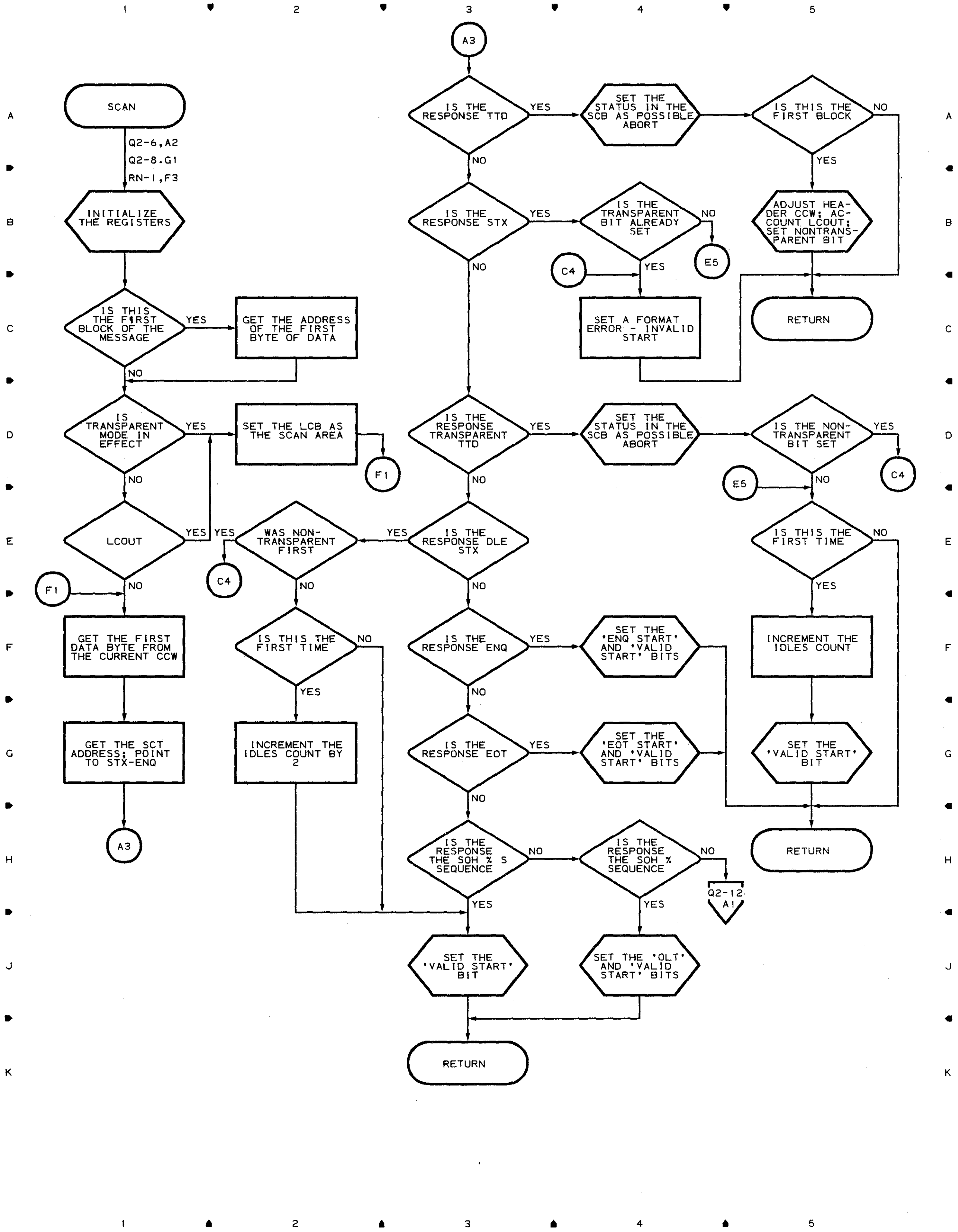

FINDBUFF
Q2-6,A1,C2,H4
Q2-7,A3 Q2-8,D3
Q2-13,D1
INITIALIZE REGISTERS FOR LOOP CONTROL
IS THIS AN INTERRUPTED CCW
YES
RETURN
NO
IS THERE ANOTHER UNIT
YES
NO
WILL A PCI FREE UP BUFFERS
YES
INCREMENT THE BUFFER COUNT
GET THE ADDRESS OF THE NEXT BUFFER
NO
IS THE TERMINAL RECEIVING
YES
SET THE PREFIX SIZE; SET THE PARAMETERS TO TPOST TO THE MH
NO
SET THE PARAMETER LIST FOR BUFFER RETURN
Q2-9 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
ENQUEUE
Q2-2,C1,C2
Q2-3,F2
Q2-6,B1,E2,K4
Q2-7,C3,D3,F3,J3
Q2-8,D5,H3,H4
Q2-9,G2
PUT QCB ADDR IN THE ELEMENT; GET ADDR OF THE LAST ELEMENT ON READY QUEUE
SET THE CURRENT ELEMENT AS THE LAST IN THE DISABLED READY QUEUE
IS THE READY QUEUE EMPTY
NO
INSERT THE CURRENT ELEMENT IN THE CHAIN OF PREVIOUS ELEMENTS
YES
PUT THE CURRENT ELEMENT AT THE TOP OF THE READY QUEUE
CLEAR THE LINK FIELD OF THE CURRENT ELEMENT
IS THE SYSTEM IN A WAIT STATE
NO
YES
POST - POST THE SYSTEM WAIT COMPLETE
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart Q2-10 LINE END APPENDAGE FOR BSC LINES



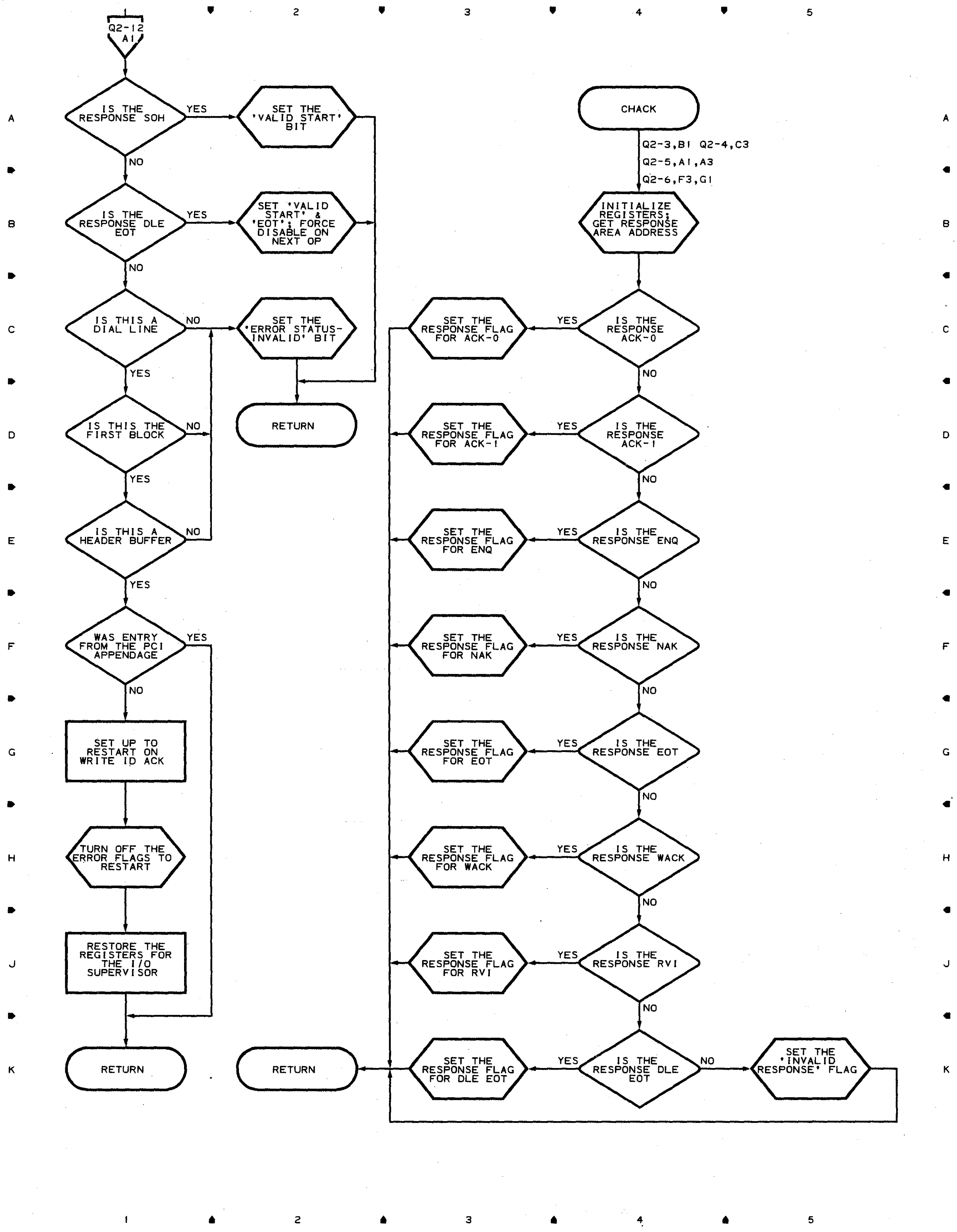
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IDCHK
Q2-4,J4
Q2-6,H2,G3
GET THE ADDRESS OF THE DCB; GET THE ID READ AREA
WAS AN ACK-0 RECEIVED
YES
ADJUST THE RESPONSE LENGTH BY 2
NO
WAS AN ENQ RECEIVED
YES
ADJUST THE RESPONSE LENGTH BY 1
NO
PAD THE RESPONSE AREA WITH BLANKS
GET # OF INVITATION LISTS AND POINTER TO INVITATION LIST FOR THIS LINE
G1
ARE IDS IN USE
NO
RETURN
YES
IS THIS THE END OF THIS LIST
YES
IS THERE ANOTHER LIST
NO
SET UP FOR AN ERROR RETURN; SET THE 'ERROR' BIT IN THE SCB
YES
G1
NO
IS THE COMPARE EQUAL
YES
SET THE SOURCE OF THE CALLING TERMINAL
NO
GET THE NEXT ENTRY IN THE LIST
RETURN
BSC270X
Q2-8,E2
IS TRANSPARENT MODE IN EFFECT
YES
ADJUST THE TIC CHAIN IN THE BUFFERS TO TPOST
NO
INITIALIZE THE REGISTERS FOR THE LOOP
IS THERE A POSSIBLE PREVIOUS UNIT
YES
ADJUST THE COUNT FOR A PREVIOUS CCW
NO
IS THE RESPONSE ETB,ENQ, OR ETX
YES
ADJUST THE CSW FOR RESTART
NO
IS THE COUNT = 0
YES
NO
INCREMENT THE CCW DATA ADDRESS BY 1
RETURN

Chart Q2-11 LINE END APPENDAGE FOR BSC LINES



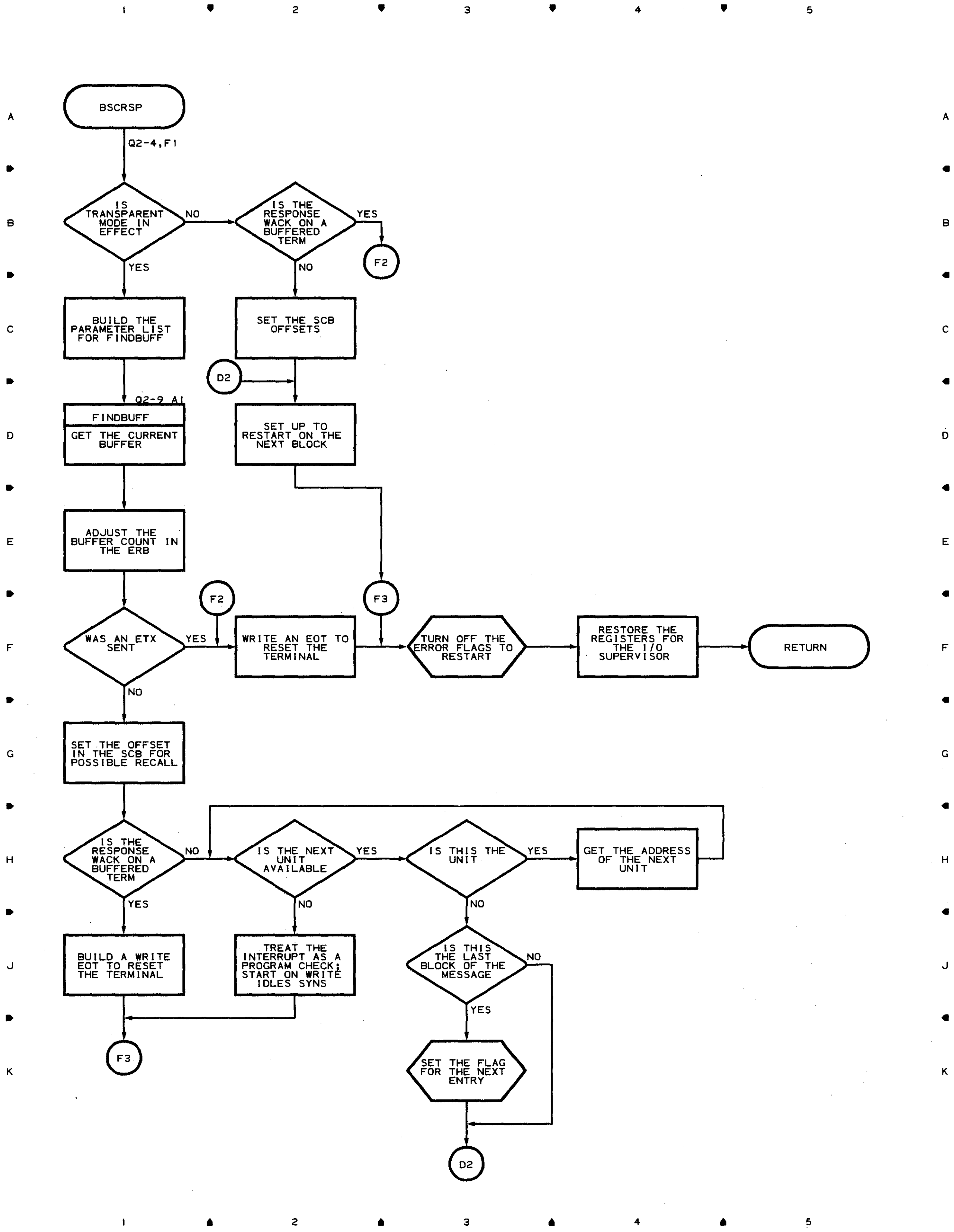
A3
SCAN
Q2-6,A2
Q2-8.G1
RN-1,F3
INITIALIZE THE REGISTERS
IS THIS THE FIRST BLOCK OF THE MESSAGE
YES
GET THE ADDRESS OF THE FIRST BYTE OF DATA
NO
IS TRANSPARENT MODE IN EFFECT
YES
SET THE LCB AS THE SCAN AREA
F1
NO
LCOUT
YES
NO
F1
GET THE FIRST DATA BYTE FROM THE CURRENT CCW
GET THE SCT ADDRESS; POINT TO STX-ENQ
A3
IS THE RESPONSE TTD
YES
SET THE STATUS IN THE SCB AS POSSIBLE ABORT
IS THIS THE FIRST BLOCK
NO
YES
ADJUST HEADER CCW; ACCOUNT LCOUT; SET NONTRANSPARENT BIT
RETURN
NO
IS THE RESPONSE STX
YES
IS THE TRANSPARENT BIT ALREADY SET
NO
E5
C4
YES
SET A FORMAT ERROR - INVALID START
NO
IS THE RESPONSE TRANSPARENT TTD
YES
SET THE STATUS IN THE SCB AS POSSIBLE ABORT
IS THE NON-TRANSPARENT BIT SET
YES
C4
E5
NO
NO
IS THE RESPONSE DLE STX
YES
WAS NON-TRANSPARENT FIRST
YES
C4
NO
IS THIS THE FIRST TIME
NO
YES
INCREMENT THE IDLES COUNT BY 2
IS THIS THE FIRST TIME
NO
YES
INCREMENT THE IDLES COUNT
SET THE 'VALID START' BIT
NO
IS THE RESPONSE ENQ
YES
SET THE 'ENQ START' AND 'VALID START' BITS
NO
IS THE RESPONSE EOT
YES
SET THE 'EOT START' AND 'VALID START' BITS
RETURN
NO
IS THE RESPONSE THE SOH % S SEQUENCE
NO
IS THE RESPONSE THE SOH % SEQUENCE
NO
Q2-12 A1
YES
YES
SET THE 'VALID START' BIT
SET THE 'OLT' AND 'VALID START' BITS
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart Q2-12 LINE END APPENDAGE FOR BSC LINES



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
Q2-12 A1
IS THE RESPONSE SOH
YES
SET THE 'VALID START' BIT
NO
IS THE RESPONSE DLE EOT
YES
SET 'VALID START' & 'EOT'; FORCE DISABLE ON NEXT OP
NO
IS THIS A DIAL LINE
NO
SET THE 'ERROR STATUS-INVALID' BIT
YES
RETURN
IS THIS THE FIRST BLOCK
NO
YES
IS THIS A HEADER BUFFER
NO
YES
WAS ENTRY FROM THE PCI APPENDAGE
YES
NO
SET UP TO RESTART ON WRITE ID ACK
TURN OFF THE ERROR FLAGS TO RESTART
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN
CHACK
Q2-3,B1 Q2-4,C3
Q2-5,A1,A3
Q2-6,F3,G1
INITIALIZE REGISTERS; GET RESPONSE AREA ADDRESS
IS THE RESPONSE ACK-0
YES
SET THE RESPONSE FLAG FOR ACK-0
NO
IS THE RESPONSE ACK-1
YES
SET THE RESPONSE FLAG FOR ACK-1
NO
IS THE RESPONSE ENQ
YES
SET THE RESPONSE FLAG FOR ENQ
NO
IS THE RESPONSE NAK
YES
SET THE RESPONSE FLAG FOR NAK
NO
IS THE RESPONSE EOT
YES
SET THE RESPONSE FLAG FOR EOT
NO
IS THE RESPONSE WACK
YES
SET THE RESPONSE FLAG FOR WACK
NO
IS THE RESPONSE RVI
YES
SET THE RESPONSE FLAG FOR RVI
NO
IS THE RESPONSE DLE EOT
YES
SET THE RESPONSE FLAG FOR DLE EOT
NO
SET THE 'INVALID RESPONSE' FLAG
RETURN

Chart Q2-13 LINE END APPENDAGE FOR BSC LINES



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
BSCRSP
Q2-4,F1
IS TRANSPARENT MODE IN EFFECT
NO
YES
IS THE RESPONSE WACK ON A BUFFERED TERM
YES
NO
F2
BUILD THE PARAMETER LIST FOR FINDBUFF
SET THE SCB OFFSETS
D2
Q2-9 A1
FINDBUFF
GET THE CURRENT BUFFER
SET UP TO RESTART ON THE NEXT BLOCK
ADJUST THE BUFFER COUNT IN THE ERB
F2
F3
WAS AN ETX SENT
YES
NO
WRITE AN EOT TO RESET THE TERMINAL
TURN OFF THE ERROR FLAGS TO RESTART
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN
SET THE OFFSET IN THE SCB FOR POSSIBLE RECALL
IS THE RESPONSE WACK ON A BUFFERED TERM
NO
YES
IS THE NEXT UNIT AVAILABLE
YES
NO
IS THIS THE UNIT
YES
NO
GET THE ADDRESS OF THE NEXT UNIT
BUILD A WRITE EOT TO RESET THE TERMINAL
TREAT THE INTERRUPT AS A PROGRAM CHECK; START ON WRITE IDLES SYNS
IS THIS THE LAST BLOCK OF THE MESSAGE
NO
YES
F3
SET THE FLAG FOR THE NEXT ENTRY
D2

Chart Q3-1 LINE END APPENDAGE FOR START/STOP LINES

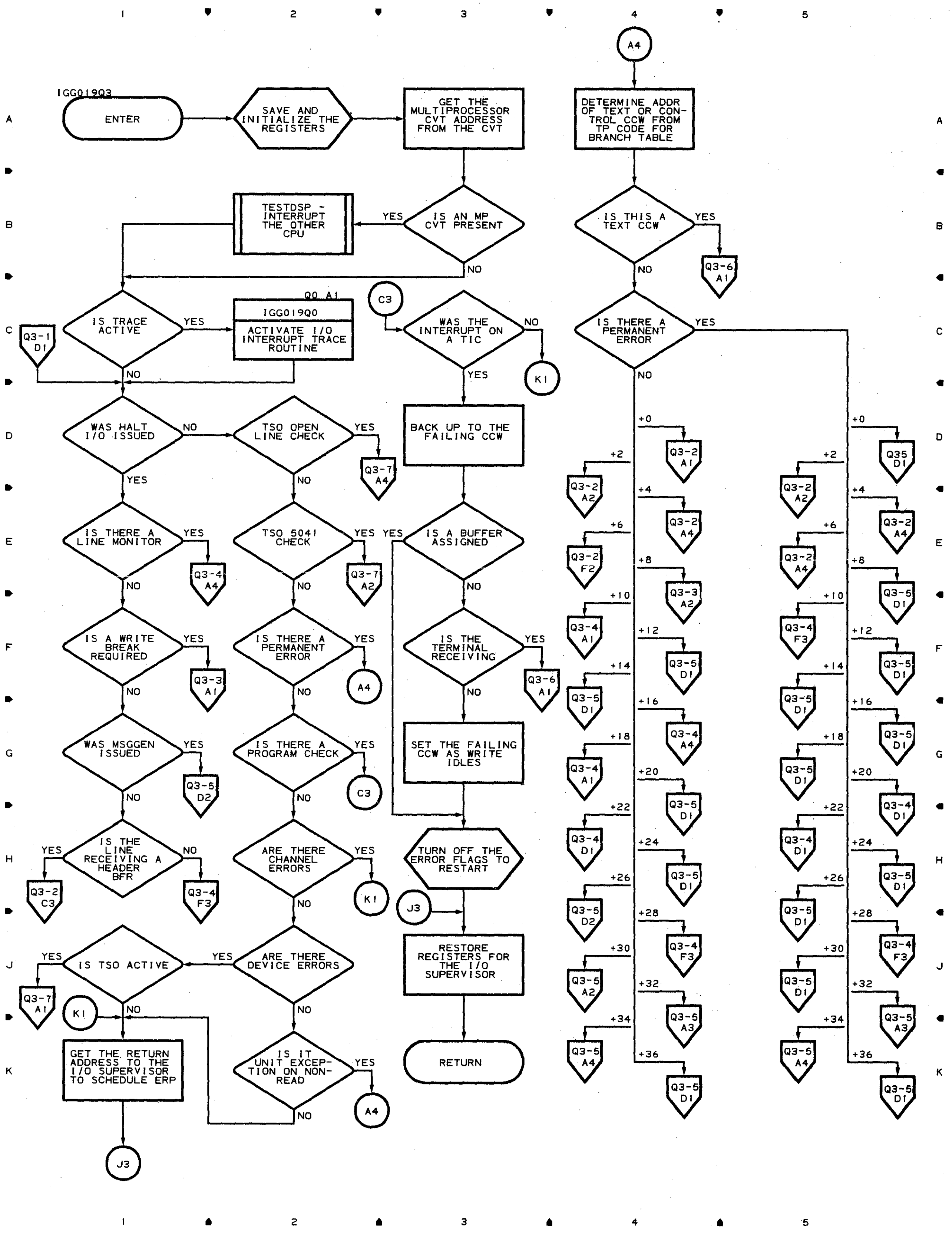
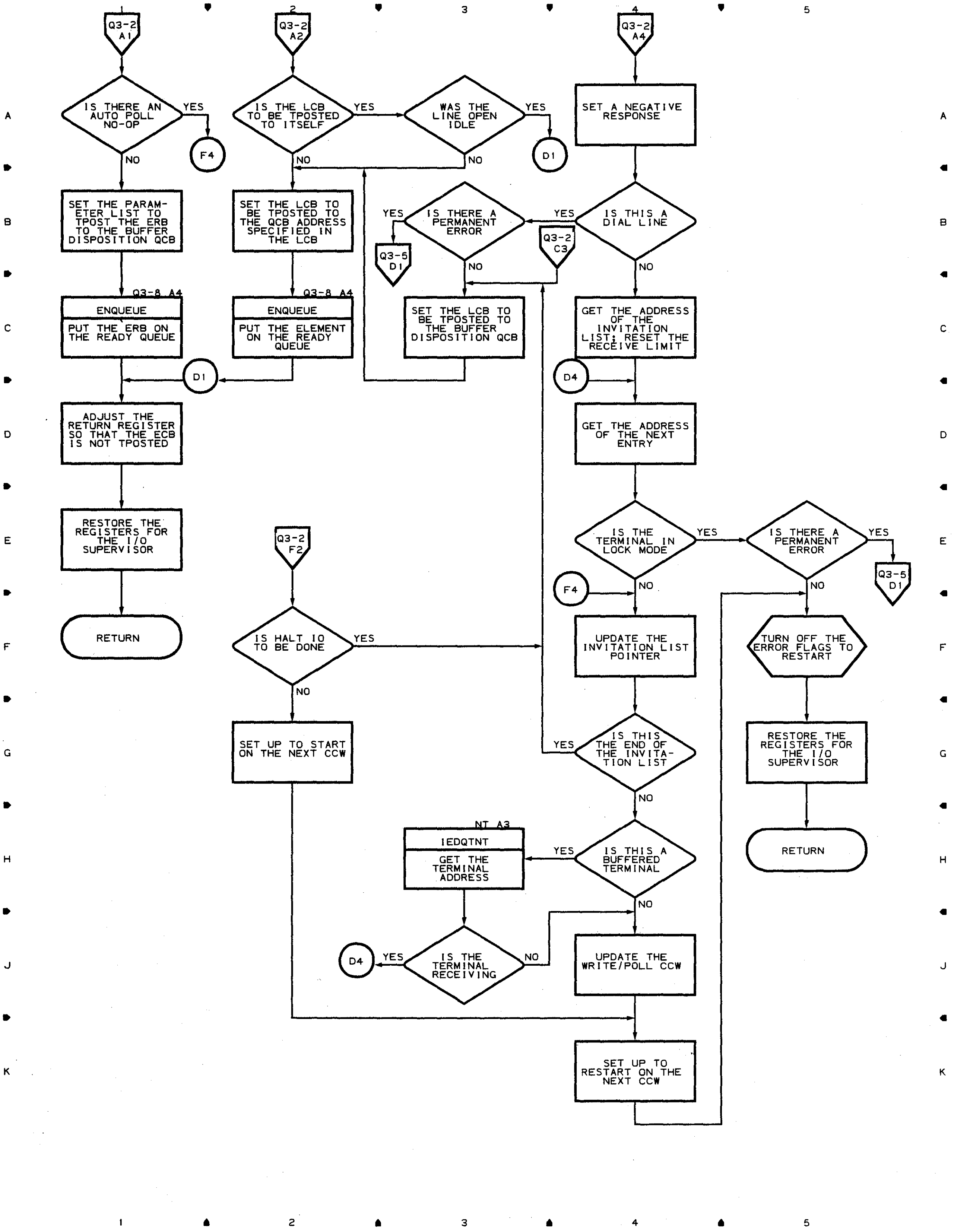


Chart Q3-2 LINE END APPENDAGE FOR START/STOP LINES



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
Q3-2 A1
IS THERE AN AUTO POLL NO-OP
YES
F4
NO
SET THE PARAMETER LIST TO TPOST THE ERB TO THE BUFFER DISPOSITION QCB
Q3-8 A4
ENQUEUE
PUT THE ERB ON THE READY QUEUE
D1
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT TPOSTED
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN
Q3-2 A2
IS THE LCB TO BE TPOSTED TO ITSELF
YES
NO
SET THE LCB TO BE TPOSTED TO THE QCB ADDRESS SPECIFIED IN THE LCB
Q3-8 A4
ENQUEUE
PUT THE ELEMENT ON THE READY QUEUE
WAS THE LINE OPEN IDLE
YES
D1
NO
IS THERE A PERMANENT ERROR
YES
Q3-5 D1
NO
SET THE LCB TO BE TPOSTED TO THE BUFFER DISPOSITION QCB
Q3-2 C3
Q3-2 F2
IS HALT IO TO BE DONE
YES
NO
SET UP TO START ON THE NEXT CCW
NT A3
IEDQTNT
GET THE TERMINAL ADDRESS
D4
YES
IS THE TERMINAL RECEIVING
NO
Q3-2 A4
SET A NEGATIVE RESPONSE
IS THIS A DIAL LINE
YES
NO
GET THE ADDRESS OF THE INVITATION LIST; RESET THE RECEIVE LIMIT
D4
GET THE ADDRESS OF THE NEXT ENTRY
IS THE TERMINAL IN LOCK MODE
YES
NO
F4
UPDATE THE INVITATION LIST POINTER
IS THIS THE END OF THE INVITATION LIST
YES
NO
IS THIS A BUFFERED TERMINAL
YES
NO
UPDATE THE WRITE/POLL CCW
SET UP TO RESTART ON THE NEXT CCW
IS THERE A PERMANENT ERROR
YES
Q3-5 D1
NO
TURN OFF THE ERROR FLAGS TO RESTART
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN

Chart Q3-3 LINE END APPENDAGE FOR START/STOP LINES

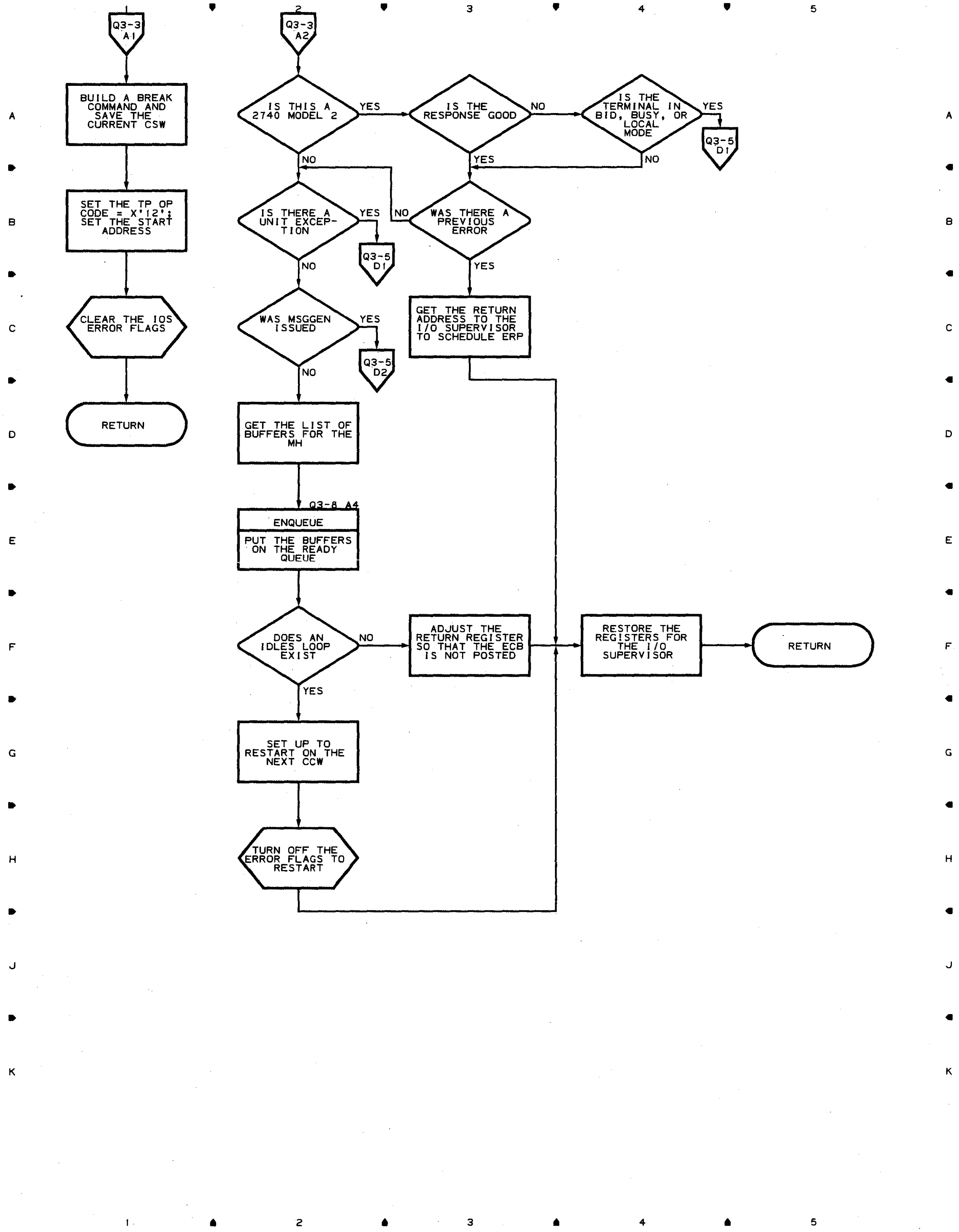

Q3-3 A1
BUILD A BREAK COMMAND AND SAVE THE CURRENT CSW
SET THE TP OP CODE = X'12'; SET THE START ADDRESS
CLEAR THE IOS ERROR FLAGS
RETURN
Q3-3 A2
IS THIS A 2740 MODEL 2
YES
NO
IS THE RESPONSE GOOD
NO
YES
IS THE TERMINAL IN BID, BUSY, OR LOCAL MODE
YES
NO
Q3-5 D1
IS THERE A UNIT EXCEPTION
YES
NO
Q3-5 D1
WAS THERE A PREVIOUS ERROR
NO
YES
WAS MSGGEN ISSUED
YES
NO
Q3-5 D2
GET THE RETURN ADDRESS TO THE I/O SUPERVISOR TO SCHEDULE ERP
GET THE LIST OF BUFFERS FOR THE MH
Q3-8 A4
ENQUEUE
PUT THE BUFFERS ON THE READY QUEUE
DOES AN IDLES LOOP EXIST
NO
YES
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN
SET UP TO RESTART ON THE NEXT CCW
TURN OFF THE ERROR FLAGS TO RESTART
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart Q3-4 LINE END APPENDAGE FOR START/STOP LINES

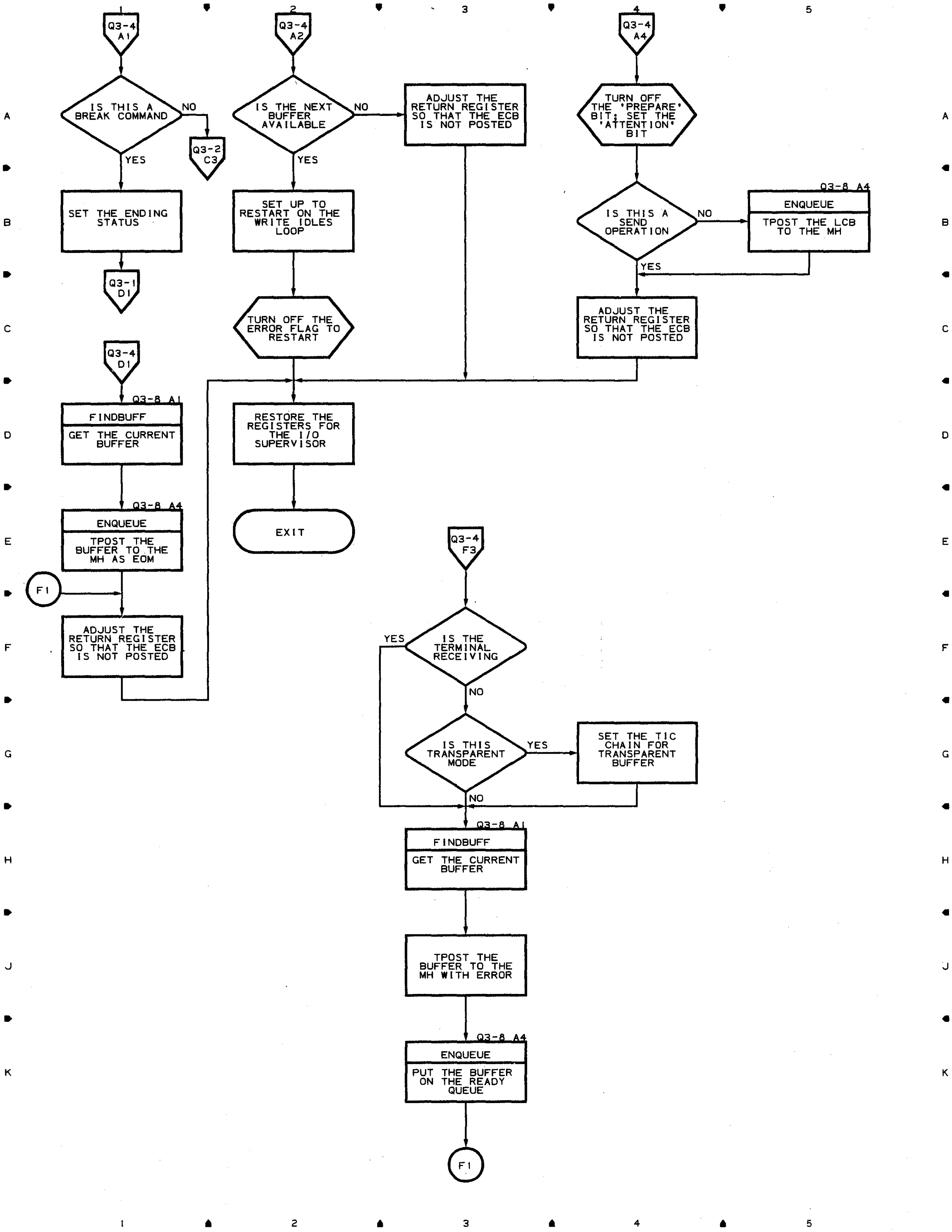
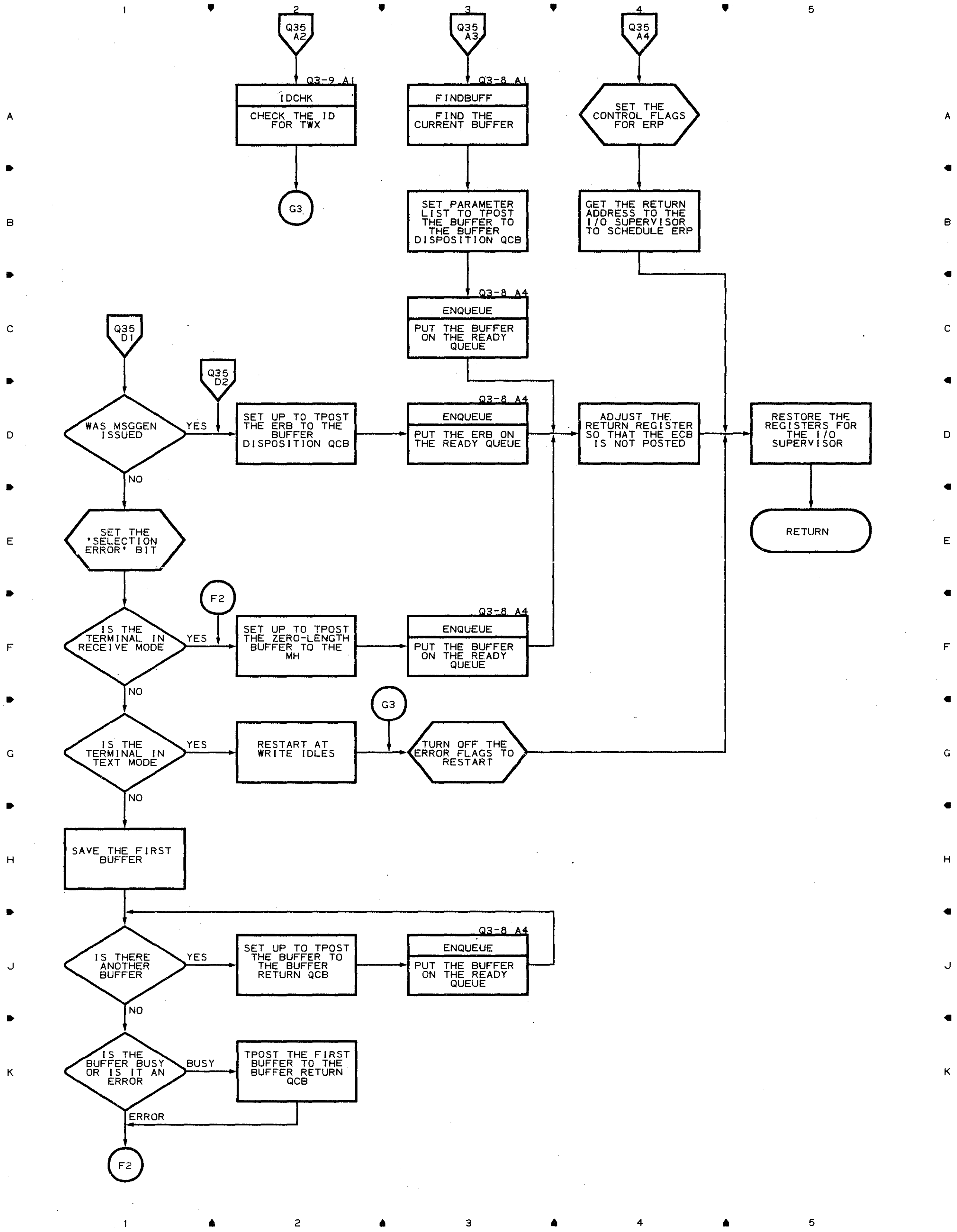

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
Q3-4 A1
IS THIS A BREAK COMMAND
NO
Q3-2 C3
YES
SET THE ENDING STATUS
Q3-1 D1
Q3-4 A2
IS THE NEXT BUFFER AVAILABLE
NO
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
YES
SET UP TO RESTART ON THE WRITE IDLES LOOP
TURN OFF THE ERROR FLAG TO RESTART
Q3-4 A4
TURN OFF THE 'PREPARE' BIT; SET THE 'ATTENTION' BIT
IS THIS A SEND OPERATION
NO
Q3-8 A4
ENQUEUE
TPOST THE LCB TO THE MH
YES
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
Q3-4 D1
Q3-8 A1
FINDBUFF
GET THE CURRENT BUFFER
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
Q3-8 A4
ENQUEUE
TPOST THE BUFFER TO THE MH AS EOM
EXIT
Q3-4 F3
F1
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
YES
IS THE TERMINAL RECEIVING
NO
IS THIS TRANSPARENT MODE
YES
SET THE TIC CHAIN FOR TRANSPARENT BUFFER
NO
Q3-8 A1
FINDBUFF
GET THE CURRENT BUFFER
TPOST THE BUFFER TO THE MH WITH ERROR
Q3-8 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
F1

Chart Q3-5 LINE END APPENDAGE FOR START/STOP LINES



1
2
3
4
5
Q35 A2
Q35 A3
Q35 A4
Q3-9 A1
IDCHK
CHECK THE ID FOR TWX
Q3-8 A1
FINDBUFF
FIND THE CURRENT BUFFER
SET THE CONTROL FLAGS FOR ERP
A
G3
SET PARAMETER LIST TO TPOST THE BUFFER TO THE BUFFER DISPOSITION QCB
GET THE RETURN ADDRESS TO THE I/O SUPERVISOR TO SCHEDULE ERP
B
Q3-8 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
C
Q35 D1
Q35 D2
WAS MSGGEN ISSUED
YES
SET UP TO TPOST THE ERB TO THE BUFFER DISPOSITION QCB
Q3-8 A4
ENQUEUE
PUT THE ERB ON THE READY QUEUE
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
D
NO
SET THE 'SELECTION ERROR' BIT
RETURN
E
F2
IS THE TERMINAL IN RECEIVE MODE
YES
SET UP TO TPOST THE ZERO-LENGTH BUFFER TO THE MH
Q3-8 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
F
NO
G3
IS THE TERMINAL IN TEXT MODE
YES
RESTART AT WRITE IDLES
TURN OFF THE ERROR FLAGS TO RESTART
G
NO
SAVE THE FIRST BUFFER
H
IS THERE ANOTHER BUFFER
YES
SET UP TO TPOST THE BUFFER TO THE BUFFER RETURN QCB
Q3-8 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
J
NO
IS THE BUFFER BUSY OR IS IT AN ERROR
BUSY
TPOST THE FIRST BUFFER TO THE BUFFER RETURN QCB
K
ERROR
F2
1
2
3
4
5

Chart Q3-6 LINE END APPENDAGE FOR START/STOP LINES

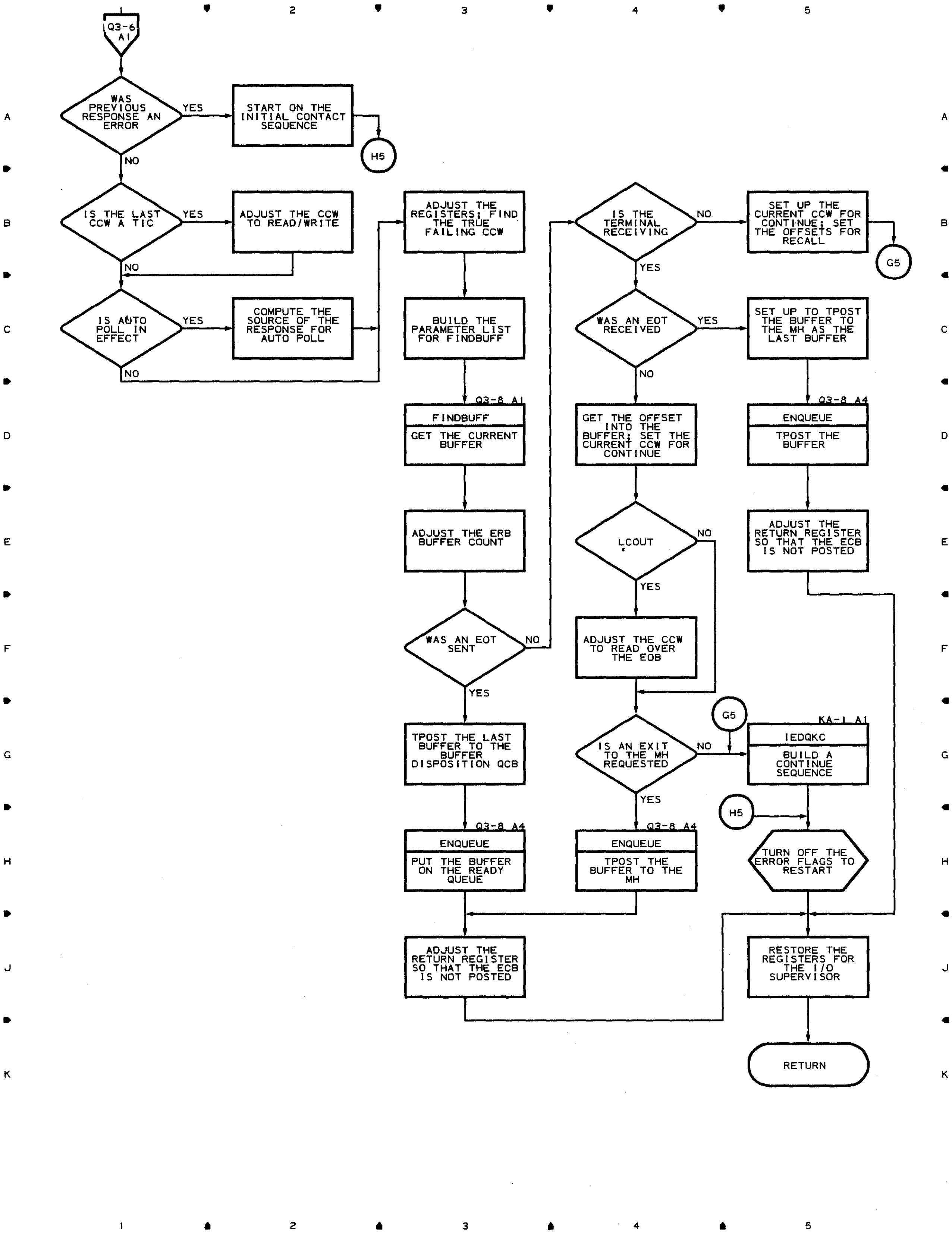

Q3-6 A1
WAS PREVIOUS RESPONSE AN ERROR
YES
START ON THE INITIAL CONTACT SEQUENCE
H5
NO
IS THE LAST CCW A TIC
YES
ADJUST THE CCW TO READ/WRITE
NO
IS AUTO POLL IN EFFECT
YES
COMPUTE THE SOURCE OF THE RESPONSE FOR AUTO POLL
NO
ADJUST THE REGISTERS; FIND THE TRUE FAILING CCW
BUILD THE PARAMETER LIST FOR FINDBUFF
Q3-8 A1
FINDBUFF
GET THE CURRENT BUFFER
ADJUST THE ERB BUFFER COUNT
WAS AN EOT SENT
NO
YES
TPOST THE LAST BUFFER TO THE BUFFER DISPOSITION QCB
Q3-8 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
IS THE TERMINAL RECEIVING
NO
YES
WAS AN EOT RECEIVED
YES
NO
GET THE OFFSET INTO THE BUFFER; SET THE CURRENT CCW FOR CONTINUE
LCOUT
NO
YES
ADJUST THE CCW TO READ OVER THE EOB
IS AN EXIT TO THE MH REQUESTED
NO
YES
Q3-8 A4
ENQUEUE
TPOST THE BUFFER TO THE MH
SET UP THE CURRENT CCW FOR CONTINUE; SET THE OFFSETS FOR RECALL
G5
SET UP TO TPOST THE BUFFER TO THE MH AS THE LAST BUFFER
Q3-8 A4
ENQUEUE
TPOST THE BUFFER
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
G5
KA-1 A1
IEDQKC
BUILD A CONTINUE SEQUENCE
H5
TURN OFF THE ERROR FLAGS TO RESTART
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart Q3-7 LINE END APPENDAGE FOR START/STOP LINES

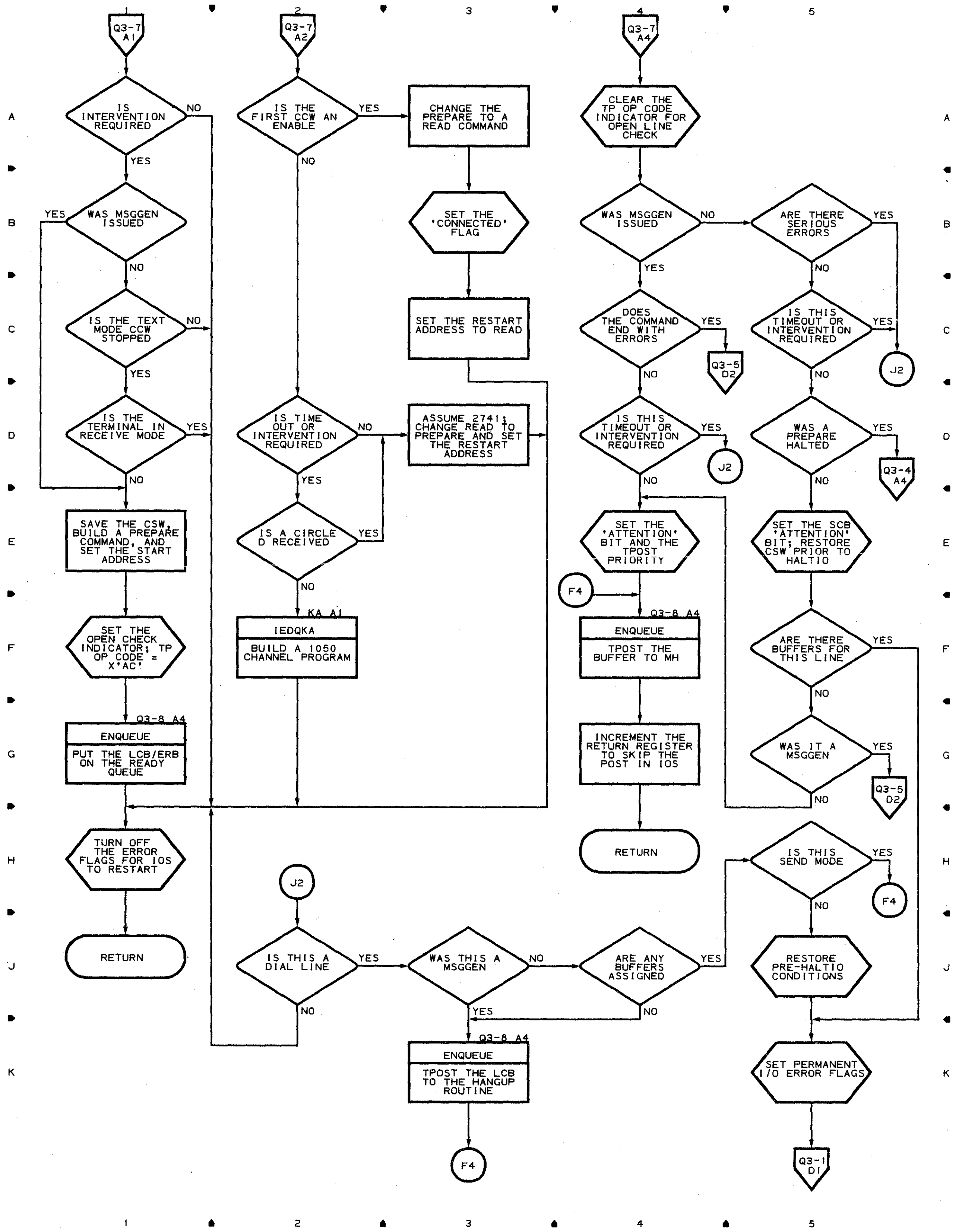

Q3-7 A1
IS INTERVENTION REQUIRED
NO
YES
WAS MSGGEN ISSUED
YES
NO
IS THE TEXT MODE CCW STOPPED
NO
YES
IS THE TERMINAL IN RECEIVE MODE
YES
NO
SAVE THE CSW, BUILD A PREPARE COMMAND, AND SET THE START ADDRESS
SET THE OPEN CHECK INDICATOR; TP OP CODE = X'AC'
Q3-8 A4
ENQUEUE
PUT THE LCB/ERB ON THE READY QUEUE
TURN OFF THE ERROR FLAGS FOR IOS TO RESTART
RETURN
Q3-7 A2
IS THE FIRST CCW AN ENABLE
YES
NO
IS TIME OUT OR INTERVENTION REQUIRED
NO
YES
IS A CIRCLE D RECEIVED
YES
NO
KA A1
IEDQKA
BUILD A 1050 CHANNEL PROGRAM
J2
IS THIS A DIAL LINE
YES
NO
CHANGE THE PREPARE TO A READ COMMAND
SET THE 'CONNECTED' FLAG
SET THE RESTART ADDRESS TO READ
ASSUME 2741; CHANGE READ TO PREPARE AND SET THE RESTART ADDRESS
WAS THIS A MSGGEN
NO
YES
Q3-8 A4
ENQUEUE
TPOST THE LCB TO THE HANGUP ROUTINE
F4
Q3-7 A4
CLEAR THE TP OP CODE INDICATOR FOR OPEN LINE CHECK
WAS MSGGEN ISSUED
NO
YES
DOES THE COMMAND END WITH ERRORS
YES
Q3-5 D2
NO
IS THIS TIMEOUT OR INTERVENTION REQUIRED
YES
J2
NO
SET THE 'ATTENTION' BIT AND THE TPOST PRIORITY
F4
Q3-8 A4
ENQUEUE
TPOST THE BUFFER TO MH
INCREMENT THE RETURN REGISTER TO SKIP THE POST IN IOS
RETURN
ARE ANY BUFFERS ASSIGNED
YES
NO
ARE THERE SERIOUS ERRORS
YES
NO
IS THIS TIMEOUT OR INTERVENTION REQUIRED
YES
J2
NO
WAS A PREPARE HALTED
YES
Q3-4 A4
NO
SET THE SCB 'ATTENTION' BIT; RESTORE CSW PRIOR TO HALTIO
ARE THERE BUFFERS FOR THIS LINE
YES
NO
WAS IT A MSGGEN
YES
Q3-5 D2
NO
IS THIS SEND MODE
YES
F4
NO
RESTORE PRE-HALTIO CONDITIONS
SET PERMANENT I/O ERROR FLAGS
Q3-1 D1
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart Q3-8 LINE END APPENDAGE FOR START/STOP LINES

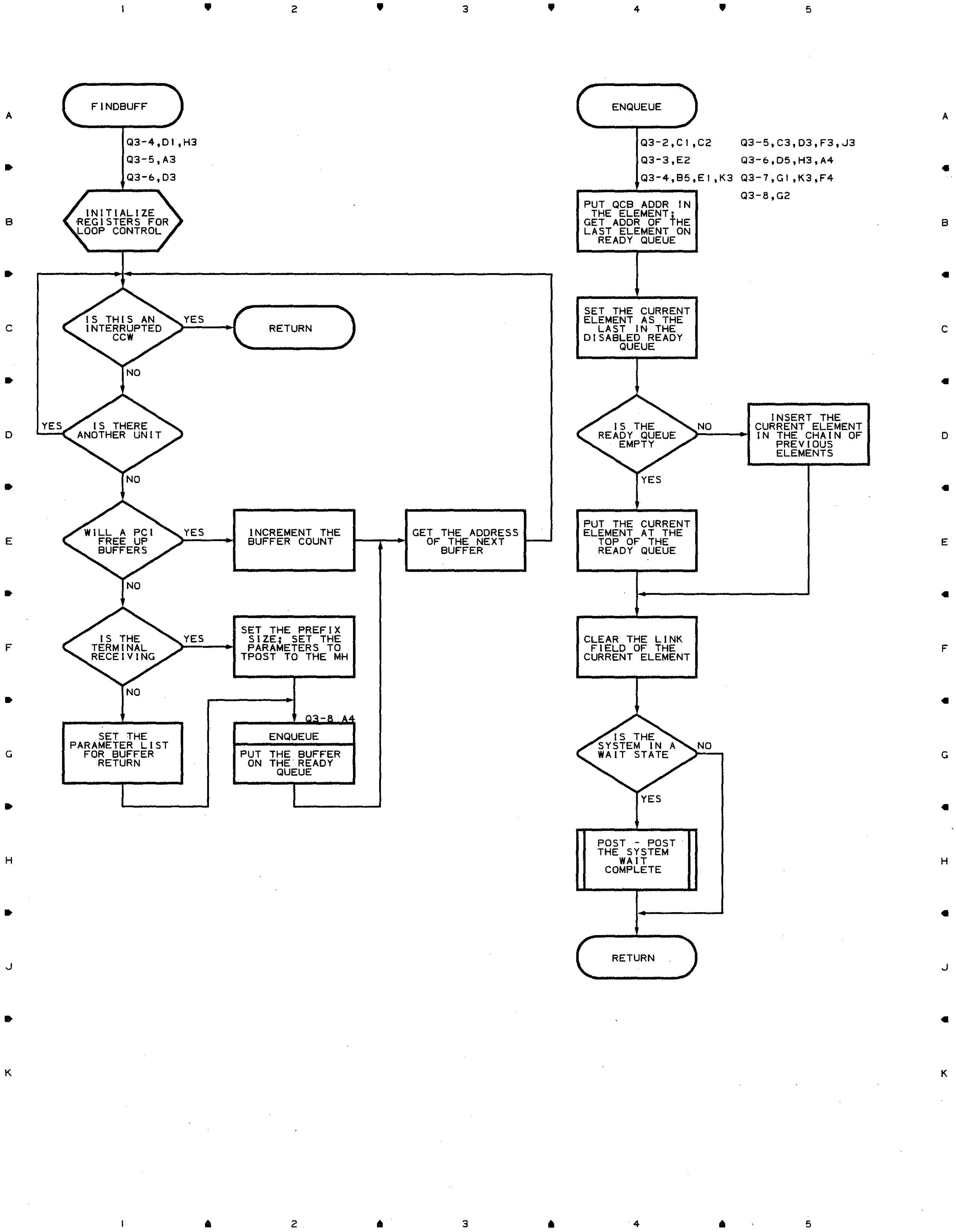
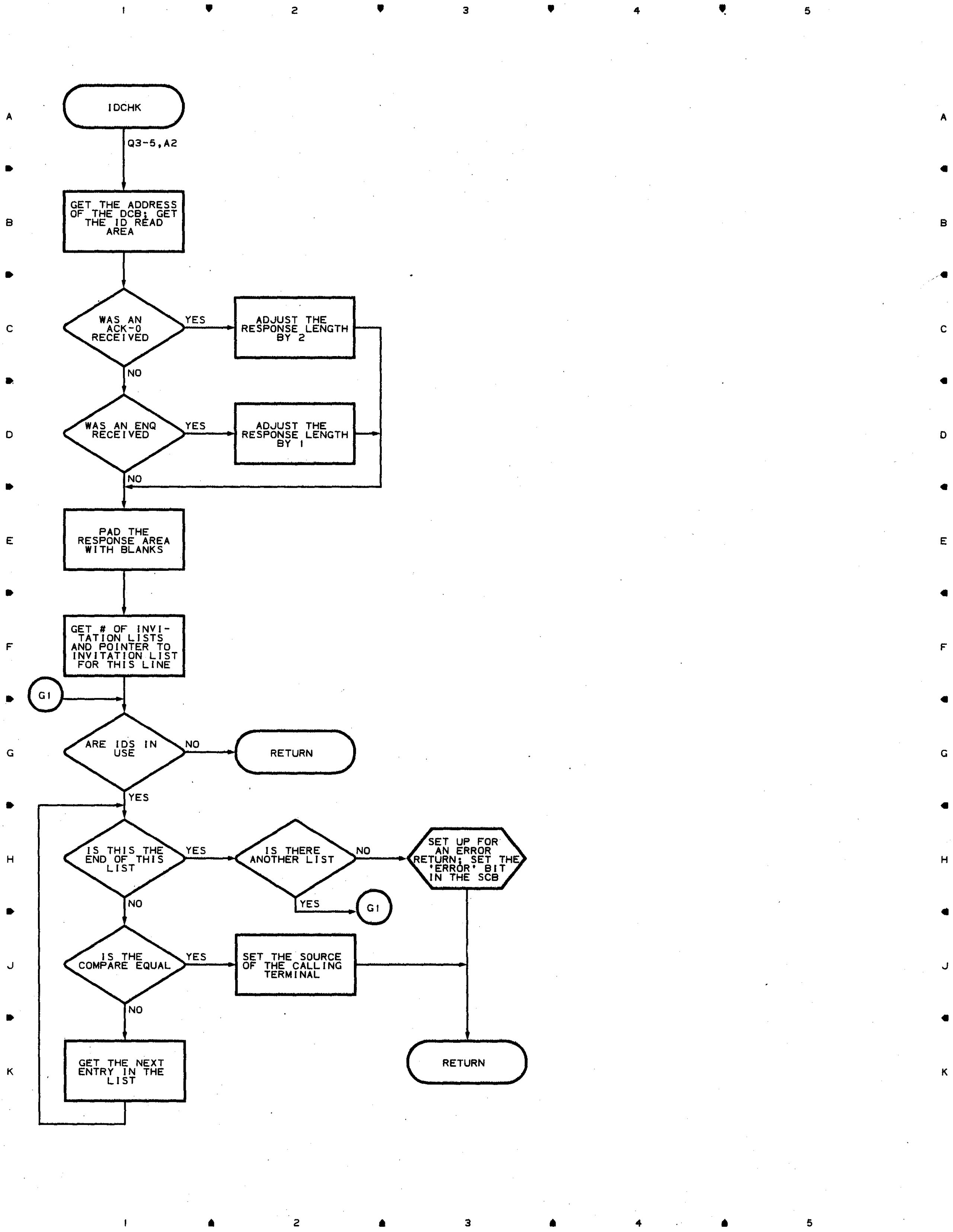

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
FINDBUFF
Q3-4,D1,H3
Q3-5,A3
Q3-6,D3
INITIALIZE REGISTERS FOR LOOP CONTROL
IS THIS AN INTERRUPTED CCW
YES
RETURN
NO
IS THERE ANOTHER UNIT
YES
NO
WILL A PCI FREE UP BUFFERS
YES
INCREMENT THE BUFFER COUNT
GET THE ADDRESS OF THE NEXT BUFFER
NO
IS THE TERMINAL RECEIVING
YES
SET THE PREFIX SIZE; SET THE PARAMETERS TO TPOST TO THE MH
NO
SET THE PARAMETER LIST FOR BUFFER RETURN
Q3-8 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
ENQUEUE
Q3-2,C1,C2
Q3-3,E2
Q3-4,B5,E1,K3
Q3-5,C3,D3,F3,J3
Q3-6,D5,H3,A4
Q3-7,G1,K3,F4
Q3-8,G2
PUT QCB ADDR IN THE ELEMENT; GET ADDR OF THE LAST ELEMENT ON READY QUEUE
SET THE CURRENT ELEMENT AS THE LAST IN THE DISABLED READY QUEUE
IS THE READY QUEUE EMPTY
NO
INSERT THE CURRENT ELEMENT IN THE CHAIN OF PREVIOUS ELEMENTS
YES
PUT THE CURRENT ELEMENT AT THE TOP OF THE READY QUEUE
CLEAR THE LINK FIELD OF THE CURRENT ELEMENT
IS THE SYSTEM IN A WAIT STATE
NO
YES
POST - POST THE SYSTEM WAIT COMPLETE
RETURN

Chart Q3-9 LINE END APPENDAGE FOR START/STOP LINES



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IDCHK
Q3-5,A2
GET THE ADDRESS OF THE DCB; GET THE ID READ AREA
WAS AN ACK-0 RECEIVED
YES
ADJUST THE RESPONSE LENGTH BY 2
NO
WAS AN ENQ RECEIVED
YES
ADJUST THE RESPONSE LENGTH BY 1
NO
PAD THE RESPONSE AREA WITH BLANKS
GET # OF INVITATION LISTS AND POINTER TO INVITATION LIST FOR THIS LINE
G1
ARE IDS IN USE
NO
RETURN
YES
IS THIS THE END OF THIS LIST
YES
IS THERE ANOTHER LIST
NO
SET UP FOR AN ERROR RETURN; SET THE 'ERROR' BIT IN THE SCB
YES
G1
NO
IS THE COMPARE EQUAL
YES
SET THE SOURCE OF THE CALLING TERMINAL
NO
GET THE NEXT ENTRY IN THE LIST
RETURN

Chart Q4-1 LINE END APPENDAGE FOR LEASED & START/STOP LINES WITH NO TSO

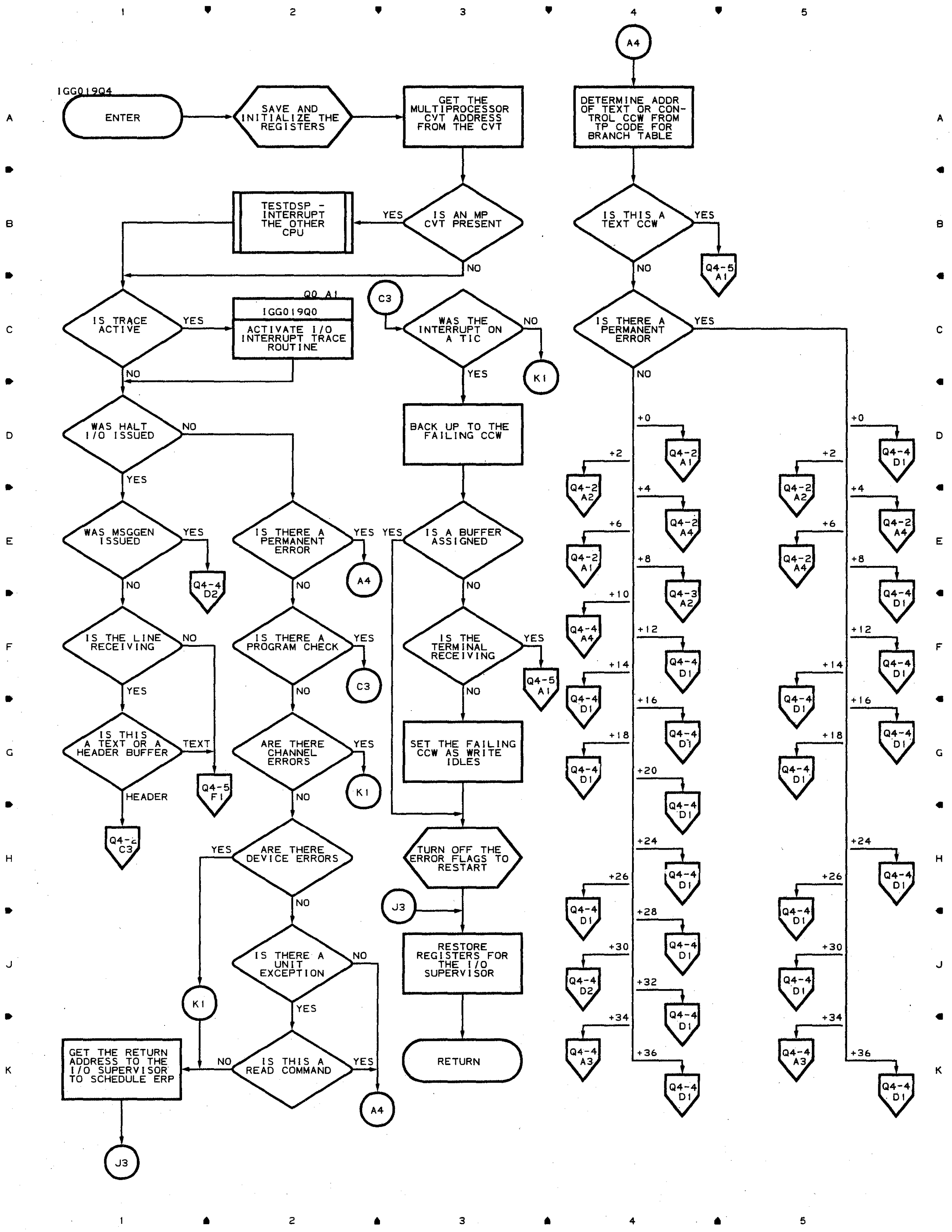
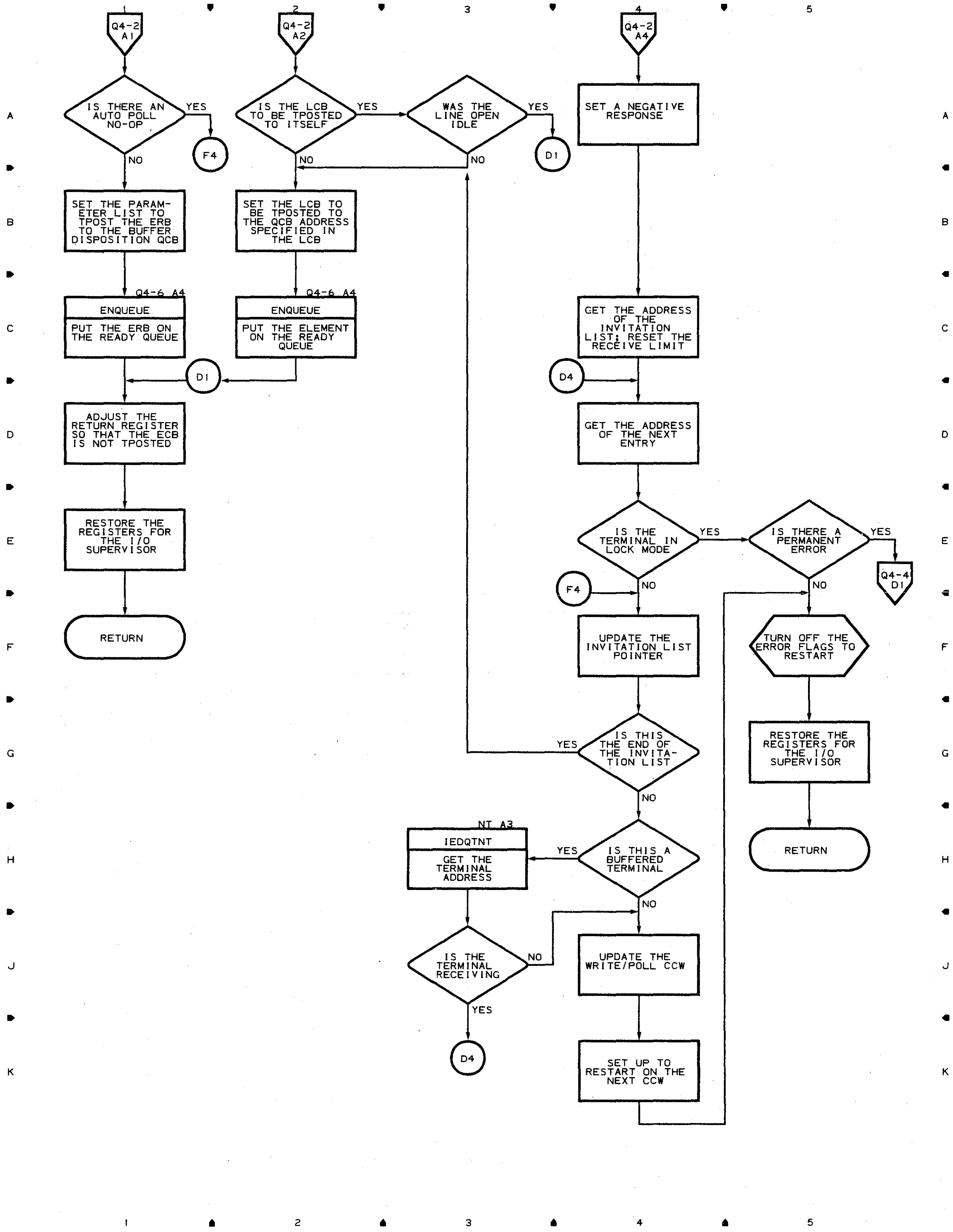


Chart Q4-2 LINE END APPENDAGE FOR LEASED & START/STOP LINES WITH NO TSO



1
2
3
4
5
Q4-2 A1
Q4-2 A2
Q4-2 A4
A
IS THERE AN AUTO POLL NO-OP
YES
NO
F4
IS THE LCB TO BE TPOSTED TO ITSELF
YES
NO
WAS THE LINE OPEN IDLE
YES
NO
D1
SET A NEGATIVE RESPONSE
B
SET THE PARAMETER LIST TO TPOST THE ERB TO THE BUFFER DISPOSITION QCB
SET THE LCB TO BE TPOSTED TO THE QCB ADDRESS SPECIFIED IN THE LCB
C
Q4-6 A4
ENQUEUE
PUT THE ERB ON THE READY QUEUE
Q4-6 A4
ENQUEUE
PUT THE ELEMENT ON THE READY QUEUE
GET THE ADDRESS OF THE INVITATION LIST; RESET THE RECEIVE LIMIT
D1
D4
D
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT TPOSTED
GET THE ADDRESS OF THE NEXT ENTRY
E
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
IS THE TERMINAL IN LOCK MODE
YES
NO
IS THERE A PERMANENT ERROR
YES
NO
Q4-4 D1
F4
F
RETURN
UPDATE THE INVITATION LIST POINTER
TURN OFF THE ERROR FLAGS TO RESTART
G
IS THIS THE END OF THE INVITATION LIST
YES
NO
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
H
NT A3
IEDQTNT
GET THE TERMINAL ADDRESS
YES
IS THIS A BUFFERED TERMINAL
NO
RETURN
J
IS THE TERMINAL RECEIVING
NO
YES
UPDATE THE WRITE/POLL CCW
K
D4
SET UP TO RESTART ON THE NEXT CCW
1
2
3
4
5

Chart Q4-3 LINE END APPENDAGE FOR LEASED & START/STOP LINES WITH NO TSO

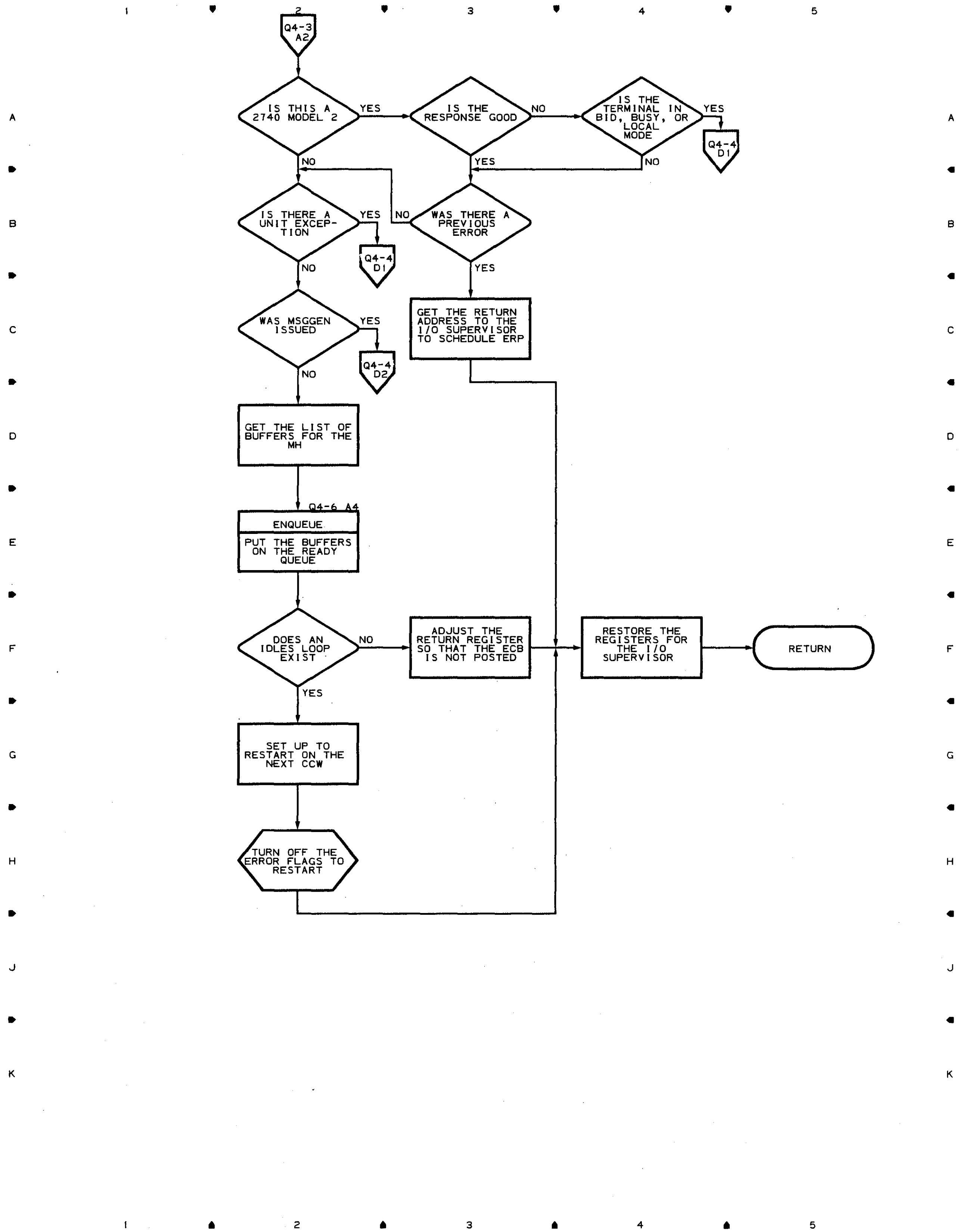

Q4-3 A2
IS THIS A 2740 MODEL 2
YES
NO
IS THE RESPONSE GOOD
NO
YES
IS THE TERMINAL IN BID, BUSY, OR LOCAL MODE
YES
NO
Q4-4 D1
IS THERE A UNIT EXCEPTION
YES
NO
Q4-4 D1
WAS THERE A PREVIOUS ERROR
NO
YES
WAS MSGGEN ISSUED
YES
NO
Q4-4 D2
GET THE RETURN ADDRESS TO THE I/O SUPERVISOR TO SCHEDULE ERP
GET THE LIST OF BUFFERS FOR THE MH
Q4-6 A4
ENQUEUE
PUT THE BUFFERS ON THE READY QUEUE
DOES AN IDLES LOOP EXIST
NO
YES
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN
SET UP TO RESTART ON THE NEXT CCW
TURN OFF THE ERROR FLAGS TO RESTART
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart Q4-4 LINE END APPENDAGE FOR LEASED & START/STOP LINES WITH NO TSO

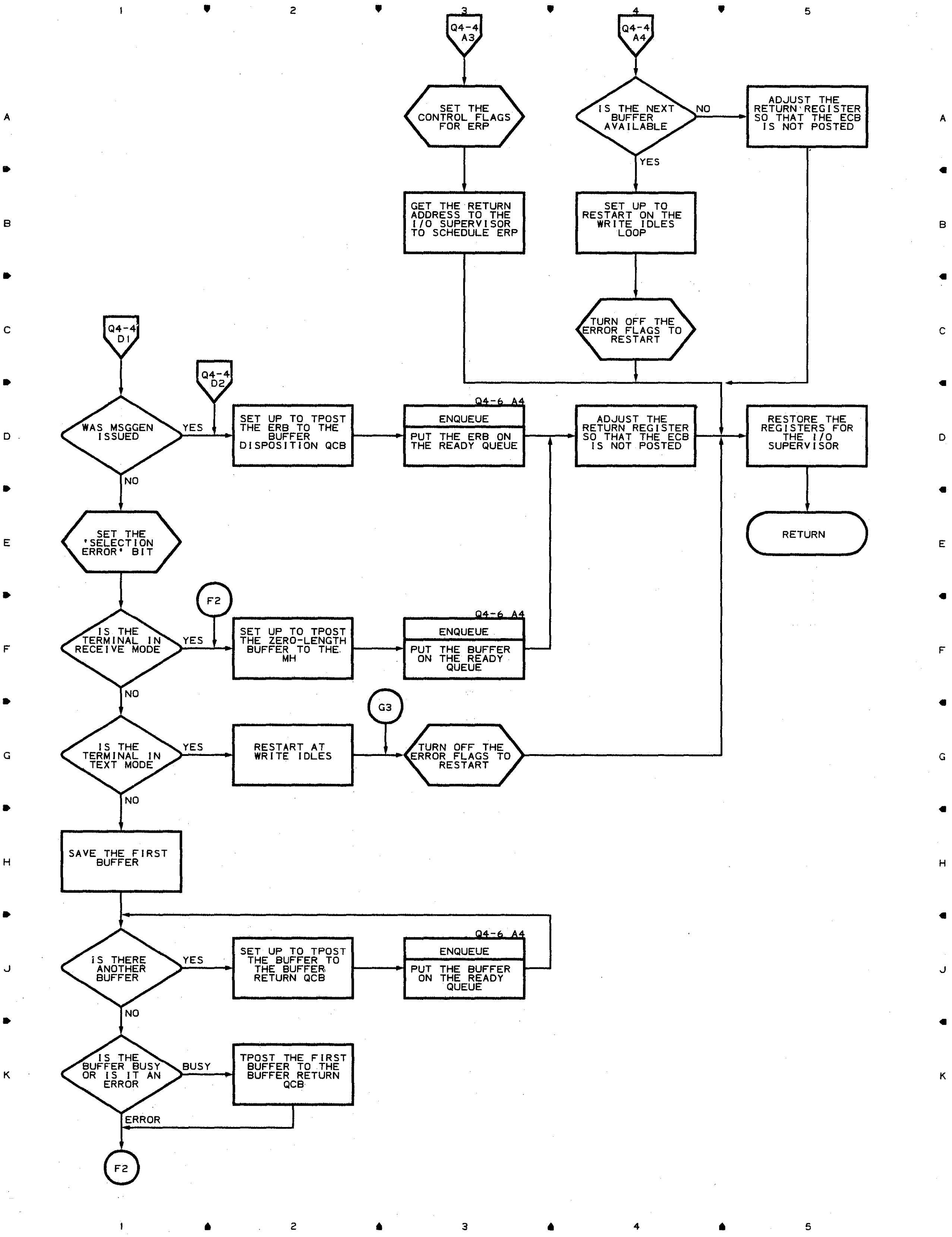
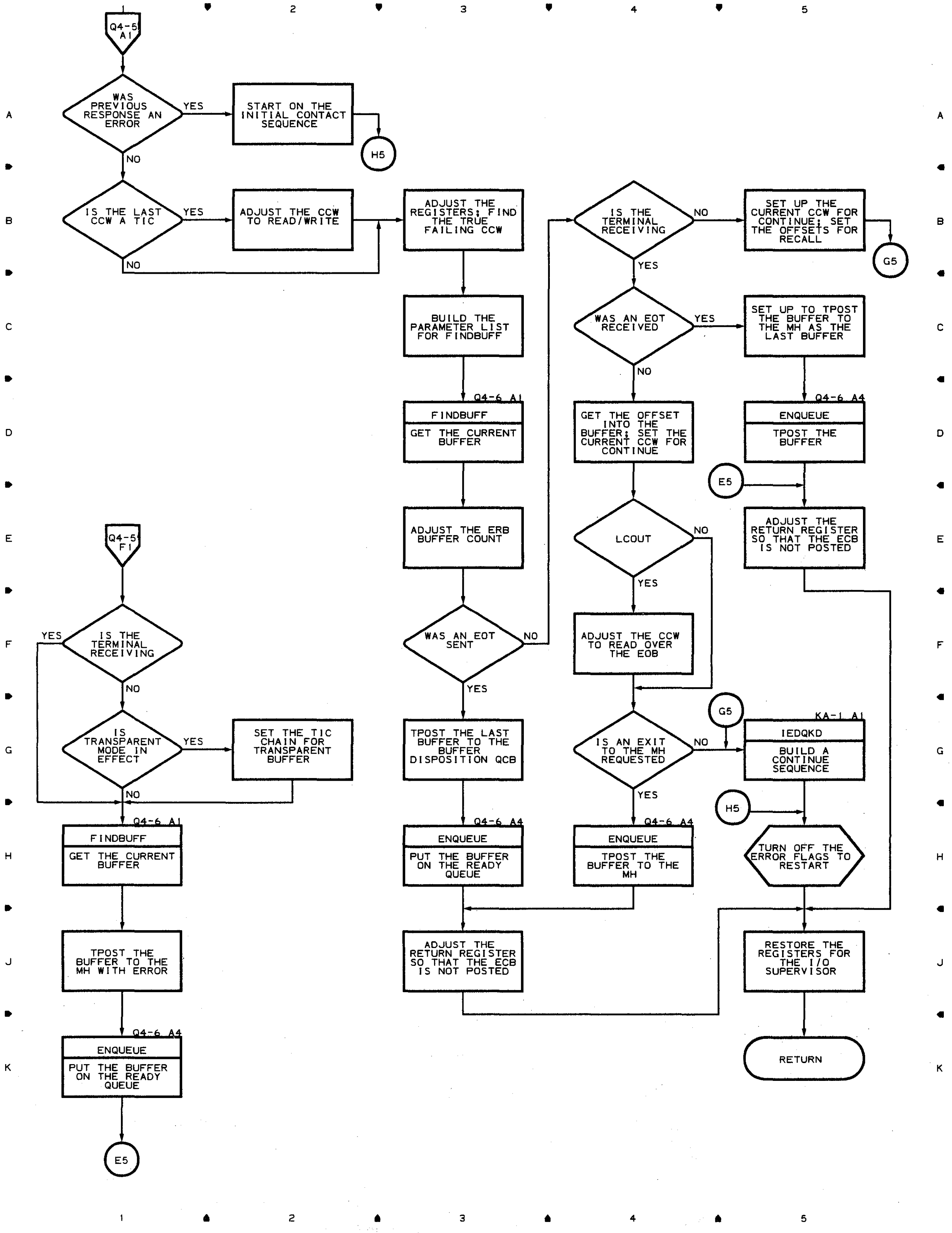

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
Q4-4 A3
Q4-4 A4
SET THE CONTROL FLAGS FOR ERP
IS THE NEXT BUFFER AVAILABLE
NO
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
YES
GET THE RETURN ADDRESS TO THE I/O SUPERVISOR TO SCHEDULE ERP
SET UP TO RESTART ON THE WRITE IDLES LOOP
TURN OFF THE ERROR FLAGS TO RESTART
Q4-4 D1
Q4-4 D2
WAS MSGGEN ISSUED
YES
SET UP TO TPOST THE ERB TO THE BUFFER DISPOSITION QCB
Q4-6 A4
ENQUEUE
PUT THE ERB ON THE READY QUEUE
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
NO
SET THE 'SELECTION ERROR' BIT
RETURN
F2
IS THE TERMINAL IN RECEIVE MODE
YES
SET UP TO TPOST THE ZERO-LENGTH BUFFER TO THE MH
Q4-6 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
NO
G3
IS THE TERMINAL IN TEXT MODE
YES
RESTART AT WRITE IDLES
TURN OFF THE ERROR FLAGS TO RESTART
NO
SAVE THE FIRST BUFFER
IS THERE ANOTHER BUFFER
YES
SET UP TO TPOST THE BUFFER TO THE BUFFER RETURN QCB
Q4-6 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
NO
IS THE BUFFER BUSY OR IS IT AN ERROR
BUSY
TPOST THE FIRST BUFFER TO THE BUFFER RETURN QCB
ERROR
F2
1
2
3
4
5

Chart Q4-5 LINE END APPENDAGE FOR LEASED & START/STOP LINES WITH NO TSO



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
Q4-5 A1
WAS PREVIOUS RESPONSE AN ERROR
YES
START ON THE INITIAL CONTACT SEQUENCE
H5
NO
IS THE LAST CCW A TIC
YES
ADJUST THE CCW TO READ/WRITE
NO
ADJUST THE REGISTERS; FIND THE TRUE FAILING CCW
BUILD THE PARAMETER LIST FOR FINDBUFF
Q4-6 A1
FINDBUFF
GET THE CURRENT BUFFER
ADJUST THE ERB BUFFER COUNT
WAS AN EOT SENT
NO
YES
TPOST THE LAST BUFFER TO THE BUFFER DISPOSITION QCB
Q4-6 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
IS THE TERMINAL RECEIVING
NO
SET UP THE CURRENT CCW FOR CONTINUE; SET THE OFFSETS FOR RECALL
G5
YES
WAS AN EOT RECEIVED
YES
SET UP TO TPOST THE BUFFER TO THE MH AS THE LAST BUFFER
NO
Q4-6 A4
ENQUEUE
TPOST THE BUFFER
GET THE OFFSET INTO THE BUFFER; SET THE CURRENT CCW FOR CONTINUE
E5
LCOUT
NO
YES
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
ADJUST THE CCW TO READ OVER THE EOB
G5
IS AN EXIT TO THE MH REQUESTED
NO
KA-1 A1
IEDQKD
BUILD A CONTINUE SEQUENCE
YES
Q4-6 A4
ENQUEUE
TPOST THE BUFFER TO THE MH
H5
TURN OFF THE ERROR FLAGS TO RESTART
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN
Q4-5 F1
IS THE TERMINAL RECEIVING
YES
NO
IS TRANSPARENT MODE IN EFFECT
YES
SET THE TIC CHAIN FOR TRANSPARENT BUFFER
NO
Q4-6 A1
FINDBUFF
GET THE CURRENT BUFFER
TPOST THE BUFFER TO THE MH WITH ERROR
Q4-6 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
E5

Chart Q4-6 LINE END APPENDAGE FOR LEASED & START/STOP LINES WITH NO TSO

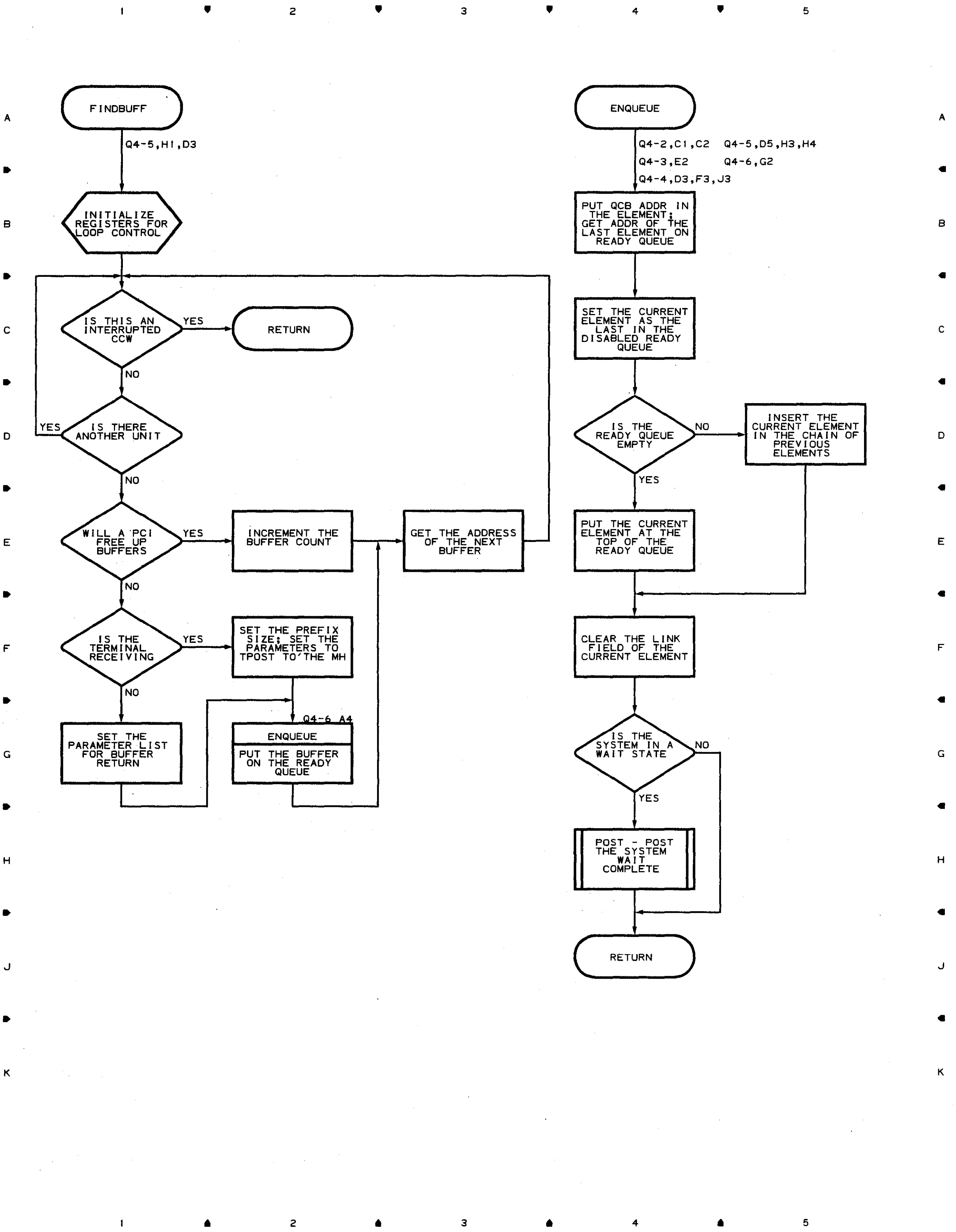

FINDBUFF
Q4-5,H1,D3
INITIALIZE REGISTERS FOR LOOP CONTROL
IS THIS AN INTERRUPTED CCW
YES
RETURN
NO
IS THERE ANOTHER UNIT
YES
NO
WILL A PCI FREE UP BUFFERS
YES
INCREMENT THE BUFFER COUNT
GET THE ADDRESS OF THE NEXT BUFFER
NO
IS THE TERMINAL RECEIVING
YES
SET THE PREFIX SIZE; SET THE PARAMETERS TO TPOST TO THE MH
NO
SET THE PARAMETER LIST FOR BUFFER RETURN
Q4-6,A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
ENQUEUE
Q4-2,C1,C2 Q4-5,D5,H3,H4
Q4-3,E2 Q4-6,G2
Q4-4,D3,F3,J3
PUT QCB ADDR IN THE ELEMENT; GET ADDR OF THE LAST ELEMENT ON READY QUEUE
SET THE CURRENT ELEMENT AS THE LAST IN THE DISABLED READY QUEUE
IS THE READY QUEUE EMPTY
NO
INSERT THE CURRENT ELEMENT IN THE CHAIN OF PREVIOUS ELEMENTS
YES
PUT THE CURRENT ELEMENT AT THE TOP OF THE READY QUEUE
CLEAR THE LINK FIELD OF THE CURRENT ELEMENT
IS THE SYSTEM IN A WAIT STATE
NO
YES
POST - POST THE SYSTEM WAIT COMPLETE
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart Q5-1 LINE END APPENDAGE FOR A QTAM-COMPATIBLE SYSTEM

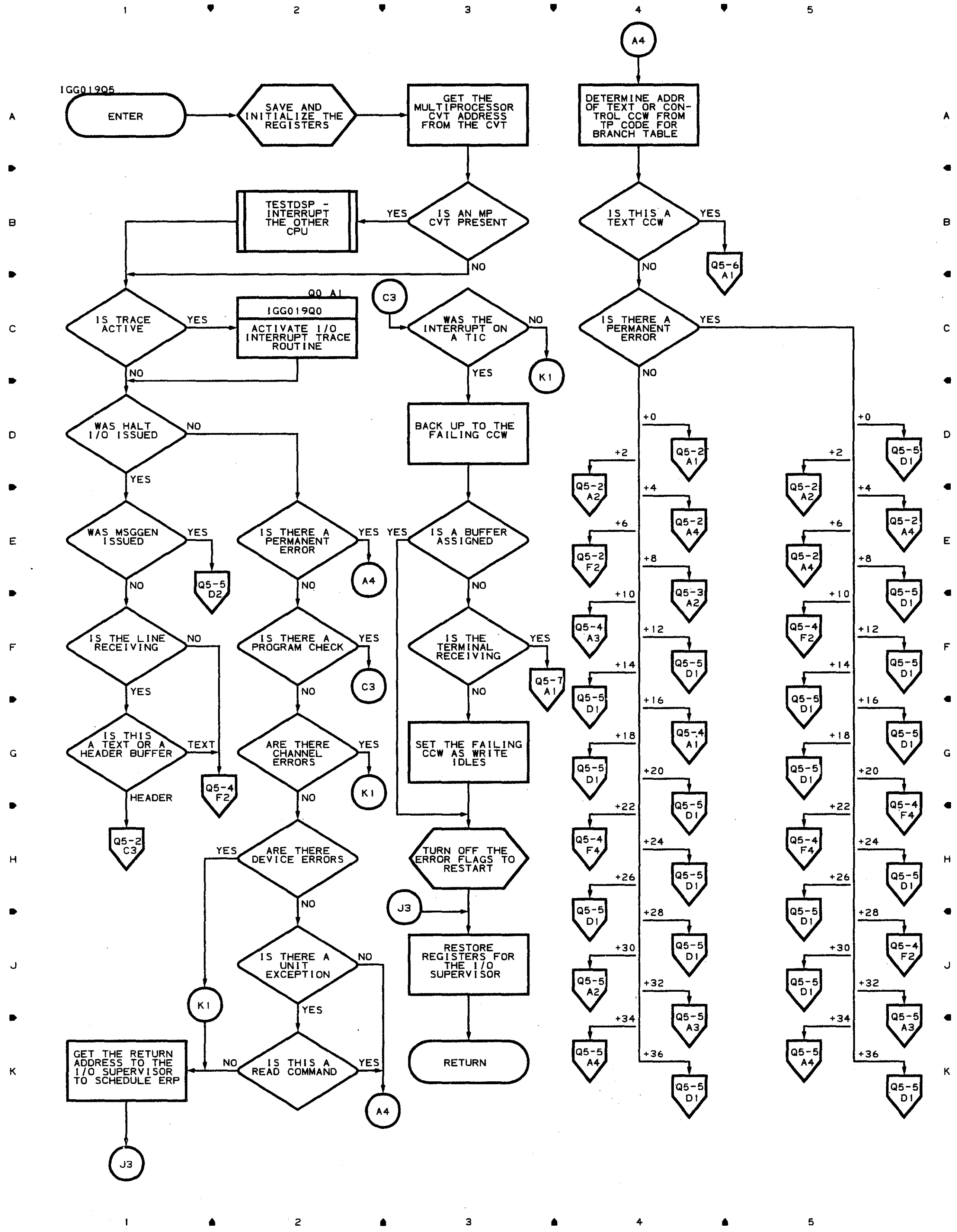


Chart Q5-2 LINE END APPENDAGE FOR A QTAM-COMPATIBLE SYSTEM

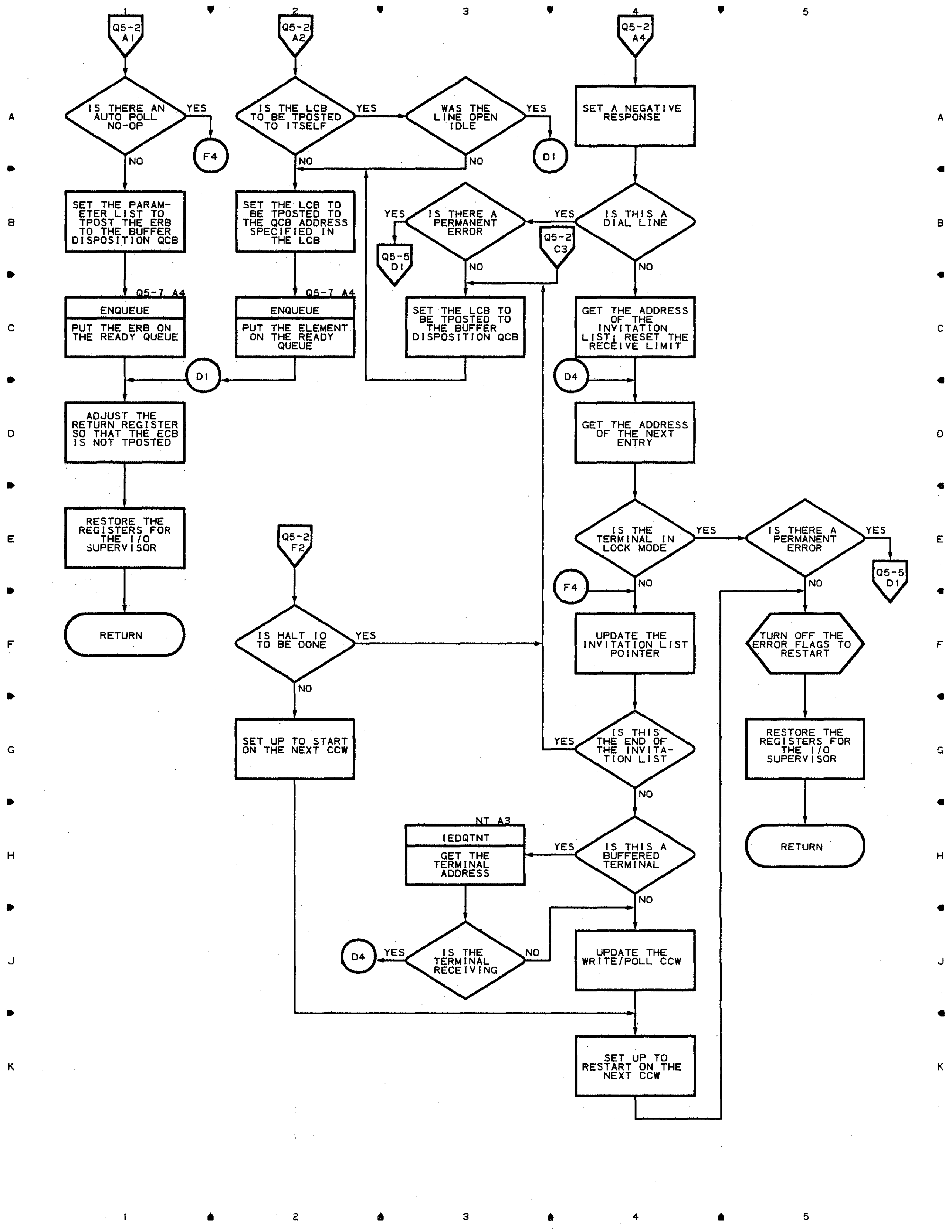
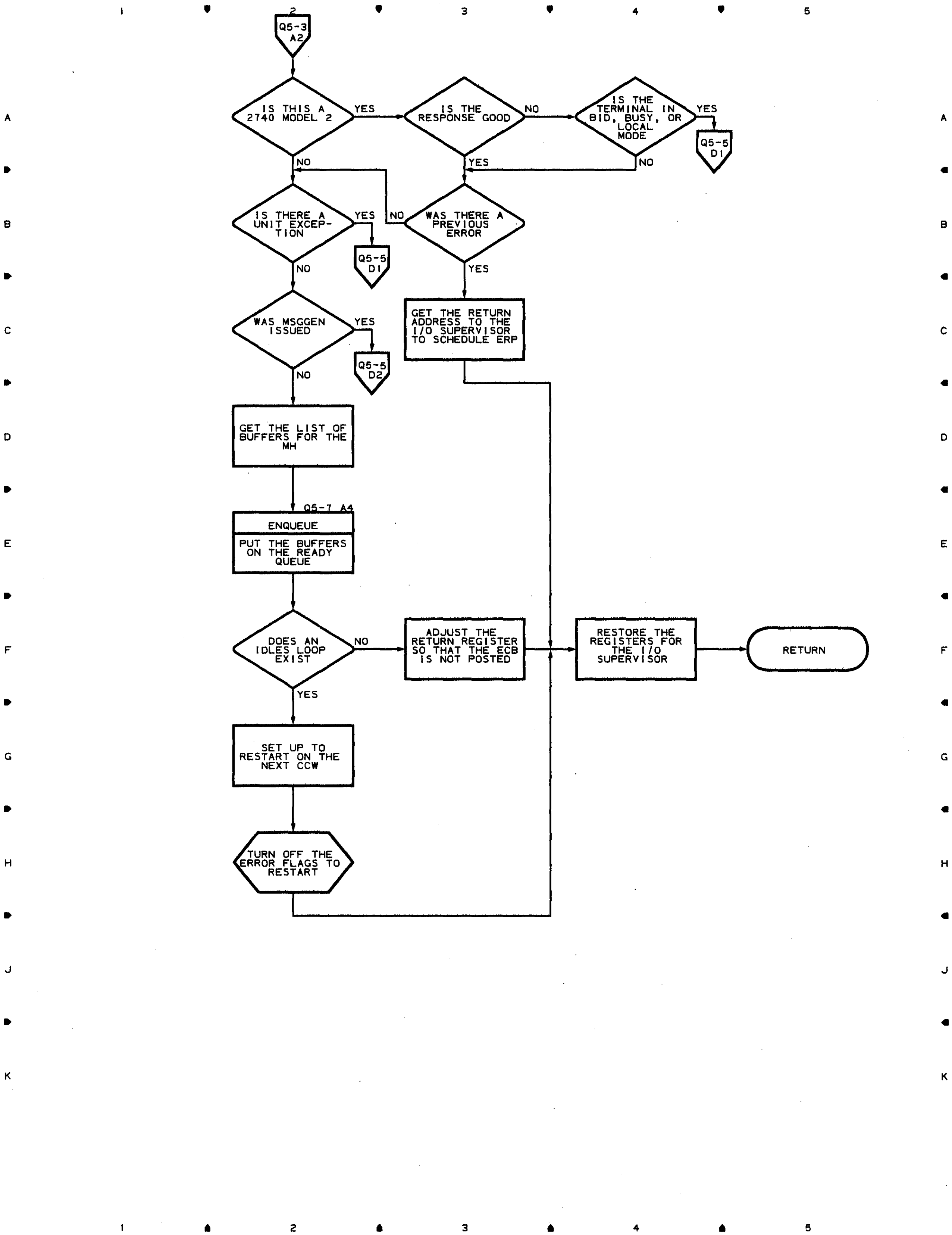
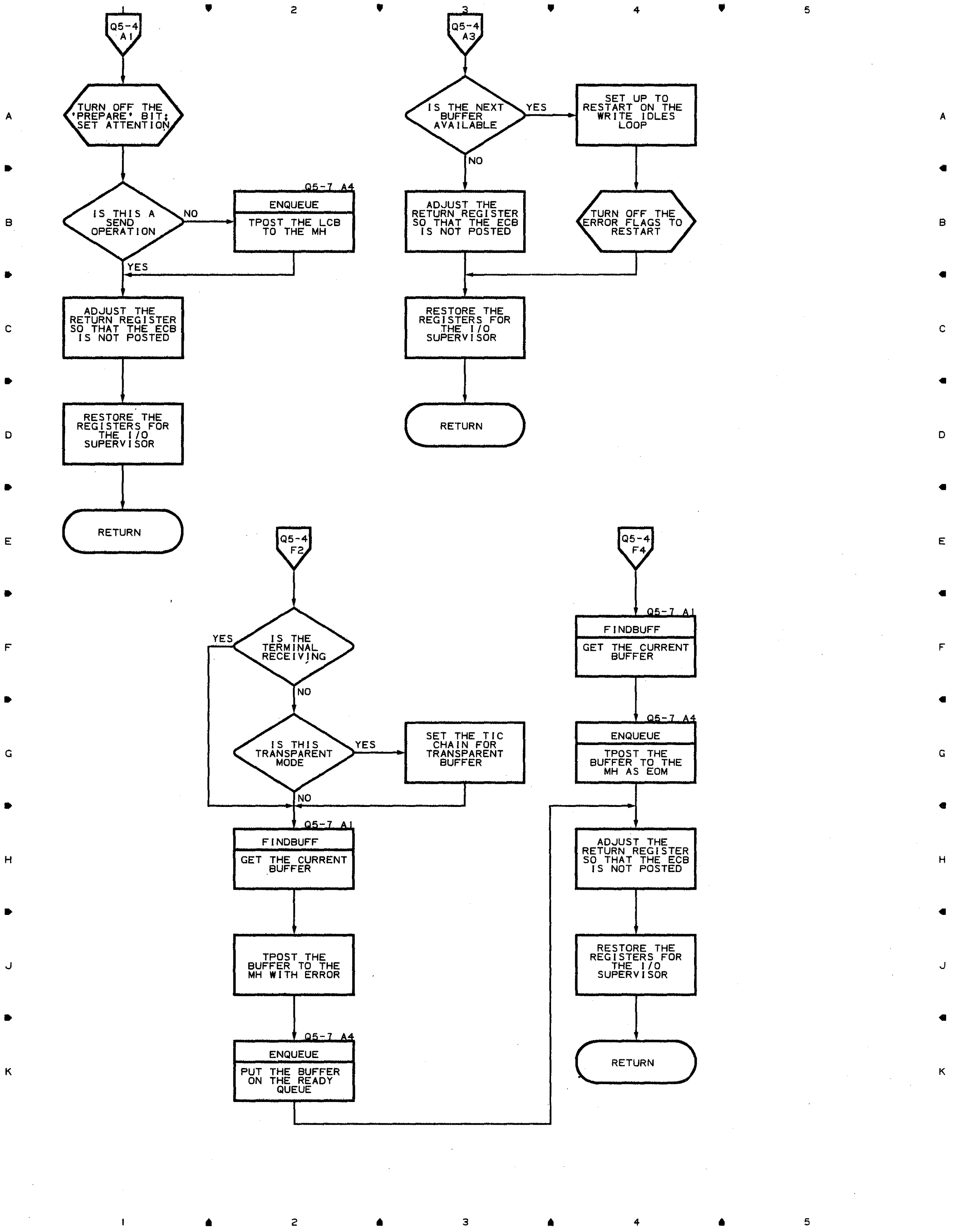

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
Q5-2 A1
IS THERE AN AUTO POLL NO-OP
YES
F4
NO
SET THE PARAMETER LIST TO TPOST THE ERB TO THE BUFFER DISPOSITION QCB
Q5-7 A4
ENQUEUE
PUT THE ERB ON THE READY QUEUE
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT TPOSTED
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN
Q5-2 A2
IS THE LCB TO BE TPOSTED TO ITSELF
YES
NO
SET THE LCB TO BE TPOSTED TO THE QCB ADDRESS SPECIFIED IN THE LCB
Q5-7 A4
ENQUEUE
PUT THE ELEMENT ON THE READY QUEUE
D1
WAS THE LINE OPEN IDLE
YES
NO
D1
YES
IS THERE A PERMANENT ERROR
Q5-5 D1
NO
Q5-2 C3
SET THE LCB TO BE TPOSTED TO THE BUFFER DISPOSITION QCB
Q5-2 F2
IS HALT IO TO BE DONE
YES
NO
SET UP TO START ON THE NEXT CCW
Q5-2 A4
SET A NEGATIVE RESPONSE
YES
IS THIS A DIAL LINE
NO
GET THE ADDRESS OF THE INVITATION LIST; RESET THE RECEIVE LIMIT
D4
GET THE ADDRESS OF THE NEXT ENTRY
IS THE TERMINAL IN LOCK MODE
YES
NO
F4
UPDATE THE INVITATION LIST POINTER
YES
IS THIS THE END OF THE INVITATION LIST
NO
NT A3
IEDQTNT
GET THE TERMINAL ADDRESS
YES
IS THIS A BUFFERED TERMINAL
NO
D4
YES
IS THE TERMINAL RECEIVING
NO
UPDATE THE WRITE/POLL CCW
SET UP TO RESTART ON THE NEXT CCW
IS THERE A PERMANENT ERROR
YES
Q5-5 D1
NO
TURN OFF THE ERROR FLAGS TO RESTART
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN

Chart Q5-3 LINE END APPENDAGE FOR A QTAM-COMPATIBLE SYSTEM



Q5-3 A2
IS THIS A 2740 MODEL 2
YES
NO
IS THE RESPONSE GOOD
NO
YES
IS THE TERMINAL IN BID, BUSY, OR LOCAL MODE
YES
NO
Q5-5 D1
IS THERE A UNIT EXCEPTION
YES
NO
Q5-5 D1
WAS THERE A PREVIOUS ERROR
NO
YES
WAS MSGGEN ISSUED
YES
NO
Q5-5 D2
GET THE RETURN ADDRESS TO THE I/O SUPERVISOR TO SCHEDULE ERP
GET THE LIST OF BUFFERS FOR THE MH
Q5-7 A4
ENQUEUE
PUT THE BUFFERS ON THE READY QUEUE
DOES AN IDLES LOOP EXIST
NO
YES
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN
SET UP TO RESTART ON THE NEXT CCW
TURN OFF THE ERROR FLAGS TO RESTART

Chart Q5-4 LINE END APPENDAGE FOR A QTAM-COMPATIBLE SYSTEM



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
Q5-4 A1
TURN OFF THE 'PREPARE' BIT; SET ATTENTION
IS THIS A SEND OPERATION
NO
YES
Q5-7 A4
ENQUEUE
TPOST THE LCB TO THE MH
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN
Q5-4 A3
IS THE NEXT BUFFER AVAILABLE
YES
NO
SET UP TO RESTART ON THE WRITE IDLES LOOP
TURN OFF THE ERROR FLAGS TO RESTART
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN
Q5-4 F2
IS THE TERMINAL RECEIVING
YES
NO
IS THIS TRANSPARENT MODE
YES
NO
SET THE TIC CHAIN FOR TRANSPARENT BUFFER
Q5-7 A1
FINDBUFF
GET THE CURRENT BUFFER
TPOST THE BUFFER TO THE MH WITH ERROR
Q5-7 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
Q5-4 F4
Q5-7 A1
FINDBUFF
GET THE CURRENT BUFFER
Q5-7 A4
ENQUEUE
TPOST THE BUFFER TO THE MH AS EOM
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN

Chart Q5-5 LINE END APPENDAGE FOR A QTAM-COMPATIBLE SYSTEM

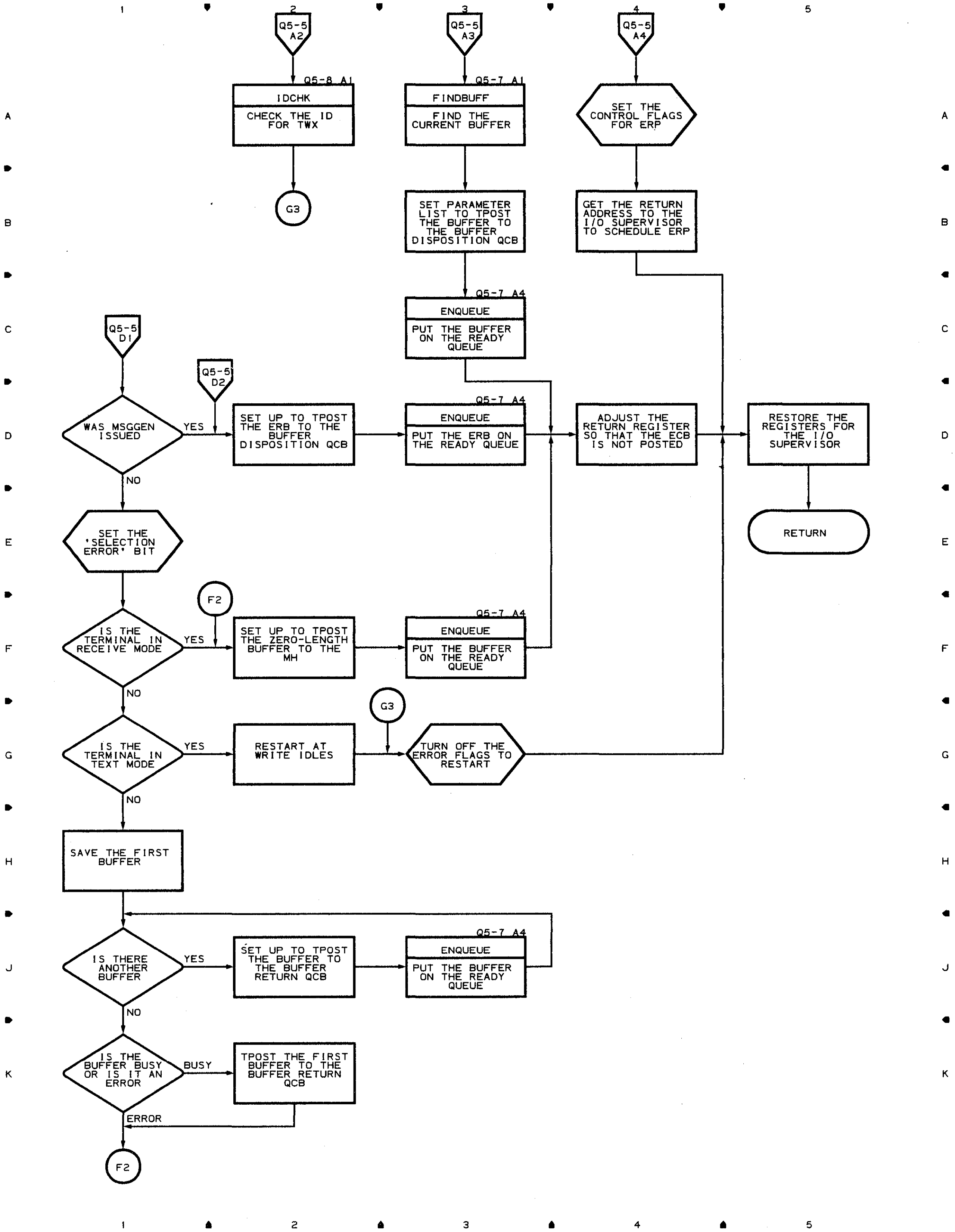

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
Q5-5 A2
Q5-8 A1
IDCHK
CHECK THE ID FOR TWX
G3
Q5-5 A3
Q5-7 A1
FINDBUFF
FIND THE CURRENT BUFFER
SET PARAMETER LIST TO TPOST THE BUFFER TO THE BUFFER DISPOSITION QCB
Q5-7 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
Q5-5 A4
SET THE CONTROL FLAGS FOR ERP
GET THE RETURN ADDRESS TO THE I/O SUPERVISOR TO SCHEDULE ERP
Q5-5 D1
Q5-5 D2
WAS MSGGEN ISSUED
YES
NO
SET UP TO TPOST THE ERB TO THE BUFFER DISPOSITION QCB
Q5-7 A4
ENQUEUE
PUT THE ERB ON THE READY QUEUE
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN
SET THE 'SELECTION ERROR' BIT
F2
IS THE TERMINAL IN RECEIVE MODE
YES
NO
SET UP TO TPOST THE ZERO-LENGTH BUFFER TO THE MH
Q5-7 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
G3
IS THE TERMINAL IN TEXT MODE
YES
NO
RESTART AT WRITE IDLES
TURN OFF THE ERROR FLAGS TO RESTART
SAVE THE FIRST BUFFER
IS THERE ANOTHER BUFFER
YES
NO
SET UP TO TPOST THE BUFFER TO THE BUFFER RETURN QCB
Q5-7 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
IS THE BUFFER BUSY OR IS IT AN ERROR
BUSY
ERROR
TPOST THE FIRST BUFFER TO THE BUFFER RETURN QCB
F2

Chart Q5-6 LINE END APPENDAGE FOR A QTAM-COMPATIBLE SYSTEM

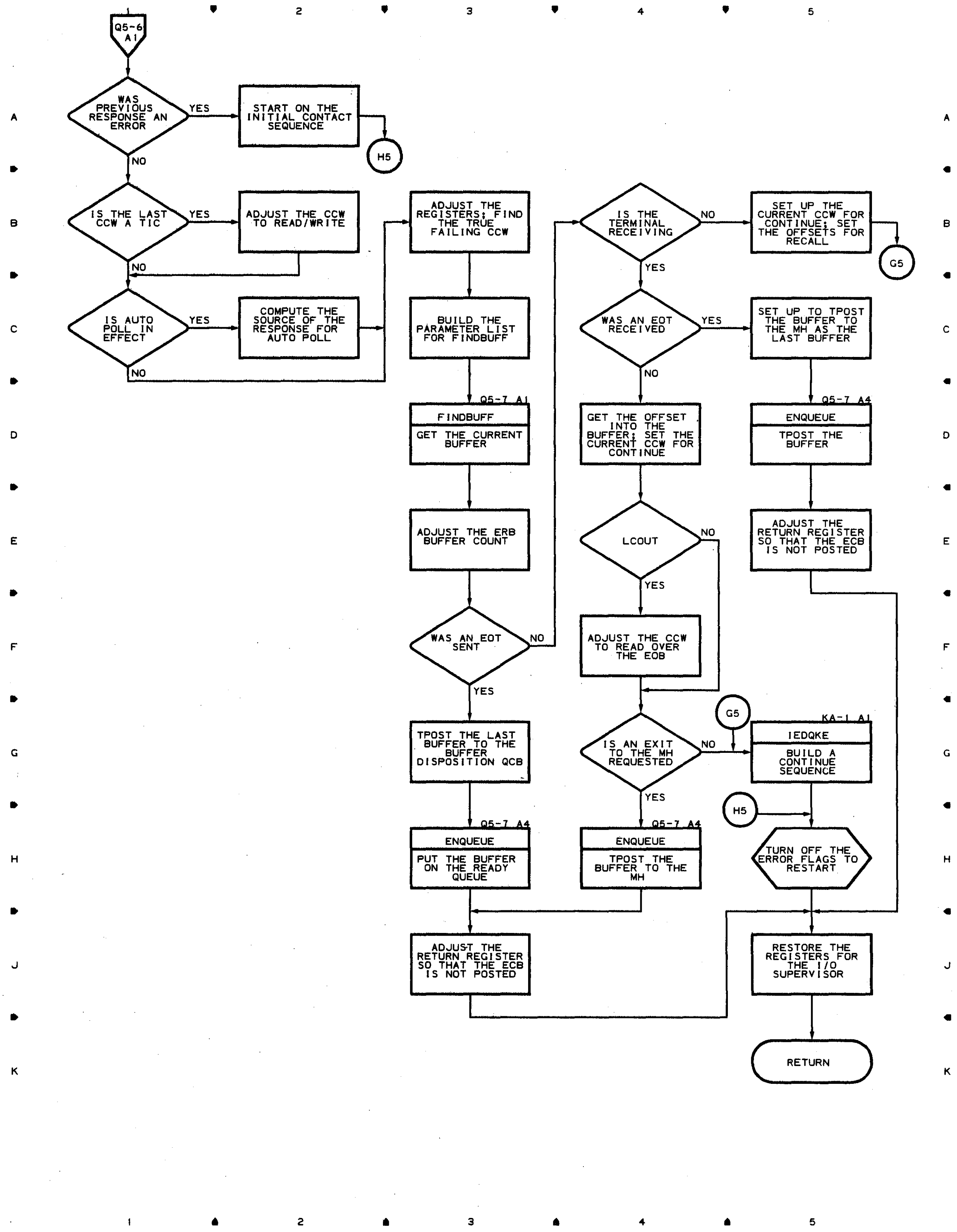

1
2
3
4
5
Q5-6 A1
A
WAS PREVIOUS RESPONSE AN ERROR
YES
START ON THE INITIAL CONTACT SEQUENCE
H5
NO
B
IS THE LAST CCW A TIC
YES
ADJUST THE CCW TO READ/WRITE
NO
ADJUST THE REGISTERS; FIND THE TRUE FAILING CCW
IS THE TERMINAL RECEIVING
NO
SET UP THE CURRENT CCW FOR CONTINUE; SET THE OFFSETS FOR RECALL
G5
YES
C
IS AUTO POLL IN EFFECT
YES
COMPUTE THE SOURCE OF THE RESPONSE FOR AUTO POLL
BUILD THE PARAMETER LIST FOR FINDBUFF
WAS AN EOT RECEIVED
YES
SET UP TO TPOST THE BUFFER TO THE MH AS THE LAST BUFFER
NO
NO
D
Q5-7 A1
FINDBUFF
GET THE CURRENT BUFFER
GET THE OFFSET INTO THE BUFFER; SET THE CURRENT CCW FOR CONTINUE
Q5-7 A4
ENQUEUE
TPOST THE BUFFER
E
ADJUST THE ERB BUFFER COUNT
LCOUT
NO
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
YES
F
WAS AN EOT SENT
NO
ADJUST THE CCW TO READ OVER THE EOB
YES
G5
KA-1 A1
G
TPOST THE LAST BUFFER TO THE BUFFER DISPOSITION QCB
IS AN EXIT TO THE MH REQUESTED
NO
IEDQKE
BUILD A CONTINUE SEQUENCE
YES
H5
Q5-7 A4
Q5-7 A4
H
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
ENQUEUE
TPOST THE BUFFER TO THE MH
TURN OFF THE ERROR FLAGS TO RESTART
J
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
K
RETURN
1
2
3
4
5

Chart Q5-7 LINE END APPENDAGE FOR A QTAM-COMPATIBLE SYSTEM

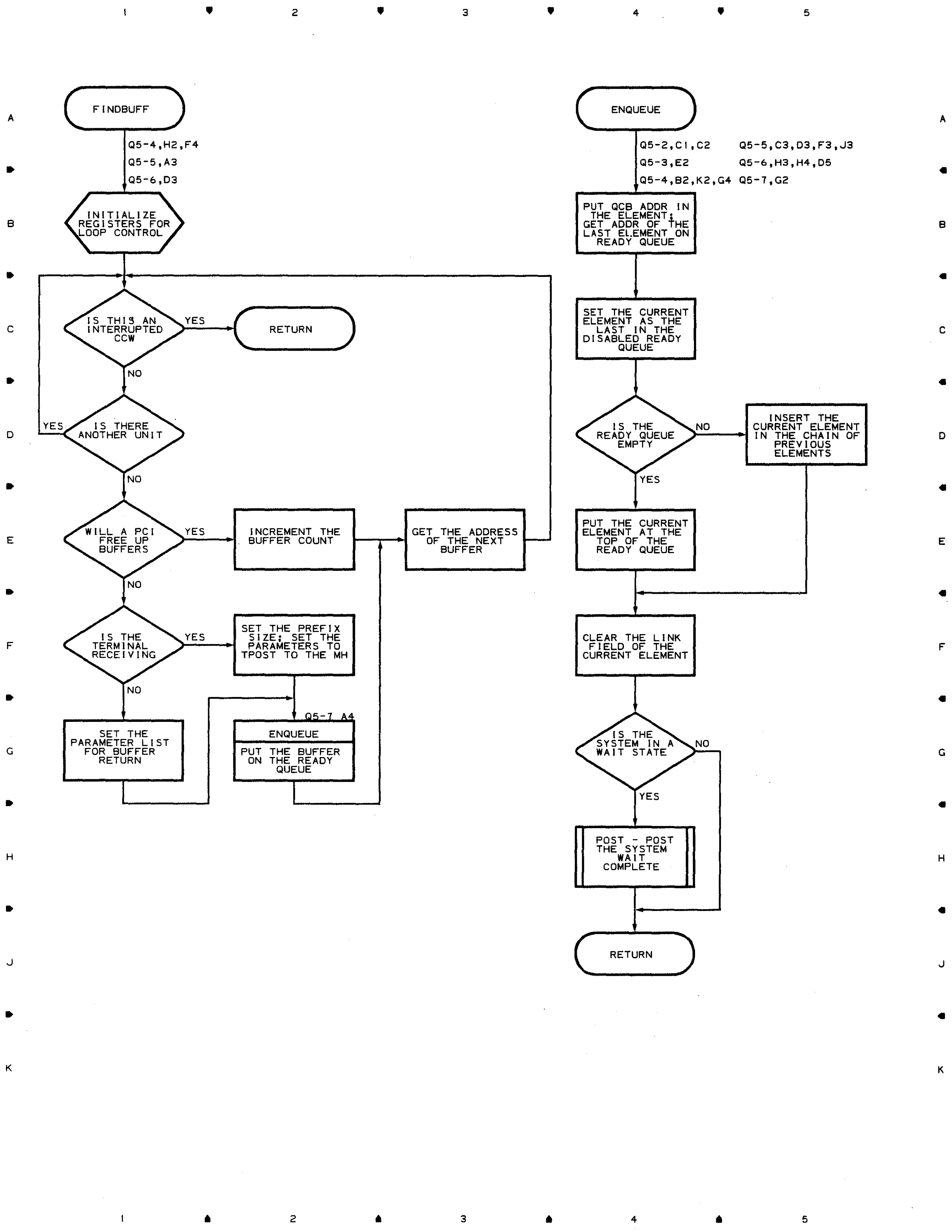

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
FINDBUFF
Q5-4,H2,F4
Q5-5,A3
Q5-6,D3
INITIALIZE REGISTERS FOR LOOP CONTROL
IS THIS AN INTERRUPTED CCW
YES
RETURN
NO
YES
IS THERE ANOTHER UNIT
NO
WILL A PCI FREE UP BUFFERS
YES
INCREMENT THE BUFFER COUNT
GET THE ADDRESS OF THE NEXT BUFFER
NO
IS THE TERMINAL RECEIVING
YES
SET THE PREFIX SIZE; SET THE PARAMETERS TO TPOST TO THE MH
NO
SET THE PARAMETER LIST FOR BUFFER RETURN
Q5-7,A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
ENQUEUE
Q5-2,C1,C2
Q5-3,E2
Q5-4,B2,K2,G4
Q5-5,C3,D3,F3,J3
Q5-6,H3,H4,D5
Q5-7,G2
PUT QCB ADDR IN THE ELEMENT; GET ADDR OF THE LAST ELEMENT ON READY QUEUE
SET THE CURRENT ELEMENT AS THE LAST IN THE DISABLED READY QUEUE
IS THE READY QUEUE EMPTY
NO
INSERT THE CURRENT ELEMENT IN THE CHAIN OF PREVIOUS ELEMENTS
YES
PUT THE CURRENT ELEMENT AT THE TOP OF THE READY QUEUE
CLEAR THE LINK FIELD OF THE CURRENT ELEMENT
IS THE SYSTEM IN A WAIT STATE
NO
YES
POST - POST THE SYSTEM WAIT COMPLETE
RETURN

Chart Q5-8 LINE END APPENDAGE FOR A QTAM-COMPATIBLE SYSTEM

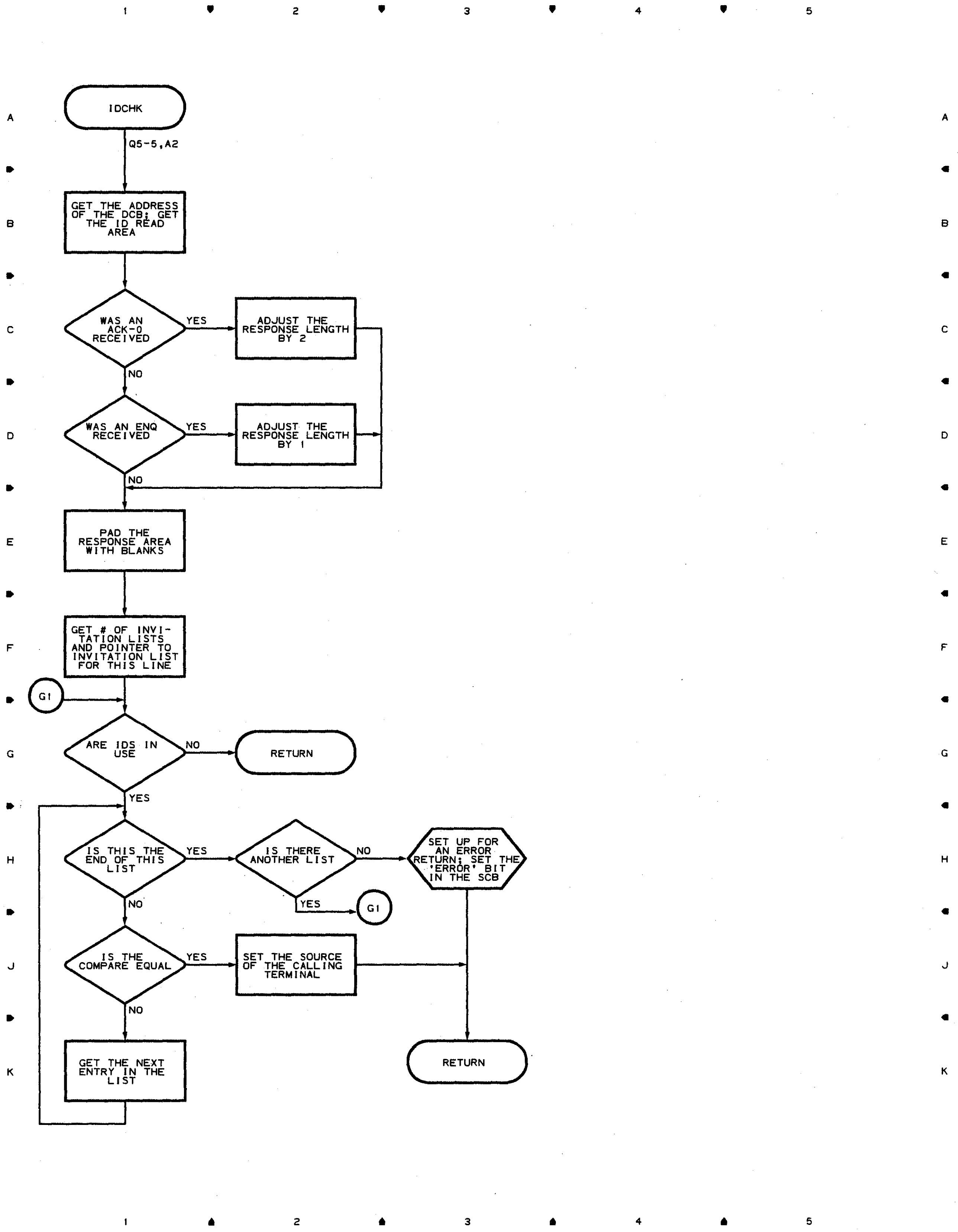

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IDCHK
Q5-5,A2
GET THE ADDRESS OF THE DCB; GET THE ID READ AREA
WAS AN ACK-0 RECEIVED
YES
ADJUST THE RESPONSE LENGTH BY 2
NO
WAS AN ENQ RECEIVED
YES
ADJUST THE RESPONSE LENGTH BY 1
NO
PAD THE RESPONSE AREA WITH BLANKS
GET # OF INVITATION LISTS AND POINTER TO INVITATION LIST FOR THIS LINE
G1
ARE IDS IN USE
NO
RETURN
YES
IS THIS THE END OF THIS LIST
YES
IS THERE ANOTHER LIST
NO
SET UP FOR AN ERROR RETURN; SET THE 'ERROR' BIT IN THE SCB
YES
G1
NO
IS THE COMPARE EQUAL
YES
SET THE SOURCE OF THE CALLING TERMINAL
NO
GET THE NEXT ENTRY IN THE LIST
RETURN

Chart Q6 SEND SCHEDULER FOR LEASED LINES AND NO TSO

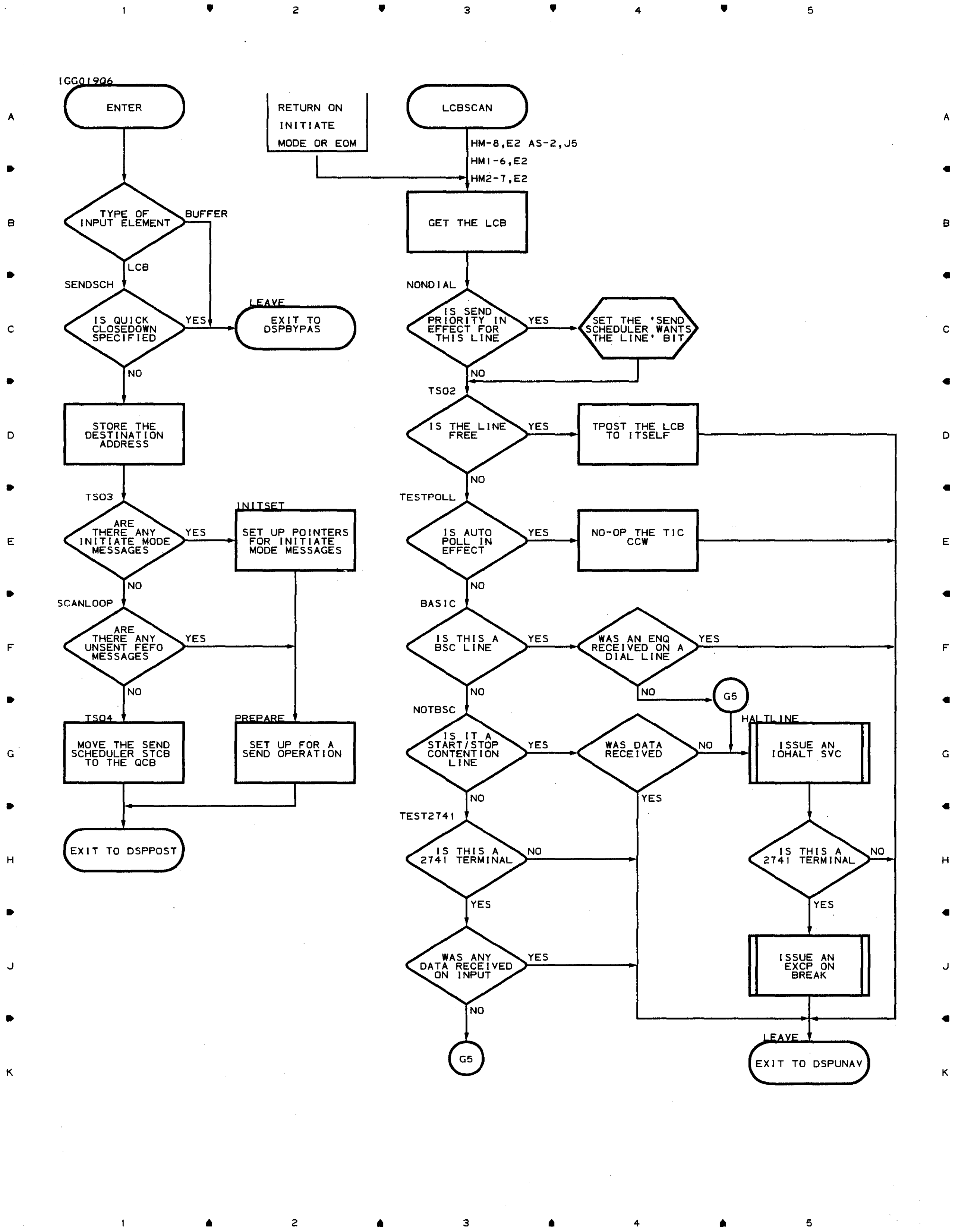
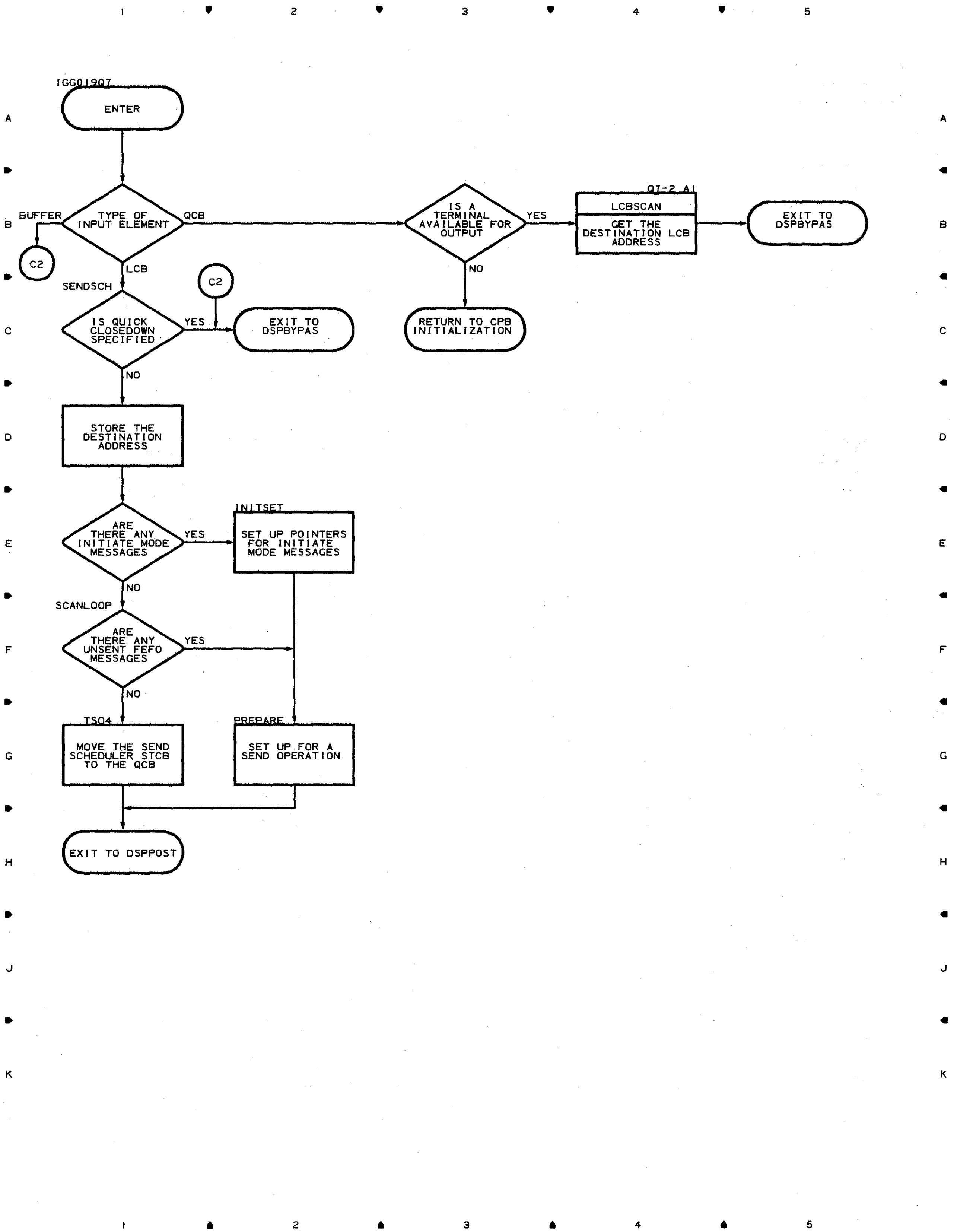


Chart Q7-1 SEND SCHEDULER WITH NO TSO



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG01907
ENTER
BUFFER
TYPE OF INPUT ELEMENT
QCB
LCB
C2
SENDSCH
IS QUICK CLOSEDOWN SPECIFIED
YES
C2
EXIT TO DSPBYPAS
NO
STORE THE DESTINATION ADDRESS
ARE THERE ANY INITIATE MODE MESSAGES
YES
INITSET
SET UP POINTERS FOR INITIATE MODE MESSAGES
NO
SCANLOOP
ARE THERE ANY UNSENT FEFO MESSAGES
YES
NO
TS04
MOVE THE SEND SCHEDULER STCB TO THE QCB
PREPARE
SET UP FOR A SEND OPERATION
EXIT TO DSPPOST
IS A TERMINAL AVAILABLE FOR OUTPUT
YES
NO
RETURN TO CPB INITIALIZATION
Q7-2 A1
LCBSCAN
GET THE DESTINATION LCB ADDRESS
EXIT TO DSPBYPAS

Chart Q7-2 SEND SCHEDULER WITH NO TSO

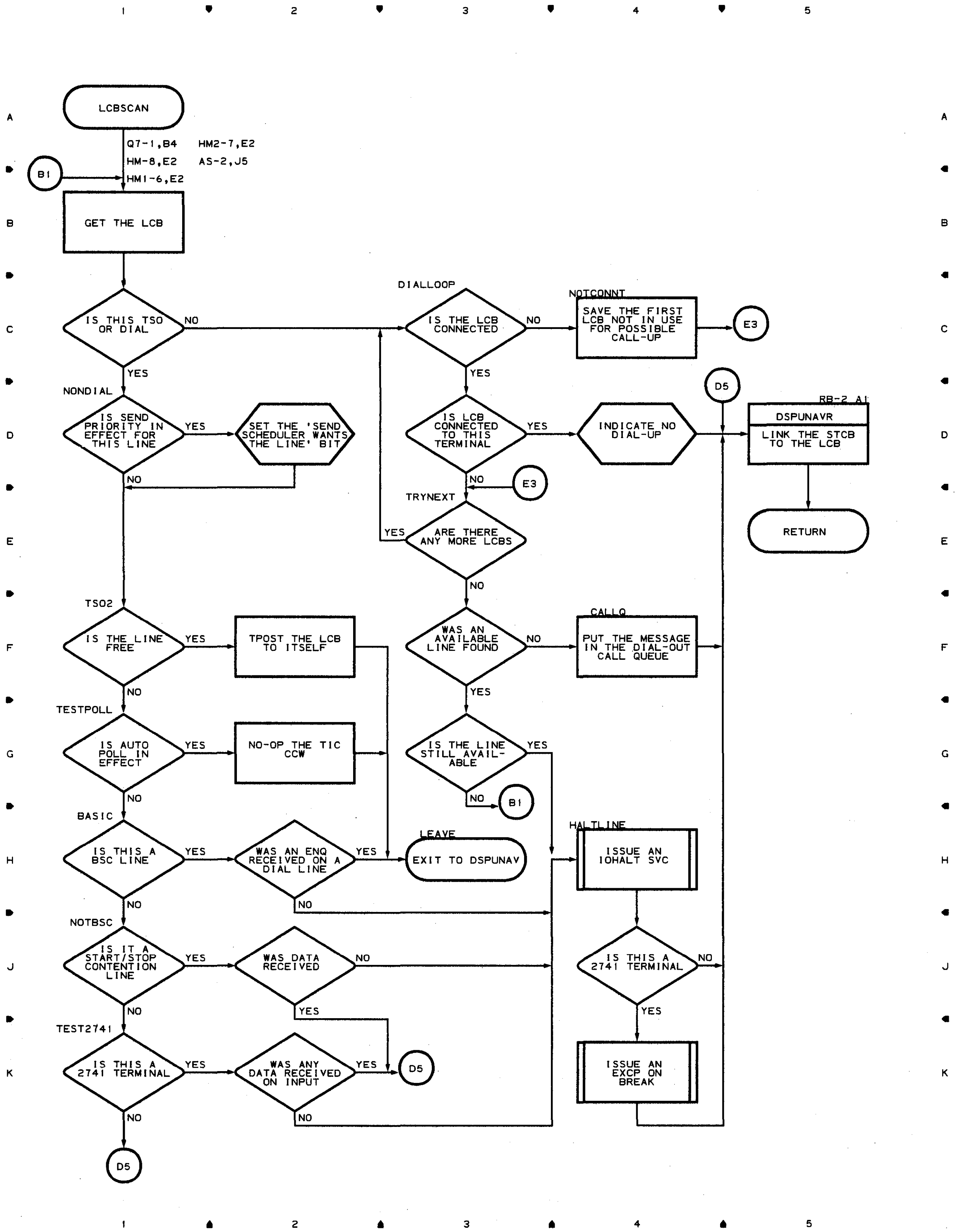


Chart Q8 CHECKPOINT CONTINUATION RESTART SUBROUTINE

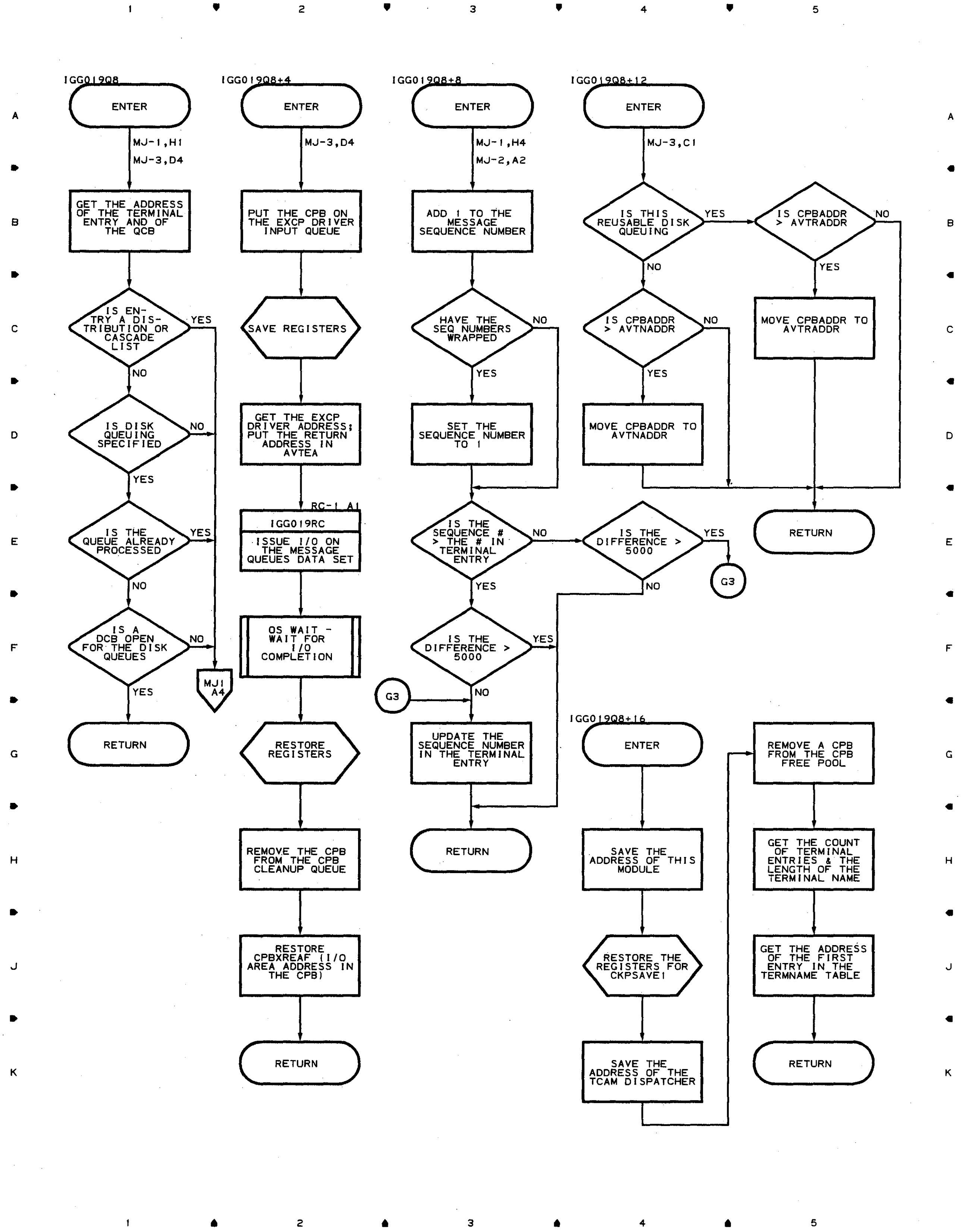

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG019Q8
ENTER
MJ-1,H1
MJ-3,D4
GET THE ADDRESS OF THE TERMINAL ENTRY AND OF THE QCB
IS ENTRY A DISTRIBUTION OR CASCADE LIST
YES
NO
IS DISK QUEUING SPECIFIED
NO
YES
IS THE QUEUE ALREADY PROCESSED
YES
NO
IS A DCB OPEN FOR THE DISK QUEUES
NO
YES
MJ1 A4
RETURN
IGG019Q8+4
ENTER
MJ-3,D4
PUT THE CPB ON THE EXCP DRIVER INPUT QUEUE
SAVE REGISTERS
GET THE EXCP DRIVER ADDRESS; PUT THE RETURN ADDRESS IN AVTEA
RC-1 A1
IGG019RC
ISSUE I/O ON THE MESSAGE QUEUES DATA SET
OS WAIT - WAIT FOR I/O COMPLETION
RESTORE REGISTERS
REMOVE THE CPB FROM THE CPB CLEANUP QUEUE
RESTORE CPBXREAF (I/O AREA ADDRESS IN THE CPB)
RETURN
IGG019Q8+8
ENTER
MJ-1,H4
MJ-2,A2
ADD 1 TO THE MESSAGE SEQUENCE NUMBER
HAVE THE SEQ NUMBERS WRAPPED
NO
YES
SET THE SEQUENCE NUMBER TO 1
IS THE SEQUENCE # > THE # IN TERMINAL ENTRY
NO
YES
IS THE DIFFERENCE > 5000
YES
NO
G3
UPDATE THE SEQUENCE NUMBER IN THE TERMINAL ENTRY
RETURN
IGG019Q8+12
ENTER
MJ-3,C1
IS THIS REUSABLE DISK QUEUING
YES
NO
IS CPBADDR > AVTNADDR
NO
YES
MOVE CPBADDR TO AVTNADDR
IS THE DIFFERENCE > 5000
YES
NO
G3
IS CPBADDR > AVTRADDR
NO
YES
MOVE CPBADDR TO AVTRADDR
RETURN
IGG019Q8+16
ENTER
SAVE THE ADDRESS OF THIS MODULE
RESTORE THE REGISTERS FOR CKPSAVE1
SAVE THE ADDRESS OF THE TCAM DISPATCHER
REMOVE A CPB FROM THE CPB FREE POOL
GET THE COUNT OF TERMINAL ENTRIES & THE LENGTH OF THE TERMINAL NAME
GET THE ADDRESS OF THE FIRST ENTRY IN THE TERMNAME TABLE
RETURN

Chart RA CHECKPOINT DISK END APPENDAGE

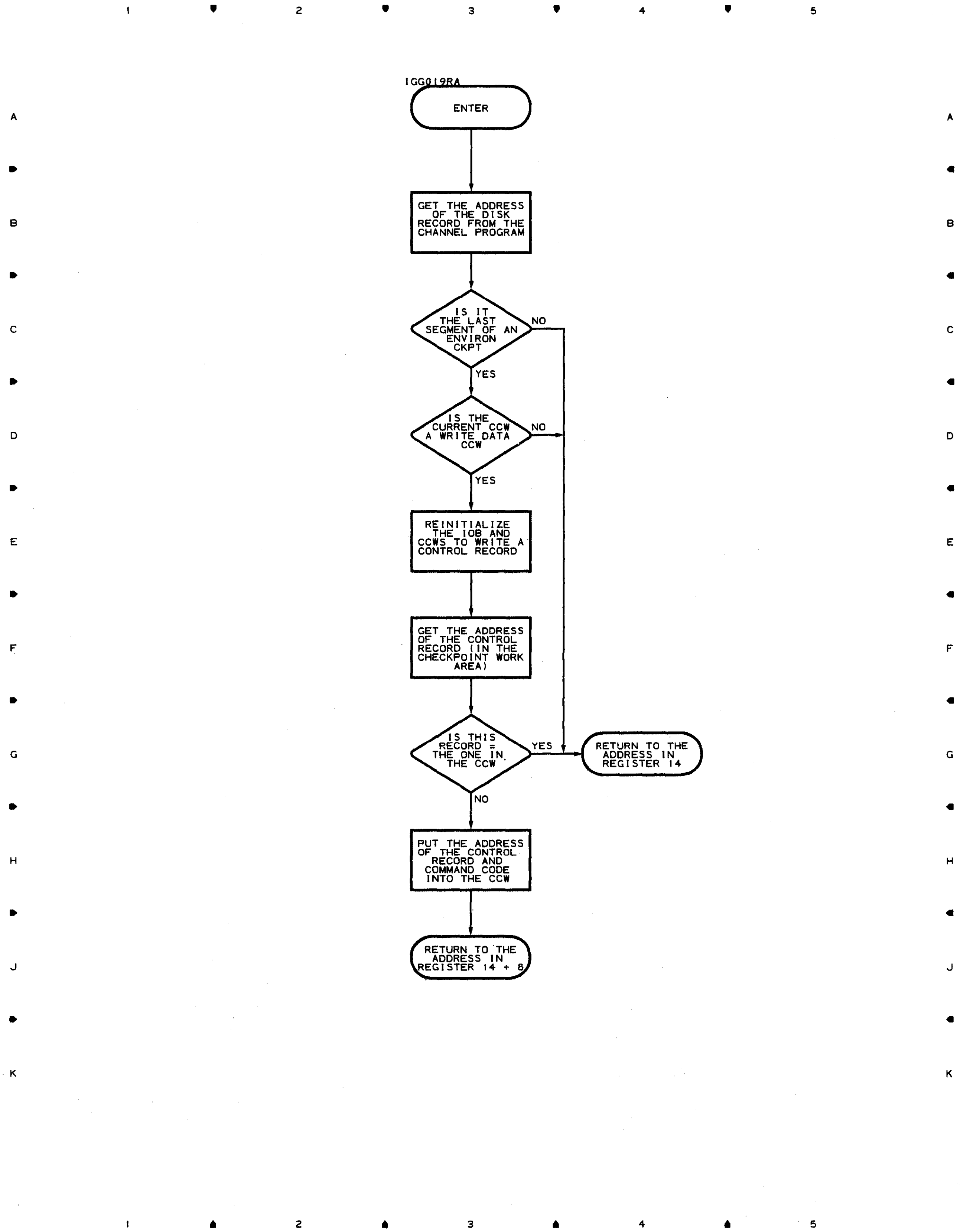

1 2 3 4 5
A B C D E F G H J K
IGG019RA
ENTER
GET THE ADDRESS OF THE DISK RECORD FROM THE CHANNEL PROGRAM
IS IT THE LAST SEGMENT OF AN ENVIRON CKPT
NO
YES
IS THE CURRENT CCW A WRITE DATA CCW
NO
YES
REINITIALIZE THE IOB AND CCWS TO WRITE A CONTROL RECORD
GET THE ADDRESS OF THE CONTROL RECORD (IN THE CHECKPOINT WORK AREA)
IS THIS RECORD = THE ONE IN THE CCW
YES
RETURN TO THE ADDRESS IN REGISTER 14
NO
PUT THE ADDRESS OF THE CONTROL RECORD AND COMMAND CODE INTO THE CCW
RETURN TO THE ADDRESS IN REGISTER 14 + 8

Chart RB-1 TCAM DISPATCHER

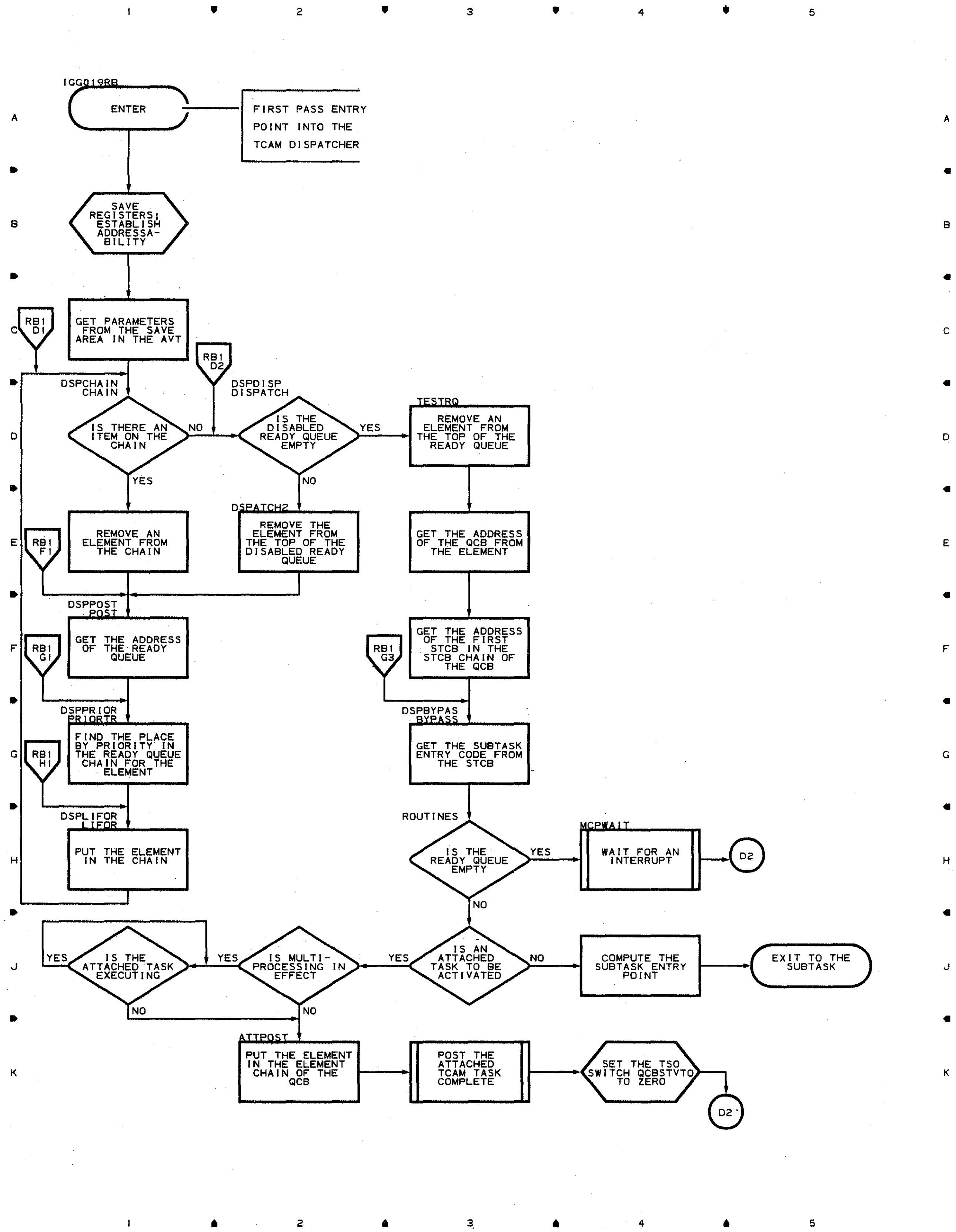
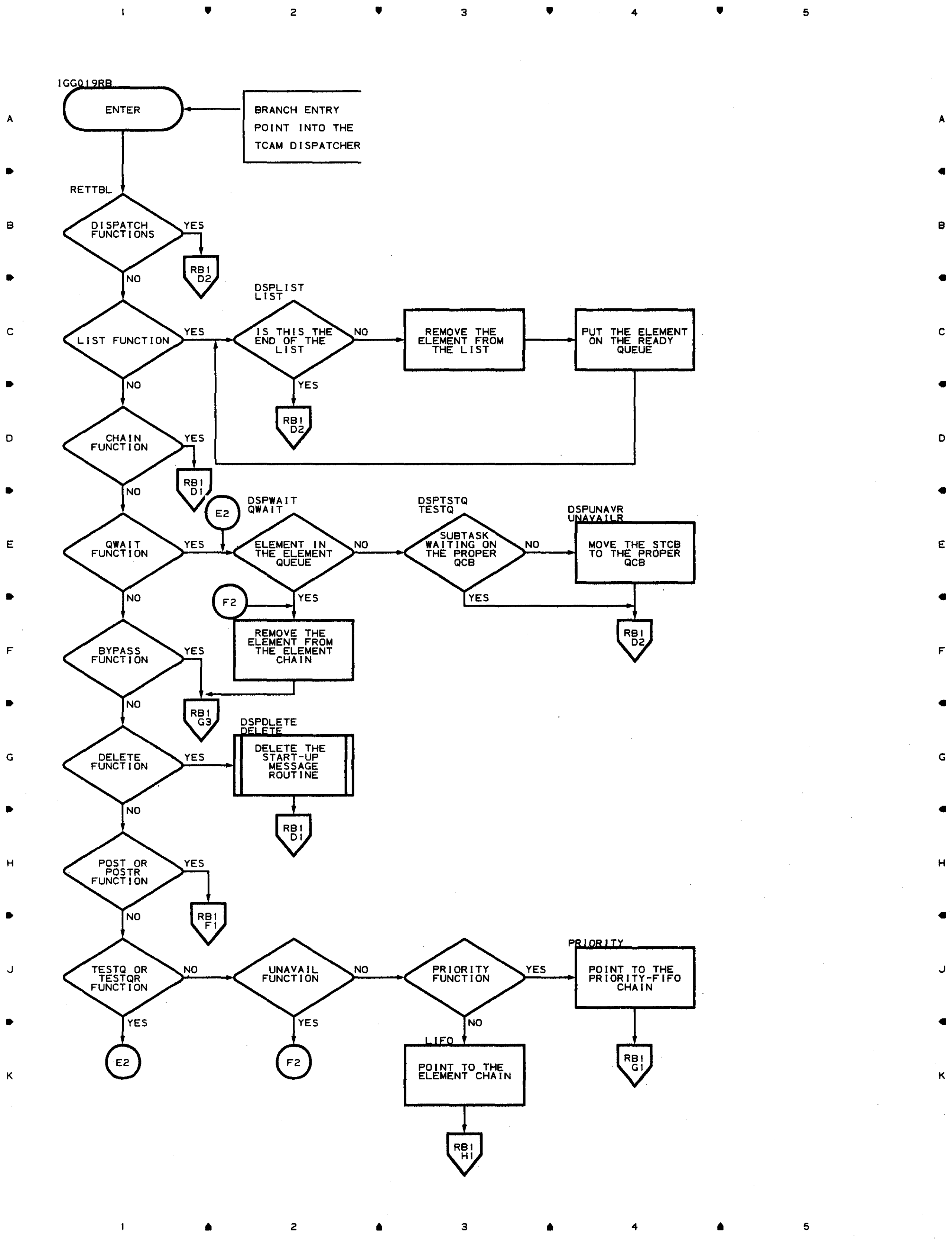

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG019RB
ENTER
FIRST PASS ENTRY POINT INTO THE TCAM DISPATCHER
SAVE REGISTERS; ESTABLISH ADDRESSA-BILITY
RB1 D1
GET PARAMETERS FROM THE SAVE AREA IN THE AVT
RB1 D2
DSPCHAIN CHAIN
IS THERE AN ITEM ON THE CHAIN
NO
YES
DSPDISP DISPATCH
IS THE DISABLED READY QUEUE EMPTY
YES
NO
TESTRQ
REMOVE AN ELEMENT FROM THE TOP OF THE READY QUEUE
RB1 F1
REMOVE AN ELEMENT FROM THE CHAIN
DSPATCH2
REMOVE THE ELEMENT FROM THE TOP OF THE DISABLED READY QUEUE
GET THE ADDRESS OF THE QCB FROM THE ELEMENT
DSPPOST POST
RB1 G1
GET THE ADDRESS OF THE READY QUEUE
RB1 G3
GET THE ADDRESS OF THE FIRST STCB IN THE STCB CHAIN OF THE QCB
DSPPRIOR PRIORTR
RB1 H1
FIND THE PLACE BY PRIORITY IN THE READY QUEUE CHAIN FOR THE ELEMENT
DSPBYPAS BYPASS
GET THE SUBTASK ENTRY CODE FROM THE STCB
DSPLIFOR LIFOR
PUT THE ELEMENT IN THE CHAIN
ROUTINES
IS THE READY QUEUE EMPTY
YES
MCPWAIT
WAIT FOR AN INTERRUPT
D2
NO
YES
IS THE ATTACHED TASK EXECUTING
YES
IS MULTI-PROCESSING IN EFFECT
YES
IS AN ATTACHED TASK TO BE ACTIVATED
NO
COMPUTE THE SUBTASK ENTRY POINT
EXIT TO THE SUBTASK
NO
NO
ATTPOST
PUT THE ELEMENT IN THE ELEMENT CHAIN OF THE QCB
POST THE ATTACHED TCAM TASK COMPLETE
SET THE TSO SWITCH QCBSTVTO TO ZERO
D2

Chart RB-2 TCAM DISPATCHER



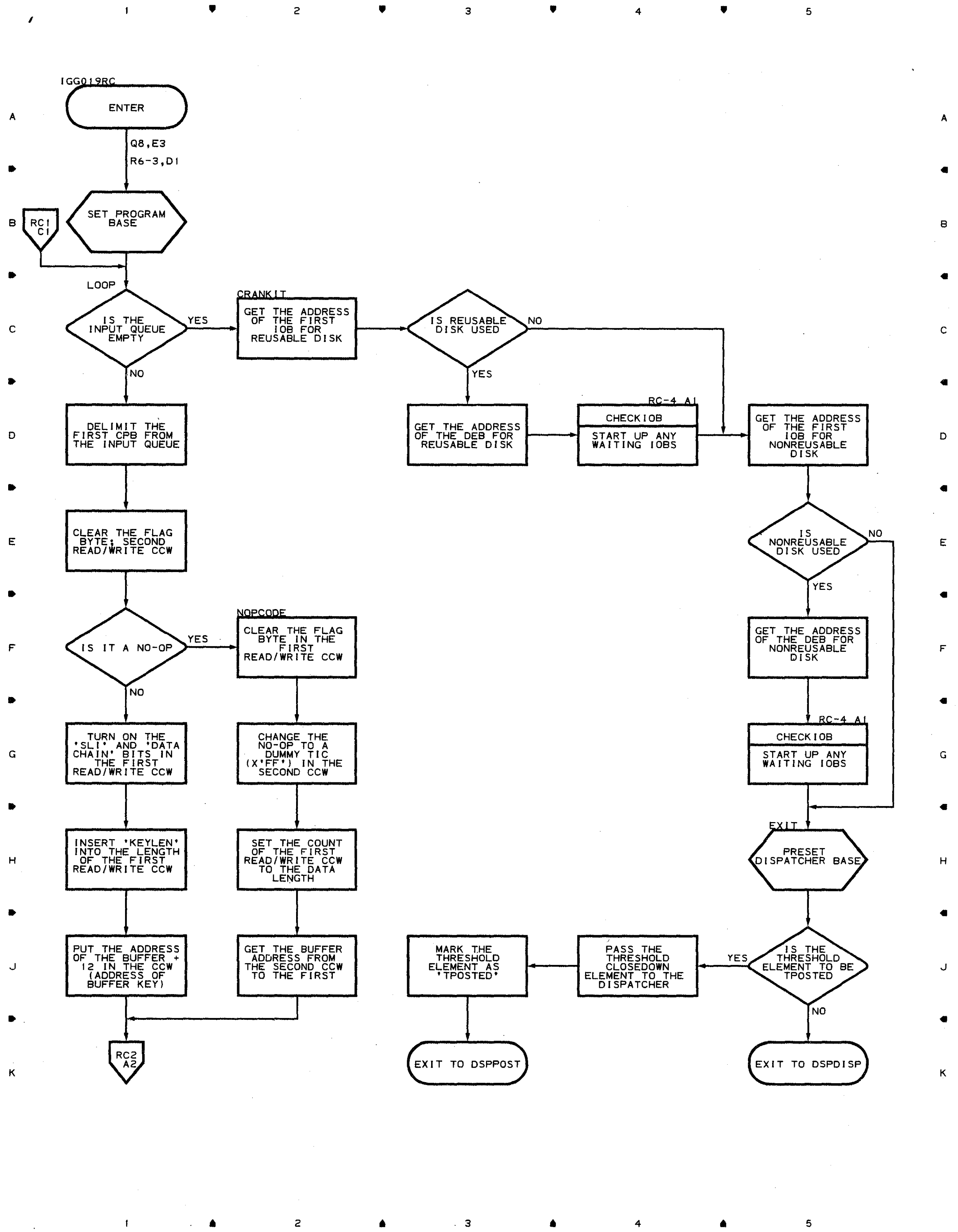

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG019RC
ENTER
Q8,E3
R6-3,D1
SET PROGRAM BASE
RC1 C1
LOOP
IS THE INPUT QUEUE EMPTY
YES
NO
CRANKIT
GET THE ADDRESS OF THE FIRST IOB FOR REUSABLE DISK
IS REUSABLE DISK USED
NO
YES
DELIMIT THE FIRST CPB FROM THE INPUT QUEUE
GET THE ADDRESS OF THE DEB FOR REUSABLE DISK
RC-4 A1
CHECKIOB
START UP ANY WAITING IOBS
GET THE ADDRESS OF THE FIRST IOB FOR NONREUSABLE DISK
CLEAR THE FLAG BYTE; SECOND READ/WRITE CCW
IS NONREUSABLE DISK USED
NO
YES
IS IT A NO-OP
YES
NO
NOPCODE
CLEAR THE FLAG BYTE IN THE FIRST READ/WRITE CCW
GET THE ADDRESS OF THE DEB FOR NONREUSABLE DISK
TURN ON THE 'SLI' AND 'DATA CHAIN' BITS IN THE FIRST READ/WRITE CCW
CHANGE THE NO-OP TO A DUMMY TIC (X'FF') IN THE SECOND CCW
RC-4 A1
CHECKIOB
START UP ANY WAITING IOBS
EXIT
INSERT 'KEYLEN' INTO THE LENGTH OF THE FIRST READ/WRITE CCW
SET THE COUNT OF THE FIRST READ/WRITE CCW TO THE DATA LENGTH
PRESET DISPATCHER BASE
PUT THE ADDRESS OF THE BUFFER + 12 IN THE CCW (ADDRESS OF BUFFER KEY)
GET THE BUFFER ADDRESS FROM THE SECOND CCW TO THE FIRST
MARK THE THRESHOLD ELEMENT AS 'TPOSTED'
PASS THE THRESHOLD CLOSEDOWN ELEMENT TO THE DISPATCHER
YES
IS THE THRESHOLD ELEMENT TO BE TPOSTED
NO
RC2 A2
EXIT TO DSPPOST
EXIT TO DSPDISP

Chart RC-2 EXCP DRIVER

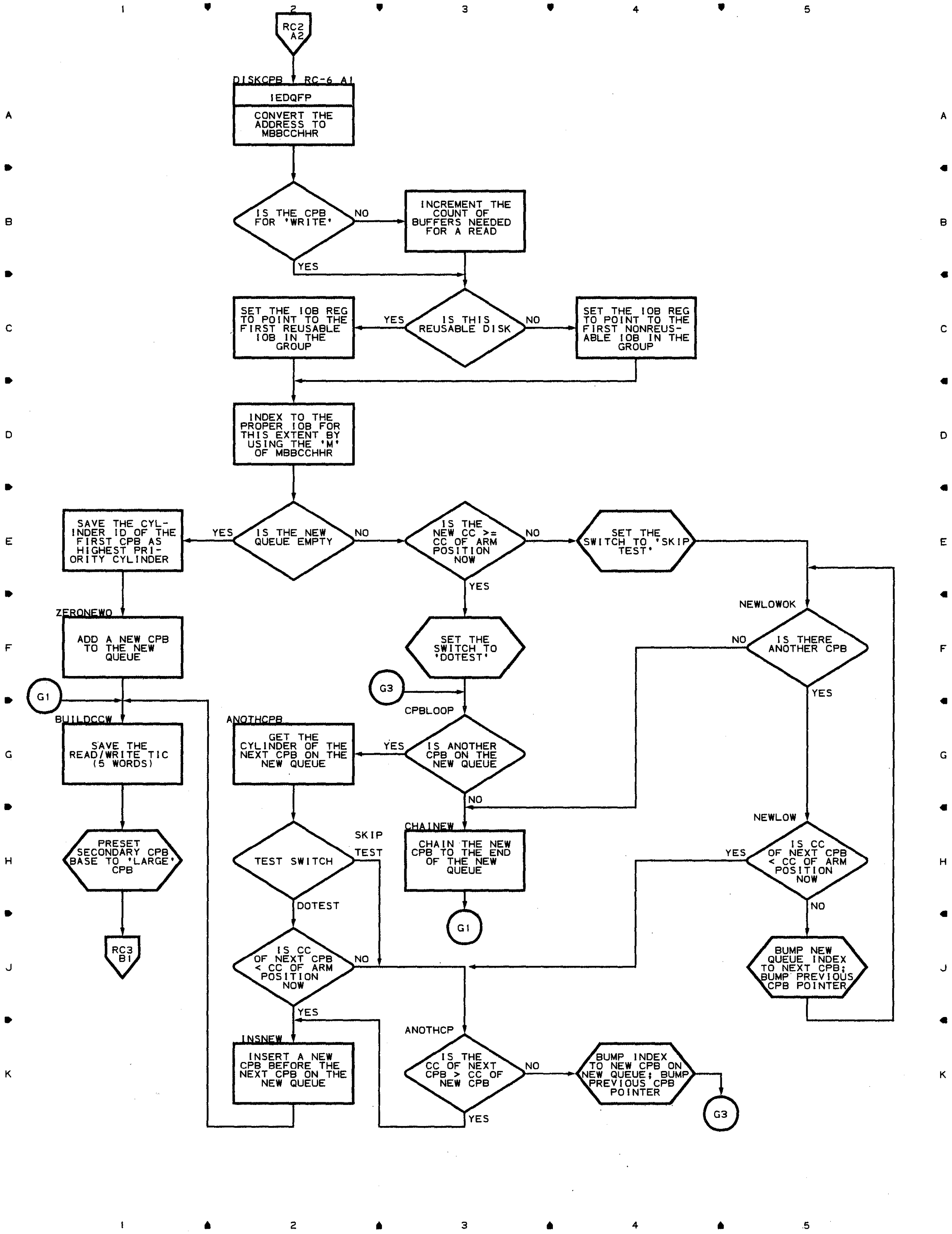
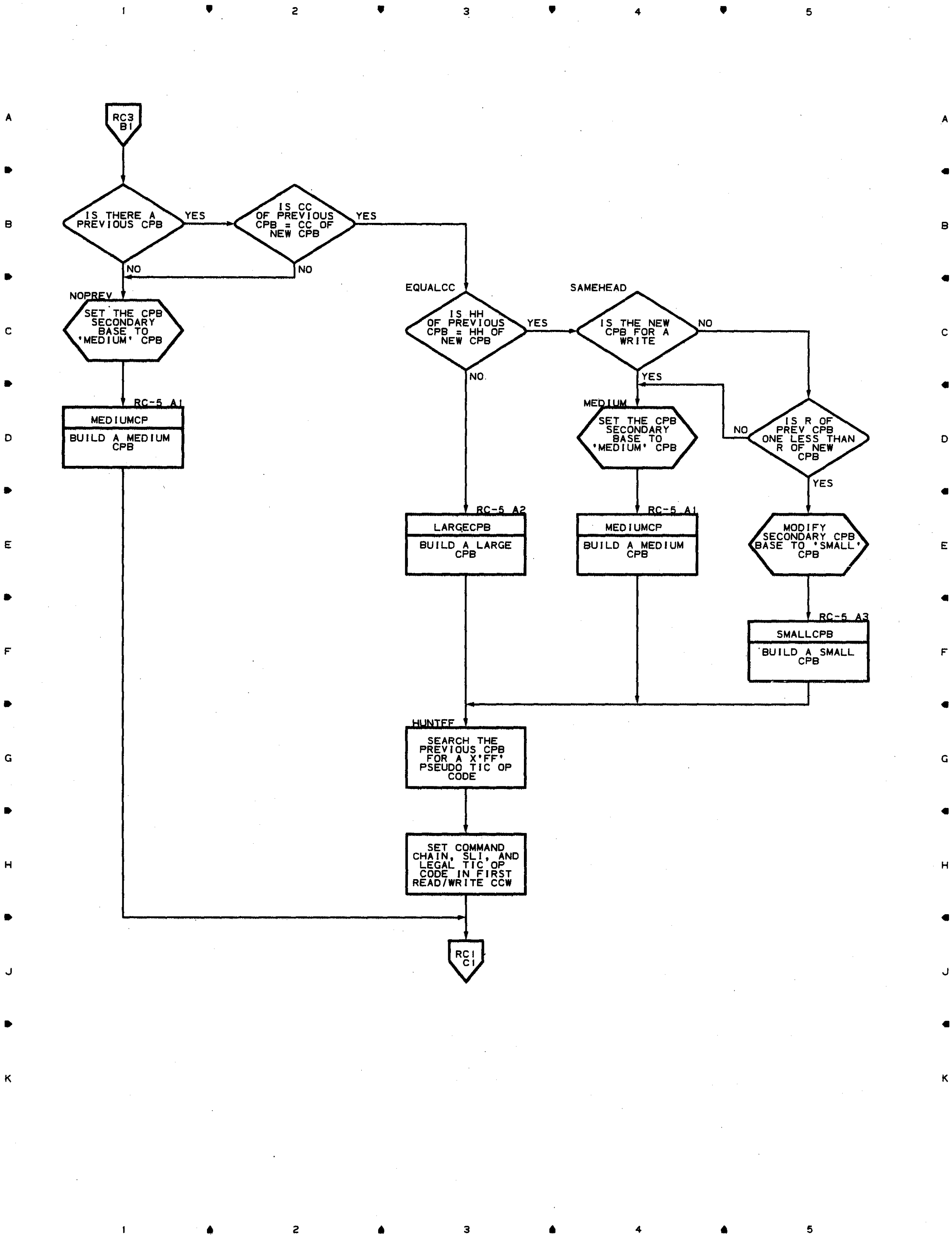

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
RC2 A2
DISKCPB
RC-6 A1
IEDQFP
CONVERT THE ADDRESS TO MBBCCHHR
IS THE CPB FOR 'WRITE'
NO
YES
INCREMENT THE COUNT OF BUFFERS NEEDED FOR A READ
SET THE IOB REG TO POINT TO THE FIRST REUSABLE IOB IN THE GROUP
YES
IS THIS REUSABLE DISK
NO
SET THE IOB REG TO POINT TO THE FIRST NONREUSABLE IOB IN THE GROUP
INDEX TO THE PROPER IOB FOR THIS EXTENT BY USING THE 'M' OF MBBCCHHR
SAVE THE CYLINDER ID OF THE FIRST CPB AS HIGHEST PRIORITY CYLINDER
YES
IS THE NEW QUEUE EMPTY
NO
IS THE NEW CC >= CC OF ARM POSITION NOW
NO
SET THE SWITCH TO 'SKIP TEST'
YES
ZERONEWQ
ADD A NEW CPB TO THE NEW QUEUE
SET THE SWITCH TO 'DOTEST'
NEWLOWOK
NO
IS THERE ANOTHER CPB
YES
G1
G3
CPBLOOP
BUILDCCW
SAVE THE READ/WRITE TIC (5 WORDS)
ANOTHCPB
GET THE CYLINDER OF THE NEXT CPB ON THE NEW QUEUE
YES
IS ANOTHER CPB ON THE NEW QUEUE
NO
NEWLOW
PRESET SECONDARY CPB BASE TO 'LARGE' CPB
TEST SWITCH
SKIP TEST
CHAINEW
CHAIN THE NEW CPB TO THE END OF THE NEW QUEUE
YES
IS CC OF NEXT CPB < CC OF ARM POSITION NOW
NO
DOTEST
G1
RC3 B1
IS CC OF NEXT CPB < CC OF ARM POSITION NOW
NO
BUMP NEW QUEUE INDEX TO NEXT CPB; BUMP PREVIOUS CPB POINTER
YES
INSNEW
ANOTHCP
INSERT A NEW CPB BEFORE THE NEXT CPB ON THE NEW QUEUE
IS THE CC OF NEXT CPB > CC OF NEW CPB
NO
BUMP INDEX TO NEW CPB ON NEW QUEUE; BUMP PREVIOUS CPB POINTER
YES
G3

Chart RC-3 EXCP DRIVER



RC3 B1
IS THERE A PREVIOUS CPB
YES
IS CC OF PREVIOUS CPB = CC OF NEW CPB
YES
NO
NO
NOPREV
SET THE CPB SECONDARY BASE TO 'MEDIUM' CPB
RC-5 A1
MEDIUMCP
BUILD A MEDIUM CPB
EQUALCC
IS HH OF PREVIOUS CPB = HH OF NEW CPB
YES
NO
SAMEHEAD
IS THE NEW CPB FOR A WRITE
NO
YES
MEDIUM
SET THE CPB SECONDARY BASE TO 'MEDIUM' CPB
NO
IS R OF PREV CPB ONE LESS THAN R OF NEW CPB
YES
RC-5 A2
LARGECPB
BUILD A LARGE CPB
RC-5 A1
MEDIUMCP
BUILD A MEDIUM CPB
MODIFY SECONDARY CPB BASE TO 'SMALL' CPB
RC-5 A3
SMALLCPB
BUILD A SMALL CPB
HUNTFF
SEARCH THE PREVIOUS CPB FOR A X'FF' PSEUDO TIC OP CODE
SET COMMAND CHAIN, SLI, AND LEGAL TIC OP CODE IN FIRST READ/WRITE CCW
RC1 C1
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart RC-4 EXCP DRIVER

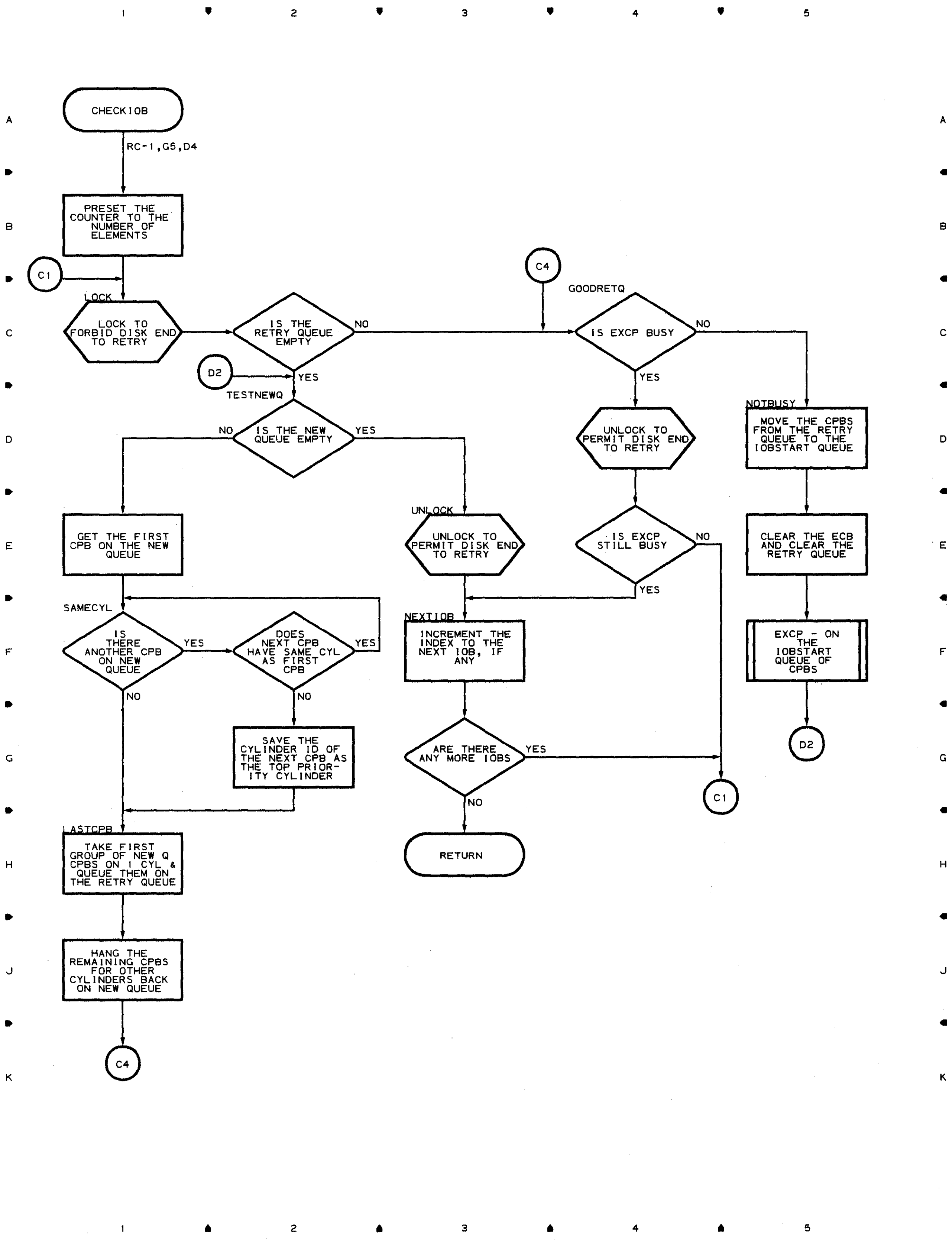
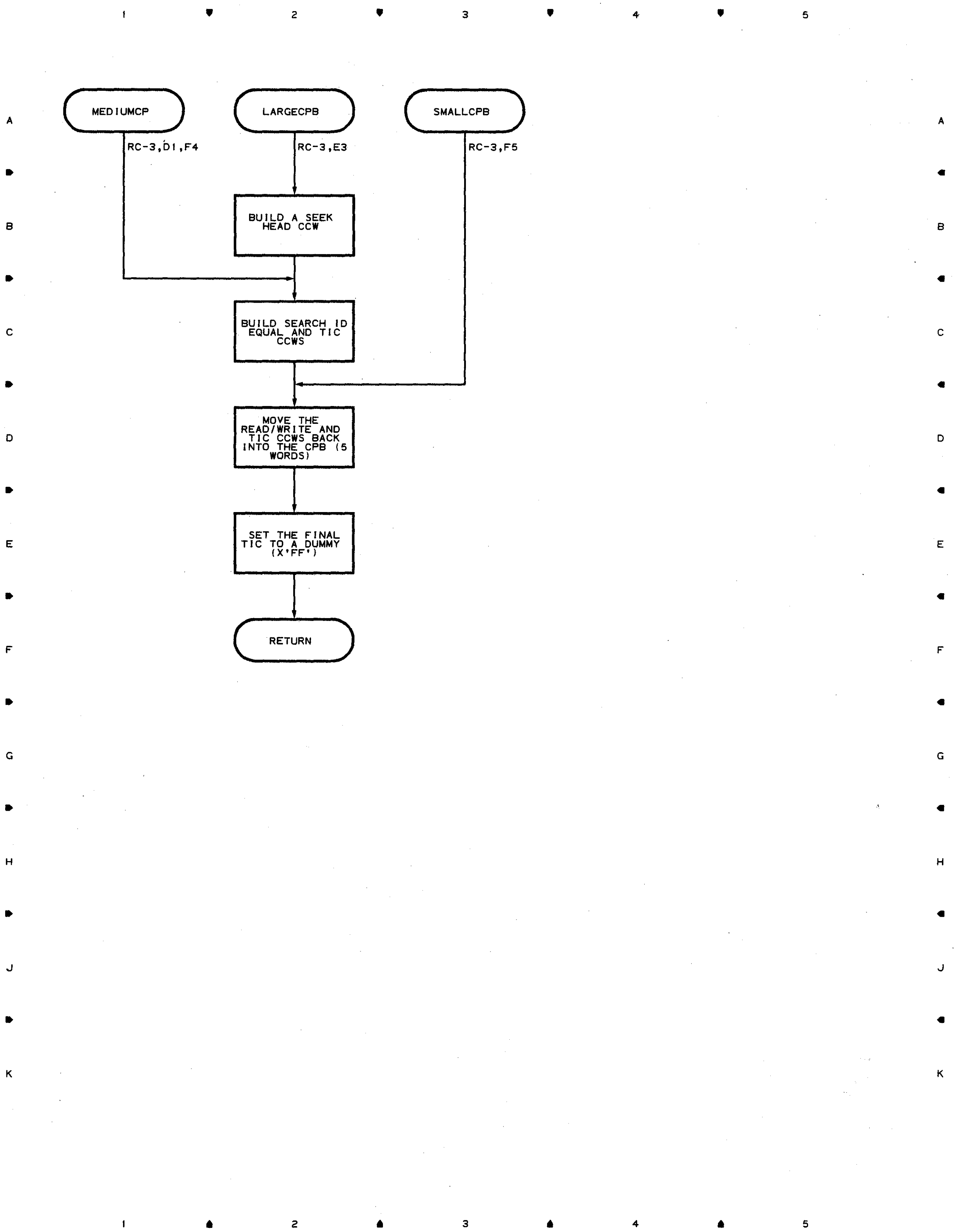

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
CHECKIOB
RC-1,G5,D4
PRESET THE COUNTER TO THE NUMBER OF ELEMENTS
C1
LOCK
LOCK TO FORBID DISK END TO RETRY
IS THE RETRY QUEUE EMPTY
NO
YES
D2
TESTNEWQ
IS THE NEW QUEUE EMPTY
NO
YES
GET THE FIRST CPB ON THE NEW QUEUE
SAMECYL
IS THERE ANOTHER CPB ON NEW QUEUE
YES
NO
DOES NEXT CPB HAVE SAME CYL AS FIRST CPB
YES
NO
SAVE THE CYLINDER ID OF THE NEXT CPB AS THE TOP PRIOR-ITY CYLINDER
LASTCPB
TAKE FIRST GROUP OF NEW Q CPBS ON 1 CYL & QUEUE THEM ON THE RETRY QUEUE
HANG THE REMAINING CPBS FOR OTHER CYLINDERS BACK ON NEW QUEUE
C4
UNLOCK
UNLOCK TO PERMIT DISK END TO RETRY
NEXTIOB
INCREMENT THE INDEX TO THE NEXT IOB, IF ANY
ARE THERE ANY MORE IOBS
YES
NO
RETURN
C4
GOODRETQ
IS EXCP BUSY
NO
YES
UNLOCK TO PERMIT DISK END TO RETRY
IS EXCP STILL BUSY
NO
YES
C1
NOTBUSY
MOVE THE CPBS FROM THE RETRY QUEUE TO THE IOBSTART QUEUE
CLEAR THE ECB AND CLEAR THE RETRY QUEUE
EXCP - ON THE IOBSTART QUEUE OF CPBS
D2

Chart RC-5 EXCP DRIVER



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
MEDIUMCP
RC-3,D1,F4
LARGECPB
RC-3,E3
SMALLCPB
RC-3,F5
BUILD A SEEK HEAD CCW
BUILD SEARCH ID EQUAL AND TIC CCWS
MOVE THE READ/WRITE AND TIC CCWS BACK INTO THE CPB (5 WORDS)
SET THE FINAL TIC TO A DUMMY (X'FF')
RETURN

Chart RC-6 EXCP DRIVER

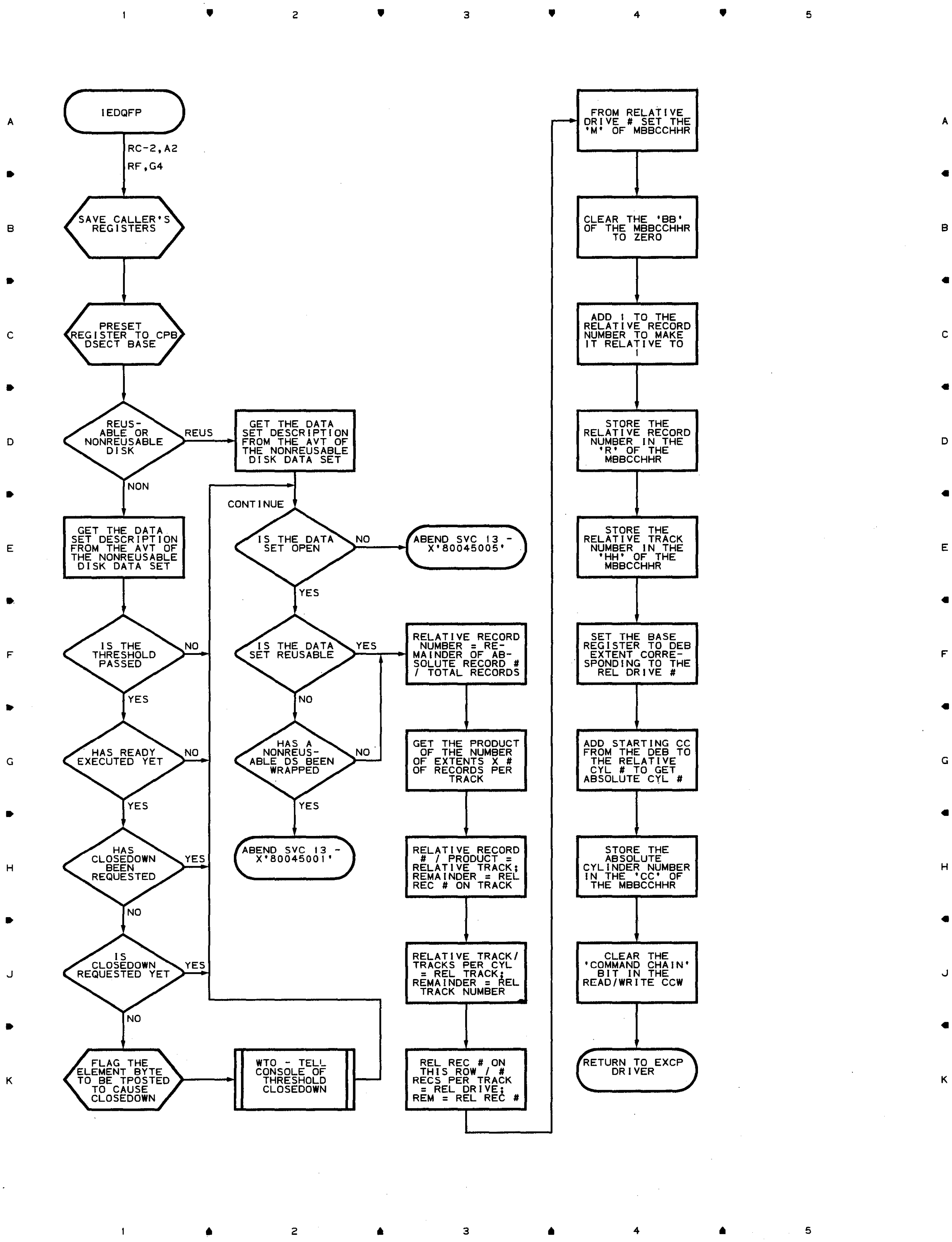
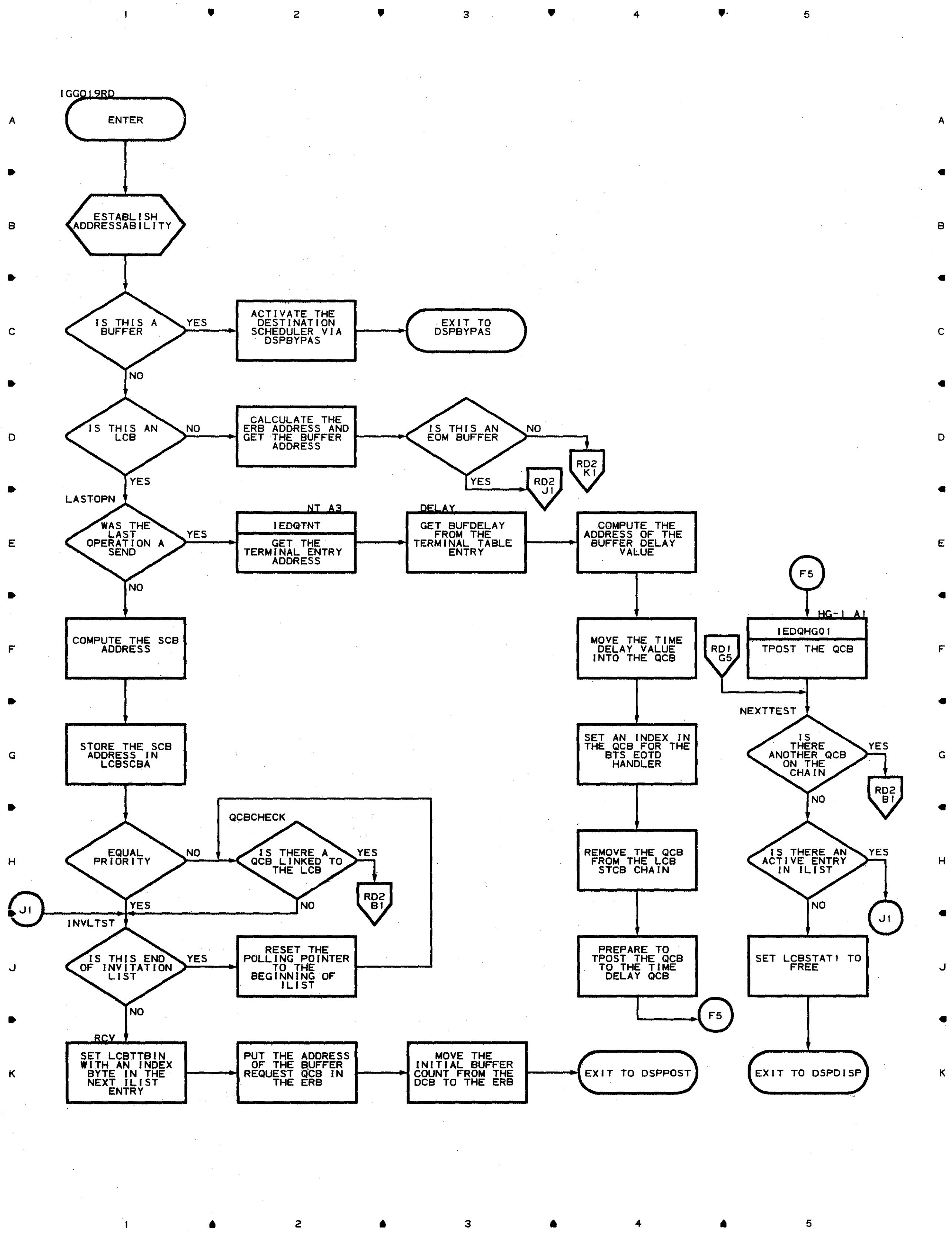

IEDQFP
RC-2,A2
RF,G4
SAVE CALLER'S REGISTERS
PRESET REGISTER TO CPB DSECT BASE
REUS-ABLE OR NONREUSABLE DISK
REUS
NON
GET THE DATA SET DESCRIPTION FROM THE AVT OF THE NONREUSABLE DISK DATA SET
GET THE DATA SET DESCRIPTION FROM THE AVT OF THE NONREUSABLE DISK DATA SET
IS THE THRESHOLD PASSED
NO
YES
HAS READY EXECUTED YET
NO
YES
HAS CLOSEDOWN BEEN REQUESTED
YES
NO
IS CLOSEDOWN REQUESTED YET
YES
NO
FLAG THE ELEMENT BYTE TO BE TPOSTED TO CAUSE CLOSEDOWN
WTO - TELL CONSOLE OF THRESHOLD CLOSEDOWN
CONTINUE
IS THE DATA SET OPEN
NO
ABEND SVC 13 - X'80045005'
YES
IS THE DATA SET REUSABLE
YES
NO
HAS A NONREUS-ABLE DS BEEN WRAPPED
NO
YES
ABEND SVC 13 - X'80045001'
RELATIVE RECORD NUMBER = RE-MAINDER OF AB-SOLUTE RECORD # / TOTAL RECORDS
GET THE PRODUCT OF THE NUMBER OF EXTENTS X # OF RECORDS PER TRACK
RELATIVE RECORD # / PRODUCT = RELATIVE TRACK; REMAINDER = REL REC # ON TRACK
RELATIVE TRACK/ TRACKS PER CYL = REL TRACK; REMAINDER = REL TRACK NUMBER
REL REC # ON THIS ROW / # RECS PER TRACK = REL DRIVE; REM = REL REC #
FROM RELATIVE DRIVE # SET THE 'M' OF MBBCCHHR
CLEAR THE 'BB' OF THE MBBCCHHR TO ZERO
ADD 1 TO THE RELATIVE RECORD NUMBER TO MAKE IT RELATIVE TO 1
STORE THE RELATIVE RECORD NUMBER IN THE 'R' OF THE MBBCCHHR
STORE THE RELATIVE TRACK NUMBER IN THE 'HH' OF THE MBBCCHHR
SET THE BASE REGISTER TO DEB EXTENT CORRE-SPONDING TO THE REL DRIVE #
ADD STARTING CC FROM THE DEB TO THE RELATIVE CYL # TO GET ABSOLUTE CYL #
STORE THE ABSOLUTE CYLINDER NUMBER IN THE 'CC' OF THE MBBCCHHR
CLEAR THE 'COMMAND CHAIN' BIT IN THE READ/WRITE CCW
RETURN TO EXCP DRIVER
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart RD-1 BUFFERED TERMINAL SCHEDULER



IGG019RD
ENTER
ESTABLISH ADDRESSABILITY
IS THIS A BUFFER
YES
ACTIVATE THE DESTINATION SCHEDULER VIA DSPBYPAS
EXIT TO DSPBYPAS
NO
IS THIS AN LCB
NO
CALCULATE THE ERB ADDRESS AND GET THE BUFFER ADDRESS
IS THIS AN EOM BUFFER
NO
RD2 K1
YES
RD2 J1
YES
LASTOPN
WAS THE LAST OPERATION A SEND
YES
NT A3
IEDQTNT
GET THE TERMINAL ENTRY ADDRESS
DELAY
GET BUFDELAY FROM THE TERMINAL TABLE ENTRY
COMPUTE THE ADDRESS OF THE BUFFER DELAY VALUE
F5
NO
COMPUTE THE SCB ADDRESS
MOVE THE TIME DELAY VALUE INTO THE QCB
RD1 G5
HG-1 A1
IEDQHG01
TPOST THE QCB
NEXTTEST
STORE THE SCB ADDRESS IN LCBSCBA
SET AN INDEX IN THE QCB FOR THE BTS EOTD HANDLER
IS THERE ANOTHER QCB ON THE CHAIN
YES
RD2 B1
NO
QCBCHECK
EQUAL PRIORITY
NO
IS THERE A QCB LINKED TO THE LCB
YES
RD2 B1
REMOVE THE QCB FROM THE LCB STCB CHAIN
IS THERE AN ACTIVE ENTRY IN ILIST
YES
J1
YES
NO
NO
J1
INVLTST
IS THIS END OF INVITATION LIST
YES
RESET THE POLLING POINTER TO THE BEGINNING OF ILIST
PREPARE TO TPOST THE QCB TO THE TIME DELAY QCB
SET LCBSTAT1 TO FREE
NO
F5
RCV
SET LCBTTBIN WITH AN INDEX BYTE IN THE NEXT ILIST ENTRY
PUT THE ADDRESS OF THE BUFFER REQUEST QCB IN THE ERB
MOVE THE INITIAL BUFFER COUNT FROM THE DCB TO THE ERB
EXIT TO DSPPOST
EXIT TO DSPDISP
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart RD-2 BUFFERED TERMINAL SCHEDULER

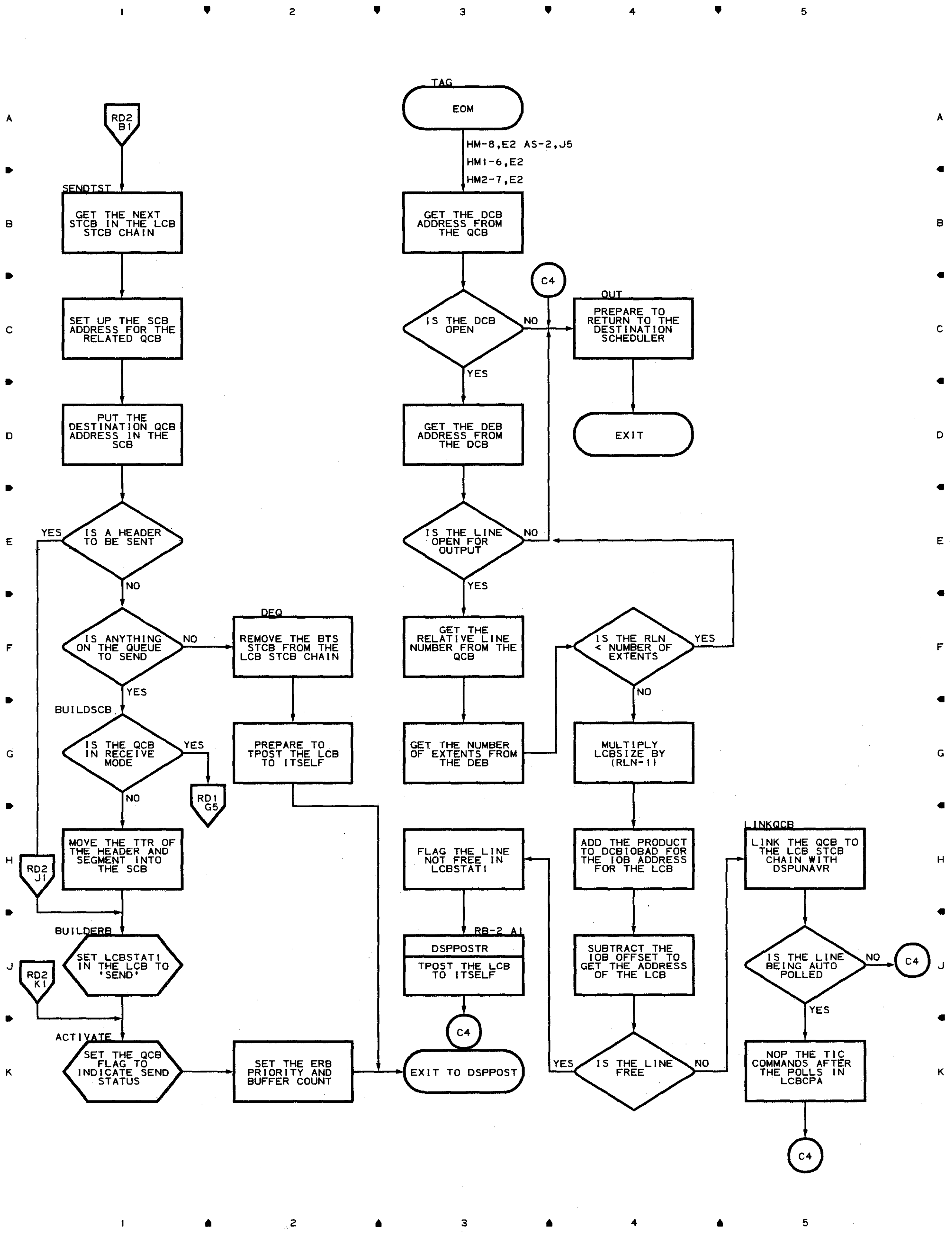

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
RD2 B1
SENDTST
GET THE NEXT STCB IN THE LCB STCB CHAIN
SET UP THE SCB ADDRESS FOR THE RELATED QCB
PUT THE DESTINATION QCB ADDRESS IN THE SCB
IS A HEADER TO BE SENT
YES
NO
IS ANYTHING ON THE QUEUE TO SEND
NO
YES
DEQ
REMOVE THE BTS STCB FROM THE LCB STCB CHAIN
PREPARE TO TPOST THE LCB TO ITSELF
BUILDSCB
IS THE QCB IN RECEIVE MODE
YES
NO
RD1 G5
MOVE THE TTR OF THE HEADER AND SEGMENT INTO THE SCB
RD2 J1
BUILDERB
SET LCBSTAT1 IN THE LCB TO 'SEND'
RD2 K1
ACTIVATE
SET THE QCB FLAG TO INDICATE SEND STATUS
SET THE ERB PRIORITY AND BUFFER COUNT
TAG
EOM
HM-8,E2 AS-2,J5
HM1-6,E2
HM2-7,E2
GET THE DCB ADDRESS FROM THE QCB
IS THE DCB OPEN
NO
YES
C4
OUT
PREPARE TO RETURN TO THE DESTINATION SCHEDULER
EXIT
GET THE DEB ADDRESS FROM THE DCB
IS THE LINE OPEN FOR OUTPUT
NO
YES
GET THE RELATIVE LINE NUMBER FROM THE QCB
GET THE NUMBER OF EXTENTS FROM THE DEB
IS THE RLN < NUMBER OF EXTENTS
YES
NO
MULTIPLY LCBSIZE BY (RLN-1)
ADD THE PRODUCT TO DCBIOBAD FOR THE IOB ADDRESS FOR THE LCB
SUBTRACT THE IOB OFFSET TO GET THE ADDRESS OF THE LCB
IS THE LINE FREE
YES
NO
FLAG THE LINE NOT FREE IN LCBSTAT1
RB-2 A1
DSPPOSTR
TPOST THE LCB TO ITSELF
C4
EXIT TO DSPPOST
LINKQCB
LINK THE QCB TO THE LCB STCB CHAIN WITH DSPUNAVR
IS THE LINE BEING AUTO POLLED
NO
C4
YES
NOP THE TIC COMMANDS AFTER THE POLLS IN LCBCPA
C4

Chart RD-3 BUFFERED TERMINAL SCHEDULER

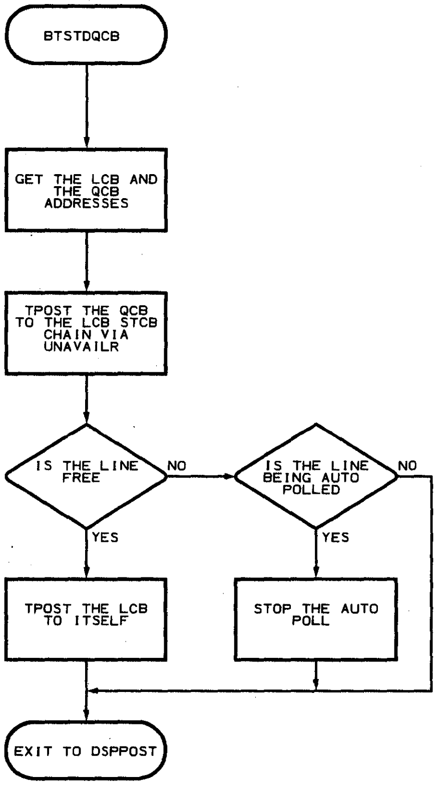

BTSTDQCB
GET THE LCB AND THE QCB ADDRESSES
TPOST THE QCB TO THE LCB STCB CHAIN VIA UNAVAILR
IS THE LINE FREE
NO
YES
IS THE LINE BEING AUTO POLLED
NO
YES
TPOST THE LCB TO ITSELF
STOP THE AUTO POLL
EXIT TO DSPPOST

Chart RF EXCP DRIVER FOR A SINGLE CPB

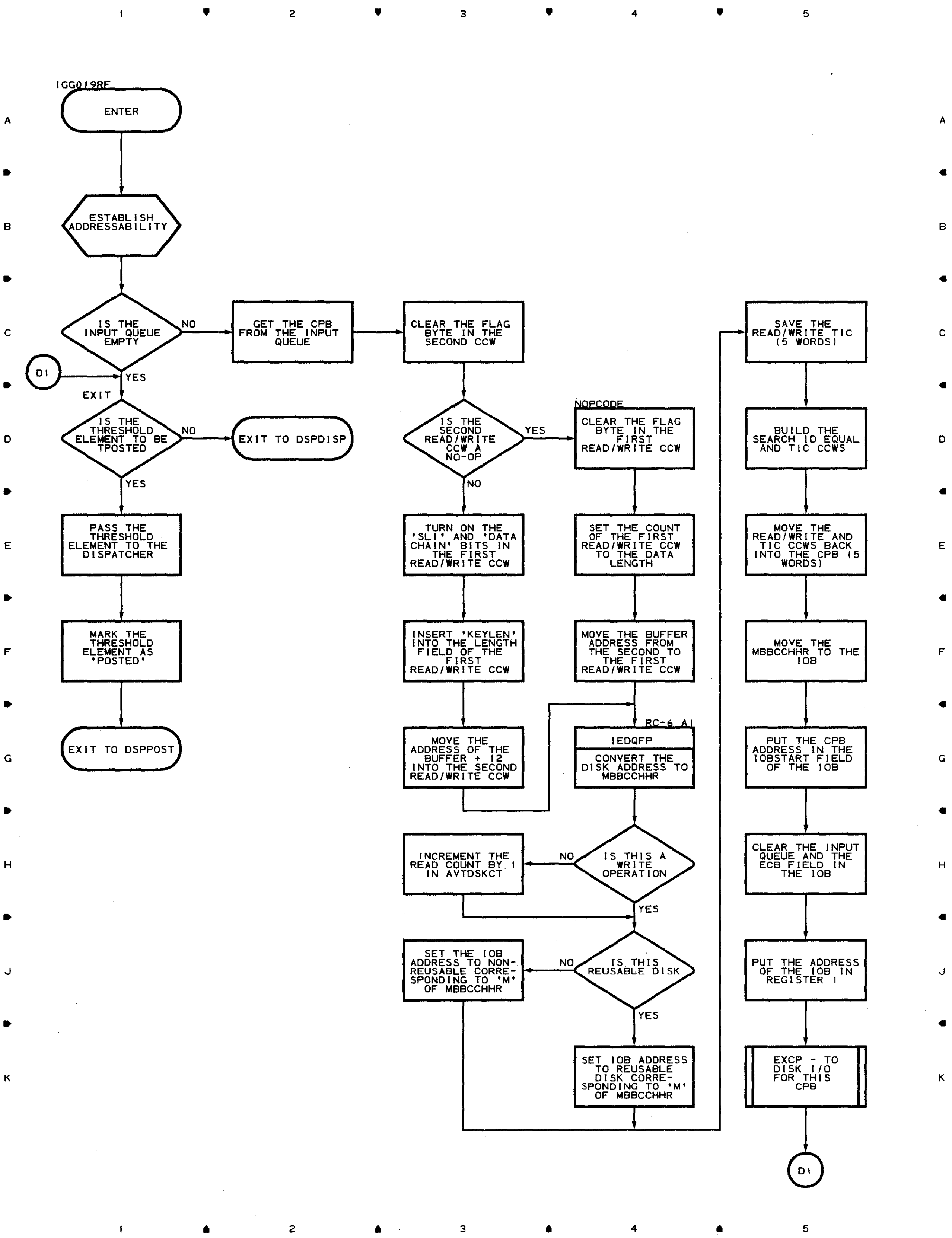
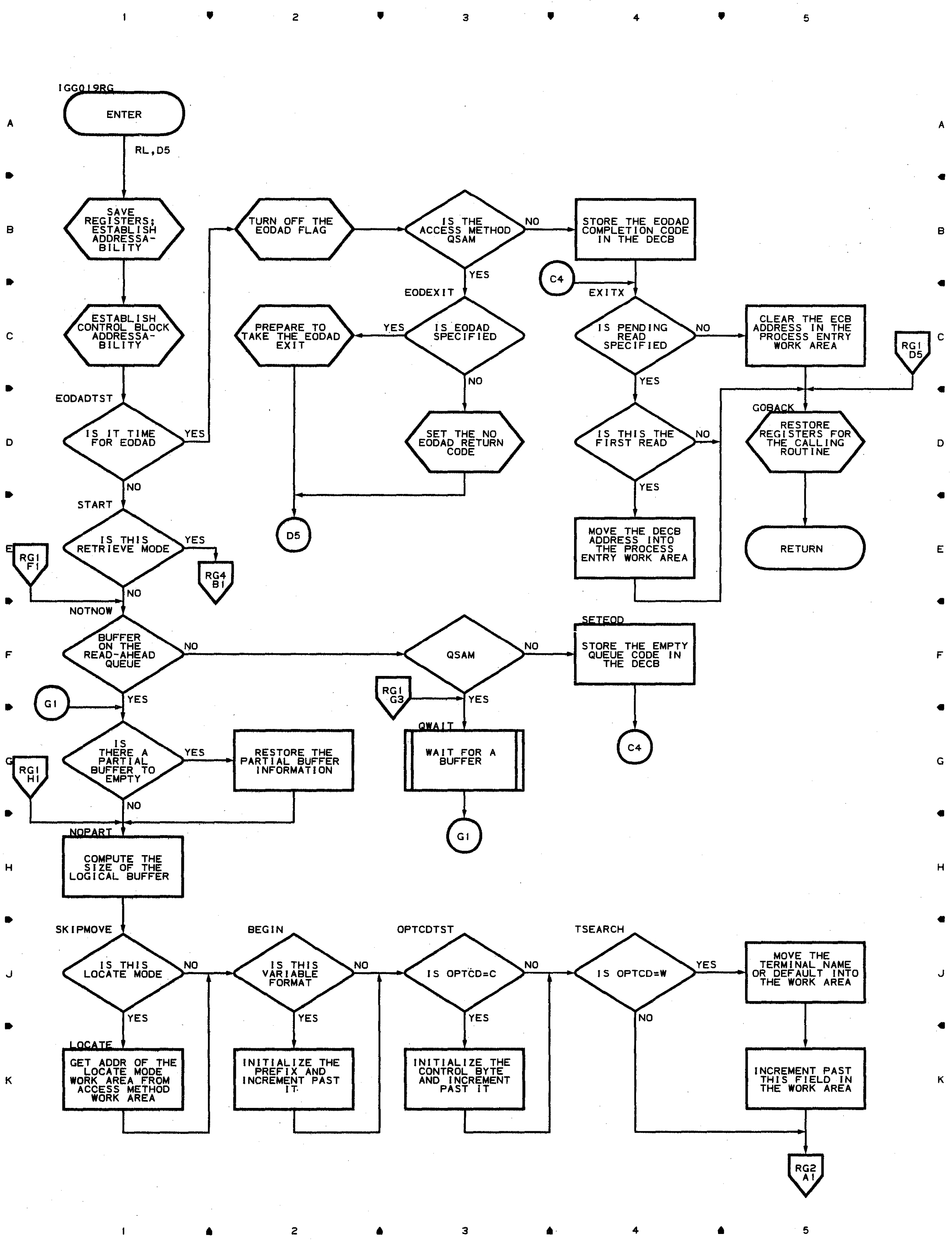

IGG019RF
ENTER
ESTABLISH ADDRESSABILITY
IS THE INPUT QUEUE EMPTY
NO
YES
D1
EXIT
IS THE THRESHOLD ELEMENT TO BE TPOSTED
NO
EXIT TO DSPDISP
YES
PASS THE THRESHOLD ELEMENT TO THE DISPATCHER
MARK THE THRESHOLD ELEMENT AS 'POSTED'
EXIT TO DSPPOST
GET THE CPB FROM THE INPUT QUEUE
CLEAR THE FLAG BYTE IN THE SECOND CCW
IS THE SECOND READ/WRITE CCW A NO-OP
YES
NO
TURN ON THE 'SLI' AND 'DATA CHAIN' BITS IN THE FIRST READ/WRITE CCW
INSERT 'KEYLEN' INTO THE LENGTH FIELD OF THE FIRST READ/WRITE CCW
MOVE THE ADDRESS OF THE BUFFER + 12 INTO THE SECOND READ/WRITE CCW
INCREMENT THE READ COUNT BY 1 IN AVTDSKCT
SET THE IOB ADDRESS TO NON-REUSABLE CORRE-SPONDING TO 'M' OF MBBCCHHR
NOPCODE
CLEAR THE FLAG BYTE IN THE FIRST READ/WRITE CCW
SET THE COUNT OF THE FIRST READ/WRITE CCW TO THE DATA LENGTH
MOVE THE BUFFER ADDRESS FROM THE SECOND TO THE FIRST READ/WRITE CCW
RC-6 A1
IEDQFP
CONVERT THE DISK ADDRESS TO MBBCCHHR
IS THIS A WRITE OPERATION
NO
YES
IS THIS REUSABLE DISK
NO
YES
SET IOB ADDRESS TO REUSABLE DISK CORRE-SPONDING TO 'M' OF MBBCCHHR
SAVE THE READ/WRITE TIC (5 WORDS)
BUILD THE SEARCH ID EQUAL AND TIC CCWS
MOVE THE READ/WRITE AND TIC CCWS BACK INTO THE CPB (5 WORDS)
MOVE THE MBBCCHHR TO THE IOB
PUT THE CPB ADDRESS IN THE IOBSTART FIELD OF THE IOB
CLEAR THE INPUT QUEUE AND THE ECB FIELD IN THE IOB
PUT THE ADDRESS OF THE IOB IN REGISTER 1
EXCP - TO DISK I/O FOR THIS CPB
D1
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart RG-1 GET/READ ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG019RG
ENTER
RL,D5
SAVE REGISTERS; ESTABLISH ADDRESSA-BILITY
TURN OFF THE EODAD FLAG
IS THE ACCESS METHOD QSAM
NO
STORE THE EODAD COMPLETION CODE IN THE DECB
YES
EODEXIT
C4
EXITX
ESTABLISH CONTROL BLOCK ADDRESSA-BILITY
PREPARE TO TAKE THE EODAD EXIT
YES
IS EODAD SPECIFIED
IS PENDING READ SPECIFIED
NO
CLEAR THE ECB ADDRESS IN THE PROCESS ENTRY WORK AREA
RG1 D5
NO
YES
EODADTST
GOBACK
IS IT TIME FOR EODAD
YES
SET THE NO EODAD RETURN CODE
IS THIS THE FIRST READ
NO
RESTORE REGISTERS FOR THE CALLING ROUTINE
NO
YES
START
IS THIS RETRIEVE MODE
YES
D5
MOVE THE DECB ADDRESS INTO THE PROCESS ENTRY WORK AREA
RETURN
RG1 F1
RG4 B1
NO
NOTNOW
SETEOD
BUFFER ON THE READ-AHEAD QUEUE
NO
QSAM
NO
STORE THE EMPTY QUEUE CODE IN THE DECB
G1
YES
RG1 G3
YES
QWAIT
IS THERE A PARTIAL BUFFER TO EMPTY
YES
RESTORE THE PARTIAL BUFFER INFORMATION
WAIT FOR A BUFFER
C4
RG1 H1
NO
G1
NOPART
COMPUTE THE SIZE OF THE LOGICAL BUFFER
SKIPMOVE
BEGIN
OPTCDTST
TSEARCH
IS THIS LOCATE MODE
NO
IS THIS VARIABLE FORMAT
NO
IS OPTCD=C
NO
IS OPTCD=W
YES
MOVE THE TERMINAL NAME OR DEFAULT INTO THE WORK AREA
YES
YES
YES
NO
LOCATE
GET ADDR OF THE LOCATE MODE WORK AREA FROM ACCESS METHOD WORK AREA
INITIALIZE THE PREFIX AND INCREMENT PAST IT
INITIALIZE THE CONTROL BYTE AND INCREMENT PAST IT
INCREMENT PAST THIS FIELD IN THE WORK AREA
RG2 A1

Chart RG-2 GET/READ ROUTINE

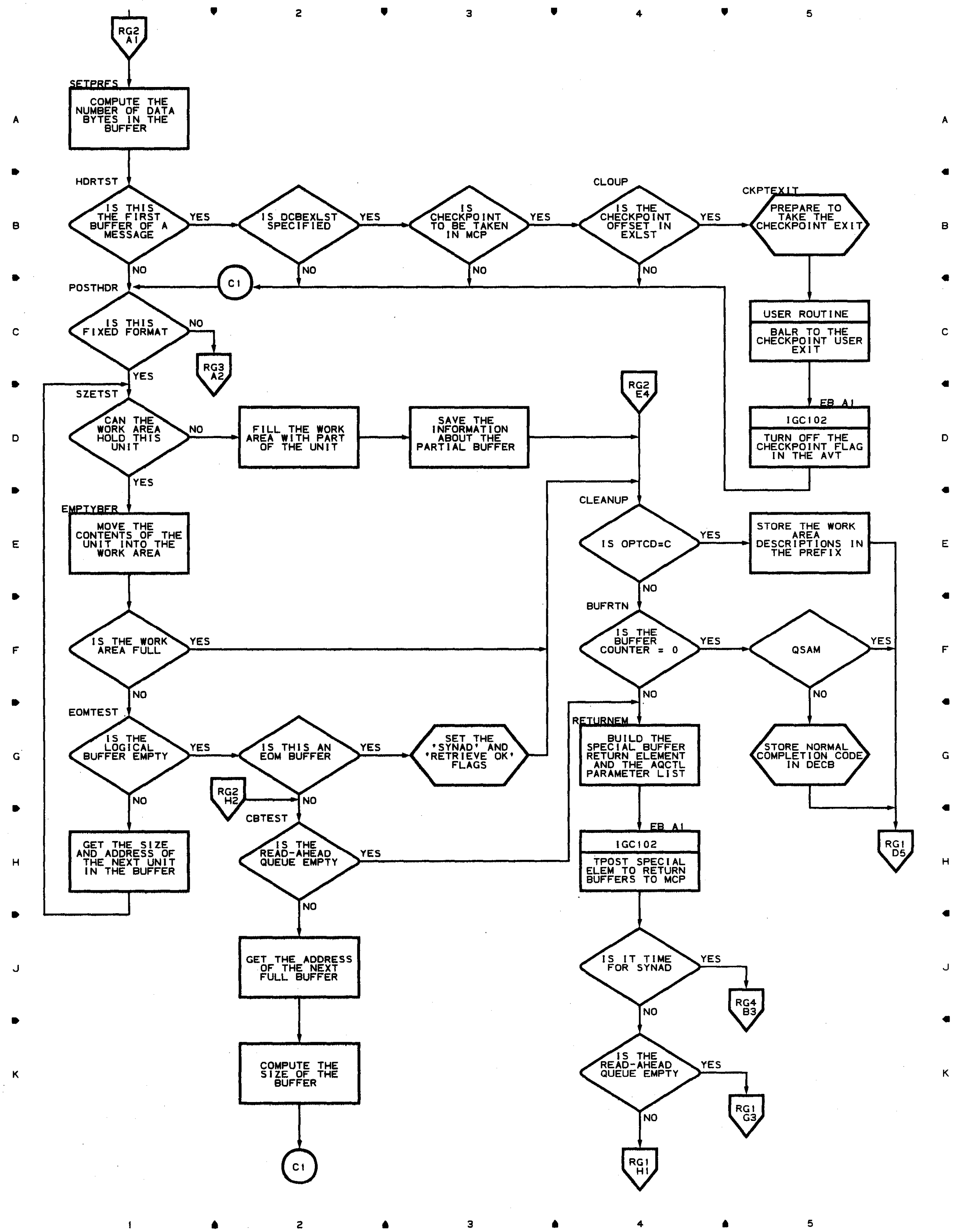

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
RG2 A1
SETPREFS
COMPUTE THE NUMBER OF DATA BYTES IN THE BUFFER
HDRTST
IS THIS THE FIRST BUFFER OF A MESSAGE
YES
IS DCBEXLST SPECIFIED
YES
IS CHECKPOINT TO BE TAKEN IN MCP
YES
CLOUP
IS THE CHECKPOINT OFFSET IN EXLST
YES
CKPTEXIT
PREPARE TO TAKE THE CHECKPOINT EXIT
NO
NO
NO
NO
C1
POSTHDR
IS THIS FIXED FORMAT
NO
RG3 A2
USER ROUTINE
BALR TO THE CHECKPOINT USER EXIT
YES
SZETST
RG2 E4
EB A1
IGC102
TURN OFF THE CHECKPOINT FLAG IN THE AVT
CAN THE WORK AREA HOLD THIS UNIT
NO
FILL THE WORK AREA WITH PART OF THE UNIT
SAVE THE INFORMATION ABOUT THE PARTIAL BUFFER
YES
EMPTYBFR
CLEANUP
MOVE THE CONTENTS OF THE UNIT INTO THE WORK AREA
IS OPTCD=C
YES
STORE THE WORK AREA DESCRIPTIONS IN THE PREFIX
NO
BUFRTN
IS THE WORK AREA FULL
YES
IS THE BUFFER COUNTER = 0
YES
QSAM
YES
NO
NO
NO
EOMTEST
RETURNEM
IS THE LOGICAL BUFFER EMPTY
YES
IS THIS AN EOM BUFFER
YES
SET THE 'SYNAD' AND 'RETRIEVE OK' FLAGS
BUILD THE SPECIAL BUFFER RETURN ELEMENT AND THE AQCTL PARAMETER LIST
STORE NORMAL COMPLETION CODE IN DECB
NO
RG2 H2
NO
CBTEST
EB A1
GET THE SIZE AND ADDRESS OF THE NEXT UNIT IN THE BUFFER
IS THE READ-AHEAD QUEUE EMPTY
YES
IGC102
TPOST SPECIAL ELEM TO RETURN BUFFERS TO MCP
RG1 D5
NO
GET THE ADDRESS OF THE NEXT FULL BUFFER
IS IT TIME FOR SYNAD
YES
RG4 B3
NO
COMPUTE THE SIZE OF THE BUFFER
IS THE READ-AHEAD QUEUE EMPTY
YES
RG1 G3
NO
C1
RG1 H1

Chart RG-3 GET/READ ROUTINE

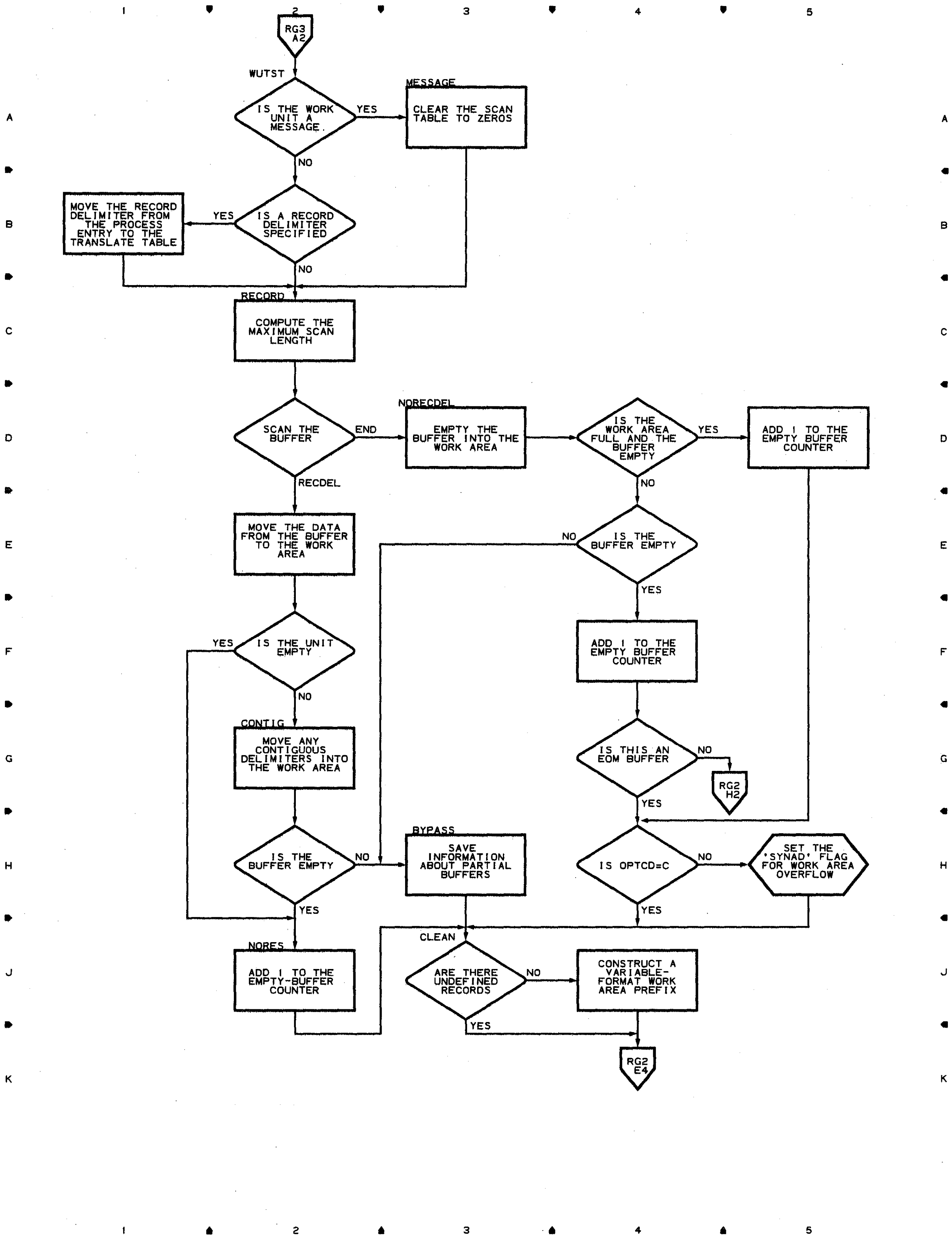

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
RG3 A2
WUTST
IS THE WORK UNIT A MESSAGE
YES
MESSAGE
CLEAR THE SCAN TABLE TO ZEROS
NO
MOVE THE RECORD DELIMITER FROM THE PROCESS ENTRY TO THE TRANSLATE TABLE
YES
IS A RECORD DELIMITER SPECIFIED
NO
RECORD
COMPUTE THE MAXIMUM SCAN LENGTH
SCAN THE BUFFER
END
NORECDEL
EMPTY THE BUFFER INTO THE WORK AREA
IS THE WORK AREA FULL AND THE BUFFER EMPTY
YES
ADD 1 TO THE EMPTY BUFFER COUNTER
RECDEL
NO
MOVE THE DATA FROM THE BUFFER TO THE WORK AREA
NO
IS THE BUFFER EMPTY
YES
YES
IS THE UNIT EMPTY
ADD 1 TO THE EMPTY BUFFER COUNTER
NO
CONTIG
MOVE ANY CONTIGUOUS DELIMITERS INTO THE WORK AREA
IS THIS AN EOM BUFFER
NO
RG2 H2
YES
BYPASS
IS THE BUFFER EMPTY
NO
SAVE INFORMATION ABOUT PARTIAL BUFFERS
IS OPTCD=C
NO
SET THE 'SYNAD' FLAG FOR WORK AREA OVERFLOW
YES
YES
CLEAN
NORES
ADD 1 TO THE EMPTY-BUFFER COUNTER
ARE THERE UNDEFINED RECORDS
NO
CONSTRUCT A VARIABLE-FORMAT WORK AREA PREFIX
YES
RG2 E4

Chart RG-4 GET/READ ROUTINE

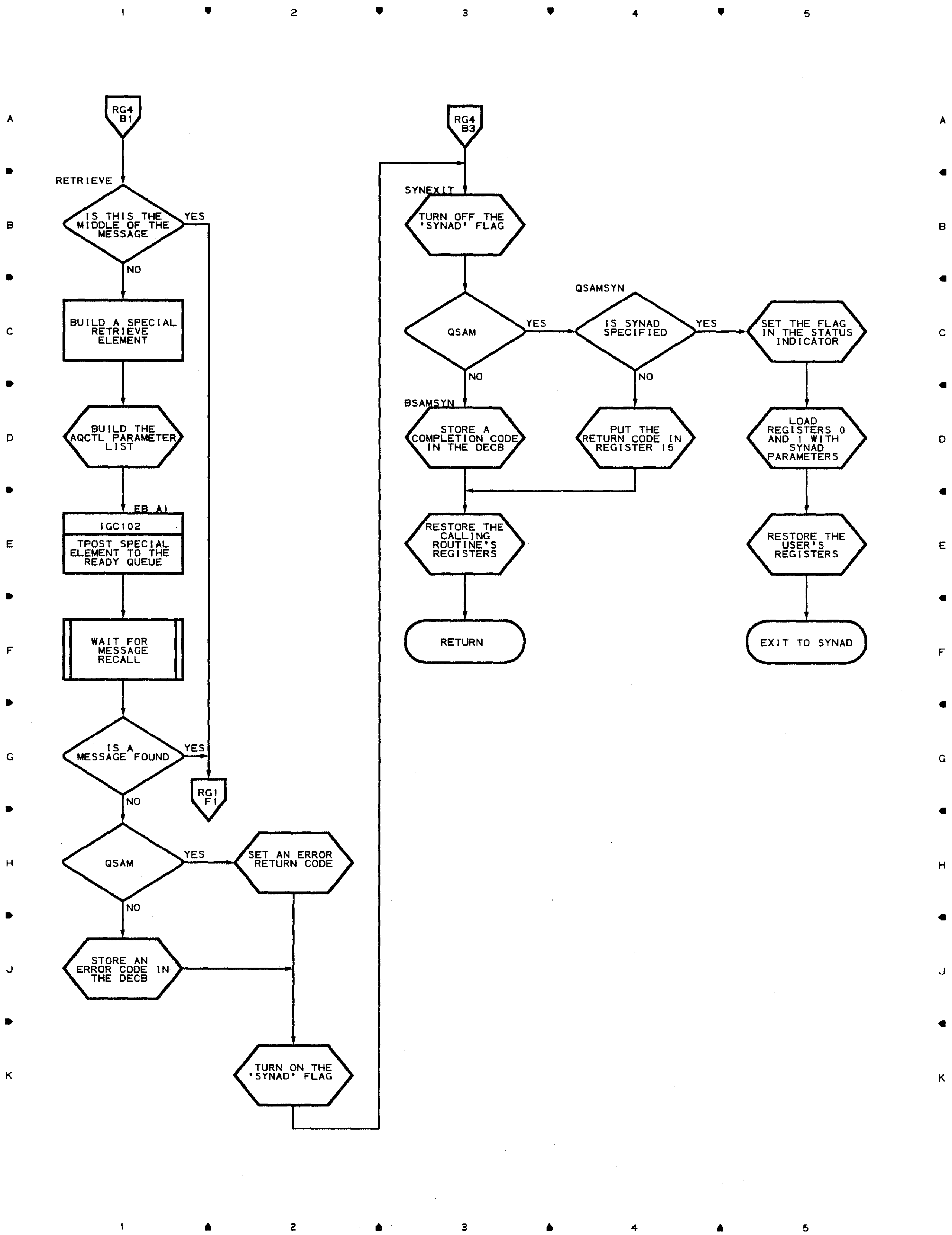

RG4 B1
RETRIEVE
IS THIS THE MIDDLE OF THE MESSAGE
YES
NO
BUILD A SPECIAL RETRIEVE ELEMENT
BUILD THE AQCTL PARAMETER LIST
EB A1
IGC102
TPOST SPECIAL ELEMENT TO THE READY QUEUE
WAIT FOR MESSAGE RECALL
IS A MESSAGE FOUND
YES
NO
RG1 F1
QSAM
YES
NO
SET AN ERROR RETURN CODE
STORE AN ERROR CODE IN THE DECB
TURN ON THE 'SYNAD' FLAG
RG4 B3
SYNEXIT
TURN OFF THE 'SYNAD' FLAG
QSAM
YES
NO
QSAMSYN
IS SYNAD SPECIFIED
YES
NO
SET THE FLAG IN THE STATUS INDICATOR
BSAMSYN
STORE A COMPLETION CODE IN THE DECB
PUT THE RETURN CODE IN REGISTER 15
LOAD REGISTERS 0 AND 1 WITH SYNAD PARAMETERS
RESTORE THE CALLING ROUTINE'S REGISTERS
RESTORE THE USER'S REGISTERS
RETURN
EXIT TO SYNAD
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart RH-1 GET COMPATIBLE ROUTINE

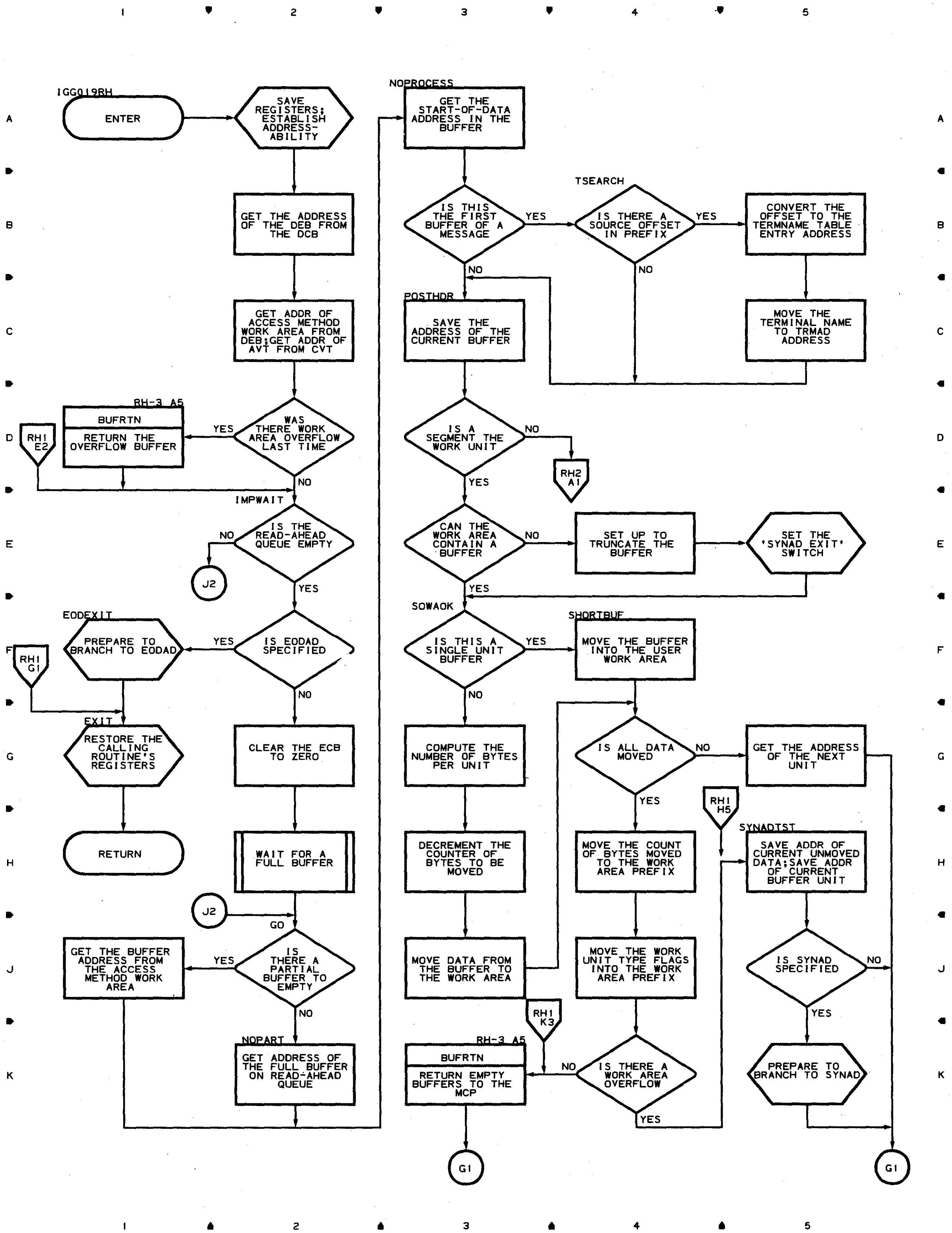

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG019RH
ENTER
SAVE REGISTERS; ESTABLISH ADDRESS-ABILITY
GET THE ADDRESS OF THE DEB FROM THE DCB
GET ADDR OF ACCESS METHOD WORK AREA FROM DEB;GET ADDR OF AVT FROM CVT
WAS THERE WORK AREA OVERFLOW LAST TIME
YES
NO
RH-3 A5
BUFRTN
RETURN THE OVERFLOW BUFFER
RH1 E2
IMPWAIT
IS THE READ-AHEAD QUEUE EMPTY
NO
YES
J2
EODEXIT
PREPARE TO BRANCH TO EODAD
IS EODAD SPECIFIED
RH1 G1
EXIT
RESTORE THE CALLING ROUTINE'S REGISTERS
RETURN
CLEAR THE ECB TO ZERO
WAIT FOR A FULL BUFFER
J2
GO
IS THERE A PARTIAL BUFFER TO EMPTY
GET THE BUFFER ADDRESS FROM THE ACCESS METHOD WORK AREA
NOPART
GET ADDRESS OF THE FULL BUFFER ON READ-AHEAD QUEUE
NOPROCESS
GET THE START-OF-DATA ADDRESS IN THE BUFFER
IS THIS THE FIRST BUFFER OF A MESSAGE
TSEARCH
IS THERE A SOURCE OFFSET IN PREFIX
CONVERT THE OFFSET TO THE TERMNAME TABLE ENTRY ADDRESS
MOVE THE TERMINAL NAME TO TRMAD ADDRESS
POSTHDR
SAVE THE ADDRESS OF THE CURRENT BUFFER
IS A SEGMENT THE WORK UNIT
RH2 A1
CAN THE WORK AREA CONTAIN A BUFFER
SET UP TO TRUNCATE THE BUFFER
SET THE 'SYNAD EXIT' SWITCH
SOWAOK
IS THIS A SINGLE UNIT BUFFER
SHORTBUF
MOVE THE BUFFER INTO THE USER WORK AREA
COMPUTE THE NUMBER OF BYTES PER UNIT
DECREMENT THE COUNTER OF BYTES TO BE MOVED
MOVE DATA FROM THE BUFFER TO THE WORK AREA
IS ALL DATA MOVED
GET THE ADDRESS OF THE NEXT UNIT
MOVE THE COUNT OF BYTES MOVED TO THE WORK AREA PREFIX
MOVE THE WORK UNIT TYPE FLAGS INTO THE WORK AREA PREFIX
RH1 H5
SYNADTST
SAVE ADDR OF CURRENT UNMOVED DATA;SAVE ADDR OF CURRENT BUFFER UNIT
IS SYNAD SPECIFIED
PREPARE TO BRANCH TO SYNAD
RH1 K3
RH-3 A5
BUFRTN
RETURN EMPTY BUFFERS TO THE MCP
IS THERE A WORK AREA OVERFLOW
G1
G1

Chart RH-2 GET COMPATIBLE ROUTINE

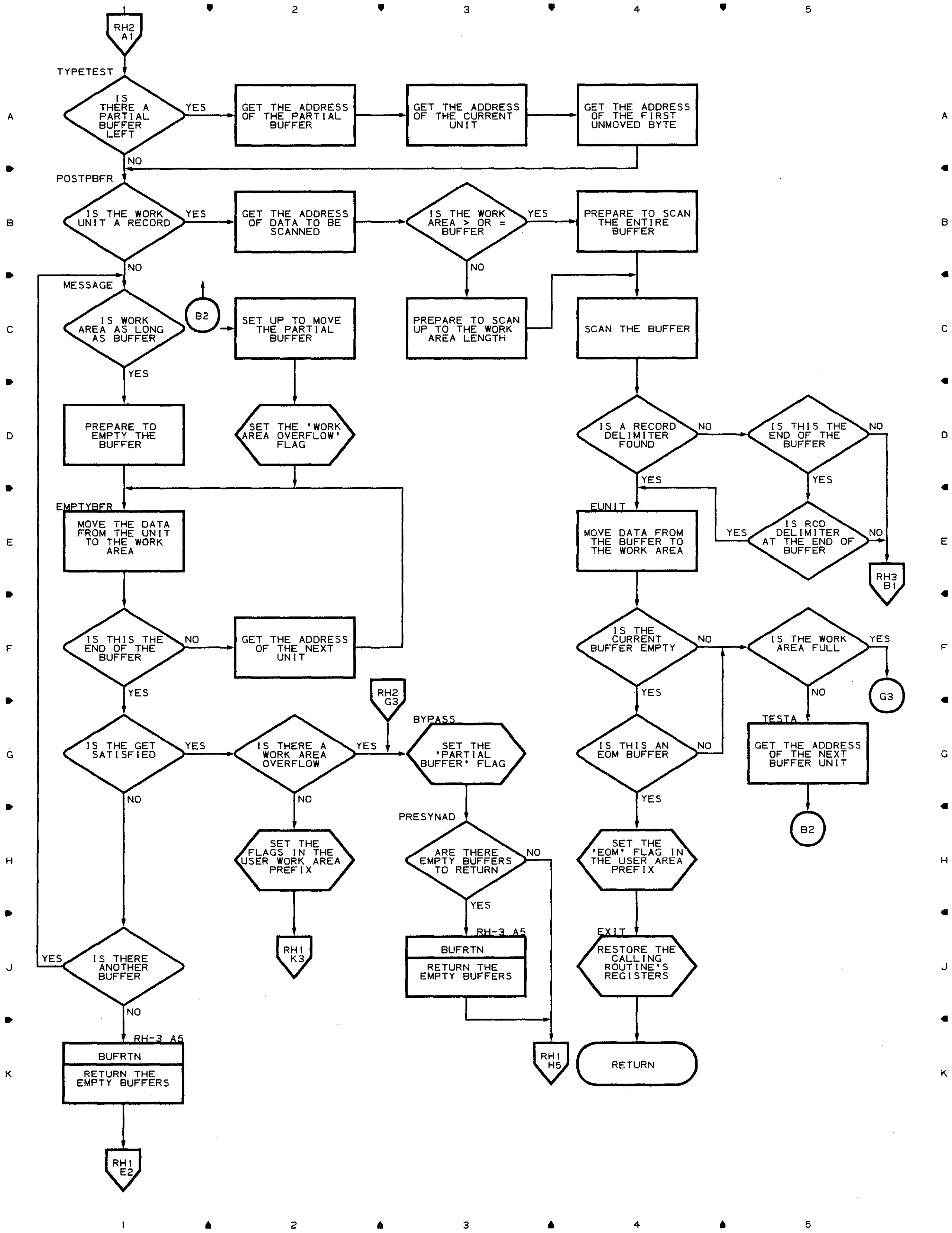

RH2 A1
TYPETEST
IS THERE A PARTIAL BUFFER LEFT
YES
GET THE ADDRESS OF THE PARTIAL BUFFER
GET THE ADDRESS OF THE CURRENT UNIT
GET THE ADDRESS OF THE FIRST UNMOVED BYTE
NO
POSTPBFR
IS THE WORK UNIT A RECORD
YES
GET THE ADDRESS OF DATA TO BE SCANNED
IS THE WORK AREA > OR = BUFFER
YES
PREPARE TO SCAN THE ENTIRE BUFFER
NO
NO
MESSAGE
IS WORK AREA AS LONG AS BUFFER
B2
SET UP TO MOVE THE PARTIAL BUFFER
PREPARE TO SCAN UP TO THE WORK AREA LENGTH
SCAN THE BUFFER
YES
PREPARE TO EMPTY THE BUFFER
SET THE 'WORK AREA OVERFLOW' FLAG
IS A RECORD DELIMITER FOUND
NO
IS THIS THE END OF THE BUFFER
NO
YES
YES
EMPTYBFR
MOVE THE DATA FROM THE UNIT TO THE WORK AREA
EUNIT
MOVE DATA FROM THE BUFFER TO THE WORK AREA
YES
IS RCD DELIMITER AT THE END OF BUFFER
NO
RH3 B1
IS THIS THE END OF THE BUFFER
NO
GET THE ADDRESS OF THE NEXT UNIT
IS THE CURRENT BUFFER EMPTY
NO
IS THE WORK AREA FULL
YES
YES
RH2 G3
YES
NO
G3
BYPASS
TESTA
IS THE GET SATISFIED
YES
IS THERE A WORK AREA OVERFLOW
YES
SET THE 'PARTIAL BUFFER' FLAG
IS THIS AN EOM BUFFER
NO
GET THE ADDRESS OF THE NEXT BUFFER UNIT
NO
NO
PRESYNAD
YES
B2
SET THE FLAGS IN THE USER WORK AREA PREFIX
ARE THERE EMPTY BUFFERS TO RETURN
NO
SET THE 'EOM' FLAG IN THE USER AREA PREFIX
YES
RH-3 A5
EXIT
YES
IS THERE ANOTHER BUFFER
RH1 K3
BUFRTN
RETURN THE EMPTY BUFFERS
RESTORE THE CALLING ROUTINE'S REGISTERS
NO
RH-3 A5
BUFRTN
RETURN THE EMPTY BUFFERS
RH1 H5
RETURN
RH1 E2
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart RH-3 GET COMPATIBLE ROUTINE

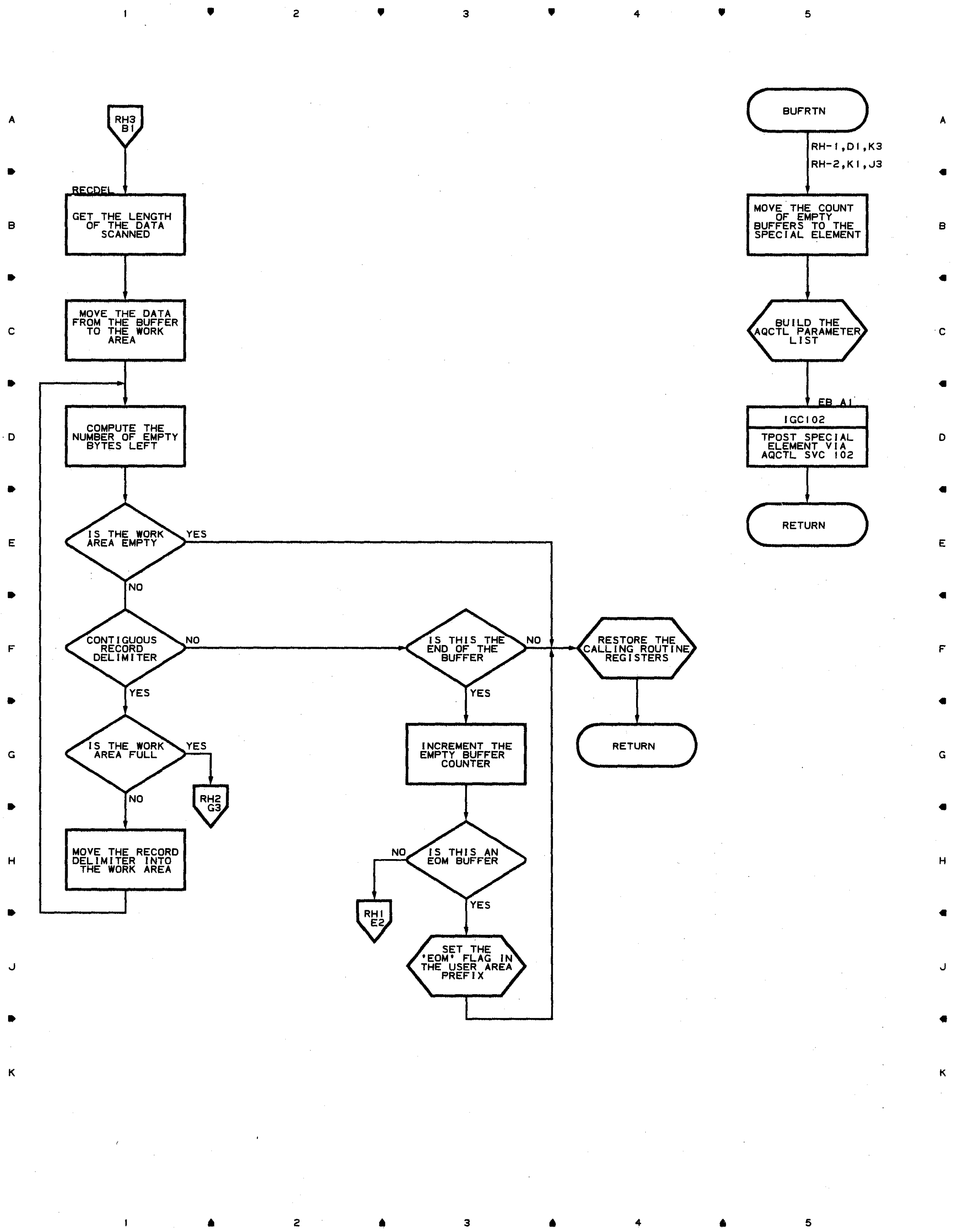

RH3 B1
RECDEL
GET THE LENGTH OF THE DATA SCANNED
MOVE THE DATA FROM THE BUFFER TO THE WORK AREA
COMPUTE THE NUMBER OF EMPTY BYTES LEFT
IS THE WORK AREA EMPTY
YES
NO
CONTIGUOUS RECORD DELIMITER
NO
YES
IS THE WORK AREA FULL
YES
RH2 G3
NO
MOVE THE RECORD DELIMITER INTO THE WORK AREA
IS THIS THE END OF THE BUFFER
NO
YES
INCREMENT THE EMPTY BUFFER COUNTER
IS THIS AN EOM BUFFER
NO
RH1 E2
YES
SET THE 'EOM' FLAG IN THE USER AREA PREFIX
RESTORE THE CALLING ROUTINE REGISTERS
RETURN
BUFRTN
RH-1,D1,K3
RH-2,K1,J3
MOVE THE COUNT OF EMPTY BUFFERS TO THE SPECIAL ELEMENT
BUILD THE AQCTL PARAMETER LIST
EB A1
IGC102
TPOST SPECIAL ELEMENT VIA AQCTL SVC 102
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart RI PUT/WRITE ROUTINE

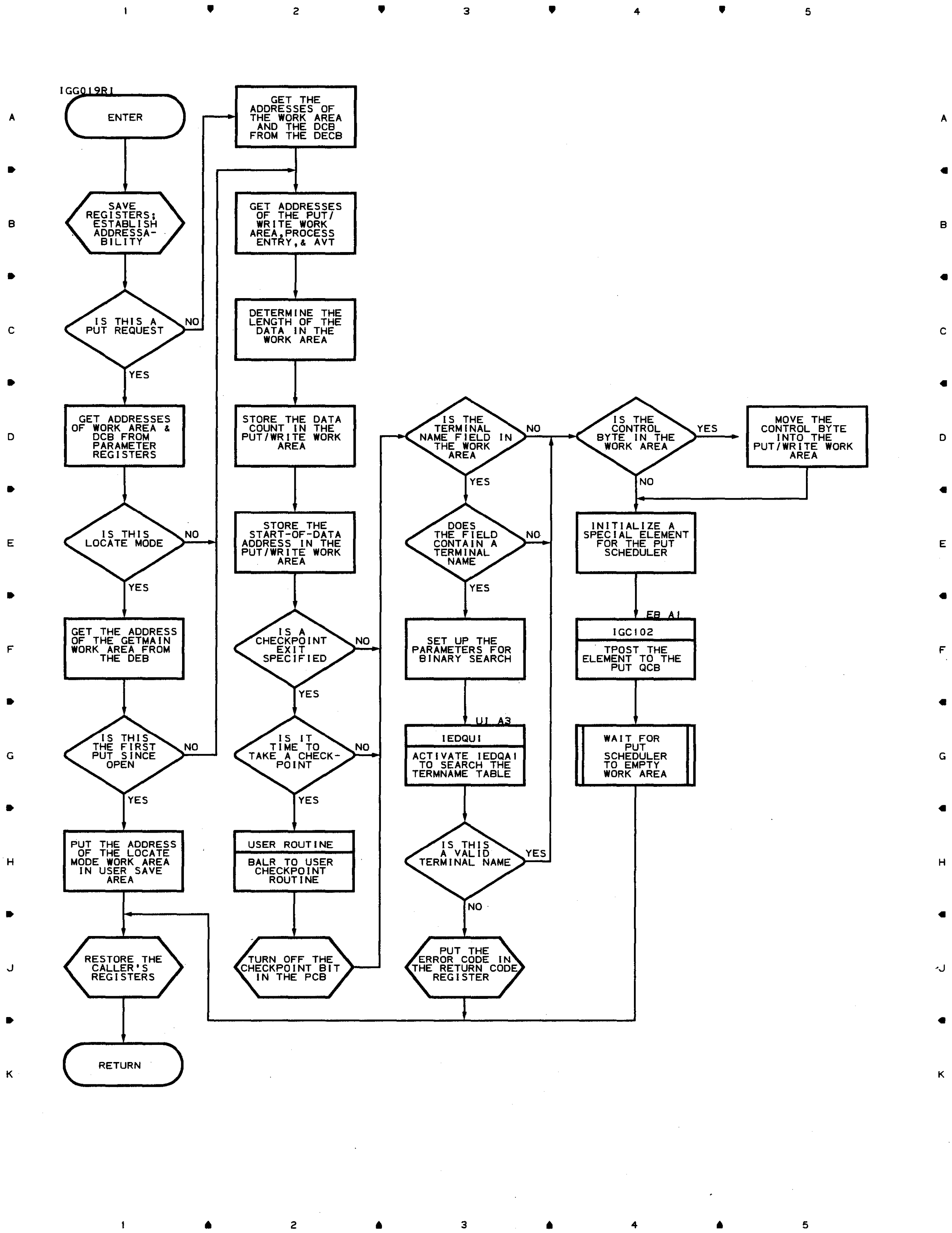

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG019RI
ENTER
SAVE REGISTERS; ESTABLISH ADDRESSA-BILITY
IS THIS A PUT REQUEST
NO
YES
GET ADDRESSES OF WORK AREA & DCB FROM PARAMETER REGISTERS
IS THIS LOCATE MODE
NO
YES
GET THE ADDRESS OF THE GETMAIN WORK AREA FROM THE DEB
IS THIS THE FIRST PUT SINCE OPEN
NO
YES
PUT THE ADDRESS OF THE LOCATE MODE WORK AREA IN USER SAVE AREA
RESTORE THE CALLER'S REGISTERS
RETURN
GET THE ADDRESSES OF THE WORK AREA AND THE DCB FROM THE DECB
GET ADDRESSES OF THE PUT/ WRITE WORK AREA,PROCESS ENTRY,& AVT
DETERMINE THE LENGTH OF THE DATA IN THE WORK AREA
STORE THE DATA COUNT IN THE PUT/WRITE WORK AREA
STORE THE START-OF-DATA ADDRESS IN THE PUT/WRITE WORK AREA
IS A CHECKPOINT EXIT SPECIFIED
NO
YES
IS IT TIME TO TAKE A CHECK-POINT
NO
YES
USER ROUTINE
BALR TO USER CHECKPOINT ROUTINE
TURN OFF THE CHECKPOINT BIT IN THE PCB
IS THE TERMINAL NAME FIELD IN THE WORK AREA
NO
YES
DOES THE FIELD CONTAIN A TERMINAL NAME
NO
YES
SET UP THE PARAMETERS FOR BINARY SEARCH
UI A3
IEDQUI
ACTIVATE IEDQAI TO SEARCH THE TERMNAME TABLE
IS THIS A VALID TERMINAL NAME
YES
NO
PUT THE ERROR CODE IN THE RETURN CODE REGISTER
IS THE CONTROL BYTE IN THE WORK AREA
YES
NO
MOVE THE CONTROL BYTE INTO THE PUT/WRITE WORK AREA
INITIALIZE A SPECIAL ELEMENT FOR THE PUT SCHEDULER
EB A1
IGC102
TPOST THE ELEMENT TO THE PUT QCB
WAIT FOR PUT SCHEDULER TO EMPTY WORK AREA

Chart RJ PUT COMPATIBLE ROUTINE

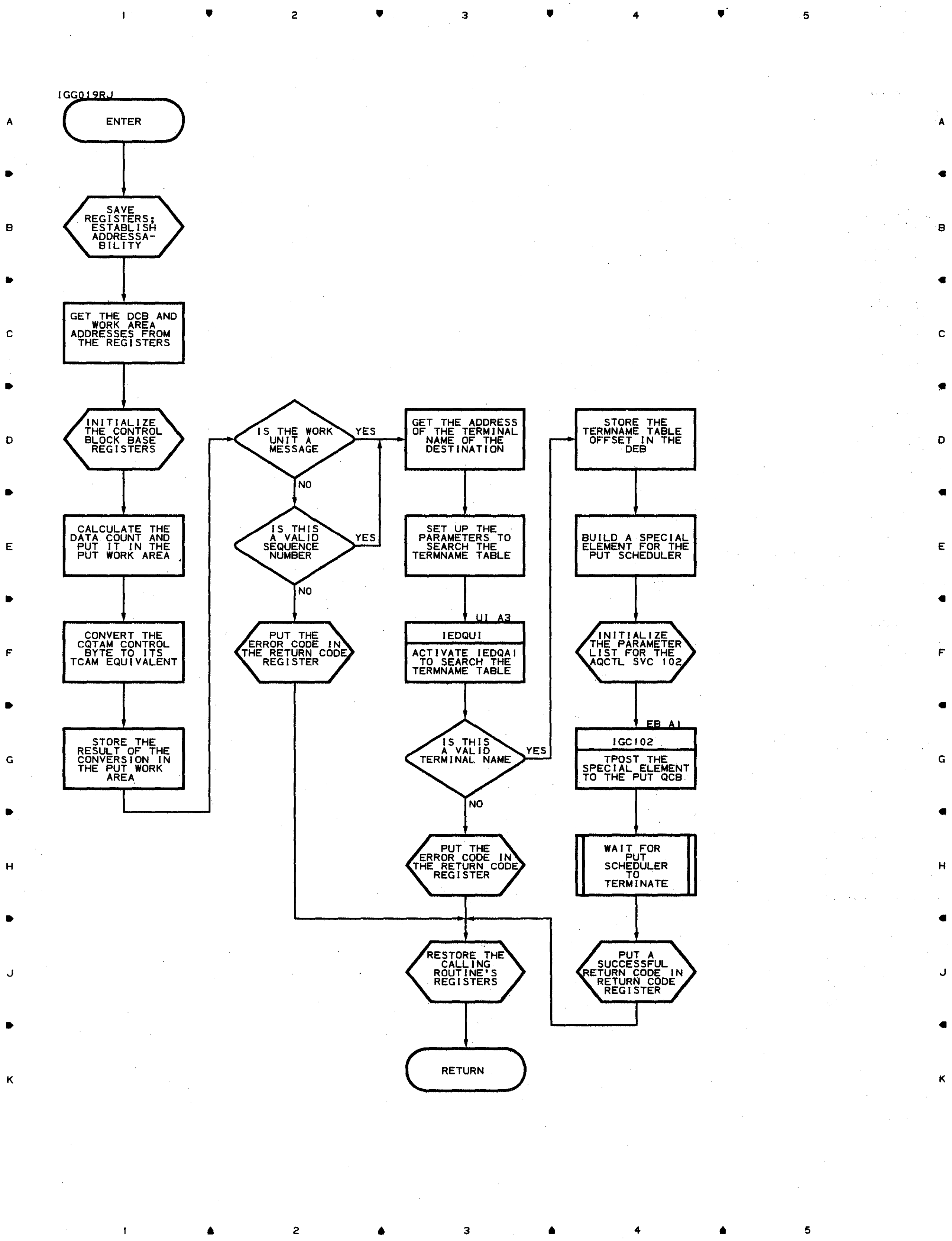
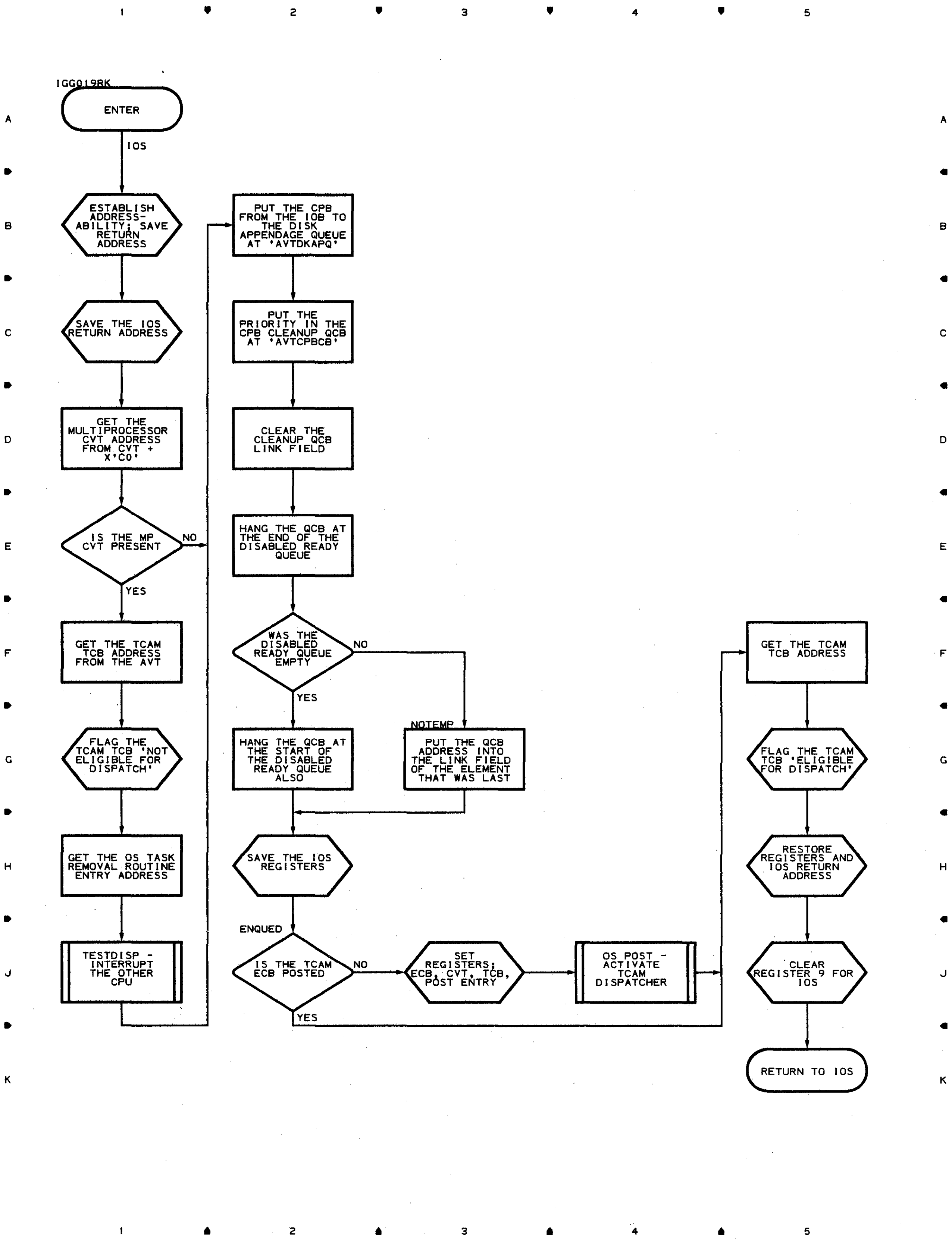

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG019RJ
ENTER
SAVE REGISTERS; ESTABLISH ADDRESSA-BILITY
GET THE DCB AND WORK AREA ADDRESSES FROM THE REGISTERS
INITIALIZE THE CONTROL BLOCK BASE REGISTERS
CALCULATE THE DATA COUNT AND PUT IT IN THE PUT WORK AREA
CONVERT THE CQTAM CONTROL BYTE TO ITS TCAM EQUIVALENT
STORE THE RESULT OF THE CONVERSION IN THE PUT WORK AREA
IS THE WORK UNIT A MESSAGE
YES
NO
IS THIS A VALID SEQUENCE NUMBER
YES
NO
PUT THE ERROR CODE IN THE RETURN CODE REGISTER
GET THE ADDRESS OF THE TERMINAL NAME OF THE DESTINATION
SET UP THE PARAMETERS TO SEARCH THE TERMNAME TABLE
UI A3
IEDQUI
ACTIVATE IEDQAI TO SEARCH THE TERMNAME TABLE
IS THIS A VALID TERMINAL NAME
YES
NO
PUT THE ERROR CODE IN THE RETURN CODE REGISTER
RESTORE THE CALLING ROUTINE'S REGISTERS
RETURN
STORE THE TERMNAME TABLE OFFSET IN THE DEB
BUILD A SPECIAL ELEMENT FOR THE PUT SCHEDULER
INITIALIZE THE PARAMETER LIST FOR THE AQCTL SVC 102
EB A1
IGC102
TPOST THE SPECIAL ELEMENT TO THE PUT QCB
WAIT FOR PUT SCHEDULER TO TERMINATE
PUT A SUCCESSFUL RETURN CODE IN RETURN CODE REGISTER

Chart RK DISK END APPENDAGE FOR A SINGLE CPB



IGG019RK
ENTER
IOS
ESTABLISH ADDRESS-ABILITY; SAVE RETURN ADDRESS
SAVE THE IOS RETURN ADDRESS
GET THE MULTIPROCESSOR CVT ADDRESS FROM CVT + X'C0'
IS THE MP CVT PRESENT
NO
YES
GET THE TCAM TCB ADDRESS FROM THE AVT
FLAG THE TCAM TCB 'NOT ELIGIBLE FOR DISPATCH'
GET THE OS TASK REMOVAL ROUTINE ENTRY ADDRESS
TESTDISP - INTERRUPT THE OTHER CPU
PUT THE CPB FROM THE IOB TO THE DISK APPENDAGE QUEUE AT 'AVTDKAPQ'
PUT THE PRIORITY IN THE CPB CLEANUP QCB AT 'AVTCPBCB'
CLEAR THE CLEANUP QCB LINK FIELD
HANG THE QCB AT THE END OF THE DISABLED READY QUEUE
WAS THE DISABLED READY QUEUE EMPTY
NO
YES
HANG THE QCB AT THE START OF THE DISABLED READY QUEUE ALSO
NOTEMP
PUT THE QCB ADDRESS INTO THE LINK FIELD OF THE ELEMENT THAT WAS LAST
SAVE THE IOS REGISTERS
ENQUED
IS THE TCAM ECB POSTED
NO
YES
SET REGISTERS; ECB, CVT, TCB, POST ENTRY
OS POST - ACTIVATE TCAM DISPATCHER
GET THE TCAM TCB ADDRESS
FLAG THE TCAM TCB 'ELIGIBLE FOR DISPATCH'
RESTORE REGISTERS AND IOS RETURN ADDRESS
CLEAR REGISTER 9 FOR IOS
RETURN TO IOS
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart RL CHECK ROUTINE

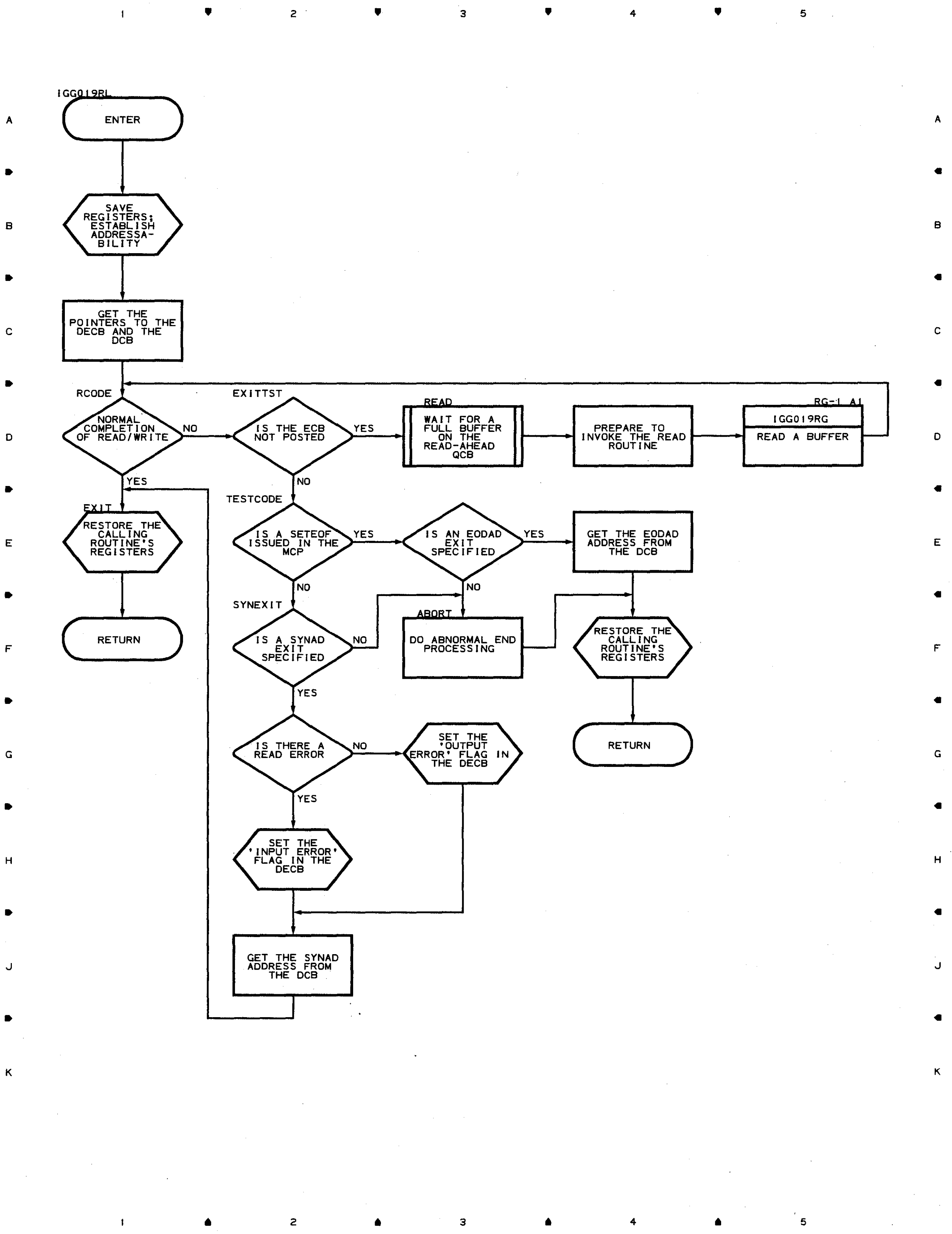

IGG019RL
ENTER
SAVE REGISTERS; ESTABLISH ADDRESSA-BILITY
GET THE POINTERS TO THE DECB AND THE DCB
RCODE
NORMAL COMPLETION OF READ/WRITE
NO
YES
EXITTST
IS THE ECB NOT POSTED
YES
NO
READ
WAIT FOR A FULL BUFFER ON THE READ-AHEAD QCB
PREPARE TO INVOKE THE READ ROUTINE
RG-1 A1
IGG019RG
READ A BUFFER
EXIT
RESTORE THE CALLING ROUTINE'S REGISTERS
RETURN
TESTCODE
IS A SETEOF ISSUED IN THE MCP
YES
NO
IS AN EODAD EXIT SPECIFIED
YES
NO
GET THE EODAD ADDRESS FROM THE DCB
SYNEXIT
IS A SYNAD EXIT SPECIFIED
NO
YES
ABORT
DO ABNORMAL END PROCESSING
RESTORE THE CALLING ROUTINE'S REGISTERS
RETURN
IS THERE A READ ERROR
NO
YES
SET THE 'OUTPUT ERROR' FLAG IN THE DECB
SET THE 'INPUT ERROR' FLAG IN THE DECB
GET THE SYNAD ADDRESS FROM THE DCB
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart RM POINT ROUTINE

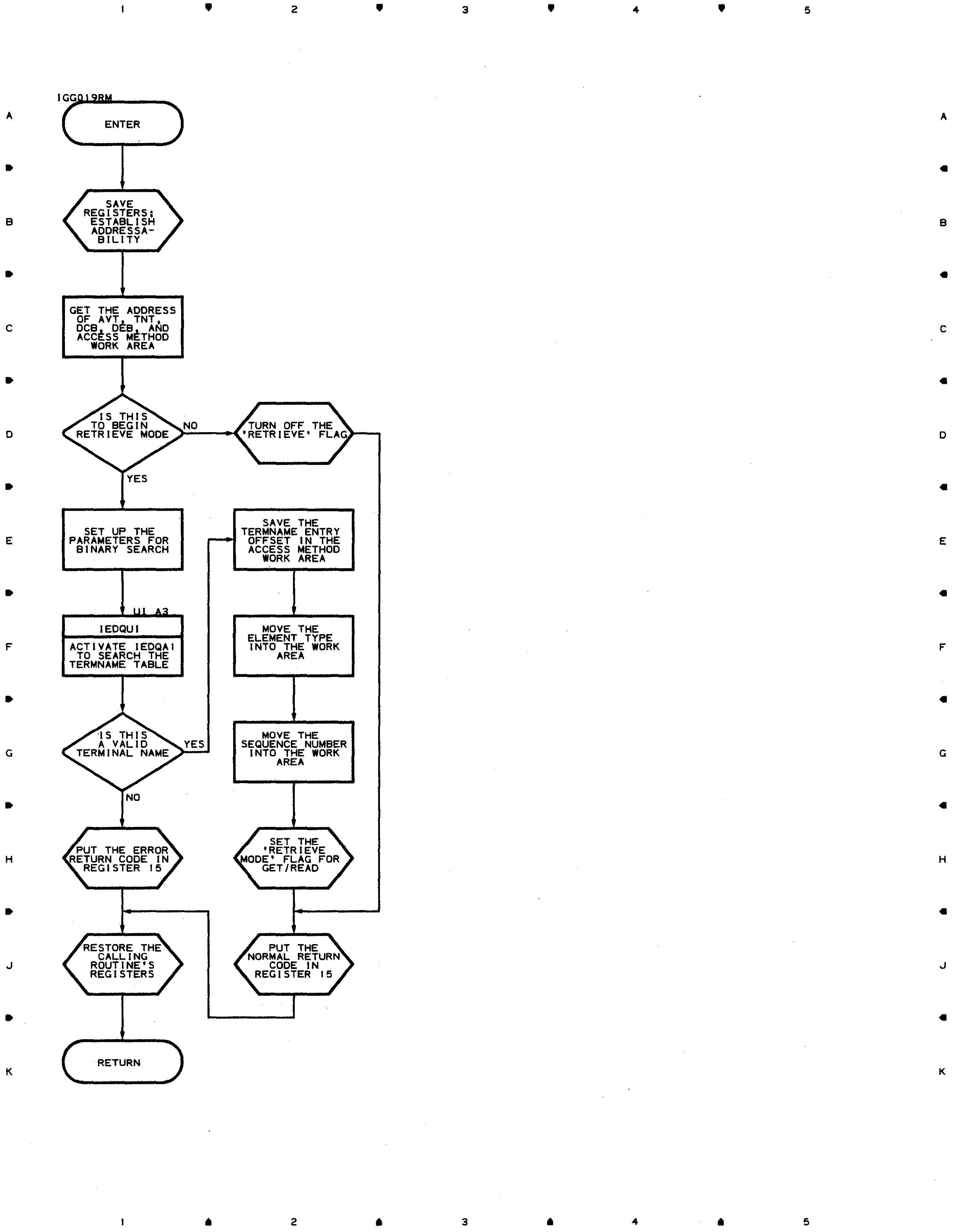

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG019RM
ENTER
SAVE REGISTERS; ESTABLISH ADDRESSA- BILITY
GET THE ADDRESS OF AVT, TNT, DCB, DEB, AND ACCESS METHOD WORK AREA
IS THIS TO BEGIN RETRIEVE MODE
NO
TURN OFF THE 'RETRIEVE' FLAG
YES
SET UP THE PARAMETERS FOR BINARY SEARCH
SAVE THE TERMNAME ENTRY OFFSET IN THE ACCESS METHOD WORK AREA
UI A3
IEDQUI
ACTIVATE IEDQAI TO SEARCH THE TERMNAME TABLE
MOVE THE ELEMENT TYPE INTO THE WORK AREA
IS THIS A VALID TERMINAL NAME
YES
MOVE THE SEQUENCE NUMBER INTO THE WORK AREA
NO
PUT THE ERROR RETURN CODE IN REGISTER 15
SET THE 'RETRIEVE MODE' FLAG FOR GET/READ
RESTORE THE CALLING ROUTINE'S REGISTERS
PUT THE NORMAL RETURN CODE IN REGISTER 15
RETURN

Chart RN-1 PCI APPENDAGE

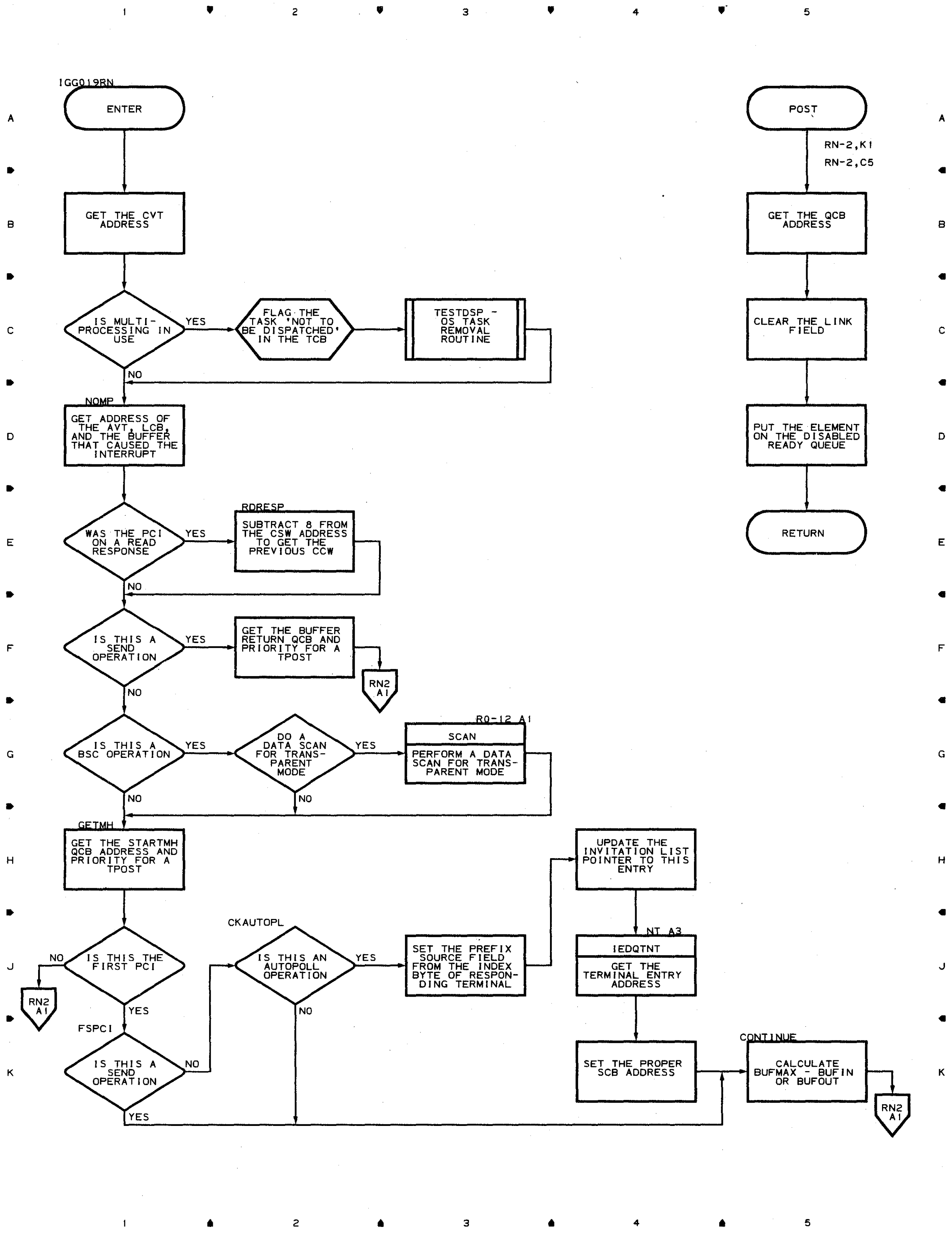
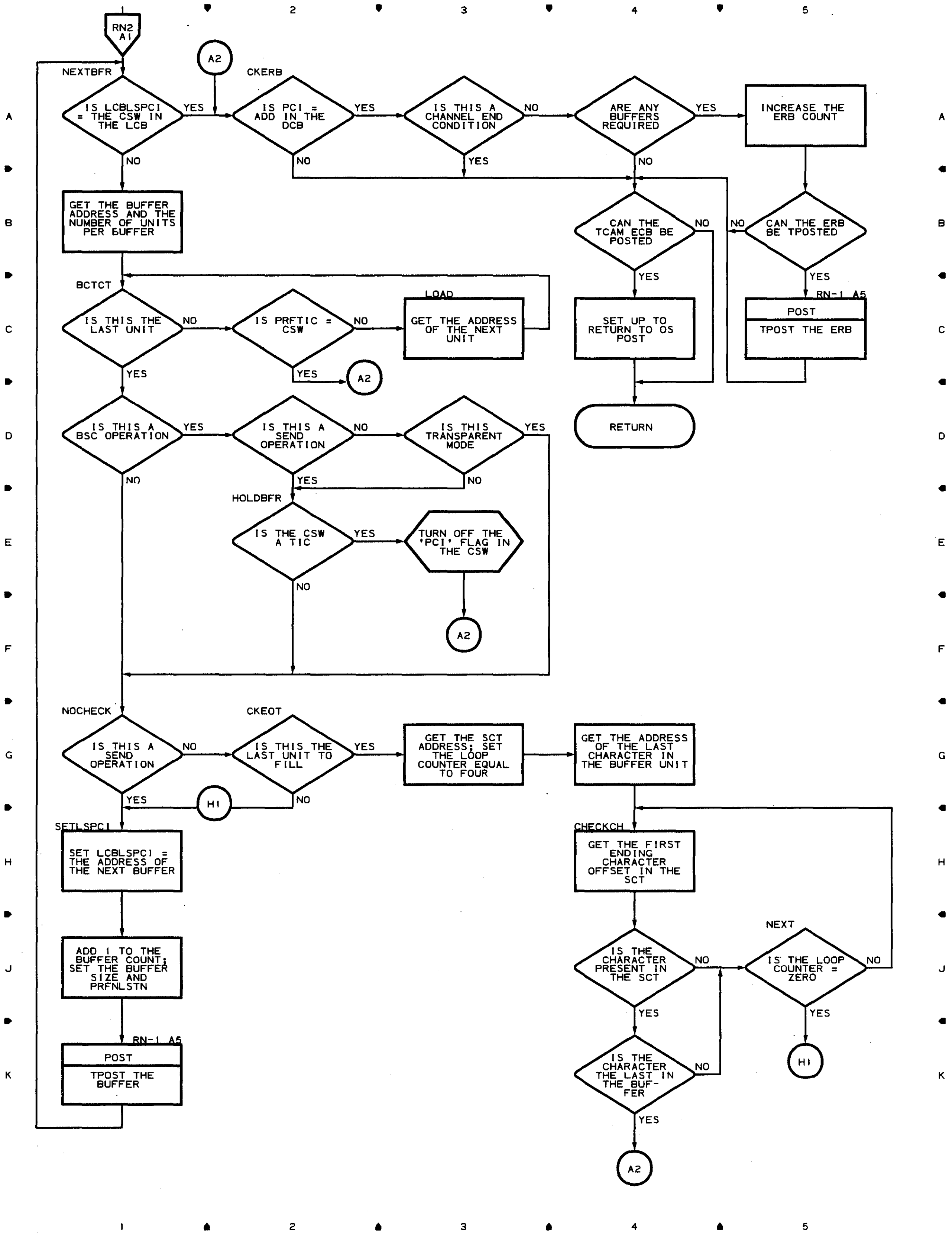

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG019RN
ENTER
GET THE CVT ADDRESS
IS MULTI-PROCESSING IN USE
YES
FLAG THE TASK 'NOT TO BE DISPATCHED' IN THE TCB
TESTDSP - OS TASK REMOVAL ROUTINE
NO
NOMP
GET ADDRESS OF THE AVT, LCB, AND THE BUFFER THAT CAUSED THE INTERRUPT
WAS THE PCI ON A READ RESPONSE
YES
RDRESP
SUBTRACT 8 FROM THE CSW ADDRESS TO GET THE PREVIOUS CCW
NO
IS THIS A SEND OPERATION
YES
GET THE BUFFER RETURN QCB AND PRIORITY FOR A TPOST
RN2 A1
NO
IS THIS A BSC OPERATION
YES
DO A DATA SCAN FOR TRANS-PARENT MODE
YES
RO-12 A1
SCAN
PERFORM A DATA SCAN FOR TRANS-PARENT MODE
NO
NO
GETMH
GET THE STARTMH QCB ADDRESS AND PRIORITY FOR A TPOST
IS THIS THE FIRST PCI
NO
RN2 A1
YES
FSPCI
IS THIS A SEND OPERATION
NO
YES
CKAUTOPL
IS THIS AN AUTOPOLL OPERATION
YES
NO
SET THE PREFIX SOURCE FIELD FROM THE INDEX BYTE OF RESPON-DING TERMINAL
UPDATE THE INVITATION LIST POINTER TO THIS ENTRY
NT A3
IEDQTNT
GET THE TERMINAL ENTRY ADDRESS
SET THE PROPER SCB ADDRESS
CONTINUE
CALCULATE BUFMAX - BUFIN OR BUFOUT
RN2 A1
POST
RN-2,K1
RN-2,C5
GET THE QCB ADDRESS
CLEAR THE LINK FIELD
PUT THE ELEMENT ON THE DISABLED READY QUEUE
RETURN

Chart RN-2 PCI APPENDAGE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
RN2 A1
A2
NEXTBFR
IS LCBLSPCI = THE CSW IN THE LCB
YES
NO
CKERB
IS PCI = ADD IN THE DCB
YES
NO
IS THIS A CHANNEL END CONDITION
NO
YES
ARE ANY BUFFERS REQUIRED
YES
NO
INCREASE THE ERB COUNT
GET THE BUFFER ADDRESS AND THE NUMBER OF UNITS PER BUFFER
CAN THE TCAM ECB BE POSTED
NO
YES
CAN THE ERB BE TPOSTED
NO
YES
RN-1 A5
POST
TPOST THE ERB
BCTCT
IS THIS THE LAST UNIT
NO
YES
IS PRFTIC = CSW
NO
YES
A2
LOAD
GET THE ADDRESS OF THE NEXT UNIT
SET UP TO RETURN TO OS POST
RETURN
IS THIS A BSC OPERATION
YES
NO
IS THIS A SEND OPERATION
NO
YES
IS THIS TRANSPARENT MODE
YES
NO
HOLDBFR
IS THE CSW A TIC
YES
NO
TURN OFF THE 'PCI' FLAG IN THE CSW
A2
NOCHECK
IS THIS A SEND OPERATION
NO
YES
CKEOT
IS THIS THE LAST UNIT TO FILL
YES
NO
H1
GET THE SCT ADDRESS; SET THE LOOP COUNTER EQUAL TO FOUR
GET THE ADDRESS OF THE LAST CHARACTER IN THE BUFFER UNIT
SETLSPCI
SET LCBLSPCI = THE ADDRESS OF THE NEXT BUFFER
CHECKCH
GET THE FIRST ENDING CHARACTER OFFSET IN THE SCT
ADD 1 TO THE BUFFER COUNT; SET THE BUFFER SIZE AND PRFNLSTN
NEXT
IS THE CHARACTER PRESENT IN THE SCT
NO
YES
IS THE LOOP COUNTER = ZERO
NO
YES
RN-1 A5
POST
TPOST THE BUFFER
IS THE CHARACTER THE LAST IN THE BUF- FER
NO
YES
H1
A2

Chart RO-1 TCAM DISPATCHER WITH SUBTASK TRACE

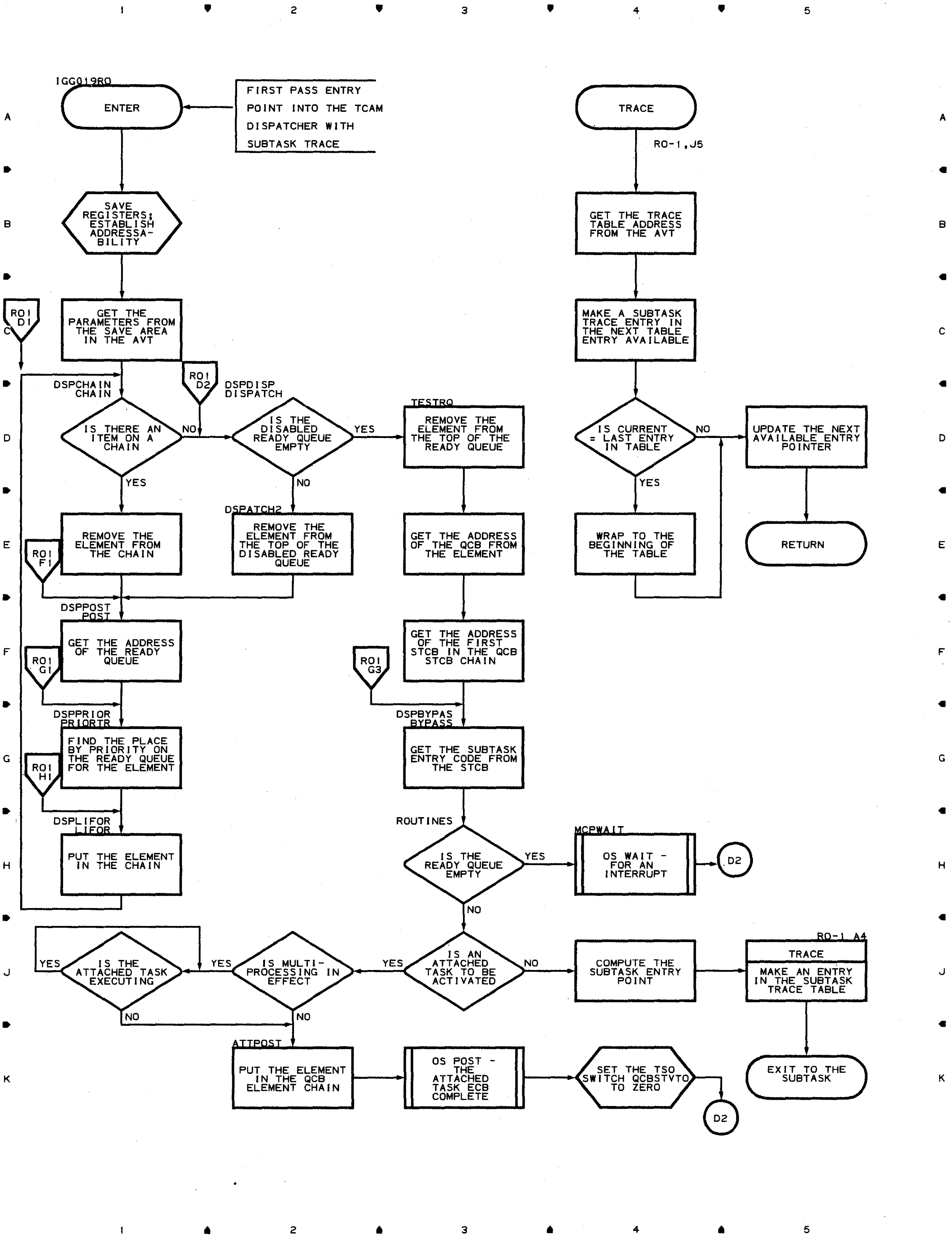

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG019RO
ENTER
FIRST PASS ENTRY POINT INTO THE TCAM DISPATCHER WITH SUBTASK TRACE
SAVE REGISTERS; ESTABLISH ADDRESSA-BILITY
RO1 D1
GET THE PARAMETERS FROM THE SAVE AREA IN THE AVT
DSPCHAIN CHAIN
IS THERE AN ITEM ON A CHAIN
NO
YES
RO1 D2
DSPDISP DISPATCH
IS THE DISABLED READY QUEUE EMPTY
YES
NO
TESTRQ
REMOVE THE ELEMENT FROM THE TOP OF THE READY QUEUE
REMOVE THE ELEMENT FROM THE CHAIN
RO1 F1
DSPATCH2
REMOVE THE ELEMENT FROM THE TOP OF THE DISABLED READY QUEUE
GET THE ADDRESS OF THE QCB FROM THE ELEMENT
DSPPOST POST
GET THE ADDRESS OF THE READY QUEUE
RO1 G1
GET THE ADDRESS OF THE FIRST STCB IN THE QCB STCB CHAIN
RO1 G3
DSPPRIOR PRIORTR
FIND THE PLACE BY PRIORITY ON THE READY QUEUE FOR THE ELEMENT
RO1 H1
DSPBYPAS BYPASS
GET THE SUBTASK ENTRY CODE FROM THE STCB
DSPLIFOR LIFOR
PUT THE ELEMENT IN THE CHAIN
ROUTINES
IS THE READY QUEUE EMPTY
YES
NO
MCPWAIT
OS WAIT - FOR AN INTERRUPT
D2
TRACE
RO-1,J5
GET THE TRACE TABLE ADDRESS FROM THE AVT
MAKE A SUBTASK TRACE ENTRY IN THE NEXT TABLE ENTRY AVAILABLE
IS CURRENT = LAST ENTRY IN TABLE
NO
YES
UPDATE THE NEXT AVAILABLE ENTRY POINTER
WRAP TO THE BEGINNING OF THE TABLE
RETURN
YES
IS THE ATTACHED TASK EXECUTING
NO
YES
IS MULTI-PROCESSING IN EFFECT
NO
YES
IS AN ATTACHED TASK TO BE ACTIVATED
NO
COMPUTE THE SUBTASK ENTRY POINT
RO-1 A4
TRACE
MAKE AN ENTRY IN THE SUBTASK TRACE TABLE
ATTPOST
PUT THE ELEMENT IN THE QCB ELEMENT CHAIN
OS POST - THE ATTACHED TASK ECB COMPLETE
SET THE TSO SWITCH QCBSTVTO TO ZERO
D2
EXIT TO THE SUBTASK

Chart RO-2 TCAM DISPATCHER WITH SUBTASK TRACE

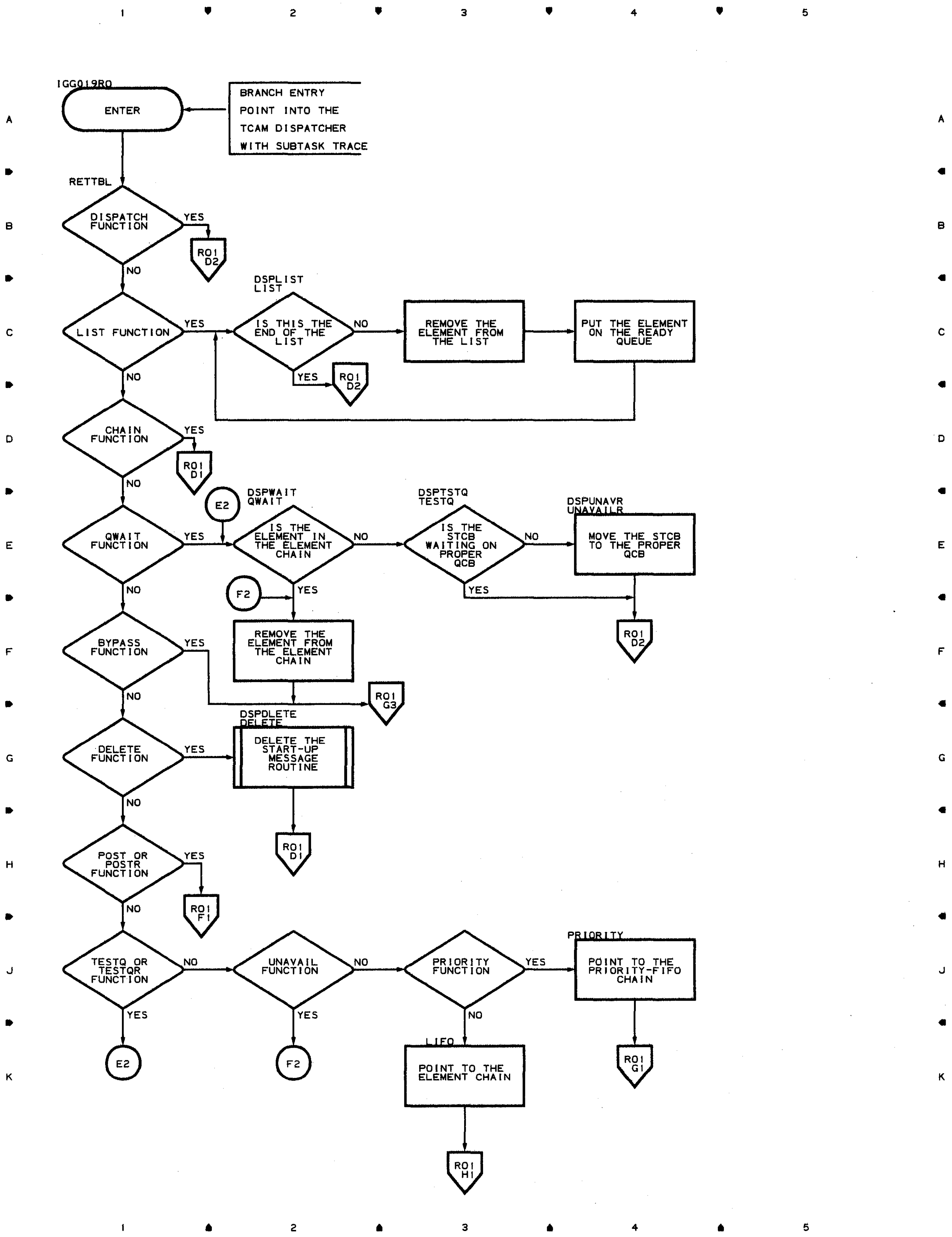


Chart RP-1 REUSABILITY-COPY SUBTASK

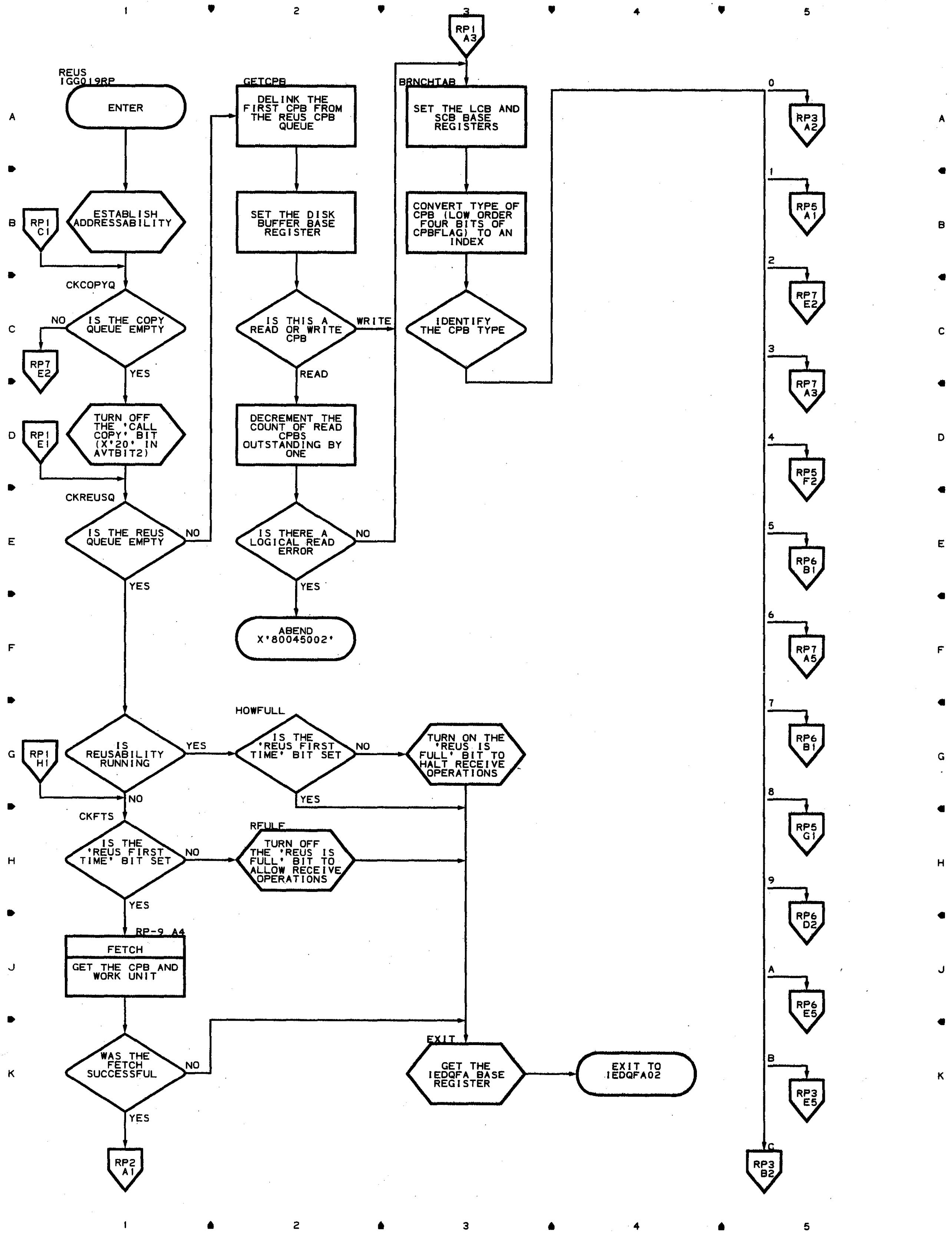
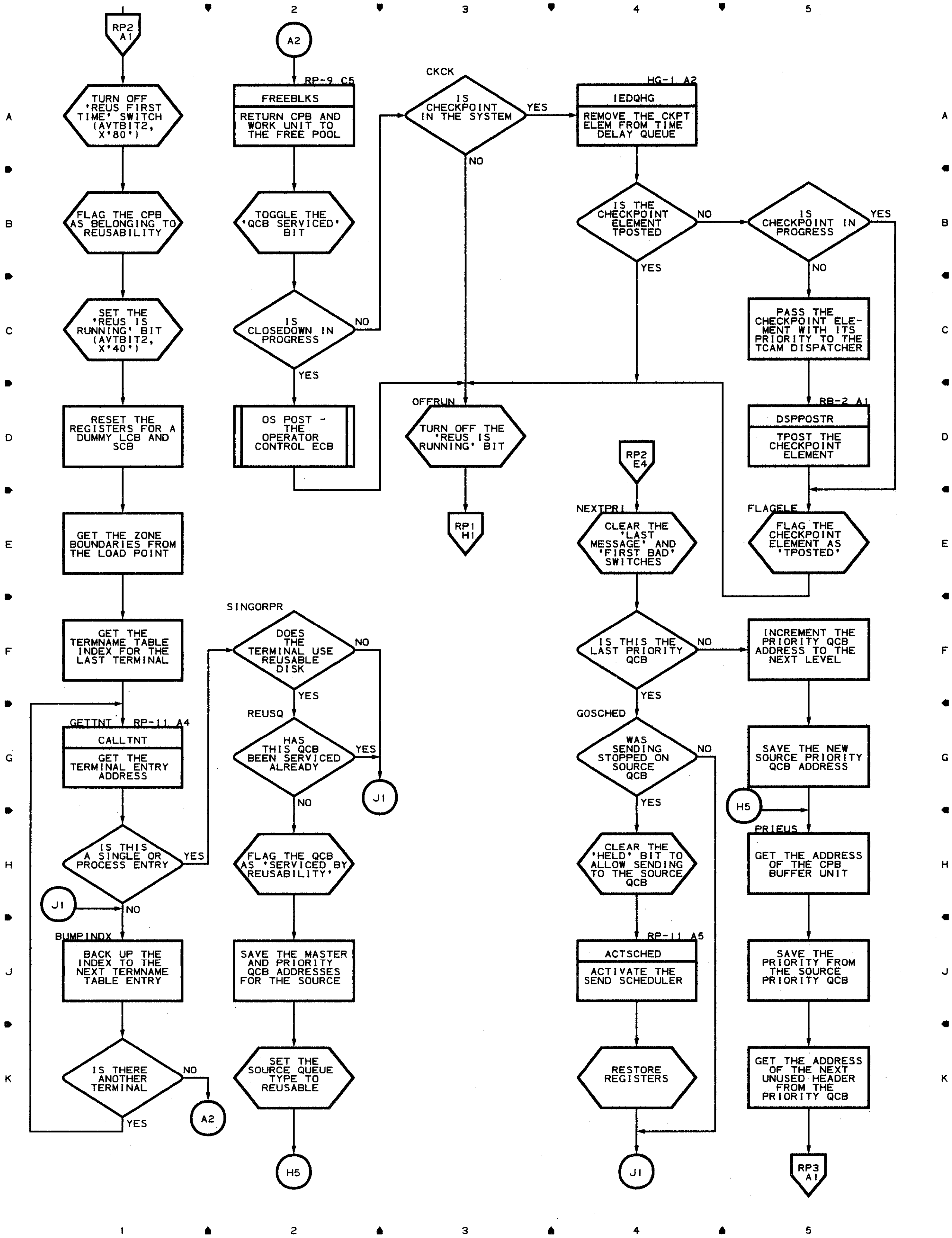
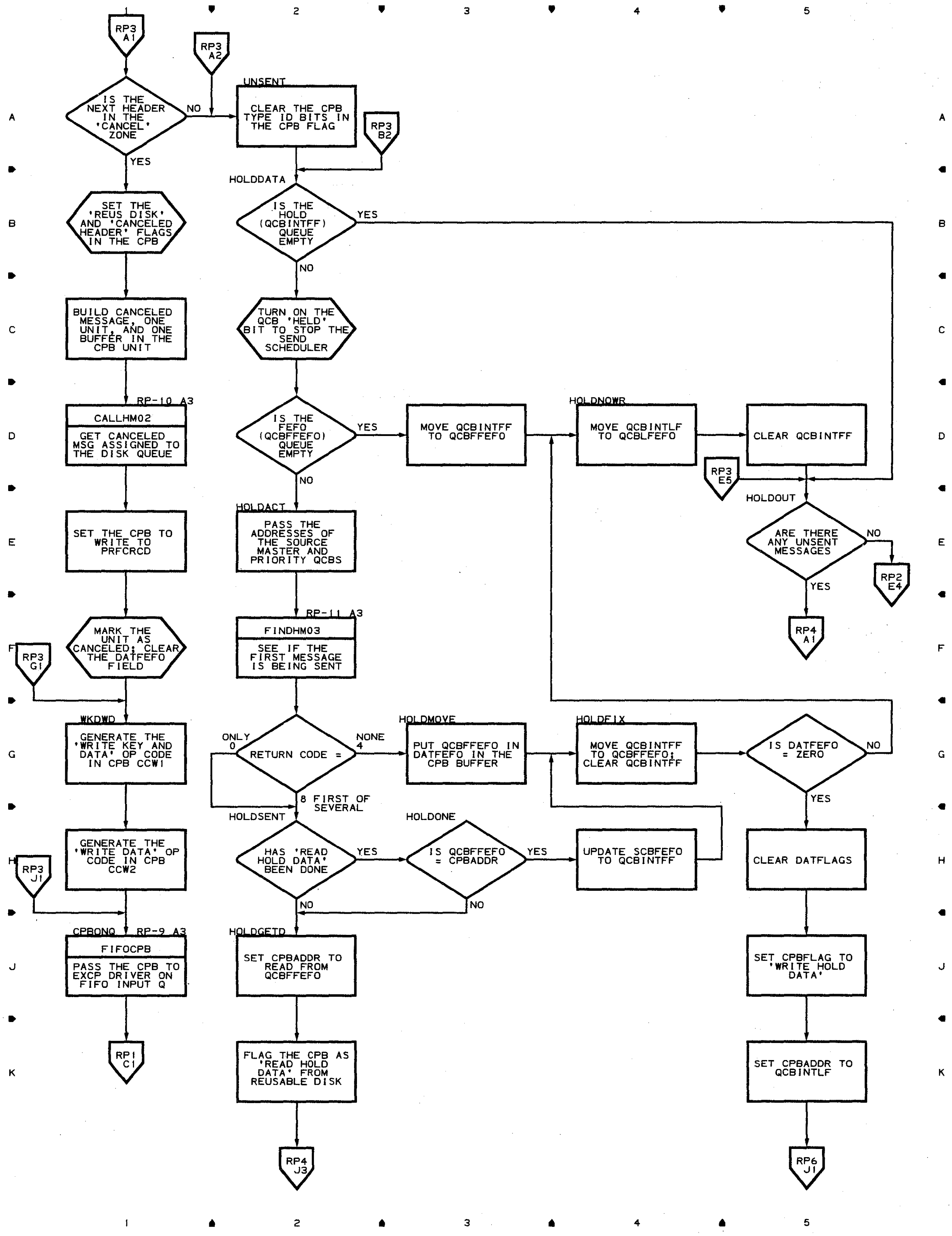

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
REUS
IGG019RP
ENTER
ESTABLISH ADDRESSABILITY
RP1 C1
CKCOPYQ
IS THE COPY QUEUE EMPTY
NO
RP7 E2
YES
TURN OFF THE 'CALL COPY' BIT (X'20' IN AVTBIT2)
RP1 E1
CKREUSQ
IS THE REUS QUEUE EMPTY
NO
YES
IS REUSABILITY RUNNING
YES
NO
RP1 H1
CKFTS
IS THE 'REUS FIRST TIME' BIT SET
NO
YES
RP-9 A4
FETCH
GET THE CPB AND WORK UNIT
WAS THE FETCH SUCCESSFUL
NO
YES
RP2 A1
GETCPB
DELINK THE FIRST CPB FROM THE REUS CPB QUEUE
SET THE DISK BUFFER BASE REGISTER
IS THIS A READ OR WRITE CPB
WRITE
READ
DECREMENT THE COUNT OF READ CPBS OUTSTANDING BY ONE
IS THERE A LOGICAL READ ERROR
NO
YES
ABEND X'80045002'
HOWFULL
IS THE 'REUS FIRST TIME' BIT SET
NO
YES
RFULF
TURN OFF THE 'REUS IS FULL' BIT TO ALLOW RECEIVE OPERATIONS
RP1 A3
BRNCHTAB
SET THE LCB AND SCB BASE REGISTERS
CONVERT TYPE OF CPB (LOW ORDER FOUR BITS OF CPBFLAG) TO AN INDEX
IDENTIFY THE CPB TYPE
TURN ON THE 'REUS IS FULL' BIT TO HALT RECEIVE OPERATIONS
EXIT
GET THE IEDQFA BASE REGISTER
EXIT TO IEDQFA02
0
RP3 A2
1
RP5 A1
2
RP7 E2
3
RP7 A3
4
RP5 F2
5
RP6 B1
6
RP7 A5
7
RP6 B1
8
RP5 G1
9
RP6 D2
A
RP6 E5
B
RP3 E5
C
RP3 B2

Chart RP-2 REUSABILITY-COPY SUBTASK



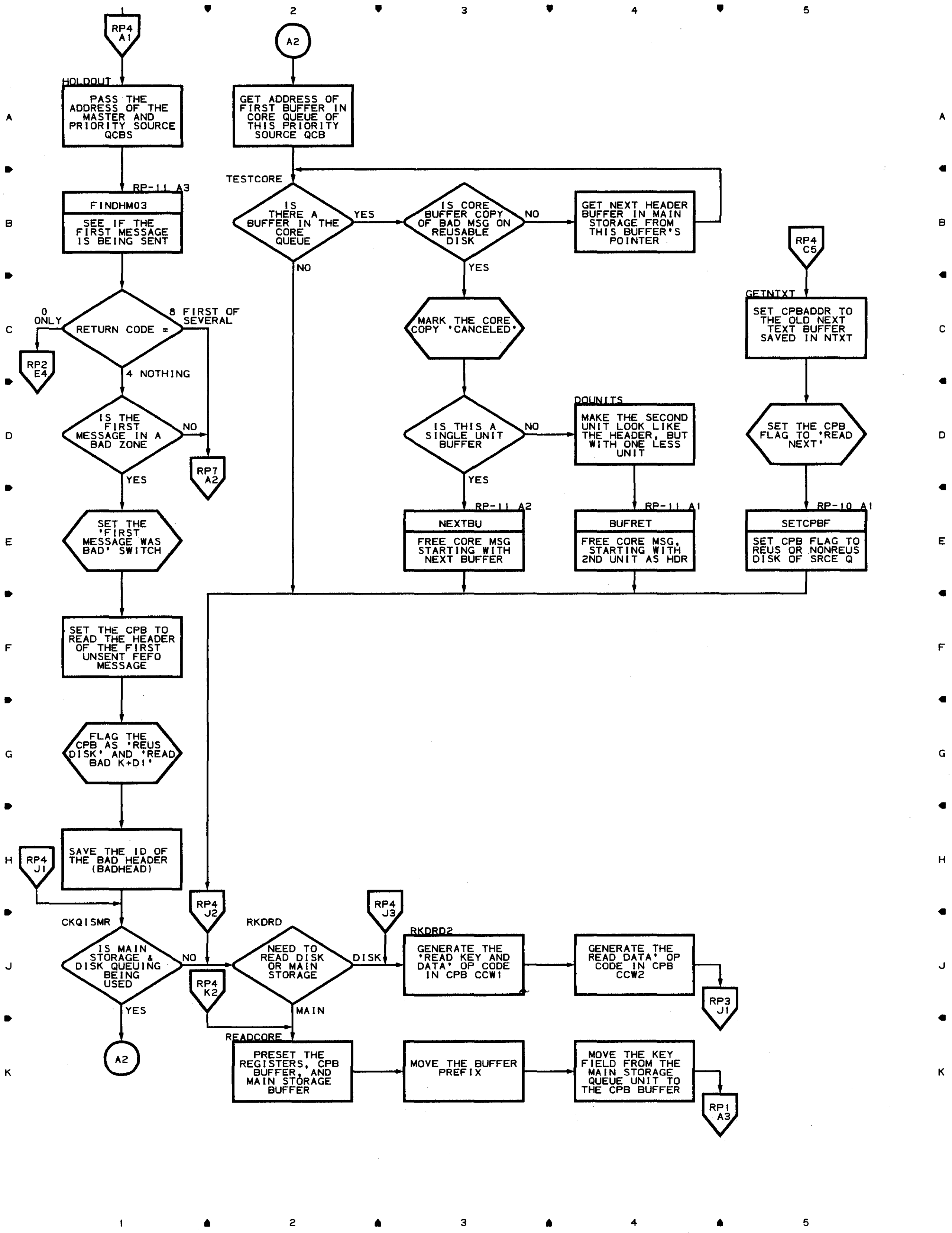
RP2 A1
TURN OFF 'REUS FIRST TIME' SWITCH (AVTBIT2, X'80')
FLAG THE CPB AS BELONGING TO REUSABILITY
SET THE 'REUS IS RUNNING' BIT (AVTBIT2, X'40')
RESET THE REGISTERS FOR A DUMMY LCB AND SCB
GET THE ZONE BOUNDARIES FROM THE LOAD POINT
GET THE TERMNAME TABLE INDEX FOR THE LAST TERMINAL
GETTNT RP-11 A4
CALLTNT
GET THE TERMINAL ENTRY ADDRESS
IS THIS A SINGLE OR PROCESS ENTRY
YES
NO
J1
BUMPINDX
BACK UP THE INDEX TO THE NEXT TERMNAME TABLE ENTRY
IS THERE ANOTHER TERMINAL
NO
YES
A2
A2
RP-9 C5
FREEBLKS
RETURN CPB AND WORK UNIT TO THE FREE POOL
TOGGLE THE 'QCB SERVICED' BIT
IS CLOSEDOWN IN PROGRESS
NO
YES
OS POST - THE OPERATOR CONTROL ECB
SINGORPR
DOES THE TERMINAL USE REUSABLE DISK
NO
YES
REUSQ
HAS THIS QCB BEEN SERVICED ALREADY
YES
NO
J1
FLAG THE QCB AS 'SERVICED BY REUSABILITY'
SAVE THE MASTER AND PRIORITY QCB ADDRESSES FOR THE SOURCE
SET THE SOURCE QUEUE TYPE TO REUSABLE
H5
CKCK
IS CHECKPOINT IN THE SYSTEM
YES
NO
OFFRUN
TURN OFF THE 'REUS IS RUNNING' BIT
RP1 H1
HG-1 A2
IEDQHG
REMOVE THE CKPT ELEM FROM TIME DELAY QUEUE
IS THE CHECKPOINT ELEMENT TPOSTED
NO
YES
RP2 E4
NEXTPRI
CLEAR THE 'LAST MESSAGE' AND 'FIRST BAD' SWITCHES
IS THIS THE LAST PRIORITY QCB
NO
YES
GOSCHED
WAS SENDING STOPPED ON SOURCE QCB
NO
YES
CLEAR THE 'HELD' BIT TO ALLOW SENDING TO THE SOURCE QCB
RP-11 A5
ACTSCHED
ACTIVATE THE SEND SCHEDULER
RESTORE REGISTERS
J1
IS CHECKPOINT IN PROGRESS
YES
NO
PASS THE CHECKPOINT ELEMENT WITH ITS PRIORITY TO THE TCAM DISPATCHER
RB-2 A1
DSPPOSTR
TPOST THE CHECKPOINT ELEMENT
FLAGELE
FLAG THE CHECKPOINT ELEMENT AS 'TPOSTED'
INCREMENT THE PRIORITY QCB ADDRESS TO THE NEXT LEVEL
SAVE THE NEW SOURCE PRIORITY QCB ADDRESS
H5
PRIEUS
GET THE ADDRESS OF THE CPB BUFFER UNIT
SAVE THE PRIORITY FROM THE SOURCE PRIORITY QCB
GET THE ADDRESS OF THE NEXT UNUSED HEADER FROM THE PRIORITY QCB
RP3 A1
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart RP-3 REUSABILITY-COPY SUBTASK



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
RP3 A1
RP3 A2
IS THE NEXT HEADER IN THE 'CANCEL' ZONE
NO
YES
UNSENT
CLEAR THE CPB TYPE ID BITS IN THE CPB FLAG
RP3 B2
HOLDDATA
IS THE HOLD (QCBINTFF) QUEUE EMPTY
YES
NO
SET THE 'REUS DISK' AND 'CANCELED HEADER' FLAGS IN THE CPB
BUILD CANCELED MESSAGE, ONE UNIT, AND ONE BUFFER IN THE CPB UNIT
TURN ON THE QCB 'HELD' BIT TO STOP THE SEND SCHEDULER
RP-10 A3
CALLHM02
GET CANCELED MSG ASSIGNED TO THE DISK QUEUE
IS THE FEFO (QCBFFEFO) QUEUE EMPTY
YES
NO
MOVE QCBINTFF TO QCBFFEFO
HOLDNOWR
MOVE QCBINTLF TO QCBLFEFO
CLEAR QCBINTFF
RP3 E5
HOLDOUT
SET THE CPB TO WRITE TO PRFCRCD
HOLDACT
PASS THE ADDRESSES OF THE SOURCE MASTER AND PRIORITY QCBS
ARE THERE ANY UNSENT MESSAGES
NO
YES
RP2 E4
RP-11 A3
FINDHM03
SEE IF THE FIRST MESSAGE IS BEING SENT
RP4 A1
MARK THE UNIT AS CANCELED; CLEAR THE DATFEFO FIELD
RP3 G1
WKDWD
GENERATE THE 'WRITE KEY AND DATA' OP CODE IN CPB CCW1
ONLY 0
RETURN CODE =
NONE 4
8 FIRST OF SEVERAL
HOLDMOVE
PUT QCBFFEFO IN DATFEFO IN THE CPB BUFFER
HOLDFIX
MOVE QCBINTFF TO QCBFFEFO; CLEAR QCBINTFF
IS DATFEFO = ZERO
NO
YES
HOLDSENT
HOLDONE
GENERATE THE 'WRITE DATA' OP CODE IN CPB CCW2
RP3 J1
HAS 'READ HOLD DATA' BEEN DONE
YES
NO
IS QCBFFEFO = CPBADDR
YES
NO
UPDATE SCBFEFO TO QCBINTFF
CLEAR DATFLAGS
CPBONQ
RP-9 A3
FIFOCPB
PASS THE CPB TO EXCP DRIVER ON FIFO INPUT Q
HOLDGETD
SET CPBADDR TO READ FROM QCBFFEFO
SET CPBFLAG TO 'WRITE HOLD DATA'
RP1 C1
FLAG THE CPB AS 'READ HOLD DATA' FROM REUSABLE DISK
SET CPBADDR TO QCBINTLF
RP4 J3
RP6 J1

Chart RP-4 REUSABILITY-COPY SUBTASK



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
RP4 A1
HOLDOUT
PASS THE ADDRESS OF THE MASTER AND PRIORITY SOURCE QCBS
RP-11 A3
FINDHM03
SEE IF THE FIRST MESSAGE IS BEING SENT
RETURN CODE =
0 ONLY
8 FIRST OF SEVERAL
4 NOTHING
RP2 E4
IS THE FIRST MESSAGE IN A BAD ZONE
NO
YES
RP7 A2
SET THE 'FIRST MESSAGE WAS BAD' SWITCH
SET THE CPB TO READ THE HEADER OF THE FIRST UNSENT FEFO MESSAGE
FLAG THE CPB AS 'REUS DISK' AND 'READ BAD K+DI'
RP4 J1
SAVE THE ID OF THE BAD HEADER (BADHEAD)
CKQISMR
IS MAIN STORAGE & DISK QUEUING BEING USED
NO
YES
A2
A2
GET ADDRESS OF FIRST BUFFER IN CORE QUEUE OF THIS PRIORITY SOURCE QCB
TESTCORE
IS THERE A BUFFER IN THE CORE QUEUE
YES
NO
IS CORE BUFFER COPY OF BAD MSG ON REUSABLE DISK
NO
YES
GET NEXT HEADER BUFFER IN MAIN STORAGE FROM THIS BUFFER'S POINTER
MARK THE CORE COPY 'CANCELED'
IS THIS A SINGLE UNIT BUFFER
NO
YES
DOUNITS
MAKE THE SECOND UNIT LOOK LIKE THE HEADER, BUT WITH ONE LESS UNIT
RP-11 A2
NEXTBU
FREE CORE MSG STARTING WITH NEXT BUFFER
RP-11 A1
BUFRET
FREE CORE MSG, STARTING WITH 2ND UNIT AS HDR
RP4 C5
GETNTXT
SET CPBADDR TO THE OLD NEXT TEXT BUFFER SAVED IN NTXT
SET THE CPB FLAG TO 'READ NEXT'
RP-10 A1
SETCPBF
SET CPB FLAG TO REUS OR NONREUS DISK OF SRCE Q
RP4 J2
RKDRD
NEED TO READ DISK OR MAIN STORAGE
DISK
MAIN
RP4 K2
RP4 J3
RKDRD2
GENERATE THE 'READ KEY AND DATA' OP CODE IN CPB CCW1
GENERATE THE READ DATA' OP CODE IN CPB CCW2
RP3 J1
READCORE
PRESET THE REGISTERS, CPB BUFFER, AND MAIN STORAGE BUFFER
MOVE THE BUFFER PREFIX
MOVE THE KEY FIELD FROM THE MAIN STORAGE QUEUE UNIT TO THE CPB BUFFER
RP1 A3

Chart RP-5 REUSABILITY-COPY SUBTASK

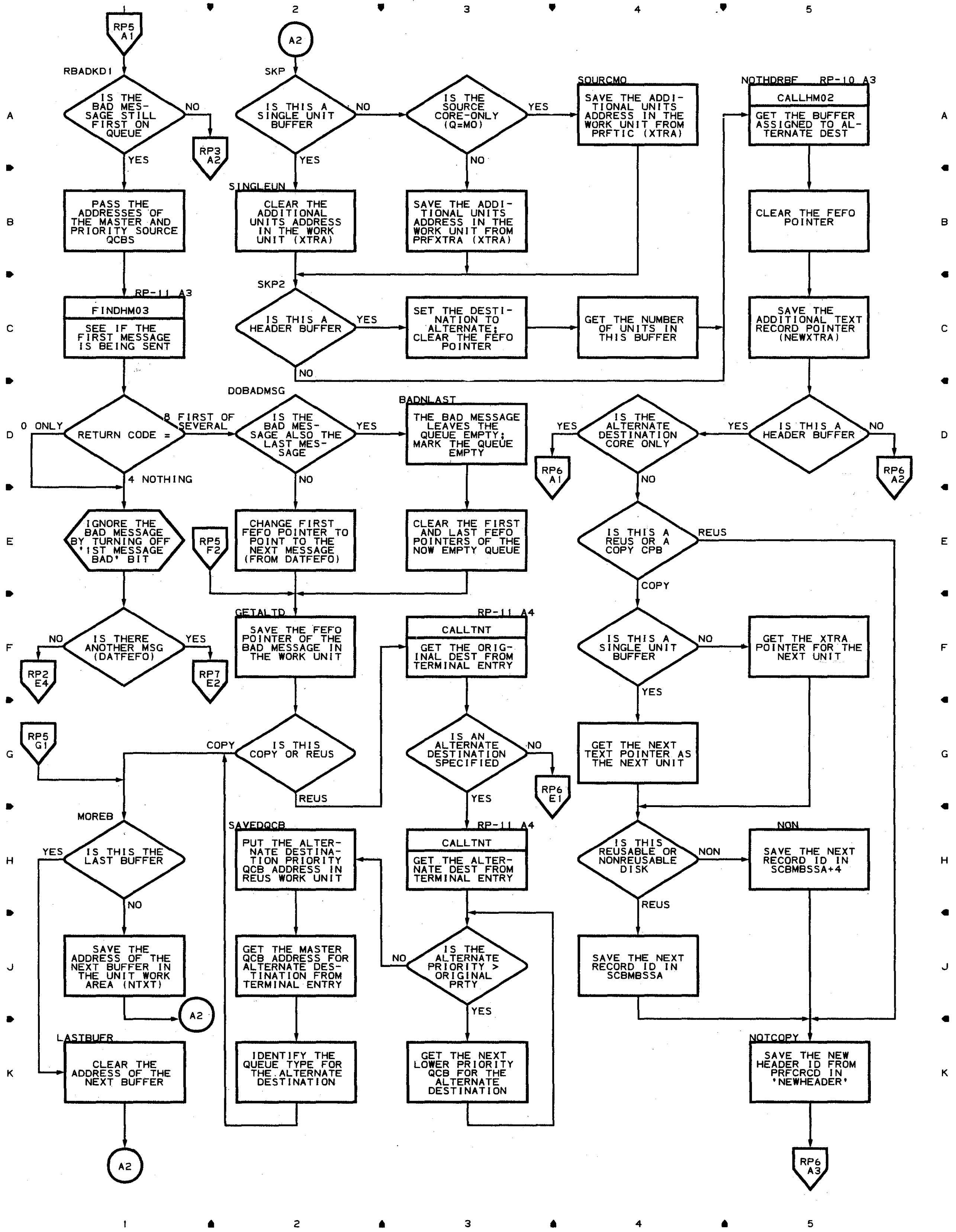

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
RP5 A1
RBADKD1
IS THE BAD MES- SAGE STILL FIRST ON QUEUE
NO
RP3 A2
YES
PASS THE ADDRESSES OF THE MASTER AND PRIORITY SOURCE QCBS
RP-11 A3
FINDHM03
SEE IF THE FIRST MESSAGE IS BEING SENT
RETURN CODE =
0 ONLY
8 FIRST OF SEVERAL
4 NOTHING
IGNORE THE BAD MESSAGE BY TURNING OFF '1ST MESSAGE BAD' BIT
IS THERE ANOTHER MSG (DATFEFO)
NO
YES
RP2 E4
RP7 E2
RP5 G1
MOREB
IS THIS THE LAST BUFFER
YES
NO
SAVE THE ADDRESS OF THE NEXT BUFFER IN THE UNIT WORK AREA (NTXT)
A2
LASTBUFR
CLEAR THE ADDRESS OF THE NEXT BUFFER
A2
A2
SKP
IS THIS A SINGLE UNIT BUFFER
NO
YES
SINGLEUN
CLEAR THE ADDITIONAL UNITS ADDRESS IN THE WORK UNIT (XTRA)
SKP2
IS THIS A HEADER BUFFER
YES
NO
DOBADMSG
IS THE BAD MES- SAGE ALSO THE LAST MES- SAGE
YES
NO
CHANGE FIRST FEFO POINTER TO POINT TO THE NEXT MESSAGE (FROM DATFEFO)
RP5 F2
GETALTD
SAVE THE FEFO POINTER OF THE BAD MESSAGE IN THE WORK UNIT
IS THIS COPY OR REUS
COPY
REUS
SAVEDQCB
PUT THE ALTER- NATE DESTINA- TION PRIORITY QCB ADDRESS IN REUS WORK UNIT
GET THE MASTER QCB ADDRESS FOR ALTERNATE DES- TINATION FROM TERMINAL ENTRY
IDENTIFY THE QUEUE TYPE FOR THE ALTERNATE DESTINATION
IS THE SOURCE CORE-ONLY (Q=MO)
YES
NO
SAVE THE ADDI- TIONAL UNITS ADDRESS IN THE WORK UNIT FROM PRFXTRA (XTRA)
SET THE DESTI- NATION TO ALTERNATE; CLEAR THE FEFO POINTER
BADNLAST
THE BAD MESSAGE LEAVES THE QUEUE EMPTY; MARK THE QUEUE EMPTY
CLEAR THE FIRST AND LAST FEFO POINTERS OF THE NOW EMPTY QUEUE
RP-11 A4
CALLTNT
GET THE ORIG- INAL DEST FROM TERMINAL ENTRY
IS AN ALTERNATE DESTINATION SPECIFIED
NO
YES
RP6 E1
RP-11 A4
CALLTNT
GET THE ALTER- NATE DEST FROM TERMINAL ENTRY
IS THE ALTERNATE PRIORITY > ORIGINAL PRTY
NO
YES
GET THE NEXT LOWER PRIORITY QCB FOR THE ALTERNATE DESTINATION
SOURCMO
SAVE THE ADDI- TIONAL UNITS ADDRESS IN THE WORK UNIT FROM PRFTIC (XTRA)
GET THE NUMBER OF UNITS IN THIS BUFFER
IS THE ALTERNATE DESTINATION CORE ONLY
YES
NO
RP6 A1
IS THIS A REUS OR A COPY CPB
REUS
COPY
IS THIS A SINGLE UNIT BUFFER
NO
YES
GET THE NEXT TEXT POINTER AS THE NEXT UNIT
IS THIS REUSABLE OR NONREUSABLE DISK
NON
REUS
SAVE THE NEXT RECORD ID IN SCBMBSSA
NOTHDRBF
RP-10 A3
CALLHM02
GET THE BUFFER ASSIGNED TO AL- TERNATE DEST
CLEAR THE FEFO POINTER
SAVE THE ADDITIONAL TEXT RECORD POINTER (NEWXTRA)
IS THIS A HEADER BUFFER
YES
NO
RP6 A2
GET THE XTRA POINTER FOR THE NEXT UNIT
NON
SAVE THE NEXT RECORD ID IN SCBMBSSA+4
NOTCOPY
SAVE THE NEW HEADER ID FROM PRFCRCD IN 'NEWHEADER'
RP6 A3

Chart RP-6 REUSABILITY-COPY SUBTASK

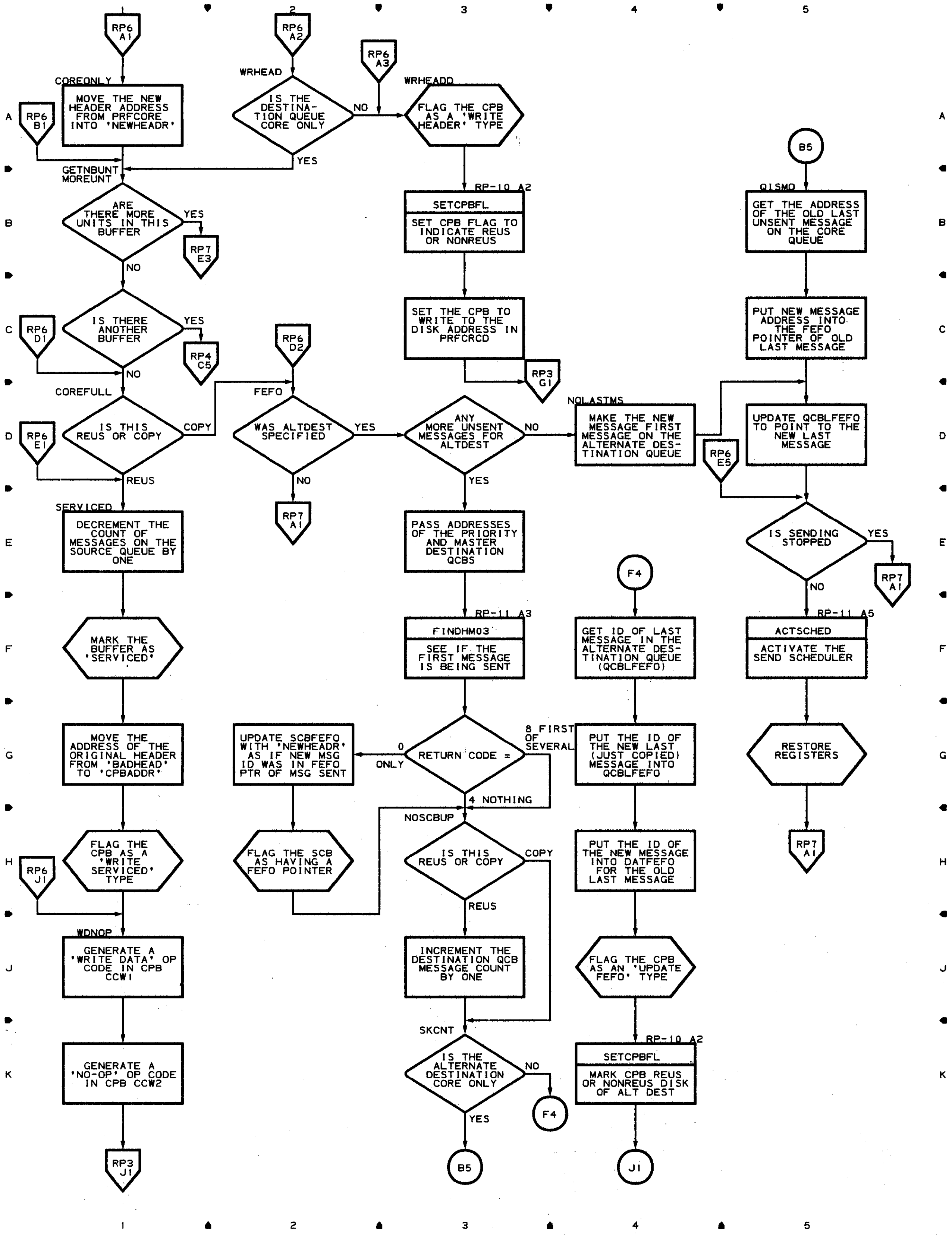

RP6 A1
COREONLY
MOVE THE NEW HEADER ADDRESS FROM PRFCORE INTO 'NEWHEADR'
RP6 B1
RP6 A2
WRHEAD
IS THE DESTINA-TION QUEUE CORE ONLY
NO
YES
RP6 A3
WRHEADD
FLAG THE CPB AS A 'WRITE HEADER' TYPE
B5
QISMO
GET THE ADDRESS OF THE OLD LAST UNSENT MESSAGE ON THE CORE QUEUE
GETNBUNT
MOREUNT
ARE THERE MORE UNITS IN THIS BUFFER
YES
RP7 E3
NO
RP-10 A2
SETCPBFL
SET CPB FLAG TO INDICATE REUS OR NONREUS
PUT NEW MESSAGE ADDRESS INTO THE FEFO POINTER OF OLD LAST MESSAGE
RP6 D1
IS THERE ANOTHER BUFFER
YES
RP4 C5
NO
SET THE CPB TO WRITE TO THE DISK ADDRESS IN PRFCRCD
RP3 G1
RP6 D2
COREFULL
FEFO
NOLASTMS
RP6 E1
IS THIS REUS OR COPY
COPY
REUS
WAS ALTDEST SPECIFIED
YES
NO
RP7 A1
ANY MORE UNSENT MESSAGES FOR ALTDEST
NO
YES
MAKE THE NEW MESSAGE FIRST MESSAGE ON THE ALTERNATE DES-TINATION QUEUE
UPDATE QCBLFEFO TO POINT TO THE NEW LAST MESSAGE
RP6 E5
SERVICED
DECREMENT THE COUNT OF MESSAGES ON THE SOURCE QUEUE BY ONE
PASS ADDRESSES OF THE PRIORITY AND MASTER DESTINATION QCBS
F4
IS SENDING STOPPED
YES
RP7 A1
NO
MARK THE BUFFER AS 'SERVICED'
RP-11 A3
FINDHM03
SEE IF THE FIRST MESSAGE IS BEING SENT
GET ID OF LAST MESSAGE IN THE ALTERNATE DES-TINATION QUEUE (QCBLFEFO)
RP-11 A5
ACTSCHED
ACTIVATE THE SEND SCHEDULER
MOVE THE ADDRESS OF THE ORIGINAL HEADER FROM 'BADHEAD' TO 'CPBADDR'
UPDATE SCBFEFO WITH 'NEWHEADR' AS IF NEW MSG ID WAS IN FEFO PTR OF MSG SENT
0
ONLY
RETURN CODE =
8 FIRST OF SEVERAL
4 NOTHING
PUT THE ID OF THE NEW LAST (JUST COPIED) MESSAGE INTO QCBLFEFO
RESTORE REGISTERS
NOSCBUP
RP6 J1
FLAG THE CPB AS A 'WRITE SERVICED' TYPE
FLAG THE SCB AS HAVING A FEFO POINTER
IS THIS REUS OR COPY
COPY
REUS
PUT THE ID OF THE NEW MESSAGE INTO FOR THE OLD LAST MESSAGE
RP7 A1
WDNOP
GENERATE A 'WRITE DATA' OP CODE IN CPB CCW1
INCREMENT THE DESTINATION QCB MESSAGE COUNT BY ONE
FLAG THE CPB AS AN 'UPDATE FEFO' TYPE
SKCNT
RP-10 A2
SETCPBFL
GENERATE A 'NO-OP' OP CODE IN CPB CCW2
IS THE ALTERNATE DESTINATION CORE ONLY
NO
F4
YES
MARK CPB REUS OR NONREUS DISK OF ALT DEST
RP3 J1
B5
J1
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart RP-7 REUSABILITY-COPY SUBTASK

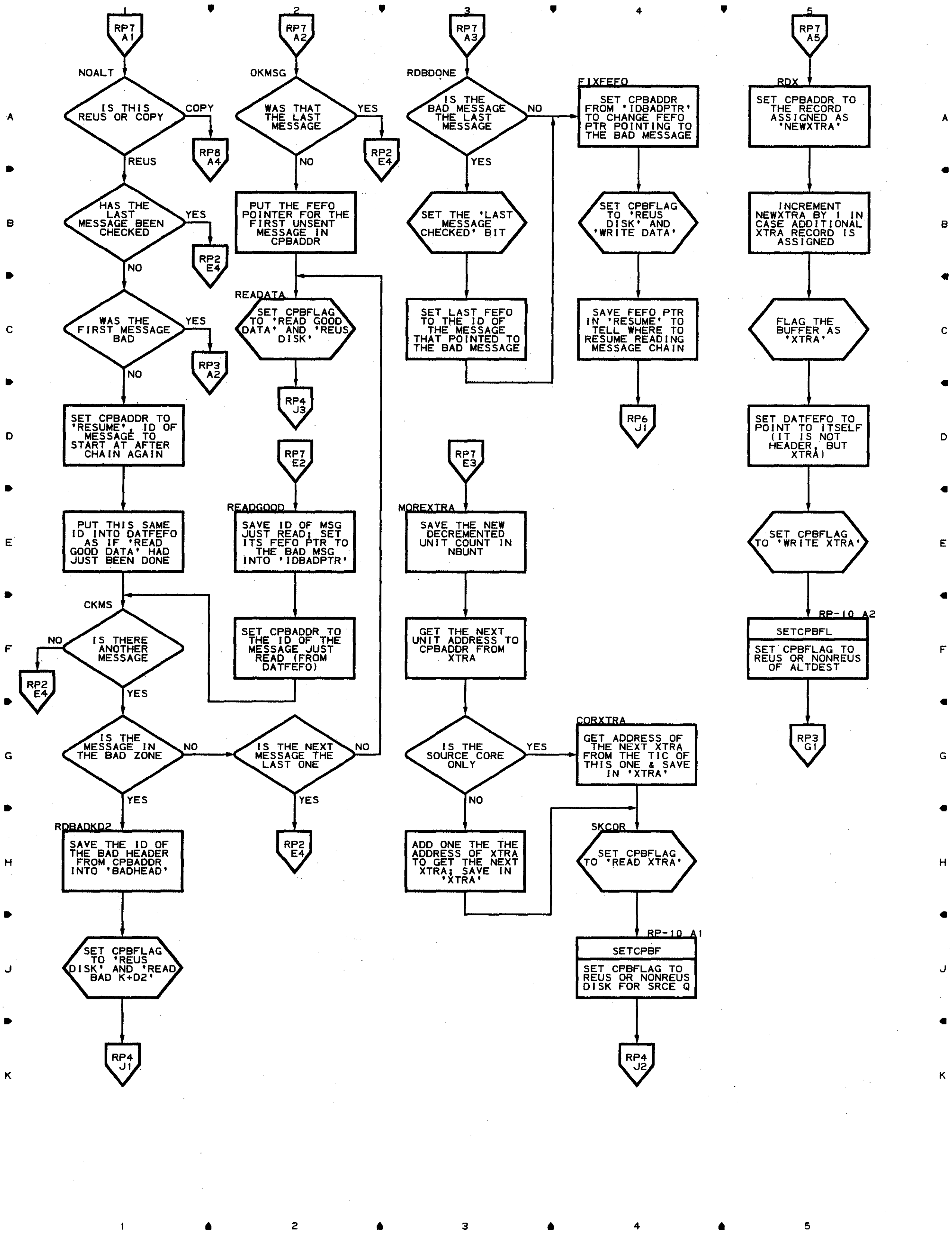
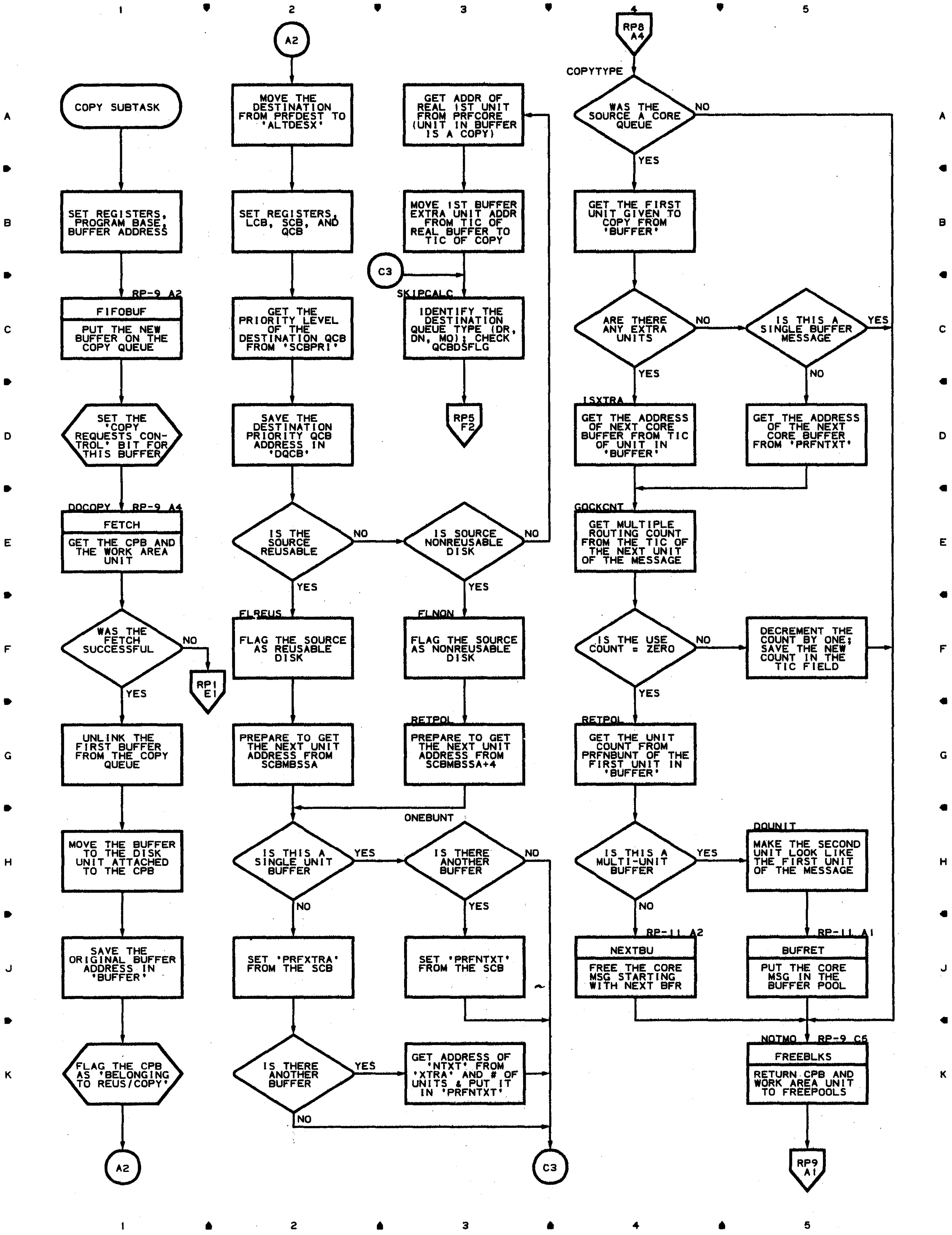

RP7 A1
NOALT
IS THIS REUS OR COPY
COPY
RP8 A4
REUS
HAS THE LAST MESSAGE BEEN CHECKED
YES
RP2 E4
NO
WAS THE FIRST MESSAGE BAD
YES
RP3 A2
NO
SET CPBADDR TO 'RESUME', ID OF MESSAGE TO START AT AFTER CHAIN AGAIN
PUT THIS SAME ID INTO DATFEFO AS IF 'READ GOOD DATA' HAD JUST BEEN DONE
CKMS
IS THERE ANOTHER MESSAGE
NO
RP2 E4
YES
IS THE MESSAGE IN THE BAD ZONE
NO
YES
RDBADKD2
SAVE THE ID OF THE BAD HEADER FROM CPBADDR INTO 'BADHEAD'
SET CPBFLAG TO 'REUS DISK' AND 'READ BAD K+D2'
RP4 J1
RP7 A2
OKMSG
WAS THAT THE LAST MESSAGE
YES
RP2 E4
NO
PUT THE FEFO POINTER FOR THE FIRST UNSENT MESSAGE IN CPBADDR
READATA
SET CPBFLAG TO 'READ GOOD DATA' AND 'REUS DISK'
RP4 J3
RP7 E2
READGOOD
SAVE ID OF MSG JUST READ; SET ITS FEFO PTR TO THE BAD MSG INTO 'IDBADPTR'
SET CPBADDR TO THE ID OF THE MESSAGE JUST READ (FROM DATFEFO)
IS THE NEXT MESSAGE THE LAST ONE
NO
YES
RP2 E4
RP7 A3
RDBDONE
IS THE BAD MESSAGE THE LAST MESSAGE
NO
YES
SET THE 'LAST MESSAGE CHECKED' BIT
SET LAST FEFO TO THE ID OF THE MESSAGE THAT POINTED TO THE BAD MESSAGE
RP7 E3
MOREXTRA
SAVE THE NEW DECREMENTED UNIT COUNT IN NBUNT
GET THE NEXT UNIT ADDRESS TO CPBADDR FROM XTRA
IS THE SOURCE CORE ONLY
YES
NO
ADD ONE THE THE ADDRESS OF XTRA TO GET THE NEXT XTRA; SAVE IN 'XTRA'
FIXFEFO
SET CPBADDR FROM 'IDBADPTR' TO CHANGE FEFO PTR POINTING TO THE BAD MESSAGE
SET CPBFLAG TO 'REUS DISK' AND 'WRITE DATA'
SAVE FEFO PTR IN 'RESUME' TO TELL WHERE TO RESUME READING MESSAGE CHAIN
RP6 J1
CORXTRA
GET ADDRESS OF THE NEXT XTRA FROM THE TIC OF THIS ONE & SAVE IN 'XTRA'
SKCOR
SET CPBFLAG TO 'READ XTRA'
RP-10 A1
SETCPBF
SET CPBFLAG TO REUS OR NONREUS DISK FOR SRCE Q
RP4 J2
RP7 A5
RDX
SET CPBADDR TO THE RECORD ASSIGNED AS 'NEWXTRA'
INCREMENT NEWXTRA BY 1 IN CASE ADDITIONAL XTRA RECORD IS ASSIGNED
FLAG THE BUFFER AS 'XTRA'
SET DATFEFO TO POINT TO ITSELF (IT IS NOT HEADER, BUT XTRA)
SET CPBFLAG TO 'WRITE XTRA'
RP-10 A2
SETCPBFL
SET CPBFLAG TO REUS OR NONREUS OF ALTDEST
RP3 G1
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart RP-8 REUSABILITY-COPY SUBTASK



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
COPY SUBTASK
SET REGISTERS, PROGRAM BASE, BUFFER ADDRESS
RP-9 A2
FIFOBUF
PUT THE NEW BUFFER ON THE COPY QUEUE
SET THE 'COPY REQUESTS CONTROL' BIT FOR THIS BUFFER
DOCOPY
RP-9 A4
FETCH
GET THE CPB AND THE WORK AREA UNIT
WAS THE FETCH SUCCESSFUL
NO
RP1 E1
YES
UNLINK THE FIRST BUFFER FROM THE COPY QUEUE
MOVE THE BUFFER TO THE DISK UNIT ATTACHED TO THE CPB
SAVE THE ORIGINAL BUFFER ADDRESS IN 'BUFFER'
FLAG THE CPB AS 'BELONGING TO REUS/COPY'
A2
A2
MOVE THE DESTINATION FROM PRFDEST TO 'ALTDESX'
SET REGISTERS, LCB, SCB, AND QCB
GET THE PRIORITY LEVEL OF THE DESTINATION QCB FROM 'SCBPRI'
SAVE THE DESTINATION PRIORITY QCB ADDRESS IN 'DQCB'
IS THE SOURCE REUSABLE
NO
YES
FLREUS
FLAG THE SOURCE AS REUSABLE DISK
PREPARE TO GET THE NEXT UNIT ADDRESS FROM SCBMBSSA
IS THIS A SINGLE UNIT BUFFER
YES
NO
SET 'PRFXTRA' FROM THE SCB
IS THERE ANOTHER BUFFER
YES
NO
GET ADDR OF REAL 1ST UNIT FROM PRFCORE (UNIT IN BUFFER IS A COPY)
MOVE 1ST BUFFER EXTRA UNIT ADDR FROM TIC OF REAL BUFFER TO TIC OF COPY
C3
SKIPCALC
IDENTIFY THE DESTINATION QUEUE TYPE (DR, DN, MO); CHECK QCBDSFLG
RP5 F2
IS SOURCE NONREUSABLE DISK
NO
YES
FLNON
FLAG THE SOURCE AS NONREUSABLE DISK
RETPOL
PREPARE TO GET THE NEXT UNIT ADDRESS FROM SCBMBSSA+4
ONEBUNT
IS THERE ANOTHER BUFFER
NO
YES
SET 'PRFNTXT' FROM THE SCB
GET ADDRESS OF 'NTXT' FROM 'XTRA' AND # OF UNITS & PUT IT IN 'PRFNTXT'
C3
RP8 A4
COPYTYPE
WAS THE SOURCE A CORE QUEUE
NO
YES
GET THE FIRST UNIT GIVEN TO COPY FROM 'BUFFER'
ARE THERE ANY EXTRA UNITS
NO
YES
ISXTRA
GET THE ADDRESS OF NEXT CORE BUFFER FROM TIC OF UNIT IN 'BUFFER'
GOCKCNT
GET MULTIPLE ROUTING COUNT FROM THE TIC OF THE NEXT UNIT OF THE MESSAGE
IS THE USE COUNT = ZERO
NO
YES
RETPOL
GET THE UNIT COUNT FROM PRFNBUNT OF THE FIRST UNIT IN 'BUFFER'
IS THIS A MULTI-UNIT BUFFER
YES
NO
RP-11 A2
NEXTBU
FREE THE CORE MSG STARTING WITH NEXT BFR
IS THIS A SINGLE BUFFER MESSAGE
YES
NO
GET THE ADDRESS OF THE NEXT CORE BUFFER FROM 'PRFNTXT'
DECREMENT THE COUNT BY ONE; SAVE THE NEW COUNT IN THE TIC FIELD
DOUNIT
MAKE THE SECOND UNIT LOOK LIKE THE FIRST UNIT OF THE MESSAGE
RP-11 A1
BUFRET
PUT THE CORE MSG IN THE BUFFER POOL
NOTMO
RP-9 C5
FREEBLKS
RETURN CPB AND WORK AREA UNIT TO FREEPOOLS
RP9 A1

Chart RP-9 REUSABILITY-COPY SUBTASK

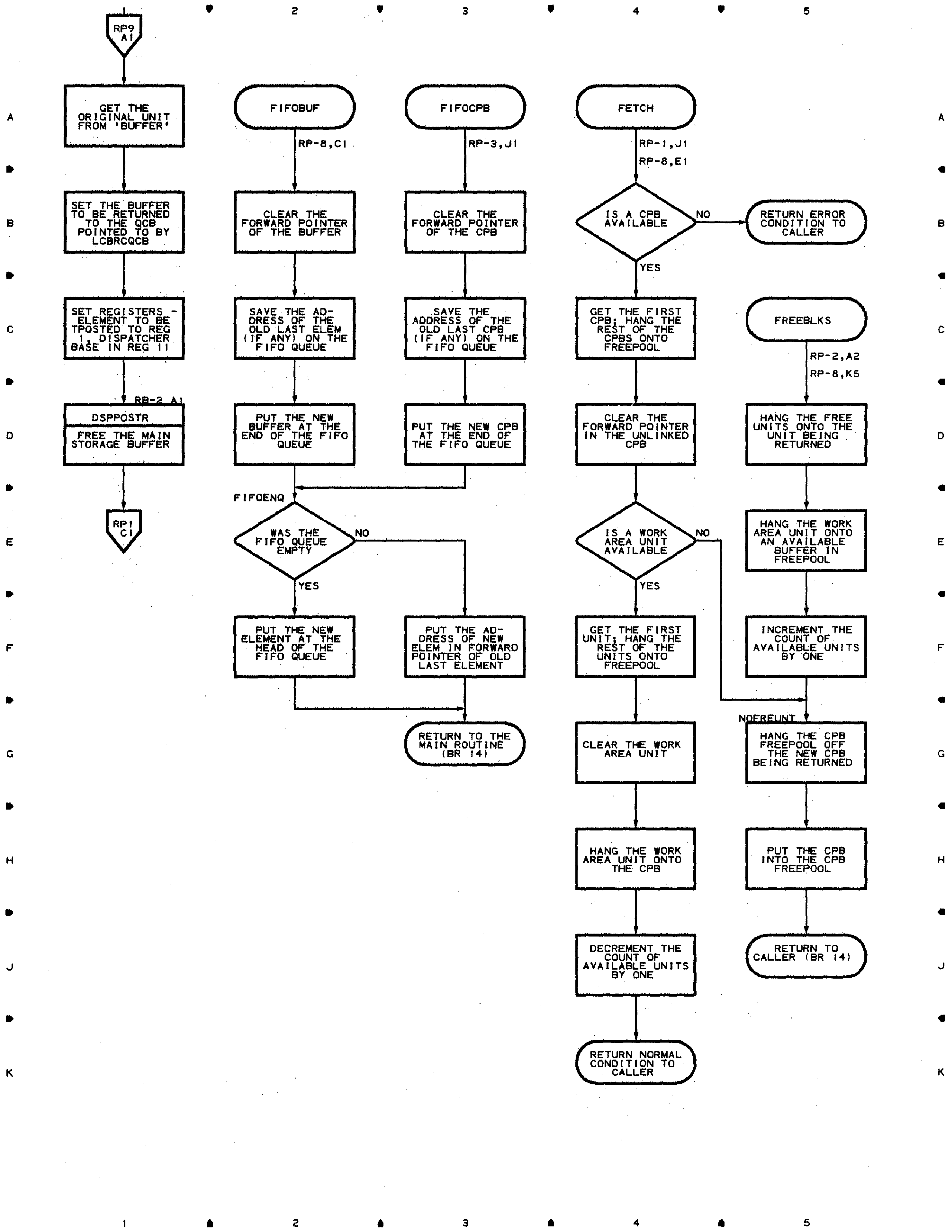
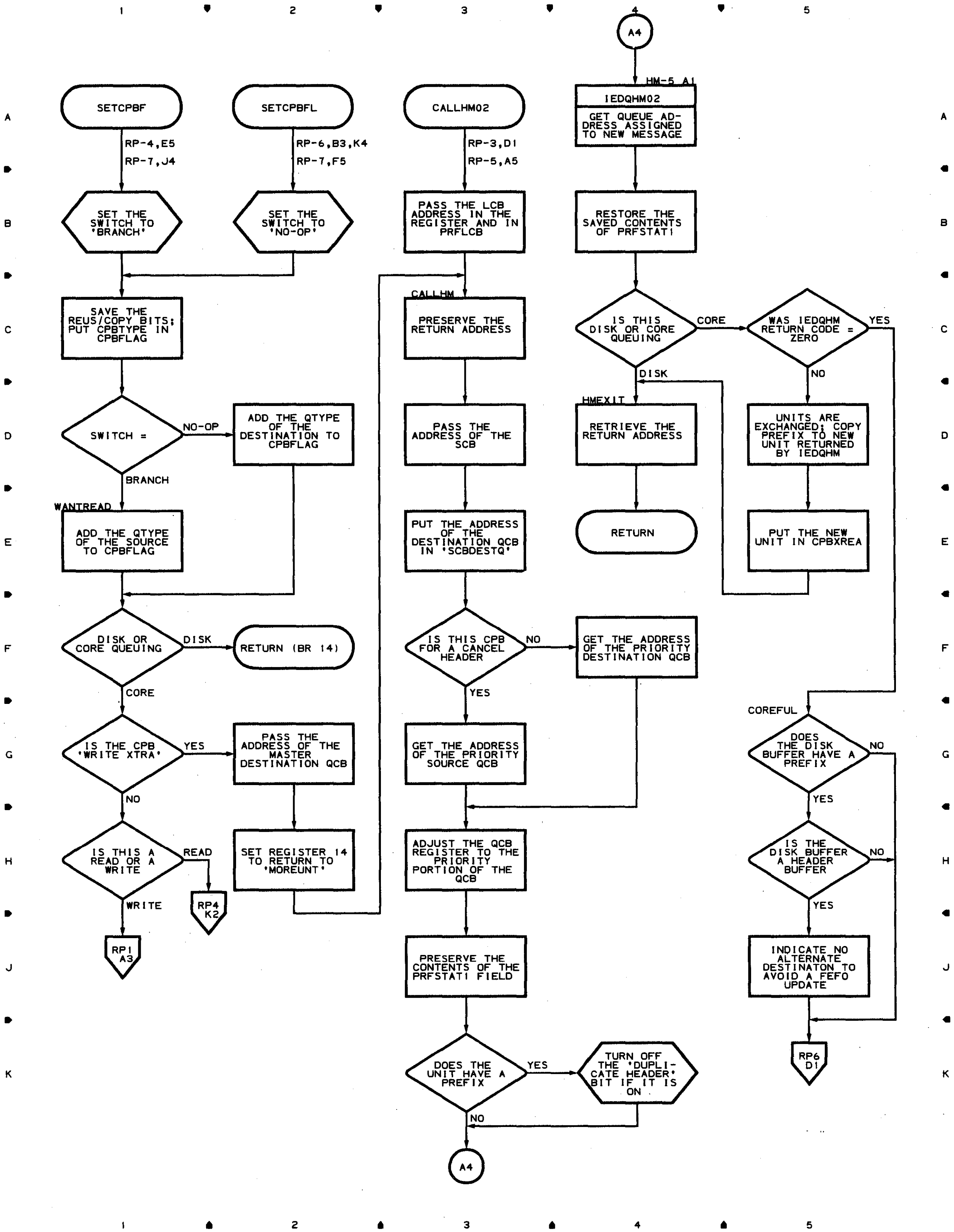

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
RP9 A1
GET THE ORIGINAL UNIT FROM 'BUFFER'
SET THE BUFFER TO BE RETURNED TO THE QCB POINTED TO BY LCBRCQCB
SET REGISTERS - ELEMENT TO BE TPOSTED TO REG 1, DISPATCHER BASE IN REG 11
RB-2 A1
DSPPOSTR
FREE THE MAIN STORAGE BUFFER
RP1 C1
FIFOBUF
RP-8,C1
CLEAR THE FORWARD POINTER OF THE BUFFER
SAVE THE ADDRESS OF THE OLD LAST ELEM (IF ANY) ON THE FIFO QUEUE
PUT THE NEW BUFFER AT THE END OF THE FIFO QUEUE
FIFOENQ
WAS THE FIFO QUEUE EMPTY
NO
YES
PUT THE NEW ELEMENT AT THE HEAD OF THE FIFO QUEUE
FIFOCPB
RP-3,J1
CLEAR THE FORWARD POINTER OF THE CPB
SAVE THE ADDRESS OF THE OLD LAST CPB (IF ANY) ON THE FIFO QUEUE
PUT THE NEW CPB AT THE END OF THE FIFO QUEUE
PUT THE ADDRESS OF NEW ELEM IN FORWARD POINTER OF OLD LAST ELEMENT
RETURN TO THE MAIN ROUTINE (BR 14)
FETCH
RP-1,J1
RP-8,E1
IS A CPB AVAILABLE
NO
YES
RETURN ERROR CONDITION TO CALLER
GET THE FIRST CPB; HANG THE REST OF THE CPBS ONTO FREEPOOL
CLEAR THE FORWARD POINTER IN THE UNLINKED CPB
IS A WORK AREA UNIT AVAILABLE
NO
YES
GET THE FIRST UNIT; HANG THE REST OF THE UNITS ONTO FREEPOOL
CLEAR THE WORK AREA UNIT
HANG THE WORK AREA UNIT ONTO THE CPB
DECREMENT THE COUNT OF AVAILABLE UNITS BY ONE
RETURN NORMAL CONDITION TO CALLER
FREEBLKS
RP-2,A2
RP-8,K5
HANG THE FREE UNITS ONTO THE UNIT BEING RETURNED
HANG THE WORK AREA UNIT ONTO AN AVAILABLE BUFFER IN FREEPOOL
INCREMENT THE COUNT OF AVAILABLE UNITS BY ONE
NOFREUNT
HANG THE CPB FREEPOOL OFF THE NEW CPB BEING RETURNED
PUT THE CPB INTO THE CPB FREEPOOL
RETURN TO CALLER (BR 14)

Chart RP-10 REUSABILITY-COPY SUBTASK



SETCPBF
RP-4,E5
RP-7,J4
SET THE SWITCH TO 'BRANCH'
SETCPBFL
RP-6,B3,K4
RP-7,F5
SET THE SWITCH TO 'NO-OP'
SAVE THE REUS/COPY BITS; PUT CPBTYPE IN CPBFLAG
SWITCH =
NO-OP
ADD THE QTYPE OF THE DESTINATION TO CPBFLAG
BRANCH
WANTREAD
ADD THE QTYPE OF THE SOURCE TO CPBFLAG
DISK OR CORE QUEUING
DISK
RETURN (BR 14)
CORE
IS THE CPB 'WRITE XTRA'
YES
PASS THE ADDRESS OF THE MASTER DESTINATION QCB
NO
IS THIS A READ OR A WRITE
READ
RP4 K2
WRITE
RP1 A3
SET REGISTER 14 TO RETURN TO 'MOREUNT'
CALLHM02
RP-3,D1
RP-5,A5
PASS THE LCB ADDRESS IN THE REGISTER AND IN PRFLCB
CALLHM
PRESERVE THE RETURN ADDRESS
PASS THE ADDRESS OF THE SCB
PUT THE ADDRESS OF THE DESTINATION QCB IN 'SCBDESTQ'
IS THIS CPB FOR A CANCEL HEADER
NO
GET THE ADDRESS OF THE PRIORITY DESTINATION QCB
YES
GET THE ADDRESS OF THE PRIORITY SOURCE QCB
ADJUST THE QCB REGISTER TO THE PRIORITY PORTION OF THE QCB
PRESERVE THE CONTENTS OF THE PRFSTATI FIELD
DOES THE UNIT HAVE A PREFIX
YES
TURN OFF THE 'DUPLICATE HEADER' BIT IF IT IS ON
NO
A4
A4
HM-5 A1
IEDQHM02
GET QUEUE ADDRESS ASSIGNED TO NEW MESSAGE
RESTORE THE SAVED CONTENTS OF PRFSTATI
IS THIS DISK OR CORE QUEUING
CORE
DISK
HMEXIT
RETRIEVE THE RETURN ADDRESS
RETURN
WAS IEDQHM RETURN CODE = ZERO
YES
NO
UNITS ARE EXCHANGED; COPY PREFIX TO NEW UNIT RETURNED BY IEDQHM
PUT THE NEW UNIT IN CPBXREA
COREFUL
DOES THE DISK BUFFER HAVE A PREFIX
NO
YES
IS THE DISK BUFFER A HEADER BUFFER
NO
YES
INDICATE NO ALTERNATE DESTINATON TO AVOID A FEFO UPDATE
RP6 D1
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart RP-11 REUSABILITY-COPY SUBTASK

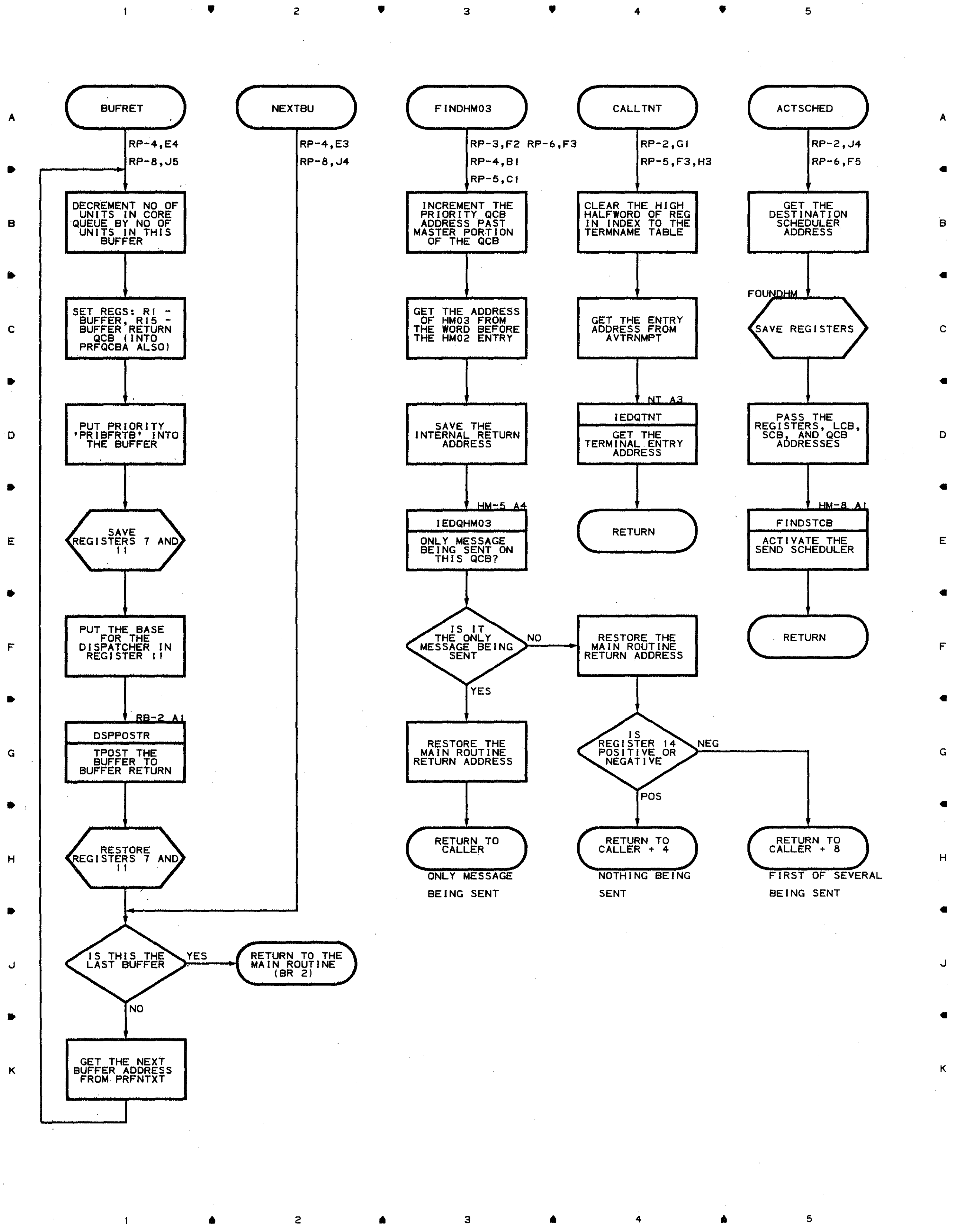
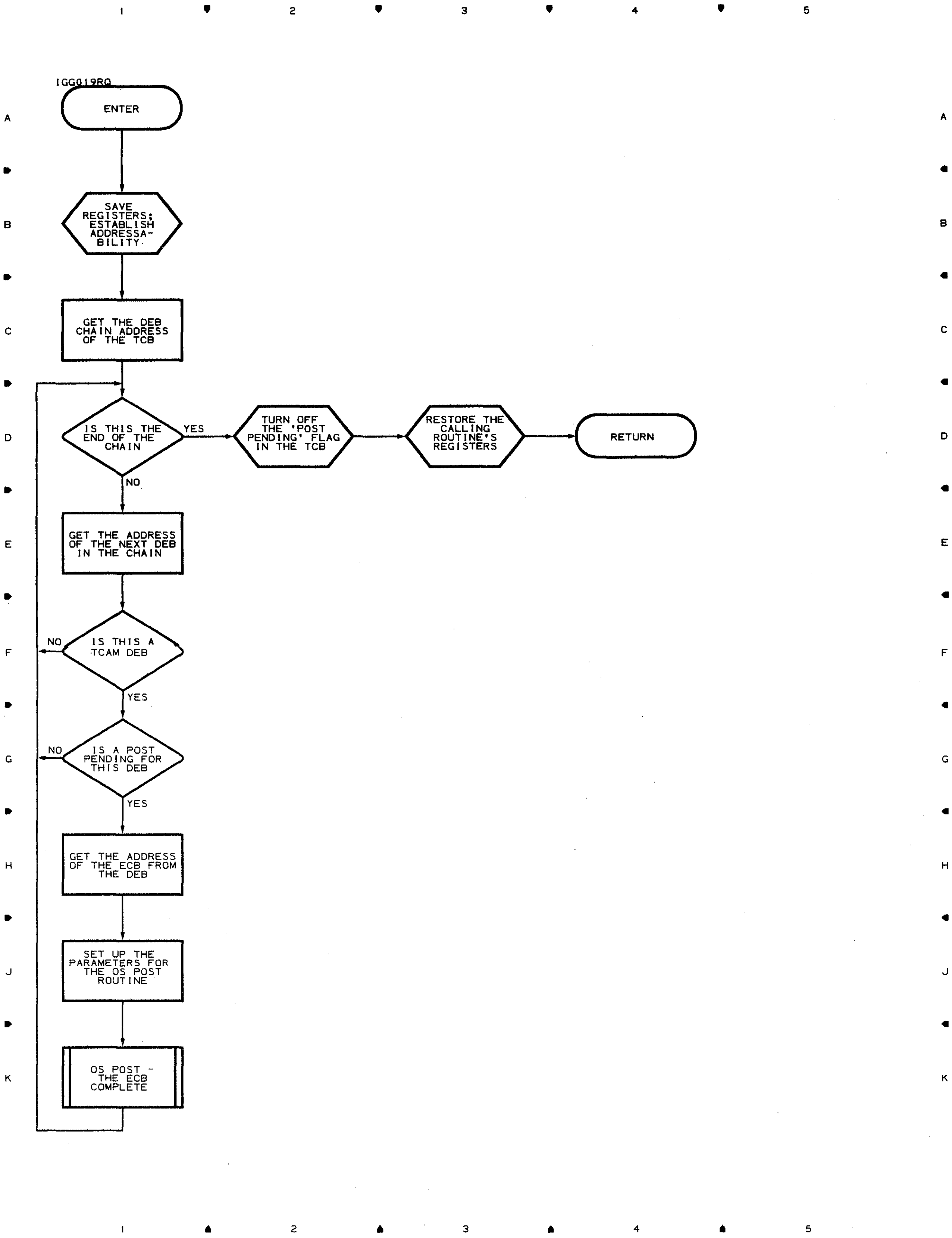

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
BUFRET
RP-4,E4
RP-8,J5
DECREMENT NO OF UNITS IN CORE QUEUE BY NO OF UNITS IN THIS BUFFER
SET REGS: R1 - BUFFER, R15 - BUFFER RETURN QCB (INTO PRFQCBA ALSO)
PUT PRIORITY 'PRIBFRTB' INTO THE BUFFER
SAVE REGISTERS 7 AND 11
PUT THE BASE FOR THE DISPATCHER IN REGISTER 11
RB-2 A1
DSPPOSTR
TPOST THE BUFFER TO BUFFER RETURN
RESTORE REGISTERS 7 AND 11
IS THIS THE LAST BUFFER
YES
NO
GET THE NEXT BUFFER ADDRESS FROM PRFNTXT
NEXTBU
RP-4,E3
RP-8,J4
RETURN TO THE MAIN ROUTINE (BR 2)
FINDHM03
RP-3,F2 RP-6,F3
RP-4,B1
RP-5,C1
INCREMENT THE PRIORITY QCB ADDRESS PAST MASTER PORTION OF THE QCB
GET THE ADDRESS OF HM03 FROM THE WORD BEFORE THE HM02 ENTRY
SAVE THE INTERNAL RETURN ADDRESS
HM-5 A4
IEDQHM03
ONLY MESSAGE BEING SENT ON THIS QCB?
IS IT THE ONLY MESSAGE BEING SENT
NO
YES
RESTORE THE MAIN ROUTINE RETURN ADDRESS
RETURN TO CALLER
ONLY MESSAGE BEING SENT
CALLTNT
RP-2,G1
RP-5,F3,H3
CLEAR THE HIGH HALFWORD OF REG IN INDEX TO THE TERMNAME TABLE
GET THE ENTRY ADDRESS FROM AVTRNMPT
NT A3
IEDQTNT
GET THE TERMINAL ENTRY ADDRESS
RETURN
RESTORE THE MAIN ROUTINE RETURN ADDRESS
IS REGISTER 14 POSITIVE OR NEGATIVE
NEG
POS
RETURN TO CALLER + 4
NOTHING BEING SENT
ACTSCHED
RP-2,J4
RP-6,F5
GET THE DESTINATION SCHEDULER ADDRESS
FOUNDHM
SAVE REGISTERS
PASS THE REGISTERS, LCB, SCB, AND QCB ADDRESSES
HM-8 A1
FINDSTCB
ACTIVATE THE SEND SCHEDULER
RETURN
RETURN TO CALLER + 8
FIRST OF SEVERAL BEING SENT

Chart RQ POST PENDING ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG019RQ
ENTER
SAVE REGISTERS; ESTABLISH ADDRESSA-BILITY
GET THE DEB CHAIN ADDRESS OF THE TCB
IS THIS THE END OF THE CHAIN
YES
TURN OFF THE 'POST PENDING' FLAG IN THE TCB
RESTORE THE CALLING ROUTINE'S REGISTERS
RETURN
NO
GET THE ADDRESS OF THE NEXT DEB IN THE CHAIN
NO
IS THIS A TCAM DEB
YES
NO
IS A POST PENDING FOR THIS DEB
YES
GET THE ADDRESS OF THE ECB FROM THE DEB
SET UP THE PARAMETERS FOR THE OS POST ROUTINE
OS POST - THE ECB COMPLETE

Chart R0-1 LINE END APPENDAGE

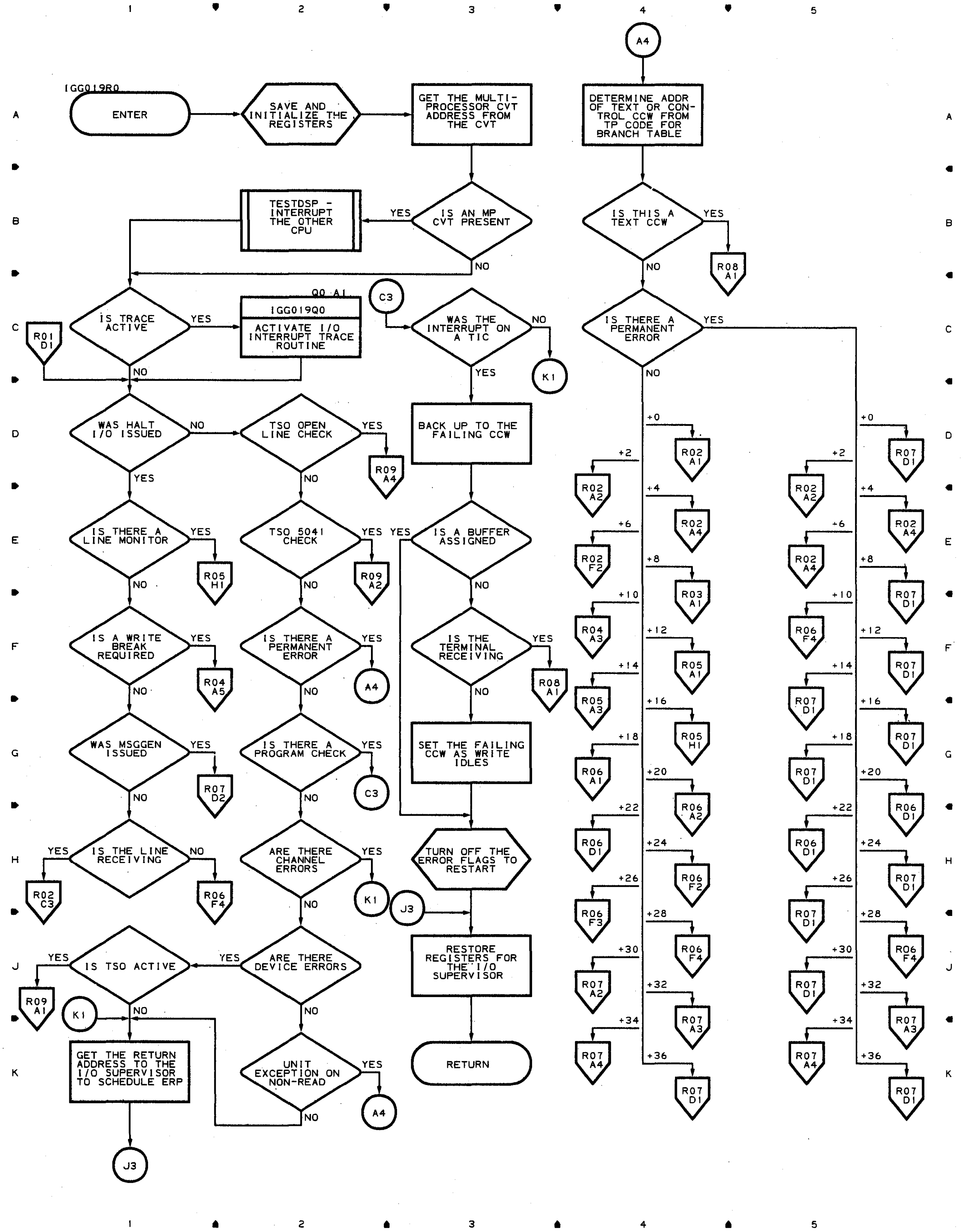


Chart R0-2 LINE END APPENDAGE

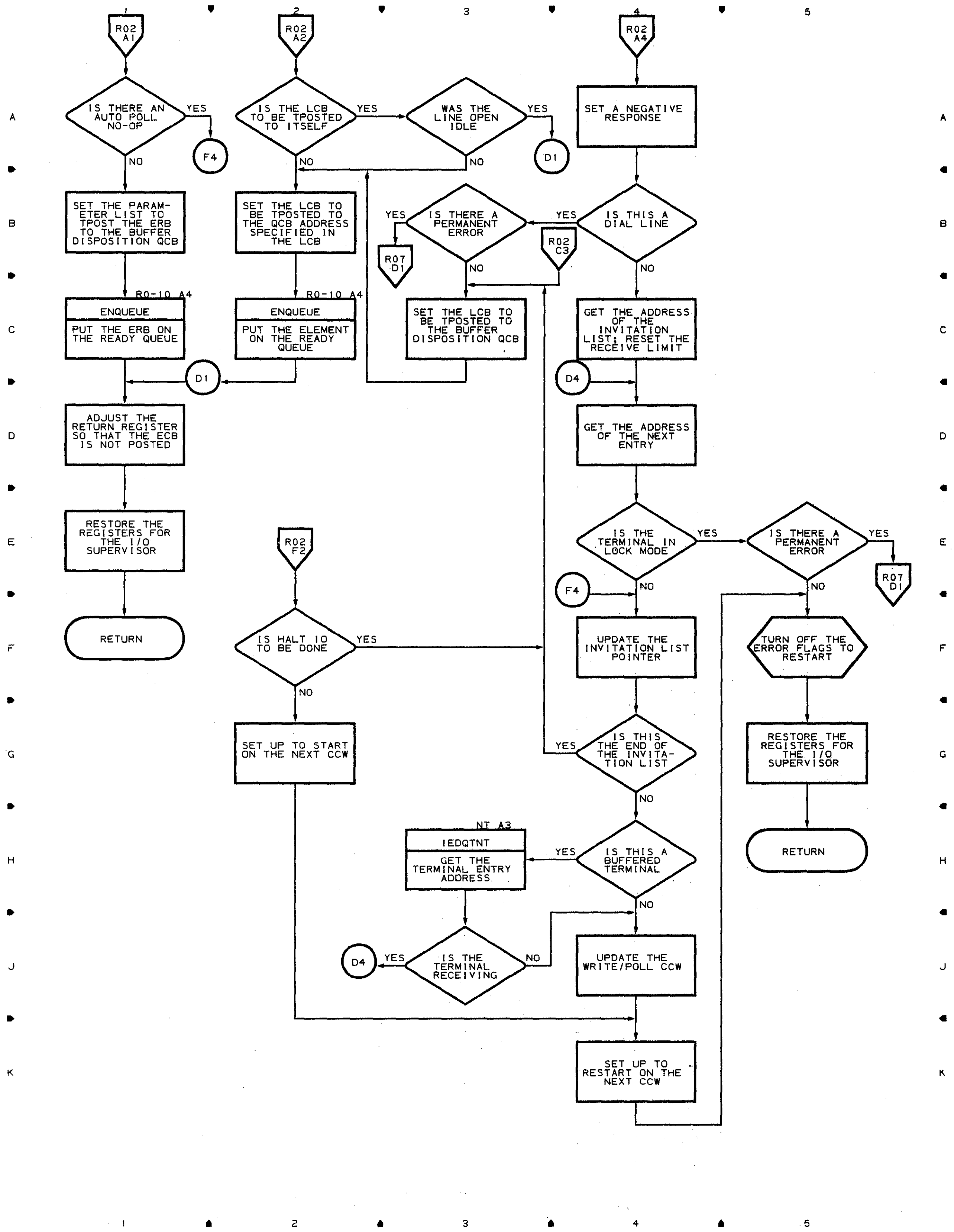

R02 A1
IS THERE AN AUTO POLL NO-OP
YES
F4
NO
SET THE PARAMETER LIST TO TPOST THE ERB TO THE BUFFER DISPOSITION QCB
R0-10 A4
ENQUEUE
PUT THE ERB ON THE READY QUEUE
D1
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN
R02 A2
IS THE LCB TO BE TPOSTED TO ITSELF
YES
NO
SET THE LCB TO BE TPOSTED TO THE QCB ADDRESS SPECIFIED IN THE LCB
R0-10 A4
ENQUEUE
PUT THE ELEMENT ON THE READY QUEUE
WAS THE LINE OPEN IDLE
YES
D1
NO
IS THERE A PERMANENT ERROR
YES
R07 D1
NO
SET THE LCB TO BE TPOSTED TO THE BUFFER DISPOSITION QCB
R02 C3
R02 F2
IS HALT IO TO BE DONE
YES
NO
SET UP TO START ON THE NEXT CCW
R02 A4
SET A NEGATIVE RESPONSE
IS THIS A DIAL LINE
YES
NO
GET THE ADDRESS OF THE INVITATION LIST; RESET THE RECEIVE LIMIT
D4
GET THE ADDRESS OF THE NEXT ENTRY
IS THE TERMINAL IN LOCK MODE
YES
NO
F4
UPDATE THE INVITATION LIST POINTER
IS THIS THE END OF THE INVITATION LIST
YES
NO
IS THIS A BUFFERED TERMINAL
YES
NO
NT A3
IEDQTNT
GET THE TERMINAL ENTRY ADDRESS.
IS THE TERMINAL RECEIVING
YES
D4
NO
UPDATE THE WRITE/POLL CCW
SET UP TO RESTART ON THE NEXT CCW
IS THERE A PERMANENT ERROR
YES
R07 D1
NO
TURN OFF THE ERROR FLAGS TO RESTART
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart R0-3 LINE END APPENDAGE

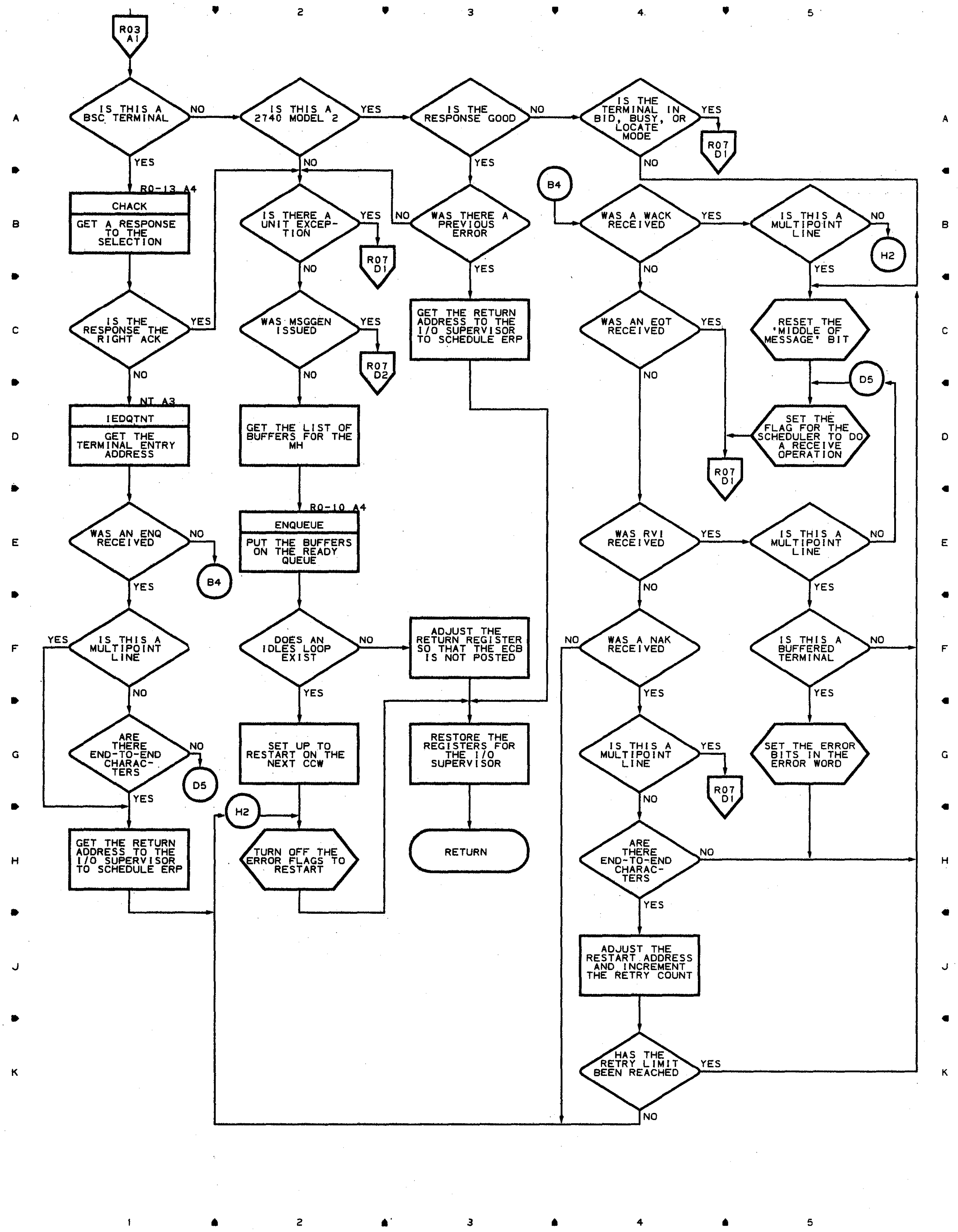

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
R03 A1
IS THIS A BSC TERMINAL
NO
YES
R0-13 A4
CHACK
GET A RESPONSE TO THE SELECTION
IS THE RESPONSE THE RIGHT ACK
YES
NO
NT A3
IEDQTNT
GET THE TERMINAL ENTRY ADDRESS
WAS AN ENQ RECEIVED
NO
B4
YES
IS THIS A MULTIPOINT LINE
YES
NO
ARE THERE END-TO-END CHARAC-TERS
NO
D5
YES
GET THE RETURN ADDRESS TO THE I/O SUPERVISOR TO SCHEDULE ERP
IS THIS A 2740 MODEL 2
YES
NO
IS THERE A UNIT EXCEP-TION
YES
R07 D1
NO
WAS MSGGEN ISSUED
YES
R07 D2
NO
GET THE LIST OF BUFFERS FOR THE MH
R0-10 A4
ENQUEUE
PUT THE BUFFERS ON THE READY QUEUE
DOES AN IDLES LOOP EXIST
NO
YES
SET UP TO RESTART ON THE NEXT CCW
H2
TURN OFF THE ERROR FLAGS TO RESTART
IS THE RESPONSE GOOD
NO
YES
NO
WAS THERE A PREVIOUS ERROR
YES
GET THE RETURN ADDRESS TO THE I/O SUPERVISOR TO SCHEDULE ERP
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN
IS THE TERMINAL IN BID, BUSY, OR LOCATE MODE
YES
R07 D1
NO
B4
WAS A WACK RECEIVED
YES
NO
WAS AN EOT RECEIVED
YES
R07 D1
NO
WAS RVI RECEIVED
YES
NO
WAS A NAK RECEIVED
NO
YES
IS THIS A MULTIPOINT LINE
YES
R07 D1
NO
ARE THERE END-TO-END CHARAC-TERS
NO
YES
ADJUST THE RESTART ADDRESS AND INCREMENT THE RETRY COUNT
HAS THE RETRY LIMIT BEEN REACHED
YES
NO
IS THIS A MULTIPOINT LINE
NO
H2
YES
RESET THE 'MIDDLE OF MESSAGE' BIT
D5
SET THE FLAG FOR THE SCHEDULER TO DO A RECEIVE OPERATION
IS THIS A MULTIPOINT LINE
NO
YES
IS THIS A BUFFERED TERMINAL
NO
YES
SET THE ERROR BITS IN THE ERROR WORD

Chart R0-4 LINE END APPENDAGE

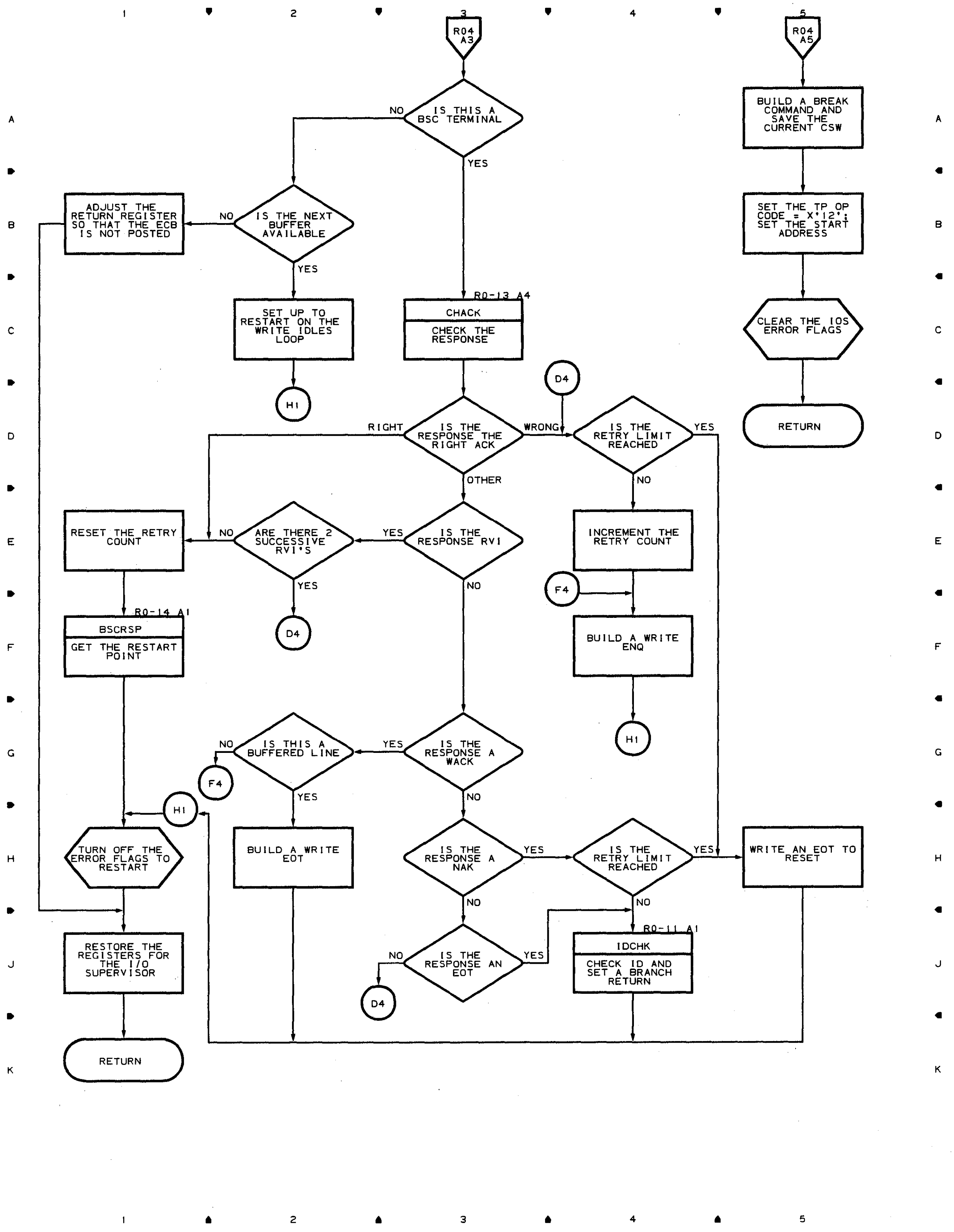

1
2
3
4
5
R04 A3
R04 A5
IS THIS A BSC TERMINAL
NO
YES
BUILD A BREAK COMMAND AND SAVE THE CURRENT CSW
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
NO
IS THE NEXT BUFFER AVAILABLE
YES
SET THE TP OP CODE = X'12'; SET THE START ADDRESS
SET UP TO RESTART ON THE WRITE IDLES LOOP
R0-13 A4
CHACK
CHECK THE RESPONSE
CLEAR THE IOS ERROR FLAGS
H1
D4
RETURN
RIGHT
IS THE RESPONSE THE RIGHT ACK
WRONG
IS THE RETRY LIMIT REACHED
YES
OTHER
NO
RESET THE RETRY COUNT
NO
ARE THERE 2 SUCCESSIVE RVI'S
YES
IS THE RESPONSE RVI
INCREMENT THE RETRY COUNT
YES
NO
F4
R0-14 A1
BSCRSP
GET THE RESTART POINT
D4
BUILD A WRITE ENQ
H1
NO
IS THIS A BUFFERED LINE
YES
IS THE RESPONSE A WACK
F4
YES
NO
H1
TURN OFF THE ERROR FLAGS TO RESTART
BUILD A WRITE EOT
IS THE RESPONSE A NAK
YES
IS THE RETRY LIMIT REACHED
YES
WRITE AN EOT TO RESET
NO
NO
R0-11 A1
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
NO
IS THE RESPONSE AN EOT
YES
IDCHK
CHECK ID AND SET A BRANCH RETURN
D4
RETURN
A
B
C
D
E
F
G
H
J
K

Chart R0-5 LINE END APPENDAGE

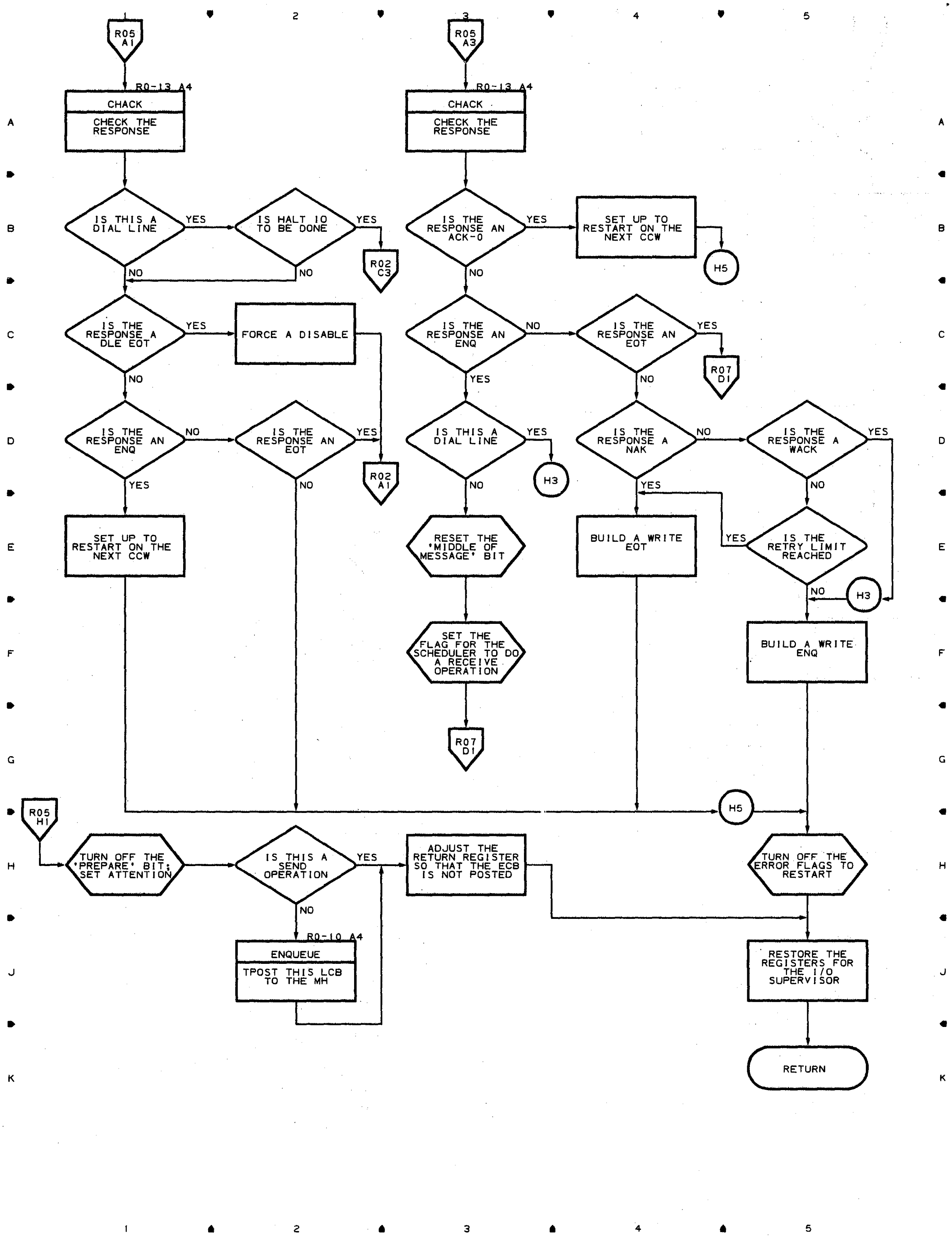


Chart R0-6 LINE END APPENDAGE

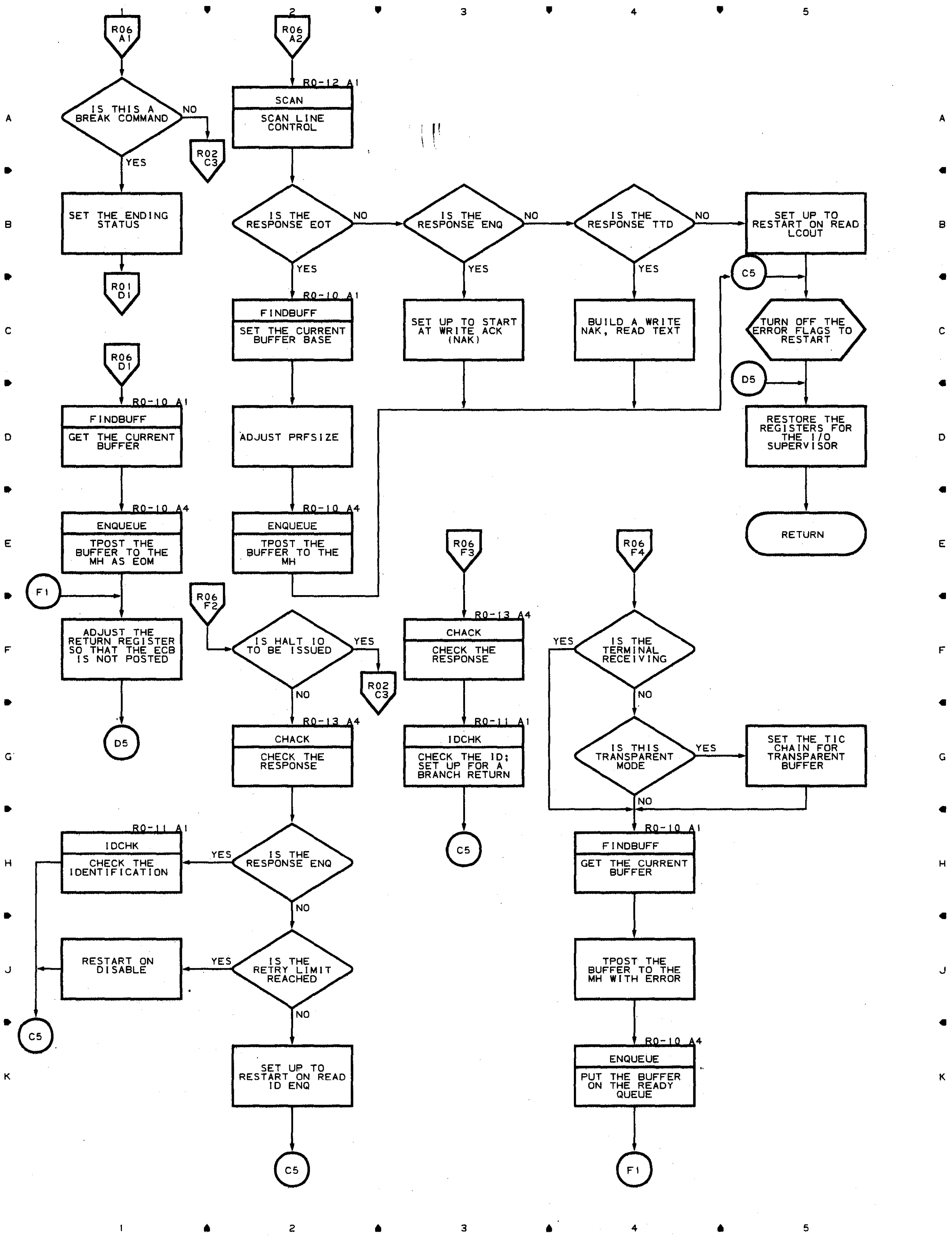


Chart R0-7 LINE END APPENDAGE

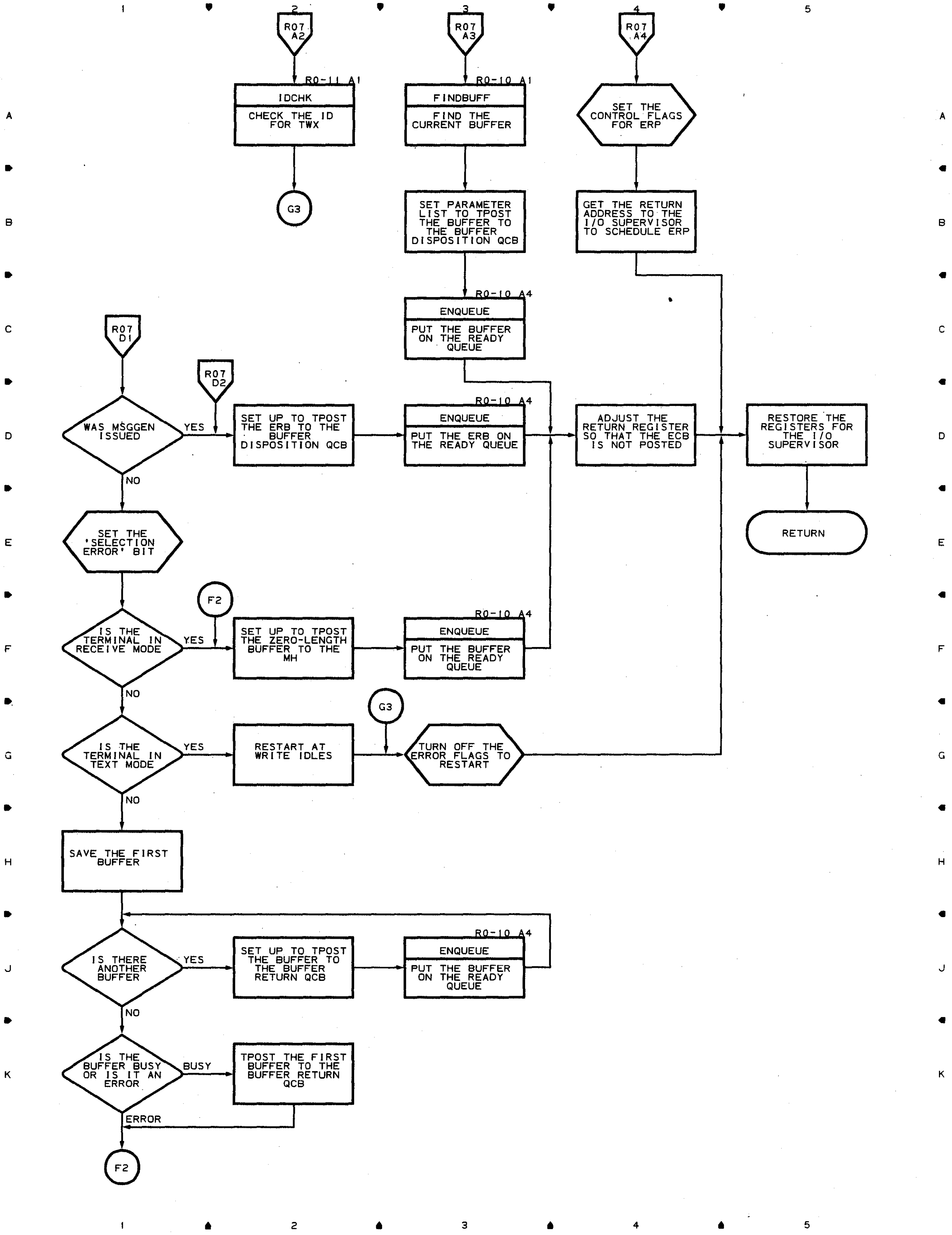

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
R07 A2
R0-11 A1
IDCHK
CHECK THE ID FOR TWX
G3
R07 A3
R0-10 A1
FINDBUFF
FIND THE CURRENT BUFFER
SET PARAMETER LIST TO TPOST THE BUFFER TO THE BUFFER DISPOSITION QCB
R0-10 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
R07 A4
SET THE CONTROL FLAGS FOR ERP
GET THE RETURN ADDRESS TO THE I/O SUPERVISOR TO SCHEDULE ERP
R07 D1
R07 D2
WAS MSGGEN ISSUED
YES
NO
SET UP TO TPOST THE ERB TO THE BUFFER DISPOSITION QCB
R0-10 A4
ENQUEUE
PUT THE ERB ON THE READY QUEUE
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN
SET THE 'SELECTION ERROR' BIT
F2
IS THE TERMINAL IN RECEIVE MODE
YES
NO
SET UP TO TPOST THE ZERO-LENGTH BUFFER TO THE MH
R0-10 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
G3
IS THE TERMINAL IN TEXT MODE
YES
NO
RESTART AT WRITE IDLES
TURN OFF THE ERROR FLAGS TO RESTART
SAVE THE FIRST BUFFER
IS THERE ANOTHER BUFFER
YES
NO
SET UP TO TPOST THE BUFFER TO THE BUFFER RETURN QCB
R0-10 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
IS THE BUFFER BUSY OR IS IT AN ERROR
BUSY
ERROR
TPOST THE FIRST BUFFER TO THE BUFFER RETURN QCB
F2

Chart R08 LINE END APPENDAGE

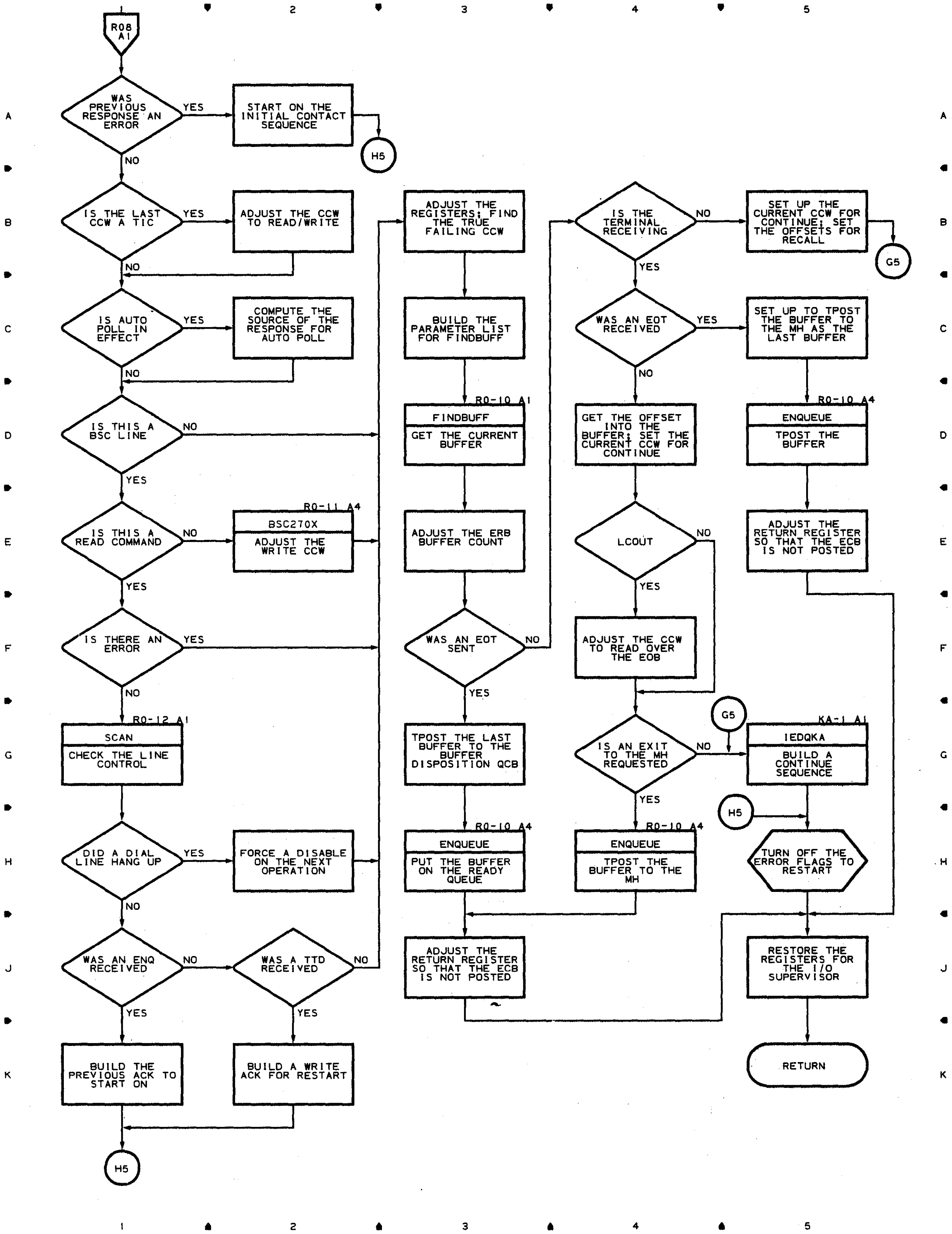
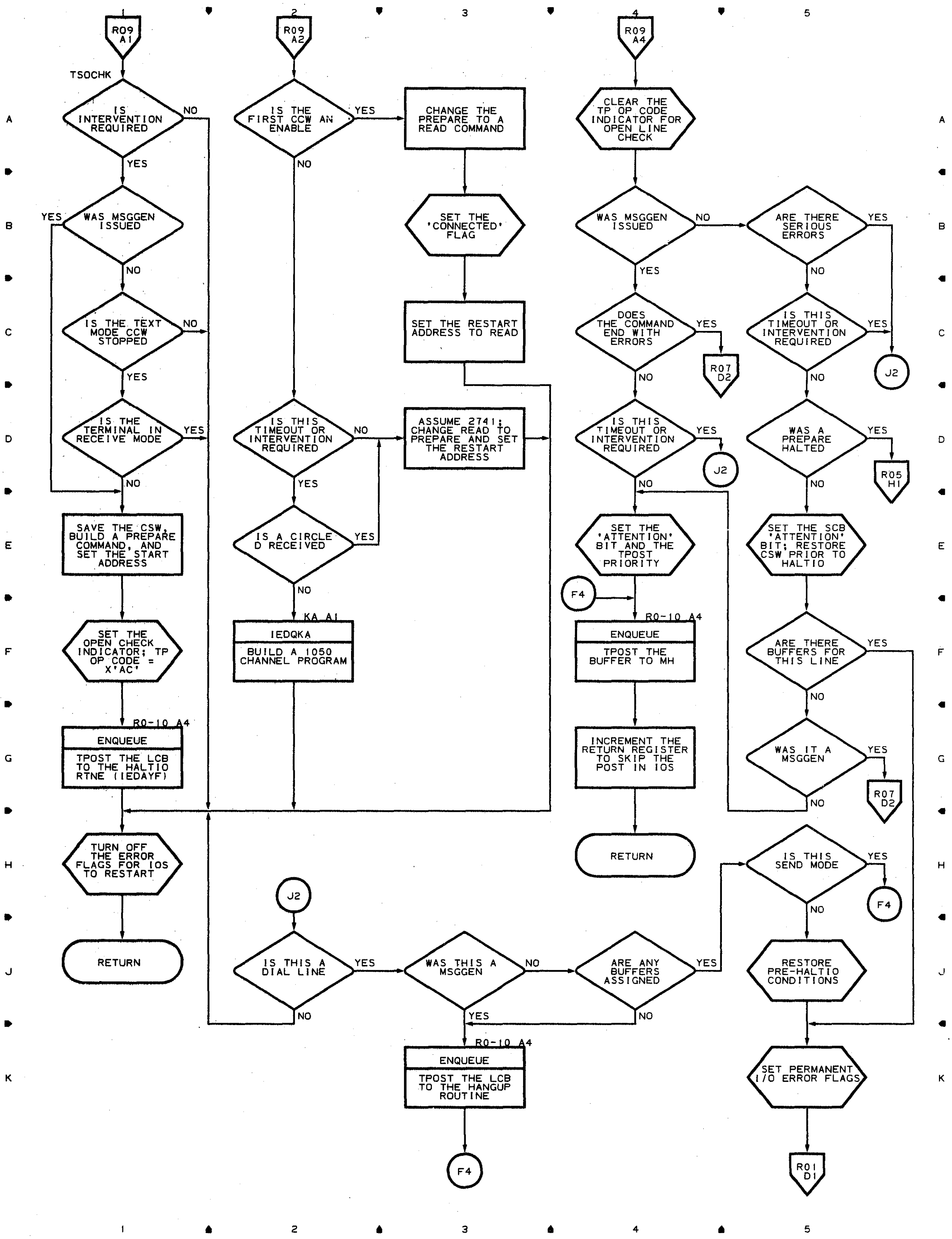
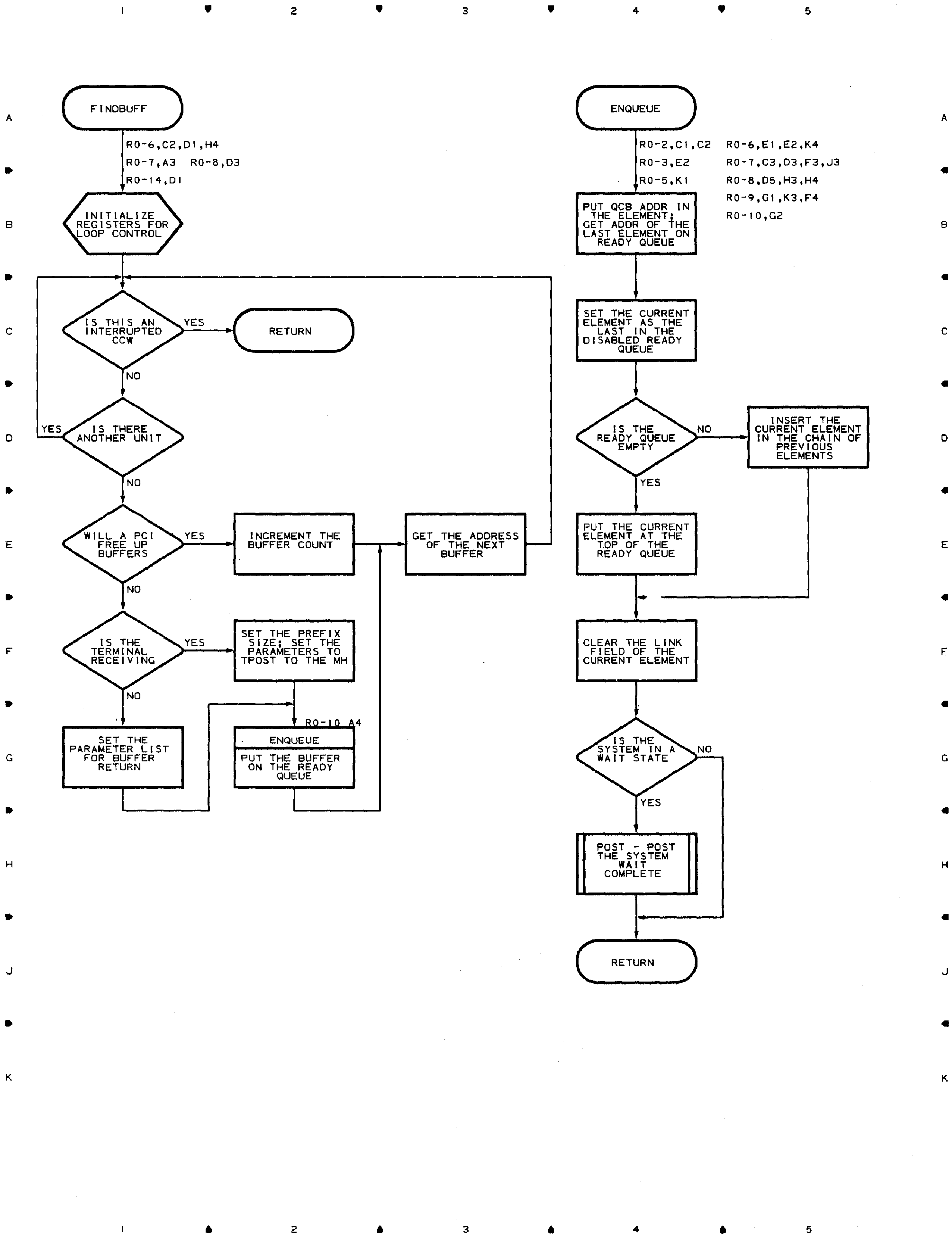

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
R08 A1
WAS PREVIOUS RESPONSE AN ERROR
YES
START ON THE INITIAL CONTACT SEQUENCE
H5
NO
IS THE LAST CCW A TIC
YES
ADJUST THE CCW TO READ/WRITE
NO
IS AUTO POLL IN EFFECT
YES
COMPUTE THE SOURCE OF THE RESPONSE FOR AUTO POLL
NO
IS THIS A BSC LINE
NO
YES
IS THIS A READ COMMAND
NO
R0-11 A4
BSC270X
ADJUST THE WRITE CCW
YES
IS THERE AN ERROR
YES
NO
R0-12 A1
SCAN
CHECK THE LINE CONTROL
DID A DIAL LINE HANG UP
YES
FORCE A DISABLE ON THE NEXT OPERATION
NO
WAS AN ENQ RECEIVED
NO
WAS A TTD RECEIVED
NO
YES
YES
BUILD THE PREVIOUS ACK TO START ON
BUILD A WRITE ACK FOR RESTART
H5
ADJUST THE REGISTERS; FIND THE TRUE FAILING CCW
BUILD THE PARAMETER LIST FOR FINDBUFF
R0-10 A1
FINDBUFF
GET THE CURRENT BUFFER
ADJUST THE ERB BUFFER COUNT
WAS AN EOT SENT
NO
YES
TPOST THE LAST BUFFER TO THE BUFFER DISPOSITION QCB
R0-10 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
IS THE TERMINAL RECEIVING
NO
YES
WAS AN EOT RECEIVED
YES
NO
GET THE OFFSET INTO THE BUFFER; SET THE CURRENT CCW FOR CONTINUE
LCOUT
NO
YES
ADJUST THE CCW TO READ OVER THE EOB
IS AN EXIT TO THE MH REQUESTED
NO
YES
R0-10 A4
ENQUEUE
TPOST THE BUFFER TO THE MH
SET UP THE CURRENT CCW FOR CONTINUE; SET THE OFFSETS FOR RECALL
G5
SET UP TO TPOST THE BUFFER TO THE MH AS THE LAST BUFFER
R0-10 A4
ENQUEUE
TPOST THE BUFFER
ADJUST THE RETURN REGISTER SO THAT THE ECB IS NOT POSTED
G5
KA-1 A1
IEDQKA
BUILD A CONTINUE SEQUENCE
H5
TURN OFF THE ERROR FLAGS TO RESTART
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN

Chart R0-9 LINE END APPENDAGE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
R09 A1
TSOCHK
IS INTERVENTION REQUIRED
NO
YES
WAS MSGGEN ISSUED
YES
NO
IS THE TEXT MODE CCW STOPPED
NO
YES
IS THE TERMINAL IN RECEIVE MODE
YES
NO
SAVE THE CSW, BUILD A PREPARE COMMAND, AND SET THE START ADDRESS
SET THE OPEN CHECK INDICATOR; TP OP CODE = X'AC'
R0-10 A4
ENQUEUE
TPOST THE LCB TO THE HALTIO RTNE (IEDAYF)
TURN OFF THE ERROR FLAGS FOR IOS TO RESTART
RETURN
R09 A2
IS THE FIRST CCW AN ENABLE
YES
NO
IS THIS TIMEOUT OR INTERVENTION REQUIRED
NO
YES
IS A CIRCLE D RECEIVED
YES
NO
KA A1
IEDQKA
BUILD A 1050 CHANNEL PROGRAM
J2
IS THIS A DIAL LINE
YES
NO
CHANGE THE PREPARE TO A READ COMMAND
SET THE 'CONNECTED' FLAG
SET THE RESTART ADDRESS TO READ
ASSUME 2741; CHANGE READ TO PREPARE AND SET THE RESTART ADDRESS
WAS THIS A MSGGEN
NO
YES
R0-10 A4
ENQUEUE
TPOST THE LCB TO THE HANGUP ROUTINE
F4
R09 A4
CLEAR THE TP OP CODE INDICATOR FOR OPEN LINE CHECK
WAS MSGGEN ISSUED
NO
YES
DOES THE COMMAND END WITH ERRORS
YES
R07 D2
NO
IS THIS TIMEOUT OR INTERVENTION REQUIRED
YES
J2
NO
SET THE 'ATTENTION' BIT AND THE TPOST PRIORITY
F4
R0-10 A4
ENQUEUE
TPOST THE BUFFER TO MH
INCREMENT THE RETURN REGISTER TO SKIP THE POST IN IOS
RETURN
ARE ANY BUFFERS ASSIGNED
YES
NO
ARE THERE SERIOUS ERRORS
YES
NO
IS THIS TIMEOUT OR INTERVENTION REQUIRED
YES
J2
NO
WAS A PREPARE HALTED
YES
R05 H1
NO
SET THE SCB 'ATTENTION' BIT; RESTORE CSW PRIOR TO HALTIO
ARE THERE BUFFERS FOR THIS LINE
YES
NO
WAS IT A MSGGEN
YES
R07 D2
NO
IS THIS SEND MODE
YES
F4
NO
RESTORE PRE-HALTIO CONDITIONS
SET PERMANENT I/O ERROR FLAGS
R01 D1

Chart R0-10 LINE END APPENDAGE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
FINDBUFF
R0-6,C2,D1,H4
R0-7,A3 R0-8,D3
R0-14,D1
INITIALIZE REGISTERS FOR LOOP CONTROL
IS THIS AN INTERRUPTED CCW
YES
RETURN
NO
YES
IS THERE ANOTHER UNIT
NO
WILL A PCI FREE UP BUFFERS
YES
INCREMENT THE BUFFER COUNT
GET THE ADDRESS OF THE NEXT BUFFER
NO
IS THE TERMINAL RECEIVING
YES
SET THE PREFIX SIZE; SET THE PARAMETERS TO TPOST TO THE MH
NO
SET THE PARAMETER LIST FOR BUFFER RETURN
R0-10 A4
ENQUEUE
PUT THE BUFFER ON THE READY QUEUE
ENQUEUE
R0-2,C1,C2
R0-3,E2
R0-5,K1
R0-6,E1,E2,K4
R0-7,C3,D3,F3,J3
R0-8,D5,H3,H4
R0-9,G1,K3,F4
R0-10,G2
PUT QCB ADDR IN THE ELEMENT; GET ADDR OF THE LAST ELEMENT ON READY QUEUE
SET THE CURRENT ELEMENT AS THE LAST IN THE DISABLED READY QUEUE
IS THE READY QUEUE EMPTY
NO
INSERT THE CURRENT ELEMENT IN THE CHAIN OF PREVIOUS ELEMENTS
YES
PUT THE CURRENT ELEMENT AT THE TOP OF THE READY QUEUE
CLEAR THE LINK FIELD OF THE CURRENT ELEMENT
IS THE SYSTEM IN A WAIT STATE
NO
YES
POST - POST THE SYSTEM WAIT COMPLETE
RETURN

Chart R0-11 LINE END APPENDAGE

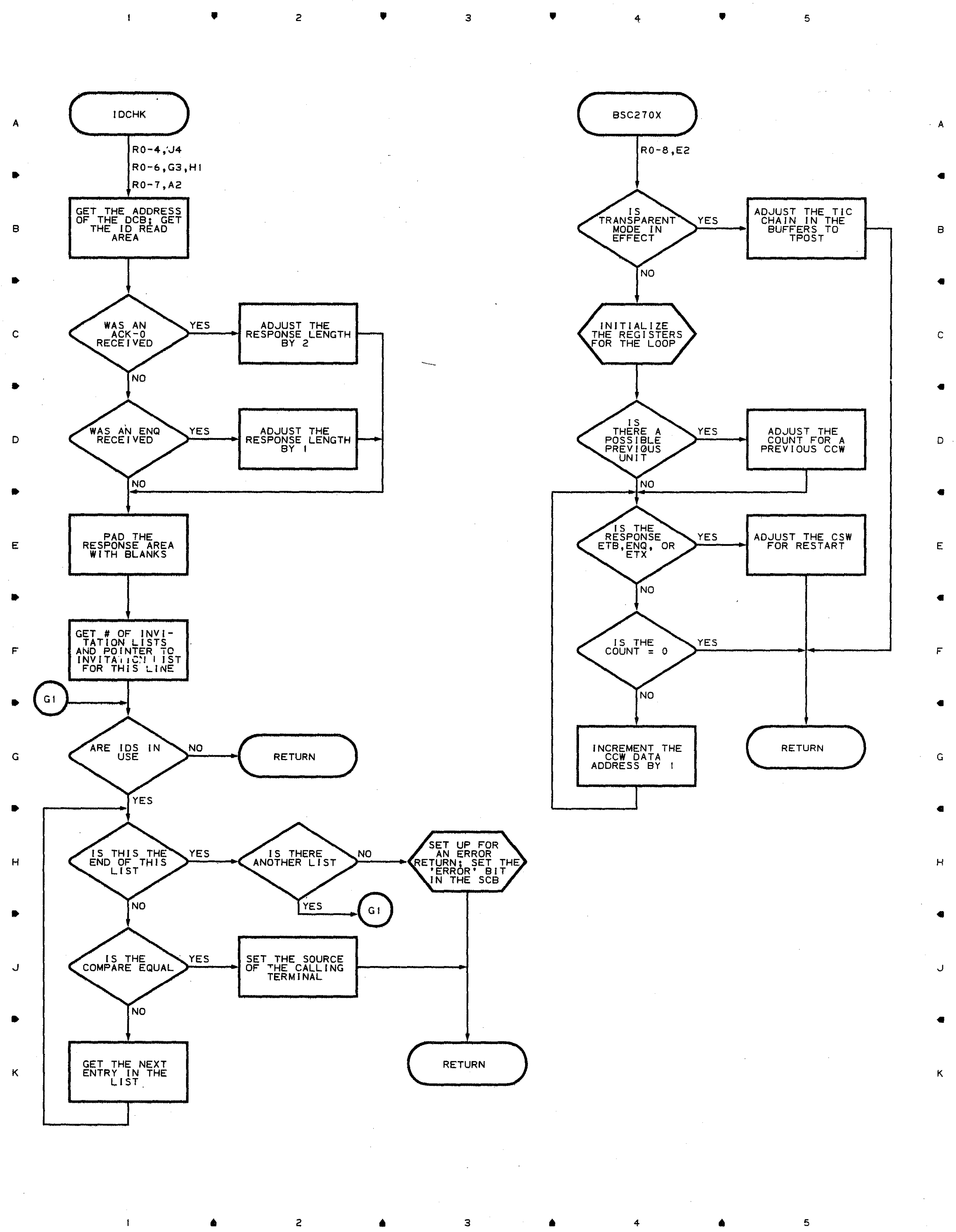

IDCHK
R0-4,J4
R0-6,G3,H1
R0-7,A2
GET THE ADDRESS OF THE DCB; GET THE ID READ AREA
WAS AN ACK-0 RECEIVED
YES
ADJUST THE RESPONSE LENGTH BY 2
NO
WAS AN ENQ RECEIVED
YES
ADJUST THE RESPONSE LENGTH BY 1
NO
PAD THE RESPONSE AREA WITH BLANKS
GET # OF INVITATION LISTS AND POINTER TO INVITATION LIST FOR THIS LINE
G1
ARE IDS IN USE
NO
RETURN
YES
IS THIS THE END OF THIS LIST
YES
IS THERE ANOTHER LIST
NO
SET UP FOR AN ERROR RETURN; SET THE 'ERROR' BIT IN THE SCB
YES
G1
NO
IS THE COMPARE EQUAL
YES
SET THE SOURCE OF THE CALLING TERMINAL
NO
GET THE NEXT ENTRY IN THE LIST
RETURN
BSC270X
R0-8,E2
IS TRANSPARENT MODE IN EFFECT
YES
ADJUST THE TIC CHAIN IN THE BUFFERS TO TPOST
NO
INITIALIZE THE REGISTERS FOR THE LOOP
IS THERE A POSSIBLE PREVIOUS UNIT
YES
ADJUST THE COUNT FOR A PREVIOUS CCW
NO
IS THE RESPONSE ETB,ENQ, OR ETX
YES
ADJUST THE CSW FOR RESTART
NO
IS THE COUNT = 0
YES
NO
INCREMENT THE CCW DATA ADDRESS BY 1
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart R0-12 LINE END APPENDAGE

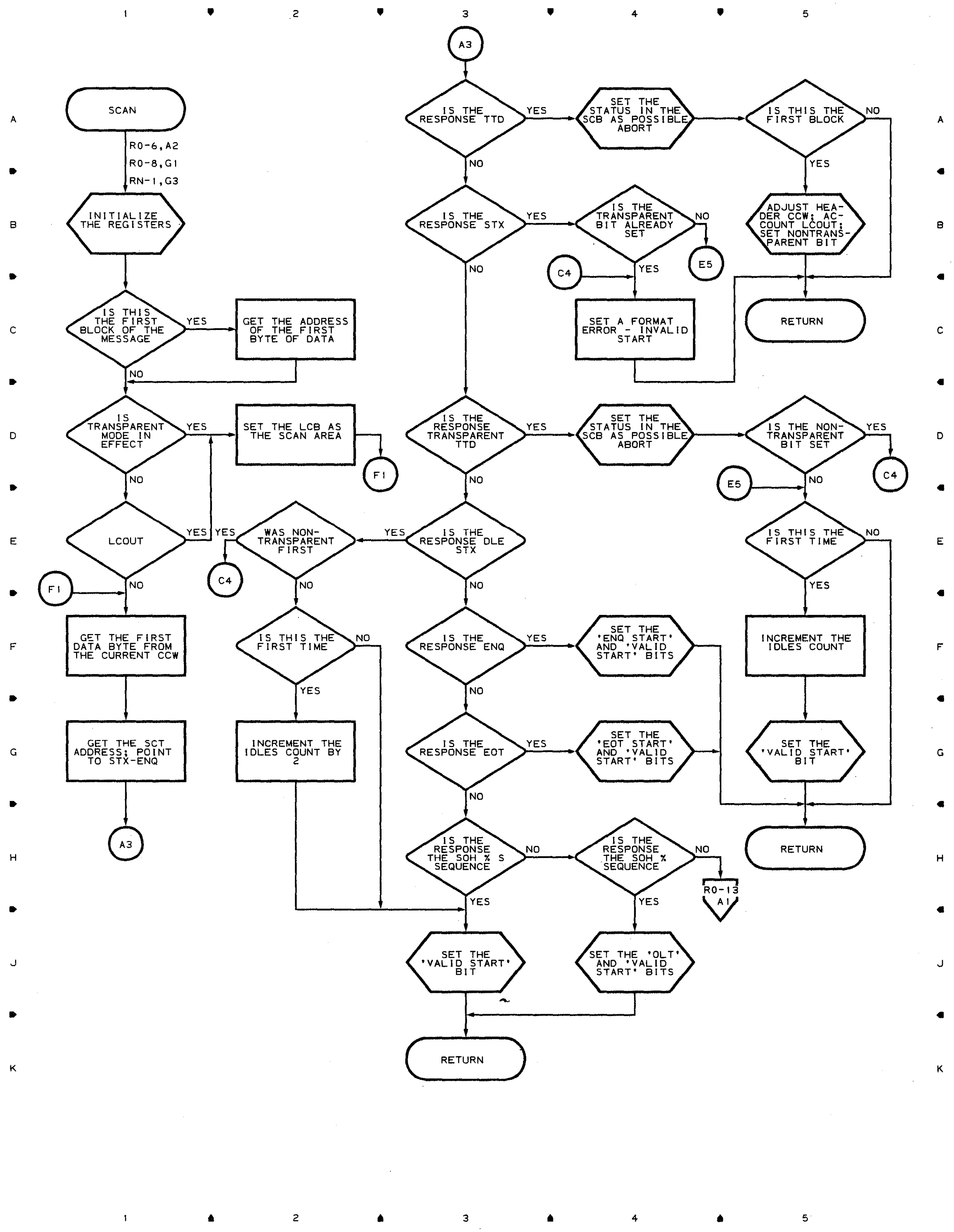

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
A3
SCAN
R0-6,A2
R0-8,G1
RN-1,G3
INITIALIZE THE REGISTERS
IS THIS THE FIRST BLOCK OF THE MESSAGE
YES
GET THE ADDRESS OF THE FIRST BYTE OF DATA
NO
IS TRANSPARENT MODE IN EFFECT
YES
SET THE LCB AS THE SCAN AREA
F1
NO
LCOUT
YES
YES
WAS NON-TRANSPARENT FIRST
C4
F1
NO
NO
GET THE FIRST DATA BYTE FROM THE CURRENT CCW
GET THE SCT ADDRESS; POINT TO STX-ENQ
A3
IS THIS THE FIRST TIME
NO
YES
INCREMENT THE IDLES COUNT BY 2
IS THE RESPONSE TTD
YES
SET THE STATUS IN THE SCB AS POSSIBLE ABORT
IS THIS THE FIRST BLOCK
NO
YES
ADJUST HEADER CCW; ACCOUNT LCOUT; SET NONTRANSPARENT BIT
RETURN
NO
IS THE RESPONSE STX
YES
IS THE TRANSPARENT BIT ALREADY SET
NO
E5
C4
YES
SET A FORMAT ERROR - INVALID START
NO
IS THE RESPONSE TRANSPARENT TTD
YES
SET THE STATUS IN THE SCB AS POSSIBLE ABORT
IS THE NON-TRANSPARENT BIT SET
YES
C4
E5
NO
NO
IS THE RESPONSE DLE STX
YES
IS THIS THE FIRST TIME
NO
YES
INCREMENT THE IDLES COUNT
SET THE 'VALID START' BIT
RETURN
NO
IS THE RESPONSE ENQ
YES
SET THE 'ENQ START' AND 'VALID START' BITS
NO
IS THE RESPONSE EOT
YES
SET THE 'EOT START' AND 'VALID START' BITS
NO
IS THE RESPONSE THE SOH % S SEQUENCE
NO
IS THE RESPONSE THE SOH % SEQUENCE
NO
R0-13 A1
YES
YES
SET THE 'VALID START' BIT
SET THE 'OLT' AND 'VALID START' BITS
RETURN

Chart R0-13 LINE END APPENDAGE

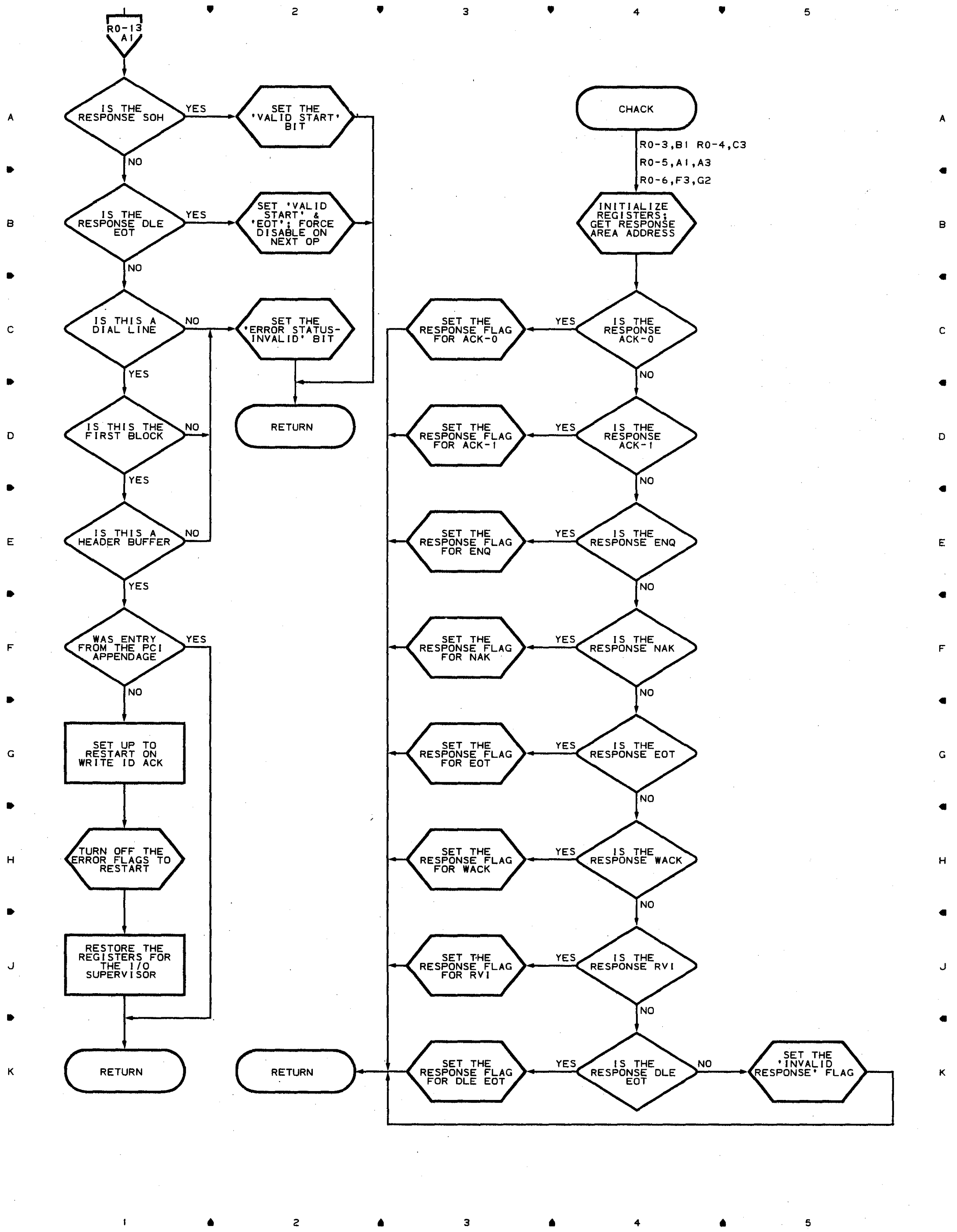

1
2
3
4
5
R0-13 A1
A
B
C
D
E
F
G
H
J
K
IS THE RESPONSE SOH
YES
SET THE 'VALID START' BIT
NO
IS THE RESPONSE DLE EOT
YES
SET 'VALID START' & 'EOT'; FORCE DISABLE ON NEXT OP
NO
IS THIS A DIAL LINE
NO
SET THE 'ERROR STATUS-INVALID' BIT
YES
IS THIS THE FIRST BLOCK
NO
RETURN
YES
IS THIS A HEADER BUFFER
NO
YES
WAS ENTRY FROM THE PCI APPENDAGE
YES
NO
SET UP TO RESTART ON WRITE ID ACK
TURN OFF THE ERROR FLAGS TO RESTART
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN
RETURN
CHACK
R0-3,B1 R0-4,C3
R0-5,A1,A3
R0-6,F3,G2
INITIALIZE REGISTERS; GET RESPONSE AREA ADDRESS
IS THE RESPONSE ACK-0
YES
SET THE RESPONSE FLAG FOR ACK-0
NO
IS THE RESPONSE ACK-1
YES
SET THE RESPONSE FLAG FOR ACK-1
NO
IS THE RESPONSE ENQ
YES
SET THE RESPONSE FLAG FOR ENQ
NO
IS THE RESPONSE NAK
YES
SET THE RESPONSE FLAG FOR NAK
NO
IS THE RESPONSE EOT
YES
SET THE RESPONSE FLAG FOR EOT
NO
IS THE RESPONSE WACK
YES
SET THE RESPONSE FLAG FOR WACK
NO
IS THE RESPONSE RVI
YES
SET THE RESPONSE FLAG FOR RVI
NO
IS THE RESPONSE DLE EOT
YES
SET THE RESPONSE FLAG FOR DLE EOT
NO
SET THE 'INVALID RESPONSE' FLAG

Chart R0-14 LINE END APPENDAGE

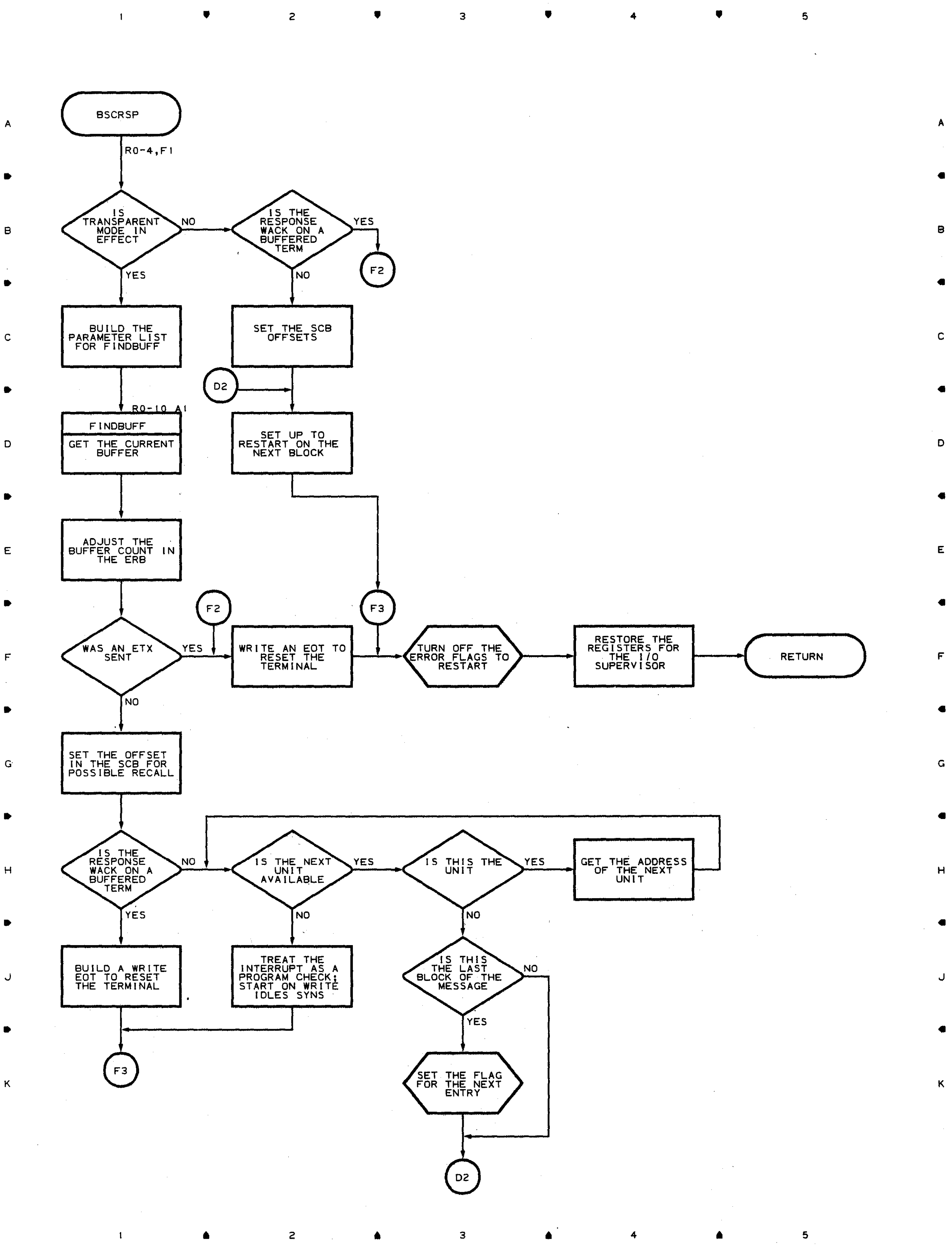

BSCRSP
R0-4,F1
IS TRANSPARENT MODE IN EFFECT
NO
YES
IS THE RESPONSE WACK ON A BUFFERED TERM
YES
NO
F2
BUILD THE PARAMETER LIST FOR FINDBUFF
SET THE SCB OFFSETS
D2
R0-10 A1
FINDBUFF
GET THE CURRENT BUFFER
SET UP TO RESTART ON THE NEXT BLOCK
ADJUST THE BUFFER COUNT IN THE ERB
F2
F3
WAS AN ETX SENT
YES
NO
WRITE AN EOT TO RESET THE TERMINAL
TURN OFF THE ERROR FLAGS TO RESTART
RESTORE THE REGISTERS FOR THE I/O SUPERVISOR
RETURN
SET THE OFFSET IN THE SCB FOR POSSIBLE RECALL
IS THE RESPONSE WACK ON A BUFFERED TERM
NO
YES
IS THE NEXT UNIT AVAILABLE
YES
NO
IS THIS THE UNIT
YES
NO
GET THE ADDRESS OF THE NEXT UNIT
BUILD A WRITE EOT TO RESET THE TERMINAL
TREAT THE INTERRUPT AS A PROGRAM CHECK; START ON WRITE IDLES SYNS
IS THIS THE LAST BLOCK OF THE MESSAGE
NO
YES
F3
SET THE FLAG FOR THE NEXT ENTRY
D2
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart R1-1 DIAL RECEIVE SCHEDULER

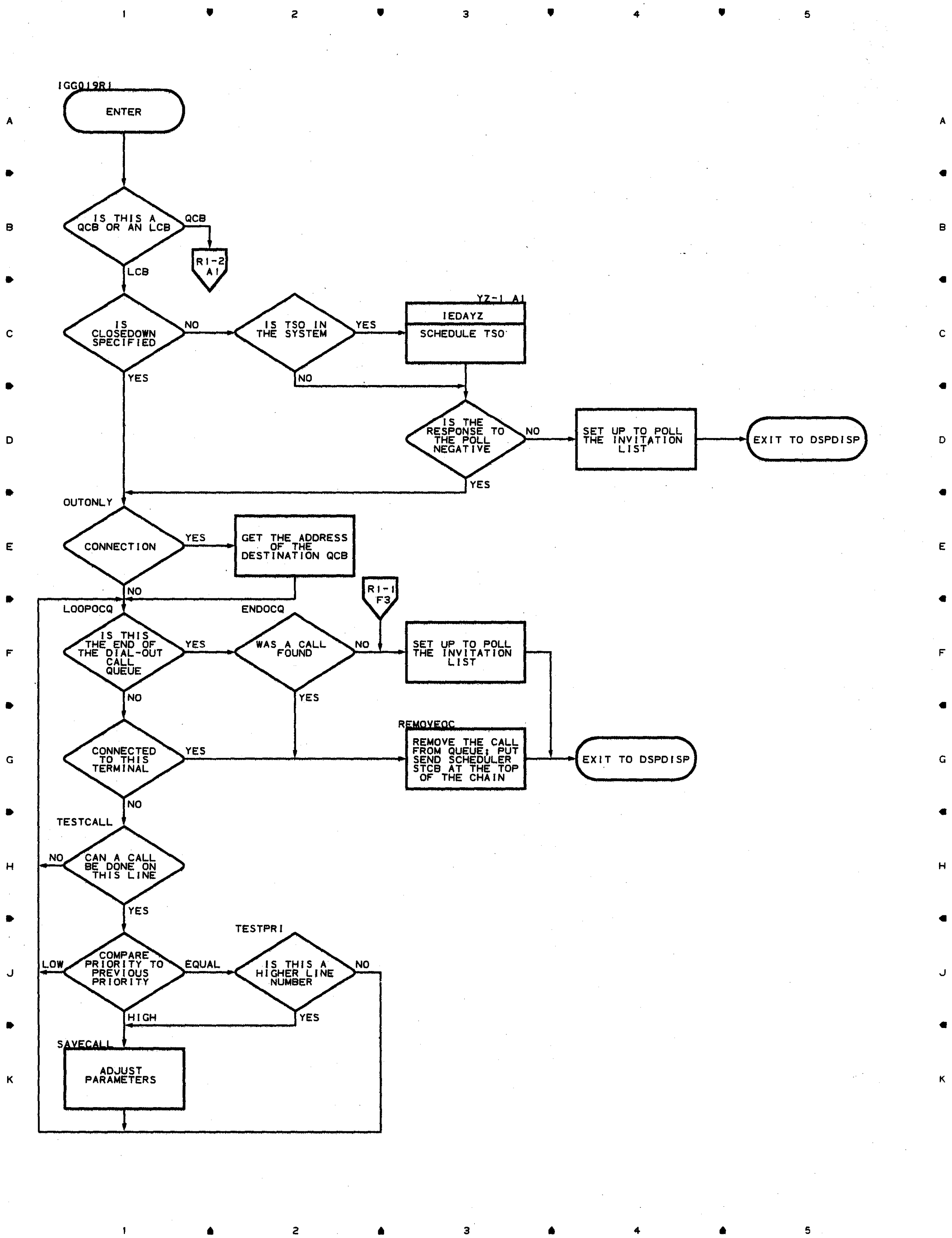

IGG019R1
ENTER
IS THIS A QCB OR AN LCB
QCB
R1-2 A1
LCB
IS CLOSEDOWN SPECIFIED
NO
IS TSO IN THE SYSTEM
YES
YZ-1 A1
IEDAYZ
SCHEDULE TSO
NO
YES
IS THE RESPONSE TO THE POLL NEGATIVE
NO
SET UP TO POLL THE INVITATION LIST
EXIT TO DSPDISP
YES
OUTONLY
CONNECTION
YES
GET THE ADDRESS OF THE DESTINATION QCB
NO
R1-1 F3
LOOPOCQ
IS THIS THE END OF THE DIAL-OUT CALL QUEUE
YES
ENDOCQ
WAS A CALL FOUND
NO
SET UP TO POLL THE INVITATION LIST
NO
YES
REMOVEOC
CONNECTED TO THIS TERMINAL
YES
REMOVE THE CALL FROM QUEUE; PUT SEND SCHEDULER STCB AT THE TOP OF THE CHAIN
EXIT TO DSPDISP
NO
TESTCALL
NO
CAN A CALL BE DONE ON THIS LINE
YES
TESTPRI
LOW
COMPARE PRIORITY TO PREVIOUS PRIORITY
EQUAL
IS THIS A HIGHER LINE NUMBER
NO
HIGH
YES
SAVECALL
ADJUST PARAMETERS
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart R1-2 DIAL RECEIVE SCHEDULER

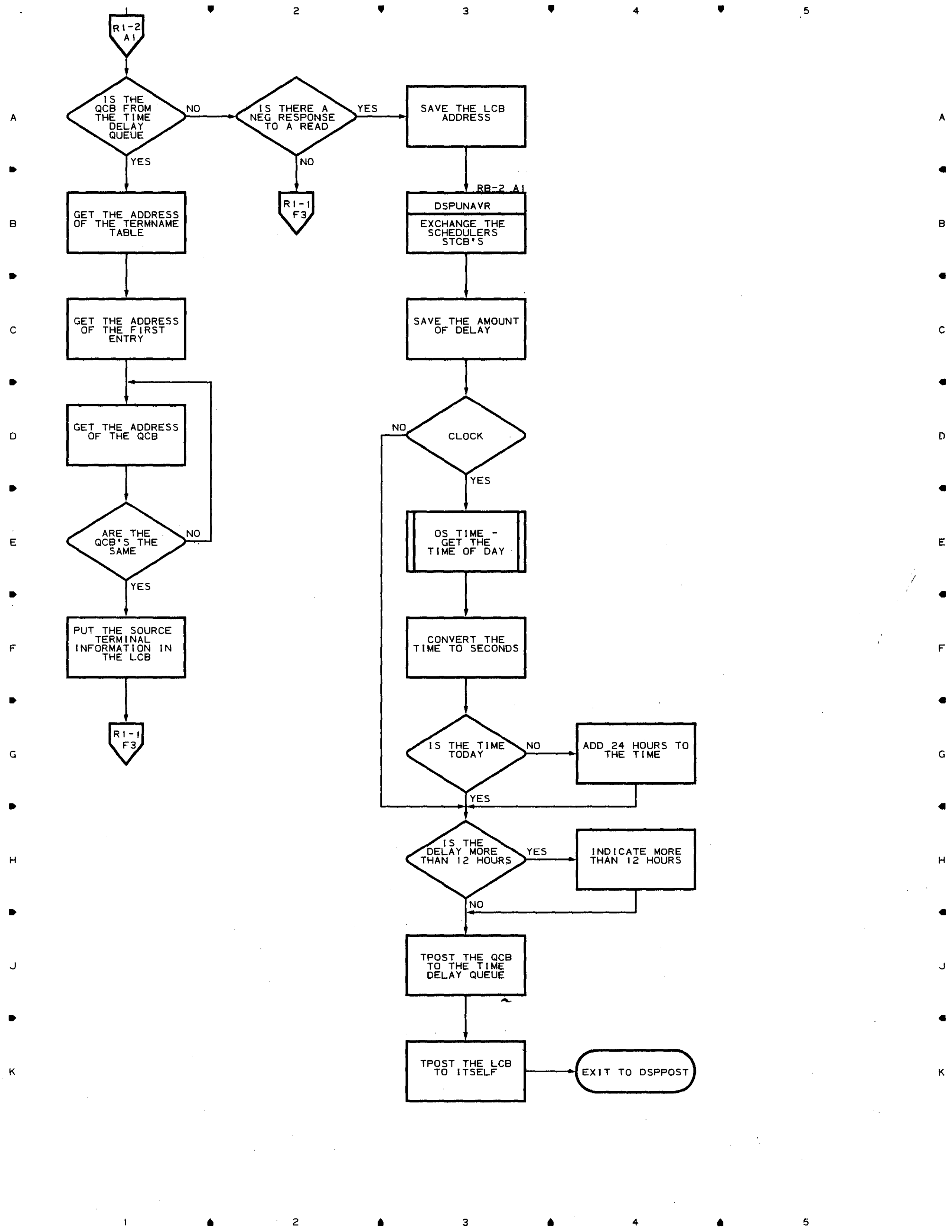

R1-2 A1
IS THE QCB FROM THE TIME DELAY QUEUE
NO
YES
IS THERE A NEG RESPONSE TO A READ
YES
NO
R1-1 F3
SAVE THE LCB ADDRESS
RB-2 A1
DSPUNAVR
EXCHANGE THE SCHEDULERS STCB'S
GET THE ADDRESS OF THE TERMNAME TABLE
GET THE ADDRESS OF THE FIRST ENTRY
SAVE THE AMOUNT OF DELAY
GET THE ADDRESS OF THE QCB
NO
CLOCK
YES
ARE THE QCB'S THE SAME
NO
YES
OS TIME - GET THE TIME OF DAY
PUT THE SOURCE TERMINAL INFORMATION IN THE LCB
CONVERT THE TIME TO SECONDS
R1-1 F3
IS THE TIME TODAY
NO
ADD 24 HOURS TO THE TIME
YES
IS THE DELAY MORE THAN 12 HOURS
YES
INDICATE MORE THAN 12 HOURS
NO
TPOST THE QCB TO THE TIME DELAY QUEUE
TPOST THE LCB TO ITSELF
EXIT TO DSPPOST
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart R2 DISK END APPENDAGE

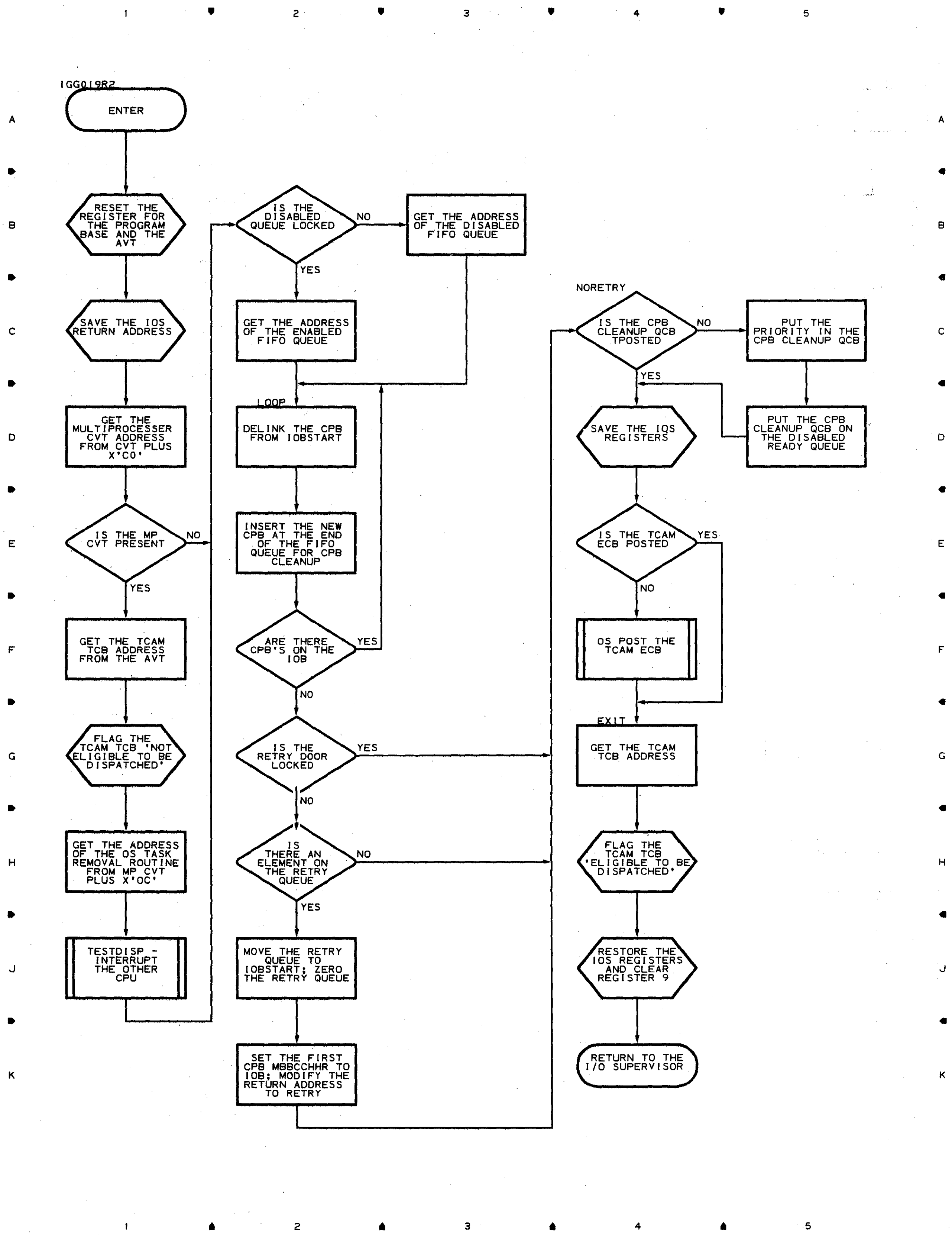

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG019R2
ENTER
RESET THE REGISTER FOR THE PROGRAM BASE AND THE AVT
SAVE THE IOS RETURN ADDRESS
GET THE MULTIPROCESSER CVT ADDRESS FROM CVT PLUS X'C0'
IS THE MP CVT PRESENT
NO
YES
GET THE TCAM TCB ADDRESS FROM THE AVT
FLAG THE TCAM TCB 'NOT ELIGIBLE TO BE DISPATCHED'
GET THE ADDRESS OF THE OS TASK REMOVAL ROUTINE FROM MP CVT PLUS X'0C'
TESTDISP - INTERRUPT THE OTHER CPU
IS THE DISABLED QUEUE LOCKED
NO
YES
GET THE ADDRESS OF THE DISABLED FIFO QUEUE
GET THE ADDRESS OF THE ENABLED FIFO QUEUE
LOOP
DELINK THE CPB FROM IOBSTART
INSERT THE NEW CPB AT THE END OF THE FIFO QUEUE FOR CPB CLEANUP
ARE THERE CPB'S ON THE IOB
YES
NO
IS THE RETRY DOOR LOCKED
YES
NO
IS THERE AN ELEMENT ON THE RETRY QUEUE
NO
YES
MOVE THE RETRY QUEUE TO IOBSTART; ZERO THE RETRY QUEUE
SET THE FIRST CPB MBBCCHHR TO IOB; MODIFY THE RETURN ADDRESS TO RETRY
NORETRY
IS THE CPB CLEANUP QCB TPOSTED
NO
YES
PUT THE PRIORITY IN THE CPB CLEANUP QCB
PUT THE CPB CLEANUP QCB ON THE DISABLED READY QUEUE
SAVE THE IOS REGISTERS
IS THE TCAM ECB POSTED
YES
NO
OS POST THE TCAM ECB
EXIT
GET THE TCAM TCB ADDRESS
FLAG THE TCAM TCB 'ELIGIBLE TO BE DISPATCHED'
RESTORE THE IOS REGISTERS AND CLEAR REGISTER 9
RETURN TO THE I/O SUPERVISOR

Chart R3 LEASED RECEIVE SCHEDULER

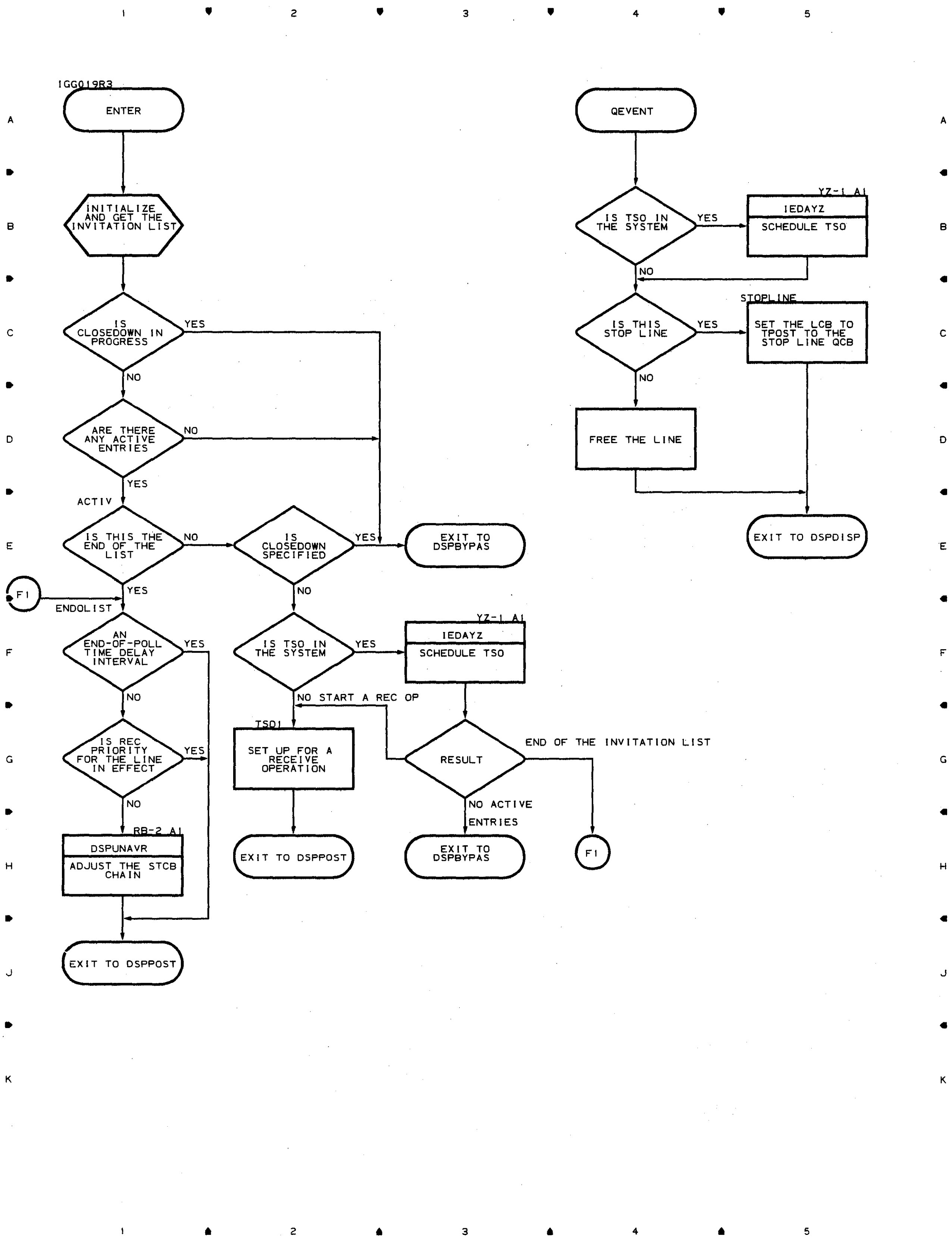


Chart R4-1 SEND SCHEDULER

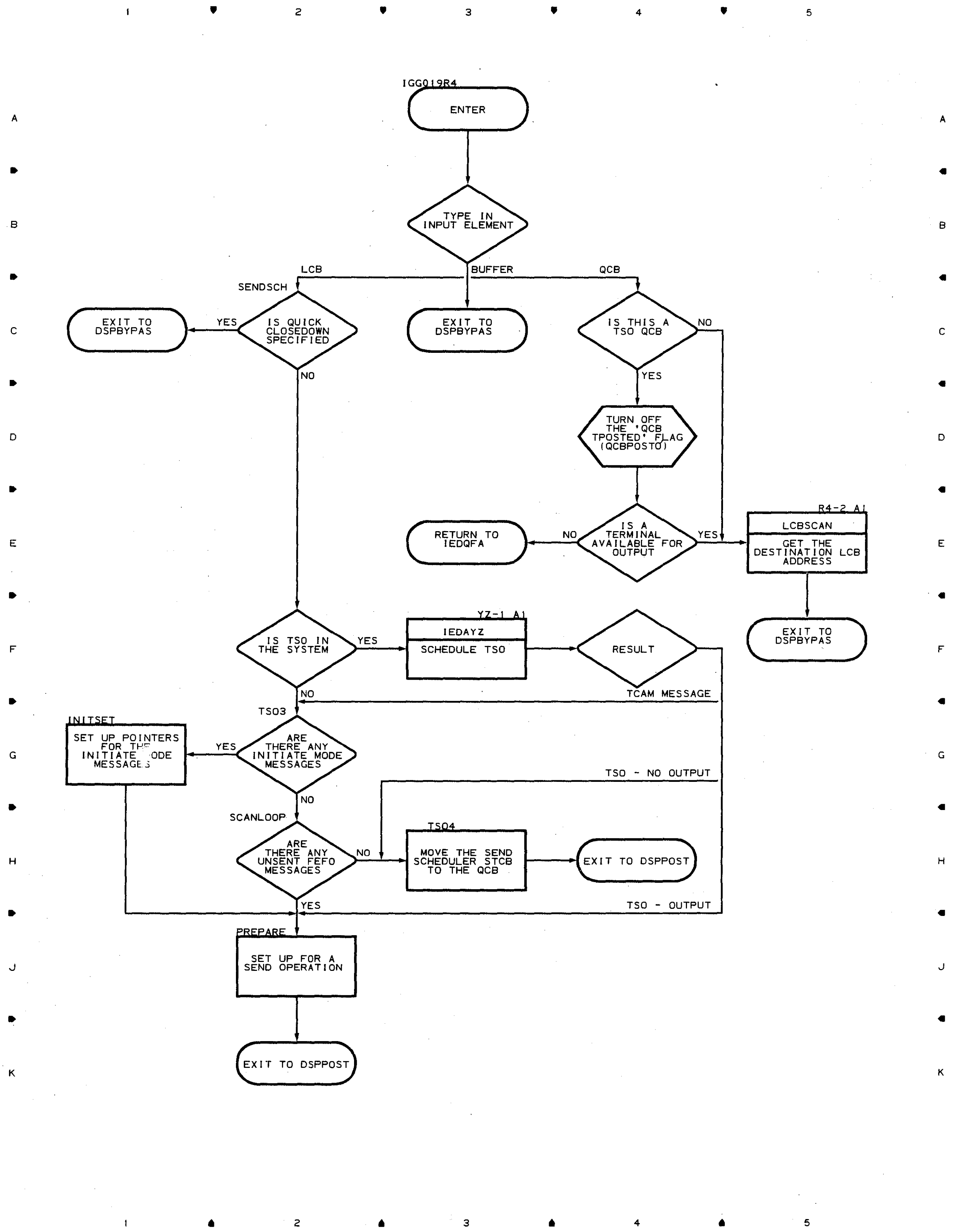


Chart R4-2 SEND SCHEDULER

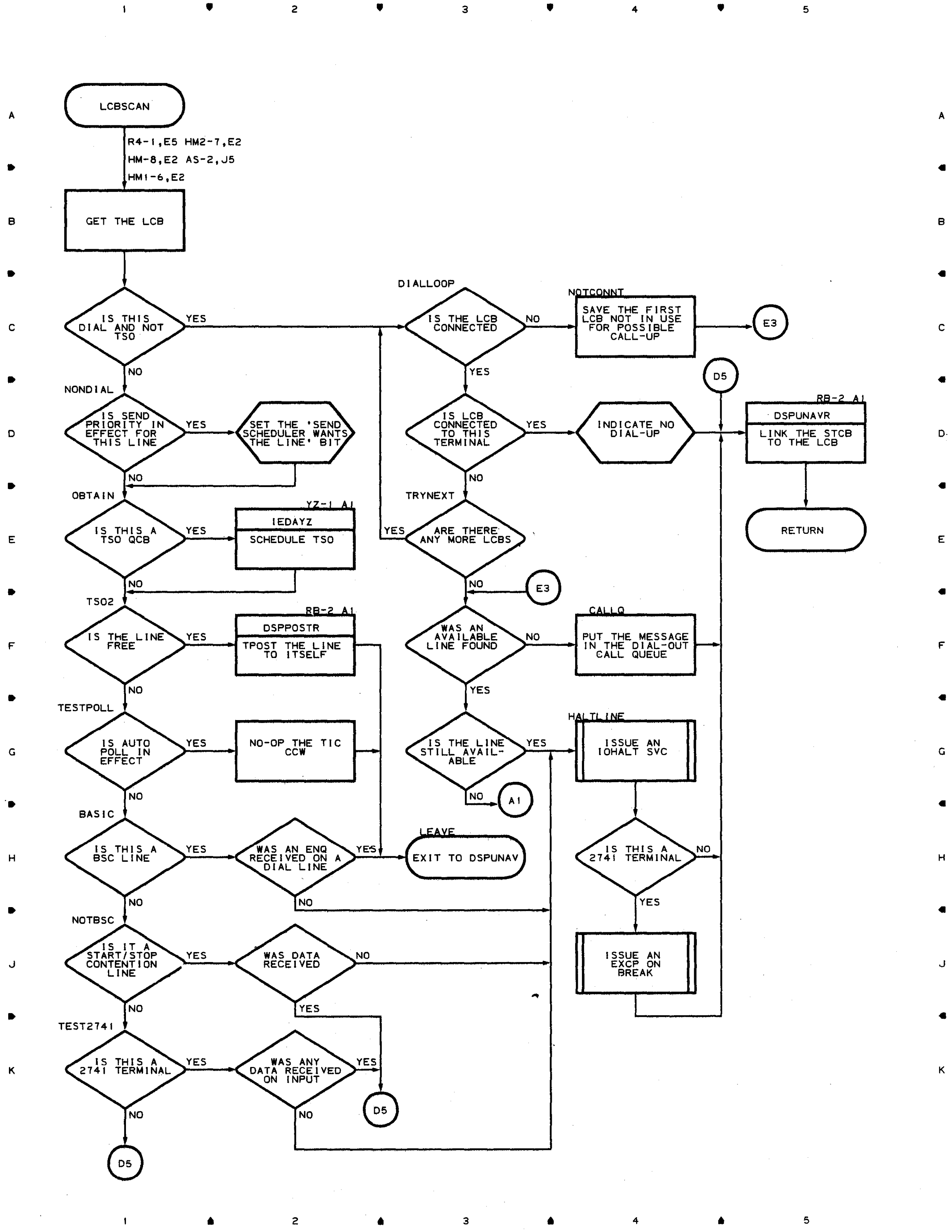


Chart R5 ATTENTION HANDLING ROUTINE

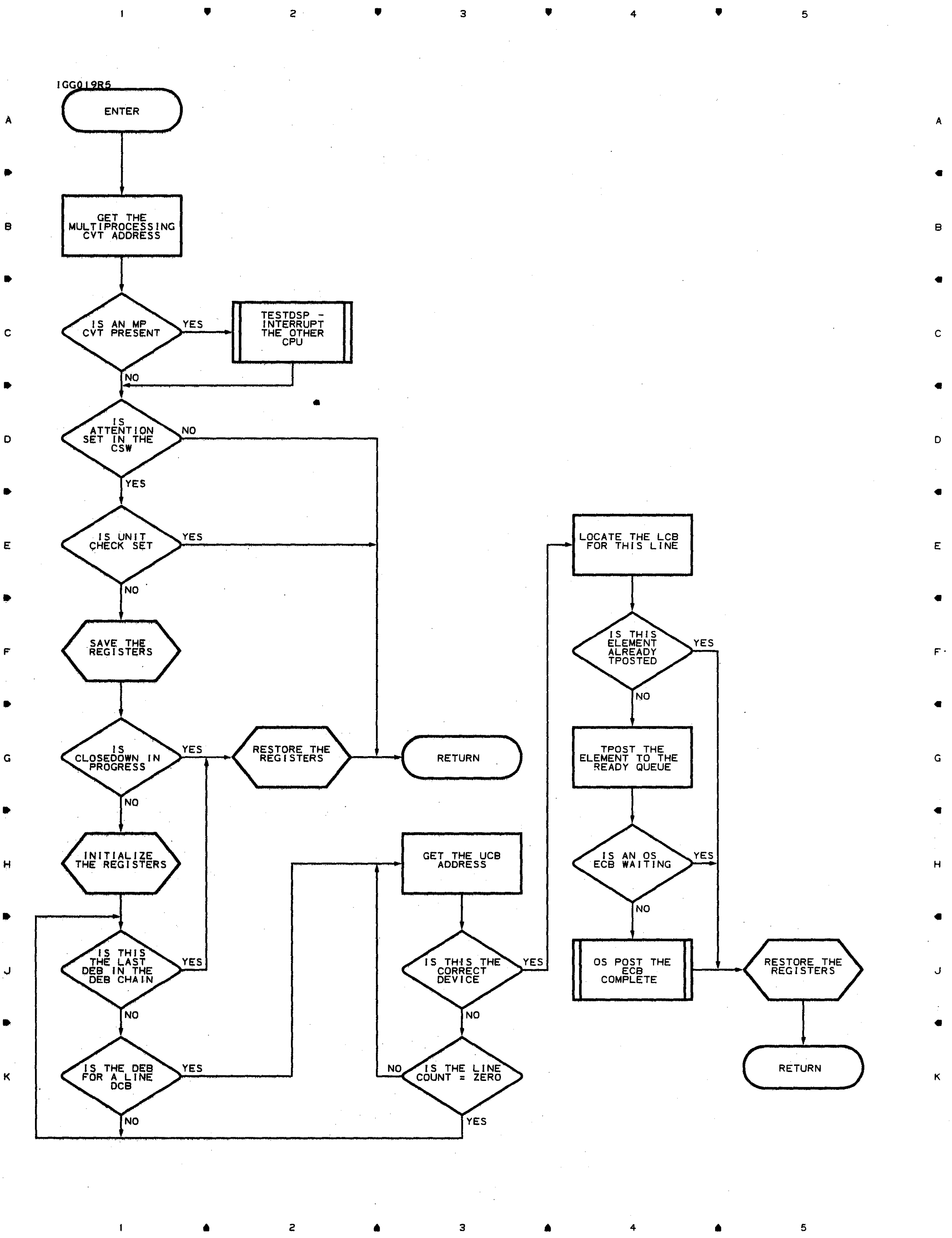


Chart R6-1 START-UP MESSAGE ROUTINE

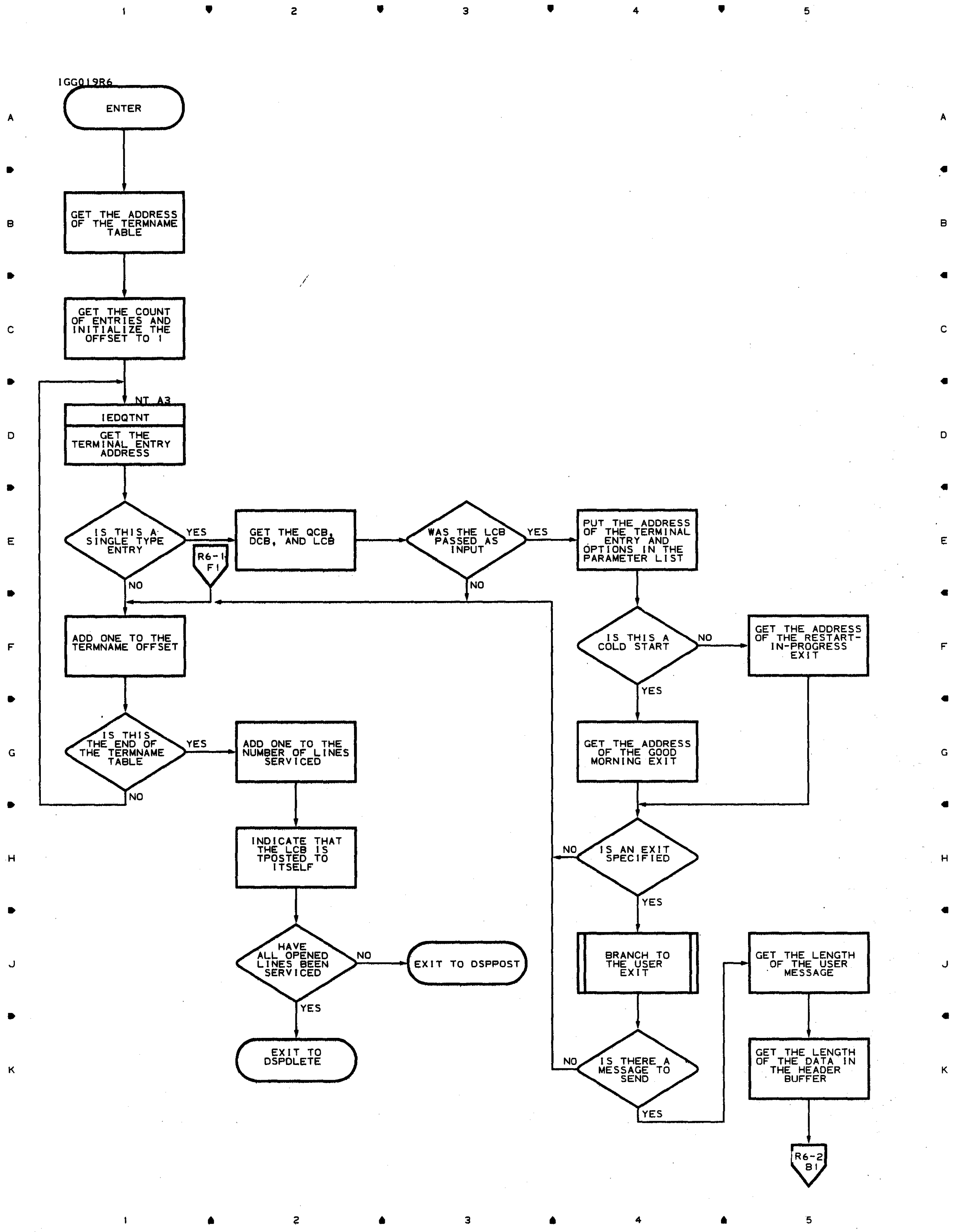

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGG019R6
ENTER
GET THE ADDRESS OF THE TERMNAME TABLE
GET THE COUNT OF ENTRIES AND INITIALIZE THE OFFSET TO 1
NT A3
IEDQTNT
GET THE TERMINAL ENTRY ADDRESS
IS THIS A SINGLE TYPE ENTRY
YES
NO
R6-1 F1
GET THE QCB, DCB, AND LCB
WAS THE LCB PASSED AS INPUT
YES
NO
PUT THE ADDRESS OF THE TERMINAL ENTRY AND OPTIONS IN THE PARAMETER LIST
ADD ONE TO THE TERMNAME OFFSET
IS THIS A COLD START
NO
YES
GET THE ADDRESS OF THE RESTART-IN-PROGRESS EXIT
IS THIS THE END OF THE TERMNAME TABLE
YES
NO
ADD ONE TO THE NUMBER OF LINES SERVICED
GET THE ADDRESS OF THE GOOD MORNING EXIT
INDICATE THAT THE LCB IS TPOSTED TO ITSELF
IS AN EXIT SPECIFIED
NO
YES
HAVE ALL OPENED LINES BEEN SERVICED
NO
YES
EXIT TO DSPPOST
BRANCH TO THE USER EXIT
GET THE LENGTH OF THE USER MESSAGE
EXIT TO DSPDLETE
IS THERE A MESSAGE TO SEND
NO
YES
GET THE LENGTH OF THE DATA IN THE HEADER BUFFER
R6-2 B1

Chart R6-2 START-UP MESSAGE ROUTINE

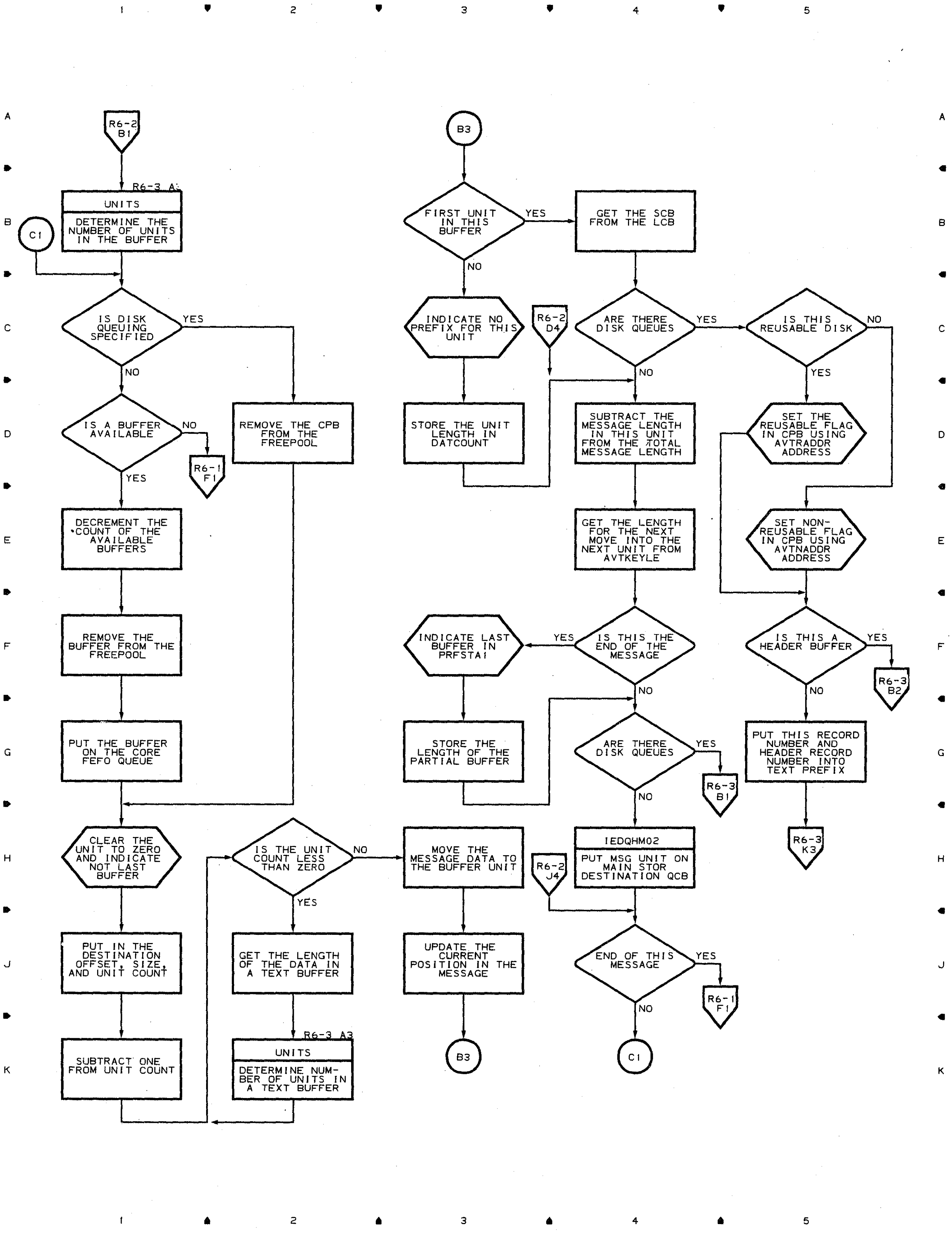
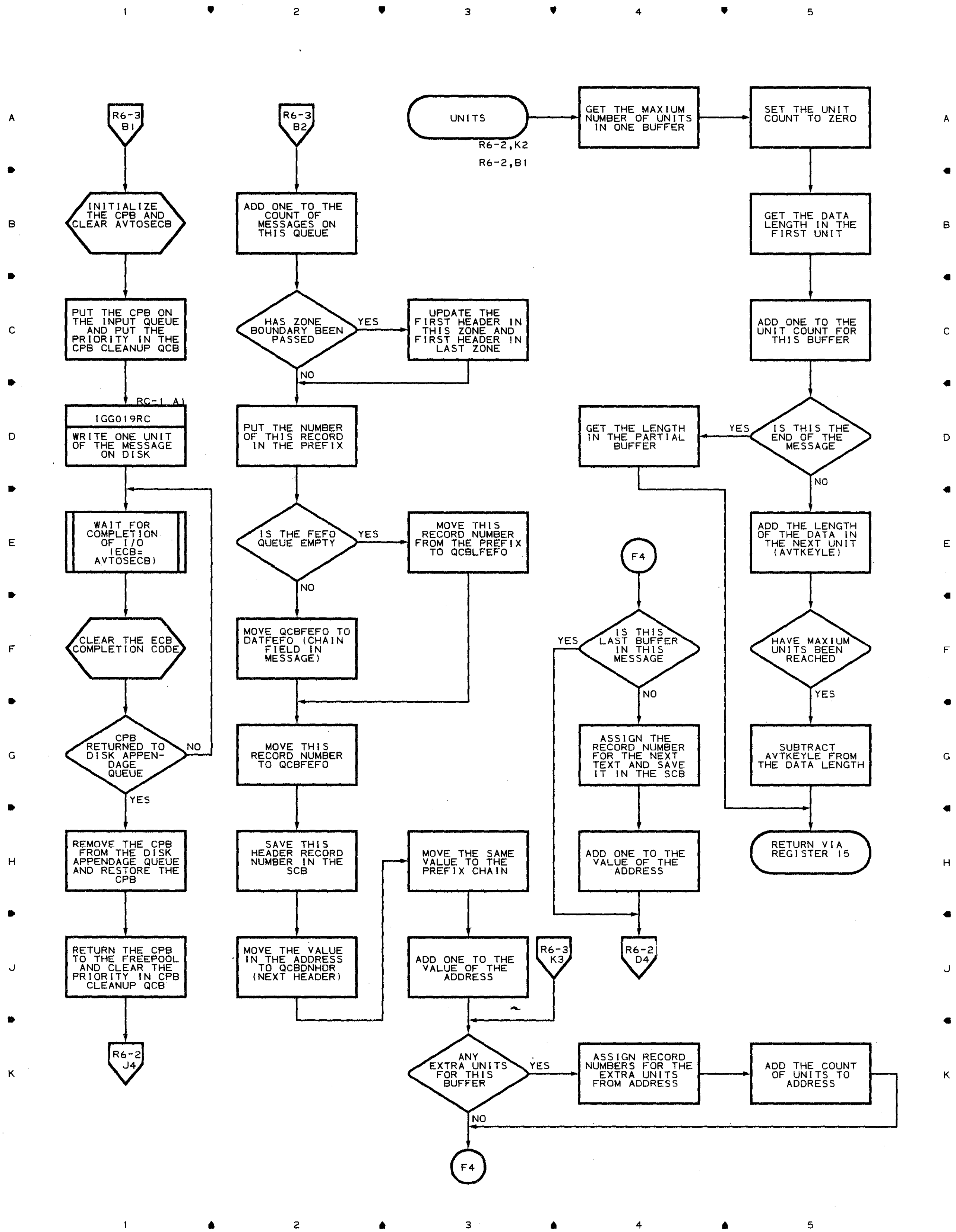

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
R6-2 B1
R6-3 A1
UNITS
DETERMINE THE NUMBER OF UNITS IN THE BUFFER
C1
IS DISK QUEUING SPECIFIED
YES
NO
IS A BUFFER AVAILABLE
NO
R6-1 F1
YES
REMOVE THE CPB FROM THE FREEPOOL
DECREMENT THE COUNT OF THE AVAILABLE BUFFERS
REMOVE THE BUFFER FROM THE FREEPOOL
PUT THE BUFFER ON THE CORE FEFO QUEUE
CLEAR THE UNIT TO ZERO AND INDICATE NOT LAST BUFFER
PUT IN THE DESTINATION OFFSET, SIZE, AND UNIT COUNT
SUBTRACT ONE FROM UNIT COUNT
IS THE UNIT COUNT LESS THAN ZERO
NO
YES
GET THE LENGTH OF THE DATA IN A TEXT BUFFER
R6-3 A3
UNITS
DETERMINE NUMBER OF UNITS IN A TEXT BUFFER
B3
FIRST UNIT IN THIS BUFFER
YES
NO
INDICATE NO PREFIX FOR THIS UNIT
STORE THE UNIT LENGTH IN DATCOUNT
R6-2 D4
GET THE SCB FROM THE LCB
ARE THERE DISK QUEUES
YES
NO
SUBTRACT THE MESSAGE LENGTH IN THIS UNIT FROM THE TOTAL MESSAGE LENGTH
GET THE LENGTH FOR THE NEXT MOVE INTO THE NEXT UNIT FROM AVTKEYLE
IS THIS THE END OF THE MESSAGE
YES
NO
INDICATE LAST BUFFER IN PRFSTA1
STORE THE LENGTH OF THE PARTIAL BUFFER
ARE THERE DISK QUEUES
YES
R6-3 B1
NO
IEDQHM02
PUT MSG UNIT ON MAIN STOR DESTINATION QCB
MOVE THE MESSAGE DATA TO THE BUFFER UNIT
R6-2 J4
UPDATE THE CURRENT POSITION IN THE MESSAGE
B3
END OF THIS MESSAGE
YES
R6-1 F1
NO
C1
IS THIS REUSABLE DISK
NO
YES
SET THE REUSABLE FLAG IN CPB USING AVTRADDR ADDRESS
SET NON-REUSABLE FLAG IN CPB USING AVTNADDR ADDRESS
IS THIS A HEADER BUFFER
YES
R6-3 B2
NO
PUT THIS RECORD NUMBER AND HEADER RECORD NUMBER INTO TEXT PREFIX
R6-3 K3

Chart R6-3 START-UP MESSAGE ROUTINE



R6-3 B1
INITIALIZE THE CPB AND CLEAR AVTOSECB
PUT THE CPB ON THE INPUT QUEUE AND PUT THE PRIORITY IN THE CPB CLEANUP QCB
RC-1 A1
IGG019RC
WRITE ONE UNIT OF THE MESSAGE ON DISK
WAIT FOR COMPLETION OF I/O (ECB= AVTOSECB)
CLEAR THE ECB COMPLETION CODE
CPB RETURNED TO DISK APPEN- DAGE QUEUE
NO
YES
REMOVE THE CPB FROM THE DISK APPENDAGE QUEUE AND RESTORE THE CPB
RETURN THE CPB TO THE FREEPOOL AND CLEAR THE PRIORITY IN CPB CLEANUP QCB
R6-2 J4
R6-3 B2
ADD ONE TO THE COUNT OF MESSAGES ON THIS QUEUE
HAS ZONE BOUNDARY BEEN PASSED
YES
NO
UPDATE THE FIRST HEADER IN THIS ZONE AND FIRST HEADER IN LAST ZONE
PUT THE NUMBER OF THIS RECORD IN THE PREFIX
IS THE FEFO QUEUE EMPTY
YES
NO
MOVE THIS RECORD NUMBER FROM THE PREFIX TO QCBLFEFO
MOVE QCBFEFO TO DATFEFO (CHAIN FIELD IN MESSAGE)
MOVE THIS RECORD NUMBER TO QCBFEFO
SAVE THIS HEADER RECORD NUMBER IN THE SCB
MOVE THE VALUE IN THE ADDRESS TO QCBDNHDR (NEXT HEADER)
UNITS
R6-2,K2
R6-2,B1
MOVE THE SAME VALUE TO THE PREFIX CHAIN
ADD ONE TO THE VALUE OF THE ADDRESS
R6-3 K3
ANY EXTRA UNITS FOR THIS BUFFER
YES
NO
F4
GET THE MAXIUM NUMBER OF UNITS IN ONE BUFFER
GET THE LENGTH IN THE PARTIAL BUFFER
YES
F4
IS THIS LAST BUFFER IN THIS MESSAGE
YES
NO
ASSIGN THE RECORD NUMBER FOR THE NEXT TEXT AND SAVE IT IN THE SCB
ADD ONE TO THE VALUE OF THE ADDRESS
R6-2 D4
ASSIGN RECORD NUMBERS FOR THE EXTRA UNITS FROM ADDRESS
SET THE UNIT COUNT TO ZERO
GET THE DATA LENGTH IN THE FIRST UNIT
ADD ONE TO THE UNIT COUNT FOR THIS BUFFER
IS THIS THE END OF THE MESSAGE
NO
ADD THE LENGTH OF THE DATA IN THE NEXT UNIT (AVTKEYLE)
HAVE MAXIUM UNITS BEEN REACHED
YES
SUBTRACT AVTKEYLE FROM THE DATA LENGTH
RETURN VIA REGISTER 15
ADD THE COUNT OF UNITS TO ADDRESS
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart TN ATTENTION ROUTINE

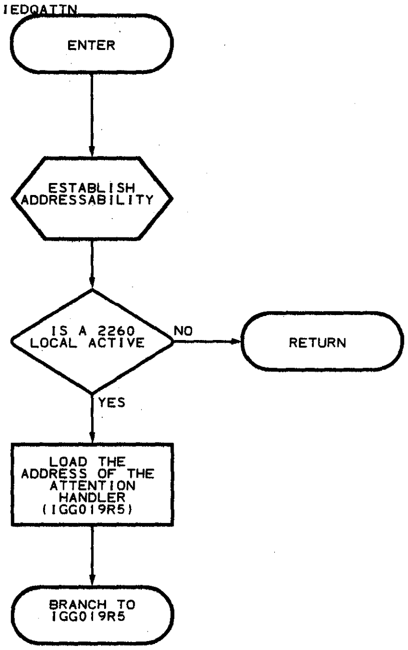


Chart UI USER INTERFACE ROUTINE

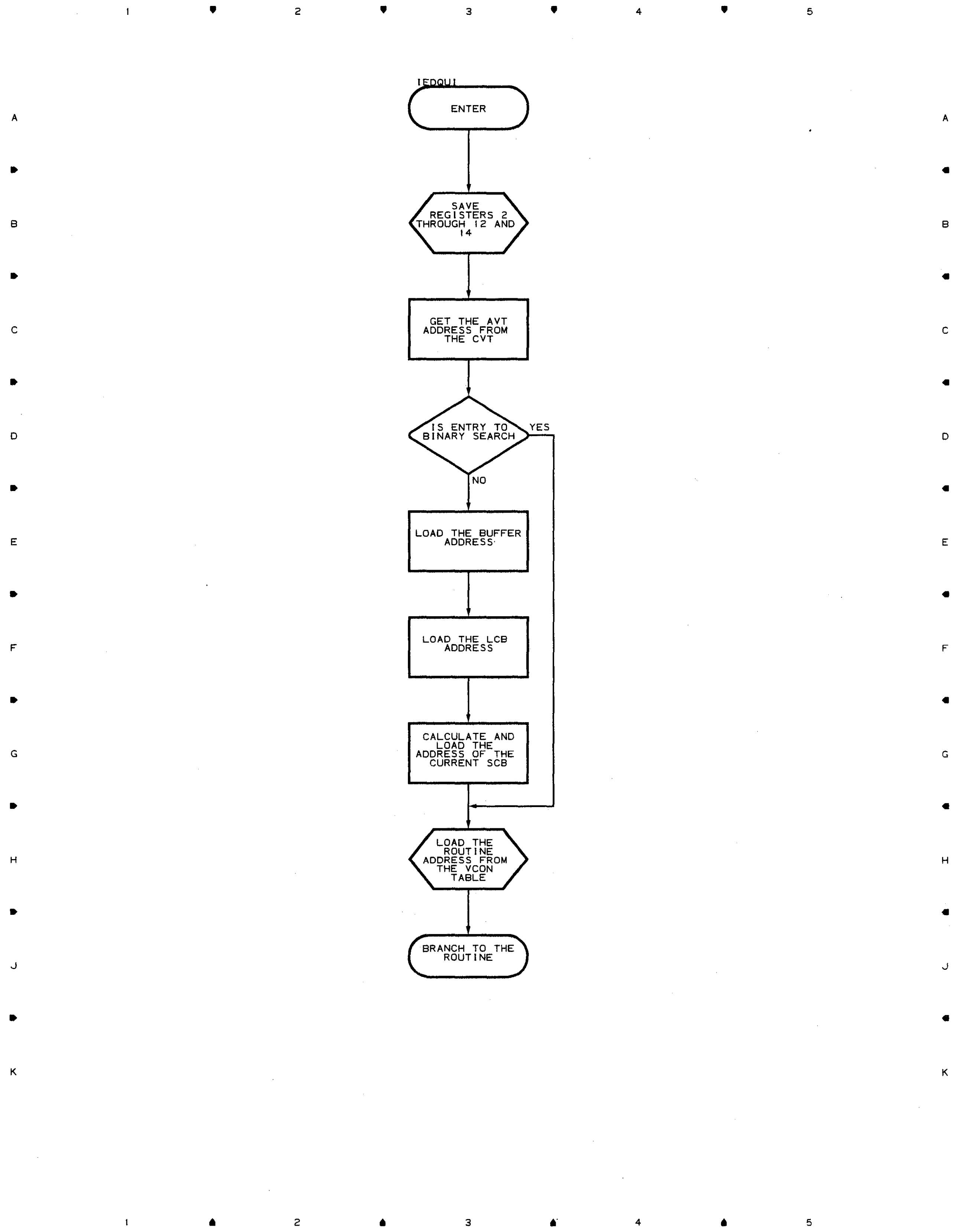

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQUI
ENTER
SAVE REGISTERS 2 THROUGH 12 AND 14
GET THE AVT ADDRESS FROM THE CVT
IS ENTRY TO BINARY SEARCH
YES
NO
LOAD THE BUFFER ADDRESS
LOAD THE LCB ADDRESS
CALCULATE AND LOAD THE ADDRESS OF THE CURRENT SCB
LOAD THE ROUTINE ADDRESS FROM THE VCON TABLE
BRANCH TO THE ROUTINE

Chart XA-1 DISK MESSAGE QUEUE INITIALIZER

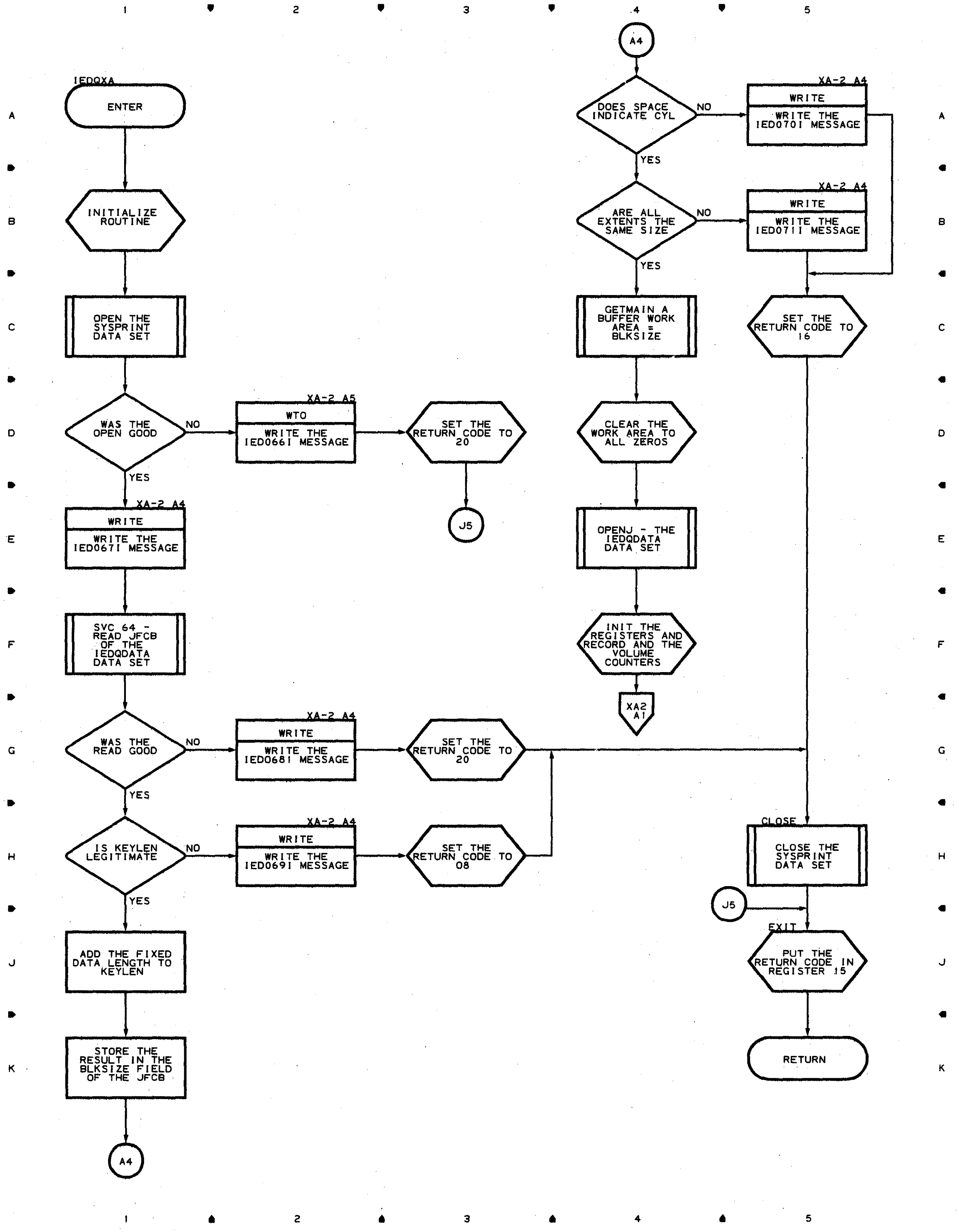
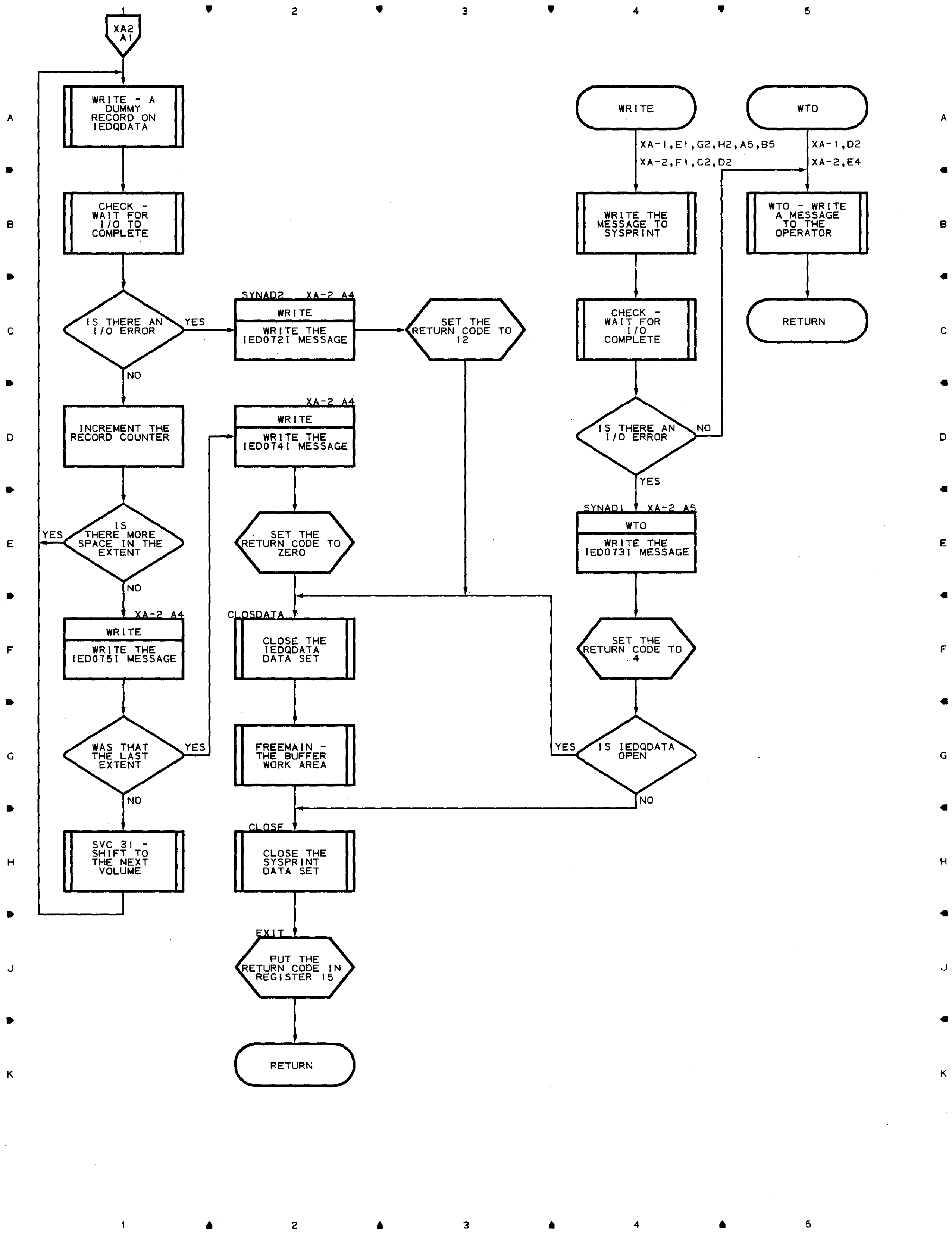
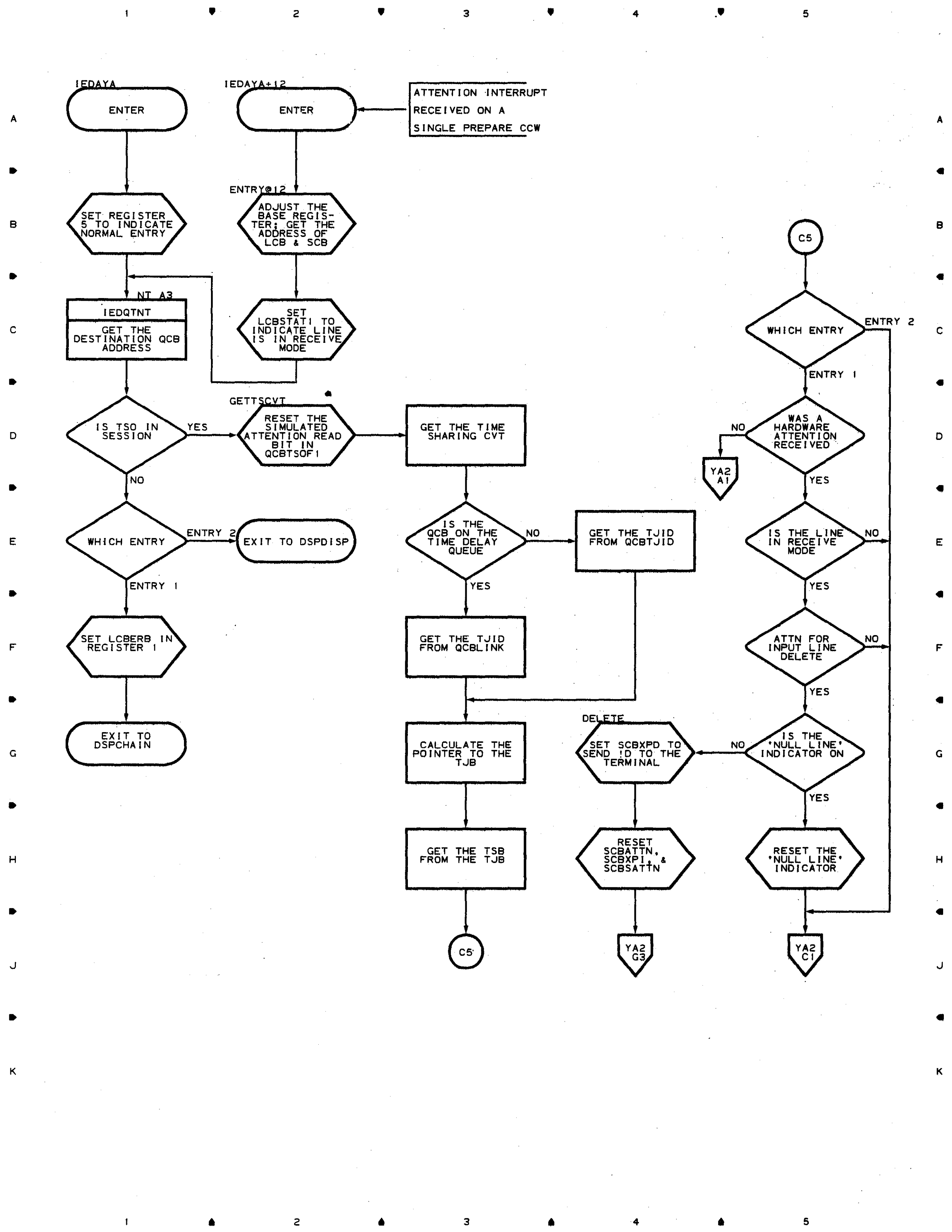

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDQXA
ENTER
INITIALIZE ROUTINE
OPEN THE SYSPRINT DATA SET
WAS THE OPEN GOOD
NO
YES
XA-2 A5
WTO
WRITE THE IED066I MESSAGE
SET THE RETURN CODE TO 20
J5
XA-2 A4
WRITE
WRITE THE IED067I MESSAGE
SVC 64 - READ JFCB OF THE IEDQDATA DATA SET
WAS THE READ GOOD
NO
YES
XA-2 A4
WRITE
WRITE THE IED068I MESSAGE
SET THE RETURN CODE TO 20
IS KEYLEN LEGITIMATE
NO
YES
XA-2 A4
WRITE
WRITE THE IED069I MESSAGE
SET THE RETURN CODE TO 08
ADD THE FIXED DATA LENGTH TO KEYLEN
STORE THE RESULT IN THE BLKSIZE FIELD OF THE JFCB
A4
A4
DOES SPACE INDICATE CYL
NO
YES
XA-2 A4
WRITE
WRITE THE IED070I MESSAGE
ARE ALL EXTENTS THE SAME SIZE
NO
YES
XA-2 A4
WRITE
WRITE THE IED071I MESSAGE
GETMAIN A BUFFER WORK AREA = BLKSIZE
SET THE RETURN CODE TO 16
CLEAR THE WORK AREA TO ALL ZEROS
OPENJ - THE IEDQDATA DATA SET
INIT THE REGISTERS AND RECORD AND THE VOLUME COUNTERS
XA2 A1
CLOSE
CLOSE THE SYSPRINT DATA SET
J5
EXIT
PUT THE RETURN CODE IN REGISTER 15
RETURN

Chart XA-2 DISK MESSAGE QUEUE INITIALIZER



XA2 A1
WRITE - A DUMMY RECORD ON IEDQDATA
CHECK - WAIT FOR I/O TO COMPLETE
IS THERE AN I/O ERROR
YES
NO
SYNAD2 XA-2 A4
WRITE
WRITE THE IED072I MESSAGE
SET THE RETURN CODE TO 12
INCREMENT THE RECORD COUNTER
XA-2 A4
WRITE
WRITE THE IED074I MESSAGE
IS THERE MORE SPACE IN THE EXTENT
YES
NO
SET THE RETURN CODE TO ZERO
XA-2 A4
WRITE
WRITE THE IED075I MESSAGE
CLOSDATA
CLOSE THE IEDQDATA DATA SET
WAS THAT THE LAST EXTENT
YES
NO
FREEMAIN - THE BUFFER WORK AREA
SVC 31 - SHIFT TO THE NEXT VOLUME
CLOSE
CLOSE THE SYSPRINT DATA SET
EXIT
PUT THE RETURN CODE IN REGISTER 15
RETURN
WRITE
XA-1,E1,G2,H2,A5,B5
XA-2,F1,C2,D2
WRITE THE MESSAGE TO SYSPRINT
CHECK - WAIT FOR I/O COMPLETE
IS THERE AN I/O ERROR
NO
YES
SYNAD1 XA-2 A5
WTO
WRITE THE IED073I MESSAGE
SET THE RETURN CODE TO 4
IS IEDQDATA OPEN
YES
NO
WTO
XA-1,D2
XA-2,E4
WTO - WRITE A MESSAGE TO THE OPERATOR
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart YA-1 TSO ATTENTION ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDAYA
ENTER
IEDAYA+12
ENTER
ATTENTION INTERRUPT RECEIVED ON A SINGLE PREPARE CCW
SET REGISTER 5 TO INDICATE NORMAL ENTRY
ENTRY@12
ADJUST THE BASE REGIS- TER; GET THE ADDRESS OF LCB & SCB
C5
NT A3
IEDQTNT
GET THE DESTINATION QCB ADDRESS
SET LCBSTATI TO INDICATE LINE IS IN RECEIVE MODE
WHICH ENTRY
ENTRY 2
ENTRY 1
GETTSCVT
IS TSO IN SESSION
YES
RESET THE SIMULATED ATTENTION READ BIT IN QCBTSOF1
GET THE TIME SHARING CVT
NO
WAS A HARDWARE ATTENTION RECEIVED
YA2 A1
NO
WHICH ENTRY
ENTRY 2
EXIT TO DSPDISP
IS THE QCB ON THE TIME DELAY QUEUE
NO
GET THE TJID FROM QCBTJID
YES
IS THE LINE IN RECEIVE MODE
NO
ENTRY 1
YES
YES
SET LCBERB IN REGISTER 1
GET THE TJID FROM QCBLINK
ATTN FOR INPUT LINE DELETE
NO
YES
DELETE
EXIT TO DSPCHAIN
CALCULATE THE POINTER TO THE TJB
SET SCBXPD TO SEND !D TO THE TERMINAL
NO
IS THE 'NULL LINE' INDICATOR ON
YES
GET THE TSB FROM THE TJB
RESET SCBATTN, SCBXPI, & SCBSATTN
RESET THE 'NULL LINE' INDICATOR
C5
YA2 G3
YA2 C1

Chart YA-2 TSO ATTENTION ROUITINE

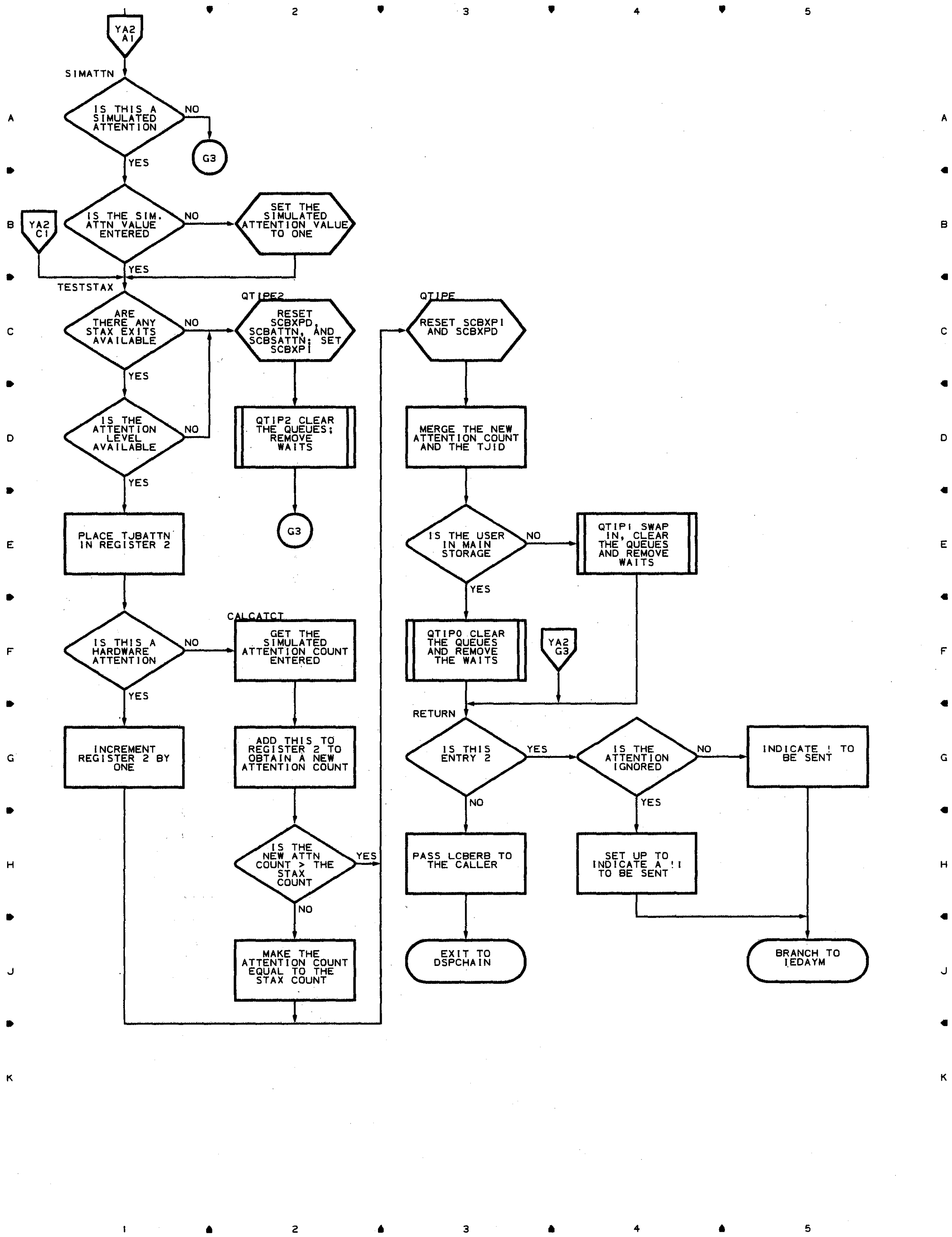
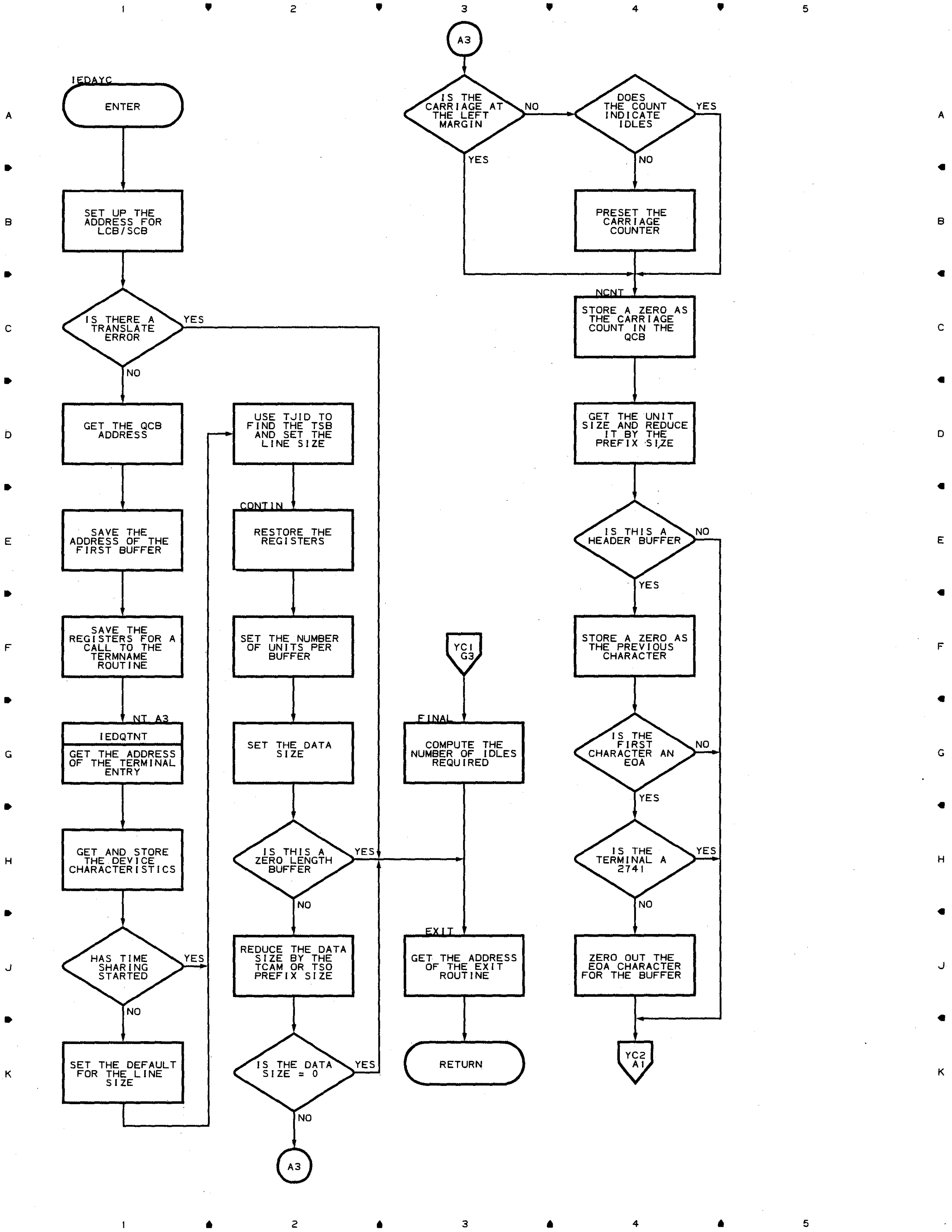

YA2 A1
SIMATTN
IS THIS A SIMULATED ATTENTION
NO
G3
YES
IS THE SIM. ATTN VALUE ENTERED
NO
SET THE SIMULATED ATTENTION VALUE TO ONE
YA2 C1
YES
TESTSTAX
ARE THERE ANY STAX EXITS AVAILABLE
NO
QTIPE2
RESET SCBXPD, SCBATTN, AND SCBSATTN; SET SCBXPI
QTIPE
RESET SCBXPI AND SCBXPD
YES
IS THE ATTENTION LEVEL AVAILABLE
NO
QTIP2 CLEAR THE QUEUES; REMOVE WAITS
MERGE THE NEW ATTENTION COUNT AND THE TJID
YES
PLACE TJBATTN IN REGISTER 2
G3
IS THE USER IN MAIN STORAGE
NO
QTIP1 SWAP IN, CLEAR THE QUEUES AND REMOVE WAITS
YES
CALCATCT
IS THIS A HARDWARE ATTENTION
NO
GET THE SIMULATED ATTENTION COUNT ENTERED
QTIP0 CLEAR THE QUEUES AND REMOVE THE WAITS
YA2 G3
YES
RETURN
INCREMENT REGISTER 2 BY ONE
ADD THIS TO REGISTER 2 TO OBTAIN A NEW ATTENTION COUNT
IS THIS ENTRY 2
YES
IS THE ATTENTION IGNORED
NO
INDICATE ! TO BE SENT
NO
YES
IS THE NEW ATTN COUNT > THE STAX COUNT
YES
PASS LCBERB TO THE CALLER
SET UP TO INDICATE A !I TO BE SENT
NO
MAKE THE ATTENTION COUNT EQUAL TO THE STAX COUNT
EXIT TO DSPCHAIN
BRANCH TO IEDAYM
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart YC-1 TSO CARRIAGE ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDAYC
ENTER
SET UP THE ADDRESS FOR LCB/SCB
IS THERE A TRANSLATE ERROR
YES
NO
GET THE QCB ADDRESS
SAVE THE ADDRESS OF THE FIRST BUFFER
SAVE THE REGISTERS FOR A CALL TO THE TERMNAME ROUTINE
NT A3
IEDQTNT
GET THE ADDRESS OF THE TERMINAL ENTRY
GET AND STORE THE DEVICE CHARACTERISTICS
HAS TIME SHARING STARTED
YES
NO
SET THE DEFAULT FOR THE LINE SIZE
USE TJID TO FIND THE TSB AND SET THE LINE SIZE
CONTIN
RESTORE THE REGISTERS
SET THE NUMBER OF UNITS PER BUFFER
SET THE DATA SIZE
IS THIS A ZERO LENGTH BUFFER
YES
NO
REDUCE THE DATA SIZE BY THE TCAM OR TSO PREFIX SIZE
IS THE DATA SIZE = 0
YES
NO
A3
A3
IS THE CARRIAGE AT THE LEFT MARGIN
NO
YES
YC1 G3
FINAL
COMPUTE THE NUMBER OF IDLES REQUIRED
EXIT
GET THE ADDRESS OF THE EXIT ROUTINE
RETURN
DOES THE COUNT INDICATE IDLES
YES
NO
PRESET THE CARRIAGE COUNTER
NCNT
STORE A ZERO AS THE CARRIAGE COUNT IN THE QCB
GET THE UNIT SIZE AND REDUCE IT BY THE PREFIX SIZE
IS THIS A HEADER BUFFER
NO
YES
STORE A ZERO AS THE PREVIOUS CHARACTER
IS THE FIRST CHARACTER AN EOA
NO
YES
IS THE TERMINAL A 2741
YES
NO
ZERO OUT THE EOA CHARACTER FOR THE BUFFER
YC2 A1

Chart YC-2 TSO CARRIAGE ROUTINE

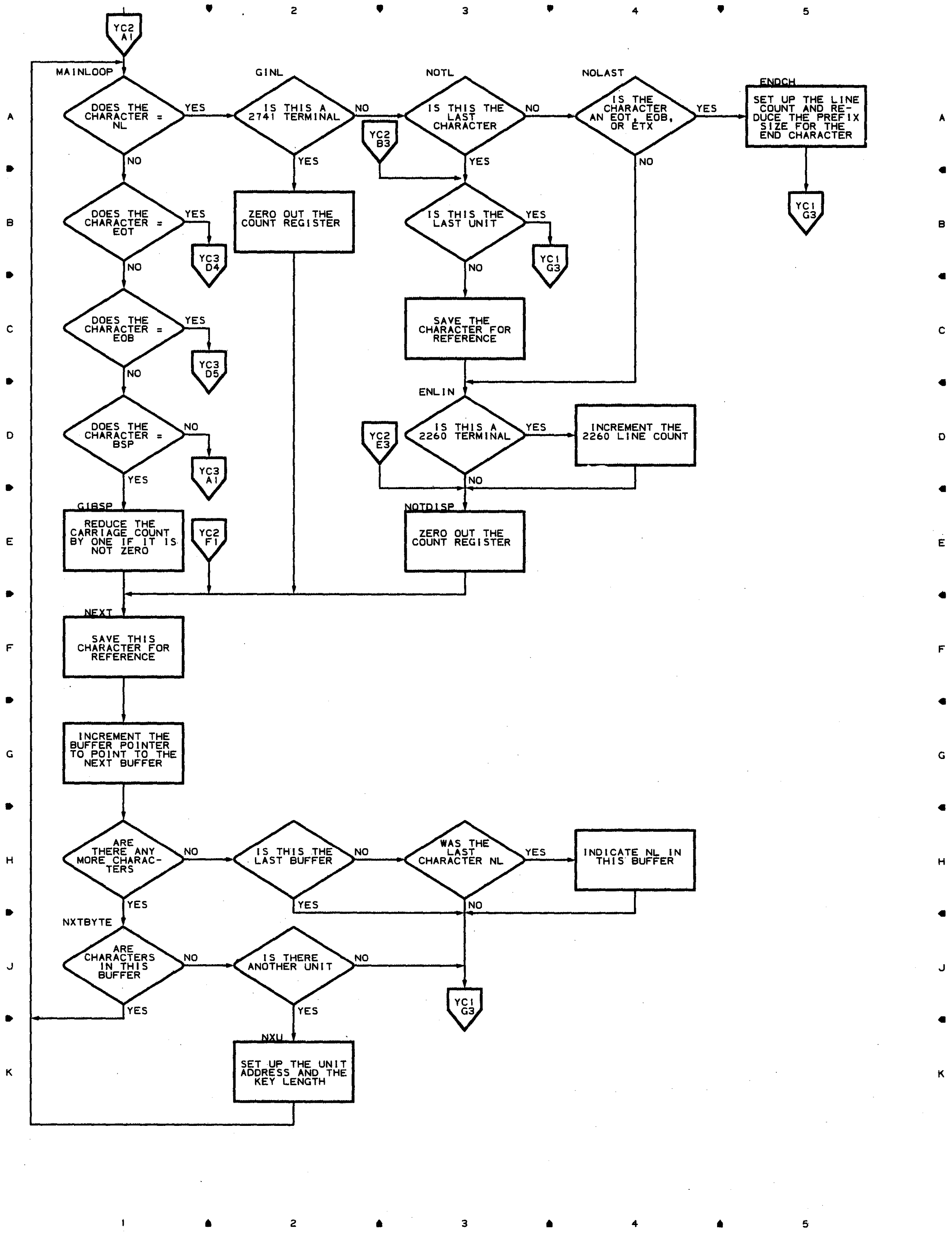

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
YC2 A1
MAINLOOP
DOES THE CHARACTER = NL
YES
GINL
IS THIS A 2741 TERMINAL
NO
NOTL
IS THIS THE LAST CHARACTER
NO
NOLAST
IS THE CHARACTER AN EOT, EOB, OR ETX
YES
ENDCH
SET UP THE LINE COUNT AND RE- DUCE THE PREFIX SIZE FOR THE END CHARACTER
YC1 G3
NO
YES
YC2 B3
YES
NO
DOES THE CHARACTER = EOT
YES
YC3 D4
ZERO OUT THE COUNT REGISTER
IS THIS THE LAST UNIT
YES
YC1 G3
NO
NO
DOES THE CHARACTER = EOB
YES
YC3 D5
SAVE THE CHARACTER FOR REFERENCE
NO
ENLIN
DOES THE CHARACTER = BSP
NO
YC3 A1
YC2 E3
IS THIS A 2260 TERMINAL
YES
INCREMENT THE 2260 LINE COUNT
YES
NO
GIBSP
REDUCE THE CARRIAGE COUNT BY ONE IF IT IS NOT ZERO
YC2 F1
NOTDISP
ZERO OUT THE COUNT REGISTER
NEXT
SAVE THIS CHARACTER FOR REFERENCE
INCREMENT THE BUFFER POINTER TO POINT TO THE NEXT BUFFER
ARE THERE ANY MORE CHARAC- TERS
NO
IS THIS THE LAST BUFFER
NO
WAS THE LAST CHARACTER NL
YES
INDICATE NL IN THIS BUFFER
YES
YES
NO
NXTBYTE
ARE CHARACTERS IN THIS BUFFER
NO
IS THERE ANOTHER UNIT
NO
YES
YES
YC1 G3
NXU
SET UP THE UNIT ADDRESS AND THE KEY LENGTH

Chart YC-3 TSO CARRIAGE ROUTINE

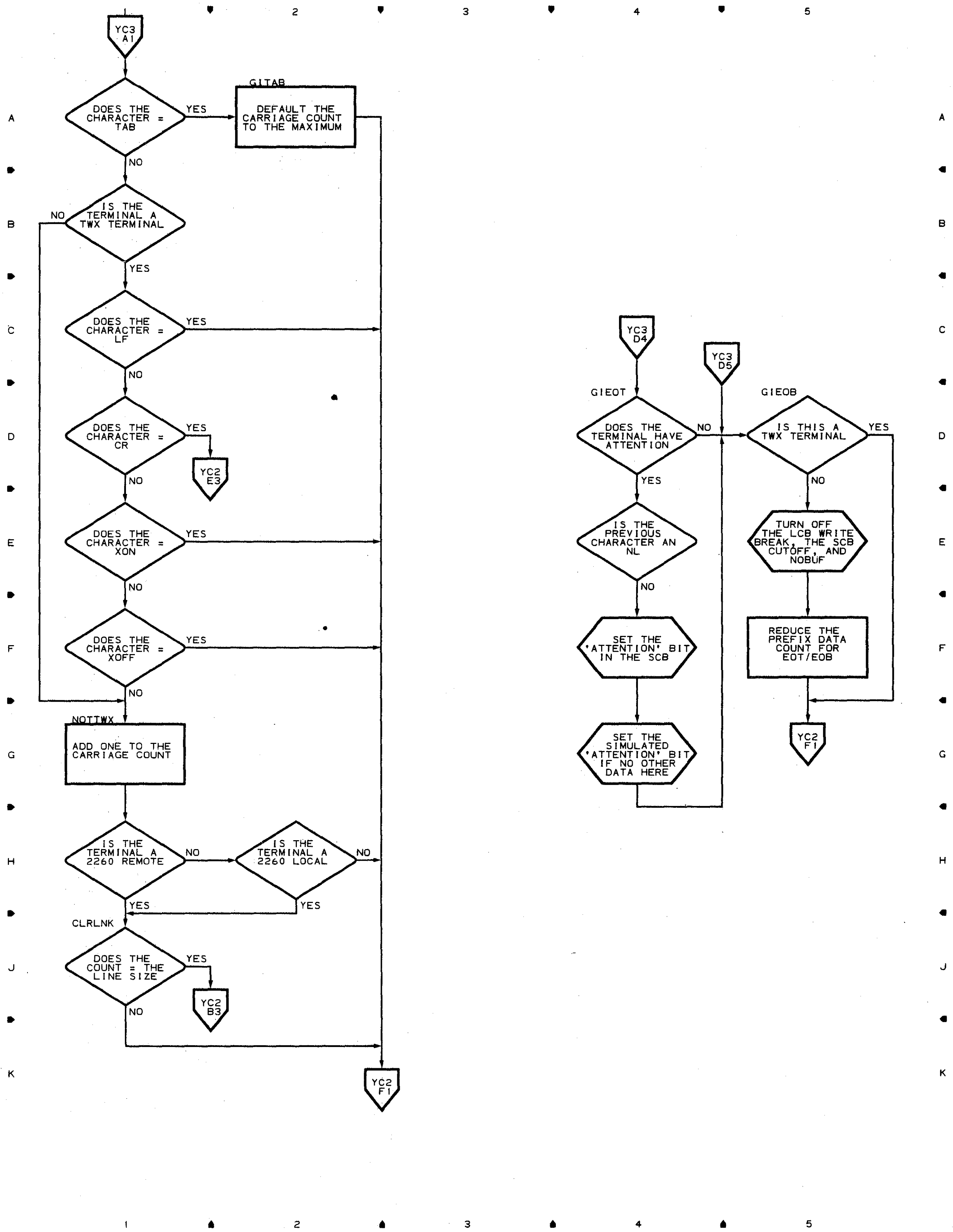

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
YC3 A1
DOES THE CHARACTER = TAB
YES
GITAB
DEFAULT THE CARRIAGE COUNT TO THE MAXIMUM
NO
IS THE TERMINAL A TWX TERMINAL
NO
YES
DOES THE CHARACTER = LF
YES
NO
DOES THE CHARACTER = CR
YES
YC2 E3
NO
DOES THE CHARACTER = XON
YES
NO
DOES THE CHARACTER = XOFF
YES
NO
NOTTWX
ADD ONE TO THE CARRIAGE COUNT
IS THE TERMINAL A 2260 REMOTE
NO
IS THE TERMINAL A 2260 LOCAL
NO
YES
YES
CLRLNK
DOES THE COUNT = THE LINE SIZE
YES
YC2 B3
NO
YC2 F1
YC3 D4
YC3 D5
GIEOT
DOES THE TERMINAL HAVE ATTENTION
NO
YES
IS THE PREVIOUS CHARACTER AN NL
NO
SET THE 'ATTENTION' BIT IN THE SCB
SET THE SIMULATED 'ATTENTION' BIT IF NO OTHER DATA HERE
GIEOB
IS THIS A TWX TERMINAL
YES
NO
TURN OFF THE LCB WRITE BREAK, THE SCB CUTOFF, AND NOBUF
REDUCE THE PREFIX DATA COUNT FOR EOT/EOB
YC2 F1

Chart YD TSO DESTINATION SCHEDULER

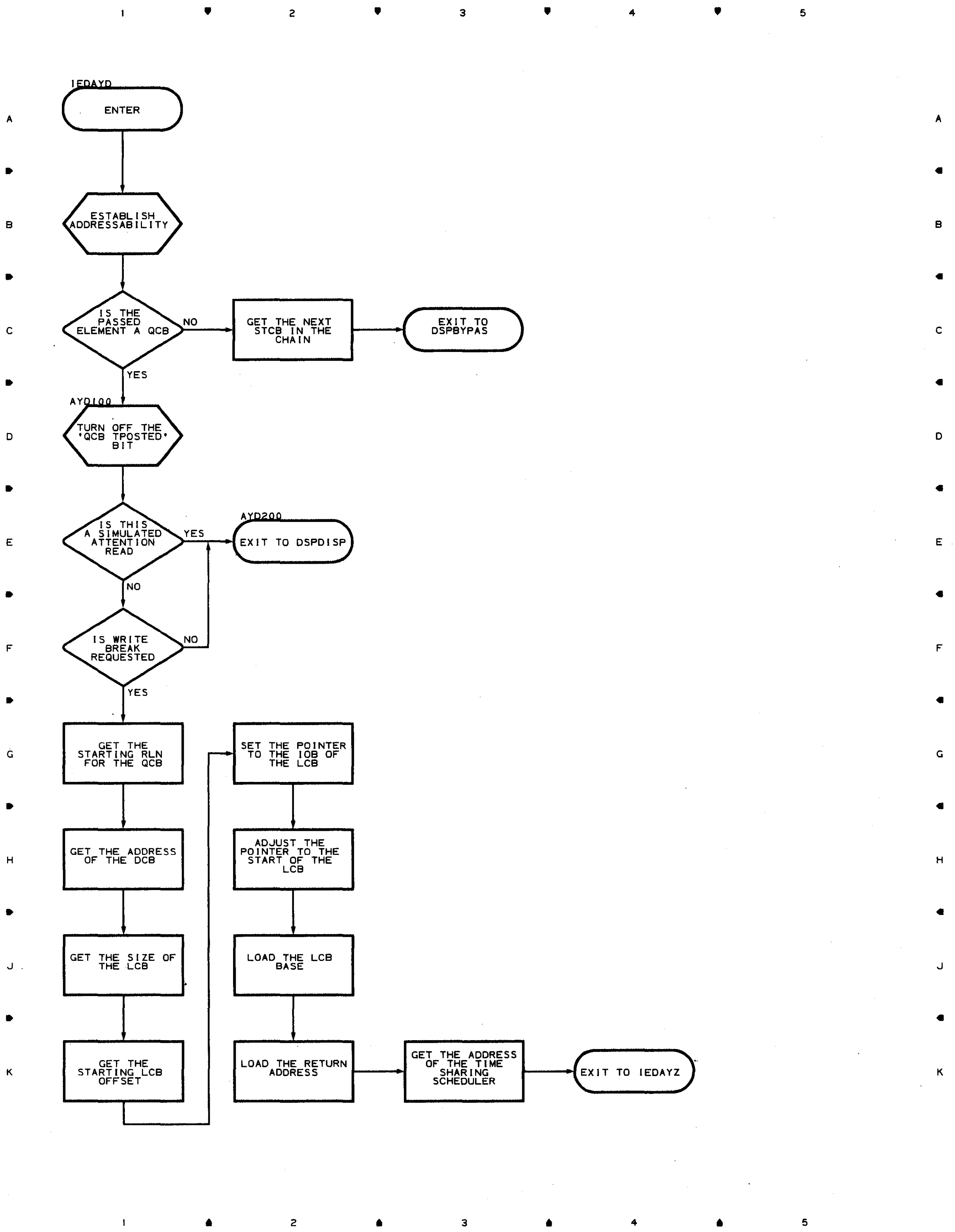
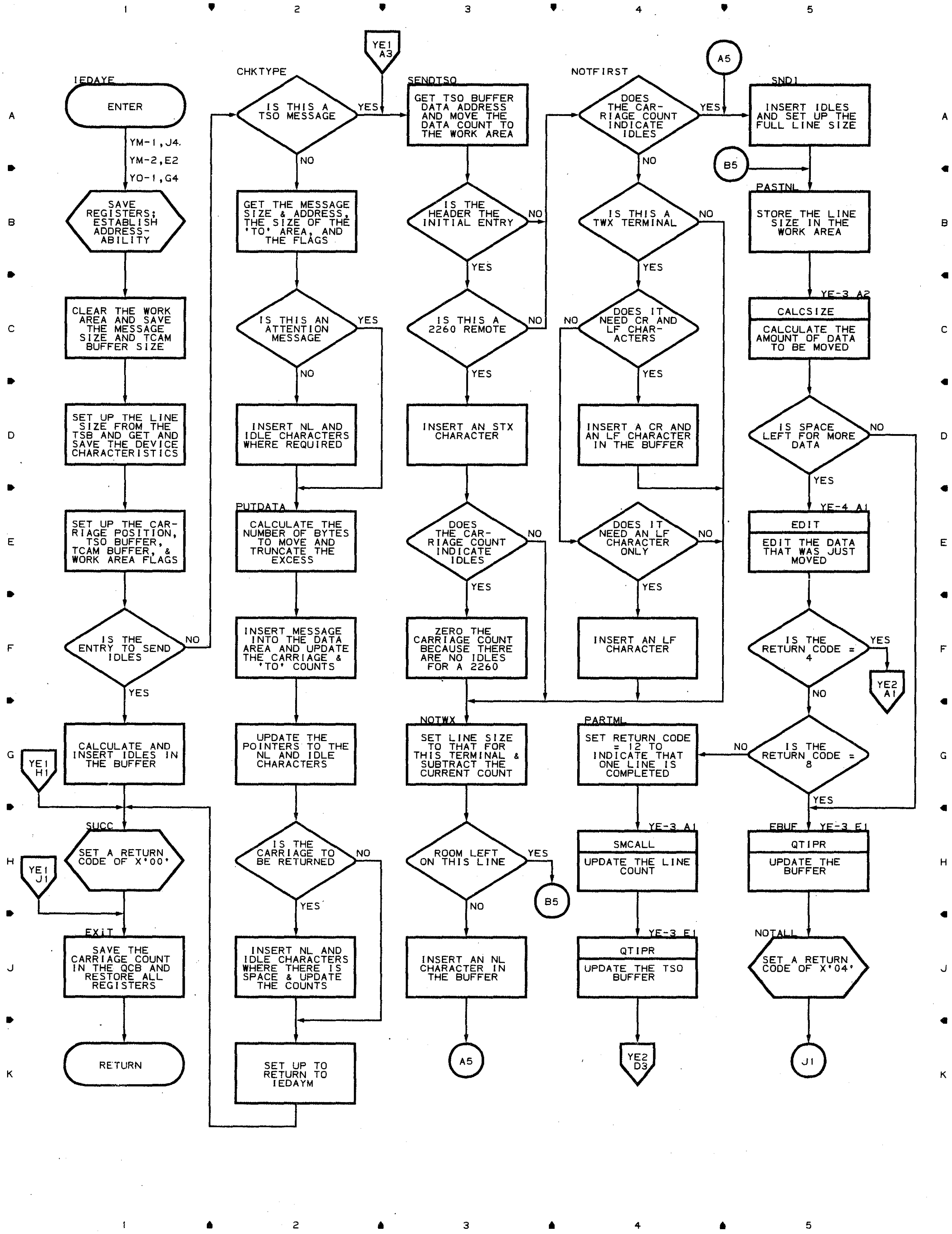

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDAYD
ENTER
ESTABLISH ADDRESSABILITY
IS THE PASSED ELEMENT A QCB
NO
GET THE NEXT STCB IN THE CHAIN
EXIT TO DSPBYPAS
YES
AYD100
TURN OFF THE 'QCB TPOSTED' BIT
IS THIS A SIMULATED ATTENTION READ
YES
AYD200
EXIT TO DSPDISP
NO
IS WRITE BREAK REQUESTED
NO
YES
GET THE STARTING RLN FOR THE QCB
SET THE POINTER TO THE IOB OF THE LCB
GET THE ADDRESS OF THE DCB
ADJUST THE POINTER TO THE START OF THE LCB
GET THE SIZE OF THE LCB
LOAD THE LCB BASE
GET THE STARTING LCB OFFSET
LOAD THE RETURN ADDRESS
GET THE ADDRESS OF THE TIME SHARING SCHEDULER
EXIT TO IEDAYZ

Chart YE-1 TSO TIOC EDIT ROUTINE



IEDAYE
ENTER
YM-1,J4.
YM-2,E2
YO-1,G4
SAVE REGISTERS; ESTABLISH ADDRESS-ABILITY
CLEAR THE WORK AREA AND SAVE THE MESSAGE SIZE AND TCAM BUFFER SIZE
SET UP THE LINE SIZE FROM THE TSB AND GET AND SAVE THE DEVICE CHARACTERISTICS
SET UP THE CARRIAGE POSITION, TSO BUFFER, TCAM BUFFER, & WORK AREA FLAGS
IS THE ENTRY TO SEND IDLES
NO
YES
CALCULATE AND INSERT IDLES IN THE BUFFER
YE1 H1
SUCC
SET A RETURN CODE OF X'00'
YE1 J1
EXIT
SAVE THE CARRIAGE COUNT IN THE QCB AND RESTORE ALL REGISTERS
RETURN
CHKTYPE
IS THIS A TSO MESSAGE
YES
NO
YE1 A3
GET THE MESSAGE SIZE & ADDRESS, THE SIZE OF THE 'TO' AREA, AND THE FLAGS
IS THIS AN ATTENTION MESSAGE
YES
NO
INSERT NL AND IDLE CHARACTERS WHERE REQUIRED
PUTDATA
CALCULATE THE NUMBER OF BYTES TO MOVE AND TRUNCATE THE EXCESS
INSERT MESSAGE INTO THE DATA AREA AND UPDATE THE CARRIAGE & 'TO' COUNTS
UPDATE THE POINTERS TO THE NL AND IDLE CHARACTERS
IS THE CARRIAGE TO BE RETURNED
NO
YES
INSERT NL AND IDLE CHARACTERS WHERE THERE IS SPACE & UPDATE THE COUNTS
SET UP TO RETURN TO IEDAYM
SENDTSO
GET TSO BUFFER DATA ADDRESS AND MOVE THE DATA COUNT TO THE WORK AREA
IS THE HEADER THE INITIAL ENTRY
NO
YES
IS THIS A 2260 REMOTE
NO
YES
INSERT AN STX CHARACTER
DOES THE CARRIAGE COUNT INDICATE IDLES
NO
YES
ZERO THE CARRIAGE COUNT BECAUSE THERE ARE NO IDLES FOR A 2260
NOTWX
SET LINE SIZE TO THAT FOR THIS TERMINAL & SUBTRACT THE CURRENT COUNT
ROOM LEFT ON THIS LINE
YES
NO
B5
INSERT AN NL CHARACTER IN THE BUFFER
A5
NOTFIRST
A5
DOES THE CARRIAGE COUNT INDICATE IDLES
YES
NO
IS THIS A TWX TERMINAL
NO
YES
DOES IT NEED CR AND LF CHARACTERS
NO
YES
INSERT A CR AND AN LF CHARACTER IN THE BUFFER
DOES IT NEED AN LF CHARACTER ONLY
NO
YES
INSERT AN LF CHARACTER
PARTML
SET RETURN CODE = 12 TO INDICATE THAT ONE LINE IS COMPLETED
YE-3 A1
SMCALL
UPDATE THE LINE COUNT
YE-3 E1
QTIPR
UPDATE THE TSO BUFFER
YE2 D3
SND1
INSERT IDLES AND SET UP THE FULL LINE SIZE
B5
PASTNL
STORE THE LINE SIZE IN THE WORK AREA
YE-3 A2
CALCSIZE
CALCULATE THE AMOUNT OF DATA TO BE MOVED
IS SPACE LEFT FOR MORE DATA
NO
YES
YE-4 A1
EDIT
EDIT THE DATA THAT WAS JUST MOVED
IS THE RETURN CODE = 4
YES
NO
YE2 A1
IS THE RETURN CODE = 8
NO
YES
EBUF
YE-3 E1
QTIPR
UPDATE THE BUFFER
NOTALL
SET A RETURN CODE OF X'04'
J1
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart YE-2 TSO TIOC EDIT ROUTINE

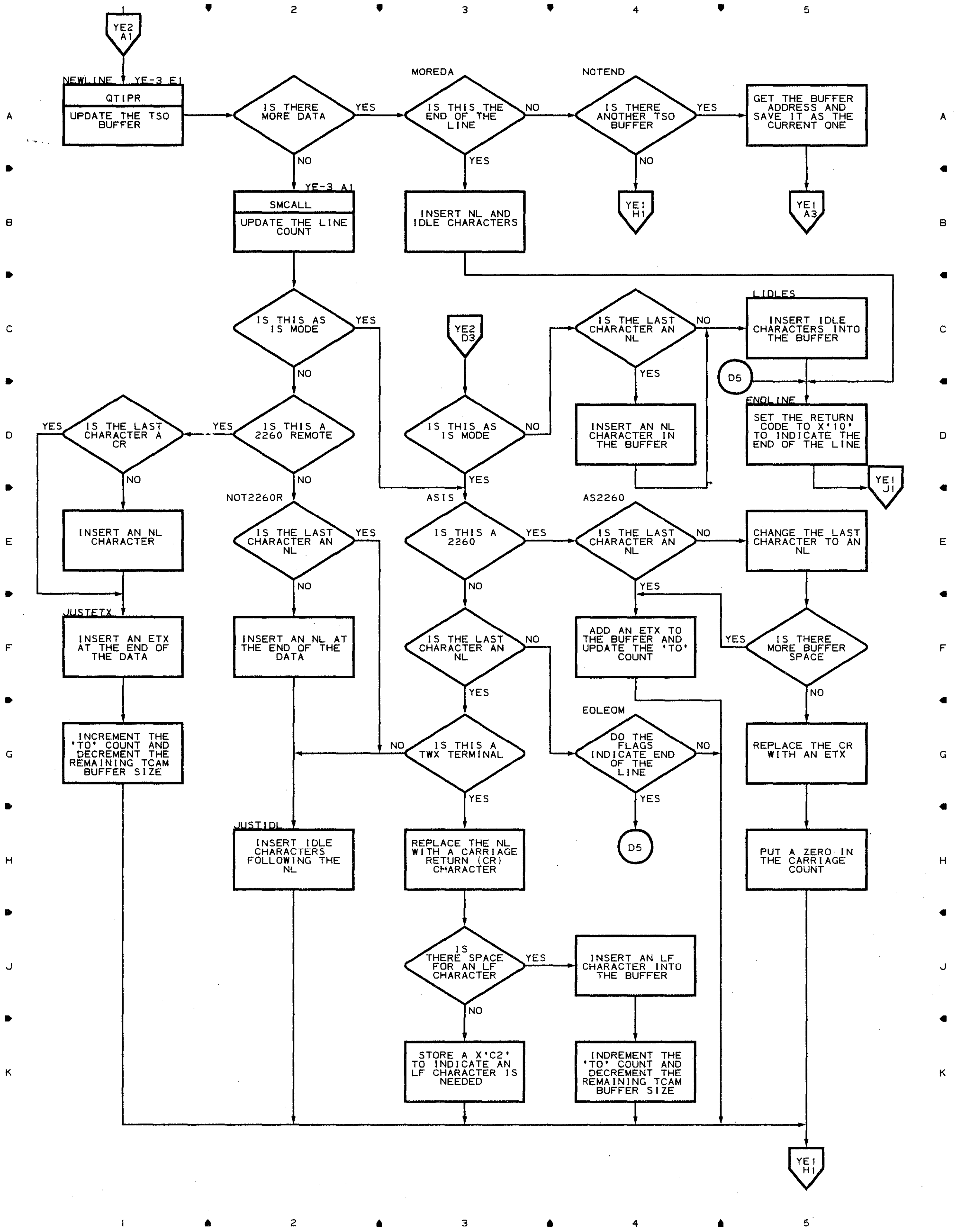
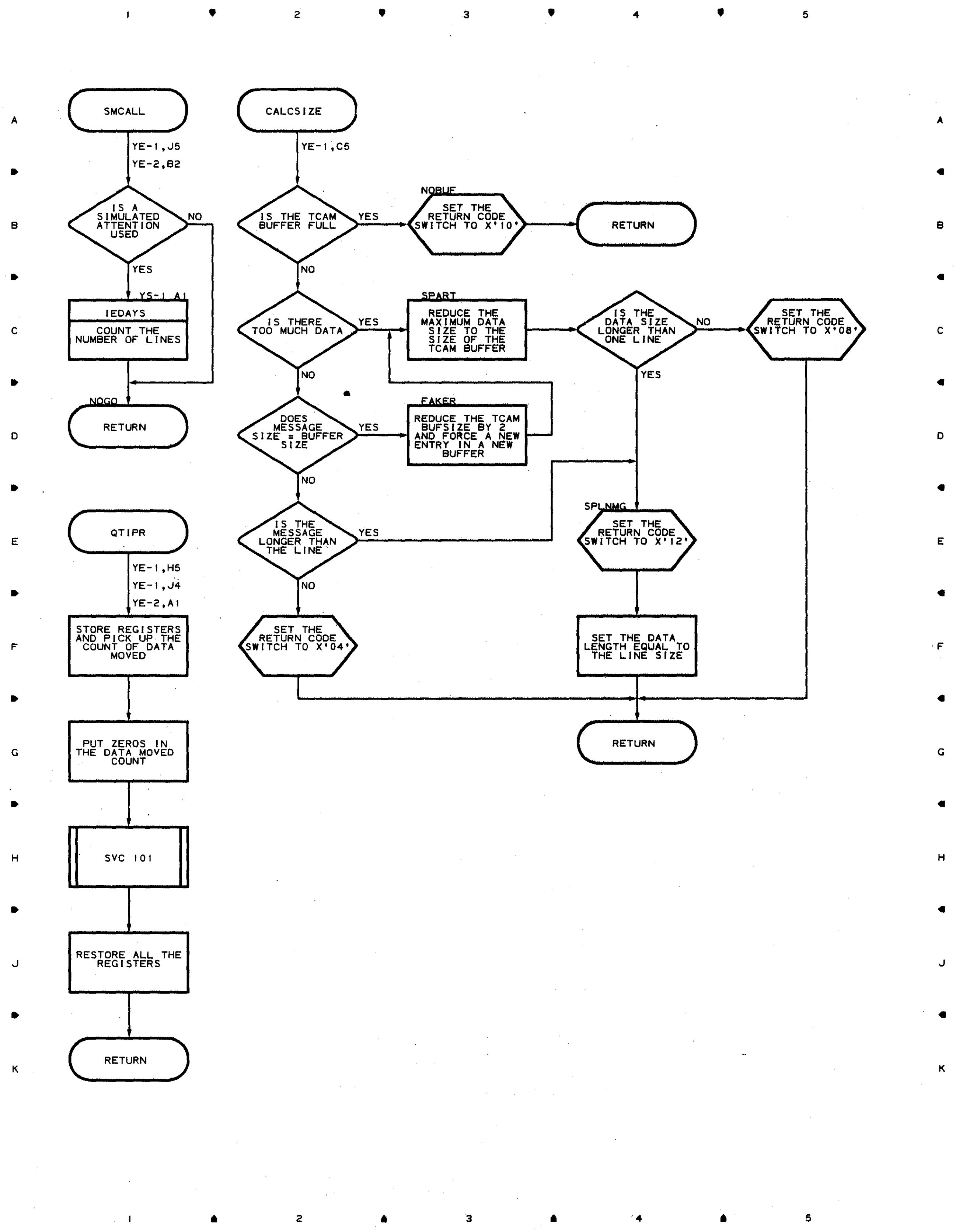

YE2 A1
NEWLINE YE-3 E1
QTIPR
UPDATE THE TSO BUFFER
IS THERE MORE DATA
YES
NO
MOREDA
IS THIS THE END OF THE LINE
NO
YES
NOTEND
IS THERE ANOTHER TSO BUFFER
YES
NO
GET THE BUFFER ADDRESS AND SAVE IT AS THE CURRENT ONE
YE-3 A1
SMCALL
UPDATE THE LINE COUNT
INSERT NL AND IDLE CHARACTERS
YE1 H1
YE1 A3
IS THIS AS IS MODE
YES
NO
YE2 D3
IS THE LAST CHARACTER AN NL
NO
YES
LIDLES
INSERT IDLE CHARACTERS INTO THE BUFFER
D5
ENDLINE
SET THE RETURN CODE TO X'10' TO INDICATE THE END OF THE LINE
YE1 J1
IS THE LAST CHARACTER A CR
YES
NO
IS THIS A 2260 REMOTE
YES
NO
IS THIS AS IS MODE
NO
YES
INSERT AN NL CHARACTER IN THE BUFFER
NOT2260R
ASIS
AS2260
INSERT AN NL CHARACTER
IS THE LAST CHARACTER AN NL
YES
NO
IS THIS A 2260
YES
NO
IS THE LAST CHARACTER AN NL
NO
YES
CHANGE THE LAST CHARACTER TO AN NL
JUSTETX
INSERT AN ETX AT THE END OF THE DATA
INSERT AN NL AT THE END OF THE DATA
IS THE LAST CHARACTER AN NL
NO
YES
ADD AN ETX TO THE BUFFER AND UPDATE THE 'TO' COUNT
IS THERE MORE BUFFER SPACE
YES
NO
INCREMENT THE 'TO' COUNT AND DECREMENT THE REMAINING TCAM BUFFER SIZE
IS THIS A TWX TERMINAL
NO
YES
EOLEOM
DO THE FLAGS INDICATE END OF THE LINE
NO
YES
REPLACE THE CR WITH AN ETX
JUSTIDL
INSERT IDLE CHARACTERS FOLLOWING THE NL
REPLACE THE NL WITH A CARRIAGE RETURN (CR) CHARACTER
D5
PUT A ZERO IN THE CARRIAGE COUNT
IS THERE SPACE FOR AN LF CHARACTER
YES
NO
INSERT AN LF CHARACTER INTO THE BUFFER
STORE A X'C2' TO INDICATE AN LF CHARACTER IS NEEDED
INDREMENT THE 'TO' COUNT AND DECREMENT THE REMAINING TCAM BUFFER SIZE
YE1 H1
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart YE-3 TSO TIOC EDIT ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
SMCALL
YE-1,J5
YE-2,B2
IS A SIMULATED ATTENTION USED
NO
YES
YS-1,A1
IEDAYS
COUNT THE NUMBER OF LINES
NOGO
RETURN
CALCSIZE
YE-1,C5
IS THE TCAM BUFFER FULL
YES
NOBUF
SET THE RETURN CODE SWITCH TO X'10'
RETURN
NO
IS THERE TOO MUCH DATA
YES
SPART
REDUCE THE MAXIMUM DATA SIZE TO THE SIZE OF THE TCAM BUFFER
IS THE DATA SIZE LONGER THAN ONE LINE
NO
SET THE RETURN CODE SWITCH TO X'08'
YES
NO
DOES MESSAGE SIZE = BUFFER SIZE
YES
FAKER
REDUCE THE TCAM BUFSIZE BY 2 AND FORCE A NEW ENTRY IN A NEW BUFFER
NO
QTIPR
YE-1,H5
YE-1,J4
YE-2,A1
IS THE MESSAGE LONGER THAN THE LINE
YES
SPLNMG
SET THE RETURN CODE SWITCH TO X'12'
NO
STORE REGISTERS AND PICK UP THE COUNT OF DATA MOVED
SET THE RETURN CODE SWITCH TO X'04'
SET THE DATA LENGTH EQUAL TO THE LINE SIZE
PUT ZEROS IN THE DATA MOVED COUNT
RETURN
SVC 101
RESTORE ALL THE REGISTERS
RETURN

Chart YE-4 TSO TIOC EDIT ROUTINE

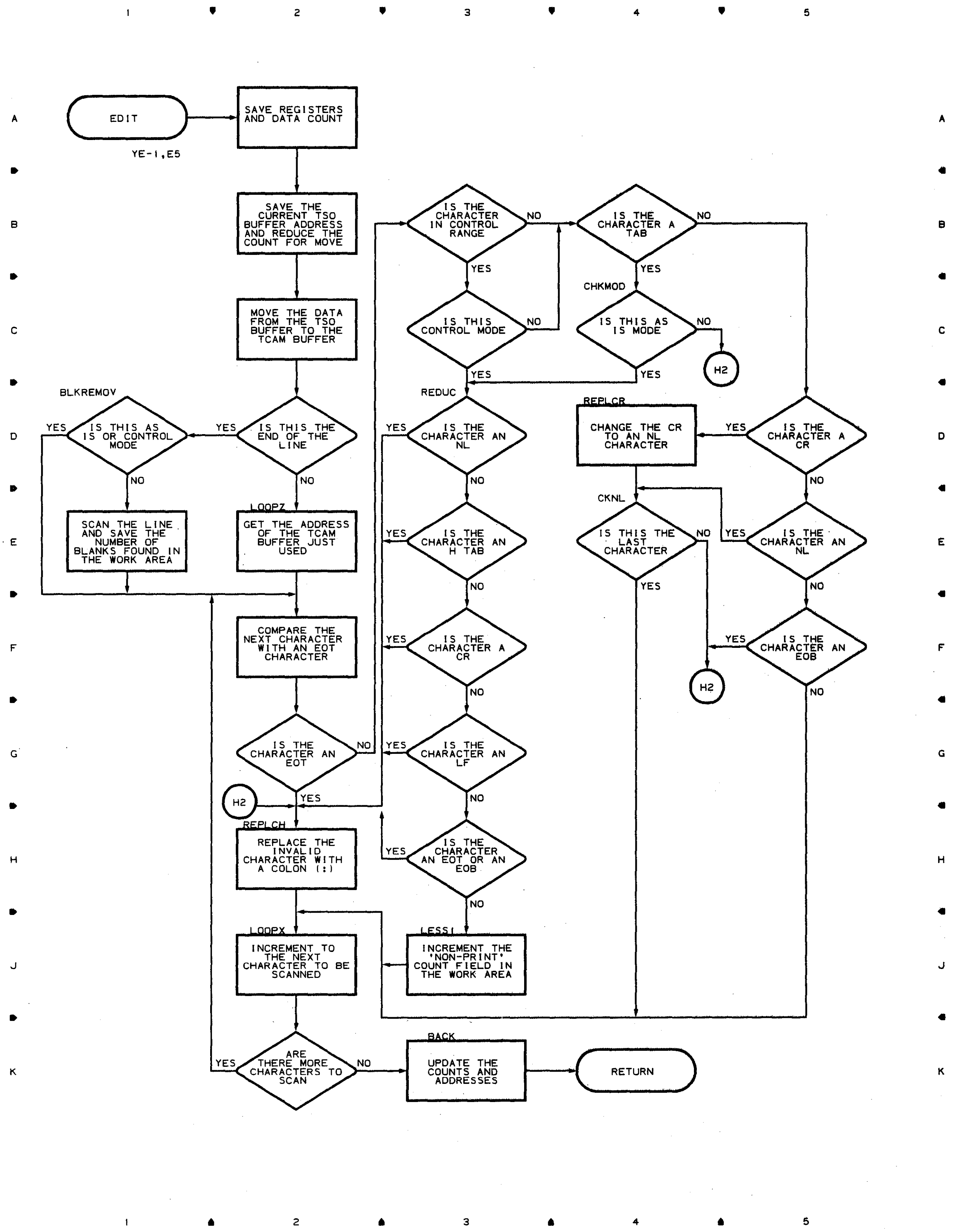

EDIT
YE-1,E5
SAVE REGISTERS AND DATA COUNT
SAVE THE CURRENT TSO BUFFER ADDRESS AND REDUCE THE COUNT FOR MOVE
MOVE THE DATA FROM THE TSO BUFFER TO THE TCAM BUFFER
IS THIS THE END OF THE LINE
YES
NO
BLKREMOV
IS THIS AS IS OR CONTROL MODE
YES
NO
SCAN THE LINE AND SAVE THE NUMBER OF BLANKS FOUND IN THE WORK AREA
LOOPZ
GET THE ADDRESS OF THE TCAM BUFFER JUST USED
COMPARE THE NEXT CHARACTER WITH AN EOT CHARACTER
IS THE CHARACTER AN EOT
NO
YES
H2
REPLCH
REPLACE THE INVALID CHARACTER WITH A COLON (:)
LOOPX
INCREMENT TO THE NEXT CHARACTER TO BE SCANNED
ARE THERE MORE CHARACTERS TO SCAN
YES
NO
BACK
UPDATE THE COUNTS AND ADDRESSES
RETURN
IS THE CHARACTER IN CONTROL RANGE
NO
YES
IS THIS CONTROL MODE
NO
YES
REDUC
IS THE CHARACTER AN NL
YES
NO
IS THE CHARACTER AN H TAB
YES
NO
IS THE CHARACTER A CR
YES
NO
IS THE CHARACTER AN LF
YES
NO
IS THE CHARACTER AN EOT OR AN EOB
YES
NO
LESS1
INCREMENT THE 'NON-PRINT' COUNT FIELD IN THE WORK AREA
IS THE CHARACTER A TAB
NO
YES
CHKMOD
IS THIS AS IS MODE
NO
YES
H2
REPLCR
CHANGE THE CR TO AN NL CHARACTER
CKNL
IS THIS THE LAST CHARACTER
NO
YES
H2
IS THE CHARACTER A CR
YES
NO
IS THE CHARACTER AN NL
YES
NO
IS THE CHARACTER AN EOB
YES
NO
1 2 3 4 5
A B C D E F G H J K

Chart YF TSO IOHALT

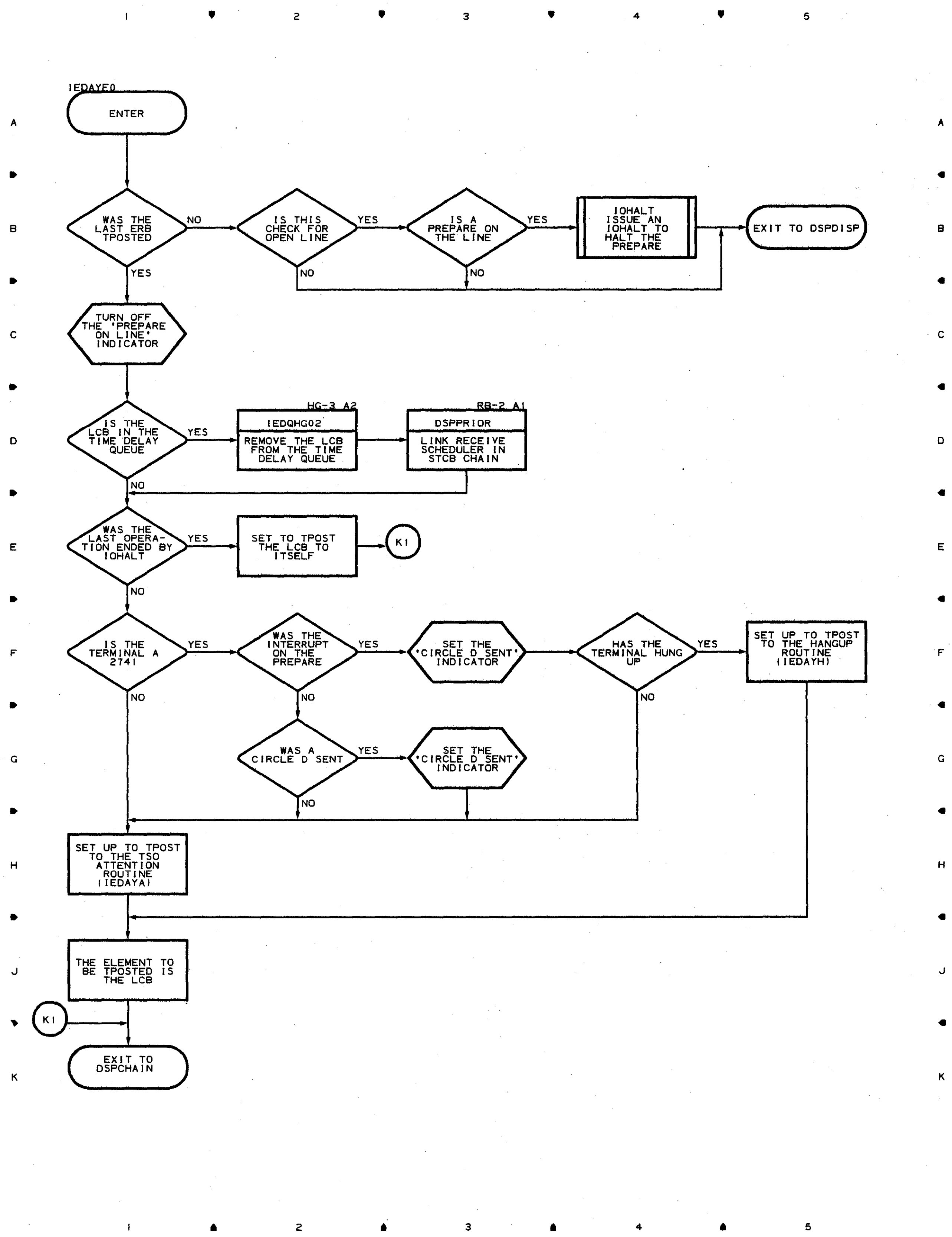

IEDAYF0
ENTER
WAS THE LAST ERB TPOSTED
NO
YES
IS THIS CHECK FOR OPEN LINE
YES
NO
IS A PREPARE ON THE LINE
YES
NO
IOHALT ISSUE AN IOHALT TO HALT THE PREPARE
EXIT TO DSPDISP
TURN OFF THE 'PREPARE ON LINE' INDICATOR
IS THE LCB IN THE TIME DELAY QUEUE
YES
NO
HG-3 A2
IEDQHG02
REMOVE THE LCB FROM THE TIME DELAY QUEUE
RB-2 A1
DSPPRIOR
LINK RECEIVE SCHEDULER IN STCB CHAIN
WAS THE LAST OPERATION ENDED BY IOHALT
YES
NO
SET TO TPOST THE LCB TO ITSELF
K1
IS THE TERMINAL A 2741
YES
NO
WAS THE INTERRUPT ON THE PREPARE
YES
NO
SET THE 'CIRCLE D SENT' INDICATOR
HAS THE TERMINAL HUNG UP
YES
NO
SET UP TO TPOST TO THE HANGUP ROUTINE (IEDAYH)
WAS A CIRCLE D SENT
YES
NO
SET THE 'CIRCLE D SENT' INDICATOR
SET UP TO TPOST TO THE TSO ATTENTION ROUTINE (IEDAYA)
THE ELEMENT TO BE TPOSTED IS THE LCB
K1
EXIT TO DSPCHAIN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart YH TSO HANGUP ROUTINE

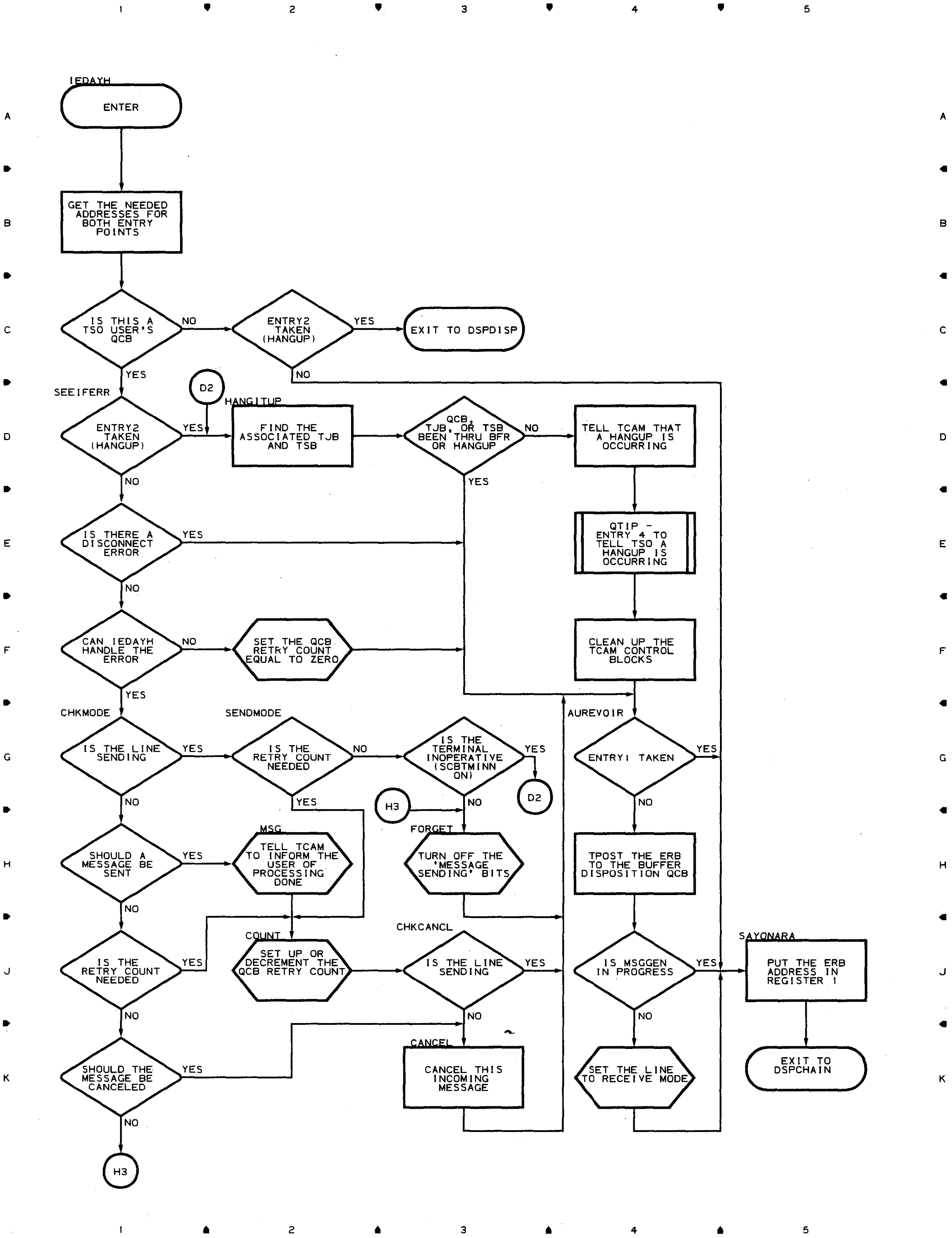
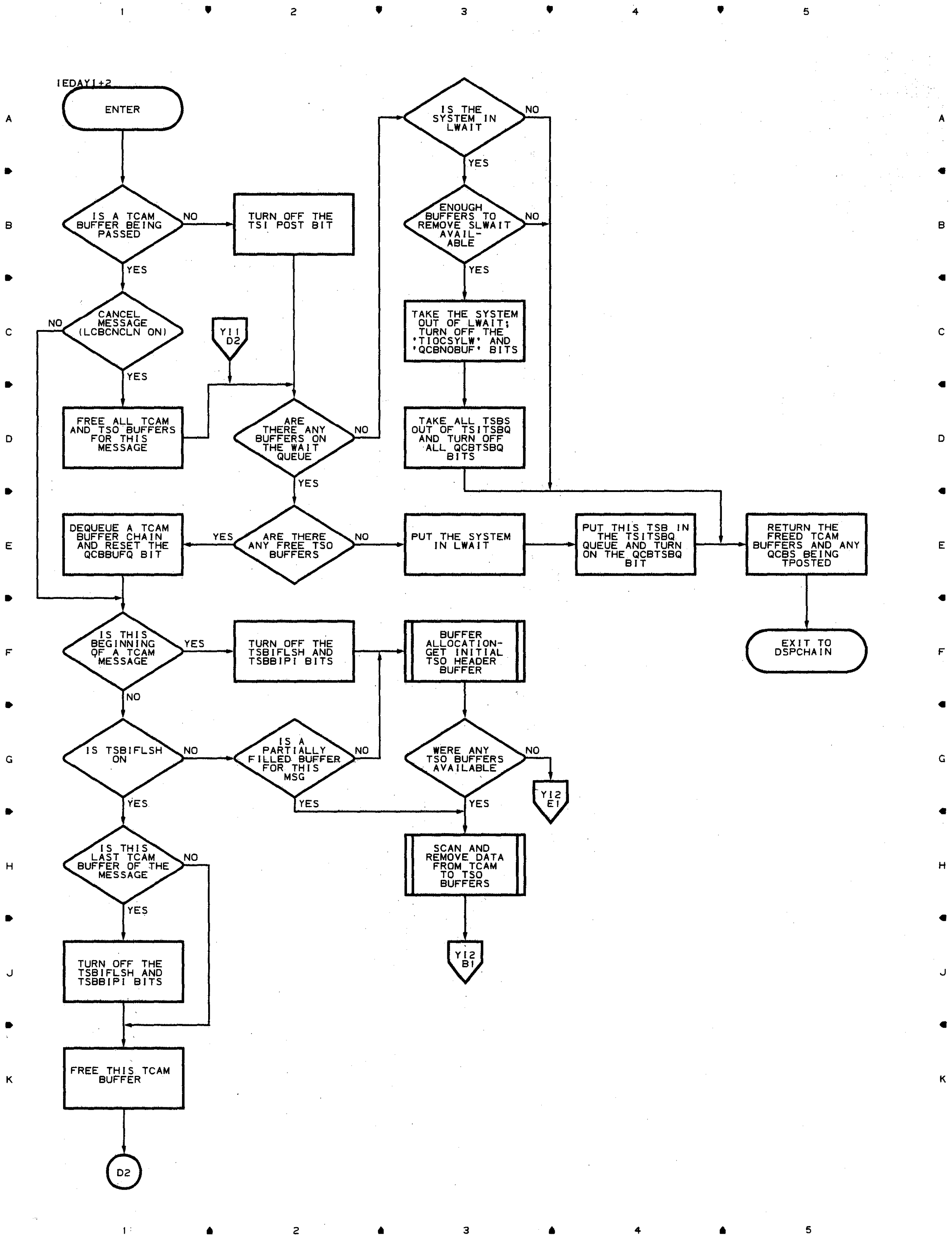
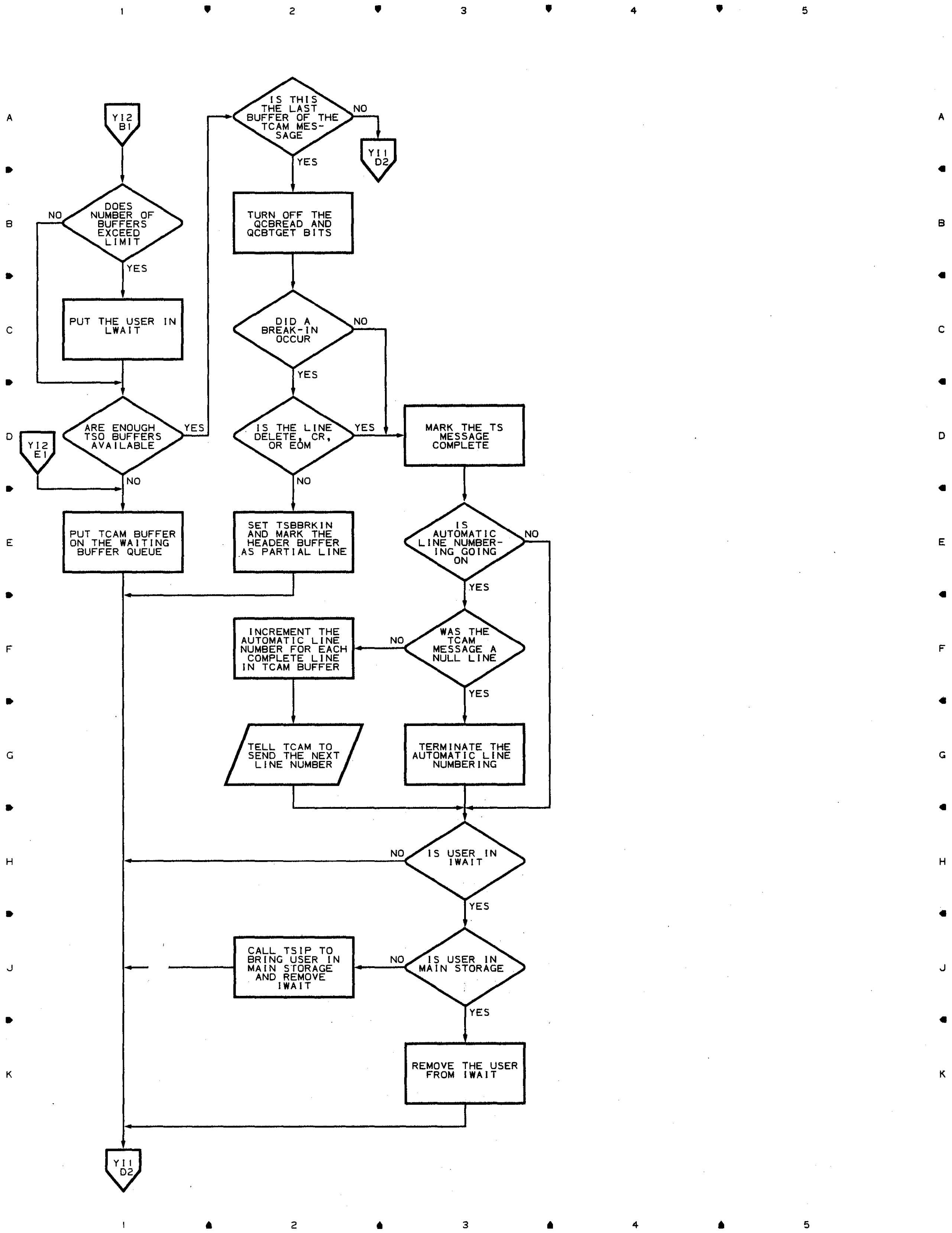

IEDAYH
ENTER
GET THE NEEDED ADDRESSES FOR BOTH ENTRY POINTS
IS THIS A TSO USER'S QCB
NO
ENTRY2 TAKEN (HANGUP)
YES
EXIT TO DSPDISP
NO
YES
SEEIFERR
D2
HANGITUP
ENTRY2 TAKEN (HANGUP)
YES
FIND THE ASSOCIATED TJB AND TSB
QCB, TJB, OR TSB BEEN THRU BFR OR HANGUP
NO
TELL TCAM THAT A HANGUP IS OCCURRING
NO
YES
IS THERE A DISCONNECT ERROR
YES
QTIP - ENTRY 4 TO TELL TSO A HANGUP IS OCCURRING
NO
CAN IEDAYH HANDLE THE ERROR
NO
SET THE QCB RETRY COUNT EQUAL TO ZERO
CLEAN UP THE TCAM CONTROL BLOCKS
YES
CHKMODE
SENDMODE
AUREVOIR
IS THE LINE SENDING
YES
IS THE RETRY COUNT NEEDED
NO
IS THE TERMINAL INOPERATIVE (SCBTMINN ON)
YES
ENTRY1 TAKEN
YES
NO
YES
H3
NO
D2
NO
MSG
FORGET
SHOULD A MESSAGE BE SENT
YES
TELL TCAM TO INFORM THE USER OF PROCESSING DONE
TURN OFF THE 'MESSAGE SENDING' BITS
TPOST THE ERB TO THE BUFFER DISPOSITION QCB
NO
COUNT
CHKCANCL
SAYONARA
IS THE RETRY COUNT NEEDED
YES
SET UP OR DECREMENT THE QCB RETRY COUNT
IS THE LINE SENDING
YES
IS MSGGEN IN PROGRESS
YES
PUT THE ERB ADDRESS IN REGISTER 1
NO
NO
NO
CANCEL
SHOULD THE MESSAGE BE CANCELED
YES
CANCEL THIS INCOMING MESSAGE
SET THE LINE TO RECEIVE MODE
EXIT TO DSPCHAIN
NO
H3
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart YI-1 TSINPUT ROUTINE



IEDAYI+2
ENTER
IS A TCAM BUFFER BEING PASSED
NO
YES
TURN OFF THE TSI POST BIT
CANCEL MESSAGE (LCBCNCLN ON)
NO
YES
YI1 D2
FREE ALL TCAM AND TSO BUFFERS FOR THIS MESSAGE
ARE THERE ANY BUFFERS ON THE WAIT QUEUE
NO
YES
IS THE SYSTEM IN LWAIT
NO
YES
ENOUGH BUFFERS TO REMOVE SLWAIT AVAIL-ABLE
NO
YES
TAKE THE SYSTEM OUT OF LWAIT; TURN OFF THE 'TIOCSYLW' AND 'QCBNOBUF' BITS
TAKE ALL TSBS OUT OF TSITSBQ AND TURN OFF ALL QCBTSBQ BITS
DEQUEUE A TCAM BUFFER CHAIN AND RESET THE QCBBUFQ BIT
YES
ARE THERE ANY FREE TSO BUFFERS
NO
PUT THE SYSTEM IN LWAIT
PUT THIS TSB IN THE TSITSBQ QUEUE AND TURN ON THE QCBTSBQ BIT
RETURN THE FREED TCAM BUFFERS AND ANY QCBS BEING TPOSTED
EXIT TO DSPCHAIN
IS THIS BEGINNING OF A TCAM MESSAGE
YES
NO
TURN OFF THE TSBIFLSH AND TSBBIPI BITS
BUFFER ALLOCATION-GET INITIAL TSO HEADER BUFFER
IS TSBIFLSH ON
NO
YES
IS A PARTIALLY FILLED BUFFER FOR THIS MSG
NO
YES
WERE ANY TSO BUFFERS AVAILABLE
NO
YES
YI2 E1
IS THIS LAST TCAM BUFFER OF THE MESSAGE
NO
YES
SCAN AND REMOVE DATA FROM TCAM TO TSO BUFFERS
YI2 B1
TURN OFF THE TSBIFLSH AND TSBBIPI BITS
FREE THIS TCAM BUFFER
D2
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart YI-2 TSINPUT ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
YI2 B1
DOES NUMBER OF BUFFERS EXCEED LIMIT
NO
YES
PUT THE USER IN LWAIT
ARE ENOUGH TSO BUFFERS AVAILABLE
YES
NO
YI2 E1
PUT TCAM BUFFER ON THE WAITING BUFFER QUEUE
IS THIS THE LAST BUFFER OF THE TCAM MES-SAGE
NO
YI1 D2
YES
TURN OFF THE QCBREAD AND QCBTGET BITS
DID A BREAK-IN OCCUR
NO
YES
IS THE LINE DELETE, CR, OR EOM
YES
NO
SET TSBBRKIN AND MARK THE HEADER BUFFER AS PARTIAL LINE
MARK THE TS MESSAGE COMPLETE
IS AUTOMATIC LINE NUMBER-ING GOING ON
NO
YES
WAS THE TCAM MESSAGE A NULL LINE
NO
YES
INCREMENT THE AUTOMATIC LINE NUMBER FOR EACH COMPLETE LINE IN TCAM BUFFER
TELL TCAM TO SEND THE NEXT LINE NUMBER
TERMINATE THE AUTOMATIC LINE NUMBERING
IS USER IN IWAIT
NO
YES
IS USER IN MAIN STORAGE
NO
YES
CALL TSIP TO BRING USER IN MAIN STORAGE AND REMOVE IWAIT
REMOVE THE USER FROM IWAIT
YI1 D2

Chart YL-1 TSO LOGON ROUTINE

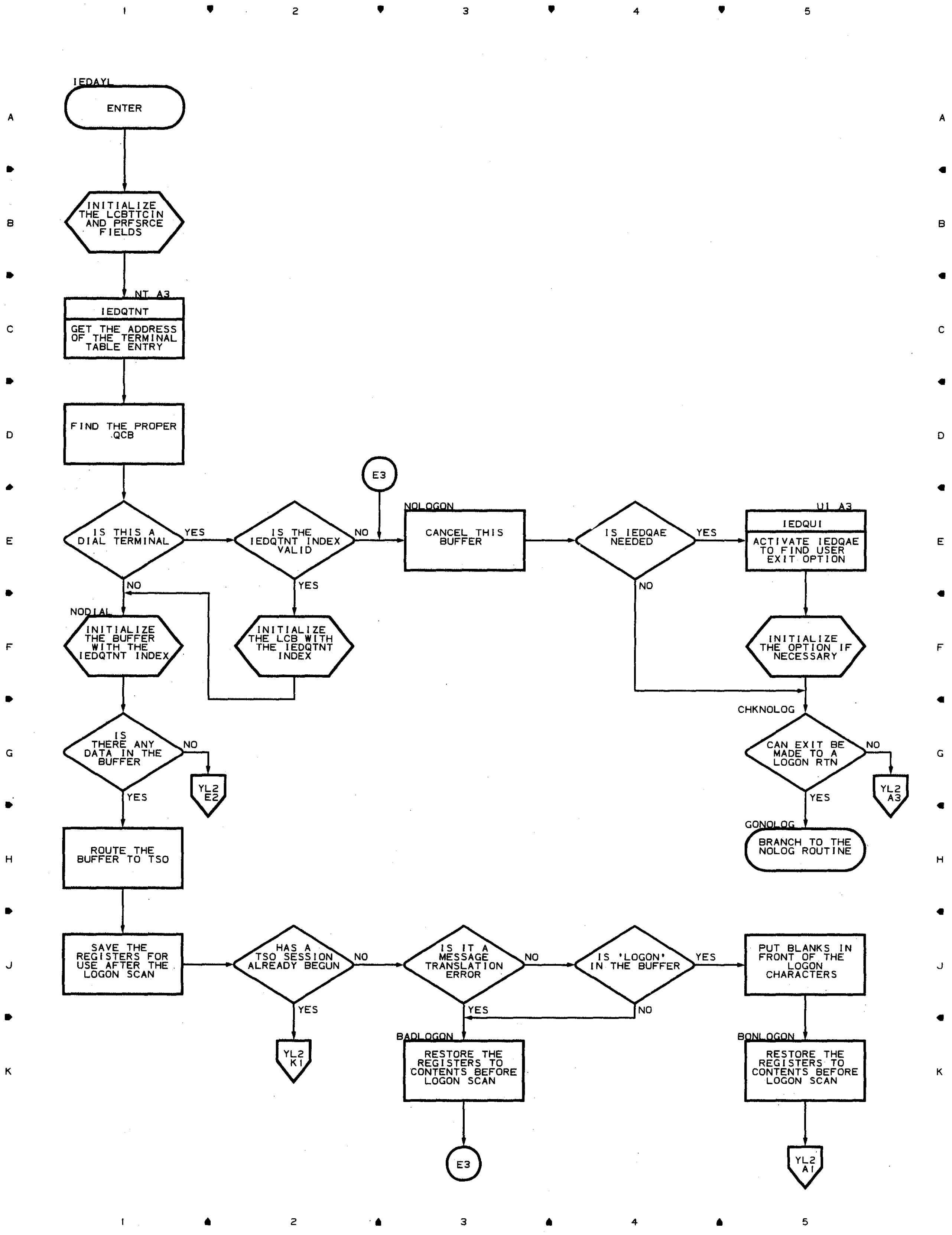
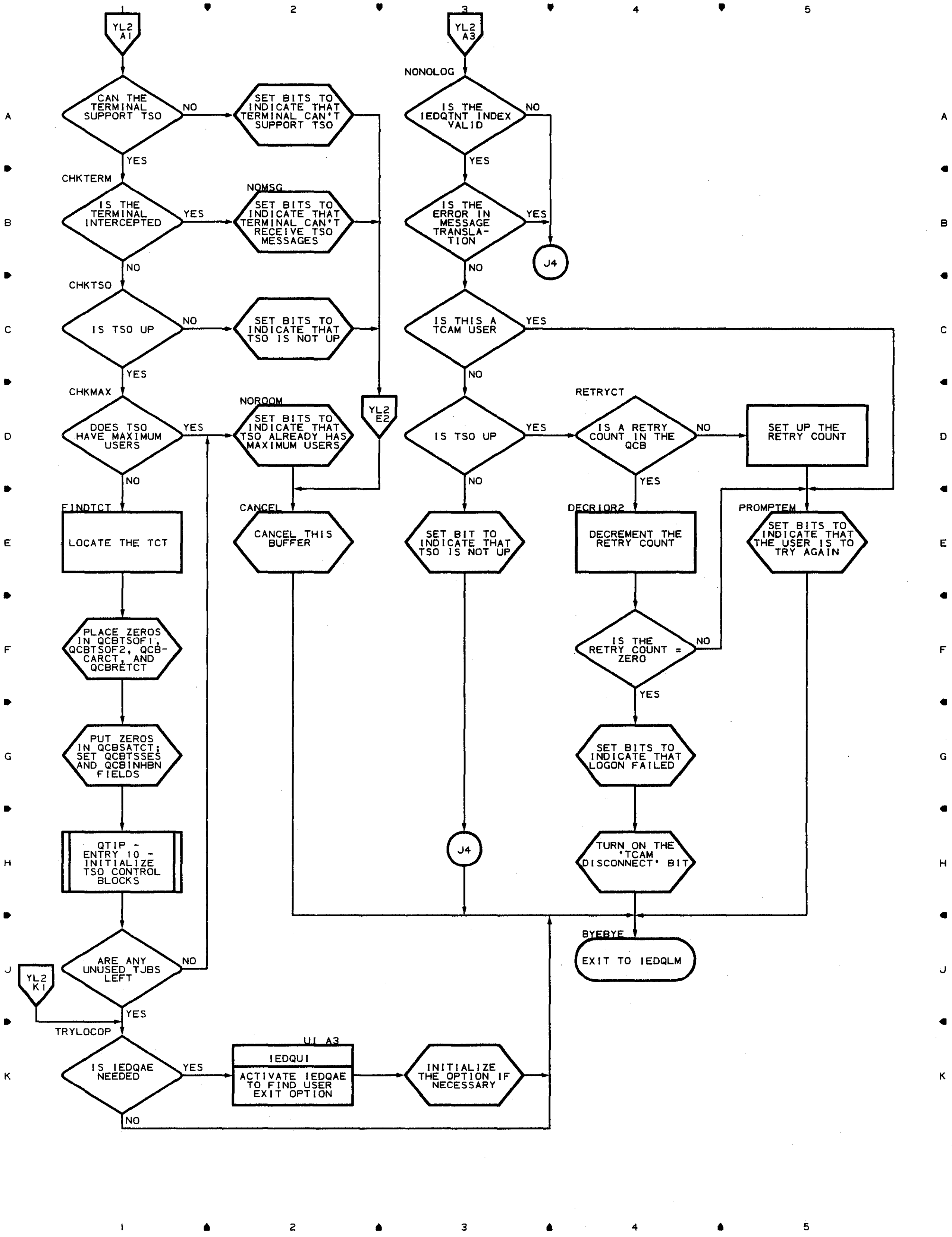
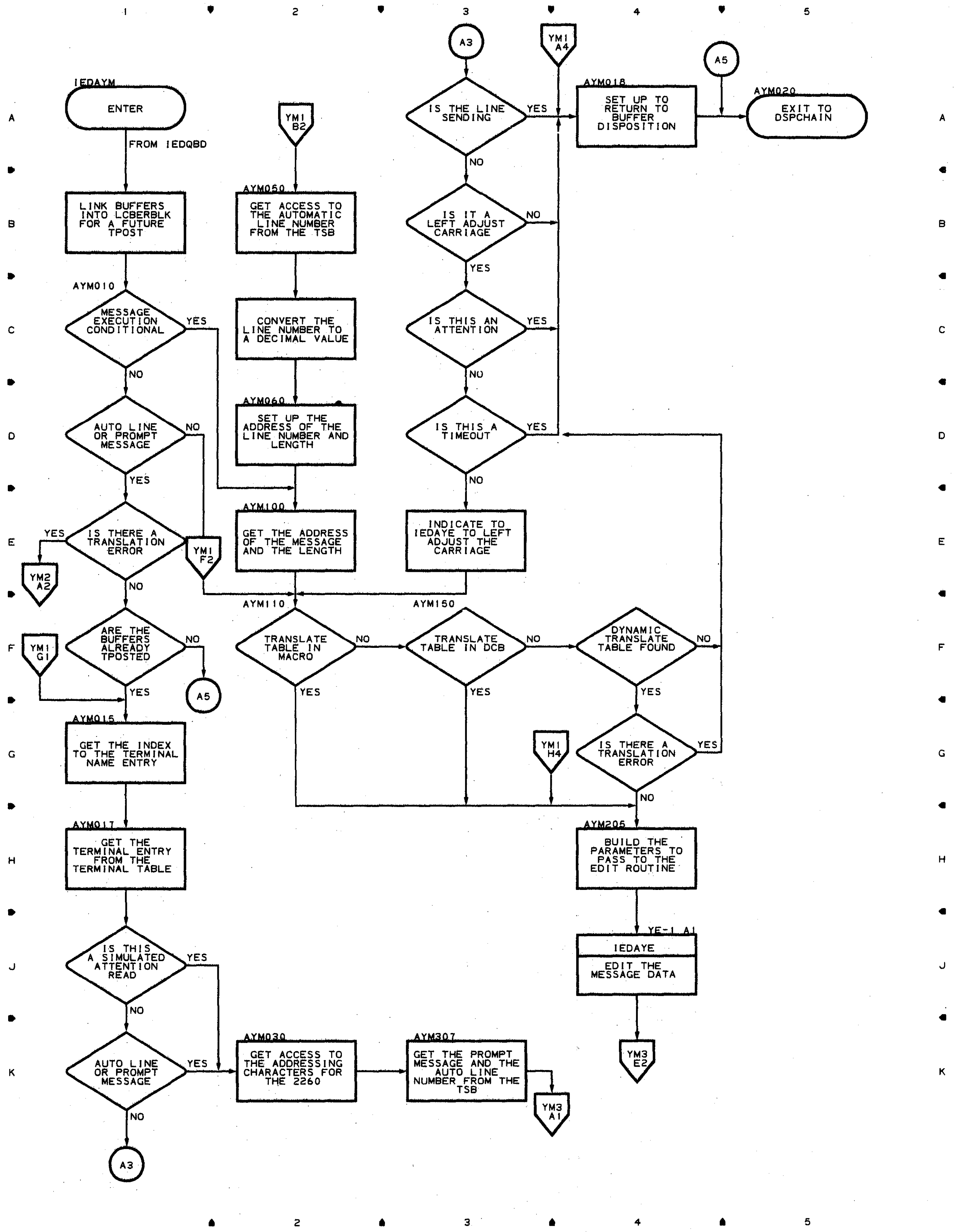

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDAYL
ENTER
INITIALIZE THE LCBTTCIN AND PRFSRCE FIELDS
NT A3
IEDQTNT
GET THE ADDRESS OF THE TERMINAL TABLE ENTRY
FIND THE PROPER QCB
E3
IS THIS A DIAL TERMINAL
YES
IS THE IEDQTNT INDEX VALID
NO
NOLOGON
CANCEL THIS BUFFER
IS IEDQAE NEEDED
YES
UI A3
IEDQUI
ACTIVATE IEDQAE TO FIND USER EXIT OPTION
NO
YES
NO
NODIAL
INITIALIZE THE BUFFER WITH THE IEDQTNT INDEX
INITIALIZE THE LCB WITH THE IEDQTNT INDEX
INITIALIZE THE OPTION IF NECESSARY
CHKNOLOG
IS THERE ANY DATA IN THE BUFFER
NO
YL2 E2
CAN EXIT BE MADE TO A LOGON RTN
NO
YL2 A3
YES
YES
GONOLOG
BRANCH TO THE NOLOG ROUTINE
ROUTE THE BUFFER TO TSO
SAVE THE REGISTERS FOR USE AFTER THE LOGON SCAN
HAS A TSO SESSION ALREADY BEGUN
NO
IS IT A MESSAGE TRANSLATION ERROR
NO
IS 'LOGON' IN THE BUFFER
YES
PUT BLANKS IN FRONT OF THE LOGON CHARACTERS
YES
YES
NO
BADLOGON
BONLOGON
YL2 K1
RESTORE THE REGISTERS TO CONTENTS BEFORE LOGON SCAN
RESTORE THE REGISTERS TO CONTENTS BEFORE LOGON SCAN
E3
YL2 A1

Chart YL-2 TSO LOGON ROUTINE



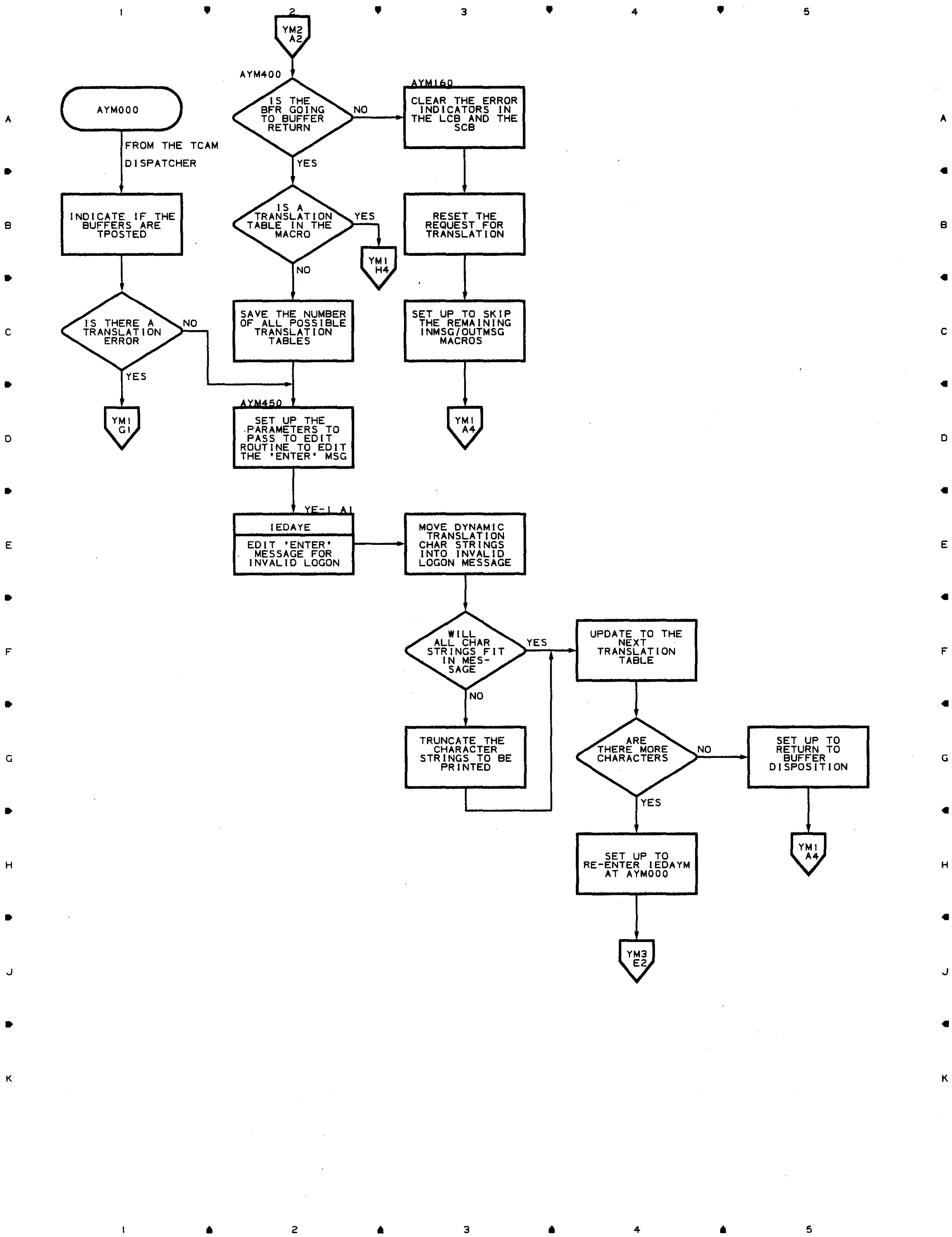
YL2 A1
CAN THE TERMINAL SUPPORT TSO
NO
SET BITS TO INDICATE THAT TERMINAL CAN'T SUPPORT TSO
YES
CHKTERM
IS THE TERMINAL INTERCEPTED
YES
NOMSG
SET BITS TO INDICATE THAT TERMINAL CAN'T RECEIVE TSO MESSAGES
NO
CHKTSO
IS TSO UP
NO
SET BITS TO INDICATE THAT TSO IS NOT UP
YES
CHKMAX
DOES TSO HAVE MAXIMUM USERS
YES
NOROOM
SET BITS TO INDICATE THAT TSO ALREADY HAS MAXIMUM USERS
YL2 E2
NO
FINDTCT
LOCATE THE TCT
CANCEL
CANCEL THIS BUFFER
PLACE ZEROS IN QCBTSOF1, QCBTSOF2, QCB-CARCT, AND QCBRETCT
PUT ZEROS IN QCBSATCT; SET QCBTSSES AND QCBINHBN FIELDS
QTIP - ENTRY 10 - INITIALIZE TSO CONTROL BLOCKS
ARE ANY UNUSED TJBS LEFT
NO
YL2 K1
YES
TRYLOCOP
IS IEDQAE NEEDED
YES
UI A3
IEDQUI
ACTIVATE IEDQAE TO FIND USER EXIT OPTION
INITIALIZE THE OPTION IF NECESSARY
NO
YL2 A3
NONOLOG
IS THE IEDQTNT INDEX VALID
NO
YES
IS THE ERROR IN MESSAGE TRANSLA-TION
YES
J4
NO
IS THIS A TCAM USER
YES
NO
IS TSO UP
YES
RETRYCT
IS A RETRY COUNT IN THE QCB
NO
SET UP THE RETRY COUNT
YES
NO
SET BIT TO INDICATE THAT TSO IS NOT UP
DECRIOR2
DECREMENT THE RETRY COUNT
PROMPTEM
SET BITS TO INDICATE THAT THE USER IS TO TRY AGAIN
IS THE RETRY COUNT = ZERO
NO
YES
SET BITS TO INDICATE THAT LOGON FAILED
J4
TURN ON THE 'TCAM DISCONNECT' BIT
BYEBYE
EXIT TO IEDQLM
1 2 3 4 5
A B C D E F G H J K

Chart YM-1 TSO MESSAGE GENERATION ROUTINE



IEDAYM
ENTER
FROM IEDQBD
LINK BUFFERS INTO LCBERBLK FOR A FUTURE TPOST
AYM010
MESSAGE EXECUTION CONDITIONAL
YES
NO
AUTO LINE OR PROMPT MESSAGE
NO
YES
IS THERE A TRANSLATION ERROR
YES
YM2 A2
YM1 F2
NO
ARE THE BUFFERS ALREADY TPOSTED
NO
A5
YM1 G1
YES
AYM015
GET THE INDEX TO THE TERMINAL NAME ENTRY
AYM017
GET THE TERMINAL ENTRY FROM THE TERMINAL TABLE
IS THIS A SIMULATED ATTENTION READ
YES
NO
AUTO LINE OR PROMPT MESSAGE
YES
NO
A3
YM1 B2
AYM050
GET ACCESS TO THE AUTOMATIC LINE NUMBER FROM THE TSB
CONVERT THE LINE NUMBER TO A DECIMAL VALUE
AYM060
SET UP THE ADDRESS OF THE LINE NUMBER AND LENGTH
AYM100
GET THE ADDRESS OF THE MESSAGE AND THE LENGTH
AYM110
TRANSLATE TABLE IN MACRO
NO
YES
AYM030
GET ACCESS TO THE ADDRESSING CHARACTERS FOR THE 2260
A3
IS THE LINE SENDING
YES
NO
IS IT A LEFT ADJUST CARRIAGE
NO
YES
IS THIS AN ATTENTION
YES
NO
IS THIS A TIMEOUT
YES
NO
INDICATE TO IEDAYE TO LEFT ADJUST THE CARRIAGE
AYM150
TRANSLATE TABLE IN DCB
NO
YES
AYM307
GET THE PROMPT MESSAGE AND THE AUTO LINE NUMBER FROM THE TSB
YM3 A1
YM1 A4
AYM018
SET UP TO RETURN TO BUFFER DISPOSITION
DYNAMIC TRANSLATE TABLE FOUND
NO
YES
YM1 H4
IS THERE A TRANSLATION ERROR
YES
NO
AYM205
BUILD THE PARAMETERS TO PASS TO THE EDIT ROUTINE
YE-1 A1
IEDAYE
EDIT THE MESSAGE DATA
YM3 E2
A5
AYM020
EXIT TO DSPCHAIN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart YM-2 TSO MESSAGE GENERATION ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
YM2 A2
AYM400
IS THE BFR GOING TO BUFFER RETURN
NO
YES
AYM160
CLEAR THE ERROR INDICATORS IN THE LCB AND THE SCB
AYM000
FROM THE TCAM DISPATCHER
INDICATE IF THE BUFFERS ARE TPOSTED
IS A TRANSLATION TABLE IN THE MACRO
YES
NO
YM1 H4
RESET THE REQUEST FOR TRANSLATION
IS THERE A TRANSLATION ERROR
NO
YES
SAVE THE NUMBER OF ALL POSSIBLE TRANSLATION TABLES
SET UP TO SKIP THE REMAINING INMSG/OUTMSG MACROS
YM1 G1
AYM450
SET UP THE PARAMETERS TO PASS TO EDIT ROUTINE TO EDIT THE 'ENTER' MSG
YM1 A4
YE-1 A1
IEDAYE
EDIT 'ENTER' MESSAGE FOR INVALID LOGON
MOVE DYNAMIC TRANSLATION CHAR STRINGS INTO INVALID LOGON MESSAGE
WILL ALL CHAR STRINGS FIT IN MES- SAGE
YES
NO
UPDATE TO THE NEXT TRANSLATION TABLE
TRUNCATE THE CHARACTER STRINGS TO BE PRINTED
ARE THERE MORE CHARACTERS
NO
YES
SET UP TO RETURN TO BUFFER DISPOSITION
SET UP TO RE-ENTER IEDAYM AT AYM000
YM1 A4
YM3 E2

Chart YM-3 TSO MESSAGE GENERATION ROUTINE

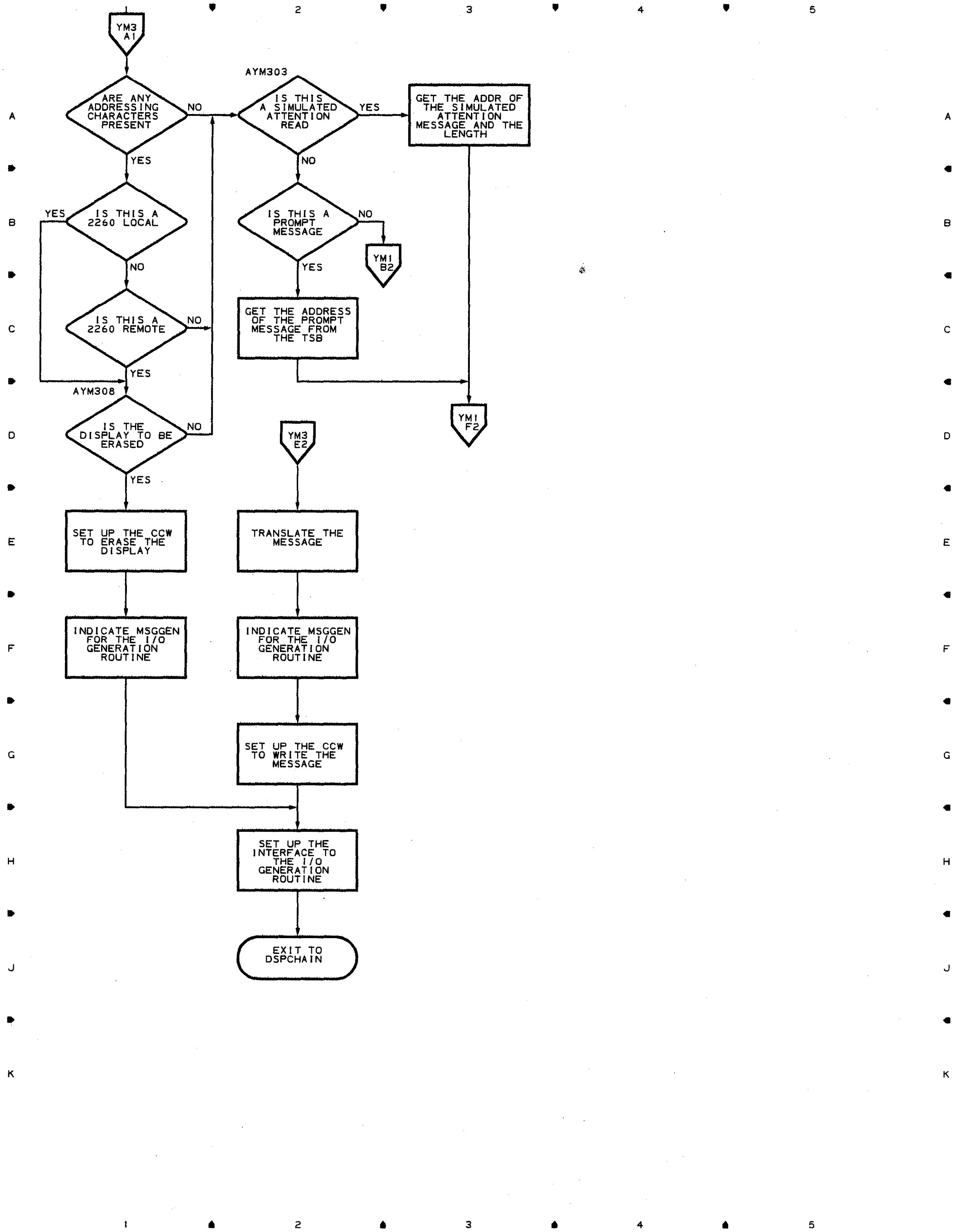
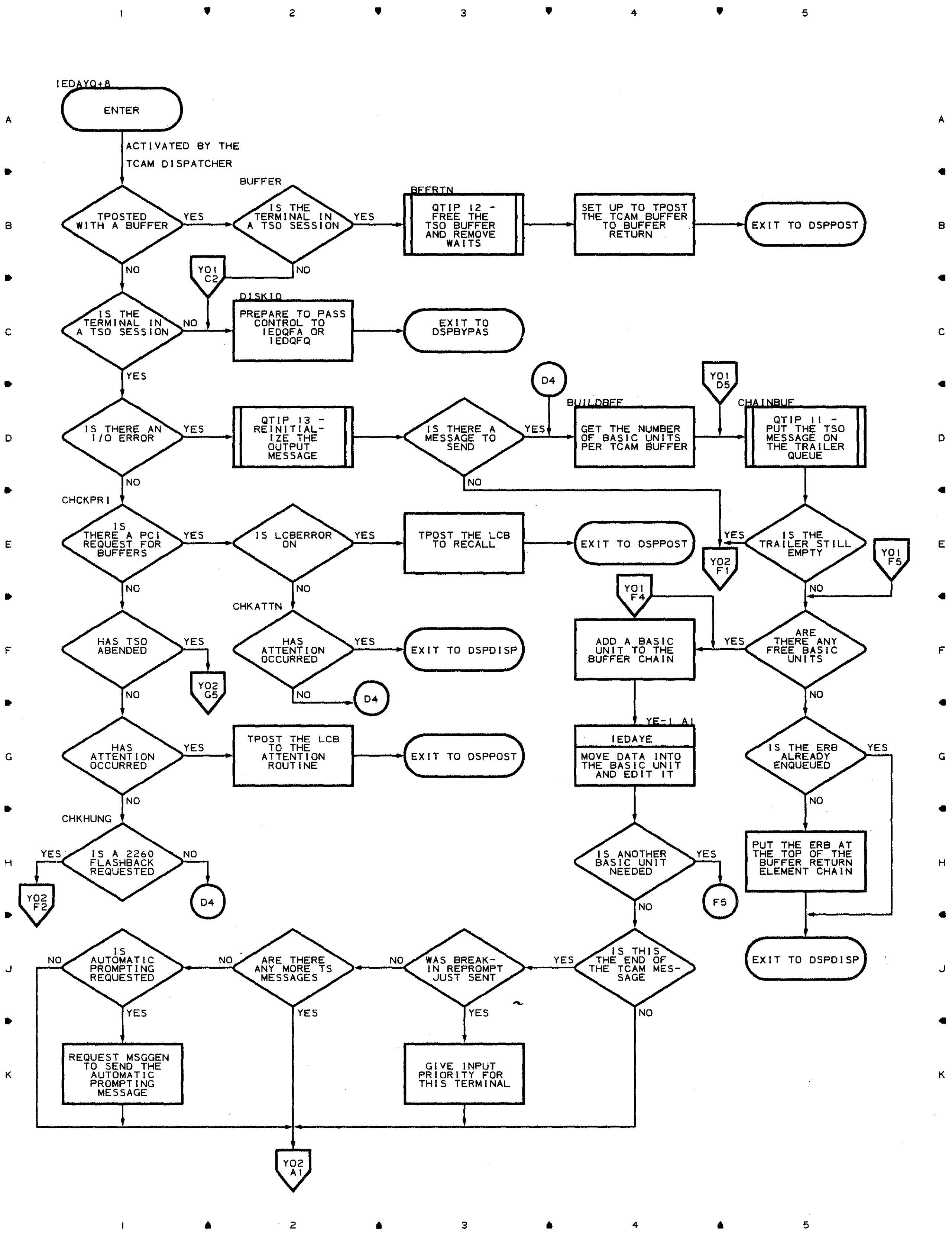

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
YM3 A1
ARE ANY ADDRESSING CHARACTERS PRESENT
NO
YES
AYM303
IS THIS A SIMULATED ATTENTION READ
YES
NO
GET THE ADDR OF THE SIMULATED ATTENTION MESSAGE AND THE LENGTH
IS THIS A 2260 LOCAL
YES
NO
IS THIS A PROMPT MESSAGE
NO
YES
YM1 B2
IS THIS A 2260 REMOTE
NO
YES
GET THE ADDRESS OF THE PROMPT MESSAGE FROM THE TSB
AYM308
IS THE DISPLAY TO BE ERASED
NO
YES
YM3 E2
YM1 F2
SET UP THE CCW TO ERASE THE DISPLAY
TRANSLATE THE MESSAGE
INDICATE MSGGEN FOR THE I/O GENERATION ROUTINE
INDICATE MSGGEN FOR THE I/O GENERATION ROUTINE
SET UP THE CCW TO WRITE THE MESSAGE
SET UP THE INTERFACE TO THE I/O GENERATION ROUTINE
EXIT TO DSPCHAIN

Chart YO-1 TSOUTPUT ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDAYO+8
ENTER
ACTIVATED BY THE TCAM DISPATCHER
TPOSTED WITH A BUFFER
YES
NO
BUFFER
IS THE TERMINAL IN A TSO SESSION
YES
NO
BFFRTN
QTIP 12 - FREE THE TSO BUFFER AND REMOVE WAITS
SET UP TO TPOST THE TCAM BUFFER TO BUFFER RETURN
EXIT TO DSPPOST
YO1 C2
IS THE TERMINAL IN A TSO SESSION
NO
YES
DISKIO
PREPARE TO PASS CONTROL TO IEDQFA OR IEDQFQ
EXIT TO DSPBYPAS
D4
YO1 D5
IS THERE AN I/O ERROR
YES
NO
QTIP 13 - REINITIAL-IZE THE OUTPUT MESSAGE
IS THERE A MESSAGE TO SEND
YES
NO
BUILDBFF
GET THE NUMBER OF BASIC UNITS PER TCAM BUFFER
CHAINBUF
QTIP 11 - PUT THE TSO MESSAGE ON THE TRAILER QUEUE
CHCKPRI
IS THERE A PCI REQUEST FOR BUFFERS
YES
NO
IS LCBERROR ON
YES
NO
TPOST THE LCB TO RECALL
EXIT TO DSPPOST
YES
YO2 F1
IS THE TRAILER STILL EMPTY
NO
YO1 F5
YO1 F4
CHKATTN
HAS TSO ABENDED
YES
NO
YO2 G5
HAS ATTENTION OCCURRED
YES
NO
EXIT TO DSPDISP
D4
ADD A BASIC UNIT TO THE BUFFER CHAIN
YES
ARE THERE ANY FREE BASIC UNITS
NO
YE-1 A1
IEDAYE
MOVE DATA INTO THE BASIC UNIT AND EDIT IT
HAS ATTENTION OCCURRED
YES
NO
TPOST THE LCB TO THE ATTENTION ROUTINE
EXIT TO DSPPOST
IS THE ERB ALREADY ENQUEUED
YES
NO
PUT THE ERB AT THE TOP OF THE BUFFER RETURN ELEMENT CHAIN
CHKHUNG
IS A 2260 FLASHBACK REQUESTED
YES
NO
YO2 F2
D4
IS ANOTHER BASIC UNIT NEEDED
YES
NO
F5
EXIT TO DSPDISP
IS AUTOMATIC PROMPTING REQUESTED
NO
YES
ARE THERE ANY MORE TS MESSAGES
NO
YES
WAS BREAK-IN REPROMPT JUST SENT
NO
YES
IS THIS THE END OF THE TCAM MES-SAGE
YES
NO
REQUEST MSGGEN TO SEND THE AUTOMATIC PROMPTING MESSAGE
GIVE INPUT PRIORITY FOR THIS TERMINAL
YO2 A1

Chart YO-2 TSOUTPUT ROUTINE

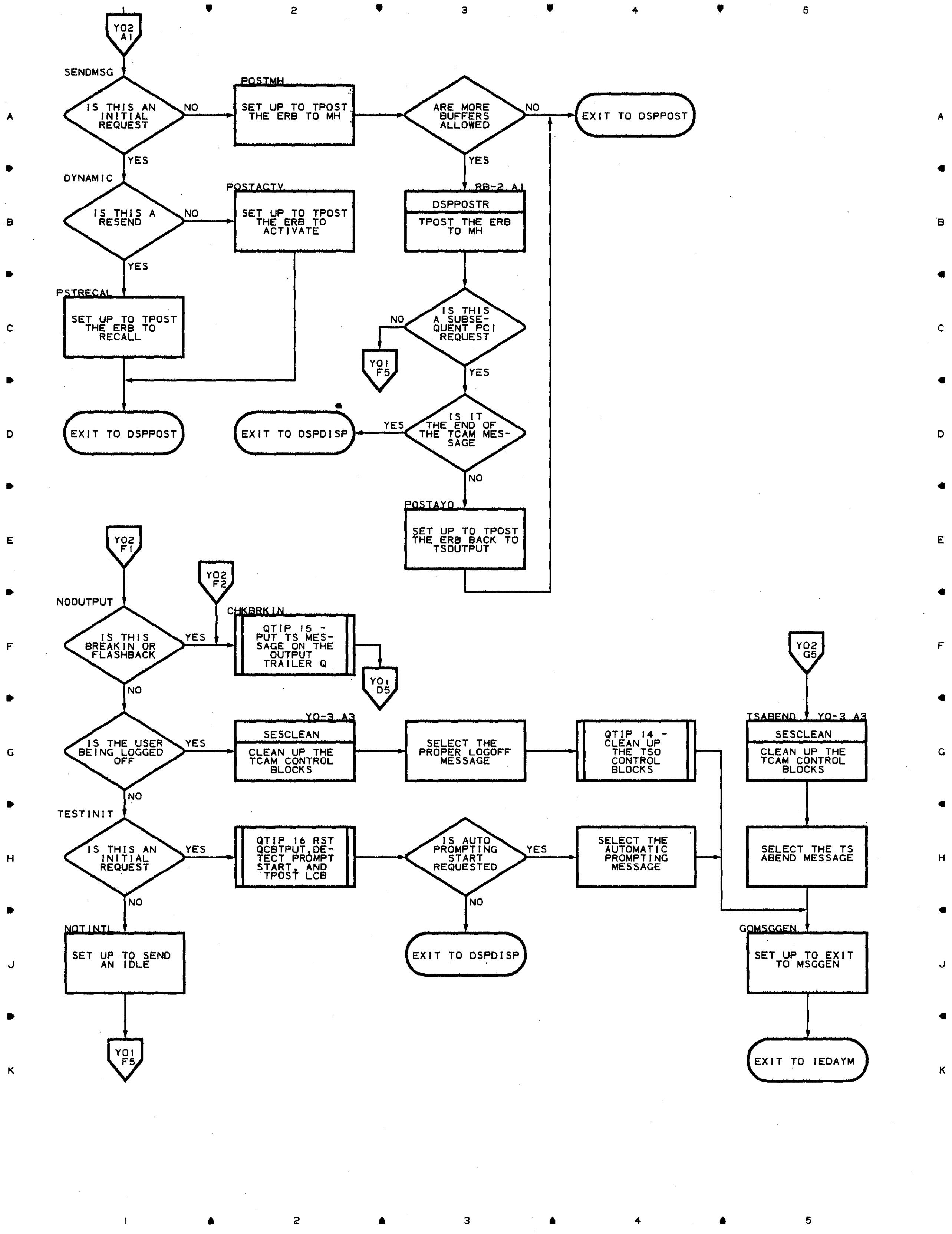
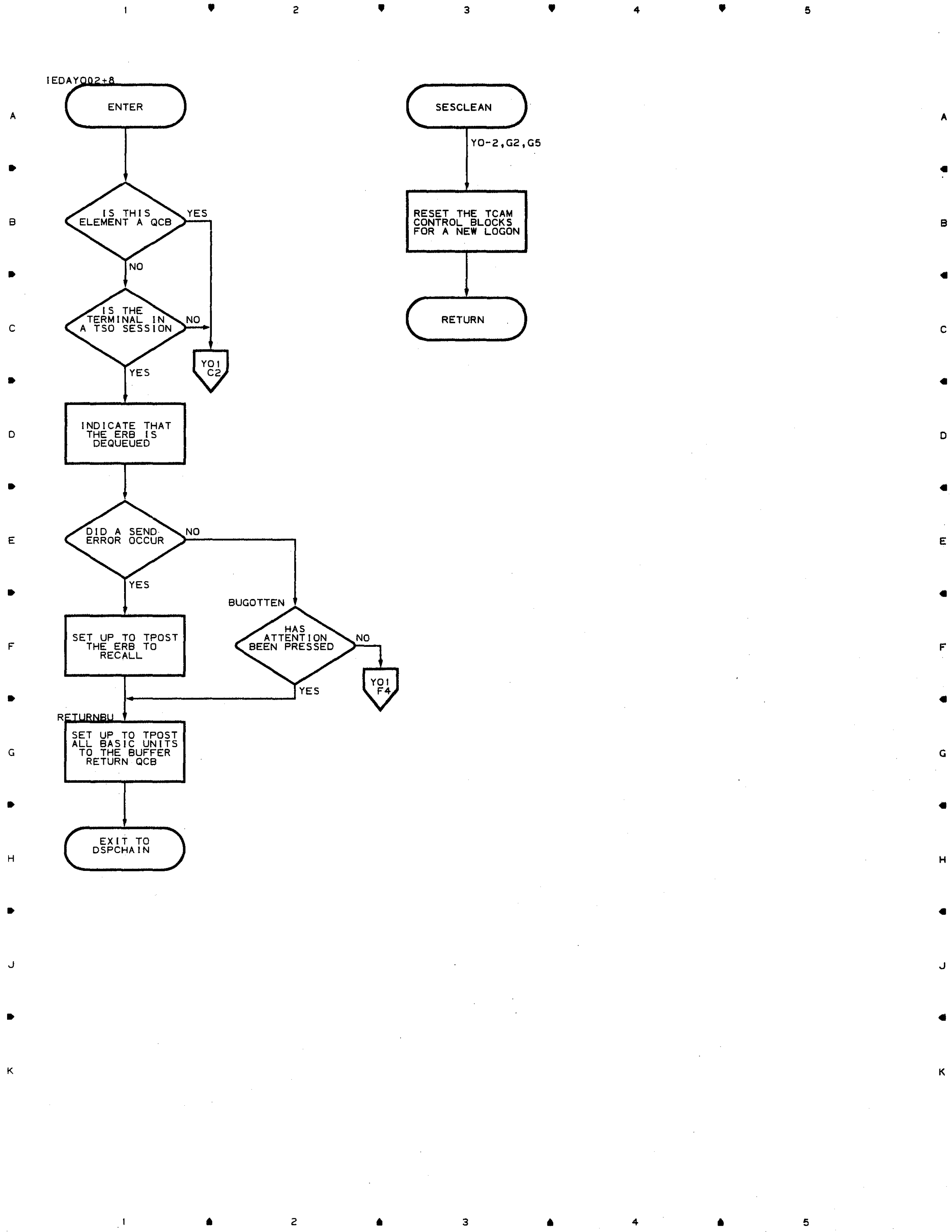

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
YO2 A1
SENDMSG
IS THIS AN INITIAL REQUEST
NO
YES
POSTMH
SET UP TO TPOST THE ERB TO MH
ARE MORE BUFFERS ALLOWED
NO
YES
EXIT TO DSPPOST
DYNAMIC
IS THIS A RESEND
NO
YES
POSTACTV
SET UP TO TPOST THE ERB TO ACTIVATE
RB-2 A1
DSPPOSTR
TPOST THE ERB TO MH
PSTRECAL
SET UP TO TPOST THE ERB TO RECALL
IS THIS A SUBSEQUENT PCI REQUEST
NO
YES
YO1 F5
EXIT TO DSPPOST
EXIT TO DSPDISP
YES
IS IT THE END OF THE TCAM MESSAGE
NO
POSTAYO
SET UP TO TPOST THE ERB BACK TO TSOUTPUT
YO2 F1
YO2 F2
NOOUTPUT
CHKBRKIN
IS THIS BREAKIN OR FLASHBACK
YES
QTIP 15 - PUT TS MESSAGE ON THE OUTPUT TRAILER Q
YO1 D5
NO
YO2 G5
YO-3 A3
SESCLEAN
IS THE USER BEING LOGGED OFF
YES
CLEAN UP THE TCAM CONTROL BLOCKS
SELECT THE PROPER LOGOFF MESSAGE
QTIP 14 - CLEAN UP THE TSO CONTROL BLOCKS
TSABEND
YO-3 A3
SESCLEAN
CLEAN UP THE TCAM CONTROL BLOCKS
NO
TESTINIT
IS THIS AN INITIAL REQUEST
YES
QTIP 16 RST QCBTPUT,DETECT PROMPT START, AND TPOST LCB
IS AUTO PROMPTING START REQUESTED
YES
SELECT THE AUTOMATIC PROMPTING MESSAGE
SELECT THE TS ABEND MESSAGE
NO
NO
NOTINTL
SET UP TO SEND AN IDLE
EXIT TO DSPDISP
GOMSGGEN
SET UP TO EXIT TO MSGGEN
YO1 F5
EXIT TO IEDAYM

Chart YO-3 TSOUTPUT ROUTINE



IEDAYO02+8
ENTER
IS THIS ELEMENT A QCB
YES
NO
IS THE TERMINAL IN A TSO SESSION
NO
YES
YO1 C2
INDICATE THAT THE ERB IS DEQUEUED
DID A SEND ERROR OCCUR
NO
YES
BUGOTTEN
HAS ATTENTION BEEN PRESSED
NO
YES
YO1 F4
SET UP TO TPOST THE ERB TO RECALL
RETURNBU
SET UP TO TPOST ALL BASIC UNITS TO THE BUFFER RETURN QCB
EXIT TO DSPCHAIN
SESCLEAN
YO-2,G2,G5
RESET THE TCAM CONTROL BLOCKS FOR A NEW LOGON
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart YR-1 STARTMH SUBTASK FOR TCAM-TSO MIXED

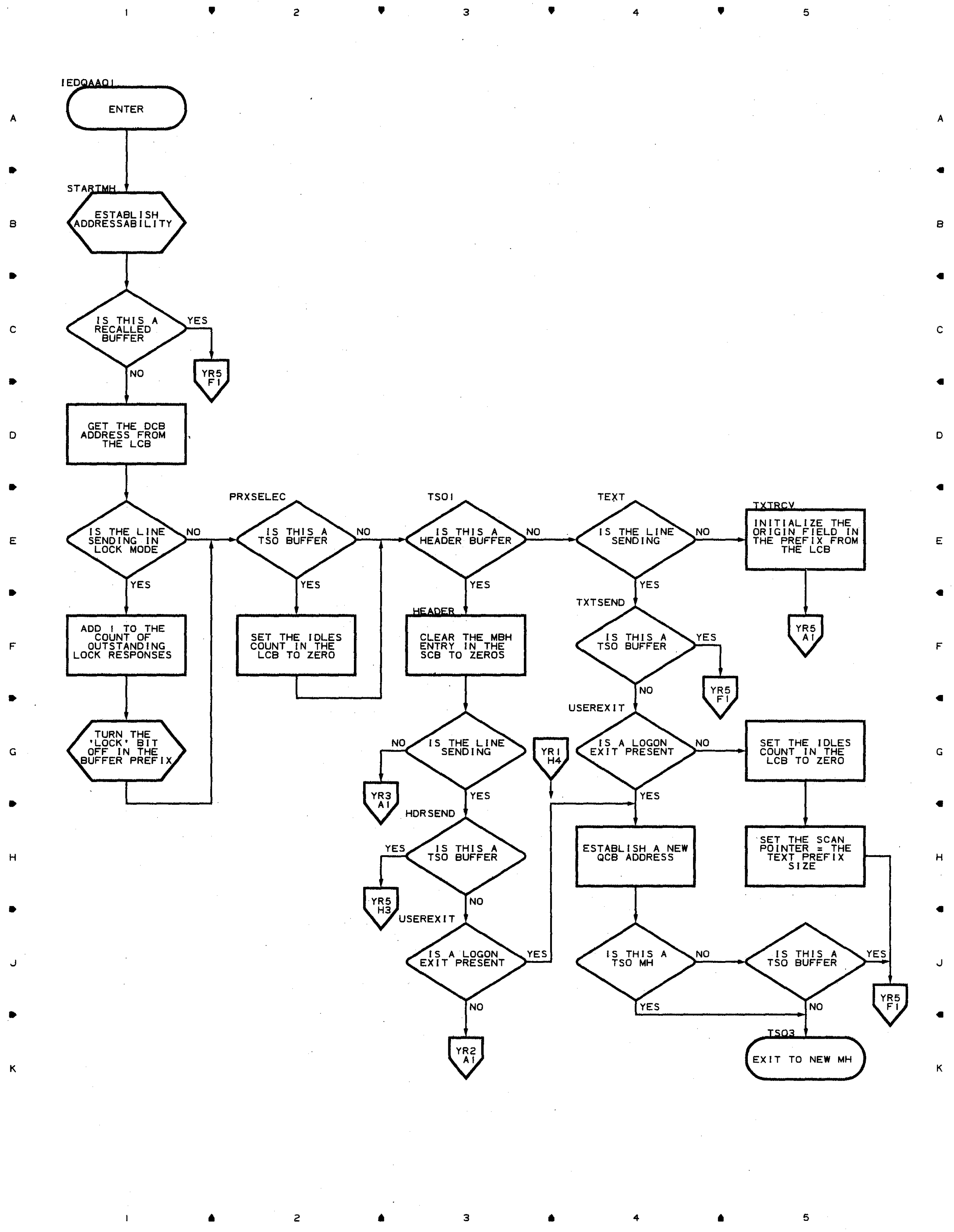


Chart YR-2 STARTMH SUBTASK FOR TCAM-TSO MIXED

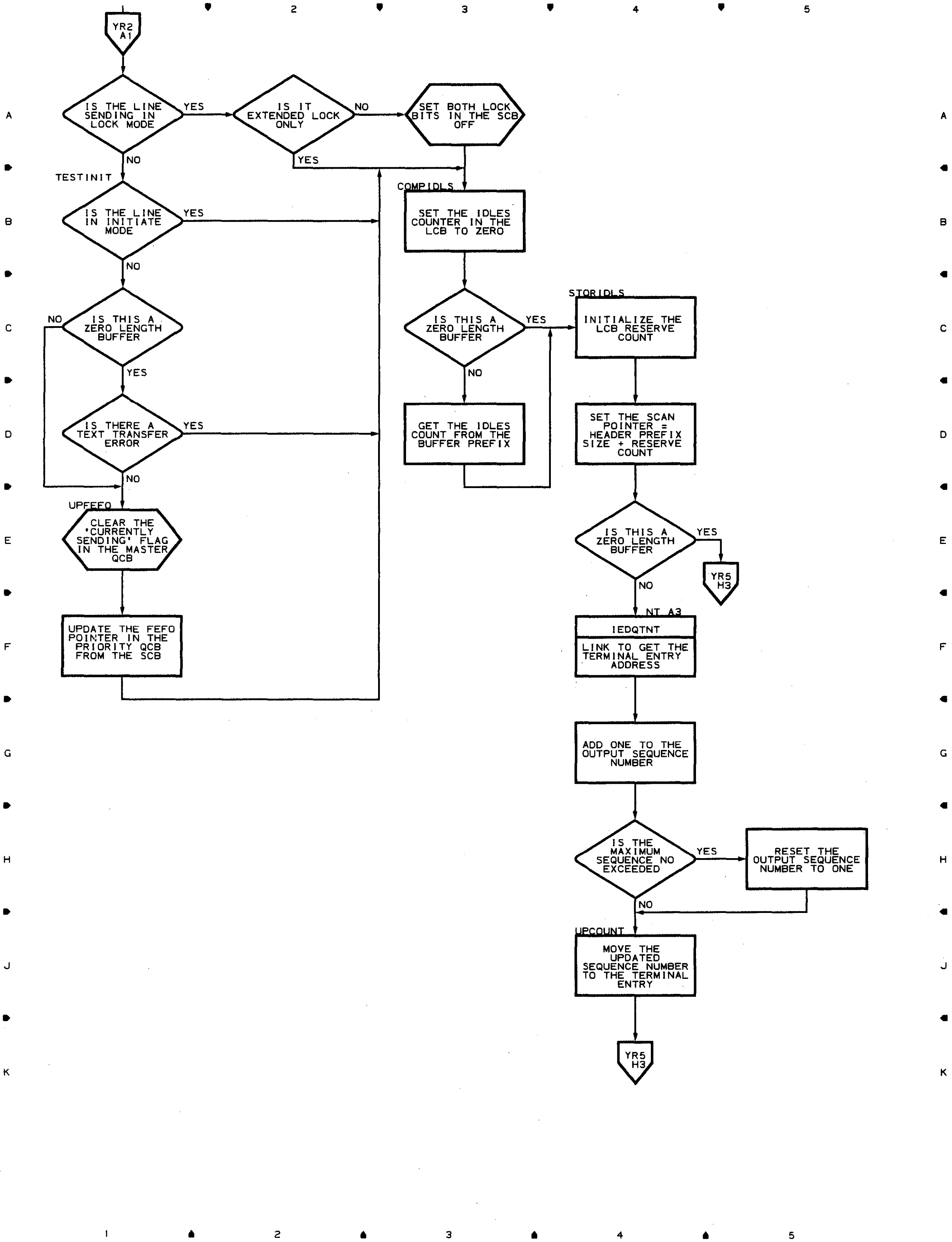

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
YR2 A1
IS THE LINE SENDING IN LOCK MODE
YES
IS IT EXTENDED LOCK ONLY
NO
SET BOTH LOCK BITS IN THE SCB OFF
YES
NO
TESTINIT
IS THE LINE IN INITIATE MODE
YES
COMPIDLS
SET THE IDLES COUNTER IN THE LCB TO ZERO
NO
IS THIS A ZERO LENGTH BUFFER
NO
YES
IS THIS A ZERO LENGTH BUFFER
YES
STORIDLS
INITIALIZE THE LCB RESERVE COUNT
NO
IS THERE A TEXT TRANSFER ERROR
YES
GET THE IDLES COUNT FROM THE BUFFER PREFIX
SET THE SCAN POINTER = HEADER PREFIX SIZE + RESERVE COUNT
NO
UPFEFO
CLEAR THE 'CURRENTLY SENDING' FLAG IN THE MASTER QCB
IS THIS A ZERO LENGTH BUFFER
YES
YR5 H3
NO
UPDATE THE FEFO POINTER IN THE PRIORITY QCB FROM THE SCB
NT A3
IEDQTNT
LINK TO GET THE TERMINAL ENTRY ADDRESS
ADD ONE TO THE OUTPUT SEQUENCE NUMBER
IS THE MAXIMUM SEQUENCE NO EXCEEDED
YES
RESET THE OUTPUT SEQUENCE NUMBER TO ONE
NO
UPCOUNT
MOVE THE UPDATED SEQUENCE NUMBER TO THE TERMINAL ENTRY
YR5 H3

Chart YR-3 STARTMH SUBTASK FOR TCAM-TSO MIXED

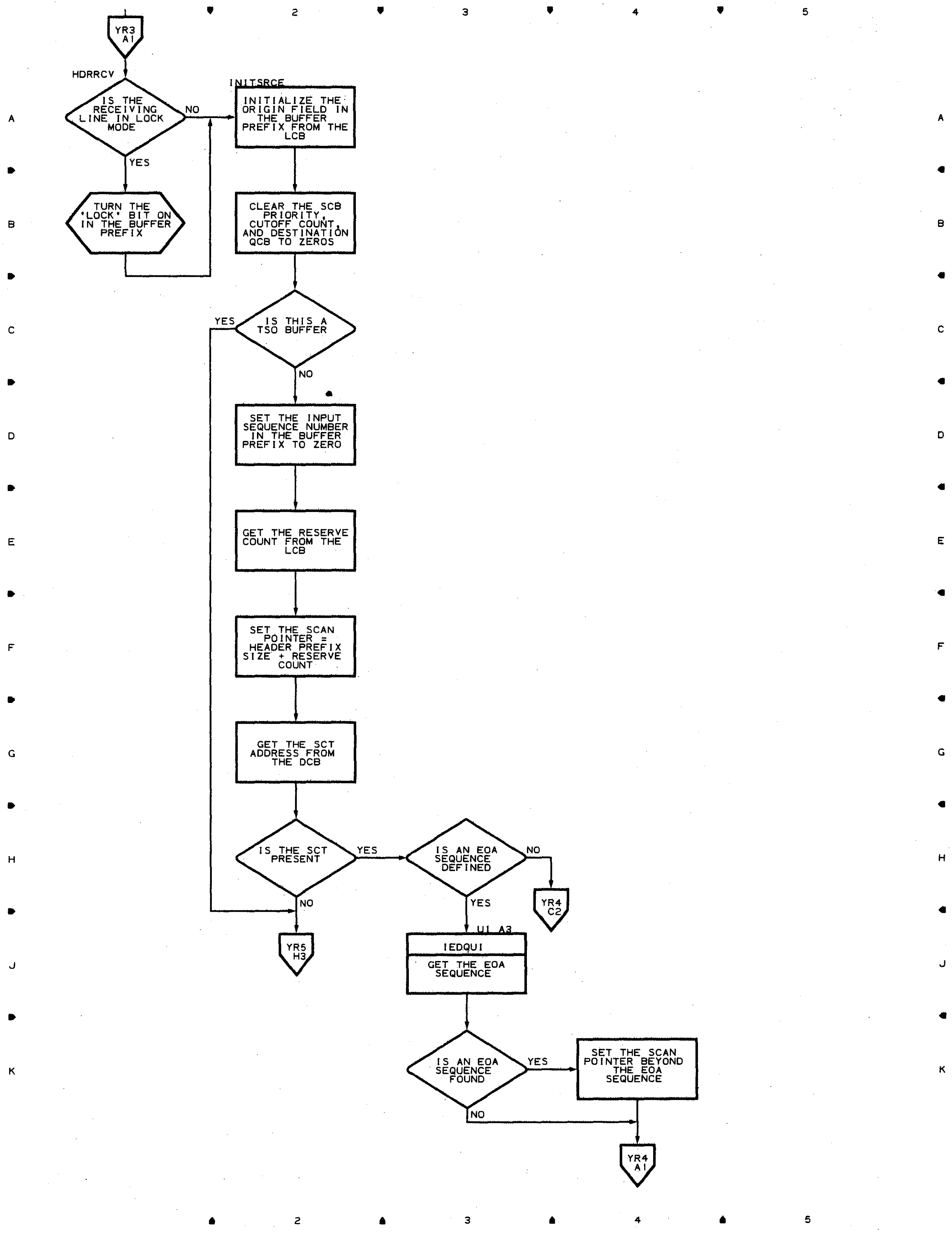
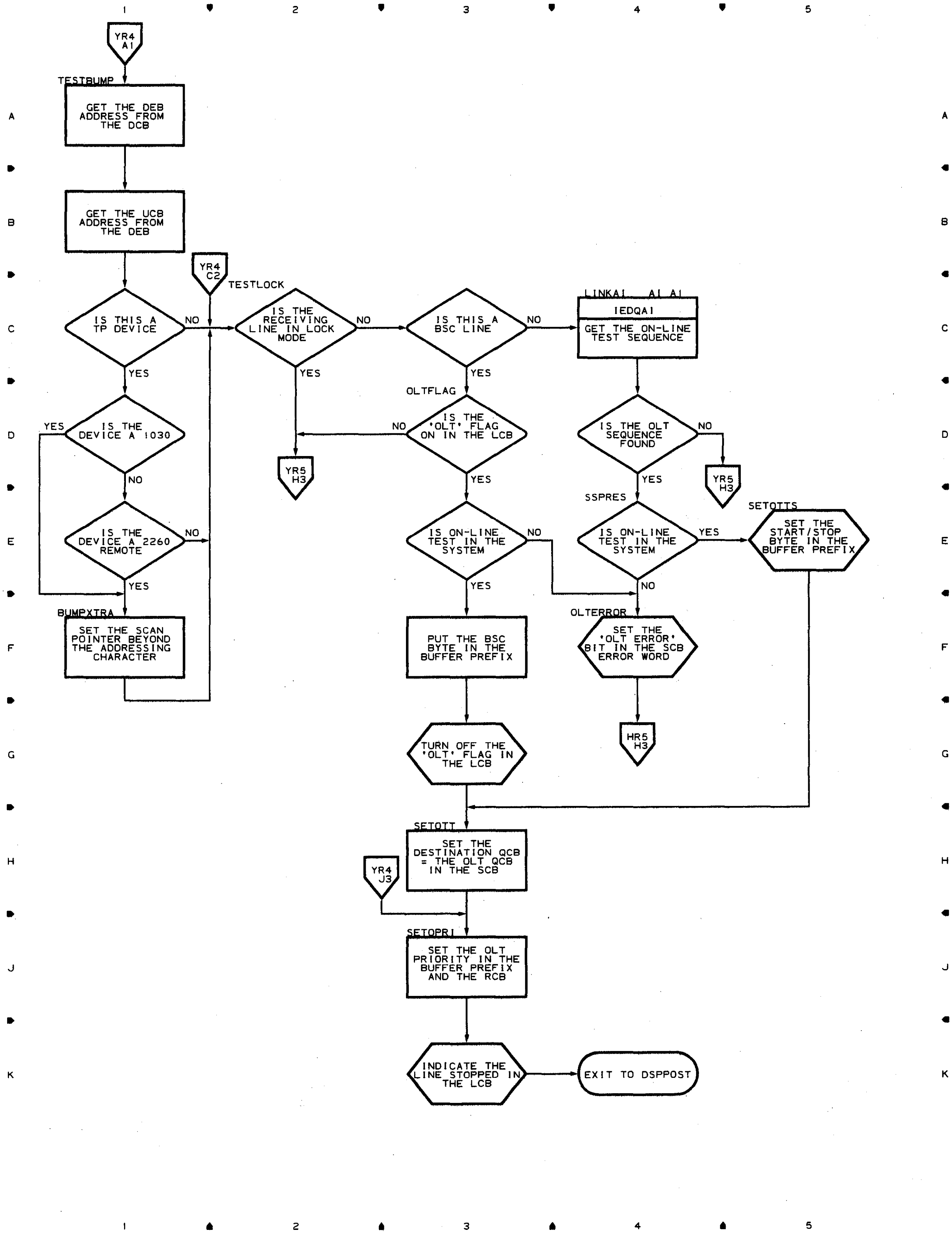

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
YR3 A1
HDRRCV
IS THE RECEIVING LINE IN LOCK MODE
NO
YES
TURN THE 'LOCK' BIT ON IN THE BUFFER PREFIX
INITSRCE
INITIALIZE THE ORIGIN FIELD IN THE BUFFER PREFIX FROM THE LCB
CLEAR THE SCB PRIORITY, CUTOFF COUNT, AND DESTINATION QCB TO ZEROS
IS THIS A TSO BUFFER
YES
NO
SET THE INPUT SEQUENCE NUMBER IN THE BUFFER PREFIX TO ZERO
GET THE RESERVE COUNT FROM THE LCB
SET THE SCAN POINTER = HEADER PREFIX SIZE + RESERVE COUNT
GET THE SCT ADDRESS FROM THE DCB
IS THE SCT PRESENT
YES
NO
YR5 H3
IS AN EOA SEQUENCE DEFINED
NO
YR4 C2
YES
UI A3
IEDQUI
GET THE EOA SEQUENCE
IS AN EOA SEQUENCE FOUND
YES
SET THE SCAN POINTER BEYOND THE EOA SEQUENCE
NO
YR4 A1

Chart YR-4 STARTMH SUBTASK FOR TCAM-TSO MIXED



YR4 A1
TESTBUMP
GET THE DEB ADDRESS FROM THE DCB
GET THE UCB ADDRESS FROM THE DEB
IS THIS A TP DEVICE
NO
YES
IS THE DEVICE A 1030
YES
NO
IS THE DEVICE A 2260 REMOTE
NO
YES
BUMPXTRA
SET THE SCAN POINTER BEYOND THE ADDRESSING CHARACTER
YR4 C2
TESTLOCK
IS THE RECEIVING LINE IN LOCK MODE
NO
YES
YR5 H3
IS THIS A BSC LINE
NO
YES
OLTFLAG
IS THE 'OLT' FLAG ON IN THE LCB
NO
YES
IS ON-LINE TEST IN THE SYSTEM
NO
YES
PUT THE BSC BYTE IN THE BUFFER PREFIX
TURN OFF THE 'OLT' FLAG IN THE LCB
LINKAI AI AI
IEDQAI
GET THE ON-LINE TEST SEQUENCE
IS THE OLT SEQUENCE FOUND
NO
YR5 H3
YES
SSPRES
IS ON-LINE TEST IN THE SYSTEM
YES
NO
SETOTTS
SET THE START/STOP BYTE IN THE BUFFER PREFIX
OLTERROR
SET THE 'OLT ERROR' BIT IN THE SCB ERROR WORD
HR5 H3
SETOTT
SET THE DESTINATION QCB = THE OLT QCB IN THE SCB
YR4 J3
SETOPRI
SET THE OLT PRIORITY IN THE BUFFER PREFIX AND THE RCB
INDICATE THE LINE STOPPED IN THE LCB
EXIT TO DSPPOST
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart YR-5 STARTMH SUBTASK FOR TCAM-TSO MIXED

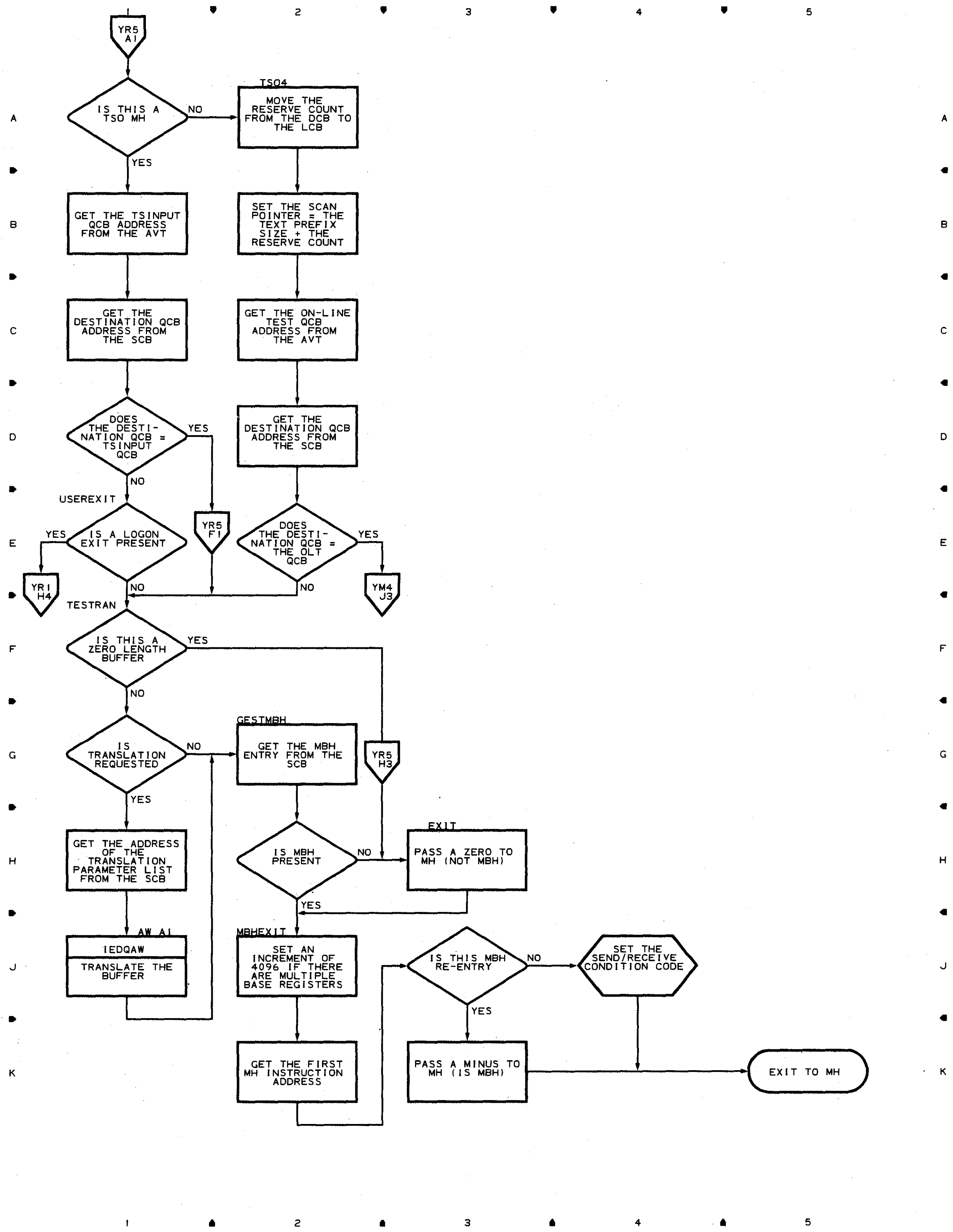
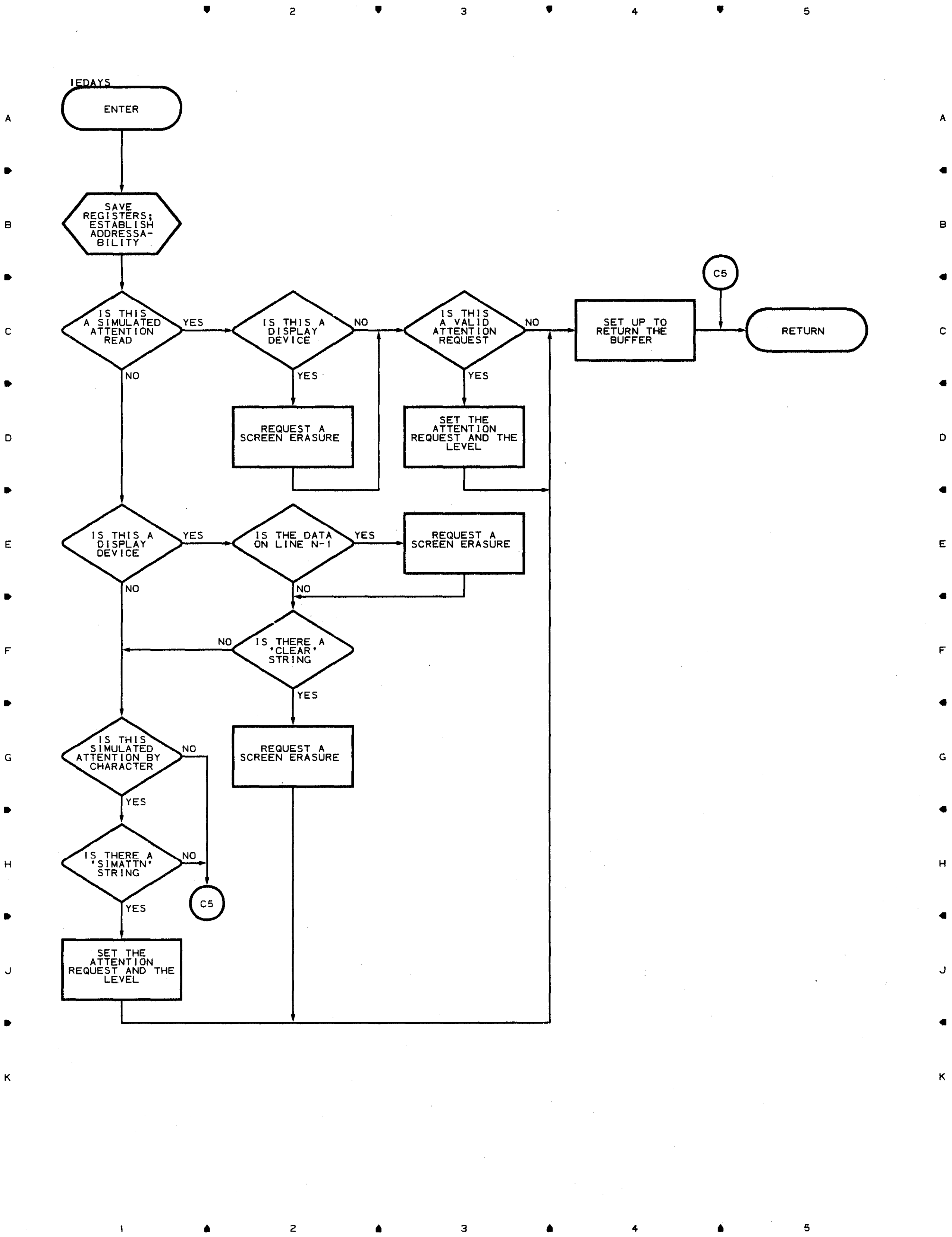
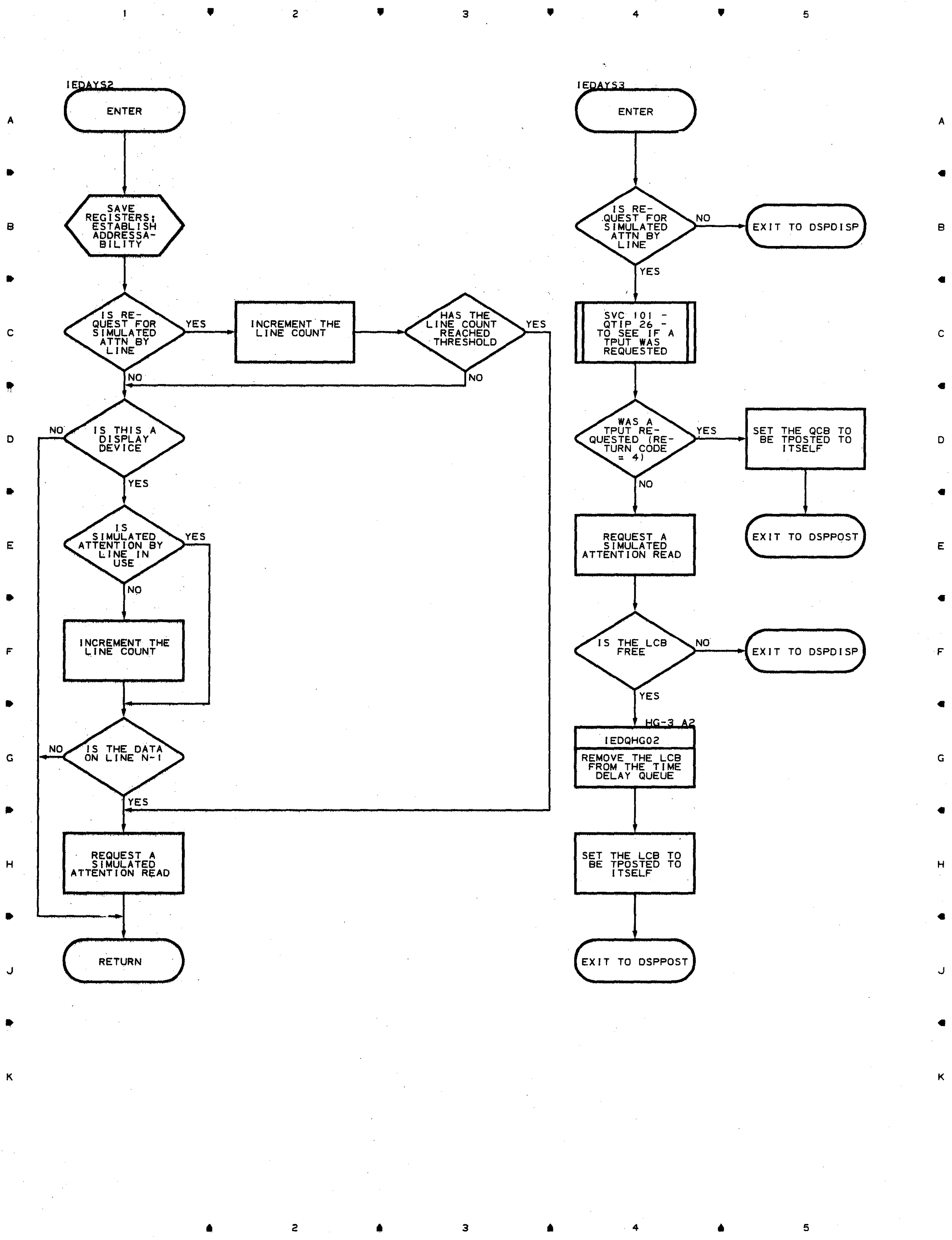

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
YR5 A1
IS THIS A TSO MH
NO
YES
TSO4
MOVE THE RESERVE COUNT FROM THE DCB TO THE LCB
GET THE TSINPUT QCB ADDRESS FROM THE AVT
SET THE SCAN POINTER = THE TEXT PREFIX SIZE + THE RESERVE COUNT
GET THE DESTINATION QCB ADDRESS FROM THE SCB
GET THE ON-LINE TEST QCB ADDRESS FROM THE AVT
DOES THE DESTI- NATION QCB = TSINPUT QCB
YES
NO
GET THE DESTINATION QCB ADDRESS FROM THE SCB
USEREXIT
YES
IS A LOGON EXIT PRESENT
YR5 F1
DOES THE DESTI- NATION QCB = THE OLT QCB
YES
YR1 H4
NO
NO
YM4 J3
TESTRAN
IS THIS A ZERO LENGTH BUFFER
YES
NO
GESTMBH
IS TRANSLATION REQUESTED
NO
GET THE MBH ENTRY FROM THE SCB
YR5 H3
YES
EXIT
GET THE ADDRESS OF THE TRANSLATION PARAMETER LIST FROM THE SCB
IS MBH PRESENT
NO
PASS A ZERO TO MH (NOT MBH)
YES
AW A1
MBHEXIT
IEDQAW
TRANSLATE THE BUFFER
SET AN INCREMENT OF 4096 IF THERE ARE MULTIPLE BASE REGISTERS
IS THIS MBH RE-ENTRY
NO
SET THE SEND/RECEIVE CONDITION CODE
YES
GET THE FIRST MH INSTRUCTION ADDRESS
PASS A MINUS TO MH (IS MBH)
EXIT TO MH

Chart YS-1 TSO SIMULATED ATTENTION ROUTINE



2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDAYS
ENTER
SAVE REGISTERS; ESTABLISH ADDRESSABILITY
IS THIS A SIMULATED ATTENTION READ
YES
NO
IS THIS A DISPLAY DEVICE
NO
YES
REQUEST A SCREEN ERASURE
IS THIS A VALID ATTENTION REQUEST
NO
YES
SET THE ATTENTION REQUEST AND THE LEVEL
SET UP TO RETURN THE BUFFER
C5
RETURN
IS THIS A DISPLAY DEVICE
YES
NO
IS THE DATA ON LINE N-1
YES
NO
REQUEST A SCREEN ERASURE
IS THERE A 'CLEAR' STRING
NO
YES
REQUEST A SCREEN ERASURE
IS THIS SIMULATED ATTENTION BY CHARACTER
NO
YES
IS THERE A 'SIMATTN' STRING
NO
YES
C5
SET THE ATTENTION REQUEST AND THE LEVEL
1
2
3
4
5

Chart YS-2 TSO SIMULATED ATTENTION ROUTINE



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IEDAYS2
ENTER
SAVE REGISTERS; ESTABLISH ADDRESSABILITY
IS REQUEST FOR SIMULATED ATTN BY LINE
YES
NO
INCREMENT THE LINE COUNT
HAS THE LINE COUNT REACHED THRESHOLD
YES
NO
IS THIS A DISPLAY DEVICE
NO
YES
IS SIMULATED ATTENTION BY LINE IN USE
YES
NO
INCREMENT THE LINE COUNT
IS THE DATA ON LINE N-1
NO
YES
REQUEST A SIMULATED ATTENTION READ
RETURN
IEDAYS3
ENTER
IS REQUEST FOR SIMULATED ATTN BY LINE
NO
EXIT TO DSPDISP
YES
SVC 101 - QTIP 26 - TO SEE IF A TPUT WAS REQUESTED
WAS A TPUT REQUESTED (RETURN CODE = 4)
YES
SET THE QCB TO BE TPOSTED TO ITSELF
EXIT TO DSPPOST
NO
REQUEST A SIMULATED ATTENTION READ
IS THE LCB FREE
NO
EXIT TO DSPDISP
YES
HG-3 A2
IEDQHG02
REMOVE THE LCB FROM THE TIME DELAY QUEUE
SET THE LCB TO BE TPOSTED TO ITSELF
EXIT TO DSPPOST

Chart YT TSO ABEND INTERFACE ROUTINE

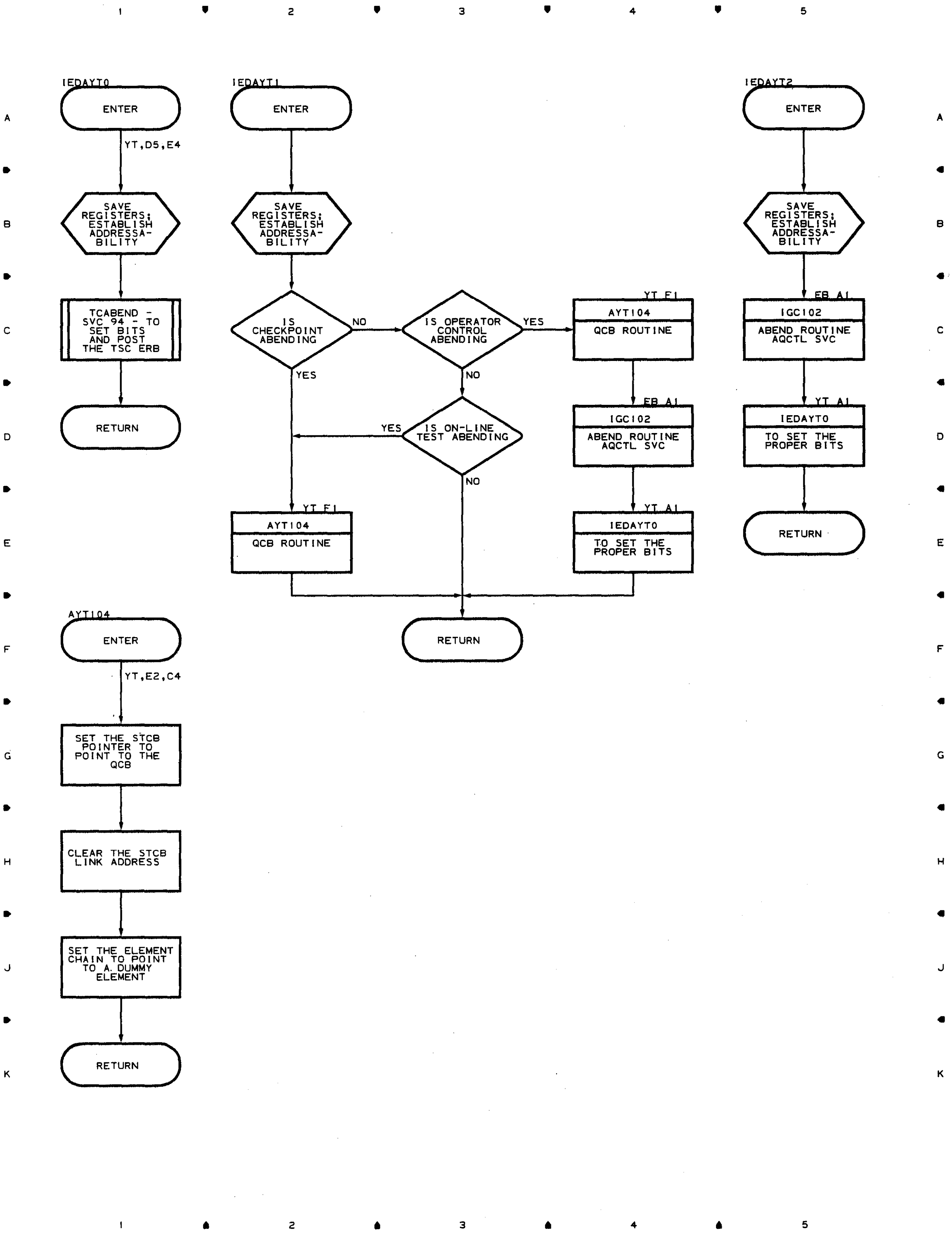


Chart YX TSO INMSG/OUTMSG LINKER

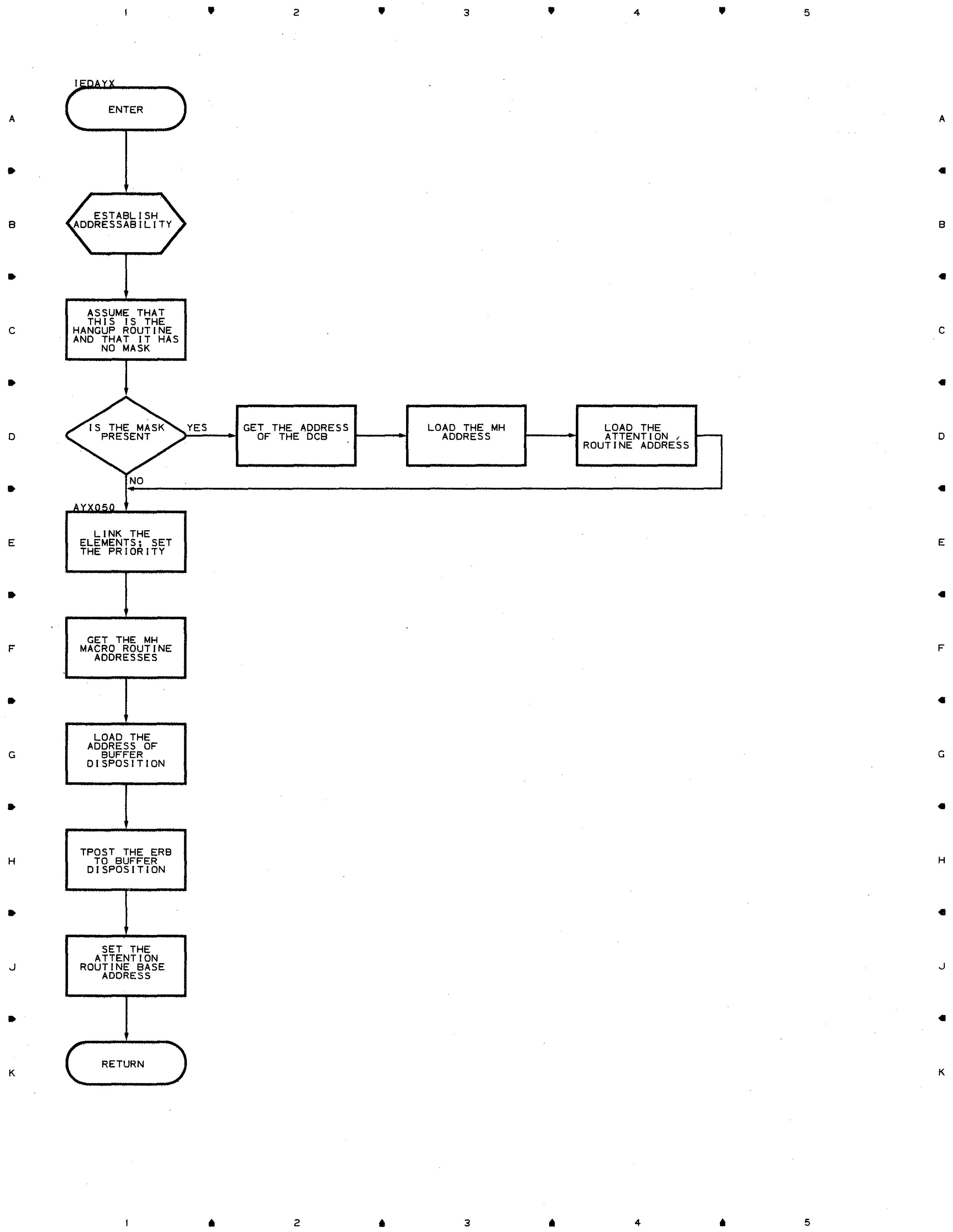

1
2
3
4
5
IEDAYX
ENTER
ESTABLISH ADDRESSABILITY
ASSUME THAT THIS IS THE HANGUP ROUTINE AND THAT IT HAS NO MASK
IS THE MASK PRESENT
YES
GET THE ADDRESS OF THE DCB
LOAD THE MH ADDRESS
LOAD THE ATTENTION ROUTINE ADDRESS
NO
AYX050
LINK THE ELEMENTS; SET THE PRIORITY
GET THE MH MACRO ROUTINE ADDRESSES
LOAD THE ADDRESS OF BUFFER DISPOSITION
TPOST THE ERB TO BUFFER DISPOSITION
SET THE ATTENTION ROUTINE BASE ADDRESS
RETURN
A
B
C
D
E
F
G
H
J
K

Chart YY TSO ASYNCHRONOUS TIME DELAY REMOVAL ROUTINE

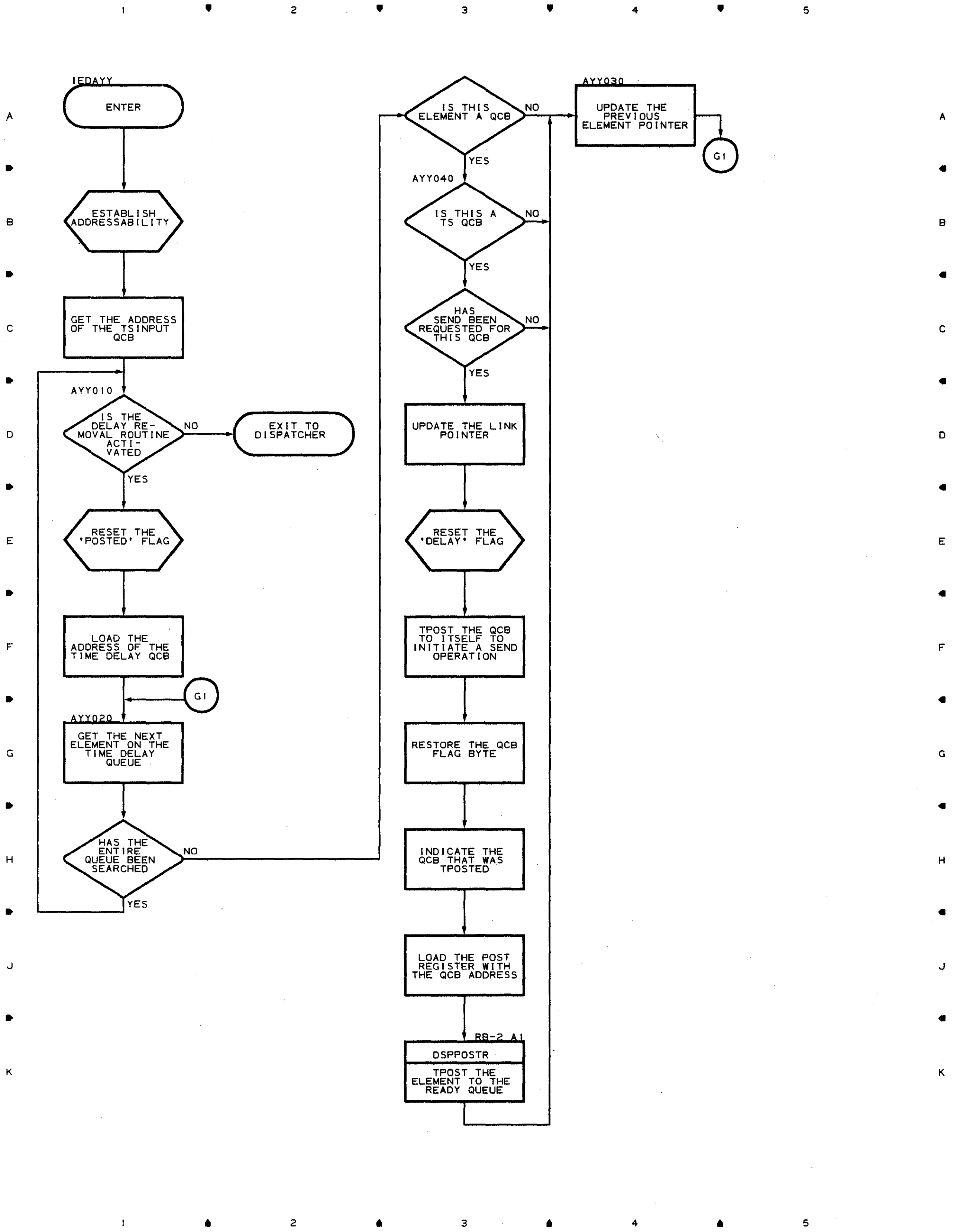

IEDAYY
ENTER
ESTABLISH ADDRESSABILITY
GET THE ADDRESS OF THE TSINPUT QCB
AYY010
IS THE DELAY RE-MOVAL ROUTINE ACTI-VATED
NO
EXIT TO DISPATCHER
YES
RESET THE 'POSTED' FLAG
LOAD THE ADDRESS OF THE TIME DELAY QCB
G1
AYY020
GET THE NEXT ELEMENT ON THE TIME DELAY QUEUE
HAS THE ENTIRE QUEUE BEEN SEARCHED
NO
YES
IS THIS ELEMENT A QCB
NO
YES
AYY030
UPDATE THE PREVIOUS ELEMENT POINTER
G1
AYY040
IS THIS A TS QCB
NO
YES
HAS SEND BEEN REQUESTED FOR THIS QCB
NO
YES
UPDATE THE LINK POINTER
RESET THE 'DELAY' FLAG
TPOST THE QCB TO ITSELF TO INITIATE A SEND OPERATION
RESTORE THE QCB FLAG BYTE
INDICATE THE QCB THAT WAS TPOSTED
LOAD THE POST REGISTER WITH THE QCB ADDRESS
RB-2 A1
DSPPOSTR
TPOST THE ELEMENT TO THE READY QUEUE
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart YZ-1 TIME SHARING SCHEDULER

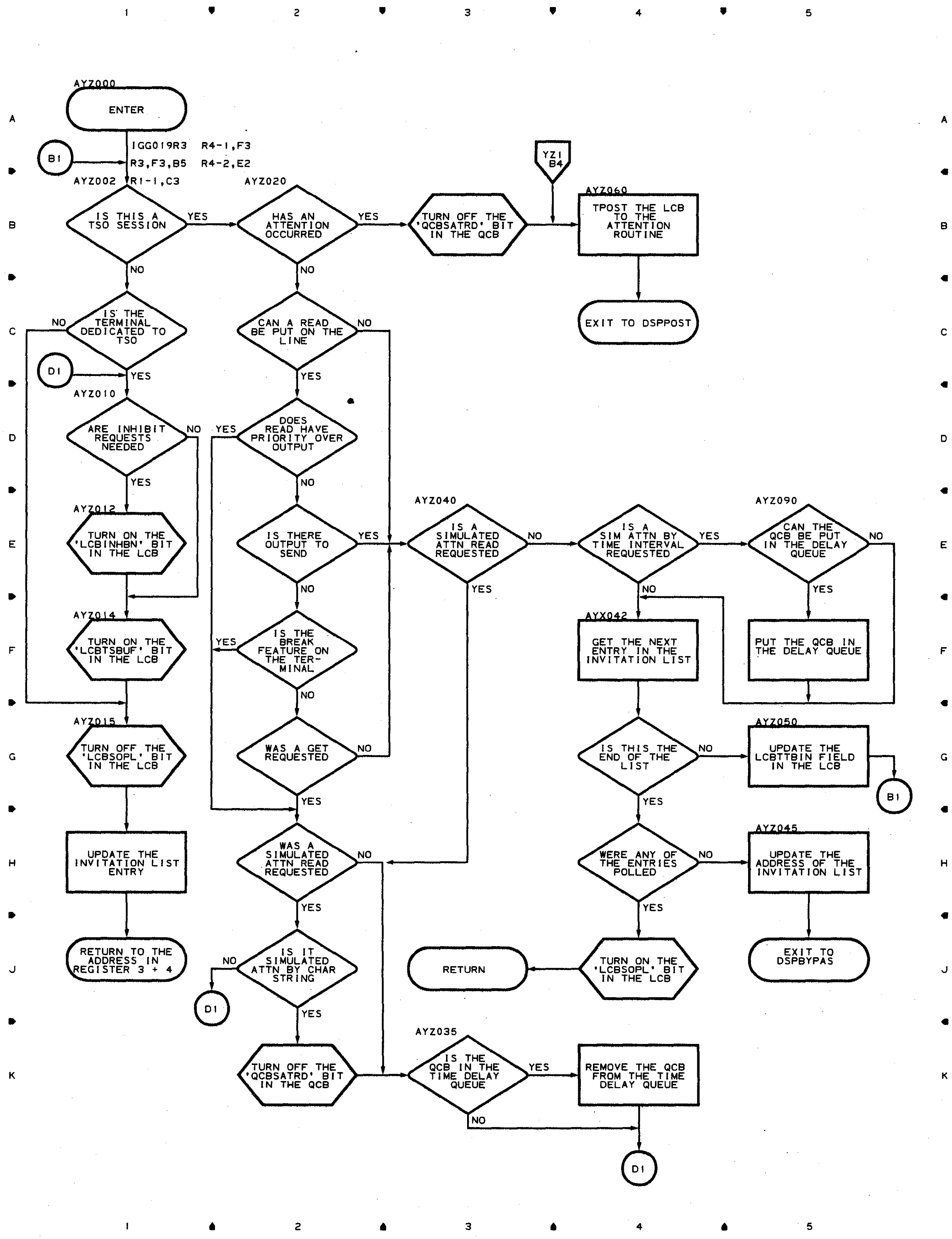

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
AYZ000
ENTER
B1
IGG019R3 R4-1,F3
R3,F3,B5 R4-2,E2
AYZ002 R1-1,C3
IS THIS A TSO SESSION
YES
NO
AYZ020
HAS AN ATTENTION OCCURRED
YES
NO
TURN OFF THE 'QCBSATRD' BIT IN THE QCB
YZ1 B4
AYZ060
TPOST THE LCB TO THE ATTENTION ROUTINE
EXIT TO DSPPOST
IS THE TERMINAL DEDICATED TO TSO
NO
YES
D1
CAN A READ BE PUT ON THE LINE
NO
YES
AYZ010
ARE INHIBIT REQUESTS NEEDED
NO
YES
DOES READ HAVE PRIORITY OVER OUTPUT
YES
NO
AYZ012
TURN ON THE 'LCBINHBN' BIT IN THE LCB
IS THERE OUTPUT TO SEND
YES
NO
AYZ040
IS A SIMULATED ATTN READ REQUESTED
NO
YES
IS A SIM ATTN BY TIME INTERVAL REQUESTED
YES
NO
AYZ090
CAN THE QCB BE PUT IN THE DELAY QUEUE
NO
YES
AYZ014
TURN ON THE 'LCBTSBUF' BIT IN THE LCB
IS THE BREAK FEATURE ON THE TER-MINAL
YES
NO
AYX042
GET THE NEXT ENTRY IN THE INVITATION LIST
PUT THE QCB IN THE DELAY QUEUE
AYZ015
TURN OFF THE 'LCBSOPL' BIT IN THE LCB
WAS A GET REQUESTED
NO
YES
IS THIS THE END OF THE LIST
NO
YES
AYZ050
UPDATE THE LCBTTBIN FIELD IN THE LCB
B1
UPDATE THE INVITATION LIST ENTRY
WAS A SIMULATED ATTN READ REQUESTED
NO
YES
WERE ANY OF THE ENTRIES POLLED
NO
YES
AYZ045
UPDATE THE ADDRESS OF THE INVITATION LIST
RETURN TO THE ADDRESS IN REGISTER 3 + 4
IS IT SIMULATED ATTN BY CHAR STRING
NO
YES
D1
RETURN
TURN ON THE 'LCBSOPL' BIT IN THE LCB
EXIT TO DSPBYPAS
AYZ035
TURN OFF THE 'QCBSATRD' BIT IN THE QCB
IS THE QCB IN THE TIME DELAY QUEUE
YES
NO
REMOVE THE QCB FROM THE TIME DELAY QUEUE
D1

Chart YZ-2 TIME SHARING SCHEDULER

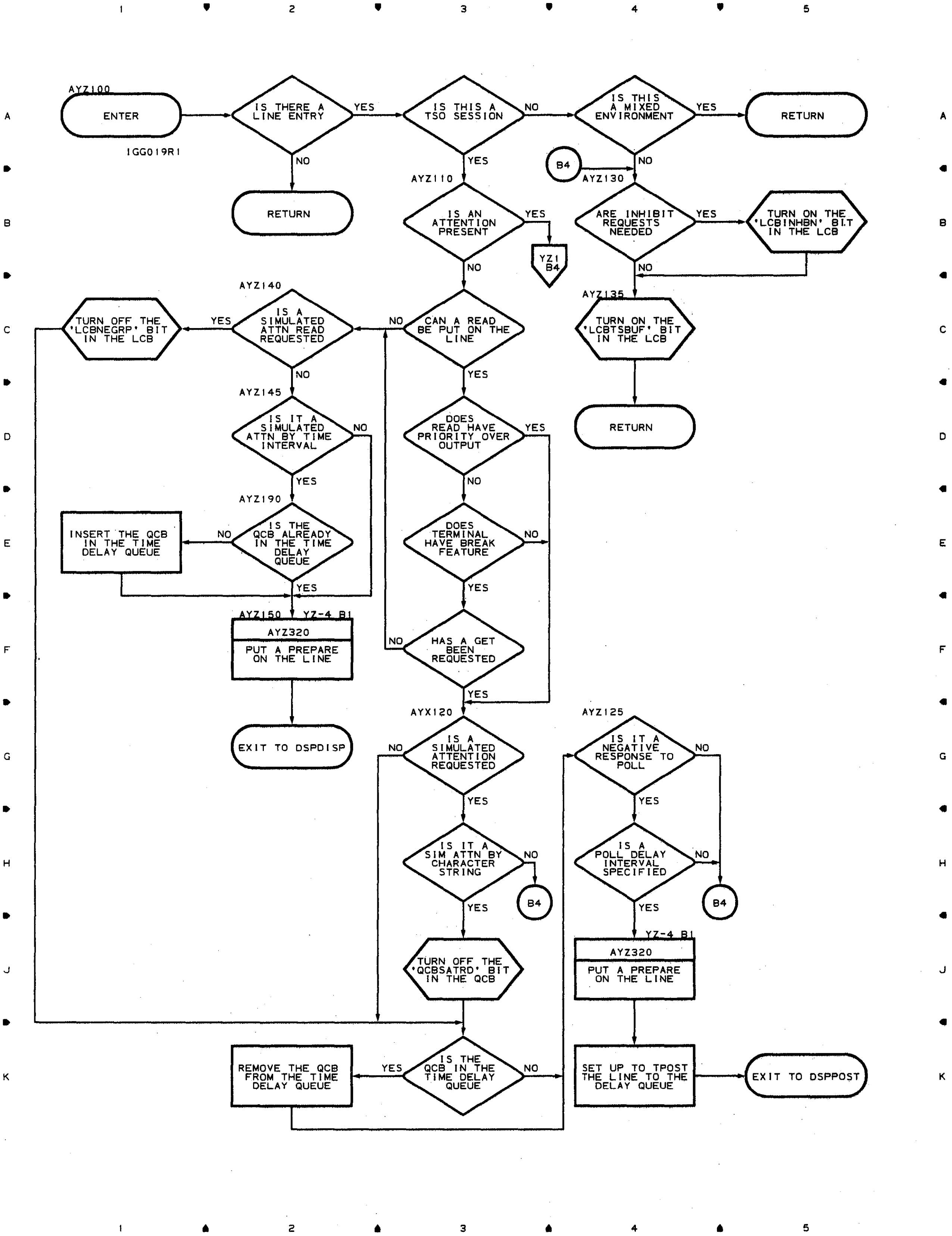

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
AYZ100
ENTER
IGG019R1
IS THERE A LINE ENTRY
YES
NO
RETURN
IS THIS A TSO SESSION
NO
YES
IS THIS A MIXED ENVIRONMENT
YES
RETURN
B4
NO
AYZ110
IS AN ATTENTION PRESENT
YES
YZ1 B4
NO
AYZ130
ARE INHIBIT REQUESTS NEEDED
YES
TURN ON THE 'LCBINHBN' BIT IN THE LCB
NO
AYZ135
TURN ON THE 'LCBTSBUF' BIT IN THE LCB
RETURN
AYZ140
IS A SIMULATED ATTN READ REQUESTED
YES
TURN OFF THE 'LCBNEGRP' BIT IN THE LCB
NO
CAN A READ BE PUT ON THE LINE
YES
NO
AYZ145
IS IT A SIMULATED ATTN BY TIME INTERVAL
NO
YES
DOES READ HAVE PRIORITY OVER OUTPUT
YES
NO
AYZ190
IS THE QCB ALREADY IN THE TIME DELAY QUEUE
NO
INSERT THE QCB IN THE TIME DELAY QUEUE
YES
DOES TERMINAL HAVE BREAK FEATURE
NO
YES
AYZ150
YZ-4 B1
AYZ320
PUT A PREPARE ON THE LINE
HAS A GET BEEN REQUESTED
NO
YES
EXIT TO DSPDISP
AYX120
IS A SIMULATED ATTENTION REQUESTED
NO
YES
AYZ125
IS IT A NEGATIVE RESPONSE TO POLL
NO
YES
IS IT A SIM ATTN BY CHARACTER STRING
NO
B4
YES
IS A POLL DELAY INTERVAL SPECIFIED
NO
B4
YES
YZ-4 B1
AYZ320
PUT A PREPARE ON THE LINE
TURN OFF THE 'QCBSATRD' BIT IN THE QCB
REMOVE THE QCB FROM THE TIME DELAY QUEUE
YES
IS THE QCB IN THE TIME DELAY QUEUE
NO
SET UP TO TPOST THE LINE TO THE DELAY QUEUE
EXIT TO DSPPOST

Chart YZ-3 TIME SHARING SCHEDULER

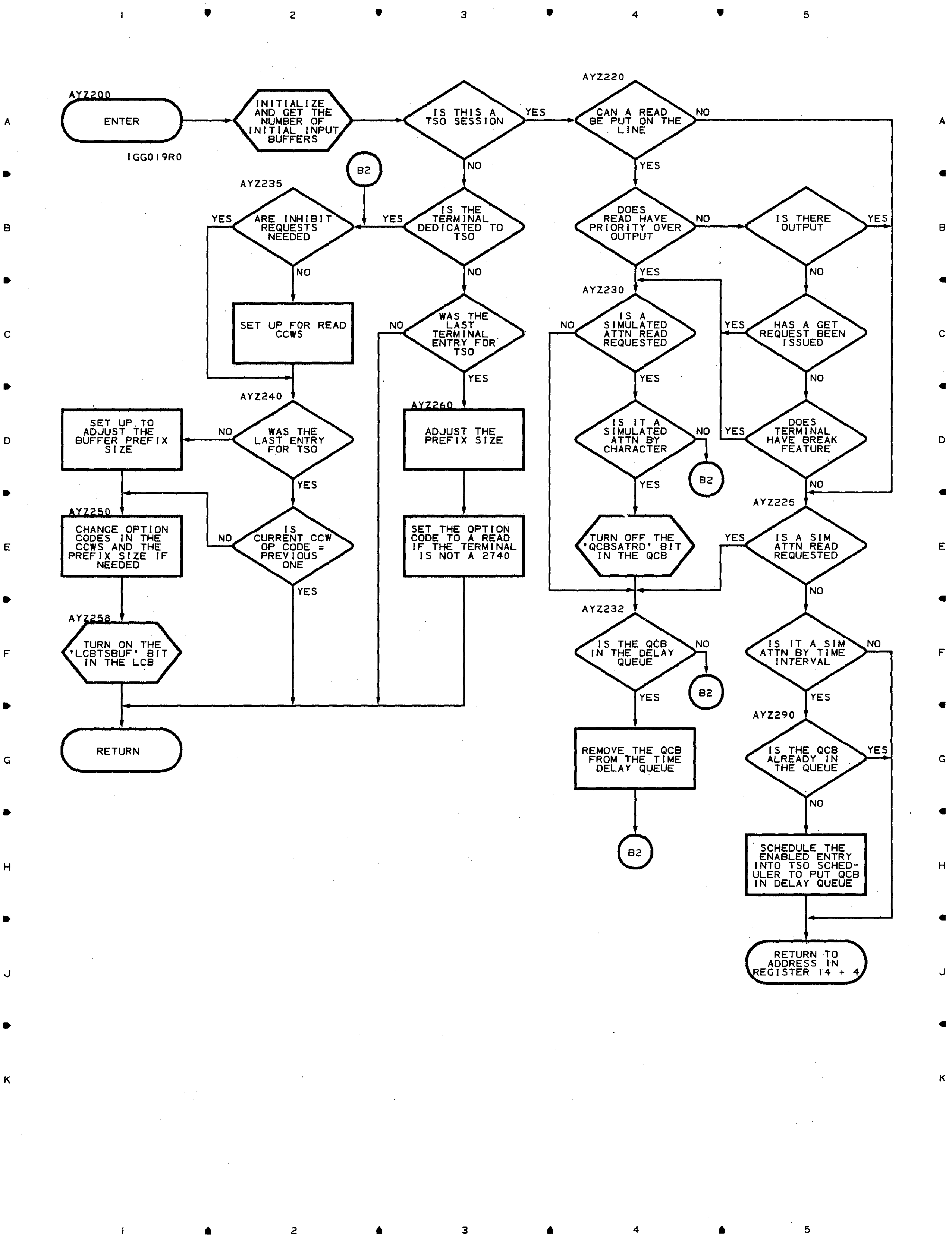

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
AYZ200
ENTER
IGG019R0
INITIALIZE AND GET THE NUMBER OF INITIAL INPUT BUFFERS
IS THIS A TSO SESSION
YES
NO
AYZ220
CAN A READ BE PUT ON THE LINE
NO
YES
B2
AYZ235
ARE INHIBIT REQUESTS NEEDED
YES
NO
IS THE TERMINAL DEDICATED TO TSO
YES
NO
DOES READ HAVE PRIORITY OVER OUTPUT
NO
YES
IS THERE OUTPUT
YES
NO
SET UP FOR READ CCWS
WAS THE LAST TERMINAL ENTRY FOR TSO
NO
YES
AYZ230
IS A SIMULATED ATTN READ REQUESTED
NO
YES
HAS A GET REQUEST BEEN ISSUED
YES
NO
AYZ240
SET UP TO ADJUST THE BUFFER PREFIX SIZE
WAS THE LAST ENTRY FOR TSO
NO
YES
AYZ260
ADJUST THE PREFIX SIZE
IS IT A SIMULATED ATTN BY CHARACTER
NO
YES
B2
DOES TERMINAL HAVE BREAK FEATURE
YES
NO
AYZ225
AYZ250
CHANGE OPTION CODES IN THE CCWS AND THE PREFIX SIZE IF NEEDED
IS CURRENT CCW OP CODE = PREVIOUS ONE
NO
YES
SET THE OPTION CODE TO A READ IF THE TERMINAL IS NOT A 2740
TURN OFF THE 'QCBSATRD' BIT IN THE QCB
IS A SIM ATTN READ REQUESTED
YES
NO
AYZ258
TURN ON THE 'LCBTSBUF' BIT IN THE LCB
AYZ232
IS THE QCB IN THE DELAY QUEUE
NO
YES
B2
IS IT A SIM ATTN BY TIME INTERVAL
NO
YES
AYZ290
RETURN
REMOVE THE QCB FROM THE TIME DELAY QUEUE
IS THE QCB ALREADY IN THE QUEUE
YES
NO
B2
SCHEDULE THE ENABLED ENTRY INTO TSO SCHEDULER TO PUT QCB IN DELAY QUEUE
RETURN TO ADDRESS IN REGISTER 14 + 4

Chart YZ-4 TIME SHARING SCHEDULER

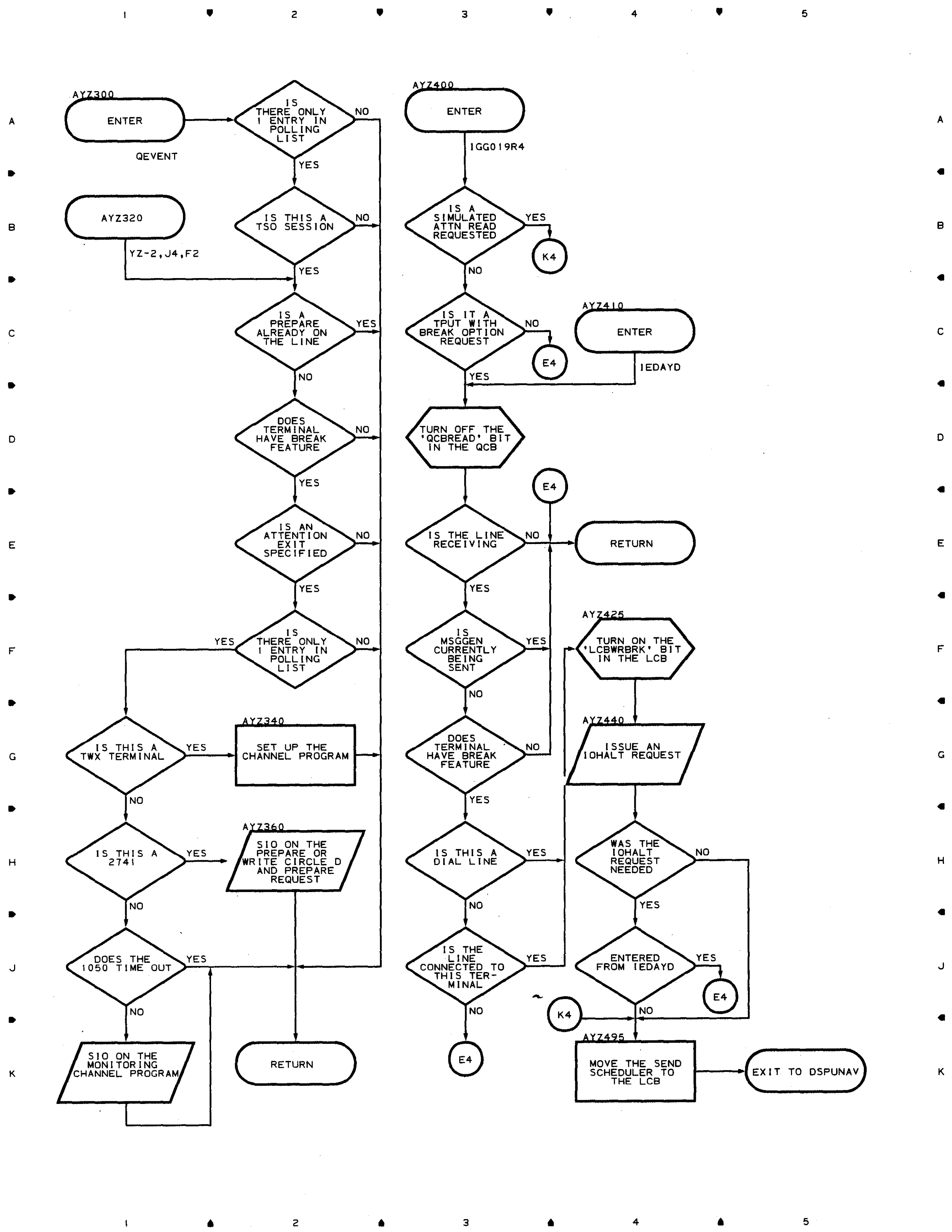

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
AYZ300
ENTER
QEVENT
IS THERE ONLY 1 ENTRY IN POLLING LIST
NO
YES
AYZ320
YZ-2,J4,F2
IS THIS A TSO SESSION
NO
YES
IS A PREPARE ALREADY ON THE LINE
YES
NO
DOES TERMINAL HAVE BREAK FEATURE
NO
YES
IS AN ATTENTION EXIT SPECIFIED
NO
YES
IS THERE ONLY 1 ENTRY IN POLLING LIST
YES
NO
IS THIS A TWX TERMINAL
YES
NO
AYZ340
SET UP THE CHANNEL PROGRAM
IS THIS A 2741
YES
NO
AYZ360
SIO ON THE PREPARE OR WRITE CIRCLE D AND PREPARE REQUEST
DOES THE 1050 TIME OUT
YES
NO
SIO ON THE MONITORING CHANNEL PROGRAM
RETURN
AYZ400
ENTER
IGG019R4
IS A SIMULATED ATTN READ REQUESTED
YES
K4
NO
IS IT A TPUT WITH BREAK OPTION REQUEST
NO
E4
YES
AYZ410
ENTER
IEDAYD
TURN OFF THE 'QCBREAD' BIT IN THE QCB
E4
IS THE LINE RECEIVING
NO
RETURN
YES
IS MSGGEN CURRENTLY BEING SENT
YES
NO
AYZ425
TURN ON THE 'LCBWRBRK' BIT IN THE LCB
DOES TERMINAL HAVE BREAK FEATURE
NO
YES
AYZ440
ISSUE AN IOHALT REQUEST
IS THIS A DIAL LINE
YES
NO
WAS THE IOHALT REQUEST NEEDED
NO
YES
IS THE LINE CONNECTED TO THIS TER-MINAL
YES
NO
ENTERED FROM IEDAYD
YES
E4
K4
NO
E4
AYZ495
MOVE THE SEND SCHEDULER TO THE LCB
EXIT TO DSPUNAV
1
2
3
4
5

Chart YZ-5 TIME SHARING SCHEDULER

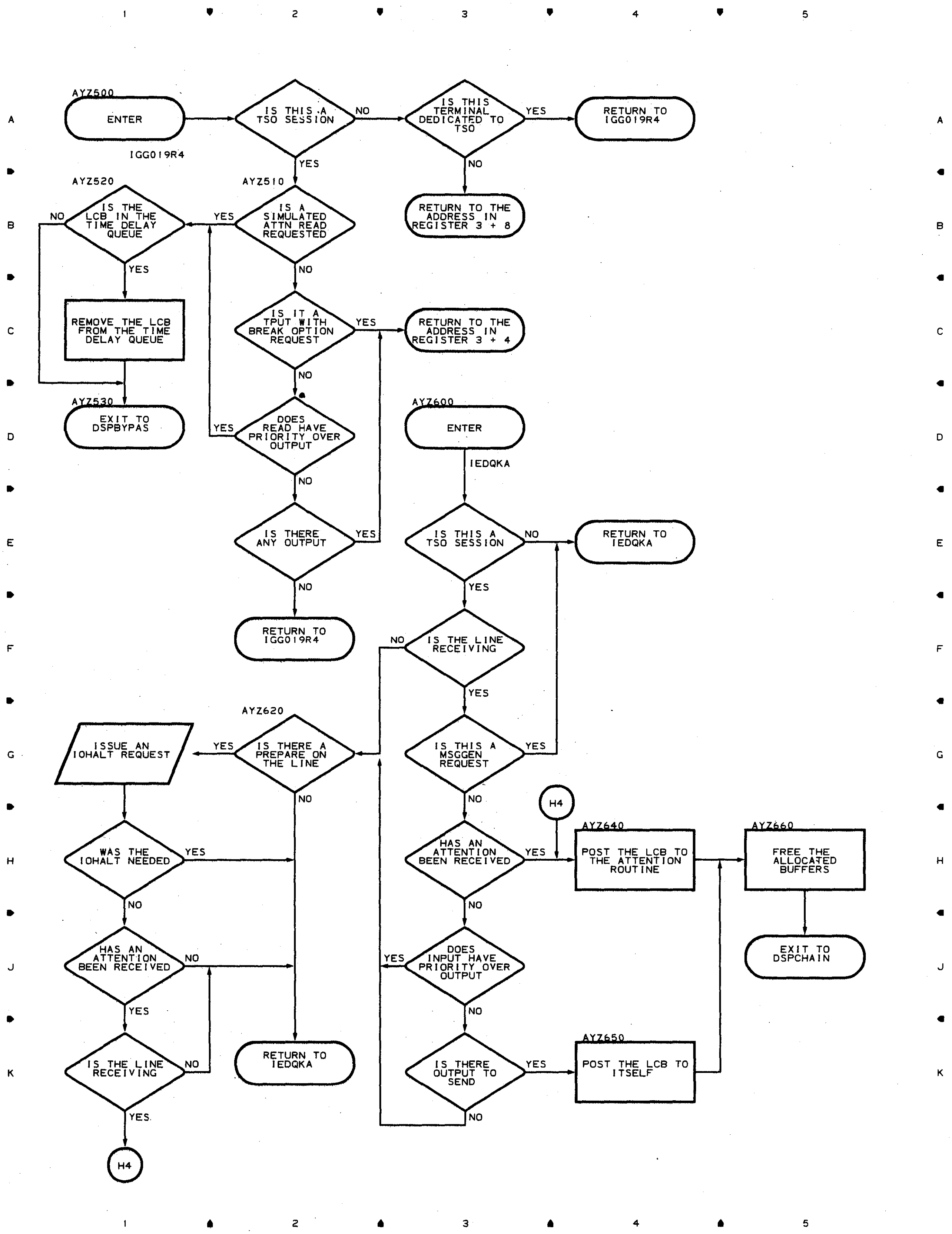


Chart Z1-1 OPERATOR CONTROL CONTROL MODULE - LOAD 0

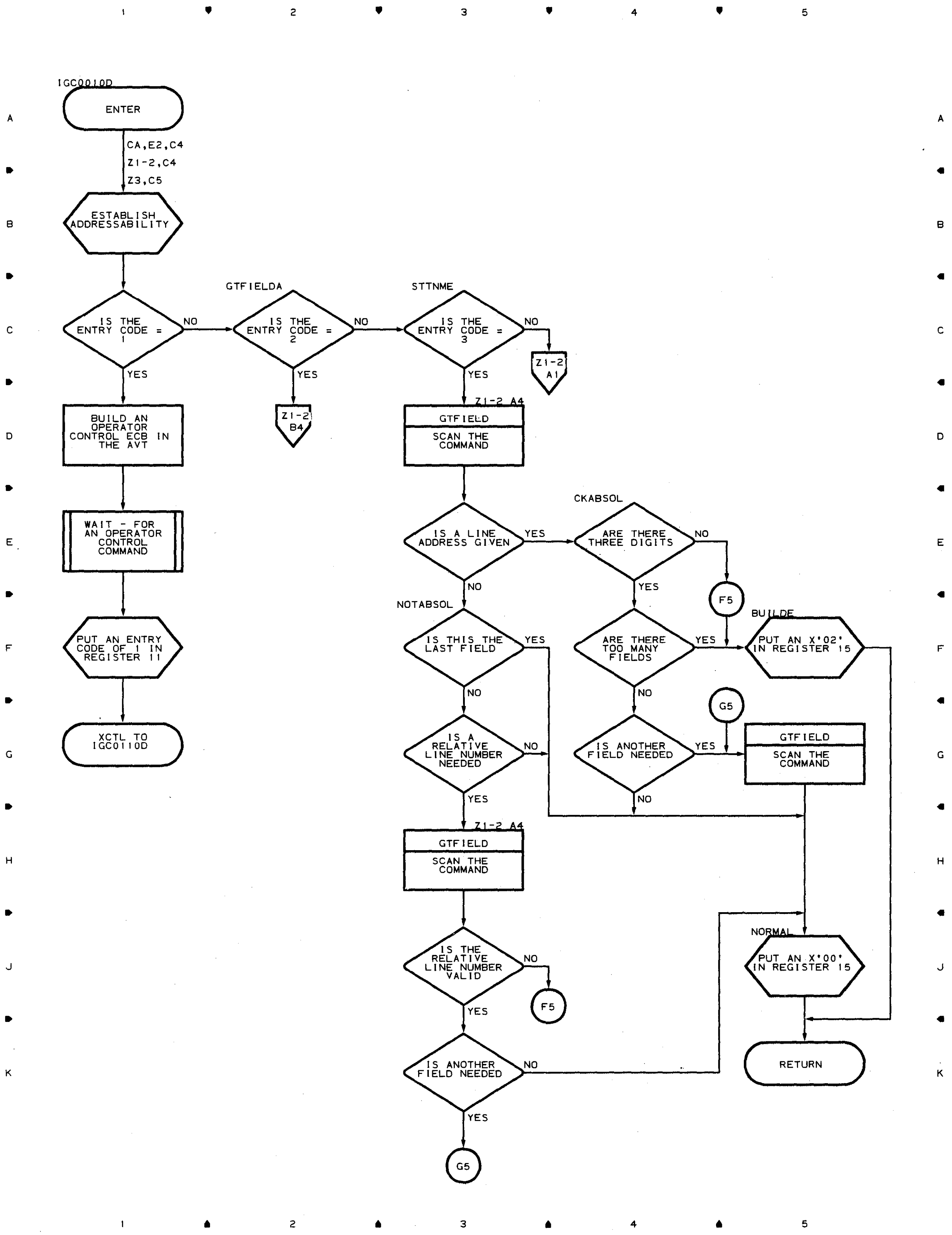


Chart Z1-2 OPERATOR CONTROL CONTROL MODULE - LOAD 0

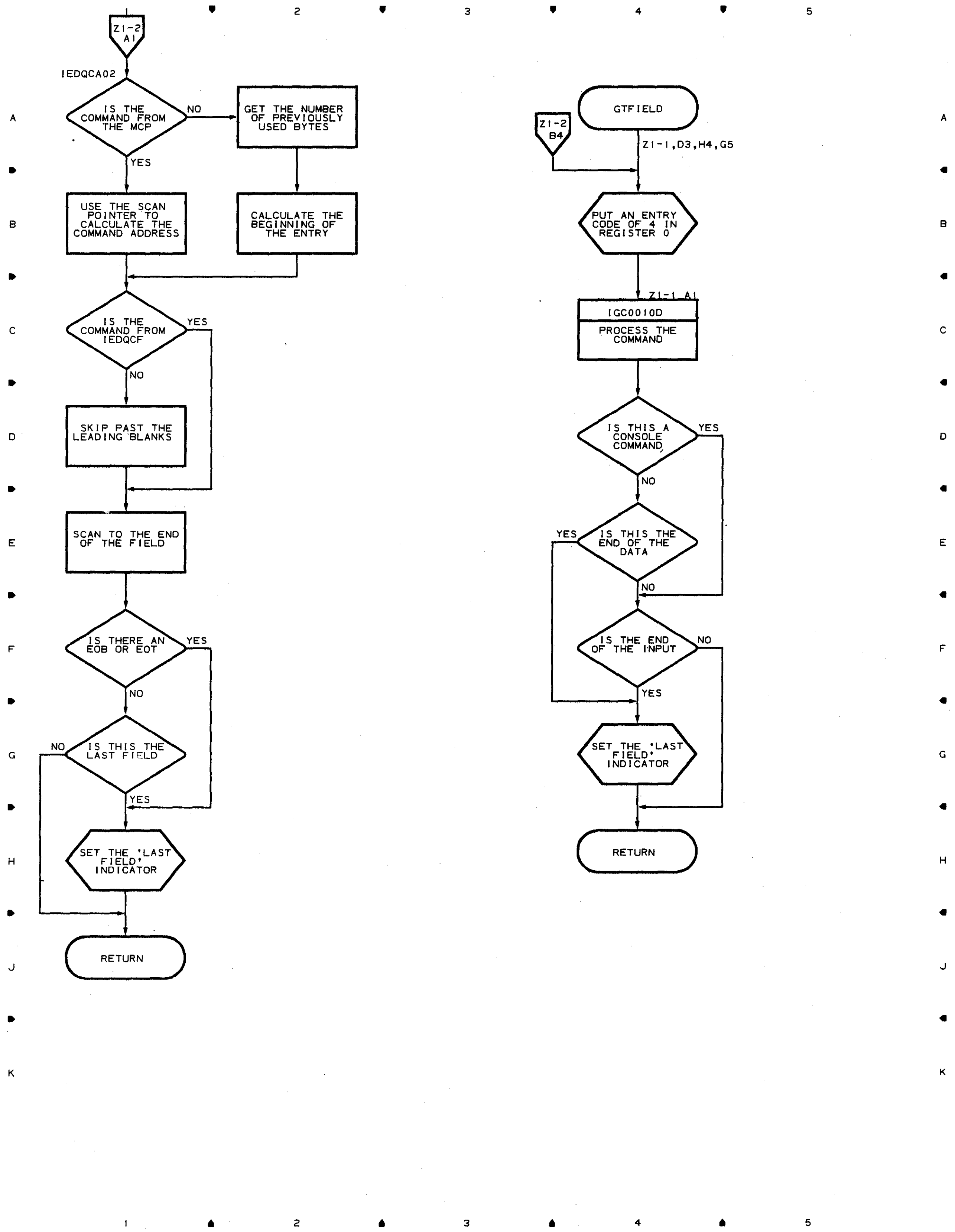

Z1-2 A1
IEDQCA02
IS THE COMMAND FROM THE MCP
NO
YES
GET THE NUMBER OF PREVIOUSLY USED BYTES
USE THE SCAN POINTER TO CALCULATE THE COMMAND ADDRESS
CALCULATE THE BEGINNING OF THE ENTRY
IS THE COMMAND FROM IEDQCF
YES
NO
SKIP PAST THE LEADING BLANKS
SCAN TO THE END OF THE FIELD
IS THERE AN EOB OR EOT
YES
NO
IS THIS THE LAST FIELD
NO
YES
SET THE 'LAST FIELD' INDICATOR
RETURN
GTFIELD
Z1-2 B4
Z1-1,D3,H4,G5
PUT AN ENTRY CODE OF 4 IN REGISTER 0
Z1-1 A1
IGC0010D
PROCESS THE COMMAND
IS THIS A CONSOLE COMMAND
YES
NO
IS THIS THE END OF THE DATA
YES
NO
IS THE END OF THE INPUT
NO
YES
SET THE 'LAST FIELD' INDICATOR
RETURN
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart Z2 OPERATOR CONTROL CONTROL MODULE - LOAD 1

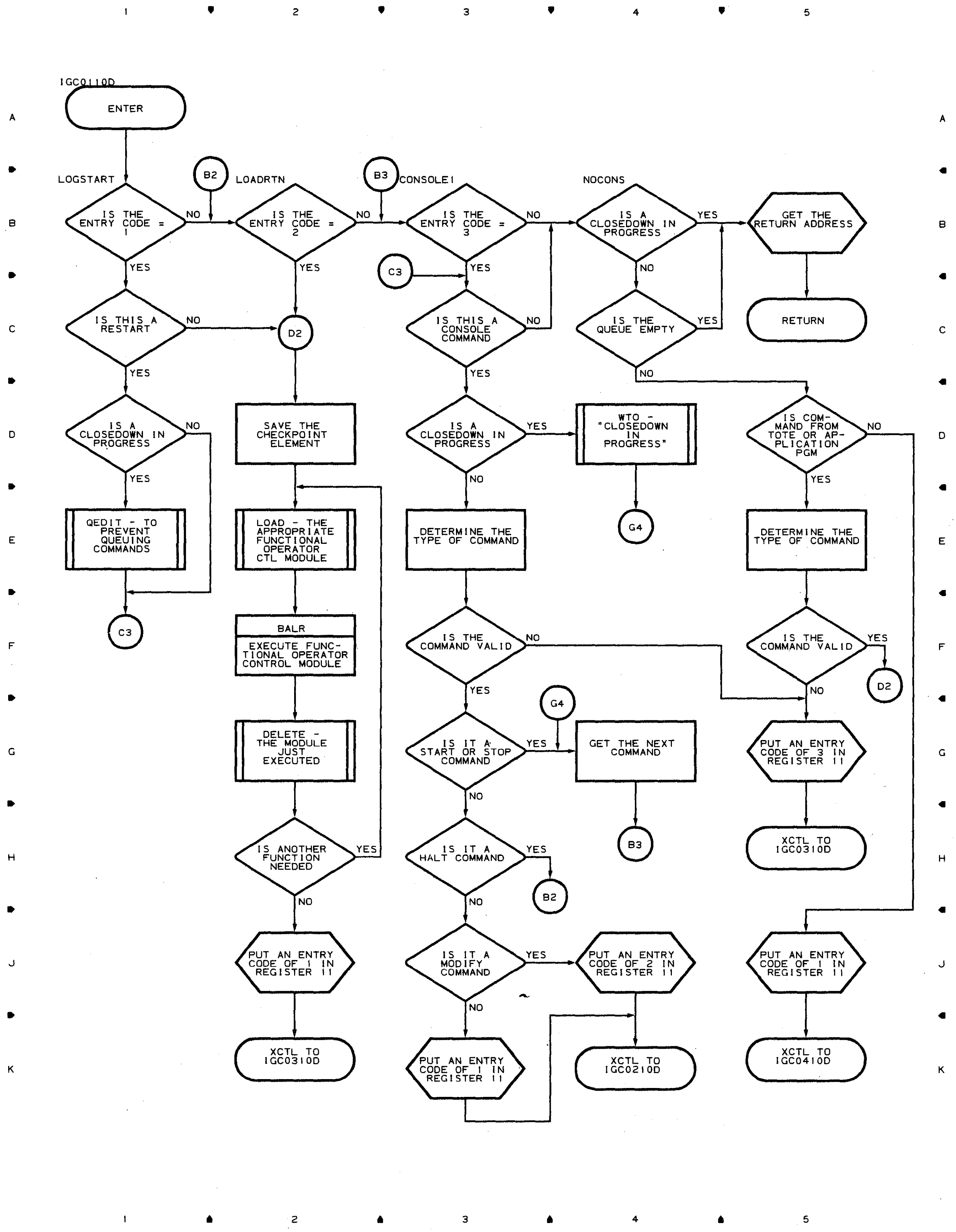

IGC0110D
ENTER
LOGSTART
IS THE ENTRY CODE = 1
NO
YES
IS THIS A RESTART
NO
YES
IS A CLOSEDOWN IN PROGRESS
NO
YES
QEDIT - TO PREVENT QUEUING COMMANDS
C3
B2
LOADRTN
IS THE ENTRY CODE = 2
NO
YES
D2
SAVE THE CHECKPOINT ELEMENT
LOAD - THE APPROPRIATE FUNCTIONAL OPERATOR CTL MODULE
BALR
EXECUTE FUNCTIONAL OPERATOR CONTROL MODULE
DELETE - THE MODULE JUST EXECUTED
IS ANOTHER FUNCTION NEEDED
YES
NO
PUT AN ENTRY CODE OF 1 IN REGISTER 11
XCTL TO IGC0310D
B3
CONSOLE1
IS THE ENTRY CODE = 3
NO
C3
YES
IS THIS A CONSOLE COMMAND
NO
YES
IS A CLOSEDOWN IN PROGRESS
YES
WTO - "CLOSEDOWN IN PROGRESS"
G4
NO
DETERMINE THE TYPE OF COMMAND
IS THE COMMAND VALID
NO
YES
G4
IS IT A START OR STOP COMMAND
YES
GET THE NEXT COMMAND
B3
NO
IS IT A HALT COMMAND
YES
B2
NO
IS IT A MODIFY COMMAND
YES
PUT AN ENTRY CODE OF 2 IN REGISTER 11
NO
PUT AN ENTRY CODE OF 1 IN REGISTER 11
XCTL TO IGC0210D
NOCONS
IS A CLOSEDOWN IN PROGRESS
YES
NO
IS THE QUEUE EMPTY
YES
NO
GET THE RETURN ADDRESS
RETURN
IS COMMAND FROM TOTE OR APPLICATION PGM
NO
YES
DETERMINE THE TYPE OF COMMAND
IS THE COMMAND VALID
YES
D2
NO
PUT AN ENTRY CODE OF 3 IN REGISTER 11
XCTL TO IGC0310D
PUT AN ENTRY CODE OF 1 IN REGISTER 11
XCTL TO IGC0410D
1
2
3
4
5
A
B
C
D
E
F
G
H
J
K

Chart Z3 OPERATOR CONTROL CONTROL MODULE - LOAD 2

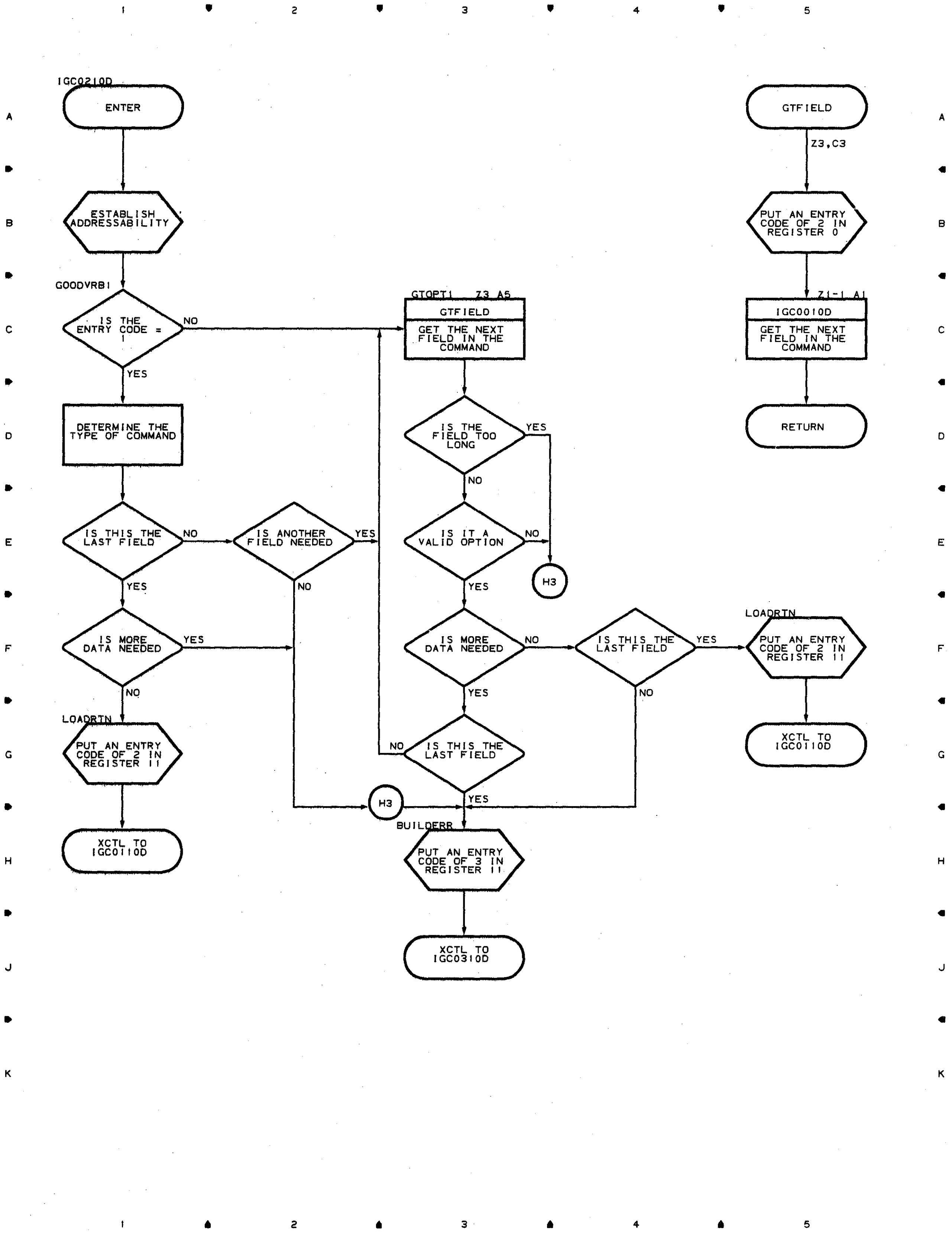


Chart Z4 OPERATOR CONTROL CONTROL MODULE - LOAD 3

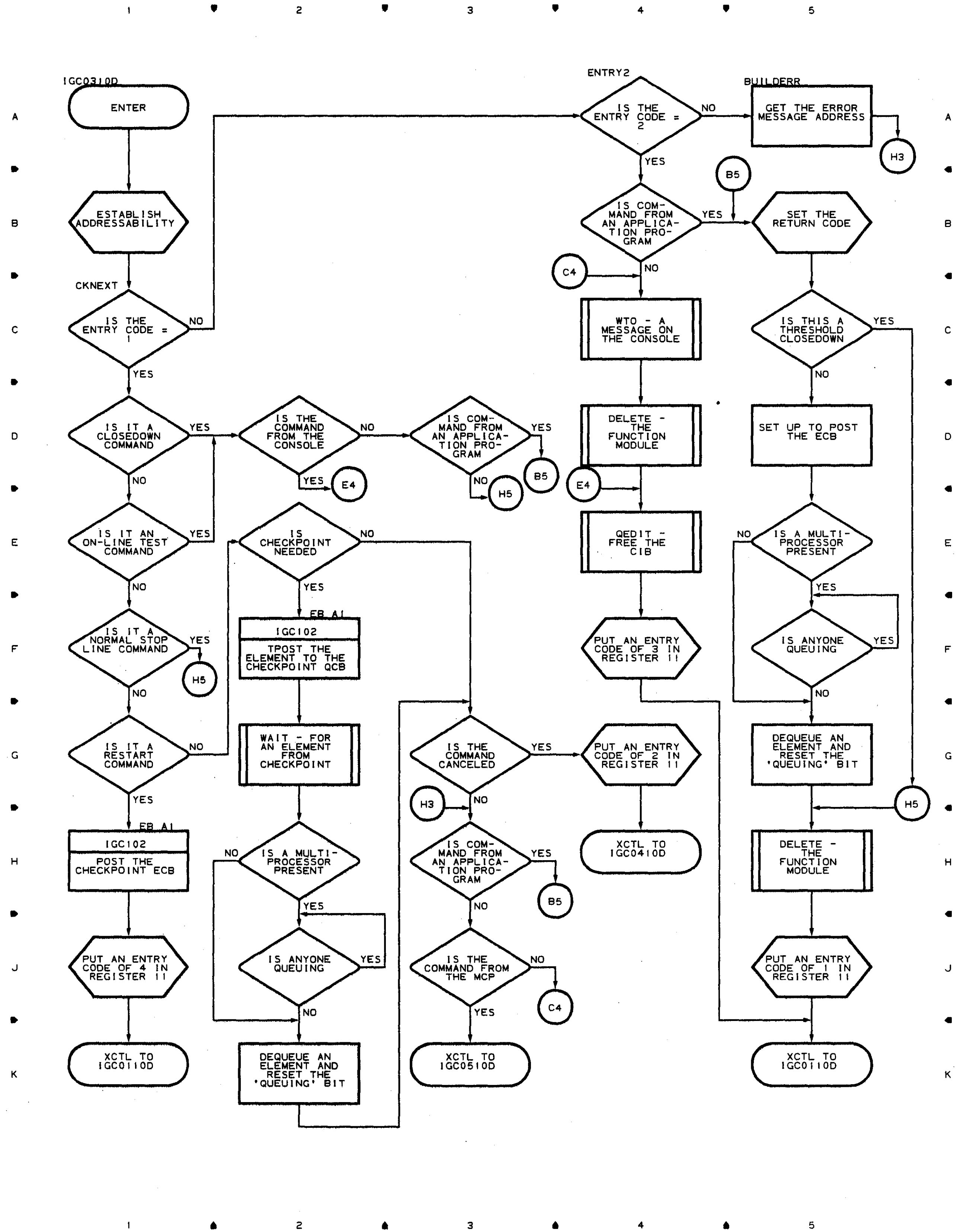
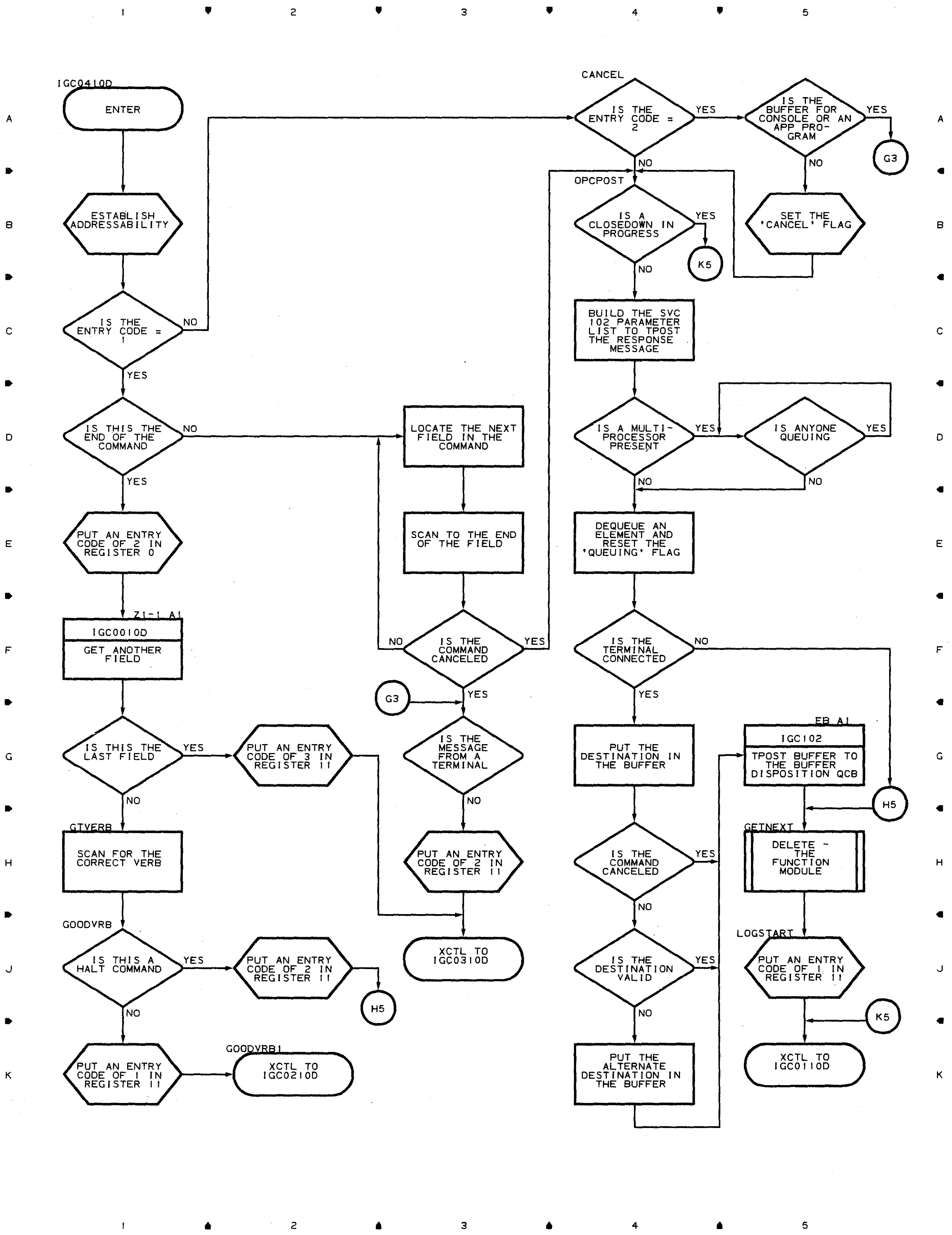
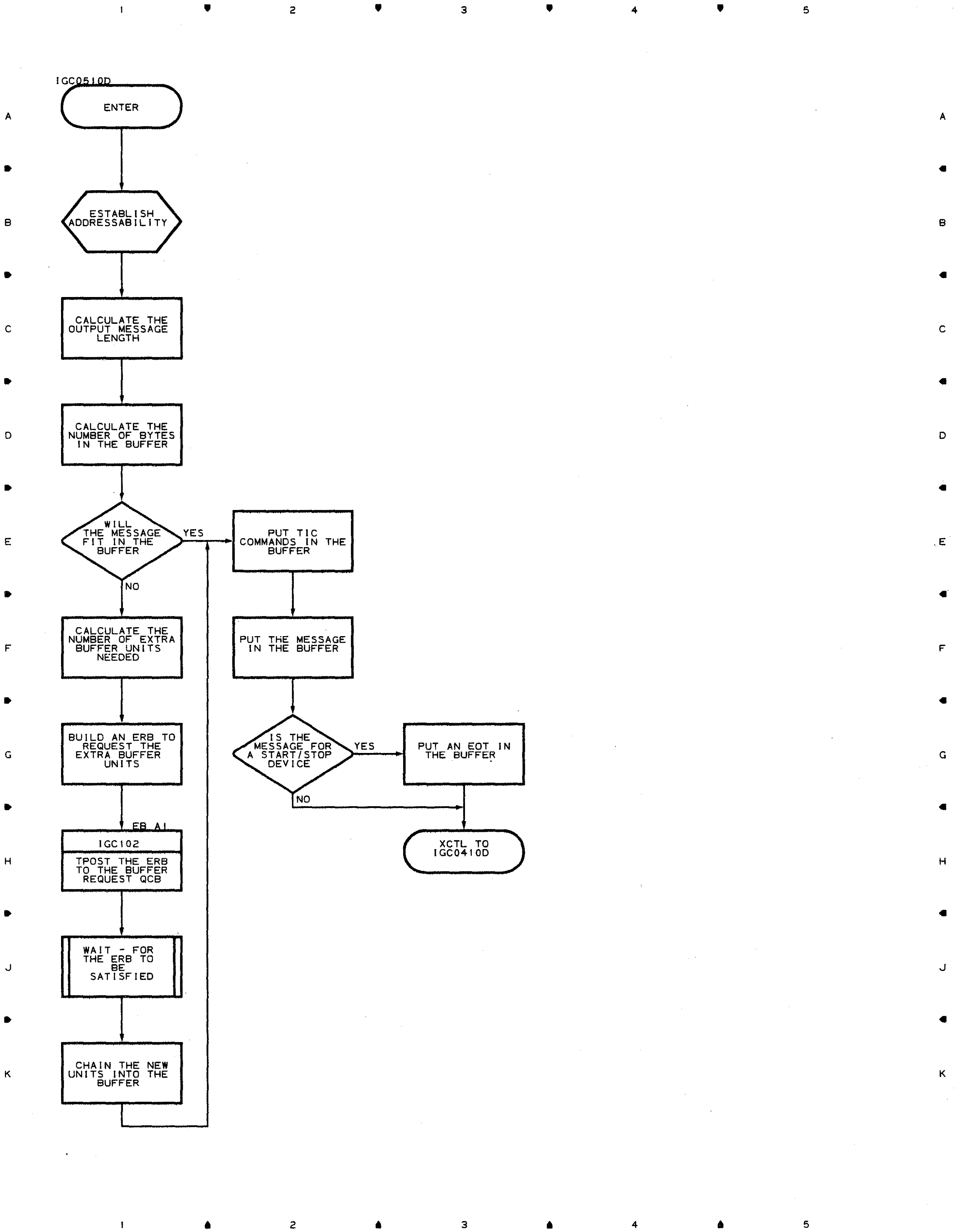

1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGC0310D
ENTER
ESTABLISH ADDRESSABILITY
CKNEXT
IS THE ENTRY CODE = 1
NO
YES
IS IT A CLOSEDOWN COMMAND
YES
NO
IS IT AN ON-LINE TEST COMMAND
YES
NO
IS IT A NORMAL STOP LINE COMMAND
YES
H5
NO
IS IT A RESTART COMMAND
NO
YES
EB A1
IGC102
POST THE CHECKPOINT ECB
PUT AN ENTRY CODE OF 4 IN REGISTER 11
XCTL TO IGC0110D
IS THE COMMAND FROM THE CONSOLE
NO
YES
E4
IS CHECKPOINT NEEDED
NO
YES
EB A1
IGC102
TPOST THE ELEMENT TO THE CHECKPOINT QCB
WAIT - FOR AN ELEMENT FROM CHECKPOINT
IS A MULTI-PROCESSOR PRESENT
NO
YES
IS ANYONE QUEUING
YES
NO
DEQUEUE AN ELEMENT AND RESET THE 'QUEUING' BIT
IS COM-MAND FROM AN APPLICA-TION PRO-GRAM
YES
B5
NO
H5
IS THE COMMAND CANCELED
YES
H3
NO
IS COM-MAND FROM AN APPLICA-TION PRO-GRAM
YES
B5
NO
IS THE COMMAND FROM THE MCP
NO
C4
YES
XCTL TO IGC0510D
ENTRY2
IS THE ENTRY CODE = 2
NO
YES
IS COM-MAND FROM AN APPLICA-TION PRO-GRAM
YES
C4
NO
WTO - A MESSAGE ON THE CONSOLE
DELETE - THE FUNCTION MODULE
E4
QEDIT - FREE THE CIB
PUT AN ENTRY CODE OF 3 IN REGISTER 11
PUT AN ENTRY CODE OF 2 IN REGISTER 11
XCTL TO IGC0410D
BUILDERR
GET THE ERROR MESSAGE ADDRESS
H3
B5
SET THE RETURN CODE
IS THIS A THRESHOLD CLOSEDOWN
YES
NO
SET UP TO POST THE ECB
IS A MULTI-PROCESSOR PRESENT
NO
YES
IS ANYONE QUEUING
YES
NO
DEQUEUE AN ELEMENT AND RESET THE 'QUEUING' BIT
H5
DELETE - THE FUNCTION MODULE
PUT AN ENTRY CODE OF 1 IN REGISTER 11
XCTL TO IGC0110D

Chart Z5 OPERATOR CONTROL CONTROL MODULE - LOAD 4



1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGC0410D
ENTER
ESTABLISH ADDRESSABILITY
IS THE ENTRY CODE = 1
NO
YES
IS THIS THE END OF THE COMMAND
NO
YES
PUT AN ENTRY CODE OF 2 IN REGISTER 0
Z1-1 A1
IGC0010D
GET ANOTHER FIELD
IS THIS THE LAST FIELD
YES
NO
PUT AN ENTRY CODE OF 3 IN REGISTER 11
GTVERB
SCAN FOR THE CORRECT VERB
GOODVRB
IS THIS A HALT COMMAND
YES
NO
PUT AN ENTRY CODE OF 2 IN REGISTER 11
H5
PUT AN ENTRY CODE OF 1 IN REGISTER 11
GOODVRB1
XCTL TO IGC0210D
LOCATE THE NEXT FIELD IN THE COMMAND
SCAN TO THE END OF THE FIELD
IS THE COMMAND CANCELED
NO
YES
G3
YES
IS THE MESSAGE FROM A TERMINAL
NO
PUT AN ENTRY CODE OF 2 IN REGISTER 11
XCTL TO IGC0310D
CANCEL
IS THE ENTRY CODE = 2
YES
NO
OPCPOST
IS A CLOSEDOWN IN PROGRESS
YES
K5
NO
BUILD THE SVC 102 PARAMETER LIST TO TPOST THE RESPONSE MESSAGE
IS A MULTI-PROCESSOR PRESENT
YES
NO
DEQUEUE AN ELEMENT AND RESET THE 'QUEUING' FLAG
IS THE TERMINAL CONNECTED
NO
YES
PUT THE DESTINATION IN THE BUFFER
IS THE COMMAND CANCELED
YES
NO
IS THE DESTINATION VALID
YES
NO
PUT THE ALTERNATE DESTINATION IN THE BUFFER
IS THE BUFFER FOR CONSOLE OR AN APP PRO-GRAM
YES
G3
NO
SET THE 'CANCEL' FLAG
IS ANYONE QUEUING
YES
NO
EB A1
IGC102
TPOST BUFFER TO THE BUFFER DISPOSITION QCB
H5
GETNEXT
DELETE - THE FUNCTION MODULE
LOGSTART
PUT AN ENTRY CODE OF 1 IN REGISTER 11
K5
XCTL TO IGC0110D

Chart Z6 OPERATOR CONTROL CONTROL MODULE - LOAD 5


1
2
3
4
5
A
B
C
D
E
F
G
H
J
K
IGC0510D
ENTER
ESTABLISH ADDRESSABILITY
CALCULATE THE OUTPUT MESSAGE LENGTH
CALCULATE THE NUMBER OF BYTES IN THE BUFFER
WILL THE MESSAGE FIT IN THE BUFFER
YES
NO
PUT TIC COMMANDS IN THE BUFFER
CALCULATE THE NUMBER OF EXTRA BUFFER UNITS NEEDED
PUT THE MESSAGE IN THE BUFFER
BUILD AN ERB TO REQUEST THE EXTRA BUFFER UNITS
IS THE MESSAGE FOR A START/STOP DEVICE
YES
NO
PUT AN EOT IN THE BUFFER
EB A1
IGC102
TPOST THE ERB TO THE BUFFER REQUEST QCB
XCTL TO IGC0410D
WAIT - FOR THE ERB TO BE SATISFIED
CHAIN THE NEW UNITS INTO THE BUFFER

MICROFICHE DIRECTORY

The modules in the TCAM system have object module names that start with the letters IEDO. The modules that interface with the Operating System have an IGG prefix, the ERP modules have an IGE prefix, the nucleus resident modules have an IGC prefix, and TCAM-TSO modules have an IEDA prefix.

The first section of this directory contains entries for the executable TCAM modules. The second section of this directory contains entries for the non-executable generated modules.

EXECUTABLE TCAM MODULES MICROFICHE DIRECTORY

Module Name	Generic Name	Entry Point	Chart IDs
IEDAYA	TSO Attention Routine	IEDAYA	YA
IEDAYC	TSO Carriage Subroutine	IEDAYC	YC
IEDAYD	Time Sharing Destination Scheduler	IEDAYD	YD
IEDAYE	TSO TIOC Edit	IEDAYE	YE
IEDAYF	TSO IOHALT	IEDAYF	YF
IEDAYH	TSO Hangup	IEDAYH	YH
IEDAYI	TSINPUT Routine	IEDAYI	YI
IEDAYL	TSO Logon	IEDAYL	YL
IEDAYM	TSO Message Generation	IEDAYM	YM
IEDAYO	TSOUTPUT Routine	IEDAYO	YO
IEDAYR	STARTMH for TCAM-TSO Mixed	IEDAYR	YR
IEDAYS	TSO Simulated Attention	IEDAYS	YS
IEDAYT	TSO Abend Interface	IEDAYT	YT
IEDAYX	TSO INMSG/OUTMSG Linker	IEDAYX	YX
IEDAYY	TSO Asynchronous Time Delay Removal	IEDAYY	YY
IEDAYZ	Time Sharing Scheduler	IEDAYZ	YZ

Module Name	Generic Name	Entry Point	Chart IDs
IEDQAA	STARTMH	IEDQAA01	AA
IEDQAC	Date and Time Provision	IEDQAC01	AC
IEDQAD	Output Sequence Number Provision	IEDQAD01	AD
IEDQAE	Locate Option Field Address	IEDQAE	AE
IEDQAF	Insert Data	IEDQAF01	AF
IEDQAG	Message Limit	IEDQAG01	AG
IEDQAH	Input Sequence Number Insertion	IEDQAH01	AH
IEDQAI	Skip Forward and Scan	IEDQAI01	AI
IEDQAJ	Skip to Character Set	IEDQAJ01	AJ
IEDQAK	Line Control Insertion	IEDQAK01	AK
IEDQAL	Address Finder Routine	ADDRCOMP	AL
IEDQAM	Origin Routine	IEDQAM01	AM
IEDQAN	Multiple Insert/Remove	IEDQAN01	AN
IEDQAO	Unit Request Interface	IEDQAO01	AO
IEDQAP	Remove at Offset	IEDQAP01	AP
IEDQAQ	Operator Control Interface	IEDQAQ01	AQ
IEDQAR	Cancel Message	IEDQAR	AR
IEDQAS	Hold/Release Terminal	IEDQAS IEDQAS01 GETCPB LCBRTN	AS
IEDQAT	Create an Error Message	IEDQAT01 STCBAT+2	AT
IEDQATTN	Attention	IEDQATTN	TN
IEDQAU	Cutoff Message Transmission	IEDQAU CUTFFQCB+12	AU
IEDQAV	Lookup Terminal Entry	IEDQAV01	AV
IEDQAW	Translate Buffer	IEDQAW01	AW

Module Name	Generic Name	Entry Point	Chart IDs
IEDQAX	Buffer Step Routine	SCAN	AX
IEDQAY	Screen	IEDQAY01	AY
IEDQAZ	Redirect a Message	IEDQAZ01	AZ
IEDQA0	Skip Backward	IEDQA001	A0
IEDQA1	Binary Search	IEDQA101	A1
IEDQA2	Insert at Offset	IEDQA201	A2
IEDQA3	Dynamic Translation	IEDQA3	A3
IEDQA4	Incoming/Outgoing Message Delimiter	IEDQA401	A4
IEDQA5	Forward Routine	IEDQA501	A5
IEDQA6	Line Control Initialization	IEDQA601	A6
IEDQA7	Counter	IEDQA701	A7
IEDQA8	Multiple Insert at Offset	IEDQA801	A8
IEDQBA	Multiple Routing	IEDQBA01+12	BA
IEDQBB	Checkpoint Request	IEDQBB	BB
IEDQBC	Distribution List	IEDQBC	BC
IEDQBD	Buffer Disposition	IEDQBD01 IEDQBD02	BD
IEDQBE	Lock	IEDQBE	BE
IEDQBF	Unlock	IEDQBF	BF
IEDQBG	Cascade List	IEDQBG	BG
IEDQBL	Message Generation Routine	IEDQBL	BL
IEDQBT	FCB/FTB Handling	IEDQBT	BT
IEDQBW	Unit Request	IEDQBW IEDQ01	BW
IEDQBX	Log Segment	IEDQBX	BX
IEDQBY	Log Message	IEDQBY	BY

Module Name	Generic Name	Entry Point	Chart IDs
IEDOBZ	Loq Scheduler	IEDQBZ	BZ
IEDOCA	Resident Operator Control Module	IEDQCA01	CA
	Operator Control Scan Subroutine	IEDQCA02	CD
IEDOCF	Modify Options	IEDQCF	CF
IEDOCG	Copy Line Information	IEDQCG	CG
IEDOCH	Copy Terminal Information	IEDQCH	CH
IEDOCI	Copy LCB Information	IEDQCI	CI
IEDOCJ	Copy QCB Information	IEDQCJ	CJ
IEDOCK	Copy Held Terminals	IEDQCK	CK
IEDOCL	Copy Invitation List Entry	IEDQCL	CL
IEDOCM	Copy Operator Control Terminal	IEDQCM	CM
IEDOCN	Chanqe Control Terminal	IEDQCN	CN
IEDQCO	Chanqe Terminal	IEDQCO	CO
IEDOCP	Alter Trace Status	IEDQCP	CP
IEDOCQ	Stop/Resume Terminal Transmission	IEDQCQ	CQ
IEDOCU	Start Line	IEDQCU	CU
IEDOCV	Stop Line	IEDQCV	CV
IEDOCW	Modify Poll	IEDQCW	CW
IEDOCX	Modify Intense	IEDQCX	CX
IEDOCZ	Chanqe Interval Type	IEDQCZ	CZ
IEDOC0	MCP Closedown Processinq	IEDQC0	C0
IEDOC1	ICHNG Processinq	IEDQC1	C1
IEDQC2	On-Line Test Interface	IEDQC2	C2
IEDQC3	Copy Invitation List Status	IEDQC3	C3
IEDOC6	DEBUG Service Aid Router	IEDQC6	C6

Module Name	Generic Name	Entry Point	Chart IDs
IEDQEC	Put Scheduler	IEDQEC	EC
IEDQES	Retrieve Service	IEDQES	ES
IEDQET	Operator Control/Application Program Interface	IEDQET	ET
IEDQEU	Open/Close	IEDQEU	EU
IEDQEW	Get Scheduler	IEDQEW	EW
IEDQEZ	Get Scheduler FIFO	IEDQEZ	EZ
IEDQE1	TCOPY Service	IEDQE1	E1
IEDQE2	QCOPY Service	IEDQE2	E2
IEDQE3	TCHNG Service	IEDQE3	E3
IEDQE4	ICOPY Service	IEDQE4	E4
IEDQE6	Password Scrambler	IEDQE6	E6
IEDQE7	Retrieve Scheduler	IEDQE7	E7
IEDQFA	CPB Initialization CPB Cleanup Routine	IEDQFA IEDQFQ	FA
IEDQFA1	CPB Initialization-Main Storage Queuing Only	IEDQFA1 IEDQFQ	FA1
IEDQFA2	CPB Initialization-Disk Queuing Only	IEDQFA2 IEDQFQ	FA2
IEDQGA	Buffer Management Buffer Request Buffer Return Buffer Association	IEDQGA IEDQGA IEDQGB IEDQGD	GA
IEDQGT	Transparent Transmission CCW Building	IEDQGT	GT
IEDQHG	Time Delay	IEDQHG IEDQHG01 IEDQHG02 IEDQHG03	HG
IEDQHI	System Delay	IEDQHI	HI

Module Name	Generic Name	Entry Point	Chart IDs
IEDQHK	Stop Line I/O	IEDQHK01	HK
IEDQHM	Destination Scheduler	IEDQHM IEDQHM02	HM
IEDQHM1	Destination Scheduler-Main Storage Queuing Only	IEDQHM1	HM1
IEDQHM2	Destination Scheduler-Disk Queuing Only	IEDQHM2	HM2
IEDQKA	Activate-I/O Generator	IEDQKA IEDQKA02	KA
IEDQKB	Activate-I/O Generator for BSC Lines	IEDQKB IEDQKA02	KB
IEDQKC	Activate-I/O Generator for Start/Stop Lines	IEDQKC IEDQKCA02	KC
IEDQKD	Activate-I/O Generator for Leased and Start/Stop Lines and No TSO	IEDQKD IEDQKA02	KD
IEDQKE	Activate-I/O Generator for a QTAM Compatible System	IEDQKE IEDQKA02	KE KE
IEDQLM	Return Interface	IEDQLM	LM
IEDQNA	Resident Closedown Completicn	IEDQNA IEDQNA3	NA
IEDQNA2	Nonresident Closedown Completion	IEDQNA2	NA2
IEDQNB	Application Program/Checkpcint Interface	IEDQNB IEDQNB02 IEDQNB05	NB
IEDQND	Ready	IEDQND	ND
IEDQNF	Checkpoint Executor	IEDQNF	NF
IEDQNG	Build Incident Record for MH	IEDQNG	NG
IEDQNH	Build Incident Record for TCHNG	IEDQNH	NH
IEDQNJ	Incident Checkpoint for Operator Control	IEDQNJ	NJ
IEDQNK	Environment Checkpoint	IEDQNK	NK

Module Name	Generic Name	Entry Point	Chart IDs
IEDQNM	Build CKREQ Disk Record	IEDQNM	NM
IEDQNO	Checkpoint Queue Manager	IEDQNO	NO
IEDQNP	Checkpoint Disk I/O	IEDQNP	NP
IEDQNQ	Checkpoint Notification and Disposition	IEDQNQ	NQ
IEDQNR	Checkpoint-No Available Core	IEDQNR	NR
IEDQNS	Checkpoint-No Incident Records	IEDQNS	NS
IEDQNX	Operator Awareness Message Router	IEDQNX	NX
IEDQOA	Link	IEDQOA	OA
IEDQOB	WTOR Interpreter	IEDQOB	OB
IEDQOG	INTRO GETMAIN	IEDQOG	OG
IEDQOM	Termname Table Sort	IEDQOM	OM
IEDQOS	Attach Routine	IEDQOS	OS
IEDQTNT	Termname Table Code	IEDQTNT	NT
IEDQUI	User Interface	IEDQUI01	UI
IEDQXA	Disk Message Queue Initializer	IEDQXA	XA
IGC0010D	Operator Control Control Module - 0	IGC0010D	Z1
IGC0110D	Operator Control Control Module - 1	IGC0110D	Z2
IGC0210D	Operator Control Control Module - 2	IGC0210D	Z3
IGC0310D	Operator Control Control Module - 3	IGC0310D	Z4
IGC0410D	Operator Control Control Module - 4	IGC0410D	Z5
IGC0510D	Operator Control Control Module - 5	IGC0510D	Z6
IGC102	AQCTL SVC 102 Routine	IGC102	EB
IGC1303D	TCAM Command Scheduler - SVC 34	IGC1303D	NZ
IGE0004G	Start/Stop ERP Control Module	IGE0004G	JC

Module Name	Generic Name	Entry Point	Chart IDs
IGE0104G	Read/Write Unit Check and Unit Exception ERP Module	IGE0104G	JD
IGE0204G	Non-operational Control Unit	IGE0204G	JE
IGE0304G	Unit Check for Non-read, Non-write, and Non-poll CCWs ERP Module	IGE0304G	JF
IGE0404G	Auto Poll and Read Response to Poll Unit Check and Unit Exception ERP Module	IGE0404G	JG
IGE0504G	Error Post and Second Level CCW Return Module	IGE0504G	JH
IGE0604G	Unit Check and Unit Exception on Read/Write CCWs for Audio and 2260 Local Devices ERP Module	IGE0604G	JI
IGE0804G	Start/Stop Channel Check Module	IGE0804G	JJ
IGE0904G	Closedown Terminal Statistics Recording	IGE0904G	JK
IGE0004H	BSC ERP Control Module	IGE0004H	JL
IGE0104H	BSC Read/Write Equipment Check, Lost Data, Intervention Required, and Unit Exception ERP Module	IGE0104H	JM
IGE0204H	BSC Read/Write Data Check, Overrun, and Command Reject ERP Module	IGE0204H	JN
IGE0404H	BSC Second Level CCW Return Module	IGE0404H	JO
IGE0504H	BSC Error Post Module	IGE0504H	JP
IGE0804H	BSC Channel Check ERP Module	IGE0804H	JQ
IGG019Q0	Line I/O Interrupt Trace	IGG019Q0	Q0
IGG019Q1	Local Receive Scheduler	IGG019Q1	Q1
IGG019Q2	Line End Appendage for BSC Lines	IGG019Q2	Q2
IGG019Q3	Line End Appendage for Start/Stop Lines	IGG019Q3	Q3

Module Name	Generic Name	Entry Point	Chart IDs
IGG019Q4	Line End Appendage for Leased and Start/Stop Lines and No TSO	IGG019Q4	Q4
IGG019Q5	Line End Appendage for a QTAM Compatible System	IGG019Q5	Q5
IGG019Q6	Send Scheduler for Leased Lines and No TSO	IGG019Q6	Q6
IGG019Q7	Send Scheduler with No TSO	IGG019Q7	Q7
IGG019Q8	Checkpoint Continuation Restart Subroutine	IGG019Q8	Q8
IGG019RA	Checkpoint Disk End Appendage	IGG019RA	RA
IGG019RB	TCAM Dispatcher	IGG019RB	RB
IGG019RC	EXCP Driver	IGG019RC	RC
IGG019RD	Buffered Terminal Scheduler	IGG019RD	RD
IGG019RF	EXCP Driver for a Single CPB	IGG019RF	RF
IGG019RG	GET/READ	IGG019RG	RG
IGG019RH	GET Compatible	IGG019RH	RH
IGG019RI	PUT/WRITE	IGG019RI	RI
IGG019RJ	PUT Compatible	IGG019RJ	RJ
IGG019RK	Disk End Appendage for a Single CPB	IGG019RK	RK
IGG019RL	Check Routine	IGG019RL	RL
IGG019RM	Point Routine	IGG019RM	RM
IGG019RN	PCI Appendage	IGG019RN	RN
IGG019RO	TCAM Dispatcher with Subtask Trace	IGG019RO	RO
IGG019RP	Reusability-Copy	IGG019RP	RP
IGG019RQ	Post Pending	IGG019RQ	RQ
IGG019R0	Line End Appendage	IGG019R0	R0

Module Name	Generic Name	Entry Point	Chart IDs
IGG019R1	Dial Receive Scheduler	IGG019R1	R1
IGG019R2	Disk End Appendage	IGG019R2	R2
IGG019R3	Leased Receive Scheduler	IGG019R3 QEVENT	R3
IGG019R4	Send Scheduler	IGG019R4	R4
IGG019R5	Attention Handler	IGG019R5	R5
IGG019R6	Startup Message Routine	IGG019R6	R6
IGG01930	Disk Message Queues Open - 1	IGG01930	LB
IGG01931	Disk Message Queues Open - 2	IGG01931	LC
IGG01933	Open Error Handler	IGG01933	LA
IGG01934	Disk Message Queues Open - 3	IGG01934	LD
IGG01935	Line Group Open - Load 1	IGG01935	LE
IGG01936	Line Group Open - Load 2	IGG01936	LF
IGG01937	Line Group Open - Load 3	IGG01937	LG
IGG01938	Line Group Open - Load 4	IGG01938	LH
IGG01939	Line Group Open - Load 5	IGG01939	LI
IGG01940	Line Group Open - Load 6	IGG01940	LJ
IGG01941	Checkpoint Open	IGG01941	MA
IGG01942	Checkpoint Disk Initialization	IGG01942	MB
IGG01943	Checkpoint/Restart from Environment Record	IGG01943	ME
IGG01944	Checkpoint/Restart from Incident and CKREQ Records	IGG01944	MG
IGG01945	Checkpoint Continuation Restart	IGG01945	MJ
IGG01946	GET/PUT and READ/WRITE Open Executor - Load 1	IGG01946	L7
IGG01947	GET/PUT and READ/WRITE Open Executor - Load 2	IGG01947	L8

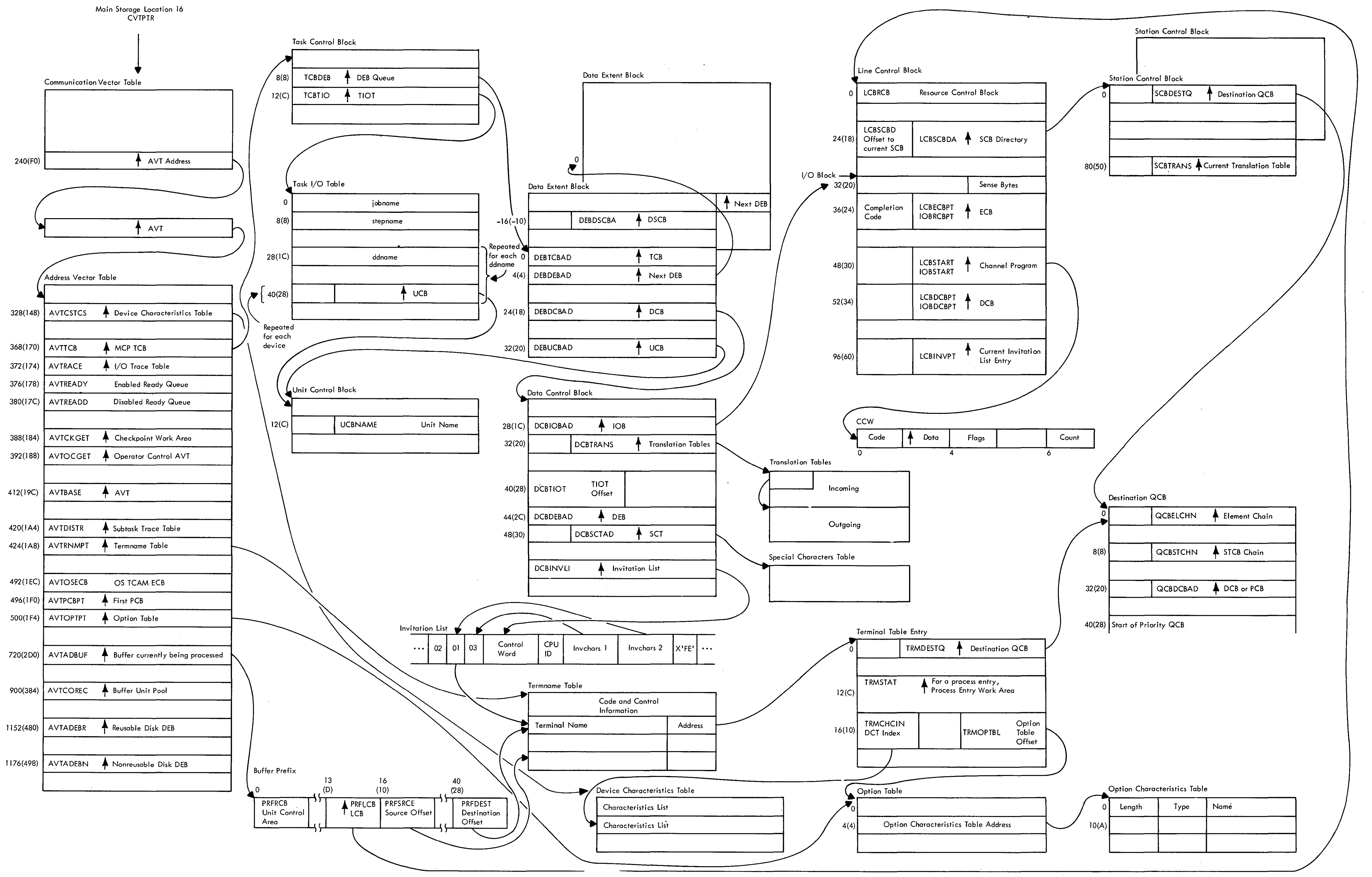
Module Name	Generic Name	Entry Point	Chart IDs
IGG01948	Line Group Open - Load 7	IGG01948	LK
IGG01949	Checkpoint Disk Allocation	IGG01949	MM
IGG02030	Disk Message Queues Close	IGG02030	L1
IGG02035	Line Group Close - Load 1	IGG02035	L4
IGG02036	Line Group Close - Load 2	IGG02036	L5
IGG02041	Checkpoint Close	IGG02041	L6
IGG02046	GET/PUT and READ/WRITE Close Executor - Load 1	IGG02046	L9
IGG02047	GET/PUT and READ/WRITE Close Executor - Load 2	IGG02047	L10

NON-EXECUTABLE TCAM MODULES MICROFICHE DIRECTORY

DSECT Name	Generic Name	DSECT Macro Name
IEDQAVTD	Address Vector Table	TAVTD
IEDQCCW	Channel Command Word	TCCWD
IEDQCDRD	Incident or Environment Checkpoint Disk Record	
IEDQCIBD	Command Input Block	CIB
IEDQCKPD	Checkpoint Work Area	TCKPD
IEDQCPB	Channel Program Block	TCPBD
IEDQCRED	Checkpoint Request Element - Incident or CKREQ	
IEDQC5	Operator Control Work Area	
IEDQDATA	Disk Data Record Area	TDATAD
IEDQDEB	Data Extent Block for TCAM Application Programs	TDEBAPD

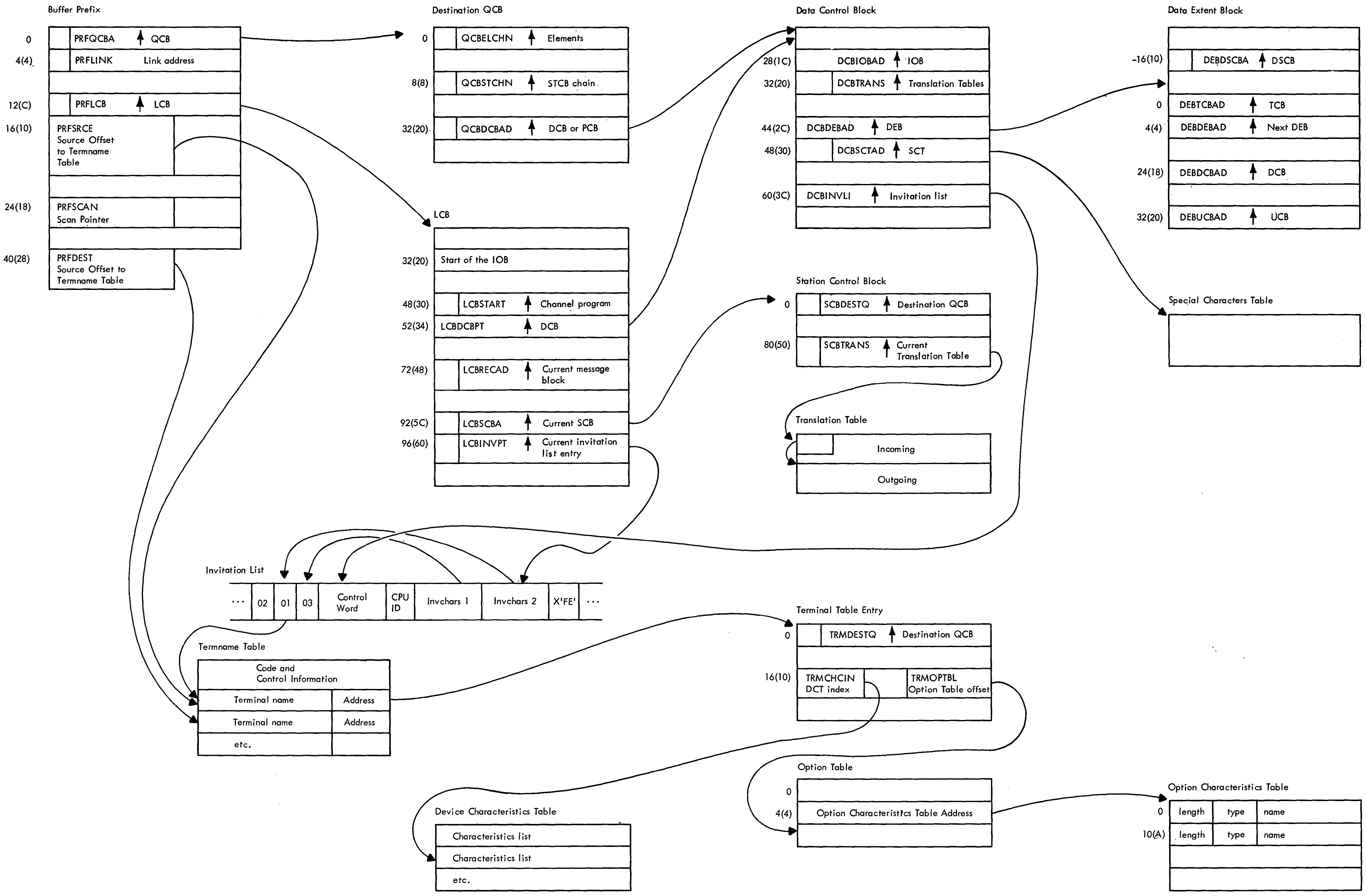
IEDQDEB	Data Extent Block	TDEBD
IEDQDISP	TCAM Dispatcher DSECT	TDISPD
IEDQIOB	Input/Output Block	TIOBD
IEDQLCB	Line Control Block	TLCBD
IEDQOPC	Operator Control AVT	TOPCD
IEDQPCB	Process Control Block	TPCBD
IEDQPEWA	Process Entry Work Area	TPEWAD
IEDQPQCB	Priority Queue Control Block	
IEDQPRF	Buffer Prefix	TPRFD
IEDQQCB	Queue Control Block	TQCBD
IEDQRECB	Resource Control Block	TRECBD
IEDQSCB	Station Control Block	TSCBD
IEDQSECT	Work Area Macro	FORECORE
IEDQSTCB	Subtask Control Block	TSTCBD
IEDQTCB	Task Control Block	TTCBD
IEDQTNTD	Termname Table	TTNTD
IEDQTRM	Terminal Table Entry	TTRMD
IEDQTSI	Time Sharing Queue Control Block	TTSID
IEDQWRKA	Access Method Work Area	TACSMD
IEDQXSA	Extended Save Area Macro	IEEXSA
IEDQ10	IBM 1030 Translate Table	
IEDQ11	IBM 1050 Translate Table	
IEDQ12	IBM 1050 Folded Translate Table	
IEDQ13	IBM 1060 Translate Table	
IEDQ14	IBM 2260 Translate Table	
IEDQ15	Alias for IEDQ14	
IEDQ16	IBM 2740 Translate Table	
IEDQ17	IBM 2740 Folded Translate Table	

IED018	World Trade Teletype Adapter (WTTA), ITA2 Translate Table	
IED019	World Trade Teletype Adapter (WTTA), ZSC3 Translate Table	
IED020	AT&T 115A or Western Union 83B3 Translate Table	
IED021	AT&T TWX, with Parity Translate Table	
IED022	AT&T TWX, without Parity Translate Table	
IED023	IBM 2780, 6-bit Code Translate Table	
IED024	USASCII Code Translate Table	
IED025	Dummy Table (EBCDIC to EBCDIC)	
IED026	IBM 2741, BCD Code Translate Table	
IED027	IBM 2741, EBCD Code Translate Table	
IED028	IBM 2741, Correspondence Code Translate Table	
IGG019RR	IBM 1030, 1050, 1060, 2740, 2741 Special Characters Table	
IGG019RS	IBM 2260 Remote Special Characters Table	
IGG019RT	AT&T 115A or Western Union 83B3 Special Characters Table	
IGG019RU	AT&T TWX, with Odd Parity Special Characters Table	
IGG019RV	IBM 2260 Local Special Characters Table	
IGG019RW	World Trade Teletype Adapter (WTTA) Special Characters Table	
IGG019RX	AT&T TWX, with Even Parity Special Characters Table	
IGG019RY	Audio Special Characters Table	
IGG019R7	BSC EBCDIC Code Special Characters Table	
IGG019R8	BSC USASCII Code Special Characters Table	
IGG019R9	BSC 6-bit Code Special Characters Table	
IHADCB	Data Control Block	DCBD


Main Storage Location 16
CVTPTR
Communication Vector Table
240(F0) AVT Address
AVT
Address Vector Table
328(148) AVTCSTCS Device Characteristics Table
368(170) AVTTCB MCP TCB
372(174) AVTRACE I/O Trace Table
376(178) AVTREADY Enabled Ready Queue
380(17C) AVTREADD Disabled Ready Queue
388(184) AVTCKGET Checkpoint Work Area
392(188) AVTOCGET Operator Control AVT
412(19C) AVTBASE AVT
420(1A4) AVTDISTR Subtask Trace Table
424(1A8) AVTRNMPT Termname Table
492(1EC) AVTOSECB OS TCAM ECB
496(1F0) AVTPCBPT First PCB
500(1F4) AVTOPTPT Option Table
720(2D0) AVTADBUF Buffer currently being processed
900(384) AVTCOREC Buffer Unit Pool
1152(480) AVTADEBR Reusable Disk DEB
1176(498) AVTADEBN Nonreusable Disk DEB
Task Control Block
8(8) TCBDEB DEB Queue
12(C) TCBTIO TIOT
Task I/O Table
0 jobname
8(8) stepname
28(1C) ddname
40(28) UCB
Repeated for each ddname
Repeated for each device
Unit Control Block
12(C) UCBNAME Unit Name
Data Extent Block
0
Data Extent Block
Next DEB
-16(-10) DEBDSCBA DSCB
0 DEBTCBAD TCB
4(4) DEBDEBAD Next DEB
24(18) DEBDCBAD DCB
32(20) DEBUCBAD UCB
Data Control Block
28(1C) DCBIOBAD IOB
32(20) DCBTRANS Translation Tables
40(28) DCBTIOT TIOT Offset
44(2C) DCBDEBAD DEB
48(30) DCBSCTAD SCT
DCBINVLI Invitation List
Translation Tables
Incoming
Outgoing
Special Characters Table
Invitation List
... 02 01 03 Control Word CPU ID Invchars 1 Invchars 2 X'FE' ...
Termname Table
Code and Control Information
Terminal Name Address
Buffer Prefix
0 PRFRCB Unit Control Area
13 (D) PRFLCB LCB
16 (10) PRFSRCE Source Offset
40 (28) PRFDEST Destination Offset
Device Characteristics Table
Characteristics List
Characteristics List
Line Control Block
0 LCBRCB Resource Control Block
24(18) LCBSCBD Offset to current SCB LCBSCBDA SCB Directory
I/O Block
32(20) Sense Bytes
36(24) Completion Code LCBECBPT IOBRCBPT ECB
48(30) LCBSTART IOBSTART Channel Program
52(34) LCBDCBPT IOBDCBPT DCB
96(60) LCBINVPT Current Invitation List Entry
CCW
Code Data Flags Count
0 4 6
Station Control Block
Station Control Block
0 SCBDESTQ Destination QCB
80(50) SCBTRANS Current Translation Table
Destination QCB
0 QCBELCHN Element Chain
8(8) QCBSTCHN STCB Chain
32(20) QCBDCBAD DCB or PCB
40(28) Start of Priority QCB
Terminal Table Entry
0 TRMDESTQ Destination QCB
12(C) TRMSTAT For a process entry, Process Entry Work Area
16(10) TRMCHCIN DCT Index TRMOPTBL Option Table Offset
Option Table
0
4(4) Option Characteristics Table Address
Option Characteristics Table
0 Length Type Name
10(A)

TCAM Control Block Linkages

LINKAGES FPOM A TCAM BUFFER PREFIX



ADDRESS VECTOR TABLE

The TCAM Address Vector Table (AVT) is assembled at the beginning of a Message Control Program. The basic AVT occupies bytes 0-1055 and is assembled when ENVIRON=TSO on the INTRO macro. If main-storage-only queuing is specified (DISK=NO,ENVIRON=TCAM or MIXED), the AVT occupies bytes 0 - 1079. When disk queuing is used, the AVT occupies bytes 0 - 1225.

When either the Disk Message Queues Open or the Line Group Open routine loads the TCAM Dispatcher, the routine also places in the CVT a pointer to a field that contains the address of the AVT. The fields in the AVT are initialized both during the assembly of the INTRO macro and at MCP initialization time.

The DSECT names of the AVT fields are shown in the following layout. A more detailed description of the fields and the data they might contain follows the DSECT layout.

IEDQAVTD

0 (0)	**AVTSAVE1**	Message Control Program Save Area
+72 (48)	**AVTSAVE2**	Dispatcher Save Area
+144 (90)	**AVTSAVE3**	Subtask Save Area
+216 (D8)	**AVTSAVE4**	First Level Subroutine Save Area
+288 (120)	**AVTSAVEX**	Disabled Save Area
+320 (140)	**AVTDLQ**	DLQ=Termname
+328 (148)	**AVTCSTCS**	Address of the First Entry in the Device Characteristics Table
+332 (14C)	**AVTDPARM**	Disabled Parameter List
+336 (150)	**AVTDOUBX**	Disabled Doubleword Scratch Area
+344 (158)	**AVTDOUBL**	Enabled Scratch Area
+352 (160)	**AVTCTLCH**	Operator Control Characters
+360 (168)	**AVTPASWD**	Password
+368 (170)	**AVTTCB**	Address of the Message Control Program's TCB; Set by OPEN
+372 (174)	**AVTRACE**	Trace Table Address
+376 (178)	**AVTREADY**	Enabled Ready Queue
+380 (17C)	**AVTREADD**	Disabled FIFO Ready Queue
+388 (184)	**AVTCKGET**	Checkpoint Work Area Address
+392 (188)	**AVTOCGET**	Operator Control Work Area Address

+396 (18C)
AVTEXA2S
Executed Instructions to Save the User's Registers

+402 (192)
AVTEXS2A
Executed Instructions to Save the User's Registers

408 (198)
AVTPARM
Address of Parameters

412 (19C)
AVTBASE
Address of the AVT

416 (1A0)
AVTPARM3
Address of Additional Optional Parameters

420 (1A4)
AVTDISTR
Address of the Dispatcher Subtask Trace Table

424 (1A8)
AVTRNMPT
Address of the Termname Table

428 (1AC)
AVTRDYA
Address of User Exit in the READY Macro Expansion

432 (1B0)
AVTBSCAN
Line End Appendage BSC Message Scan

436 (1B4)
AVTRARTN
Address of Routine to Update Line I/O Trace Table

440 (1B8)
AVTPOST
Tpost Parameter List Used by Operator Control

448 (1C0)
AVTSPLPT
Start Parameter List Pointer; Set by INTRO

452 (1C4) **AVTCIB** CIB=Integer	453 (1C5) **AVTNCKPR** CKREQS=Integer	454 (1C6) **AVTNOLBF** LNUNITS=Integer

456 (1C8)
AVTAS
Address of the Hold/Release Terminal Routine

460 (1CC)
AVTCKTCB
Address of the Checkpoint TCB

464 (1D0)
AVTOCTCB
Address of the Operator Control TCB

468 (1D4)
AVTOLTCB
Address of the On-Line Test TCB

472 (1D8)
AVTCWTCB
Address of the FE Common Write TCB

476 (1DC)		**AVTCWECA** FE Common Write ECB
480 (1E0)		**AVTCKECA** Checkpoint ECB
484 (1E4)		**AVTOLECA** On-Line Test ECB
488 (1E8)		**AVTOPECA** Operator Control ECB
492 (1EC)		**AVTOSECB** ECB Used by the Dispatcher to Cause TCAM Task to be in the Wait State
496 (1F0)		**AVTPCBPT** Address of the First Process Control Block
500 (1F4)		**AVTOPTPT** Address of the Option Table
504 (1F8)		**AVTKA02** Address of the I/O Generator in the Activate Subtask
508 (1FC)		**AVTREXIT** TREXIT=Name
512 (200)		**AVTCRSRF** CROSSRF=Integer
516 (204)		**AVTCOMPT** Address of Communications Parameter List
520 (208)		**AVTUI** Address of the User Interface Routine
524 (20C)		**AVTLM** Address of the Return Interface Routine
528 (210)	**AVTOLIST** OLTEST=Integer	**AVTHG02** Address of the Routine to Remove a Checkpoint Element from the Time Delay QCB
532 (214)		**AVTAL** Address of the Address Finder Routine
536 (218)		**AVTGD** Address of the Buffer Association Routine
540 (21C)		**AVTA3** Address of the Transparent CCW Builder Routine (IEDQBT)
544 (220)		**AVTAX** Address of the Buffer Step Routine

548 (224)	**AVTEA** Address of the TCAM Dispatcher
552 (228)	**AVTHA** Address of the Receive Scheduler
556 (22C)	**AVTHD** Address of the Send Scheduler
560 (230)	**AVTEW** Address of the Get Scheduler
564 (234)	**AVTEC** Address of the Put Scheduler
568 (238)	**AVTEZ** Address of the Get FIFO Scheduler
572 (23C)	**AVTBZ** Address of the Log Scheduler
576 (240)	**AVTR1** Address of the Dial Scheduler
580 (244)	**AVTHB** Address of the Buffered Scheduler
584 (248)	**AVTE7** Address of the Retrieve Scheduler
588 (24C)	Reserved
592 (250)	Reserved
596 (254)	Reserved
600 (258)	Reserved
604 (25C)	Reserved
608 (260)	Reserved
612 (264)	Reserved
616 (268)	Reserved

620 (26C) Reserved			
624 (270) **AVTDMECB** Dummy Line I/O ECB			
628 (274) **'AVTA3TL** Address of the Translate List for the Dynamic Translation Routine (IEDQA3)			
632 (278) **AVTTONE** WTTONE=Integer; Address of World Trade Tone Characters			
636 (27C) **AVTNX** Address of the Operator Awareness Message Routing Routine			
640 (280) **AVTIOT** Address of Line I/O Trace Table Handler			
644 (284) **AVTHI** Address of System Delay QCB			
648 (288) **AVTHK** Address of the Stopline QCB			
652 (28C) **AVTCKRMV** Request for Removal of Checkpoint Routine Element from Time Delay Queue			
668 (29C) **AVTCKELE** Checkpoint Request Element, Start of Checkpoint QCB			
676 (2A4) **AVTSCBSZ** SCB Size	677 (2A5) **AVTCKQAD** Address of the Checkpoint QCB		
680 (2A8) **AVTCKELF** Checkpoint Request Element Flags	681 (2A9) **AVTCPRCD** CPRCDS=Integer	682 (2AA) **AVTCKELV** CPINTVL=Time Interval	
684 (2AC) **AVTCKTIM** Time of Day Interrupt		686 (2AE) Index to QCB Address	287 (2AF) **AVTOPERL** OPEN Error Locator
688 (2B0) **AVTOPXCL** ID of OPEN Module with Error		690 (2B2) **AVTOPERT** OPEN Error Type	691 (2B3) **AVTCKBYT** Status at Checkpoint and Time Delay
692 (2B4) **AVTHG01** Address of Time Delay Subroutine			
696 (2B8) **AVTCKLNK** Link Field On the Time Queue			
700 (2BC) **AVTDELEM** Dummy Last Element			
704 (2C0) **AVTDELAD** Address of the Dummy Last Element			

708 (2C4)	AVTCCELE	Incident Checkpoint Request Element
716 (2CC)	AVTCLRHI	Mask for Clearing Left Two Bytes of a Register
718 (2CE)	AVTHFF	Half Word of X'FFFF'
720 (2D0)	AVTADBUF	Address of Buffer
724 (2D4)	AVT2260L	Address of 2260 Local Receive Scheduler
728 (2D8)	AVTSYSER	System Error Flags
729 (2D9)	AVTMSGS	List of Optional VCONs
732 (2DC)	AVTINSPT	Address of the QCB of Available Insert Blocks
736 (2E0)	AVTSUPPT	Address of the Start-up Message QCB
740 (2E4)	AVTTSOPT	Address of the Time Sharing Input QCB
744 (2E8)	AVTOCQPT	Address of the Application Program Open/Close Routine
748 (2EC)	AVTDELYB	Time Delay Subtask QCB
764 (2FC)	AVTREFTM	Reference Time
766 (2FE)	AVTINOUT	Dummy INEND/OUTEND Parameter List
768 (300)	AVTIMQPS	SVC 102 Parameter
776 (308)	AVTTIMQ	Time Delay Queue
780 (30C)	AVTBFREB	Buffer Request QCB
792 (318)	AVTBFRTB	Buffer Return QCB
804 (324)	AVTCKPTB	Checkpoint QCB
816 (330)	AVTOPCOB	Operator Control QCB
828 (33C)	AVTOLTQB	On-Line Test QCB

840 (348)	**AVTACTIB**	Activate QCB
852 (354)	**AVTCLOSB**	Closedown QCB
864 (360)	**AVTCPRMB**	QCB to Remove an Element from the Time Delay QCB
876 (36C)	**AVTDSIOB**	Disk I/O QCB
888 (378)	**AVTCPBCB**	CPB Cleanup QCB
900 (384)	**AVTCOREC**	Close Buffers Pool
904 (388)	**AVTCADDR**	Main Storage Queue Count
908 (38C)	**AVTFZERO**	Fullword of All Zeros
912 (390)	**AVTCAREA**	FE Common Write Interface Area — Address of the Patch Module
916 (394)	**AVTCWPM1**	FE Common Write Interface Area — First Parameter Pointer
920 (398)	**AVTCWEC1**	FE Common Write Interface Area — First ECB
924 (39C)	**AVTCWFL1**	FE Common Write — Flag Byte 1
925 (39D)	**AVTCWFL2**	FE Common Write — Flag Byte 2
926 (39E)	**AVTCWTS1**	FE Common Write — Flag Byte 3
927 (39F)	**AVTCWTS2**	FE Common Write — Flag Byte 4
928 (3A0)	**AVTCWPM2**	FE Common Write Interface Area — Second Parameter Pointer
932 (3A4)	**AVTCWEC2**	FE Common Write Interface Area — Second ECB
936 (3A8)	**AVTAFE10**	Address of FE STCB Trace Dump Routine
940 (3AC)	**AVTAFE20**	Address of FE I/O Trace Dump Routine
944 (3B0)	**AVTAFE30**	Address of FE Buffer Dump Routine
948 (3B4)	**AVTCWINT**	FE Common Write Interface Area — Patch Area

1012 (3F4)	AVTGETMN	GETMAIN Parameter List
1022 (3FE)	AVTHA2	Constant = 2
1024 (400)	AVTHA3	Constant = 3
1026 (402)	AVTHA4	Constant = 4
1028 (404)	AVTHA7	Constant = 7
1030 (406)	AVTHA16	Constant = 16
1032 (408)	AVTKEYLE	KEYLEN on the Message Queues
1034 (40A)	AVTLNCNT	Number of Lines Opened
1036 (40C)	AVTOPCNT	Number of Lines Taken by Operator Control
1038 (40E)	AVTOPCON	Termname Table Offset to the Primary Operator Control Terminal
1040 (410)	AVTAVFCT	Number of Buffers in the Buffer Units Pool
1042 (412)	AVTSMCNT	Number of Lines Serviced by the Start-up Message Subtask
1044 (414)	AVTINTLV	Number of Seconds of a System Delay INTVAL=Integer
1046 (416)	AVTDLQX	Offset in Termname Table of the Dead Letter Queue
1048 (418)	AVTDUMBR	Dummy Line Trace Table Update
1050 (41A)	AVTBIT1	Flag Bits
1051 (41B)	AVTBIT2	Flag Bits
1052 (41C)	AVTBIT3	Flag Bits
1053 (41D)	AVTCKRST	RESTART=Integer
1054 (41E)	AVTDSKCT	Number of Buffers on CPBs
1056 (420)	AVTHM02	Address of the Destination Scheduler
1060 (424)	AVTCMIN	MSMIN=Integer
1064 (428)	AVTCMAX	MSMAX=Integer
1068 (42C)	AVTTOTNC	Number of Records in the Entire Message Queues Data Set (MSUNITS=Integer)
1072 (430)	AVTNCPBQ	Queue of Buffers and ERBs Waiting to be Processed
1080 (438)	AVTFL	Address of the Disk EXCP Driver Routine
1084 (43C)	AVTIA	Address of the REUS part of the Reusability — Copy Subtask
1088 (440)	AVTCOPY	Copy Subtask QCB Pointer

1092 (444)	**AVTDKAPQ** Queue of CPBs to be Processed by CPB Cleanup (Disabled)
1100 (44C)	**AVTDKENQ** Queue of CPBs to be Processed by CPB Cleanup (Enabled)
1108 (454)	**AVTNOBFQ** Queue of CPBs without Buffers
1116 (45C)	**AVTREUSQ** Queue of CPBs Being Returned to the Reusability Subtask by CPB Cleanup
1124 (464)	**AVTINCPQ** Queue of CPBs Requesting I/O be Done by EXCP Driver
1132 (46C)	**AVTFCPB** Address of the CPB Free Pool
1136 (470)	**AVTCPBPT** Address of the CPB Free Pool to be Freed by Disk Close
1140 (474)	**AVTIOBR** Address of a Series of IOBs, One for Each Extent of the Reusable Disk Queue
1144 (478)	**AVTIOBN** Address of a Series of IOBs, One for Each Extent of the Nonreusable Disk Queue
1148 (47C)	**AVTLODPT** Absolute Disk Record Number Indicating Time to Activate the REUS part of the Reusability – Copy Subtask
1152 (480)	**AVTADEBR** Address of the DEBEOEA Field in the DEB for the Reusable Disk Message Queues Data Set
1156 (484)	**AVTNOVOR** Number of Extents in the Reusable Disk Message Queues Data Set
1160 (488)	**AVTRCTRR** Number of Records Per Track On the Reusable Disk Message Queues Data Set
1164 (48C)	**AVTTRCYR** Number of Tracks Per Cylinder On the Reusable Disk Message Queues Data Set
1168 (490)	**AVTTOTNR** Number of Records in the Entire Reusable Disk Message Queues Data Set
1172 (494)	**AVTVOLRR** Product of the Number of Extents Times the Number of Records Per Track On the Reusable Disk Message Queues Data Set
1176 (498)	**AVTADEBN** Address of the DEBEOEA Field in the DEB for the Nonreusable Disk Message Queues Data Set
1180 (49C)	**AVTNOVON** Number of Extents in the Nonreusable Disk Message Queues Data Set

1184 (4A0)	**AVTRCTRN**	Number of Records Per Track On the Nonreusable Disk Message Queues Data Set
1188 (4A4)	**AVTTRCYN**	Number of Tracks Per Cylinder On the Nonreusable Disk Message Queues Data Set
1192 (4A8)	**AVTTOTNN**	Number of Records in the Entire Nonreusable Disk Message Queues Data Set
1196 (4AC)	**AVTVOLRN**	Product of the Number of Extents Times the Number of Records Per Track On the Nonreusable Disk Message Queues Data Set
1200 (4B0)	**AVTHRESN**	Absolute Record Number (Threshold) to Cause Closedown Due to the Filling of the Nonreusable Disk Message Queues Data Set
1204 (4B4)	**AVTNADDR**	Nonreusable Disk Relative Record Number of the Next Record to be Assigned
1208 (4B8)	**AVTRADDR**	Reusable Disk Relative Record Number of the Next Record to be Assigned
1212 (4BC)	**AVTHRESE**	Nonreusable Threshold Closedown Element
1223 (4C7)	**AVTHRESS**	Status Completion Code
1224 (4C8)	**AVTCPBNO**	CPB=Integer
1226 (4CA)		Reserved

Offset		Name	Bytes	Description
0	(0)	AVTSAVE1	72	Message Control Program save area
72	(48)	AVTSAVE2	72	Dispatcher save area
144	(90)	AVTSAVE3	72	Subtask save area
216	(D8)	AVTSAVE4	72	First level subroutine save area
288	(120)	AVTSAVEX	40	Disabled save area
320	(140)	AVTDLQ	8	At assembly time, set by the DLQ=termname operand of the INTRO macro. After the Termname Table is sorted, this value is moved to AVTDLQX and this field (AVTDLQ) is overlaid and used as part of the disabled save area.
328	(148)	AVTCSTCS	4	Address of the first entry in the Device Characteristics Table
332	(14C)	AVTDPARM	4	Disabled parameter list (used with AVTDOUBX)
336	(150)	AVTDOUBX	8	Disabled doubleword scratch area
344	(158)	AVTDOUBL	8	Enabled doubleword scratch area
352	(160)	AVTCTLCH	8	Operator Control characters
360	(168)	AVTPASWD	8	Message Control Program password

Offset		Name	Bytes	Description
368	(170)	AVTTCB	4	Address of the Message Control Program TCB - set by the first OPEN routine
372	(174)	AVTRACE	4	Line I/O Interrupt Trace Table address

The following are the ready queues for the TCAM Dispatcher:

Offset		Name	Bytes	Description
376	(178)	AVTREADY	4	Enabled ready queue - points to the first item in the chain of elements that is to be processed by the TCAM Dispatcher
380	(17C)	AVTREADD	8	Disabled FIFO ready queue - controls the chain of elements tposted from disabled routines. The first word points to the first element; the second word points to the last element on the chain.
388	(184)	AVTCKGET	4	Address of the Checkpoint work area; set after a successful GETMAIN is completed by the Checkpoint Open routine
392	(188)	AVTOCGET	4	Address of the Operator Control work area
396	(18C)	AVTEXA2S	6	Instructions to be executed to save the user's registers
402	(192)	AVTEXS2A	6	Continuation of the instructions to be executed to save the user's registers
408	(198)	AVTPARM	4	Address of the parameters to be processed
412	(19C)	AVTBASE	4	Address of the AVT
416	(1A0)	AVTPARM3	4	Address of additional optional parameters
420	(1A4)	AVTDISTR	4	Address of the Dispatcher's Subtask Trace Table
424	(1A8)	AVTRNMPT	4	Address of the Termname Table
428	(1AC)	AVTRDYA	4	User exit address in the READY macro expansion
432	(1B0)	AVTBSCAN	4	Line End Appendage Address for BSC message scan
436	(1B4)	AVTRART	4	Address of the routine to update the Line I/O Interrupt trace table
440	(1B8)	AVTPOST	8	Tpost parameter list used by Operator Control
448	(1C0)	AVTSPLPT	4	Start parameter list address - set by the INTRO macro expansion
452	(1C4)	AVTCIB	1	The maximum number of Command Input Blocks that can be utilized at any one time in the TCAM system - set by the CIB=integer operand of the INTRO macro

Offset		Name	Bytes	Description
453	(1C5)	AVTNCKPR	1	The maximum decimal number of destination queues in use at any time for application programs that use a CKREQ macro - set by the CKREQS=integer operand of the INTRO macro
454	(1C6)	AVTNOLBF	2	Specifies the number of buffer units that may be used to build buffers for messages - set by the LNUNITS=integer operand of the INTRO macro
456	(1C8)	AVTAS	4	Address of the Hold/Release terminal routine

The following are the addresses of the TCBs of the attached tasks:

460	(1CC)	AVTCKTCB	4	Address of the Checkpoint TCB
464	(1D0)	AVTOCTCB	4	Address of the Operator Control TCB
468	(1D4)	AVTOLTCB	4	Address of the On-Line Test TCB
472	(1D8)	AVTCWTCB	4	Address of the FE Common Write TCB

The following are the Event Control Blocks (ECBs) for the attached tasks:

476	(1DC)	AVTCWECA	4	FE Common Write ECB
480	(1E0)	AVTCKECA	4	Checkpoint ECB
484	(1E4)	AVTOLECA	4	On-Line Test ECB
488	(1E8)	AVTOPECA	4	Operator Control ECB
492	(1EC)	AVTOSECB	4	ECB used by the Dispatcher to cause the TCAM task to be in the WAIT state
496	(1F0)	AVTPCBPT	4	Address of the first Process Control Block
500	(1F4)	AVTOPTPT	4	Address of the Option Table
504	(1F8)	AVTKA02	4	Address of the I/O Generator routine in the Activate subtask
508	(1FC)	AVTREXIT	4	Address of a user written routine to be given control when all entries in the TCAM I/O Interrupt Trace Table have been filled - set by the TREXIT=name operand of the INTRO macro
512	(200)	AVTCRSRF	4	Specifies the number of entries in the Cross Reference Table - set by the CROSSRF=integer operand of the INTRO macro. Replaced by the address of the Cross Reference Table.
516	(204)	AVTCOMPT	4	Address of the Communications Parameter List
520	(208)	AVTUI	4	Address of the User Interface routine

Offset		Name	Bytes	Description
524	(20C)	AVTLM	4	Address of the Return Interface routine
528	(210)	AVTHG02	4	Address of the routine to remove a Checkpoint element from the Time Delay QCB
528	(210)	AVTOLTST	1	Set by the OLTEST=integer operand of the INTRO macro
532	(214)	AVTAL	4	Address of the Address Finder routine
536	(218)	AVTGD	4	Address of the Buffer Association routine
540	(21C)	AVTA3	4	Address of the Transparent Transmission CCW Building routine (IEDQBT)
544	(220)	AVTAX	4	Address of the Buffer Step routine
548	(224)	AVTEA	4	Address of the TCAM Dispatcher
552	(228)	AVTHA	4	Address of the Receive Scheduler
556	(22C)	AVTHD	4	Address of the Send Scheduler
560	(230)	AVTEW	4	Address of the Get Scheduler
564	(234)	AVTEC	4	Address of the Put Scheduler
568	(238)	AVTEZ	4	Address of the Get FIFO Scheduler
572	(23C)	AVTBZ	4	Address of the Log Scheduler
576	(240)	AVTB1	4	Address of the Dial Scheduler
580	(244)	AVTHB	4	Address of the Buffered Scheduler
584	(248)	AVTE7	4	Address of the Receive Scheduler

The following are the special elements used in TCAM:

Offset		Name	Bytes	Description
588	(24C)		36	Reserved
624	(270)	AVTDMECB	4	Address of the dummy line I/O ECB
628	(274)	AVTA3TL	4	Address of the translate list for the Dynamic Translation routine
632	(278)	AVTTONE	4	Contains either a zero or the address of a field consisting of 2 halfwords; the first contains the WTTONE integer from the INTRO macro and the second a X'FF' representing the number of characters specified by WTTONE.
636	(27C)	AVTNX	4	Address of Operator Awareness Message Routing routine
640	(280)	AVTIOT	4	Address of Line I/O Trace Table routine

Offset		Name	Bytes	Description
644	(284)	AVTHI	4	Address of System Delay QCB
648	(288)	AVTHK	4	Address of Stopline QCB
652	(28C)	AVTCKRMV	16	Request for removal of Checkpoint routine element from the time delay queue
668	(29C)	AVTCKRLE	8	Checkpoint Request Element - the Time Delay or Reusability subtasks tpost this element to start the Checkpoint routine
676	(2A4)	AVTSCBSZ	1	Specifies the number of bytes in the SCB including the save area for the user's registers.
677	(2A5)	AVTCKQAD	3	Address of Checkpoint QCB
680	(2A8)	AVTCKRLF	1	Checkpoint Request Element flag bits

Bit definitions:

Name	Bit	Value	Description
AVTCRDYN	0	X'80'	Checkpoint requested by the READY macro expansion
AVTCMCPN	1	X'40'	Checkpoint requested by the MCPCLOSE macro
	2	X'20'	Unused
AVTCINCN	3	X'10'	Checkpoint requested by the No Incident Records routine
AVTCCLCN	4	X'08'	Closedown completion bit
AVTCPIPN	5	X'04'	Checkpoint in progress bit
AVTCRTLN	6	X'02'	Checkpoint requested
AVTWARM	7	X'01'	Warm restart

Offset		Name	Bytes	Description
681	(2A9)	AVTCPRCD	1	The number of environment checkpoint records to be retained in the Checkpoint Data Set at any one time - set by the CPRCDS=integer operand of the INTRO macro
682	(2AA)	AVTCKELV	2	The number of seconds between environment checkpoints - set by the CPINTVL=integer operand of the INTRO macro
684	(2AC)	AVTCKTIM	2	Time of day interrupt
686	(2AE)		1	Index to the QCB address
687	(2AF)	AVTOPERL	1	Open Error location
688	(2B0)	AVTOPXCL	2	Module ID of the routine that has an error
690	(2B2)	AVTOPERT	1	Specifies the type of open error that occurred

Offset		Name	Bytes	Description
691	(2B3)	AVTCKBYT	1	Specifies the checkpoint and time delay status
692	(2B4)	AVTHG01	4	Address of the Time Delay Subroutine
696	(2B8)	AVTCKLNK	4	Link field on the Time Queue
700	(2BC)	AVTDELEM	4	Dummy last element - used as the last element in any QCB's (or the ready queue's) element chain
704	(2C0)	AVTDELAD	4	Address of the dummy last element
708	(2C4)	AVTCCELE	8	Incident Checkpoint Request element - posted by the Operator Control task to request an incident checkpoint
716	(2CC)	AVTCLRHI	2	Mask used with the next halfword to clear the left two bytes of a register.
718	(2CE)	AVTFF	2	Halfword equal to X'FFFF'
720	(2D0)	AVTADBUF	4	Address of the buffer currently being processed
724	(2D4)	AVT2260L	4	Address of the 2260 Local Receive Scheduler
728	(2D8)	AVTSYSER	1	System error flag byte - set by the operands of the INTRO macro as follows:

Name	Bit	Value	Description
AVTCMINN	0	X'80'	The number of main storage queue units less than that specified by MSMIN=integer
AVTCMAXN	1	X'40'	The number of main storage queue units more than that specified by MSMAX=integer
	2-7		Reserved

Offset		Name	Bytes	Description
729	(2D9)	AVTMSGS	3	Address of a list of optional VCONs

The following is a list of pointers to QCBs:

Offset		Name	Bytes	Description
732	(2DC)	AVTINSPT	4	Address of the QCB of Available Insert blocks
736	(2E0)	AVTSUPPT	4	Address of the Startup Message QCB
740	(2E4)	AVTTSOPT	4	Address of the Time Sharing Input QCB
744	(2E8)	AVTOCQPT	4	Address of the application program Open/Close subtask

The following is a list of non-optional QCBs:

Offset		Name	Bytes	Description
748	(2EC)	AVTDELYB	20	Time Delay subtask QCB

Offset		Name	Bytes	Description
764	(2FC)	AVTREFTM	2	Represents the reference time, current time of day, plus or minus 6 hours
766	(2FE)	AVTINOUT	2	Dummy INEND/OUTEND parameter list
768	(300)	AVTIMOPS	8	SVC 102 parameter - to tpost the Time QCB to itself at the interrupt
776	(308)	AVTTIMQ	4	Time delay queue
780	(30C)	AVTBFREB	12	Buffer Request QCB
792	(318)	AVTBFRTB	12	Buffer Request QCB
804	(324)	AVTCKPTB	12	Checkpoint QCB
816	(330)	AVTOPCOB	12	Operator Control QCB
828	(33C)	AVTOLTOB	12	On-line Test QCB
840	(348)	AVTACTIB	12	Activate QCB
852	(354)	AVTCLOSB	12	Closedown Completion QCB
864	(360)	AVTCPRMB	12	QCB to remove an element from the Time Delay QCB
876	(36C)	AVTDSIOB	12	Disk I/O QCB
888	(378)	AVTCPBCB	12	CPB Cleanup QCB
900	(384)	AVTCOREC	4	Close buffers pool
904	(388)	AVTCADDR	4	Main Storage queue count
908	(38C)	AVTFZERO	4	Fullword of zeros

The following is the FE Common Write task interface area:

Offset		Name	Bytes	Description
912	(390)	AVTCAREA	4	Address of the Patch module for this task
916	(394)	AVTCWPM1	4	First parameter list point for this task
920	(398)	AVTCWEC1	4	First ECB for this task
924	(39C)	AVTCWFL1	1	First flag byte for this task

Bit definitions:

Name	Bit	Value	Description
AVTCOMWN	0	X'80'	Specifies that the FE Common Write task is attached; set by the COMWRTE=YES operand of the INTRO macro

Offset		Name	Bytes	Description
925	(39D)	AVTCWFL2	1	Second flag byte for this task

Offset		Name	Bytes	Description			
			Name	Bit	Value	Description	
			AVTCWACT	0	X'80'	Specifies that the FE Common Write task is active; set by the COMWRITE=YES operand of the INTRO macro.	
926	(39E)	AVTCWTS1	1	Third flag byte for this task			
927	(39F)	AVTCWTS2	1	Fourth flag byte for this task			
928	(3A0)	AVTCWPM2	4	Second parameter pointer for this task			
932	(3A4)	AVTCWEC2	4	Second ECB for this task			
936	(3A8)	AVTAFE10	4	Address of the FE STCB Trace Dump routine - IEDQFE10			
940	(3AC)	AVTAFE20	4	Address of the FE I/O Trace Dump routine - IEDQFE20			
944	(3B0)	AVTAFE30	4	Address of the FE Buffer Dump routine - IEDQFE30			
948	(3B4)		64	Patch area for this task			
1012	(3F4)	AVTGETMN	10	GETMAIN parameter list			
1022	(3FE)	AVTHA2	2	Constant = 2			
1024	(400)	AVTHA3	2	Constant = 3			
1026	(402)	AVTHA4	2	Constant = 4			
1028	(404)	AVTHA7	2	Constant = 7			
1030	(406)	AVTHA16	2	Constant = 16			
1032	(408)	AVTKEYLE	2	Specifies the size in bytes of a buffer unit - set by the KEYLEN=integer operand of the INTRO macro			
1034	(40A)	AVTLNCNT	2	Number of lines opened - set by the Line Group Open routine - checked by the Time Delay subtask			
1036	(40C)	AVTOPCNT	2	Number of lines taken by the Operator Control task - set by the System Delay subtask and the Operator Control task			
1038	(40E)	AVTOPCON	2	Termname Table offset to the entry for the primary Operator Control terminal - set by the PRIMARY=termname operand of the INTRO macro			
1040	(410)	AVTAVFCT	2	Number of buffers in the buffer unit pool - this value is equal to the sum of the LNUNITS=integer and the MSUNITS=integer operands of the INTRO macro			
1042	(412)	AVTSMCNT	2	Number of lines serviced by the Startup Message subtask			

Offset		Name	Bytes	Description
1044	(414)	AVTINTLV	2	Number of seconds of a system delay - set by Operator Control or by the INVTAL=integer operand of the INTRO macro; checked by the Time Delay subtask
1046	(416)	AVTDLOX	2	Termname Table offset of the dead letter queue - moved from the AVTDLQ field of the AVT after the Termname Table is sorted at execution time
1048	(418)	AVTDUMBR	2	Dummy Line I/O Interrupt Trace Table update
1050	(41A)	AVTBIT1	1	Flag bits

Bit Definitions:

Name	Bit	Value	Description
AVTAPIKN	0	X'80'	Prevents the Disk End appendage from adding a CPB to the Disabled Disk End QCB for CPB Cleanup
AVTAPIKF	0 Off	X'7F'	Mask to permit the Disk End appendage to add a CPB to the Disabled Disk End QCB for CPB Cleanup
AVTTSON	1	X'40'	Specifies that the TCAM environment has TSO or is mixed - set by the ENVIRON=TSO or MIXED operand of the INTRO macro
AVTAQTAN	2	X'20'	Specifies that the system environment has TCAM or is mixed - set by the ENVIRON=TCAM or MIXED operand of INTRO
AVTDLAYN	3	X'10'	Specifies that a system delay is in effect - set by the Operator Control task
AVTDLAYF	3 Off	X'EF'	Mask to specify that a system delay is not in effect - bit 3 is turned off by the Time Delay subtask
AVTREADN	4	X'08'	Specifies that the READY macro expansion has been executed - set by the READY macro expansion; checked by the Open routines
AVTCLOSN	5	X'04'	Close down indicator: 0 - closedown not requested

Offset	Name	Bytes	Description		
					1 - closedown requested
		AVTQUCKN	6	X'02'	Type of clcsedown: 0 - Flush closedown 1 - Quick closedown
		AVTDISKN	7	X'01'	Specifies that none of the message queues data sets are disk queued
1051 (41B)	AVTBIT2	1	Flag bits		

Bit definitions:

Name	Bit	Value	Description
AVTRUFTN	0	X'80'	Reusability first time switch - set by Destination Assignment; checked by CPB Cleanup; turned off by Reusability-Copy
AVTRUF	0 Off	X'7F'	Mask for the "Reusability first time" switch turned off by Reusability
AVTREUSN	1	X'40'	Specifies that Reusability is running - set by Reusability; checked by CPB Cleanup
AVTREUSF	1 Off	X'BF'	Mask to specify that Reusability is not running - turned off by Reusability
AVTCOPYN	2	X'20'	Specifies that the Reusability-Copy function is requesting control
	3	X'10'	Specifies that TOPMSG=NO is set in the INTRO macro
AVTSTRTN	4	X'08'	Restart is in progress
AVTSTRTF	4 Off	X'F7'	Mask to specify that restart is not in progress
AVTOPEIN	5	X'04'	Initial load done indicator
	6,7	X'03'	Specifies the line type as nonswitched Start/Stop only set by the Activate routine or the Line End Appendage
	6	X'02'	Specifies the line type as Start/Stop, switched or nonswitched - set by the Activate routine or the Line End Appendage
	7	X'01'	Specifies the line type as binary synchronous - set by the Activate routine Line End Appendage
	All off		Specifies the line type as both BSC and Start/Stop, switched and nonswitched, all possible line combinations - set by the Activate routine or Line End Appendage

Offset	Name	Bytes	Description
1052 (41C)	AVTBIT3	1	Flag bits

Bit Definitions:

Name	Bit	Value	Description
AVTSTAN	7	X'01'	Specifies that either a Cold or Warm restart is to perform following a normal quick close or a flush close - set by STARTUP=C or STARTUP=W operand of the INTRO macro
AVTSTACN	6	X'02'	Specifies that a Cold start is to be performed following a normal quick or a flush close and that a Continuation Restart is to be performed following system failure - set by the STARTUP=C operand of the INTRO macro
AVTSTAWN	6, off	X'FD'	Mask to specify that a warm restart is to be performed following a normal quick or a flush Close and that a Continuation Restart is to be performed following system failure - set by the STARTUP=W operand of the INTRO macro
AVTSTAIN	5	X'04'	Specifies that the status of each invitation list is to be included in the checkpoint record - set by the STARTUP=I operand of the INTRO macro
AVTSTAYN	4	X'08'	Specifies that no Continuation Restart is to be performed following a normal quick close, a flush clcse, or system failure - set by the STARTUP=Y operand of the INTRO macro
AVTOLTEN	3	X'10'	Specifies that the maximum size in the OLTEST=keyword operand in the INTRO macro (the maximum number of on-line tests that can be performed) has been reached - set, checked, and reset by TOTE
AVTTSAB	2	X'20'	Specifies that TSO has abended - set by the Time Sharing Abend module; checked by the TSINPUT and TSOUTPUT routines; reset by the Start Time Sharing routine
AVTRFULN	1	X'40'	Reusable disk zone full - set by Reusability

Offset		Name	Bytes	Description	
			AVTRFULF	0,2, X'BF' 3 off	Mask to specify that reusable disk is ready to receive - checked by Receive Scheduler and Line End Appendage; turned off by Reusability
			AVTRECVN	0 X'80'	Main storage queue is full - set by Destination Scheduler when the number of main storage queue units > or = the number specified in the MSMAX operand of the INTRO macro; turned off by Disk I/O; checked by the Receive Scheduler and Line End Appendage.
1053	(41D)	AVTCKRST	1	Specifies which checkpoint record the TCAM restart facility should use in attempting to restructure the MCP environment as it existed at the time of closedown or system failure - set by the RESTART=integer operand of the INTRO macro	
1054	(41E)	AVTDSKCT	2	Specifies the number of buffers on CPBs	

****

This is the end of the basic AVT when ENVIRON=TSO

****

Offset		Name	Bytes	Description
1056	(420)	AVTMH02	4	Address of the Destination Scheduler
1060	(424)	AVTCMIN	4	Specifies the percentage of the number of units in the message queues data below which the data set is not crowded - set by the MSMIN=integer operand of the INTRO macro
1064	(428)	AVTCMAX	4	Specifies the percentage of the number of units in the message queues data set above which the data set is nearly full - set by the MSMAX=integer operand of the INTRO macro
1068	(42C)	AVTTOTNC	4	Number of records in the entire message queues data set - set by the MSUNITS=integer operand of the INTRO macro
1072	(430)	AVTNCPBQ	8	Queue of buffers and ERBs waiting to be processed

****

This is the end of the AVT when
main storage queuing only is specified
(DISK=NO, ENVIRON=TCAM or MIXED)

****

Offset		Name	Bytes	Description
1080	(438)	AVTFL	4	Address of the Disk EXCP Driver routine
1084	(43C)	AVTIA	4	Address of the REUS part of the Reusability-Copy subtask
1088	(440)	AVTCOPY	4	Address of the Copy subtask QCB
1092	(444)	AVTDKAPQ	8	Queue of the CPBs to be processed by CPB Cleanup (disabled)
1100	(44C)	AVTDKENQ	8	Queue of CPBs to be processed by CPB Cleanup (enabled)
1108	(454)	AVTNOBFQ	8	Queue of CPBs without buffers - used by CPB Cleanup
1116	(45C)	AVTREUSQ	8	Queue of CPBs being returned to the Reusability-Copy subtask by CPB Cleanup
1124	(464)	AVTINCPQ	8	Queue of CPBs requesting that I/O be done by EXCP Driver
1132	(46C)	AVTFCPB	4	Queue of inactive CPBs - the CPB free pool
1136	(470)	AVTCPBPT	4	Address of the CPB free pool to be freed by the Disk Close routine - AVTFCPB is initially set to this same value
1140	(474)	AVTIOBR	4	Address of a series of IOBs, one for each extent of the reusable disk queue
1144	(478)	AVTIOBN	4	Address of a list of IOBs, one for each extent of the nonreusable disk queue
1148	(47C)	AVTLODPT	4	Absolute disk record number that indicates when the REUS part of the Reusability-Copy subtask is to activated - the initial value is 3/8 of the total number of records on the reusable disk message queues data set

The next 6-word area is initiated by the OPEN for the reusable disk message queues data set for use by the Record Number to MBBCCHHR Converter routine.

1152	(480)	AVTADFBR	4	Address of the DEBECEA field in the DEB for the reusable disk message queues data set
1156	(484)	AVTNOVOR	4	Number of extents in the reusable disk message queues data set
1160	(488)	AVTRCTRK	4	Number of records per track on the reusable disk message queues. data set

Offset		Name	Bytes	Description
1164	(48C)	AVTTRCYR	4	Number of tracks per cylinder on the reusable disk message queues data set
1168	(490)	AVTTOTNR	4	Number of records in the entire reusable disk message queues data set
1172	(494)	AVTVOLRR	4	Product of the number of extents times the number of records per track on the reusable disk message queues data set

The next 7-word area is initialized by the CPEN for the nonreusable disk message queues data set for use by the Record Number of MBBCCHHR Converter routine.

Offset		Name	Bytes	Description
1176	(498)	AVTADEBN	4	Address of the DEBECEA field in the DEB for the nonreusable disk message queues data set
1180	(49C)	AVTNOVON	4	Number of extents in the nonreusable disk message queues data set
1184	(4A0)	AVTRCTRN	4	Number of records per track on the nonreusable disk message queues data set
1188	(4A4)	AVTTRCYN	4	Number of tracks per cylinder on the nonreusable disk message queues data set
1192	(4A8)	AVTTOTNN	4	Number of records in the entire nonreusable disk message queues data set
1196	(4AC)	AVTVOLRN	4	Product of the number of extents times the number of records per track on the ncnreusable disk message queues data set
1200	(4B0)	AVTHRESN	4	The absolute record number that is the threshold tc cause closedown due to the filling cf the nonreusable disk message queues data set
1204	(4B4)	AVTNADDR	4	Nonreusable disk relative record number - next available lccation
1208	(4B8)	AVTRADDR	4	Reusable disk relative record number - next available location
1212	(4BC)	AVTHRESE	12	Nonreusable threshold closedown element
1223	(4C7)	AVTHRESS	1	Completion code - used to indicate status

Offset	Name	Bytes	Description
			X'FF' - an unused element X'F0' - the element has been tposted X'00',X'04' - Closedown indication
1224 (4C8)	AVTCPBNO	2	Specifies the value coded in the CPB=integer operand of the INTRO macro

TERMINAL TABLE

The Terminal Table (IEDQTRM) is a variable length table that contains blocks of device-dependent information about each terminal in the TCAM system; each such block is called a terminal entry. There are six types of terminal entries (shown below), each of which is used for a different type or group of terminals depending upon the configuration of the teleprocessing system.

The Terminal Table entries are assembled and initialized according to the specifications of the TERMINAL, TLIST, TPROCESS, TTABLE, LOGTYPE, and OPTION macro instructions. The size, structure, and contents of the Terminal Table are based on the information provided by the user in the above-listed macros. Each entry in the Terminal Table begins on a fullword boundary. The terminal entries are located through the address portion of the entries in the Termname Table.

If the user codes an OPTION macro, three fields in the Terminal Table entry are initialized and bit 6 in the TRMSTATE field is set to 1. The TRMOPNO field contains the number of option fields specified for the entry. The option offsets are positional in nature, and the number of offsets is equal to all the offsets up to and including the last option specified by the user. The next field, TRMOPTBL, contains the offset to the beginning of the Option Table data for this terminal entry. The third field, TRMOPT, is the first of the actual option offsets to the Option Table data, the beginning of which is pointed to by the TRMOPTBL field. Each option offset is a one-byte index to the corresponding Option Table data. There is an option offset for each possible option up to and including the last option specified for this terminal entry. If a particular option within that span is omitted, that option offset field is initialized to X'FF'.

The device dependent fields of an entry in the Terminal Table are used to indicate such information as the dial digits or addressing characters of the terminal. The specific type of information in these fields is noted in the two bytes of the device dependent field flags field (TRMDEVFL) of the Terminal Table. The actual entries in the device dependent fields consist of one byte, which contains the length of the entry, followed by the actual information. The location of the device dependent field is indicated by the bit settings in the first byte of the Terminal Table. If bit six (TRMOPTFN) in the status byte (TRMSTATE) is off, the device dependent field is located at +17 (X'11') in the table. If bit six is on, indicating that there are option offset fields in the table, the device dependent field starts at location 20 (X'14') plus the value in the number of option offsets field (TRMOPNO). Each option offset is one byte long, and the first option offset is located at offset 20 in a terminal entry; the device dependent field starts immediately after the last option offset.

The figures below show the formats of the various types of terminal entries; descriptions of the fields follow the illustrations.

IEDQTRM

<table>
<tr><td>0 (0)
TRMSTATE
Status Byte</td><td colspan="3">1 (1)
TRMDESTQ
Destination QCB Address</td></tr>
<tr><td colspan="4">4 (4)
TRMALNCT
Automatic Line Number Count</td></tr>
<tr><td colspan="2">TRMINSEQ
Input Sequence Number
TLSTCNT
TLIST Count of Entries</td><td colspan="2">6 (6)
TRMOUTSQ
Output Sequence Number
TLISTEN
First TLIST Entry Address</td></tr>
<tr><td colspan="2">8 (8)
TRMALTD
Alternate Destination Termname Table Offset</td><td colspan="2">10 (A)
TRMDEVFL
Device Dependent Field Flags</td></tr>
<tr><td colspan="2">12 (C)
TRMSTAT
Error Statistics
TRMSIO
Start I/O Count</td><td>14 (E)
TRMTEMPR
Temporary Error Count</td><td>15 (F)
TRMSENSE
Intensive Mode
Recording Indicator</td></tr>
<tr><td>16 (10)
TRMCHCIN
DCT Index</td><td>17 (11)
TRMOPNO
Option Field Count</td><td colspan="2">18 (12)
TRMOPTBL
Option Table Offset</td></tr>
<tr><td colspan="4">20 (14)
TRMOPT
Start of Option Offsets</td></tr>
</table>

Offset	Name	Bytes	Description
0 (0)	TRMSTATE	1	Status byte - the bit definitions are as follows:

Name	Bits	Value	Meaning
	0-2		Type of entry
		B'000'	Terminal, single, or group
		B'001'	Process
		B'010'	Cascade list
		B'100'	Line
		B'101'	Log
	3		Reserved
TRMACPTN	4	X'08'	Terminal can accept an entry for processing
TRMHELDN	5	X'04'	Terminal is held or a process entry specified SYNC=YES
TRMOPTEN	6	X'02'	Option fields are used
TRMSCNYN	7	X'01'	Control Terminal

Offset	Name	Bytes	Description
1 (1)	TRMDESTQ	3	Address of the Destination QCB for the entry or of the distribution or cascade entry QCB.
4 (4)	TRMINSEQ	2	Input sequence number
4 (4)	TLISTCNT	2	Count of entries in a TLIST
6 (6)	TRMOUTSQ	2	Output sequence number
6 (6)	TLISTEN	2	First entry in a TLIST
8 (8)	TRMALTD	2	Termname Table offset of the alternate destination
10 (A)	TRMDEVFL	2	Device dependent field flags to indicate which fields are present

Bits	Value	Meaning
0	X'8000'	BUFSIZE specified
1	X'4000'	Dial digits present
2	X'2000'	Addressing characters present
3	X'1000'	BLOCK specified
4	X'0800'	SUBBLOCK specified
5	X'0400'	Transparent block length specified
6	X'0200'	BFDELAY specified
7	X'0100'	Time Sharing field
8-15	-	Reserved

Offset	Name	Bytes	Description
12 (C)	TRMSTAT		Error statistics
12 (C)	TRMSIO	2	Number of START I/O instructions
14 (E)	TRMTEMPR	1	Number of temporary errors
15 (F)	TRMSENSE	1	Intensive mode recording indicator
16 (10)	TRMCHCIN	1	Index to the Device Characteristics Table for this entry
17 (11)	TRMOPNO	1	Number of option fields for this entry
18 (12)	TRMOPTBL		Offset to the option table for this entry
20 (14)	TRMOPT		Start of option offsets

TERMINAL TABLE ENTRY TYPE

Single and Line

Offset

0	1	4	6	8	10 (A)	12 (C)	14 (E)	15 (F)	16 (10)	17 (11)	18 (12)	20 (14)	20 + n
Status byte	Destination QCB address	Input sequence number	Output sequence number	Alternate destination offset	Device dependent field flags	Number start I/Os	Number temporary errors	Intensive mode recording indicator	DCT index	Number option offsets	Option Table offset	Start of option offsets	Start of device dependent fields
TRMSTATE	TRMDESTQ	TRMINSEQ	TRMOUTSQ	TRMALTD	TRMDEVFL	TRMSIO	TRMTEMPR	TRMSENSE	TRMCHCIN	TRMOPNO	TRMOPTBL	TRMOPT	

Group

Offset

0	1	4	6	8	10 (A)	12 (C)	14 (E)	15 (F)	16 (10)	17 (11)	18 (12)	20 (14)	20 + n
Status byte	Destination QCB address	Unused X'0000'	Output sequence number	Alternate destination offset	Device dependent field flags	Number start I/Os	Number temporary errors	Intensive mode recording indicator	DCT index	Number option offsets	Option Table offset	Start of option offsets	Start of device dependent fields
TRMSTATE	TRMDESTQ		TRMOUTSQ	TRMALTD	TRMDEVFL	TRMSIO	TRMTEMPR	TRMSENSE	TRMCHCIN	TRMOPNO	TRMOPTBL	TRMOPT	

Distribution

Offset

0	1	4	6
Status byte	Distribution List QCB address	Number entries in the list	Offset to the first entry in the list
TRMSTATE	TRMDESTQ	TLISTCNT	TLISTEN

Cascade

Offset

0	1	4	6
Status byte	Cascade list QCB address	Number entries in the list	Offset to the first entry in the list
TRMSTATE	TRMDESTQ	TLISTCNT	TLISTEN

Process

Offset

0	1	4	6	8	10 (A)	12 (C)	16 (10)	17 (11)	18 (12)	20 (14)
Status byte	Process QCB address	Input sequence number	Output sequence number	Alternate destination offset	Device dependent field flags	Process Entry Work Area address	Work unit record delimiter character	Number option offsets	Option Table offset	Start of option offsets
TRMSTATE	TRMDESTQ	TRMINSEQ	TRMOUTSQ	TRMALTD	TRMDEVFL	TRMSTAT	TRMCHCIN	TRMOPNO	TRMOPTBL	TRMOPT

Logtype

Offset

0	1	4	6	8	10 (A)	12 (C)	14 (E)	15 (F)	16 (10)	17 (11)	18 (12)
Status byte	Destination QCB address	Unused X'0000'	Unused X'0000'	Unused X'0000'	Device dependent field flags	Unused X'0000'	Unused X'00'	Unused X'00'	Unused X'00'	Unused X'00'	Buffer size (2 bytes)
TRMSTATE	TRMDESTQ				TRMDEVFL						

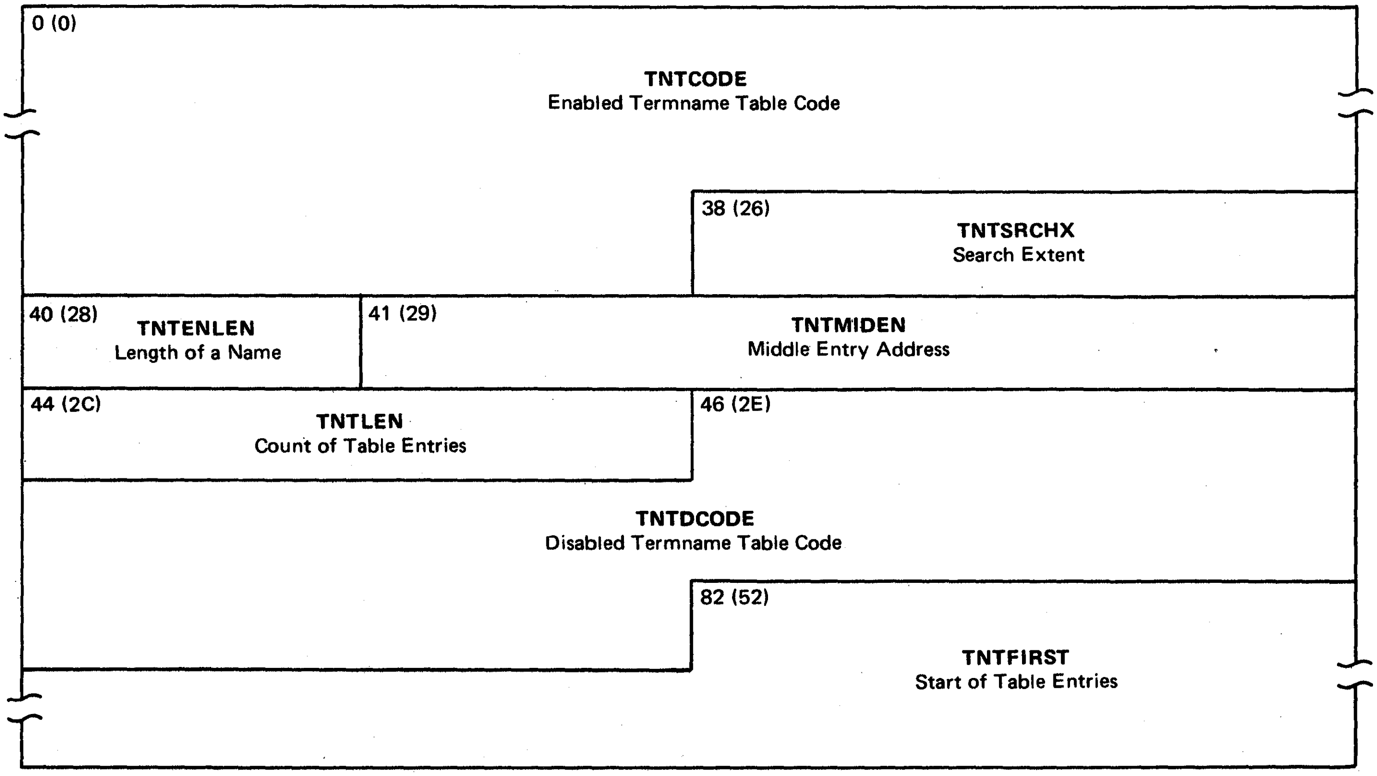
TERMNAME TABLE

The Termanme Table has an entry that contains the name and terminal entry address for each terminal, terminal component, application program, list of terminals, and logging media in the TCAM system. These entries are generated at assembly time from the TERMINAL macros in the order in which the macros are coded. At MCP initialization time the entries are sorted into collating sequence.

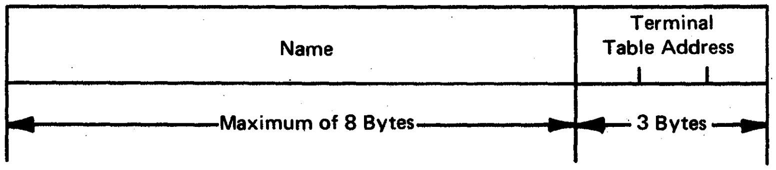
The first 82 bytes of the Termname Table contain the Termname Table code (IEDQTNT) and control information. The code can be executed as a subroutine by other TCAM modules. The control information identifies the attributes of the table.

The address of the Termname Table is in the AVTRNMPT field of the AVT. However, the individual Termname Table entries are referenced by the relative position offsets that precede the control data in each invitation list. When a TCAM module needs to find a specific entry in the Terminal Table, the module activates the Termname Table code, which translates the invitation list relative position offset to the address of the corresponding Terminal Table entry.

IEDQTNTD



Format of a Termname Table Entry:



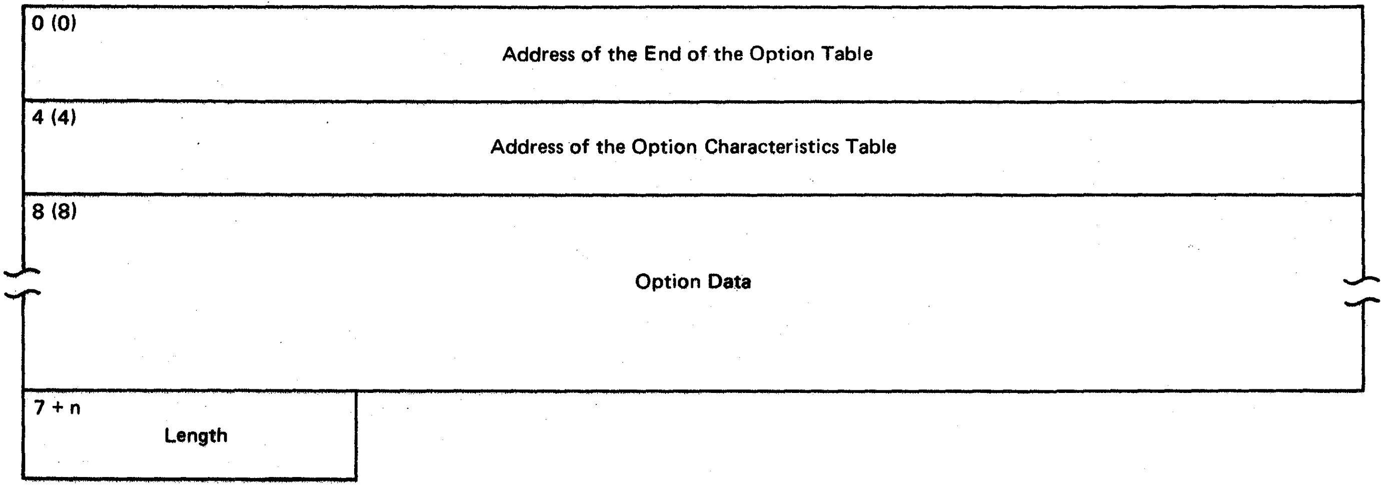
Offset	Name	Bytes	Description
0 (0)	TNTCODE	38	Enabled Termanme Table code (IEDQTNT) to convert the relative offset to a Terminal Table address
38 (26)	TNTSRCHX	2	Binary search extent - used by the Binary Search routine (IEDQA1)
40 (28)	TNTENLEN	1	Length in bytes of the name field of an entry
41 (29)	TNTMIDEN	3	Address of the middle entry in the Termname Table - used by the Binary Search routine (IEDQA1)
44 (2C)	TNTLEN	2	Total number of entries in the Termname Table
46 (2E)	TNTDCODE	36	Disabled Termname Table code
82 (52)	TNTFIRST		The beginning of the Term-name Table entries

OPTION TABLE

The Option Table (IEDQOPT) is a variable length table that contains the actual data coded by the user in the TERMINAL and TPROCESS macros in the message control program. At assembly time, this data is placed in the table with the the necessary byte alignment in the order in which it is coded. An option data field, which is not directly identifiable by the macro in which it is coded, can be referred to only through the option offset fields of a terminal entry. If only the user-coded name for a macro is known, TCAM uses the Option Characteristics Table and the terminal entry to refer to a specific data field in the Option Table. (See the discussion of the Option Characteristics Table in this section.)

At assembly time the address of the Option Table is placed in the AVTOPTPT field of the AVT. The first two words of the Option Table contain the address of the end of the Option Table and the address of the the Option Characteristics Table, respectively. The option data immediately follows these two words. The figure below is the general format of the Option Table; descriptions of the fields follow the illustration

IEDQOPT



0 (0)
Address of the End of the Option Table

4 (4)
Address of the Option Characteristics Table

8 (8)
Option Data

7 + n
Length

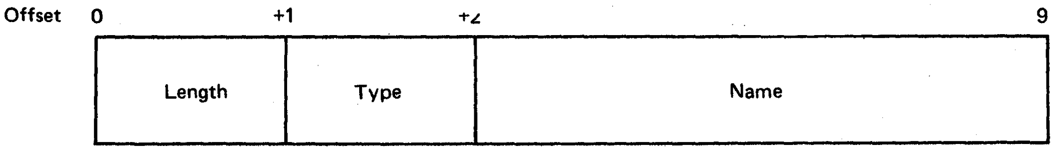
Offset	Name	Bytes	Description
0 (0)		4	The address of the first byte (7+n) following the Option Table
4 (4)		4	The address of the first byte of the Option Characteristics Table (IEDQCPTN)
8	Option data	n	The actual data coded by the user with the necessary byte alignment, in the order in which the data is coded
7+n	length	1	The length of the option data for the terminal or process entry that has the longest option data

OPTION CHARACTERISTICS TABLE

The Option Characteristics Table (IEDQOPTN) is a variable length table that contains one entry for each OPTION macro issued in the message control program (MCP). The relative position of an entry in the table directly corresponds to the relative position of an option offset in a Terminal Table entry. The option offset is an index to the actual Option Table data for the option entry in the Option Characteristics Table. The Option Characteristics Table allows TCAM routines to use the assembled name for an OPTION macro to locate the Option Table data for a specific station (terminal).

Each entry in the Option Characteristics Table contains the length of the corresponding Option Table entry, the type of option field specified, and the user-specified name of the OPTION macro. The length of the table is variable and consists of ten bytes for each OPTION macro issued plus one byte (X'FF') to indicate the end of the table. Storage is allocated and the table is initialized at assembly time. The AVT field AVTOPTPT contains the address of the Option Table, and the second word of the Option Table contains the address of the Option Characteristics Table.

The figure below is the format of an entry in the Option Characteristics Table; descriptions of the fields follow the illustration.



Offset	Name	Bytes	Description
0 (0)	length	1	The length of the corresponding Option Table entry, which is equal to the number of bytes of data specified by the TPROCESS and TERMINAL macros plus any necessary alignment bytes
1 (1)	type	1	The type of option field, indicated by one of the following bit configurations:

Hex	Code	Type of Constant	Machine Format
00	C	Character	8-bit code for each character
01	Z	Decimal	Zoned decimal format
40	P	Decimal	Packed decimal format

Offset	Name	Bytes		Description	
		81	D	Floating-Point	Long floating-point format; normally a doubleword
		80	F	Floating-Point	Short floating-point format; normally a fullword
		D0	Q	Address	Space reserved for a dummy section offset
		C8	V	Address	Space reserved for external symbol addresses; each address normally a fullword
		C4	S	Address	Base register and displacement value; a halfword
		C2	Y	Address	Value of address; normally a halfword
		C1	A	Address	Value of address; normally a fullword
		F0	F	Fixed-Point	Signed, fixed-point binary format; normally a halfword
		F6	H	Fixed-Point	Signed, fixed-point binary format; normally a halfword
		F4	X	Hexadecimal	4-bit code for each hexadecimal digit
		F2	B	Binary	Binary format
2 (2)	name	8		The name of the option field - this is the actual name the user codes in the name field of the OPTION macro	

DEVICE CHARACTERISTICS TABLE

The Device Characteristics Table (DCT) is a variable length table that contains one four-byte entry for each type of terminal or station defined in the TCAM system. At assembly time, each entry is allocated and initialized to describe the characteristics of the particular type of terminal or group of terminals; one entry is generated for all terminals that have identical characteristics.

The address of the Device Characteristics Table is assembled in the AVTCSTCS field of the Address Vector Table. The one-byte index (TRMCHCIN) in a terminal entry in the Terminal Table provides the offset to the specific Device Characteristics Table entry for a station.

Bits are set in the DCT entry to indicate the type of station. Combinations of these bit settings may be coded where applicable, The specific values for a DCT entry are outlined below.

Offset	Name	Value	Description
0 (0)			Reserved
1 (1)	CINHIBIT	X'80'	Terminal can use Read Inhibit CCWs
	CBREAK	X'40'	Terminal has the Reverse Break feature
	CATTEN	X'20'	Terminal has the Attention feature
2 (2)	CBISYNC	X'80'	BSC station

Offset	Name	Bytes	Description
	CTWX	X'20'	TWX 3335 terminal
	CSTNCTL	X'10'	Terminal has the Station Control feature
	CXMITCTL	X'08'	Terminal has the Transmit Control feature
	CCONTENT	X'04'	Contention device
	CLOCAL	X'02'	Local device
	CAUDIO	X'01'	Audio device
3 (3)	CWTTA	X'40'	World Trade Telegraph
	CENDCTL	X'20'	Terminal has end-to-end controls (2780)
	CCHECK	X'10'	Terminal has the Checking feature
	CCONTIN	X'04'	Terminal is capable of a Continue operation
	CNOIDLES	X'02'	Terminal has no idles defined
	C2760	X'01'	2760

SPECIAL CHARACTERS TABLE

A Special Characters Table (SCT) is a variable length table that consists of entries giving the special characters required for device I/O for a line group. There is one SCT for each type of line group in the TCAM system. Each SCT contains a list of the characters that the associated terminal or line group recognizes. SYS1.SVCLIB contains a Special Characters Table for each line group in the system. The various SCTs are initialized at SYSGEN time, and at open time the TCAM Line Group Open routine uses information from the UCB and the terminal entry to load the appropriate Special Characters Table.

An SCT is located by a three-byte address in the DCBSCTAD field of the DCB for the line group. The address of the DCB for the line group is in the LCBDCBPT field of the associated LCB.

An SCT is used to build channel programs. This table is also used by the error recovery procedures to retry certain text errors, and by the message handling routines to initiate on-line test procedures and to determine the message format for line control insertion.

The first 28 bytes of an SCT comprise a fixed-length directory of one-byte offsets, each of which when added to the SCT pointer in the DCB points to a one-byte length field. This length field is followed by a special characters entry of the length specified in the length field. There are as many entries in the directory as there are different sets of special characters required by the line group. If a function is not defined for the associated terminal or line group, the offset field in the directory contains a X'00' value.

The following is a list of the specific types of characters that each of the offsets in the first 28 bytes of a SCT represent.

Offset	Special Characters
0 (0)	EOT sequence
1 (1)	EOA sequence
2 (2)	PAD characters
3 (3)	Idle or reserve characters
4 (4)	Even ACK
5 (5)	Odd ACK
6 (6)	NAK
7 (7)	ENQ (inquiry)
8 (8)	EOB/ETB (for BSC DLE ETB)

Offset	Special Characters
9 (9)	DLE ETX (BSC)
10 (A)	DLE STX (BSC transparent sequence)
11 (B)	DLE/STX/ENQ (BSC transparent temporary text delay-TTD)
12 (C)	SOH (BSC - start of header character)
13 (D)	On-line Test sequence
14 (E)	WACK (BSC)
15 (F)	RVI (BSC reverse interrupt)
16 (10)	DLE EOT (BSC dial sequence)
17 (11)	DLE ENQ (BSC - use in abort sequence)
18 (12)	EOB sequence (used by IEDQAK and IEDQA6 to insert line control characters)
19 (13)	ITB sequence (used by IEDQAK and IEDQA6 to insert line control characters)
20 (14)	EOT sequence (used by IEDQAK and IEDQA6 to insert line control characters)
21 (15)	EOT sequence (used by IGG019R0)
22 (16)	EOB sequence (used by IGG019R0)
23 (17)	ETX sequence (BSC only)
24 (18)	ENQ sequence (BSC only)
25 (19)	SOH % S sequence (BSC only)
26 (1A)	SOH % 00 sequence (BSC on-line test sequence)
27 (1B)	SOH %/CANCEL/ sequence (BSC on-line test cancel sequence)

RESOURCE CONTROL BLOCK

The Resource Control Block (IEDQRECB) is a two-word prefix to an element that allows the TCAM Dispatcher to determine the disposition of an element and to determine the QCB to which an element will be tposted. Each element in the TCAM system is represented by a Resource Control Block. The first word of the RCB is a pointer to the QCB with which the element is associated; the second word is a link field which, when the element is on a chain, points to the next item on the chain. The first word in the associated QCB may point to the RCB.

Storage is allocated for the RCB at open time for the line group or for the application program. The RCB is initialized at open time and is modified when elements are passed in the system.

Below is the format of a Resource Control Block; descriptions of the fields follow the illustration.

IEDQRECB

0 (0) **RECBKEY** Key Field	1 (1) **RECBQCBA** QCB Address
4 (4) **RECBPRI** Priority	5 (5) **RECBLINK** Link Field

Offset	Name	Bytes	Description
0 (0)	RECBKEY	1	Key field
1 (1)	RECBQCBA	3	Address of the QCB to which this RCB is tposted
4 (4)	RECBPRI	1	Priority of this RCB
5 (5)	RECBLINK	3	Address of the next RCB in the chain in which this RCB is currently located

QUEUE CONTROL BLOCK

A Queue Control Block (QCB) is used to regulate the sequential use of elements among requesting tasks. Every queue, or item, that is waiting for service in the system is associated with a QCB. There is a master Destination QCB for every destination message queue. There is another type of Queue Control Block, called a priority QCB, for each priority level applicable for each Destination QCB. The first priority QCB begins at a displacement of 40 (X'28') from the beginning of the Destination QCB.

Note: There is no priority QCB for a TSO dedicated line. The QCB is truncated at the displacement 40 (X'28').

A QCB has three primary fields: a pointer to the element chain, a link address, and a pointer to the STCB chain. The element chain consists of any elements, other than the requesting resource on the ready queue, that the subtask represented by the STCB chain might need to process. The link field is used to point to another item when a QCB is on a higher queue. The STCB chain consists of pointers to the routines that are associated with the QCB.

The address of the Destination QCB is in the TRMDESTQ field of the Terminal Table entry which is, in turn, pointed to by the Termname Table entry. The address of the Termname Table is in the AVTRNMPT field of the Address Vector Table. The LCBSCBDA field of the Line Control Block points to the Station Control Block. Within an SCB is a pointer (SCBDESTQ) to the Queue Control Block.

Storage is allocated for the QCB at assembly time. The QCB is initialized partially at assembly time and partially at open time.

The figures below are the formats of the master Destination Queue Control Block and the priority QCB; descriptions of the fields follow the illustrations.

Master Queue Control Block DSECT: IEDQQCB

0 (0)	QCBDSFLG	Flag Byte
1 (1)	QCBELCHN	Element Chain
4 (4)	QCBPRI	Priority
5 (5)	QCBLINK	Pointer to the Next STCB in a Chain
8 (8)	QCBSTVTO	Index to the Entry in the Subtask Vector Table
9 (9)	QCBSTCHN	STCB Chain
12 (C)	QCBSTPRI	Priority of the STCB
13 (D)	QCBSLINK	Pointer to the Next STCB in a Chain
16 (10)	QCBEOLDT	Interrupt Time
18 (12)	QCBRETCT	TSO Retry Counters
	QCBLKRLN	Lock Relative Line Number
19 (13)	QCBSTAT	Status of this QCB
20 (14)	QCBSCBOF	Offset to the Proper SCB
21 (15)	QCBINSRC	Chain of Source LCBs Currently Sending Initiate Mode Msgs
	QCBSATCT	Sim ATTN Output Line Count
22 (16)	QCBTSOF2	Second TSO Flag Byte
23 (17)	QCBTSOF1	First TSO Flag Byte
24 (18)	QCBINTVL	Interval for Poll Delay
26 (1A)	QCBMSGCT	Count of Messages in this Queue
28 (1C)	QCBPREN	Address of Terminal Table Entry if QCB for a Process Entry
	QCBPRLVL	Highest Priority Level Message
29 (1D)	QCBLKRRN	Lock Relative Line Number
	QCBCARCT	Carriage Position Count
30 (1E)	QCBTJID	TSO Job Identification
32 (20)	QCBRELLN	Relative Line Number
33 (21)	QCBDCBAD	Address of the DCB
36 (24)	QCBFLAG	QCB Status Bits
37 (25)	QCBQBACK	QBACK Message Chain

Priority Queue Control Block DSECT: IEDPQCB

40 (28) **QCBDNHDR**
Disk Record Number to Put the Next Header Received

43 (2B) **QCBFHDLZ**
Disk Rec. No. of First Header
on Queue Placed on Last Zone

Continued

46 (2E) **QCBFHDTZ**
Disk Rscord Number of First Header
Placed in the Current Zone

Continued

49 (31) **QCBINTFF**
Disk Record Number of the First Intercepted Msg – FEFO Order

52 (34) **QCBINTLF**
Disk Record Number of Last Intercepted Message – FEFO Order

55 (37) **QCBFFEFO**
Disk Rcd. No. of First FEFO
Message or Core Rcd. No.

Continued

58 (3A) **QCBLFEFO**
Disk Rcd. No. of Last FEFO Msg
Core Rcd. No. if Core – Only Queue

Continued

61 (3D) **QCBCFHDR**
Core Record No. of First Header Appearing in this Queue

64 (40) **QCBPRIPQ**
Priority of this Priority
Level QCB

65 (41) **QCBCPVHD**
Core Address of Last Address Placed on this Queue

The following is for the master QCB:

Offset	Name	Bytes	Description
0 (0)	QCBDSFLG		Flags that indicate a specific Destination QCB to the Dispatcher and which message queues data set is to receive the messages for the destination - bit definitions are as follows:

Name	Bits	Value	Meaning
QCBFQCB	6	X'02'	Indicates a QCB
QCBREUS	3	X'10'	Indicates reusable disk queuing
QCBNREUS	2	X'20'	Indicates nonreusable disk queuing
QCBDISK	2,3	X'30'	Disk queues are used
QCBCORE	1	X'40'	Flag for main storage queues:
	1,3	X'50'	Indicates main storage queues with backup on reusable disk
	1,2	X'60'	Indicates main storage queues with backup on nonreusable disk

Offset	Name	Bytes	Description
1 (1)	QCBELCHN	3	Element chain pointer - contains the address of the QCB to be tposted when this QCB is removed from the time delay queue.
4 (4)	QCBPRI	1	Priority
5 (5)	QCBLINK	3	Pointer to the next STCB in a chain
8 (8)	QCBSTVTO	1	Index to an entry in the Subtask Vector Table
9 (9)	QCBSTCHN	3	STCB chain pointer
12 (C)	QCBSTPRI	1	Priority of the STCB
13 (D)	QCBSLINK	3	Pointer to the next STCB in a chain
16 (10)	QCBEOLDT	2	Interrupt time
18 (12)	QCBRETCT	1	TSO retry counters
18 (12)	QCBLKRLN	1	Lock relative line number
19 (13)	QCBSTAT	1	Status of this QCB - bit settings are as follows:

Name	Bits	Value	Meaning
QCBEOM	0	X'80'	End of message sent
QCBTRMHO	1	X'40'	Terminal was held
QCBBUFRD	2	X'20'	Buffered terminal
QCBSEND	3	X'10'	Sending to a buffered terminal
QCBRECEV	4	X'08'	Receiving from a buffered terminal
QCBSCHDL	5	X'04'	Put in the time delay queue when inactive
QCBCLOCK	6	X'02'	On=Clock, off=interval
QCBTIME	7	X'01'	Delay greater than 12 hours

Offset	Name	Bytes	Description
20 (14)	QCBSCBOF	1	Offset to the proper SCB for this transmission; X'00' unless this line has buffered terminals
21 (15)	QCBINSRC	3	Chain of source LCBs currently sending initiate mode messages to this Destination queue
21 (15)	QCBSATCT	1	Simulated attention output line count - TSO
22 (16)	QCBTSOF2	1	Second TSO flag byte - bit settings are as follows:

Offset	Name	Bytes	Description

Name	Bits	Value	Meaning
QCBINHBN	0	X'80'	Use inhibits with this terminal
QCBBUFQ	1	X'40'	TCAM buffer being held
QCBPOSTO	2	X'20'	QCB posted to itself
QCBDSSMI	3	X'10'	Start MI character sent - TSO
QCBSATCH	5	X'04'	Simulated attention by character
QCBSATTI	6	X'02'	Simulated attention by time
QCBSATLC	7	X'01'	Simulated attention by line

Offset	Name	Bytes	Description
23 (17)	QCBTSOF1	1	First TSO flag byte - bit settings are as follows:

Name	Bits	Value	Meaning
QCBWRBRK	0	X'80'	Issue a write break
QCBTGET	1	X'40'	TGET request
QCBTPUT	2	X'20'	TPUT request
QCBNOBUF	3	X'10'	Insufficient buffers
QCBSATRD	4	X'08'	Simulated attention read request
QCBPARTO	5	X'04'	Partial output line
QCBDELAY	6	X'02'	QCB in time delay queue
QCBDISC	7	X'01'	User to be logged off

Offset	Name	Bytes	Description
24 (18)	QCBINTVL	2	Interval for poll delay
26 (1A)	QCBMSGCT	2	Count of messages in this queue
28 (1C)	QCBPREN	4	Address of the Terminal Table entry if this is a QCB for a process entry
28 (1C)	QCBPRLVL	1	Highest priority level message
29 (1D)	QCBLKRRN	3	Lock relative line number; link field for the QCB when its on the time delay queue
29 (1D)	QCBCARCT	1	Carriage position count
30 (1E)	QCBTJID	2	TSO job identification
32 (20)	QCBRELLN	1	Relative line number for the line this QCB represents
33 (21)	QCBDCBAD	3	Address of the DCB
34 (22)	QCBFLAG	1	QCB status bits - bit settings are as follows:

Name	Bits	Value	Meaning
QCBTSSES	0	X'80'	TSO session in progress
QCBNOBRK	1	X'40'	No reverse break feature
QCBREAD	2	X'20'	Read has priority
QCBRSRV	3	X'10'	Reusability serviced
QCBTERMQ	4	X'08'	Queue by terminal
QCBSDFFO	5	X'04'	Currently sending a FEFO message
QCBPROC	6	X'02'	This QCB is for a process entry
QCBCKPT	7	X'01'	Flag for checkpoint

Offset	Name	Bytes	Description
37 (25)	QCBQBACK	3	Queue-back message chain

Offset	Name	Bytes	Description
The following is for a priority QCB:			
40 (28)	QCBDNHDR	3	Disk record number of the first unit of the first buffer of the next message
43 (2B)	QCBFHDLZ	3	Disk record number of the first buffer of the first message that was placed in the last disk zone for this queue
46 (2E)	QCBFHDTZ	3	Disk record number of the first buffer of the first message that was placed in the current disk zone for this queue
49 (31)	QCBINTFF	3	Disk record number of the first held message in FEFO order
52 (34)	QCBINTLF	3	Disk record number of the last held message in FEFO order
55 (37)	QCBFFEFO	3	Disk record number of the first message to be received. Main storage record address if this is a main-storage-only queue
58 (3A)	QCBLFEFO	3	Disk record number of the last FEFO message received. Main storage record address if this is a main-storage-only queue
61 (3D)	QCBCFHDR	3	Main storage record address of the first buffer of the first message appearing in this queue
64 (40)	QCBPRIPQ	1	The priority of this priority level QCB. This is X'00' if this is the lowest priority level.
65 (41)	QCBCPVHD	3	Main storage record address of the last address placed on this queue

SUBTASK CONTROL BLOCK

The Subtask Control Block (IEDQSTCB) is a variable length table that represents the routine that perform the work of the TCAM system. The purpose of an STCB is to cause a routine to be executed. The TCAM Dispatcher uses the STCB to determine the entry point of a subtask that is waiting for work. The address of the STCB is in the third word of the QCB.

Storage is allocated for the STCB at various times depending upon the type of QCB containing the STCB address. If the QCB is a Destination QCB, storage is allocated for the STCB at assembly time. If the QCB is in a Line Control Block or is a Read-ahead QCB, storage is allocated for the STCB at open time for the line group or for the application program DCB. If the QCB is in the AVT, storage is allocated at assembly time. In cases where the QCB is a prefix to a module, storage is allocated for the STCB at assembly time.

In the same manner, initialization of the STCB depends upon the related QCB. If the QCB is a Destination QCB, the STCB is initialized at assembly time but is modified at Open time for the DCB to which it is related. If the QCB is in the LCB or is a Read-ahead QCB, the STCB is initialized at open time. If the QCB is in the AVT, the STCB is initialized at assembly time and at link edit time. If the QCB is a prefix to a module, the STCB is initialized at assembly time.

Below is the format of a Full (eight-byte) STCB; descriptions of the fields follow the illustration. For formats of other types of STCBs see the discussion of the Functions of the TCAM Dispatcher.

IEDQSTCB

0 (0) **STCBVTO** Activation Key	1 (1) Reserved
4 (4) **STCBPRI** Priority	5 (5) **STCBLINK** Link Field

Offset	Name	Bytes	Description
0 (0)	STCBVTO	1	Activation key
1 (1)		3	Reserved
4 (4)	STCBPRI	1	Priority
5 (5)	STCBLINK	3	Link address - address of the next STCB in this STCB chain

ELEMENT REQUEST BLOCK

The Element Request Block (ERB) is a fixed-length table of fourteen bytes located at a displacement of X'4C' from the beginning of the Line Control Block. The function of the ERB is to request buffers for transmissions of data. The beginning of the Element Request Block is at a displacement of +44 (X'2C') from the beginning of the Input/Output Block within the LCB. The address of the IOB is in the DCBIOBAD field of the Data Control Block.

Storage is allocated for the Element Request Block at open time. The ERB is initialized at various times depending upon its function. When it is being used to request buffers, the ERB may be initialized by the Send Scheduler, Receive Scheduler, Get Scheduler, and/or the Put Scheduler. When it is being used to get recalled buffers, the ERB may be initialized by Buffer Disposition, EOB Check, and/or the Buffer Terminal Scheduler.

The format of the Element Request Block and descriptions of the fields are included in the discussion of the Line Control Block.

LINE CONTROL BLOCK

The Line Control Block (IEDQLCB) is a fixed length table that contains the information that must be maintained on a line or line group basis. There is one Line Control Block for each line group. The LCB maintains such information as pointers to the channel program, the corresponding DCB, the address of the QCB to which recalled buffers are to be posted, the last serviced PCI, and the chain of waiting QCBs. The LCB also contains the buffer chain, the subtask chain, and the I/O status. When the LCB is functioning as a QCB, the Line Control Block contains the address of the first STCB. Within the LCB, at a displacement of 68 (X'44'), is the Element Request Block. (For further information on the ERB see the discussion of the Element Request Block). The I/O Block is also in the LCB at a displacement of X'20' from the beginning.

To find the address of a specific LCB for a line group from a DCB, the TCAM modules first multiply the relative line number for this line times the value in DCBEIOBX and add the result to the value in DCBIOBAD. The result is the address of the IOB for this LCB. The LCB begins at -X'20' from the IOB address.

Storage is allocated and the Line Control Block is initialized at open time for the DCB for the line group.

The figure below is the format of the Line Control Block; descriptions of the fields follow the illustration.

IEDQLCB

0 (0) **LCBKEY** Element Key of Buffer **LCBRCB** Resource Control Block	1 (1) **LCBQCBA** Address of the QCB		
4 (4) **LCBPRI** Priority of Buffer	5 (5) **LCBLINK** Link Field of Buffer		
8 (8) **LCBRSKEY** Receive Scheduler Key	9 (9) **LCBSTCBA** Address of the First STCB When LCB is a QCB		
12 (C) **LCBRSPRI** Receive Scheduler Priority	13 (D) **LCBRSLNK** Address of the Next Item in the Chain		
16 (10) **LCBEOLTD** End of List Time Delay		18 (12) **LCBTDL** Time Delay Queue Offset	19 (13) **LCBTSOB** TSO Status Bits
20 (14) **LCBCHAIN** Disposition Status Bits	21 (15) **LCBINSRC** In-source Chain		
24 (18) **LCBNTXT** Save Area for PRFNTXT	25 (19) **LCBSCBDA** Address of the SCB Directory **LCBLNENT** TNT Offset to Line Entry		
28 (1C) **LCBISZE** Count of Idles Reserved	29 (1D) **LCBFSBFR** First Buffer Assigned to this LCB **LCBLSBFR** Last Buffer Assigned to this LCB		
32 (20) **LCBFLAG1** IOS Flags 1	33 (21) **LCBFLAG2** IOS Flags 2	34 (22) **LCBSENS0** Sense Byte 0	35 (23) **LCBSENS1** Sense Byte 1
36 (24) **LCBECBCC** Completion Code	37 (25) **LCBECBPT** Address of the ECB		
40 (28) **LCBFLAG3** IOS Flags 3	41 (29) **LCBCSW** Last Channel Status Word		
48 (30) **LCBSIOCC** SIO Condition Code	49 (31) **LCBSTART** Address of the Channel Program		
52 (34) **LCBDCBPT** Address of the DCB			
56 (38) **LCBRESTR** Error Message Data **LCBRCQCB** QCB to Which to Post the Recalled Buffer			
60 (3C) **LCBINCAM** IOS		**LCBTTBIN** Index to Terminal to be Connected **LCBERRCT** IOS Error Counters	

64 (40) **LCBUCBX** UCB Index	65 (41) **LCBRCBFR** Pointer to the Recalled Buffer **LCBLSPCI** Address of the Last Serviced PCI
68 (44) **LCBRECOF** Offset to the Current Block	70 (46) **LCBSTATE** Status Bits **LCBSTAT1** First Status Byte 71 (47) **LCBSTAT2** Second Status Byte
72 (48) **LCBTSTSW** Test-and-Set Switch	73 (49) **LCBRECAD** Address of the Current Message Block
76 (4C) **LCBERBKY** ERB Key **LCBERB** Element Request Block	77 (4D) **LCBERBQB** ERB QCB
80 (50) **LCBERBPY** ERB Priority	81 (51) **LCBERBLK** Address of the Next Item in the Chain
84 (54) **LCBERBST** Status of ERB	85 (55) **LCBERBCH** Address of the Chain to be Assigned Buffers
88 (58) **LCBERBCT** Count Fields	90 (5A) **LCBTTCIN** Index to the Terminal Currently Connected
92 (5C) **LCBMSGFM** Bits to Control BSC Line	93 (5D) **LCBSCBA** Address of the Current SCB
96 (60) **LCBERMSK** Error Recording Mask	97 (61) **LCBINVPT** Address of the Current Entry in the Invitation List
100 (64) **LCBTPCD** TP OP Codes	
112 (70) **LCBSNSV** Save Area for Sense Byte	113 (71) **LCBCSWSV** Save Area for Channel Status Word
120 (78) **LCBERCCW** 3 ERP Commands	141 (8D) **LCBSTICS** Characteristics Work Area
144 (90) **LCBSTICS** (Cont.) **LCBCPA** Channel Program Area	

Offset	Name	Bytes	Description
0 (0)	LCBRCB	8	Resource control block for this LCB
0 (0)	LCBKEY	1	Key field of the RCB
1 (1)	LCBQCBA	3	QCB address
4 (4)	LCBPRI	1	Priority of the RCB
5 (5)	LCBLINK	3	Address of the next element in the chain in which this RCB is currently located
8 (8)	LCBRSKEY	1	Receive Scheduler key field
9 (9)	LCBSTCBA	3	Address of the first STCB when the LCB is functioning as a QCB
12 (C)	LCBRSPRI	1	Receive Scheduler priority field
13 (D)	LCBRSLNK	3	Address of the next item in the chain in which this STCB currently resides
16 (10)	LCBEOLTD	2	End of the invitation list time delay interval
18 (12)	LCBTDL	1	Time delay queue offset to QCB address for LCB = X'14'
19 (13)	LCBTSOB	1	TSO status byte:

Name	Bits	Value	Meaning
LCBPREP	0	X'80'	Prepare on line
LCBWRBRK	0	X'80'	Write break in progress
LCBTSBUF	1	X'40'	Buffer has TSO prefix
LCBSATRD	2	X'20'	Simulated attention request
LCBSOPL	3	X'10'	Start of polling list
LCBREAD	4	X'08'	Reading partial line
LCBCIRCD	5	X'04'	Circle D sent to 2741
LCBINHBN	6	X'02'	Use inhibits for this terminal
LCB2741N	7	X'01'	2741 on 2741/1050 line

Offset	Name	Bytes	Description
20 (14)	LCBCHAIN	1	Disposition status byte:

Name	Bits	Value	Meaning
LCBSCRNN	0	X'80'	Screen change requested
LCBSCRNF	0 Off	X'7F'	Mask to specify no screen change requested
LCBEXCP	1	X'40'	Delay EXCP until association
LCBERMSG	2	X'20'	ERP message waiting
LCBNORTY	3	X'10'	Text retry not possible
LCBUREQN	4	X'08'	Unit request in progress
LCBUREQF	4 Off	X'F7'	Mask to specify that a unit request is not in progress
LCBBFRSZ	5	X'04'	Queue management flag
LCBTETEN	6	X'02'	User requested tete-a-tete
LCBTETEF	6 Off	X'FD'	Mask to specify that tete-a-tete is not requested
LCBABRTN	7	X'01'	Abort sequence must be sent
LCBABRTF	7 Off	X'FE'	Mask to specify that an abort sequence is not required

Offset	Name	Bytes	Description
21 (15)	LCBINSRC	3	In-source chain
24 (18)	LCBNTXT	1	Temporary save area for PRENTXT
25 (19)	LCBSCBDA	3	Address of the SCB directory
26 (1A)	LCBLNENT	1	Termname Table offset to the line entry

Offset	Name	Bytes	Description
28 (1C)	LCBISZE	1	Count of idles (reserve characters) reserved
29 (1D)	LCBFSBFR	3	First buffer assigned to this LCB
29 (1D)	LCBLSBFR	3	Last buffer assigned to this LCB
32 (20)	LCBFLAG1	1	IOS flags 1
33 (21)	LCBFLAG2	1	IOS flags 2
34 (22)	LCBSENS0	1	Sense byte 0
35 (23)	LCBSENS1	1	Sense byte 1
36 (24)	LCBECBCC	1	Completion code
37 (25)	LCBECBPT	3	ECB address
40 (28)	LCBFLAG3	1	IOS flags 3
41 (29)	LCBCSW	7	Last CSW
48 (30)	LCBSIOCC	1	SIO condition code
49 (31)	LCBSTART	3	Address of the channel program
52 (34)	LCBDCBPT	4	Address of the corresponding DCB
56 (38)	LCBRESTR		Start of error message data
56 (38)	LCBRCQCB	4	Address of the QCB to which recalled buffers are to be toposted
60 (3C)	LCBINCAM	2	IOS
62 (3E)	LCBTTBIN	2	Index of the terminal to be connected
62 (3E)	LCBERRCT	2	IOS error counters
64 (40)	LCBUCBX	1	UCB index
65 (41)	LCBRCBFR	3	Pointer to a recalled buffer
65 (41)	LCBLSPCI	3	Address of the last serviced PCI
68 (44)	LCBTRST	2	Offset to the start of the Buffer Translation routine
70 (46)	LCBSTATE	2	Status bits
70 (46)	LCBSTAT1	1	First status byte - bit definitions are as follows:

Name	Bits	Value	Meaning
LCBRCLLN	0	X'80'	Recall being performed
LCBRCLLF	0 Off	X'7F'	Mask to specify that no recall is being performed
LCBCTLMD	1	X'40'	Line is in control mode
LCBCVRSP	1	X'40'	First BSC output conversational block
LCBOCNI	2	X'20'	Non-immediate operator control operation is in progress
LCBINITN	3	X'10'	Receiving initiate mode message
LCBINITF	3 Off	X'EF'	Mask to specify no initiate mode message
LCBCONT	4	X'08'	Continue or reset operation in progress
LCBFREEN	5	X'04'	Line is free

LCBFREEF	5 Off	X'FB'	Mask to specify that the line is not free
LCBRECBN	6	X'02'	Line is receiving
LCBSENDN	7	X'01'	Line is sending (Line is stopped if bits 5,6, & 7 are off.)

71 (47)	LCBSTAT2		Second status byte - bit settings are as follows:

Name	Bits	Value	Meaning
LCBTRACE	0	X'80'	I/O trace active for this line
LCBLOCK	0	X'80'	Line is in lock mode
LCBTRCOF	8 Off	X'7F'	Mask to specify that I/O trace is not active for this line
LCBMSGNN	1	X'40'	MSGGEN or Startup message
LCBMSGNF	1 Off	X'BF'	Mask to specify that this is not a MSGGEN or Startup message

Offset	Name	Bytes	Description		
		LCBBFOTN	2	X'20'	EOT from a buffered terminal (no EOM)
		LCBBEOTF	2 Off	X'DF'	Mask to specify a regular EOM if EOT from a buffered terminal
		LCBSNDPR	3	X'10'	Send priority switch set by the Send Scheduler
		LCBNEGRP	4	X'08'	Negative response to polling
		LCBSYNC	5	X'04'	Line is binary synchronous
		LCBDIAL	6	X'02'	This is a dial LCB
		LCBRESP	7	X'01	A response needs to be sent to this line
72 (48)	LCBTSTSW	1	Test-and-set switch:		

Name	Bits	Value	Meaning
LCBCONCT	0	X'80'	Connection established

73 (49)	LCBRECAD	3	Address of the current message block
76 (4C)	LCBERB	4	Start of the ERB for this LCB
76 (4C)	LCBERBKY	1	Element Request Block key field
77 (4D)	LCBERBQB	3	Address of the QCB to which this ERB is currently tposted
80 (50)	LCBERBPY	1	ERB priority
81 (51)	LCBERBLK	3	Address of the next item in the chain in which this ERB currently resides
84 (54)	LCBERBST	1	ERB status - bit settings are as follows:

Name	Bits	Value	Meaning
LCBMSG	0	X'80'	End of initiate mode
LCBEOMSG	1	X'40'	End of message read from disk
LCBRDERR	2	X'20'	Logical read error
LCBRDERF	2 Off	X'DF'	Mask to specify no read error
LCBINQ	3	X'10'	ERB is waiting for buffers from IEDQHM
LCBERROR	5	X'04'	Error on the send side
LCBPRCPG	6	X'02'	After the initial request is satisfied, tpost the ERB to the QCB specified in LCBRCQCB
LCBCOMPL	6	X'02'	Disk request is complete

LCBDLNKN	7	X'01'	Delink switch - ERB is not tposted, but is eligible to be tposted
LCBDLNKF	7 Off	X'FE'	Mask to specify that the ERB is tposted, sc PCI cannot tpost

Offset	Name	Bytes	Description
			it again
85 ·(55)	LCBERBCH	3	Address of the chain to be assigned buffers
88 (58)	LCBERBCT	2	Count fields
90 (5A)	LCBTTCIN	2	Index to the terminal that is currently connected
92 (5C)	LCBMSGFM	1	Bits to control the BSC line

Name	Bits	Value	Meaning
LCBNAK	0	X'80'	Request to send a NAK response
LCBACK1	1	X'40'	ACK counter -

The following two bits indicate whether a scan of line control has been accomplished and the type of line control received.

LCBVSTRT	2	X'20'	Valid start sequence
LCBRSTRT	3	X'10'	Error start sequence
LCBTTD	4	X'08'	Temporary time delay recieved
LCBENQ	5	X'04'	ENQ received
LCBEOT	6	X'02'	EOT first character
LCBOLT	7	X'01'	Address of the current SCB

93 (5D)	LCBSCBA	3	Address of the current SCB
96 (60)	LCBERMSK	1	Error recording mask
97 (61)	LCBINVPT	3	Address of the current entry in the invitation list
100 (64)	LCBTPCD	12	TP operation codes
112 (70)	LCBSNSV	1	Save area for the sense byte
113 (71)	LCBCSWSV	7	Save area for the CSW
120 (78)	LCBERCCW	24	Three ERP commands
141 (8D)	LCBSTICS	3	Characteristics work area
144 (90)	LCBCPA	8	Channel program area

STATION CONTROL BLOCK

There is at least one Station Control Block (SCB) associated with each LCB in the TCAM system. With buffered terminals there is one SCB per terminal on a line. A buffered terminal sends a block or a part of an entire transmission at a time; while that terminal is preparing to send a subsequent block, TCAM examines and receives from other terminals on the same line. TCAM uses the SCBs to keep track of one transmission from each buffered terminal on the line.

If the terminals on a line are not buffered, or if the line with which the SCB is associated is a dial line, one terminal at a time completes its transmission. There is no need to keep track of many transmissions in parallel, so one SCB is sufficient for the entire line. In this case the address of the SCB is the LCBSCBA field of the LCB.

The address of the SCB Directory is in the LCBSCBDA field of the Line Control Block, The offset to the current SCB is in the LCBSCBO field of the LCB.

To obtain the address of any SCB associated with a QCB, TCAM first locates the LCB. This is done by multiplying the relative line number (in QCBRELLN) by the size of an LCB (DCBEIOBX) and adding the address of the pseudo IOB (DCBIOBAD). This gives TCAM the address of the IOB. At a displacement of -X'20' from the beginning of the IOB is the beginning of the LCB. TCAM then multiplies the SCB size (located in the AVTSCBSZ field of the Address Vector Table) by the offset in QCBSCBOF and adds that total to the address of the SCB Directory (LCBSCBDA). This sum then points to the desired Station Control Block.

Storage is allocated for a Station Control Block at assembly time for leased lines and at open time for dial lines. The SCB is initialized by STARTMH.

The figure below is the format of the Station Control Block; descriptions of the fields follow the illustration.

IEDQSCB

Offset	Field	Description
0 (0)	SCBSTATE	Status Bits
1 (1)	SCBDESTQ	Pointer to the Destination QCB
4 (4)	SCBSNDCT	Message Limit On Send Side
	SCBRCVCT	Message Limit On Receive Side
5 (5)	SCBMACR	First/Next IN/OUTMSG Macro to be Executed
	SCBMBHEN	Address of the Multiple Header Buffer Entry
8 (8)	SCBPRI	Priority Index to the QCB
9 (9)	SCBBKFCT	Count of Message Length for Break
10 (A)	SCBEOBSZ	Size of Logical Blocks
12 (C)	SCBSALEV	Simulated Attention Level Req
	SCBQTYPE	
13 (D)	SCBMRFPL	Address of Forward Parameter List
16 (10)	SCBERRST	Error Word Bits
	SCBERR1	First Byte
17 (11)	SCBERR2	Second Byte
18 (12)	SCBERR3	Third Byte
19 (13)	SCBERR4	Fourth Byte
20 (14)	SCBMRFSD	Multiple Router First Secondary Destination
22 (16)	SCBEOBAC	Accumulated Count Between Blocks
	SCBDLPTR	Distribution List Pointer
24 (18)	SCBBSCFM	MSGFORM Dynamic Block Changes
25 (19)	SCBMBSSA	Multiple Buffer Scan Save Area
32 (20)	SCBCPBNO	Number of Next Sequential CPB
33 (21)	SCBDCHDR	Disk Address Current Header
36 (24)	SCBDESTL	Length of Destination Names
37 (25)	SCBCCHDR	Main Storage Address of the Current Header
40 (28)	SCBITBSZ	Size of Logical Subblocks
41 (29)	SCBSCSEG	Current Segment Being Read
	SCBDNSEG	Disk Address of the Next Segment to Send from the Disk
44 (2C)	SCBHBFNO	Number of Buffers in Multiple Header
45 (2D)	SCBSCHDR	Current Header Being Sent
	SCBCLSEG	Main Storage Address of the Last Message Segment
48 (30)	SCBITBAC	Accumulated Count Between ITBs
49 (31)	SCBFEFO	Saved FEFO Pointer
	SCBDCSEG	Disk Address of the Current Segment
52 (34)	SCBDEOB	Disk Information On the Last EOB
56 (38)	SCBSRCE	Message Buffer Source Saved
58 (3A)	SCBSIZE	Message Buffer Size Saved
60 (3C)	SCBSTAT1	Status Byte
61 (3D)	SCBXTRA	Address of Additional Records Saved
	SCBCORE	Address of the Record in the Core Queue Saved
64 (40)	SCBSEQ	Sequence Out Number
	SCBSCAN	Scan Pointer
66 (42)	SCBTQBCK	Text Segment Chain Saved
	SCBNTXT	Address of the Next Text Segment Saved
	Continued	
	Continued	
69 (45)	SCBCRCD	Address of the Current Segment Saved

72 (48) **SCBNHDR** Address of the Next Header Segment Saved	75 (4B) **SCBNXCPB** Next CPB Number from Disk
SCBCHDR Address of the Current Header Segment Saved	**SCBLCSEG** Main Storage Address of Current Segment
Continued	78 (4E) **SCBEOB** Pointer to First EOB Saved
80 (50) **SCBUNTCT** Count in Disk Record of First Byte of Data	81 (51) **SCBTRANS** Current Translate Table Address
84 (54) **SCBRGSAV** Save Area for User MH Registers — if Specified on INTRO	

Offset		Name	Bytes	Description
0	(0)	SCBSTATE	1	Status bits:

Name	Bits	Value	Meaning
SCBTRANP	0	X'80'	Message in transparent mode
SCBMGFMN	1	X'40'	MSGFORM requested
SCBMGFMF	1 Off	X'BF'	Mask to specify MSGFORM not requested
SCBSEQIN	1	X'40'	Sequence-in has been executed for the current message
SCBLCK1F	2 Off	X'DF'	Mask to specify that a message is not being received in Lock
SCBMSGIN	4	X'08'	Message lock
SCBMSGLF	4 Off	X'F7'	Mask to specify extended lock
SCBCKPT	5	X'04'	Checkpoint requested
SCBPRER	6	X'02'	Previous EOB/ETB error
SCBCODE	7	X'01'	Translation requested

Offset		Name	Bytes	Description
1	(1)	SCBDESTQ	3	Address of the Destination QCB
4	(4)	SCBSNDCT	1	MSGLIMIT on Send side
4	(4)	SCBRCVCT	1	MSGLIMIT on Receive side
5	(5)	SCBMACR	3	First or next INMSG or OUTMSG macro to be executed
5	(5)	SCBMBHEN	3	Address of the multiple-buffer-header entry
8	(8)	SCBPRI	1	Priority index to the QCB
9	(9)	SCBBKFCT	3	Count of message length for Break
10	(A)	SCBEOBSZ	1	Size of logical blocks
12	(C)	SCBSALEV	1	Simulated attention level request - TSO
12	(C)	SCBQTYPE	1	Queuing medium for this message:

Name	Bits	Value	Meaning
SCBCOREQ	1	X'40'	Main storage queues
SCBREUS	2	X'20'	Reusable disk queues
SCBNREUS	3	X'10'	Nonreusable disk queues
SCBNOFF	5	X'04'	Reusability has updated SCBFEFO
SCBBFTM	6	X'02'	Buffered terminal SCB
SCBBFMM	7	X'01'	Buffered terminal in middle of message

Offset		Name	Bytes	Description
13	(D)	SCBMRFPL	3	Address of the FORWARD parameter list
16	(10)	SCBERRST		Error word bits
16	(10)	SCBERR1	1	First byte:

Name	Bits	Value	Meaning
SCBHDRRN	0	X'80'	Incomplete header
SCBHDRRF	0 Off	X'7F'	Mask to specify not an incomplete header
SCBNOLOG	0	X'80'	Invalid Logon message - TSO
SCBORIGN	1	X'40'	Invalid origin
SCBORIGF	1 Off	X'BF'	Mask to specify a valid origin
SCBHANG	1	X'40'	Logon requests hangup message - TSO
SCBNOTRM	2	X'20'	Not a TSO terminal - TSO
SCBSEQHN	3	X'10'	Sequence high
SCBSEQHF	3 Off	X'EF'	Mask to specify that sequence is not high
SCBNOTSO	3	X'10'	TSO is not in the system - TSO
SCBSEQLN	4	X'08'	Sequence low
SCBSEQLF	4 Off	X'F7'	Mask to specify that sequence is not low
SCBNOVAC	4	X'08'	Too many TSO users - TSO
SCBNOBFN	6	X'02'	Insufficient buffers
SCBCUTFN	7	X'01'	CUTOFF error
SCBCUTFF	7 Off	X'FE'	Mask to specify no CUTOFF error
SCBRVISL	7	X'01'	RVI to selection on a buffered terminal

Offset		Name	Bytes	Description
17	(11)	SCBERR2	1	Second byte:

Name	Bits	Value	Meaning
SCBCRMIN	0	X'80'	Main storage minimum passed
SCBCRMAX	1	X'40'	Main storage maximum passed
SCBCODER	2	X'20'	Error in dynamic translate - TSO
SCBALN	3	X'10'	Automatic line numbering - TSO
SCBOLTR	4	X'08'	TOTE not in the system
SCBABRTN	5	X'04'	Abort - BSC terminal
SCBFRWDN	6	X'02'	Terminal FORWARD error

Offset		Name	Bytes	Description
18	(12)	SCBERR3	1	Third byte:

Name	Bits	Value	Meaning
SCBLOSTN	0	X'80'	Message lost (overlaid)
SCBLOSTF	0 Off	X'7F'	Mask to specify message processed
SCBXPI	0	X'80'	Attention - Send - I - TSO
SCBTMIDN	1	X'40'	ID from terminal invalid
SCBTMIDF	1 Off	X'BF'	Mask to specify that terminal identification is valid
SCBXPD	1	X'40'	Attention - Send - D - TSO
SCBSATTN	3	X'10'	Simulated Attention received - TSO
SCBUSERN	4	X'08'	User error
SCBUSERF	4 Off	X'F7'	Mask to specify no user error
SCBFORMN	5	X'04'	Format error - BSC message
SCBATTN	6	X'02'	Hardware Attention - TSO
SCBXCEPN	7	X'01'	Unit exception
SCBXCEPF	7 Off	X'FE'	Mask to specify no unit exception

Offset		Name	Bytes	Description
19	(13)	SCBERR4	1	Fourth byte:

Name	Bits	Value	Meaning
SCBSLCTN	0	X'80'	Error during selection
SCBSLCTF	0 Off	X'7F'	Mask to specify no selection error
SCBTXITN	1	X'40'	Error during text transfer
SCBTXITF	1 Off	X'BF'	Mask to specify no text transfer error
SCBCONNN	2	X'20'	Error in Connect/Disconnect
SCBCONNF	2 Off	X'DF'	Mask to specify no Connect/Disconnect error
SCBTRMLN	3	X'10'	Terminal error
SCBTRMLF	3 Off	X'EF'	Mask to specify no terminal error
SCBCTLUN	5	X'04'	Error in the control unit
SCBCTLUF	5 Off	X'FB'	Mask to specify no control unit error
SCBCHANN	6	X'02'	Error in channel
SCBCHANF	6 Off	X'FD'	Mask to specify no error in channel
SCBUNDFN	7	X'01'	Undefined error - should not occur
SCBUNDFF	7 Off	X'FE'	Mask to specify no undefined error

Offset		Name	Bytes	Description
20	(14)	SCBMRFSD	2	Multiple routing first secondary destination
22	(16)	SCBEOBAC	2	Accumulated count between blocks
22	(16)	SCBDLPTR	2	Distribution list pointer
24	(18)	SCBBSCFM	1	MSGFORM dynamic block changes:

Name	Bits	Value	Meaning
SCBTRNSP	0	X'80'	Receiving transparent
SCBRCVTX	2	X'20'	ETX received from BSC
SCBNONTR	1	X'40'	Receiving non-transparent
SCBNOEOT	6	X'02'	BSC Dial no EOT before read
SCBMLMTN	7	X'01'	MSGLIMIT has been exceeded
SCBMLMTF	7 Off	X'FE'	Mask to specify that MSGLIMIT is not exceeded

Offset		Name	Bytes	Description
25	(19)	SCBMBSSA	7	Multiple buffer scan save area
32	(20)	SCBCPBNO	1	Number of the next sequential CPB to be read from disk
33	(21)	SCBDCHDR	3	Disk address of the current header
36	(24)	SCBDESTL	1	Length of destination names
37	(25)	SCBCCHDR	3	Main storage address of the current header
40	(28)	SCBITPSZ	1	Size of logical subblocks
41	(29)	SCBSCSEG	3	Current segment being read
41	(29)	SCBDNSEG	3	Disk address of the next segment from the disk to send
44	(2C)	SCBHBFNO	1	Number of buffers in the multiple-buffer header
45	(2D)	SCBSCHDR	3	Current header being sent
45	(2D)	SCBCLSEG	3	Main storage address of the last message sent

Offset		Name	Bytes	Description
48	(30)	SCBITBAC	1	Accumulated count between ITBs
49	(31)	SCBFEFO	3	Saved FEFO pointer
49	(31)	SCBDCSEG	3	Disk address of the current segment
52	(34)	SCBDEOB	4	Disk information on the last EOB

Note: The section in bytes 54-79 is a copy of the last buffer prefix processed.

Offset		Name	Bytes	Description
54	(38)	SCBSRCE	2	Message buffer source
58	(3A)	SCBSIZE	2	Message buffer size
60	(3C)	SCBSTAT1	1	Status byte
61	(3D)	SCBXTRA	3	Address of additional records
61	(3D)	SCBCORE	3	Address of the record in the main storage
64	(40)	SCBOSEQ	2	Sequence-out number
64	(40)	SCBSCAN	2	Scan pointer address
66	(42)	SCBTQBCK	3	Text segment queue-back chain
66	(42)	SCBNTXT	3	Address of the next text segment
69	(45)	SCBCRCD	3	Address of the current segment
72	(48)	SCBNHDR	3	Address of the next header segment
72	(48)	SCBCHDR	3	Address of the current header segment

75	(4B)	SCBNXCPB	1	Next CPB number from disk;
				if zero - no multiple routing

75	(4B)	SCBCCSEG	3	Main storage address of the current segment
78	(4E)	SCBEOB	2	Pointer to the first EOB
80	(50)	SCBUNTCT	1	Count of the first byte of data in the disk record
81	(51)	SCBTRANS	3	Current translation table address
84	(54)	SCBRGSAV	4	Save area for user MH registers if specified on INTRO

CHANNEL PROGRAM BLOCK

The Channel Program Block (IEDQCPB) contains the disk channel program and other information pertinent to the disk I/O involved. Within the channel program the CPB contains pointers to its associated unit and to the next CPB as well as the actual number of the unit being processed and its MBBCCHHR equivalent. The address of the first CPB is in the AVTCPBPT field of the Address Vector Table. The same address is in the AVTFCPB field of the AVT at INTRO execution time, but this field changes during the execution of the channel program as it always points to the first CPB in the LIFO CPB queue.

Storage is allocated and the CPB is initialized at execution time for the INTRO macro. At INTRO time, each CPB in the CPB free pool has a unit assigned to it. Initially this unit is contiguous with the CPB, but as processing continues, the unit may be from the buffer unit pool. The CPBXREA field points to the associated unit, which is actually the disk data area.

The figure below is the format of the Channel Program Block; descriptions of the fields follow the illustration.

IEDQCPB

0 (0) **CPBHEADF** Seek Head CCW			
CPBSEEK OP Code	1 (1) **CPBHEAD** Head ID Address		
4 (4) **CPBHEADF** (Cont.)			
CPBSEKFL Seek Flag	5 (5) Reserved	6 (6) **CPBSEKCT** Seek Count	
8 (8) **CPBSRECF** Search ID Equal CCW			
CPBSRCH OP Code	**CPBSREC** Record ID Address		
12 (C) **CPBSRECF** (Cont.)			
CPBSRHFL Search Flag	13 (D) Reserved	14 (E) **CPBSRHCT** Search Count	
16 (10) **CPBTICSF** TIC to Search CCW			
CPBTIC1 OP Code	1 (1) **CPBTICS** Search CCW Address		
20 (14) **CPBTICSF** (Cont.)			
CPBUNUSD Reserved			
24 (18) **CPBAREAF** Read/Write CCW			
CPBRDWR OP Code	25 (19) **CPBAREA** I/O Area Address		
28 (1C) **CPBAREAF** (Cont.)			
CPBRWFL Read/Write Flag	29 (1D) Reserved	30 (1E) **CPBCOUNT** Number of Bytes to Read or Write	
32 (20) **CPBXREAF** Second Read/Write CCW			
CPBXDWR OP Code	33 (21) **CPBXREA** I/O Area Address		
36 (24) **CPBXREAF** (Cont.)			
CPBXWFL Read/Write Flag	37 (25) Reserved	38 (26) **CPBXOUNT** Number of Bytes to Read or Write	
40 (28) **CPBNEXTF** TIC to Next CPB CCW			
CPBTIC2 OP Code	41 (29) **CPBNEXT** Next CPB Address		
44 (2C) **CPBNEXTF** (Cont.)			
CPBFLAG Flag Byte	**CPBADDR** Absolute Record Number		
48 (30) **CPBABSAD** MBBCCHHR Value			
56 (38) **CPBINWKA** Work Area Data Count	57 (39) **CPBTOUNT** Data to be Moved Count	58 (3A) **CPBWKACT** Work Area Start Address	59 (3B) **CPBNUMB** Current CPB Number
60 (3C) **CPBAERBF** ERB Address			
CPBUNTCT Unit Data Count	61 (3D) **CPBAERB** ERB Address		

Offset	Name	Bytes	Description
0 (0)	CPBHEADF		Start of the Seek Head CCW
0 (0)	CPBSEEK	1	Seek Head OP Code
1 (1)	CPBHEAD	3	Pointer to the head ID
4 (4)	CPBSEKFL	1	Seek CCW flag, command chaining
6 (6)	CPBSEKCT	2	Seek count of 6
8 (8)	CPBSRCH	1	Search ID Equal OP Code
9 (9)	CPBSREC	3	Pointer to the record ID
12 (C)	CPBSRHFL	1	Search CCW flag
13 (D)		1	Reserved
14 (E)	CPBSRHCT	3	Search count of 5
16 (10)	CPBTICSF		Start of the TIC to Search CCW
16 (10)	CPBTIC1	1	TIC OP code
17 (11)	CPBTICS	3	Address of the Search CCW
20 (14)	CPBUNUSD		Reserved
24 (18)	CPBAREAF		Start of the Read/Write CCW
24 (18)	CPBRDWR	1	Read/Write OP Code
25 (19)	CPBAREA	3	Address of the I/O area
28 (1C)	CPBRWFL	1	Read/Write flag
30 (1E)	CPBCOUNT	2	Number of bytes to be read or written
32 (20)	CPBXREAF		Start of the second Read/Write CCW
32 (20)	CPBXDWR	1	Read/Write OP code
33 (21)	CPBXREA	3	Address of the I/O area
36 (24)	CPBXWFL	1	Read/Write flag
38 (26)	CPBXOUNT	2	Number of bytes to be read or written
40 (28)	CPBNEXTF		Start of the TIC to Next CPB CCW
40 (28)	CPBTIC2	1	TIC OP code
41 (29)	CPBNEXT	3	Pointer to the next CPB
44 (2C)	CPBFLAG	1	Flag byte:

Name	Bits	Value	Meaning
CPBREUSN	0	X'80'	Special CPB belongs to the Reusability subtask
CPBCOPYN	1	X'40'	Special CPB belongs to the Copy subtask
CPBRDSKN	2	X'20'	Reusable disk
CPBNDSKN	3	X'10'	Non-reusable disk

The following list settings are for the reusability-copy function:

4,6	X'0A'	Update the FEFO queue pointer
4,5	X'0C'	Read the disk data field to get the destination FEFO pointer for a message being serviced; always followed by a X'0B'
4,6,7	X'0B'	Write the disk data field of the last held message to relink the held FEFO queue into the destination FEFO queue in order to obtain one queue (done at the start of servicing the Destination Priority QCB)
4,7	X'09'	Write the "serviced" bit in the original first unit after the entire message has been moved to the message queue for the alternate destination; usually followed by a X'0A'
4	X'08'	Read the first unit of the additional records; always followed by a X'05'
5,6,7	X'07'	Write the additional units for each message buffer; always follows a X'06'
5,6	X'06'	Read the additional units for each message buffer; always followed by a X'07'
5,7	X'05'	Write the first unit of each message buffer to the alternate destination
5	X'04'	Write the disk data field relink the FEFO queue to bypass the old message;

Offset	Name	Bytes	Description		
					usually follows a X'03'
			6,7	X'03'	Read the key and data fields of a subsequent message that has to be assigned to an alternate destination
			6	X'02'	Read the disk data field of the first message to check the FEFO pointer
			7	X'01'	Read the key and data fields of the first message that has to be assigned to an alternate destination
			-	X'00'	Cancel the record (text is all 'C's)
44 (2C)	CPBADDR	4	Absolute record number		
48 (30)	CPBABSAD	8	MBBCCHHR value		
56 (38)	CPBINWKA	1	Count of the data in the work area		
56 (38)		4	LCB address, if the CPB is for IGG019RP		
57 (39)	CPBTOUNT	1	Count of the data to be moved into a unit		
58 (3A)	CPBWKACT	1	Where to start in the work area		
59 (3B)	CPBNUMB	1	Sequential number of the current CPB		

Offset	Name	Bytes	Description
60 (3C)	CPBAERBF	4	Address of the ERB, or the work area unit address (for IGG019RP)
60 (3C)	CPBUNTCT	1	Count of data already in the unit
61 (3D)	CPBAERB	3	

The following are the CCW bit definitions:

	Name	Bits	Value	Meaning
CCW Flags:				
	CPBCDC	0	X'80'	Data chaining
	CPBCCC	1	X'40'	Command chaining
	CPBSLIC	2	X'20'	Suppress incorrect length
	CPBSKIPC	3	X'10'	Skip data
CCW Commands:				
	CPBTICC	4	X'08'	TIC command
	CPBSEEKC	3,4, 6,7	X'1B'	Seek Head command
	CPBRDKC	4,5,6	X'0E'	Read Key and Data command
	CPBWRKC	4,5,7	X'0D'	Write Key and Data command
	CPBRDC	5,6	X'06'	Read Data command
	CPBWRC	5,7	X'05'	Write Data command
	CPBSRCHC	2,3,7	X'31'	Search ID Equal command
	CPBNOPC	6,7	X'03'	NO OP command
	CPBWRITB	7	X'01'	Write Data or Key and Data bit
	CPBKEYB	4	X'08'	Key bit

DATA CONTROL BLOCK

The Data Control Block (DCB) is a storage area through which information needed for the access routines to store and retrieve data is communicated. The format of a TCAM DCB is determined by the character of the data set it represents. There are five types of Data Control Blocks used in TCAM message control programs and application programs They are

Line Groups
Message Queues
Checkpoint
Message Logging
Application Program

The TCAM DCB is divided into three segments - Prefix, Foundation, and Extension. The contents of the Foundation segment changes during processing. Storage is allocated for the DCB at assembly time, and it is initialized partially at assembly time and partially at execution time according to the parameters specified on the DD card. Before open time, the first doubleword of the Foundation segment (at a displacement of 40 (X'28) from the beginning of the DCB) contains the ddname of the data set to be opened. After the data set is opened, the same doubleword contains the address of the Data Extent Block. This address is used to set up linkages in the TCAM execution.

The address of the TCAM Data Control Block is in the DEBDCBAD field of the Data Extent Block. The same address is also in the QCBDCBAD field of the Destination Queue Control Block.

The figure below is the format of a Data Control Block; descriptions of the fields follow the illustration.

Data Control Block DSECT (IHADCB)

Data Set Interface

Line Group

Offset	Field	Description
20 (14)	DCBUFOU DCBUFIN	Number of Buffers
21 (15)	DCBMH	MH Address for this Line Group
24 (18)	DCBINTVL	Invitation Delay Interval
25 (19)	DCBPCI	PCI Byte
26 (1A)	DCBDSORG	Data Set Organization
28 (1C)	DCBUFMA	Maximum Buffer Count for Transfer
	DCBIOBAD	IOB Base Address
32 (20)	DCBCPRI	Priority
33 (21)	DCBTRANS	Translation Table Address
36 (24)	DCBEIOBX	Extended IOB Index
	DCBEXLST	Exit List Address

Message Queues

Offset	Field	Description
20 (14)		Reserved
26 (1A)	DCBDSORG	Data Set Organization
28 (1C)		Reserved
	DCBIOBAD	Before Open, AVT Address
32 (20)	DCBTHRES	Disk Threshold Value
33 (21)		Reserved
36 (24)		Reserved
	DCBEXLST	Exit List Address

Checkpoint

Offset	Field	Description
20 (14)		Reserved
26 (1A)	DCBDSORG	Data Set Organization
28 (1C)		Reserved
	DCBIOBAD	Before Open, AVT Address
32 (20)		Reserved
36 (24)		Reserved
	DCBEXLST	Exit List Address

Data Set Interface (Cont.)

Message Logging

20 (14)	Reserved
32 (20)	**DCBEODAD** DECB Pointer
36 (24)	Reserved

Application Program

20 (14)	Reserved		
24 (18)	**DCBBUFL** Buffer Length	26 (1A)	**DCBDSORG** Data Set Organization
28 (1C)	Reserved		
32 (20)	**DCBEODAD** End-of-File Routine Address		
36 (24)	**DCBRECFM** Record Format		**DCBEXLST** Exit List Address

Foundation

Before OPEN

40 (28) **DCBDDNAM** Data Set Name		
48 (30) **DCBOFLGS** Open Flags	49 (31) **DCBIFLG** IOS Error Flags	50 (32) **DCBMACR** Macro Instruction Reference

After OPEN

40 (28) **DCBTIOT** DD Offset	42 (2A) **DCBMACRF** Macro Instruction Reference
44 (2C) **DCBIFLGS** IOS Error Flags	**DCBDEBAD** DEB Address
48 (30) **DCBOFLGS** Open Flags	

Extension

Line Group

48 (30) Reserved	**DCBSCTAD** Special Characters Table Address	
52 (34) **DCBILCT** Count of Invitation Lists	53 (35) **DCBUNTCT** Unit Count	54 (36) **DCBBUFSI** Buffer Size
56 (38) **DCBRESER** Reserve Bytes Counts		
60 (3C) **DCBINVLI** Invitation List Address		
DCBINVLI		
DCBINVLI		
DCBINVLI		

Message Queues/Checkpoint

48 (30) Reserved	
52 (34) **DCBOPTCD** Code Byte	53 (35) Reserved

Extension (Cont.)

Message Logging

48 (30) Reserved

DCBREAD, DCBWRITE
READ or WRITE Module Address

52 (34)

Reserved

72 (48) DCBNCP
Count of Write
Operations

73 (49)

Reserved

Application Program

48 (30) Reserved

DCBREAD, DCBWRITE
DCBGET, DCBPUT
READ/WRITE or GET/PUT Module Address

52 (34) DCBOPTCD
Code Byte

DCBCHECK
CHECK Module Address

56 (38)

DCBSYNAD
Synchronizing Routine Address

60 (3C)

Reserved

62 (3E)

DCBBLKSI
Maximum Block Size

64 (40)

Reserved

82 (52)

DCBLRECL
Logical Record Length

84 (54)

DCBCNTRL, DCBNOTE, DCBPOINT
CNTRL or NOTE/POINT Routine Address

Offset	Name	Bytes	Description
LINE GROUP INTERFACE			
20 (14)	DCBBUFIN/ DCBBUFOT	1	Bits 0-2: Number of buffers assigned initially for receiving operations, for each line in line group Bits 4-7: Number of buffers assigned initially for sending operations, for each line in the line group
21 (15)	DCBMH	3	Address of the message handler for this line group
24 (18)	DCBINTVL	1	Number of seconds on invitation delay
24 (19)	DCBPCI	1	Program-Controlled Interruption handling byte: Bit Value Meaning 2 X'20' PCI=(A,) 3 X'10' PCI=(,A) 4 X'08' PCI=(N,) 5 X'04' PCI=(,N) 6 X'02' PCI=(R,) 7 X'01' PCI=(,R)
26(1A)	DCBDSORG	2	Data Set Organization: Byte 0 - Code = TX
28 (1C)	DCBBUFMA	1	Maximum number of buffers to be used for data transfer for each line in this group
28 (1C)	DCBIOBAD	4	Before open - address of AVT; after open - base for addressing IOBs (Base=address of first IOB minus length of one LCB)
32 (20)	DCBCPRI	1	Relative priority to be given to sending and receiving operations Bits Value Meaning 0-4 Reserved bits 5 X'04' R - Receiving has priority 6 X'02' E - Receiving and sending have equal priority 7 X'01' S - Sending has priority
33 (21)	DCBTRANS	3	Address of the translation table Table Code IEDQ10 1030 IEDQ11 1050 IEDQ12 105F IEDQ13 1060 IEDQ14 2260 IEDQ15 2265 IEDQ16 2740 IEDQ17 274F IEDQ18 ITA2 IEDQ19 ZSC3 IEDQ20 TTYA IEDQ21 TTYB

IEDQ22	TTYC
IEDQ23	6BTI
IEDQ24	ASCI
IEDQ25	EBCD
IEDQ26	BC41
IEDQ27	EB41
IEDQ28	CR41
user table	user table

36 (24)	DCBEIOBX	1	Extended IOB index (size of an ICB)
36 (24)	DCBEXLST	4	Address of the exit list

DIRECT ACCESS STORAGE DEVICE MESSAGE QUEUE INTERFACE, CHECKPOINT DATA SET INTERFACE, MESSAGE LOGGING INTERFACE, APPLICATION PROGRAM INTERFACE

20 (14)		4	Reserved
24 (18)	DCBBUFL	2	Length of the buffer
26 (1A)	DCBDSORG	2	Data Set Organization Byte 0 - Code = TQ
28 (1C)		1	Reserved
28 (1C)	DCBIOBAD	4	Before open - address of the AVT
32 (20)	DCBTHRES	1	Percentage of the nonreusable disk message queue records to be used before a flush closedown of the system is initiated
32 (20)	DCBEODAD	4	Message logging - work area used as a DECB pointer; application program - address of user end-of-file routine
36 (24)	DCBRECFM	1	Record format
36 (24)	DCBEXLST	4	Address of the exit list

FOUNDATION SEGMENT-BEFORE OPEN

40 (28)	DCBDDNAM	8	Data set name
48 (30)	DCBOFLGS	1	Flags used by OPEN:

Bits	Value	Meaning
0,1,2, 4,5,6		Reserved
3	X'10'	Open has been successfully completed
7	X'01'	DCB is being processed by I/O support routine

49 (31)	DCBIFLG	1	Used by IOS for error conditions

Offset	Name	Bytes	Description		
50 (32)	DCBMACR	2	Macro instruction reference:		
			Bits	Value	Meaning
		Byte 1			
			0,2,3, 4,5,6,7		Reserved
			1	X'40'	GET
		Byte 2			
			0,2,3, 4,5,6,7		Reserved
			1	X'40'	PUT

FOUNDATION SEGMENT-AFTER OPEN

Offset	Name	Bytes	Description
40 (28)	DCBTIOT	2	Offset of the DD entry from beginning of the TIOT
42 (2A)	DCBMACRF	2	Same as DCBMACR before OPEN
44 (2C)	DCBIFLGS	1	Same as DCBIFIG before OPEN
45 (2D)	DCBDEBAD	3	Address of DEB
48 (30)	DCBOFLGS		Same as DCBOFLGS before OPEN

LINE GROUP EXTENSION

Offset	Name	Bytes	Description		
49 (31)	DCBSCTAD	3	Address of Special Characters Table		
52 (34)	DCBILCT	1	Count of invitation lists		
53 (35)	DCBUNTCT	1	Before open - numerical value of the SCT After open - count of units for one buffer		
54 (36)	DCBBUFSI	2	Size of all buffers used for this line group		
56 (38)	DCBRESER	4	4 one-byte values (zero default value)		
	byte 1		Number of bytes reserved in the buffer receiving the first incoming segment of a message		
	byte 2		Number of bytes reserved in all buffers except the one containing the first segment of a message		
	bytes 3-4		Reserved		
60 (3C)	DCBINVLI	4n times	4-byte address for each (n) invitation list		
			Bits	Value	Meaning
			0,1, 3,5, 6,7,		Reserved
			2	off	[A,]
			4	off	[,A]
			2	on	[B,]
			4	on	[,B]
Bytes 2-4			Reserved		

Offset	Name	Bytes	Description
		MESSAGE QUEUES/CHECKPOINT EXTENSION	
49 (31)		3	Reserved
52 (34)	DCBOPTCD	1	Code byte: Bits Value Meaning 2 X'20' Checkpoint 6 X'02' Nonreusable disk queues 7 X'01' Reusable disk queues
53 (35)		7	Reserved
	MESSAGE LOGGING EXTENSION		
48 (30)	DCBREAD, DCBWRITE	4	Address of the READ or WRITE module
52 (34)		20	Reserved
72 (48)	DCBNCP	1	Number of Write operations that can be performed
73 (49)		15	Reserved
	APPLICATION PROGRAM EXTENSION		
48 (30)	DCBREAD, DCBWRITE DCBGET,DCBPUT	4	Address of the READ or WRITE module Address of the GET or PUT module
52 (34)	DCBOPTCD	1	Option codes
52 (34)	DCBCHECK	4	Address of the CHECK module
56 (38)	DCBSYNAD	4	Address of the user synchronizing routine
60 (3C)		2	Reserved
62 (3E)	DCBBLKSI	2	Maximum block size
64 (40)		18	Reserved
82 (52)	DCBLRECL	2	Logical record length or block size
84 (54)	DCBCNTRL, DCBNOTE, DCBPOINT		Address of the CNTRL or the NOTE/POINT module

DATA EXTENT BLOCK

The Data Extent Block (DEB) is a fixed length control block with a 36-byte prefix. The DEB describes the extents of the data set with which the DEB is associated. The DEB contains such addresses as the DCB, the UCB, and the TCB. The number of extents associated with the data set is also in the DEB. For line groups, the DEB contains the number of lines in a line group and with which line number the data set is used. For a message queue, the DEB contains the number of extents of the data set and their size. The Data Extent Block prefix contains the addresses of the data set appendages (the PCI Appendage, the Channel End Appendage, and others).

The address of the DEBTCBAD field of the Data Extent Block is in the DCBDEBAD field of the Data Control Block. The address of the beginning of the DEB prefix is at a displacement of -36(-X'24') from the address of the DEBTCBAD field. Storage is allocated for and the DEB is initialized at open time.

Note: The displacements on this control block do not agree with the TDEBD macro, which has the relative zero displacement at DEBFOEA. The disk message queues routines use the TDEBD macro offsets. The AVTADEBN and AVTADEBP fields of the TCAM AVT contain the address of the DEBFOEA field of the DEB.

The figure below is the format of the DEB prefix and the Data Extent Block itself; descriptions of the fields follow the illustration.

IEDQDEB

-36 (-24)	DEBEOEA	Address of the End-of-Extent Appendage			
-32 (-20)	DEBSIOA	Address of the Start I/O Appendage			
-28 (-1C)	DEBPCIA	Address of the PCI Appendage			
-24 (-18)	DEBCEA	Address of the Channel End Appendage			
-20 (-14)	DEBXCEA	Address of the Abnormal End Appendage			
-16 (-10)	DEBWKARA	I/O Support Work Area	-15 (-F)	DEBDSCBA	Address of the DSCB
-8 (-8)	DEBDCBMK	DCB Modification Mask			
-4 (-4)	DEBLNGTH	Length of the DEB in Double Words			
0 (0)	DEBNMSUB	Number of OPEN Subroutines		DEBTCBAD	Address of the TCB
4 (4)	DEBAMLNG	Length of Access Method Section		DEBDEBAD	Address of the Next DEB
8 (8)	DEBOFLGS	Data Set Flags		DEBIRBAD	Address of the IRB
12 (C)	DEBOPATB	Type of I/O		DEBSYSPG	Address of the First IOB in the System Purge Chain
16 (10)	DEBNMEXT	Number of Extents		DEBUSRPG	Address of the First IOB in the User Purge Chain
20 (14)	DEBPRIOR	Zero		DEBECBAD	Address of the Parameter List to Find the Purge ECB
24 (18)	DEBPROTG	Protection Key DEB ID		DEBDCBAD	Address of the DCB
28 (1C)	DEBEXSCL	Extent Scale		DEBAPPAD	Address of the I/O Appendage Vector Table
32 (20)	DEBDVMOD	Device Modifier		DEBUCBAD	Address of the UCB

Offset		Name	Bytes	Description
-36	(-24)	DEBEOEA	4	Address of End-Of-Extent Appendage
-32	(-20)	DEBSIOA	4	Address of Start I/O Appendage
-28	(-1C)	DEBPCIA	4	Address of PCI Appendage
-24	(-18)	DEBCEA	4	Address of Channel End Appendage
-20	(-14)	DEBXCEA	4	Address of Abnormal and Normal Line End Appendage
-16	(-10)	DEBWKARA	1	I/O Support work area
-15	(-F)	DEBDSCBA	7	Address of DSCB
-8	(-8)	DEBDCMK	4	DCB modification mask
-4	(-4)	DEBLNGTH	4	Length of the DEB in doublewords
0	(0)	DEBNMSUB	1	Number of OPEN subroutines
0	(0)	DEBTCBAD	4	Address of the TCB
4	(4)	DEBAMLNG		Length access method section
4	(4)	DEBDEBAD	4	Address of the next DEB
8	(8)	DEBOFLGS	1	Data set flags
8	(8)	DEBIRBAD	4	Address of the IRB
12	(C)	DEBOPATB	1	Type of I/O
12	(C)	DEBSYSPG	4	Address of the first IOB in the System Purge chain
16	(10)	DEBNMEXT	1	Number of extents
16	(10)	DEBUSRPG	4	Address of the first IOB in the User Purge chain
20	(14)	DEBPRIOR	1	Zero
20	(14)	DEBECBA	4	Address of the parameter list to find the purge ECB
24	(18)	DEBPROTG		Protection Key DEB ID
24	(18)	DEBDCBAD	4	Address of the DCB
28	(1C)	DEBEXSCL	1	Extent Scale

Offset	Name	Bytes	Description
28 (1C)	DEBAPPAD	4	Address of the I/O Appendage Vector Table
32 (20)	DEBDVMOD	1	Device modifier
32 (20)	DEBUCBAD	4	Address of the UCB

DATA EVENT CONTROL BLOCK

The Data Event Control Block (DECB) is created when a READ or WRITE macro instruction is expanded. It contains information about the input or output operation that is requested by the macro instruction.

The figure below shows the format for the Data Event Control Block; descriptions of the fields follow the illustration.

DECB

0 (0) **DECSDECB** Event Control Block	
4 (4) **DECTYPE** Reserved	6 (6) **DECLNGTH** Length of Data or of Key and Data
8 (8) **DECDCBAD** DCB Address	
12 (C) **DECAREA** Read/Write Area Address	
16 (10) **DECIOBPT** Reserved	

Offset	Name	Bytes	Description
0 (0)	DECSDECB	4	Event Control Block
4 (4)	DECTYPE	2	Reserved
6 (6)	DECLNGTH	2	Length of key and data (if there is a key); length of work area for an application program
8 (8)	DECDCBAD	4	Address of the DCB to which this I/O request is related
12 (C)	DECAREA	4	Address of the read/write area; address of work area for an application program
16 (10)	DECIOBPT	4	Reserved

OS I/O DEVICE CHARACTERISTICS TABLE

The OS I/O Device Characteristics Table is a variable length table that contains one twelve-byte entry for each direct access device in the system. The table contains such information as the number of cylinders, the number of tracks per cylinder, the overhead for each intermediate record on the track, and the tolerance factor for each intermediate record. The OS I/O Device Characteristics Table is used by the Checkpoint Disk Allocation routine (IGG01949) to obtain data about the specific direct access device used for the checkpoint data set. The table is also used by the Disk Message Queue Open - 1 routine (IGG01930) to determine the number of tracks per cylinder for the current data set being opened (to determine whether the device is a 2311 or a 2314).

The address of the OS I/O Device Characteristics Table is in the CVTZDTAB field of the CVT. The Unit Control Block contains the index to the specific entry in the table.

Storage is allocated for the OS I/O Device Characteristics Table and it is initialized at OS IPL time.

The figure below is the format of one entry in the OS I/O Device Characteristics Table; descriptions of the fields follow the illustration.

IEDQDCTD

<table>
<tr><td>0 (0)
Reserved</td><td>1 (1)
DCTCYL
Cylinder Count</td><td colspan="2">2 (2)
DCTRACKS
Number of Tracks Per Cylinder</td></tr>
<tr><td colspan="2">4 (4)
DCTBYTE
Number of Bytes Per Track</td><td>6 (6)
DCTINTRO
Overhead</td><td>7 (7)
DCTLASTO
Overhead</td></tr>
<tr><td>8 (8)
DCTKEY
Overhead</td><td>9 (9)
Reserved</td><td colspan="2">10 (A)
DCTOLERN
Tolerance Factor</td></tr>
</table>

Offset	Name	Bytes	Description
0 (0)		1	Reserved
1 (1)	DCTCYL	1	Number of cylinders
2 (2)	DCTRACKS	2	Number of tracks per cylinder
4 (4)	DCTBYTE	2	Number of bytes per track
6 (6)	DCTINTRO	1	Overhead for each intermediate record
7 (7)	DCTLASTO	1	Overhead for the last record on a track
8 (8)	DCTKEY	1	Overhead if keys are not used
9 (9)	DCTOLERN	1	Reserved
10 (A)	DCTOLERN	2	Tolerance factor for each intermediate record

BUFFER PREFIX

First buffer of a message:

Offset

0	1	4	5		8	12 (C)	13 (D)	16 (10)	18 (12)	20 (14)	21 (15)	24 (18)	26 (1A)	29 (1D)	32 (20)	35 (23)	38 (26)	40 (28)
Key **PRFKEY** or CCW OP Code	QCB address **PRFQCBA** or Next address to be transferred	Priority **PRFPRI** or CCW flags	Link field **PRFLINK** Unused	CCW count	Link to next unit and TIC CCW	Number of units in this buffer	LCB address	Source offset in the Termname Table	Size of data in this buffer	Status byte	Pointer to additional records on disk **PRFXTRA** or to the current record in main storage	Scan pointer offset	Pointer to next buffer of this message if not last buffer **PRFNTXT** or text queue-back chain if last buffer	Pointer to the first unit of the current buffer	Pointer to the first buffer of the next message	Queue-back chain of the first buffers of messages	Input sequence number	Destination offset in the Termname Table
PRFOPCDE	**PRFIOADR**	**PRFFLAGS**		**PRFCOUNT**	**PRFTIC**	**PRFNBUNT**	**PRFLCB**	**PRFSRCE**	**PRFSIZE**	**PRFSTAT1**	**PRFCORE**	**PRFSCAN**	**PRFTQBCK**	**PRFCRCD**	**PRFNHDR**	**PRFHQBCK**	**PRFISEQ**	**PRFDEST**

RCB (offsets 0–11) — First or 30-byte Buffer Prefix (offsets 12–41)

The first 12 bytes are not placed on the queue for the message queues data set.

0	12 (C)	42 (2A)
Unit control area	First buffer prefix **PRFSUNIT**	Start of the message header or data **PRFSHDR**

Subsequent buffer of a message:

Offset

0	1	4	5		8	12(C)	13 (D)	16 (10)	18 (12)	20 (14)	21 (15)	24 (18)	26 (1A)	29 (1D)	32 (20)
Key **PRFKEY** or CCW OP Code	QCB address **PRFQCBA** or Next address to be transferred	Priority **PRFPRI** or CCW flags	Link field **PRFLINK** Unused	CCW count	Link to next unit and TIC CCW	Number of units in this buffer	LCB address	Source offset in the Termname Table	Size of data in this buffer	Status byte	Pointer to additional records on disk **PRFXTRA** or to the current record in main storage	Scan pointer offset	Pointer to next buffer of this message if not last buffer **PRFNTXT** or text queue-back chain if last buffer	Pointer to the first unit of the current buffer	Pointer to the **first buffer of the** current message
PRFOPCDE	**PRFIOADR**	**PRFFLAGS**		**PRFCOUNT**	**PRFTIC**	**PRFNBUNT**	**PRFLCB**	**PRFSRCE**	**PRFSIZE**	**PRFSTAT1**	**PRFCORE**	**PRFSCAN**	**PRFTQBCK**	**PRFCRCK**	**PRFCHDR**

RCB (offsets 0–11) — Subsequent or 23-byte Buffer Prefix (offsets 12–34)

The first 12 bytes are not placed on the queue for the message queues data set.

0	12 (C)	35 (23)
Unit control area	Subsequent buffer prefix **PRFSUNIT**	Continuation of message header or start or continuation of message data **PRFSTXT**

IEDQPRF

0 (0)	**PRFRCB**	Resource Control Block
	PRFOPCDE, PRFKEY	OP Code or Key
1 (1)	**PRFIOADR, PRFQCBA**	QCB Or Next I/O Address
4 (4)	**PRFRCB**	(Cont.)
	PRFFLAGS, PRFPRI	CCW Flags Or Priority
5 (5)	**PRFLINK**	Buffer Link Field
6 (6)	**PRFCOUNT**	CCW Count
8 (8)	**PRFTIC**	TIC CCW Or Link to Next Unit
12 (C)	**PRFSUNIT, PRFNBUNT**	Number of Units in this Buffer
13 (D)	**PRFLCB**	LCB Address
16 (10)	**PRFSRCE**	Termname Table Offset for Source of Message
18 (12)	**PRFSIZE**	Size of Data in this Buffer
20 (14)	**PRFSTAT1**	Status Byte
21 (15)	**PRFSTSO**	Start of TSO Data
	PRFXTRA	Address of Additional Records
	PRFCORE	Address of the Current Record
24 (18)	**PRFSCAN**	Scan Pointer Address
26 (1A)	**PRFNTXT**	Next Text Segment Address
	PRFTQBCK	Text Queue-Back Chain
28 (1C)	**PRFNTXT, PRFTQBCK**	(Cont.)
29 (1D)	**PRFCRCD**	Current Segment Address
32 (20)	**PRFNHDR**	Address of the Next Header Segment
	PRFCHDR	Address of the Header of the Current Message
35 (23)	**PRFSTXT**	Start of Text
	PRFHQBCK	Header Queue-Back Chain
36 (24)	**PRFHQBCK**	(Cont.)
38 (26)	**PRFISEQ**	Input Sequence Number
40 (28)	**PRFDEST**	Termname Table Offset for Destination of Message

Offset	Name	Bytes	Description
0 (0)	PRFRCB	8	Resource Control Block
0 (0)	PRFOPCDE	1	CCW Operation Code, when I/O is being performed
0 (0)	PRFKEY	1	Element key of the buffer
1 (1)	PRFIOADR	3	Next data byte (address) to be transfered (read), when I/O is being performed
1 (1)	PRFQCBA	3	QCB address, when the buffer is an element
4 (4)	PRFFLAGS	1	CCW flags, when I/O is being performed
4 (4)	PRFPRI	1	Priority of the buffer, when it is an element
5 (5)	PRFLINK	3	Link field of the buffer, unused when a CCW
6 (6)	PRFCOUNT	2	CCW (Read/Write) count
8 (8)	PRFTIC	4	TIC CCW & link to the next unit
12 (C)	PRFSUNIT		Start of the logical unit
12 (C)	PRFNBUNT	1	Number of units in this buffer
13 (D)	PRFLCB	3	Pointer to the LCB
16 (10)	PRFSRCE	2	Termname Table offset for the source of the message
18 (12)	PRFSIZE	2	Size of data in this buffer
20 (14)	PRFSTAT1	1	Status byte:

Name	Bits	Value	Meaning
PRFCNCLN	0	X'80'	Cancel message has been executed
PRFCNCLF	All on	X'FF'	Mask to specify that the message is not canceled
PRFERMGN	1	X'40'	Error message is in this buffer
PRFERMGF	1 Off	X'BF'	Mask to specify that this is not an error message buffer
PRFITCPN	2	X'20'	Message is being held
PRFITCPF	2 Off	X'DF'	Mask to specify that the message is not being held
PRFTSBUF	3	X'10'	This is a TSO buffer
PRFDUPLN	4	X'08'	Duplicate-header buffer
PRFDUPLF	4 Off	X'F7'	Mask to specify an original buffer
PRFEOFN	5	X'04'	SETEOF was executed
PRFEOFF	5 Off	X'FB'	Mask to specify that SETEOF was not executed
PRFLOCK=PRFQPN			LOCK executed this message

		PRFNLSTN	6	X'02'	Not the last buffer of a message
		PRFNLSTF	6 Off	X'FD'	Mask to specify the last buffer of a message
		PRFNHDRN	7	X'01'	Not the first buffer of a message
		PRFNHDRF	7 Off	X'FE'	Mask to specify the first buffer of a message
21 (15)	PRFSTSO				Start of time sharing data
21 (15)	PRFXTRA	3			Pointer to the additional records
21 (15)	PRFCORE	3			Pointer to the current record
24 (18)	PRFSCAN	2			Scan pointer address
26 (1A)	PRFNTXT	3			Pointer to the next text segment
26 (1A)	PRFTOBCK	3			Queue-back chain of text segments
29 (1D)	PRFCRCD	3			Pointer to the current segment
32 (20)	PRFNHDR	3			Pointer to the next header segment
32 (20)	PRFCHDR	3			Pointer to the header of the current message
35 (23)	PRFSTXT				Start of text data in a subsequent buffer
35 (23)	PRFHOPCK	3			Queue-back chain of header segments
38 (26)	PRFISEQ	2			Input sequence number
40 (28)	PRFDEST	2			Termname Table offset for the destination of the message

DISK DATA AREA

The disk record is composed of count, key, and data. The count field is set at disk initialization time. When a unit is used as a disk buffer, the data portion of the disk record comes from the first six bytes of the unit, and the key portion of the disk record, which contains the text of the message itself, comes from that portion of the unit following the twelve-byte unit prefix. The Disk Data Area is the first six bytes of the unit prefix. When the unit is a disk buffer or is going through the channel, the address of the Disk Data Area is in the Read or Write Data CCW in the Channel Program Block. The address of the Disk Data Area is usually also in the CPBXREA field of the Channel Program Block.

Storage is allocated for the Disk Data Area at IEDQXA execution time. At that same time, the Disk Data Area is initialized to zeroes. The actual data in the Disk Data Area is placed there either by Destination Scheduler (IEDQHM) or by Reusability-Copy (IGG019RR).

The first six bytes of the IEDQDATA DSECT defines the data portion of the disk record (the Disk Data Area). The last two bytes of the DSECT are bytes seven and eight of the unit prefix and are used only in main storage (they are not written to disk and are, therefore, not part of the Disk Data Area).

The figure below is the format of the IEDQDATA DSECT; descriptions of the fields follow the illustration.

IEDQDATA

0 (0) **DATFLAGS** Flag Byte	1 (1) **DATFEFO** FEFO Pointer
4 (4) **DATCOUNT** Text Byte Count **DATSEQOT** Output Sequence Number	6 (6) **DATSCAN** Scan Pointer Save Area

Offset	Name	Bytes	Description
0 (0)	DATFLAGS	1	Flag byte:

Name	Bits	Value	Meaning
DATNPRFX	0	X'80'	No prefix in record
	1 Off	X'7F'	Mask to specify that a prefix is in the record
DATSENT	1	X'40'	Message has been serviced
	1 Off	X'BF'	Mask to specify that the message has not been serviced
DATCNCID	2	X'20'	Message is canceled
	2 Off	X'DF'	Mask to specify that the message is not canceled
DATLOSTN	3 Off	X'EF'	Mask to specify that a message is lost from main storage queue
		C'C'	IGG019RP has canceled an unused header

Offset	Name	Bytes	Description
1 (1)	DATFEFO	3	FEFO pointer to the next message to be completely received for this destination
4 (4)	DATCOUNT	2	For text records only, the number of bytes of significant text in this record key field, or zero if not the last text record
4 (4)	DATSEQOT	2	For header records only, the sequence-out number
6 (6)	DATSCAN	2	Saves the scan pointer (number of reserve characters remaining) while building a buffer from this unit; not used in a main storage disk message queue data set and not part of the Disk Data Area

APPLICATION PROGRAM DATA AREAS

PROCESS CONTROL BLOCK

The Process Control Block (IEDQPCB) is a fixed length table that serves as a named control block to permit inter-region communications between application programs and the message control program. There is one Process Control Block per application program.

The Process Control Block can be addressed by several means. The PEPCBAD field of process entry work area contains the address of the PCB, as do the ICBDCBPT field of the application program LCB, the DEBPCBAD field of the Data Extent Block, and the QCBDCBAD field of the Destination QCB.

Storage is allocated for the Process Control Block at assembly time for the message control program. The control block is initialized partially at assembly time for the MCP and partially at the application program open time.

The fields PCBBUFIN and PCBBUFO take up one byte in main storage. PCBBUFIN represents the first four bits of the byte and indicates the initial buffer request for PUT or WRITE. PCBBUFO represents the last four bits and indicates the initial buffer request for a GET/READ operation.

The figure below is the format of the Process Control Block; descriptions of the fields follow the illustration.

IEDQPCB

<table>
<tr><td colspan="4">0 (0)
Reserved</td></tr>
<tr><td colspan="4">8 (8)
PCBRTQCB
Message Retrieval QCB</td></tr>
<tr><td>20 (14) **PCBBUFIN**
PUT/WRITE Buffer Request
PCBBUFO
Max No. of Full QCB Buffers</td><td colspan="3">21 (15)
PCBMH
Address of the Message Handler</td></tr>
<tr><td>24 (18)
PCBUCNT
Use Count</td><td colspan="3">25 (19)
PCBLINK
Link Field</td></tr>
<tr><td>28 (1C)
PCBBUFMX
Read-Ahead Buffer Limit</td><td colspan="3">29 (1D)
Reserved</td></tr>
<tr><td colspan="4">32 (20)
PCBLCBAD
Address of the Line Control Block</td></tr>
<tr><td colspan="2">36 (24)
PCBTJID
TSO Job Identifier</td><td colspan="2">38 (26)
PCBCKPT
Checkpoint Offset</td></tr>
<tr><td colspan="4">40 (28)
PCBTCBAD
Address of the Task Control Block</td></tr>
<tr><td>44 (2C)
PCBOFLG
Flag Bit</td><td colspan="3">45 (2D)
Reserved</td></tr>
<tr><td>48 (30)
Reserved</td><td colspan="3">49 (31)
Reserved</td></tr>
<tr><td>52 (34)
Reserved</td><td>53 (35)
PCBUNTCT
Unit Count</td><td colspan="2">54 (36)
PCBBFSZE
Buffer Size</td></tr>
<tr><td>56 (38)
PCBRSERH
Header Buffer Reserve</td><td>57 (39)
PCBRSERT
Text Buffer Reserve</td><td>58 (3A)
PCBORC
Open Return Code</td><td>59 (3B)
Reserved</td></tr>
<tr><td colspan="4">60 (3C)
PCBWRKA
Operator Control/Application Program Interface Work Area</td></tr>
<tr><td>88 (58) **PCBEND**
End of the PCB
PCBSIZE
PCB Size in Bytes</td><td colspan="3"></td></tr>
</table>

Offset		Name	Bytes	Description
0	(0)		8	Reserved
8	(8)	PCBRTQCB	12	Message Retrieval QCB
20	(14)	PCBBUFIN	1	Initial buffer request for PUT or WRITE
20	(14)	PCBBUFO	1	Maximum number of full buffers on the Read-ahead QCB
21	(15)	PCBMH	3	Address of the message handler
24	(18)	PCBUCNT	1	Use ccunt
25	(19)	PCBLINK	3	Link field
28	(1C)	PCBBUFMX	1	Read-Ahead buffer limit
29	(1D)		3	Reserved
32	(20)	PCBLCBAD	4	Address of the Line Ccntrol Block
36	(24)	PCBTJID	2	TSO job identifier
38	(26)	PCBCKPT	2	Checkpoint offset
40	(28)	PCBTCBAD	4	Address of the Task Control Block for the related application program
44	(2C)	PCBOFLG	1	Flag byte - bit settings for this field are as follows:

Name	Bits	Value	Meaning
PCBRORIN	0	X'80'	Application program can be rolled out
PCBRORIF	0 Off	X'7F'	Mask to specify that an application program cannot be rolled out
PCBTSON	1	X'40'	Application program is TSO
PCBTSOF	1 Off	X'BF'	Mask to specify that an application program is not TSO
PCBCKPTN	2	X'20'	Environment checkpoint has been taken in the MCP
PCBCKPTF	2 Off	X'DF'	Mask to specify that an environment checkpoint has not been taken in the MCP
PCBRETVN	3	X'10'	Subsequent retrieval
PCBRETVF	3 Off		Mask to specify no subsequent retrieval

Offset		Name	Bytes	Description
45	(2D)		3	Reserved
48	(30)		1	Reserved
49	(31)		3	Reserved
52	(34)		1	Reserved

Offset		Name	Bytes	Description
53	(35)	PCBUNTCT	1	Unit Count
54	(36)	PCBBFSZE	2	Buffer size
56	(38)	PCBRSERH	1	Header buffer reserve
57	(39)	PCBRSERT	1	Text buffer reserve
58	(3A)	PCBORC	1	Open return code
59	(3B)		1	Reserved
60	(3C)	PCBWRKA	28	Operator Control/Application Program interface work area
88	(58)	PCBEND	1	End of the PCB
88	(58)	PCBSIZE	1	Size in bytes of the PCB

DATA EXTENT BLOCK FOR APPLICATION PROGRAMS

There is a special application program Data Extent Block (DEB) that has the same DSECT name, IEDQDEB, as the regular TCAM DEB.
The figure below is the format of this special DEB and descriptions of the fields follow the illustration.

IEDQDEB — Application Program

0 (0)	DEBTAMID	TCAM DEB Identifier
1 (1)	DEBTCBAD	Address of the TCB for this DEB
4 (4)		Reserved
5 (5)	DEBDEBAD	Address of the Next DEB
8 (8)		Reserved
9 (9)	DEBPCBAD	Address of the Process Control Block
12 (C)	DEBTAMOS	Process Entry Termname Table Offset
14 (E)	DEBSOWA	Size of Locate Mode Work Area
16 (10)	DEBTAMPP	Post Pending Flag Byte
17 (11)	DEBQCBAD	Address of Read-Ahead QCB
20 (14)		Reserved
21 (15)	DEBTAMWA	Address of TCAM Access Method Work Area
24 (18)		Reserved
25 (19)	DEBDCBAD	Address of the DCB for this DEB
28 (1C)		Reserved
29 (1D)	DEBLCMWA	Address of Locate Mode Work Area
32 (20)	DEBEND DEBSIZE	End of DEB Size of DEB

Offset	Name	Bytes	Description
0 (0)	DEBTAMID	1	TCAM DEB identifier
1 (1)	DEBTCBAD	3	Address of the TCB for this DEB
4 (4)		1	Reserved
5 (5)	DEBDEBAD	3	Address of the next DEB in the same task
8 (8)		1	Reserved
9 (9)	DEBPCBAD	3	Address of the process control block for this task
12 (C)	DEBTAMOS	2	Offset to the Termname Table entry for the corresponding process entry
14 (E)	DEBSOWA	2	Size of the Locate Mode Work Area
16 (10)	DEBTAMPP	1	Post-pending flag byte
17 (11)	DEBQCBAD	3	Address of the Read-ahead QCB
20 (14)		1	Reserved
21 (15)	DEBTAMWA	3	Address of the TCAM Access Method Work Area
24 (18)		1	Reserved
25 (19)	DEBDCBAD	3	Address of the DCB for this DEB
28 (1C)		1	Reserved
29 (1D)	DEBLCMWA	3	Address of the Locate Mode Work Area
32 (20)	DEBEND	1	End of the DEB indicator
32 (20)	DEBSIZE	1	Size of the DEB in bytes

ACCESS METHOD WORK AREA

The access method work area (IEDQWRKA) is a variable length table that provides intermediate storage fields, pointers to control blocks, switches, and space for a work area. When a DCB in an application program is being opened, the GET/PUT and READ/WRITE Open Executor (IGG01946) allocates main storage for and initializes the access method work area.

The Open Executor puts the address of the work area in the DEBTAMWA field of the Data Extent Block (DEB) for the application program. The address of the DEB is in the DCBDEBAD field in the associated Data Control Block (DCB) in the application program. The DEB address is also in the PEWADEB field of the process entry work area in the MCP so that routines in the MCP can refer to the access method work area by first examining the DEB.

The access method work area is variable in length depending upon whether or not the user specified a SYNAD exit routine. If the user does not specify a SYNAD exit routine, the fullword field GWASTAT/PWASTAT is set to zero (0). If, however, the user does specify such a routine, the field GWASTAT/PWASTAT contains the address of the status indicators. The status indicators are in a fourteen-byte field that is added to the end of the access method work area when required by a SYNAD routine request. There are two status indicators for the SYNAD routine. The first is bit zero of the second byte of the fourteen-byte area. When this bit is set to 1, the command issued is rejected because work units are out of sequence. The second status indicator is bit one of the thirteenth byte. When this bit is set to 1, an incorrect length has been specified, thus creating a work area overflow.

The figure below shows the format of the access method work area; descriptions of the fields follow the illustration.

IEDQWRKA

0 (0)	**PWASAVE** **GWASAVE**	Address of User's Register Save Area
4 (4)	**PWAPEWA**	Address of the Process Entry Work Area
8 (8)	**GWAPEB**	Address of a Part-Empty Buffer
12 (C)	**PWASTART**	Address of the First Byte of Data in the Work Area
	GWAMOVE	Address of the Next Byte in a Buffer to be Moved
16 (10)	**PWACKPT** **GWACKPT**	Address of the User's Checkpoint Routine
20 (14)	**GWAPEWA**	Address of the Next Empty Byte in the User's Work Area
24 (18)	**PWAECB**	PUT/WRITE ECB
	GWAECB	GET/READ ECB
28 (1C)	**PWAELEM** **GWAELEM**	Special AQCTL Element
48 (30)	**PWALIST** **GWALIST**	AQCTL Parameter List
	MOVEAD	Address of the Field to be Moved
52 (34)	**TARGETAD**	Address of Where the Data is to be Moved
56 (38)	**PFLAG**	End-of-List Indicator
57 (39)	**LENGTHAD**	Address of the Length of the Field
60 (3C)	**PWASAVA**	PUT/WRITE Save Area
	GWASAVA	GET/READ Save Area
132 (84)	**PWAFLG**	PUT/WRITE Reader Needed EOM Processed GET/READ
134 (86)		Reserved
136 (88)	**IOBPSAVE**	Address of a Partly Empty Buffer Unit
140 (8C)	**GWASTAT**	Address of GET/READ Status Indicators
	PWASTAT	Address of PUT/WRITE Status Indicators
144 (90)	**PWASOWA** **GWASOWA**	Size of the User's Work Area
146 (92)	**PWACTL**	Work Area Contents Descriptor Byte
147 (93)	**GWARDEL**	Record Delimiter
148 (94)	**GWABUFL**	Size of an MCP Buffer
150 (96)	**PWAOPTCD** **GWAOPTCD**	General Switches
151 (97)	**PWARECFM** **GWARECFM**	General Switches

152 (98) **GWALRECL** Size of a Logical Work Unit	154 (9A) **PWAOFF** Termname Table Offset for Message Destination
156 (9C) **CTLADDR** Address of the Work Area Control Byte	
160 (A0) **GWASCAN** Size of Field to be Scanned	162 (A2) **BUFCNT** Empty-Buffer Counter
164 (A4) **IOBUSZE** Count of Data in a Logical Buffer	166 (A6) **IOBPSZE** Prefix Size Work Area
168 (A8) **IOBSRCE** Termname Table Offset	170 (AA) Reserved
172 (AC) **GWARTVE** Message Retrieval Work Area	

Note: When there are two field names for one field, those field names beginning with P are used when the user is coding in PUT mode, and those field names beginning with G are used when the user is coding in GET mode.

Offset	Name	Bytes	Description
0 (0)	PWASAVE	4	Address of the user register save area
0 (0)	GWASAVE	4	Address of the user register save area
4 (4)	PWAPEWA	4	Address of the process entry work area
8 (8)	GWAPEB	4	Address of a partially empty buffer - the one being used
12 (C)	PWASTART	4	Address of the first byte of data in the user work area
12 (C)	GWAMOVE	4	Address of next byte to be moved in a buffer
16 (10)	PWACKPT	4	Reserved
16 (10)	GWACKPT	4	Reserved
20 (14)	GWAPEWA	4	Address of next empty byte in user work area
24 (18)	PWAECB	4	PUT/WRITE ECB

Offset	Name	Bytes	Description
24 (18)	GWAECB	4	GET/READ ECB
28 (1C)	PWAELEM	20	Special AQCTL element
28 (1C)	GWAELEM	20	Special AQCTL element
48 (30)	PWALIST	4	AQCTL parameter list
48 (30)	GWALIST	4	AQCTL parameter list
48 (30)	MOVEAD	4	Address of the field to be moved
52 (34)	TARGETAD	4	Address of the area into which data is to be moved
56 (38)	PFLAG	1	Indicator of end of parameter list
57 (39)	LENGTHAD	3	Address of the length field of the parameter list
60 (3C)	PWASAVA	72	PUT/WRITE save area
60 (3C)	GWASAVA	72	READ/CHECK save area
132 (84)	PWAFLG	2	X'20' header needed (PUT/WRITE)
	PWAFLG+1		X'80' EOM processed (GET/READ)
136 (88)	IOBPSAVE	4	Address of partially empty buffer unit
140 (8C)	GWASTAT	4	Address of status indicators
140 (8C)	PWASTAT	4	Address of status indicators
144 (90)	PWASOWA	2	Size of user work area
144 (90)	GWASOWA	2	Size of user work area
146 (92)	PWACTL	1	Work area contents descriptor byte - contains a value indicating whether the message in the work area is the first, intermediate, or last segment of the message. The following are the bit settings:

Bits	Value	Meaning
0,1,2,3,7	X'F1'	first segment (header)
0,1,2,3,6	X'F2'	last segment (EOM)
0,1,2,3,6, 7	X'F3'	entire message
1	X'40'	intermediate segment

147 (93)	GWAPDEL	1	End of record for GET/PUT - copied from the process entry
148 (94)	GWABUFL	2	Size of MCP buffer
150 (96)	PWAOPTCD	1	General switch - bit settings as follows:

Name	Bits	Value	Meaning
FIRSTPUT	7	X'01'	first-time switch for locate mode

Offset	Name	Bytes	Description
150 (96)	GWAOPTCD	1	General switch - bit settings as follows:

Name	Bits	Value	Meaning
TNMEFLG	0	X'80'	OPTCD=W (source terminal field)
MSGFLG	1	X'40'	OPTCB=U (message rather than record format)
CTLBYTE	2	X'20'	OPTCD=C (Control byte flag)
EODADFLG	3	X'10'	EODAD exit flag mask
RECDEL	4	X'08'	First-time RECDEL flag
RTVFLG	5	X'04'	Retrieve mode switch mask
PARTBUF	6	X'02'	Partially empty buffer left on main storage QCB
SYNADFLG	7	X'01'	DCBOPTCD bit which, if set, effects exit to SYNAD routine

Offset	Name	Bytes	Description
151 (97)	PWARECFM	1	PUT/WRITE - no bits set
151 (97)	GWARECFM	1	GET/READ - bit settings are as follows:

Name	Bits	Value	Meaning
RETFLG	5	X'04'	Retrieve mode may be entered
INCWA	7	X'01'	Incomplete work area

Offset	Name	Bytes	Description
152 (98)	GWALRECL	2	Size of logical work unit
154 (9A)	PWAOFF	2	PUT Scheduler - Termname Table offset for message destination
156 (9C)	CTLADDR	4	Address of work area control byte; address of PWACTL within the work area
160 (A0)	GWASCAN	2	Size of field to be scanned
162 (A2)	BUFCNT	2	Empty-buffer counter
164 (A4)	IOBUSZE	2	Count of data in a logical buffer - number of bytes in a buffer unit
166 (A6)	IOBPSZE	2	Number of bytes in a buffer - prefix size work area
168 (A8)	IOBSRCE	2	Termname Table offset
170 (AA)		2	Reserved
172 (AC)	GWARTVE	8	Message retrieval work area

PROCESS ENTRY WORK AREA

The process entry work area (IEDQPEWA) is a fixed-length table in the message control program. This work area provides a logical extension of the process entry for the associated application program. The work area also provides storage for the control blocks for the GET and PUT Schedulers. The function of the work area varies depending upon the functions of the GET or PUT Scheduler.

The address of the process entry work area is in the TRMSTAT field of a terminal entry when that entry has been generated by a TPROCESS macro instruction. The address is also in the PWAPEWA field of the access method work area in the associated application program.

When a DCB in an application program is being opened, the OPEN/CLOSE subtask (IEDQEU) allocates main storage for and initializes the process entry work area.

The figure below shows the format of the process entry work area; descriptions of the fields follow the illustration.

IEDQPEWA

0 (0)	PEWARES	Reserved
8 (8)	PEWAISZE	Count of Idles Reserved
12 (C)	PEAQCTL	AQCTL Parameter List
24 (18)	PEWAECBA	Address of the Application Program ECB
28 (1C)	PEWASOWA	Work Area Data Length
29 (1D)	PEWAFLG	General Flag Byte
30 (1E)	PEBFCT	Buffer Limit
32 (20)	PEUNCT	Units Per Buffer
	PEPCBAD	Address of Process Control Block
36 (24)	PERCQCB	Address of the QCB Associated with the ERB Below
40 (28)		Reserved
44 (2C)	PEWALCBA	Address of the LCB
48 (30)		Reserved
52 (34)	PECBUF	Address of First Empty Byte in Current Unit – for PUT Address of the Chain of Read-Ahead Buffers Not Processed by MH – for GET
56 (38)	PEERB	Element Request Block
80 (50)	PEWAELEM	Special Element
96 (60)	PERAQCB	Read-Ahead QCB
108 (6C)	EOMSAVE	Address of the Last EOM for GET
	PEWATIC	Current Unit Address for PUT
112 (70)	PEPSSTCB	Put Scheduler STCB
	PEGSSTCB	Get Scheduler STCB
120 (78)	PEWADEB	Data Extent Block Address
124 (7C)	PEGFSTCB	Get FIFO STCB

132 (84)	**PEWAPROC** Address of the Process Entry
136 (88)	**PESAVE** Register Save Area

Offset	Name	Bytes	Description
0 (0)	PEWARES	8	Reserved
8 (8)	PEWAISZE	4	Count of idle (reserve) characters reserved
12 (C)	PEAQCTL	12	AQCTL parameter list
24 (18)	PEWAECBA	4	Address of the application program ECB
28 (1C)	PEWASOWA	2	Work area data length
30 (1E)	PEWAFLG	1	General flag byte - bit settings are as follows:

For the GET Scheduler:

Name	Bit	Value	Meaning
ERBBUSY	0	X'80'	ERB tposted to the Disk I/O QCB
CFLG	1	X'40'	Closedown in progress
POSTAP	2	X'20'	Need to tpost the application program ERB
FIRSTR	5	X'04'	First-time Retrieve flag
MHOK	6	X'02'	Buffer may be tposted to the message handler
RFLG	7	X'01'	Retrieve mode

Offset	Name	Bytes	Description
31 (1F)	PEBFCT	1	Buffer limit - number of buffers that may be on the Read-ahead QCB at any one time
32 (20)	PEUNCT	1	Number of units per buffer - fixed per process entry
33 (21)	PEPCBAD	3	Address of the Process Control Block
36 (24)	PEPCQCB		Address of the QCB associated with the ERB below
40 (28)		4	Reserved
44 (2C)	PEWALCBA	4	Address of the ICB
48 (30)		4	Reserved
52 (34)	PECBUF	4	If PUT Scheduler - address of the first empty byte in the current unit; GET Scheduler - address of the chain of read ahead buffers not processed by the message handler
56 (38)	PEERB	24	Element request block
80 (50)	PEWAELEM	16	Special element

Offset	Name	Bytes	Description
96 (60)	PERAOCB	12	Read-ahead QCB
108 (6C)	EOMSAVE	4	Address of the last EOM for GET
108 (6C)	PEWATIC	4	Current unit address for PUT
112 (70)	PEPSSTCB	8	PUT Scheduler STCB
112 (70)	PEGSSTCB	8	GET Scheduler STCB
120 (78)	PEWADEB	4	Address of the Data Extent Block
124 (7C)	PEGFSTCB	8	GET FIFO STCB
132 (84)	PEWAPROC	4	Address of the process entry
136 (88)	PESAVE	56	Register save area

OPERATOR CONTROL DATA AREAS

OPERATOR CONTROL ADDRESS VECTOR TABLE

The Operator Control Address Vector Table (IFDQOPC) is a fixed length table that serves as a general work area for the use of operator control. The table is never referred to unless an operator control command is entered. Once such a command is entered, the Operator Control Address Vector Table contains entry points for modules, two save areas, bit switches, pointers, and checkpoint element.

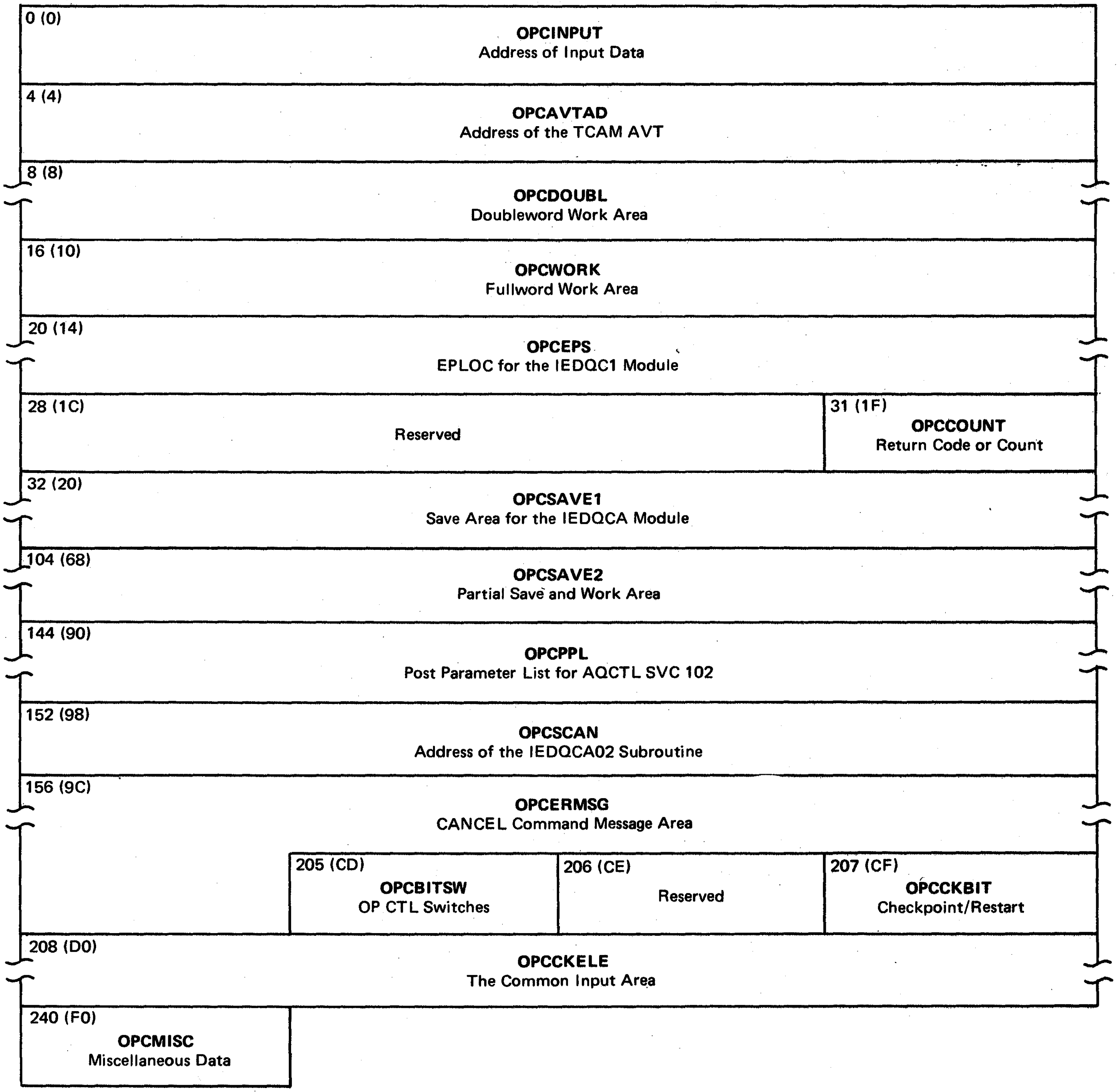
The address of the Operator Control Address Vector Table is the AVTOCGET field of the Address Vector Table.

Because the Operator Control AVT is an attached module, storage is allocated for the table at the time of execution of the INTRO macro. The table is initialized at assembly time.

The Operator Control control module work area is a table of approximately 400 bytes that is attached to the end of the Operator Control AVT at a displacement of X'F1'. This area is not discussed below.

The figure below is the format of the Operator Control Address Vector Table: descriptions of the fields follow the illustration.

IEDQOPC



0 (0) OPCINPUT Address of Input Data

4 (4) OPCAVTAD Address of the TCAM AVT

8 (8) OPCDOUBL Doubleword Work Area

16 (10) OPCWORK Fullword Work Area

20 (14) OPCEPS EPLOC for the IEDQC1 Module

28 (1C) Reserved

31 (1F) OPCCOUNT Return Code or Count

32 (20) OPCSAVE1 Save Area for the IEDQCA Module

104 (68) OPCSAVE2 Partial Save and Work Area

144 (90) OPCPPL Post Parameter List for AQCTL SVC 102

152 (98) OPCSCAN Address of the IEDQCA02 Subroutine

156 (9C) OPCERMSG CANCEL Command Message Area

205 (CD) OPCBITSW OP CTL Switches

206 (CE) Reserved

207 (CF) OPCCKBIT Checkpoint/Restart

208 (D0) OPCCKELE The Common Input Area

240 (F0) OPCMISC Miscellaneous Data

Offset	Name	Bytes	Description
0 (0)	OPCINPUT	4	Address of the input data
4 (4)	OPCAVTAD	4	Address of the TCAM Address Vector Table
8 (8)	OPCDOUBL	8	Doubleword work area
16 (10)	OPCWORK	4	Fullword work area
20 (14)	OPCEPS	8	Entry point for IEDQC1 used by the control module for operator control (IEDQCA)

Offset	Name	Bytes	Description
28 (1C)		3	Reserved
31 (1F)	CPCCOUNT	1	Count or return code
32 (20)	OPCSAVE1	72	IEDQCA save area
104 (68)	OPCSAVE2	40	Partial save area and work area
144 (90)	CPCPPI	8	Tpost parameter list for IGC102
152 (98)	CPCSCAN	4	Address of IEDQCA02 - SCAN subroutine
156 (9C)	CPCERMSG	49	Cancel command message
205 (CD)	OPCBITSW	1	Operator control bit switches:

Name	Bits	Value	Meaning
OPCCONSN	0	X'80'	Source is the system console
OPCMPPN	1	X'40'	Source is an application program
OPCMCPN	2	X'20'	Source is the message control program
OPCPROCN	3	X'10'	Process bit on
OPCPROCF	3 Off	X'EF'	Process bit off
OPCFRSTN	4	X'08'	First-time bit on
OPCFRSTF	4 Off	X'F7'	First-time bit off
OPCLCLN	5	X'04'	Local wait bit on
OPCLCLF	5 Off	X'FB'	Local wait bit off
OPCQUCKN	6	X'02'	Closedown is quick
OPCFLSHF	6 Off	X'FD'	Closedown is flush
OPCTEMPN	7	X'01'	Work bit on
OPCTEMPF	7 Off	X'FE'	Work bit off

Offset	Name	Bytes	Description
206 (CE)		1	One-byte work area referred to only by IEDQCA

207 (CF)	OPCCKPIT		Checkpoint/Restart bits - bit settings for this field are as follows:

Name	Bits	Value	Meaning
OPCRSTN	0	X'80'	Restart in progress
OPCRSTF	0 Off	X'7F'	Mask to specify first restart not in progress
OPCCKPTN	1	X'40'	Checkpoint to be done
OPCCKPTF	1 Off	X'BF'	Mask to specify no checkpoint to be done
OPCINVN	2	X'20'	Checkpoint for invitation lists
OPCINVF	2 Off	X'DF'	Mask to specify no checkpoint for invitation lists

208 (D0)	OPCCKELE	32	Common input block for the operator control routines

The fields in the common input block are represented as follows:

Field	Label	Bytes	Description
OPCED	=OPCCKELE	1	Restart flag
OPCIND	=OPCCKELE+1	1	ID for routine to be loaded
OPCBIT1	=OPCCKELE+2	1	Internal flags
OPCBIT2	=OPCCKELE+3	1	Internal flags
OPCLEN	=OPCCKELE+4	1	Length of relative line number
OPCRLN	=OPCCKELE+5	3	Relative line number
OPCTNME	=OPCCKELE+8	8	Terminal name, ddname, or address
OPCVBCD1	=OPCCKELE+16	1	First verb code
OPCVBCD2	=OPCCKELE+17	1	Second verb code
OPCFLG	=OPCCKELE+18	1	Command dependent flag bits
OPCINFO	=OPCCKELE+19	1	Internal flags
OPCSNSE	=OPCCKELE+20	2	MODIFY interval sense value
OPCOTHR	=OPCCKELE+22	2	MODIFY interval sense count
OPCOPFLD	=OPCCKELE+24	8	Option field name for DISPLAY option and MODIFY option commands

240 (F0)	OPCMISC	1	Data for IEDQCA

COMMAND INPUT BUFFER

The Command Input Buffer (IEZCIB) is a variable length communication parameter list that is used by Operator Control to process a command. The buffer describes the command sent from the console. The CIB shows the command code, the identification of the console that issued the command, and the actual data in the command.

When the INTRO macro instruction is expanded at TCAM execution time, the INTRO macro generates linkage to a module that issues an EXTRACT macro. The FIELDS= parameter specified on the EXTRACT macro is FIELDS=COMM, which calls for the Communication Parameter List. AVTCOMPT is specified as the answer area address on the EXTRACT macro. The Operating System places the address of the Communication Parameter List (Command Input Buffer) in the AVTCOMPT field of the Address Vector Table.

When a command is entered, SVC 34 performs a GETMAIN for the area required by the Command Input Buffer, and the buffer is initialized at that time.

The figure below shows the format of the Command Input Buffer; descriptions of the fields follow the illustration.

IEZCIB

0 (0) **CIBNEXT** Address of the Next CIB in the Queue		
4 (4) **CIBVERB** Code Byte	5 (5) **CIBLEN** Buffer Length	6 (6) Reserved
12 (C) **CIBCONID** Console ID	13 (D) Reserved	14 (E) **CIBDATLN** CIB Data Length
16 (10) **CIBDATA** CIB Data		

Offset	Name	Byte	Description
0 (0)	CIBNEXT	4	Address of the next CIB in the queue (0 for last)
4 (4)	CIBVERB	1	Bits settings for this field are as follows:

Name	Bits	Value	Meaning
CIBSTART	5	X'04'	START command code
CIBMODFY	1,5	X'44'	MODIFY command code
CIBSTOP	1,2,5	X'64'	STOP command code
CIBVARY	2,4	X'28'	VARY command code
CIBHALT	2,3,4,5	X'3C'	HALT command code
CIBDISPL	1,2,4	X'68'	DISPLAY command code
CIBHOLD	1,2,4,5	X'6C'	HOLD command code
CIBRELSE	1,2,3	X'70'	RELEASE command code

Offset	Name	Byte	Description
5 (5)	CIBLEN	1	Length of the buffer (including control fields) in doublewords
6 (6)		6	Reserved
12 (C)	DIBCONID	1	Identifier of the console issuing the command
13 (D)		1	Reserved

Offset	Name	Bytes	Description
14 (E)	CIBDATLN	2	Length of data in the CIB
16 (10)	CIBDATA	n	Beginning of the data from the command operand:

START data - contains the fourth positional parameter, 'PARMVALUE'
MODIFY data - contais the residual operand image following the comma, terminating the first positional parameter
STOP data .- none, CIB generated only to give the console ID to the recipient task
VARY data -,contains the operand field for the command issued
HALT data - contains the operand field for the command issued
DISPLAY data - contains the operand field for the command issued
HOLD data - contains the operand field for the command issued
RELEASE data - contains the operand field for the command issued

CHECKPOINT DATA AREAS

Checkpoint Elements

Environment Checkpoint Request Element:

Defined at AVTCKELE in the AVT

Four words long

Key field - always B'01110000'

Source flag -

B'10000000' - requested by READY

B'01000000' - requested by MCPCLOSE

B'00010000 - requested by the Checkpoint-No incident Records routine

B'00100000' - requested by other routines

Offset			
0	Key X'70'	QCB address	
4	Link address		
8	Source flag	Reserved	Checkpoint time interval
12 (C)	Time of interrupt		Reserved

MH Checkpoint Request Element:

Defined as the ICB

Key field - always B'00000000'

Offset		
0	Key X'00'	Checkpoint QCB address
4	Link address	
60 (5A)	Terminal name offset	

Application Program Checkpoint Request Element:

Defined at PCBWRKA in the PCB - one for each application program

Four words long

Key field - depends on the macro

B'01100000' - requested by CKREQ

B'00010000' - requested by TCHNG

TCHNG

Offset		
0	Key X'10'	Checkpoint QCB address
4	Link address	
8	ECB	
12 (C)	Terminal name offset	Reserved

Offset	CKREQ	
0	Key X'60'	Checkpoint QCB address
4	Link address	
8	ECB	
12 (C)	DEB chain address	

Operator Control Checkpoint Request Element:

Defined at AVTCCELE in the AVT

Two words long

Key field - B'01000000' - requested by VARY, MODIFY, RELEASE, HOLD, ICHNG, MRELEASE, or RELEASEM

Offset		
0	Key X'40'	Checkpoint QCB address
4	Link address	

Checkpoint QCB:

Defined at AVTCKPTP in the AVT

Three words long

Third word always points to the key field of this QCB. The key field is the offset to the Checkpoint STCB

Key field - B'00000010' - tells the TCAM Dispatcher to POST the ECB in the second word and to chain the element at the top of the ready queue off the request element chain in the first word.

Offset		
0	Key X'02'	Request element chain
4	ECB	
8	Address of the STCB code offset	

Checkpoint Work Area

The checkpoint work area is a local constants and variables area that is used by all of the checkpoint routines. This work area contains the checkpoint data set control record, as well as pointers to the other checkpoint records. The checkpoint work area is allocated by a GETMAIN in the Checkpoint Open routine (IGG01941), which also places the address of the work area in the AVTCKGET field of the AVT. During a cold startup, the constant fields in the work area are initialized by the Checkpoint Open routine, the Checkpoint Disk Initialization routine (IGG01942), and the Checkpoint Disk Allocation routine (IGG01949). The variable fields in the checkpoint work area are initialized and changed as required by the checkpoint routines.

IEDQCKPD

0 (0)	**CKPSAVE1**	Save Area for the Load Module
72 (48)	**CKPIOB**	IOB for Checkpoint Disk I/O
	CKPIOFL1	
	CKPIOFL2	
	CKPIOSN0	
	CKPIOSN1	
76 (4C)	**CKPIOECB**	ECB Address
80 (50)	**CKPIOFL3**	
81 (51)	**CKPIOCSW**	Channel Status Word
88 (58)	**CKPIOSIO**	Condition Codes
89 (59)	**CKPIOCPA**	Channel Program Address
92 (5C)		Reserved
93 (5D)	**CKPIODCB**	DCB Address
96 (60)		Reserved
97 (61)	**CKPIORST**	Restart Address
100 (64)	**CKPIOBCI**	Block Count Increment
102 (66)	**CKPIORC**	Error Count
104 (68)	**CKPIOM**	M Seek Address
105 (69)	**CKPIOBB**	BB Seek Address
107 (6B)	**CKPIOCC**	CC Seek Address
		Continued
109 (6D)	**CKPIOHH**	HH Seek Address
111 (6F)	**CKPIOR**	R Seek Address
112 (70)	**CKPECB**	ECB Posted by IOS
116 (74)	**CKPEXCP**	Address of the Current Record Being Written
120 (78)	**CKPCNVRT**	Label Used for the CVD Instruction
	CKPECBL	ECB List for WAIT
	CKPEPLOC	EPLOC for the LOAD Macro
128 (80)	**CKPIOQF**	Address of the First Record On the Checkpoint Disk I/O Queue
132 (84)	**CKPIOQL**	Address of the Last Record On the Checkpoint Disk I/O Queue
136 (88)	**CKPLREB**	Address of the Last Request Element for Which a Record Was Built

140 (8C)	**CKPLDRB**	Address of the Last Disk Record Built
144 (90)	**CKPCTTRB**	Beginning of the CKREQ-TTR Table
148 (94)	**CKPCPARM**	Parameters for the Convert Routine
156 (9C)	**CKPCRLEN**	Control Record Length
157 (9D)	**CKPSWCH1**	Switch for the Checkpoint QCB
158 (9E)	**CKPSWCH2**	Switch for the Checkpointed Invitation List
159 (9F)	**CKPERRCT**	Count Or Read Errors Found by IGG01943
160 (A0)	**CKPCCWS** **CKPSEEKC**	Channel Program Seek Cylinder
168 (A8)	**CKPSCHID**	Search ID
176 (B0)	**CKPTIC**	TIC Command
181 (B5)	**CKPTTRLT**	TTR of the Last Segment Written
184 (B8)	**CKPRW**	Read/Write
192 (C0)	**CKPGETML**	GETMAIN Parameter List
202 (CA)	**CKPWKALN**	Checkpoint Work Area Length
204 (CC)	**CKPMSG** **CKPMSGLN**	Message Buffer Length
206 (CE)		Reserved
208 (D0)	**CKPMSGTX** **CKPSAVE2**	Message Text Save Area
245 (F5)	**CKPMSGTP**	Type of Checkpoint Record
246 (F6)	**CKPRCDSR**	No Segments in One Environment Checkpoint
248 (F8)		
250 (FA)	**CKPTRKSA**	Number of Tracks Available
251 (FB)	**CKPMSGL** **CKPMSGPN**	Message Length Process Entry Name
252 (FC)	**CKPMSGPN**	(Cont.)
265 (109)	**CKPMSGGL**	GETMAIN Length That Was Not Satisfied
268 (10C)	**CKPMSGGL**	(Cont.)
	CKPTRMAD	Terminal Entry Address
272 (110)	**CKPCNTLR**	Beginning of the Checkpoint Control Record

Temporary Use of the Checkpoint Work Area During Checkpoint Open:

116 (74) **CKPCYLNO** Cylinder Number		118 (76) **CKPHEDNO** Head Number
120 (78) **CKPRCDNO** Record Number	121 (79) **CKPKEYLN** Key Length	122 (7A) **CKPDATLN** Data Length
124 (7C) **CKPCTTRC** Current Entry in the CKREQ-TTR Table		
128 (80) **CKPDATIM** Date and Time of the Last Environment Checkpoint		
136 (88) **CKPIPERE** Number of Incident Or CKREQ Records in One Environment Record Segment		138 (8A) Reserved
140 (8C) **CKPCTTRA** Address of the TTR of the Environment Record to be Used for Restart		

Offset	Name	Bytes	Description
0 (0)	CKPSAVE1	72	Save area for the lcad module
72 (48)	CKPIOB	40	IOB for the checkpoint disk I/C operations
72 (48)	CKPIOFL1	1	I/O error flags
73 (49)	CKPIOFL2	1	I/O error flags
74 (4A)	CKPIOSN0	1	
75 (4B)	CKPIOSN1	1	
76 (4C)	CKPIOECB	4	ECB address
80 (50)	CKPIOFL3	1	I/O error flags
81 (51)	CKPIOCSW	7	Channel Status Word
88 (58)	CKPIOSIO	1	Start I/O condition codes

Offset	Name	Bytes	Description
89 (59)	CKPIOCPA	3	Channel program address
92 (5C)		1	Reserved
93 (5D)	CKPIODCB	3	DCB address
96 (60)		1	Reserved
97 (61)	CKPIORST	3	Restart address
100 (64)	CKPIOBCI	2	Block count increment
102 (66)	CKPIOEC	2	Error count
104 (68)	CKPIOM	1	M seek address
105 (69)	CKPIOBB	2	BB seek address
107 (6B)	CKPIOCC	2	CC seek address
109 (6D)	CKPIOHH	2	HH seek address
111 (6F)	CKPIOR	1	R seek address
112 (70)	CKPECB	4	ECB posted by the I/O Supervisor
116 (74)	CKPEXCP	4	Address of the current record being written
116 (74)	CKPCYLNO	2	During checkpoint open, the cylinder number
118 (76)	CKPHEDNO	2	During checkpoint open, the head number
120 (78)	CKPCNVRT	8	Label used for the CVD instruction
120 (78)	CKPECBL	8	ECB list for WAIT
120 (78)	CKPEPLOC	8	EPLOC for the LOAD macro
120 (78)	CKPRCDNO	1	During checkpoint open, the record number
121 (79)	CKPKEYLN	1	During checkpoint open, the key·length.
122 (7A)	CKPDATLN	2	During checkpoint open, the data length
124 (7C)	CKPCTTRC	4	Address of the current entry in the CKPEQ-TTR Table - used for restart open
128 (80)	CKPIOQF	4	Address of the first record on the checkpoint disk I/O queue
128 (80)	CKPDATIM	8	Date and time of the last environment checkpoint, used during checkpoint open
132 (84)	CKPIOQL	4	Address of the last record on the checkpoint disk I/O queue
136 (88)	CKPLRER	4	Address of the last request element for which a checkpoint record was built

Offset	Name	Bytes	Description
136 (88)	CKPIPERE	2	During checkpoint open, the number of incident or CKREQ checkpoints in one environment record segment
140 (8C)	CKPLDRB	4	Address of the last disk record built
140 (8C)	CKPCTTRA	4	During checkpoint open, the address of the TTR of the environment record being used for restart
144 (90)	CKPCTTRB	4	Address of the beginning of the CKREQ-TTR Table
148 (94)	CKPCPARM	8	Parameters for the Convert routine: the address of the DEB and the address for the conversion result
156 (9C)	CKPCRLEN	1	Length of the control record
157 (9D)	CKPSWCH1	1	Switch used for comparing a QCB to see if it has been checkpointed
158 (9E)	CKPSWCH2	1	Switch used for comparing an invitation list to determine whether it has been checkpointed
159 (9F)	CKPERRCT	1	Count of the read errors found by IGG01943
160 (A0)	CKPCCWS	32	Channel program
160 (A0)	CKPSEEKC	8	Seek Cylinder command
168 (A8)	CKPSCHID	8	Search ID command
176 (B0)	CKPTIC	8	TIC command
181 (B5)	CKPTTRLT	3	TTR of the last environment segment written
184 (B8)	CKPRW	8	Read/Write command
	CKPREAD		Read Data CCW
	CKPWRITE		Write Data CCW
	CKPWCKD		Write Count, Key, and Data CCW
192 (C0)	CKPGETML	10	GETMAIN parameter list
202 (CA)	CKPWKALN	2	Length of the checkpoint work area
204 (CC)	CKPMSG		Message buffer used for WTO
204 (CC)	CKPMSGLN	2	Length of the message buffer
206 (CE)		2	Reserved
208 (D0)	CKPMSGTX	37	Message text
208 (D0)	CKPSAVE2	15	Temporary storage area
245 (F5)	CKPMSGTP	20	Type of checkpoint record

Offset	Name	Bytes	Description
246 (F6)	CKPRCDSR	2	Number of segments in one environment checkpoint
248 (F8)	CKPCKDLT	2	Reserved
250 (FA)	CKPTRKSA	2	Number of tracks available in the checkpoint data set
251 (FB)	CKPMSGL	4	Message length
251 (FB)	CKPMSGPN	4	Process entry name
265 (109)	CKPMSGGL	4	GETMAIN length that could not be satisfied
268 (10C)	CKPTRMAD	4	Terminal entry address
272 (110)	CKPCNTLR		Beginning of the Checkpoint Control Record

Checkpoint Disk Records

Checkpoint Control Record: The checkpoint control record is written on disk from the area starting at CKPCNTLR in the checkpoint work area each time that an environment checkpoint record is written.

Offset

0	1	2	3	4	5	8
Flag byte	Index to the current environment record	Number of incident records	Number of available incident records	TTR of the last CKREQ record on first CKREQ records track	TTR of the first CKREQ record	TTR of the last incident record on the first incident records track
CKPFLAGS	CKPTTRCT	CKPINCNT	CKPINCNO	CKPCRRNO	CKPTTRCR	CKPINRNO

Offset

9	12 (C)	14 (E)	15 (F)	16 (10)	17 (11)
TTR of the first incident record	Number of bytes in an environment record segment	Value of the INTRO operand CKREQS	Value of the INTRO operand CPRCDS	Number of incident records per track	Number of CKREQ records per track
CKPTTRIN	CKPBPERR	CKPCKRQS	CKPCPRCD	CKPIPERT	CKPCPERT

18 (12)	20 (14)	21 (15)	24 (18)	26 (1A)	29 (1D)	
Length of a CKREQ record	Number of environment record segments per track	TTR of the last incident record written	Length of an incident record	TTR of the first environment record	TTR of the second environment record	TTR of the last environment record
CKPCKRLN	CKPRPERT	CKPTTRLI	CKPINCLN	CKPTTRT1		

Offset	Name	Bytes	Description	Initialized By	Altered By
0 (0)	CKPFLAGS	1	Flag byte:		
			X'80'-normal closedown	Set by IGG01943 Turned off by IGG01944	
			X'10' - Open CKREQ		IGG01944
			X'20' - Open incident		IGG01944
			X'40' - Open environment		IGG01943
			X'08' - No environment records are available		IGG01943
			X'04' - Value of startup parameter that indicates whether invitation lists are to be checkpointed	Set by IGG01949	IGG01943
			X'02' - OS synchronous checkpoint	IGG01949	IGG01949
			X'01' - Operator control incident records are present		
1 (1)	CKPTTRCT	1	Index to the current environment checkpoint record	IGG01942 initializes this field to 1	IEDQNP changes this field after each environment checkpoint
2 (2)	CKPINCNT	1	Total number of incident records in the data set	IGG01949	
3 (3)	CKPINCNO	1	Number of incident records that are available for use	Cold start- IGG01949 Warm start- IGG01941	IEDQNG,IEDQNH, IEDQNI,IEDQNJ, IEDQNO
4 (4)	CKPCRRNO	1	TTR of the last CKREQ record on the first track that contains CKREQ records	IGG01942	
5 (5)	CKPTTRCR	3	TTR of the first CKREQ record	IGG01942	
8 (8)	CKPINRNO	1	TTR of the last incident record on the first track that contains incident records	IGGC1942	
9 (9)	CKPTTRIN	3	TTR of the first incident record	IGG01942	
12 (C)	CKPBPERR	2	Number of bytes in each environment record segment	IGG01949	
14 (E)	CKPRKRQS	1	Value of CKREQS (from INTRO) for the last startup - used at restart time instead of the corresponding value in the AVT	Cold start - IGG01942 Warm start - IGG01944	

15 (F)	CKPCPRCD	1	Value of CPRCDS (from INTRO) for last startup - used at restart time instead of the corresponding value in the AVT	Cold start - IGG01942 Warm start - IGG01943	
16 (10)	CKPIRERT	1	Number of incident records per track	IGG01949	
17 (11)	CKPPRQNO	1	Maximum number of priority QCBs used by an OS synchronous process entry	IGG01949	
	CKPCRERT		Number of CKREQ records per track (overlays CKPPRQNO)	IGG01949	
18 (12)	CKPCKRLN	2	Length of a CKREQ record, depends on the number of option fields	IGG01949	
20 (14)	CKPRPERT	1	Number of environment record segments per track	IGG01949	
21 (15)	CKPTTRLI	3	TTR of the last incident record written	IGG01941	IEDQNP
24 (18)	CKPINCLN	2	Length of an incident record	IGG01949	
26 (1A)	CKPTTRT1	3	TTR of the first environment record	IGG01942	IEDQNP, IGG01943

There are as many three-byte TTR fields for environment checkpoint records as there are records indicated in CKPCPRCD.

Environment Checkpoint Record Segment: Main storage in which to build an environment checkpoint record segment is obtained by the Environment Checkpoint routine (IEDQNK) each time that an environment checkpoint is requested. The format and length of an environment checkpoint vary according to Option Table and Terminal Table entries. The entire Option Table is included in the environment record, and there is one section of data for each single, group, line, and process entry of the Terminal Table in the record.

Offset	Name	Bytes	Description	Initialized By
0 (0)	CDRDATE	4	Date of the checkpoint	IEDQNP
4 (4)	CDRTIME	4	Time that the record is written	IEDQNP
8 (8)	CKRKEY	1	Key byte: X'1C' - last segment of an environment checkpoint record X'20' - a segment that is not the last segment of an environment checkpoint record	IEDQNK

Offset	Name	Bytes	Description	Initialized By	
9 (9)	CKPTTRLI	3	TTR of the last incident record written	IEDQNP	
12 (C)		2	Termname Table offset to the primary operator control terminal - from the AVT field AVTOPCON	IEDQNK	
14 (E)		2	Number of seconds in a system delay - from the AVT field AVTINTLV	IEDQNK	
16 (10)		1	TCAM status byte - from the AVT field AVTBIT1	IEDQNK	
17 (11)		1	TCAM status byte - from the AVT field AVTBIT2	IEDQNK	
18 (12)		4	Nonreusable disk relative record address - from the AVT field AVTNADDR	IEDQNK	
22 (16)		4	Reusable disk relative record address - from the AVT field AVTRADDR	IEDQNK	
24 (1A)		n	Option Table - the address of the Option Table is in AVTOPTPT and the first word of the table contains the address of the end of the table; the address of the end less the address of the beginning equals the length	IEDQNK	
26 (1A)	CKPTTRT1	3	TTR of the first environment record	IGG01942	IEDQNP, IGG01943

There are as many three-byte TTR fields for environment checkpoint records as there are records indicated in CKPCPRCD.

Environment Checkpoint Record Segment: Main storage in which to build an environment checkpoint record segment is obtained by the Environment Checkpoint routine (IEDQNK) each time that an environment checkpoint is requested. The format and length of an environment checkpoint vary according to Option Table and Terminal Table entries. The entire Option Table is included in the environment record, and there is one section of data for each single, group, line, and process entry of the Terminal Table in the record.

Offset	Name	Bytes	Description	Initialized By
0 (0)	CDRDATE	4	Date of the checkpoint	IEDQNP
4 (4)	CDRTIME	4	Time that the record is written	IEDQNP
8 (8)	CDRKEY	1	Key byte: X'1C' - last segment of an environment checkpoint record X'20' - a segment that is not the last segment of an environment checkpoint record	IEDQNK
9 (9)	CDRTTRLI	3	TTR of the last incident record written	IEDQNP
12 (C)		2	Termname Table offset tc the primary operator control terminal - from the AVT field AVTOPCON	IEDQNK
14 (E)		2	Number of seconds in a system delay - from the AVT field AVTINTLV	IEDQNK
16 (10)		1	TCAM status byte - from the AVT field AVTBIT1	IEDQNK
17 (11)		1	TCAM status byte - from the AVT field AVTBIT2	IEDQNK
18 (12)		4	Nonreusable disk relative record address - from the AVT field AVTNADDR	IEDQNK
22 (16)		4	Reusable disk relative record address - from the AVT field AVTRADDR	IEDQNK
24 (1A)		n	Option Table - the address of the Option Table is in AVTOPTPT and the first word of the table contains the address of the end of the table; the address of the end less the address of the beginning equals the length	IEDQNK

26+n This is the point at which the checkpointed fields from the Terminal Table start. Only single, group, line, and process entries are checkpcinted, and different fields are included under different conditions. These conditicns are stated as each item is described. Each entry is checkpoint as follows:

Bytes	Description	Initialized By
1	Terminal entry status byte (from TRMSTATE) included only for a single, group, or line entry	IEDQNK
2	Input sequence number (from TRMINSEQ) included only for a single, group, line, or process entry that is disk queued	IEDQNK
2	Output sequence number (from TRMOUTSQ) included only for a single. group, line, or process entry that is disk queued	IEDQNK

2	Count of messages for this destination (from QCBMSGCT in the QCB referred to by TRMDESTQ) included for any single, group, line, or process entry that has not had its QCB checkpointed	IEDQNK
3	Queue-back message chain pointer (from QCBQBACK) included for any single, group, line, or process entry that has not had its QCB checkpointed	IEDQNK
21	Disk pointers from QCBDNHDR through QCBIFEFC in a priority level QCB that is attached to this Destination QCB; there is one of these 21-byte entries for each priority level QCB attached to a Destination QCB that is being checkpointed	IEDQNK
3	LCBSTAT1, LCBSTAT2, DCBINTVL for any single, group, or line entry	IEDQNK
n	Invitation list for any single, line, or group entry that has not had its Destination QCB checkpointed; QCBDCBAD points to the DCB, and DCBINVLI points to the invitation list: the length of the list is equal to the number of entries times the length of each entry plus 8 control bytes	IEDQNK

In summary, the general format of an environment checkpoint record is as follows:

Offset

0	4	8	9								
Date	Time	Key X'20'	AVT Fields	Terminal Table data for the first entry	Option fields for the first entry	QCB data for the first entry	LCB data for the first entry	DCB data for the first entry	Invitation List for the first entry	Terminal Table data for the second entry	Option fields for the second entry

0	8	9									
	Key X'1C'	QCB data for the second entry	LCB data for the second entry	DCB data for the second entry	Invitation List for the second entry	Terminal Table data for the last entry	Option fields for the last entry	QCB data for the last entry	LCB data for the last entry	DCB data for the last entry	Invitation List for the last entry

Incident Checkpoint Record for the CHECKPT Macro: The Build Incident Record for MH routine (IEDQNG) issues a GETMAIN for main storage in which to build this incident checkpoint record and places the address of this area at CKPLDRB in the checkpoint work area. If the CHECKPT macro is issued in the incoming group of MH, the terminal that sent the current buffer is checkpointed. If the CHECKPT macro is issued in the outgoing group of MH, the terminal that is to receive the current message is checkpointed. The length of this record depends on which Option Table fields are used for the terminal being checkpointed. The Incident Record for MH routine uses the LCB field LCBTTCIN, the offset to the current Termname Table entry, as input to the Termname Table code (IEDQTNT) to get the correct terminal entry address. The terminal entry field TRMOPTBL is an offset to the beginning of the Option Table fields for this terminal. The routine adds the Option Table offset from the terminal entry to the Option Table address (from AVTOPTPT in the AVT) to refer to the beginning of the Option Table data for this terminal and uses the individual option entry offsets in the terminal entry to refer to the specific option data entries for this terminal. The second word of the Option Table contains the address of the Option Characteristics Table, each entry of which corresponds in consecutive order to each option entry offset in a terminal entry. If the Build Incident Record for MH routine finds that a halfword option entry offset in the terminal entry does not contain X'FF', the routine gets the address of the option data by adding the halfword option entry offset to the beginning of the option data for this terminal to get the beginning of this data field, gets the length of this option data field for the corresponding Option Characteristics Table entry, and moves the option data to the next available location in the incident checkpoint record.

Offset	Name	Bytes	Description	Initialized By
0 (0)	CDRDATE	4	Date of the checkpoint	IEDQNP
4 (4)	CDRTIME	4	Time that the record is written	IEDQNP
8 (8)	CDRKEY	1	Key byte: D'00' - CHECKPT record	IEDQNG
9 (9)		2	The offset to the terminal that is currently connected on the line of the LCB that is the request element - from LCBTTCIN	IEDQNG
11 (B)		1	The terminal status - from TRMSTATE	IEDQNG
12 (C)			Beginning of the option fields defined for the terminal referred to by the offset in bytes 9-10. The manner in which IEDQNG checkpoints these option fields is described in the writeup that precedes this record layout.	IEDQNG

In summary, the general format of an incident checkpoint record for the CHECKPT macro is as follows:

Offset

0	4	8	9	11 (B)	12 (C)
Date	Time	Key D'00'	Terminal offset	Terminal Status	Option data fields

Incident Checkpoint for Operator Control: The Build Incident Checkpoint for Operator Control routine (IEDQNJ) issues a GETMAIN for main storage in which to build this incident checkpoint record and places the address of this area at CKPLDRB in the checkpoint work area. This routine initializes this checkpoint record from the operator control checkpoint element at OPCCKELE in the Operator Control AVT.

Offset	Name	Bytes	Description	Initialized By
0 (0)	CDRDATE	4	Date of the checkpoint	IEDQNP
4 (4)	CDRTIME	4	Time that the record is written	IEDQNP
8 (8)	CDRKEY	1	Key byte: D'16' - Operator Control record	IEDQNJ
9 (9)		3	Reserved	
12 (C)		32	Operator Control checkpcint element from CPCCKELE in the Operator Control	IEDQNJ

AVT

In summary, the format of an incident checkpoint record for operator control is as follows:

Offset

0	4	8	9	12 (C)
Date	Time	Key D'16'	Reserved	Operator Control Checkpoint Element

Incident Checkpoint for the TCHNG Macro: The Build Incident Checkpoint for TCHNG routine (IEDQNH) issues a GETMAIN for main storage in which to build this incident checkpoint record and places the address of this area at CKPLDRB in the checkpoint work area. The checkpoint of the option data fields is handled exactly as explained in the Incident Checkpoint for the CHECKPT Macro discussicn.

Offset	Name	Bytes	Description	Initialized By
0 (0)	CDRDATE	4	Date of the checkpoint	IEDQNP
4 (4)	CDRTIME	4	Time that the record is written	IEDQNP
8 (8)	CDRKEY	1	Key byte: D'04' = TCHNG record	IEDQNH
9 (9)		2	Offset tc the Termname Table entry for the terminal being checkpcinted - frcm bytes 12-13 of the checkpoint request element	IEDQNH
11 (B)		1	Terminal entry status byte - from TRMSTATE	IEDQNH
12 (C)			Beginning of the option fields defined for the terminal referred to by the offset in bytes 9-10.	IEDQNH

In summary, the general format of an incident checkpoint for TCHNG record is as follows:

Offset

0	4	8	9	11 (B)	12 (C)
Date	Time	Key D'04'	Terminal offset	Terminal status byte	Option data fields

<u>CKREQ Checkpoint Record</u>: The Build CKREQ Disk Record routine (IEDQNM) issues a GETMAIN macro for main storage in which to build this CKREQ checkpoint record and places a pointer to this area in the CKPLDRB field of the checkpoint work area. The format and length of this checkpoint record depends upon the number of priority QCBs associated with the Destination QCB that is being checkpointed; there is one 21-byte area of QCB disk pointers for each priority level. The checkpoint of the option data fields is handled exactly as explained in the Incident Checkpoint for the CHECKPT Macro discussion. The CKREQ record DSECT is IEDQCDRD.

Offset	Name	Bytes	Description	Initialized By
0 (0)	CDRCKFLG	1	Flag bits: Bit 0 - On - CKREQ is not complete Off- CKREQ is complete	 IEDQNM
1 (1)		3	Link address of the Checkpoint Disk I/O queue - from CKPIOQF and CKPIOQL in the checkpoint work area	IEDQNM
4 (4)	CDRCKIN	2	Input sequence number - from TRMINSEQ in the terminal entry that is referred to by the offset at CDRCKOFF	IEDQNM
6 (6)	CDRCKOUT	2	Output sequence number - from TRMOUTSQ in the terminal entry that is referred to by the offset at CDRCKOFF	IEDQNM
10 (A)	CDRCKOFF	2	Termname Table offset - from DEBTAMOS in the associated DEB	IEDQNM
12 (C)	CDRCKMSG	2	QCB message count - from QCBMSGCT in the Destination QCB	IEDQNM
14 (E)	CDRCKQBC	3	Queue-back chain pointer - from QCBQBACK in the Destination QCB	IEDQNM
17 (11)	CDRCKQCB	21	Priority QCB disk pointers - from the first 21 bytes of the priority level QCB: QCBDNDHR - disk record number for the next first unit of a message received QCBFHDIZ - disk record number of the first unit of the first message in the last zone used for this queue QCBFHDTZ - disk record number of the first unit of the first message for this queue in the current zone QCBINTFF - disk record number of the first held message in FEFO order QCBINTLF - disk record number of the last held message in FEFO order QCBFFEFO - disk record number of the first message received in FEFO order QCBIFEFO - disk record number of the last message received in FEFO order	IEDQNM

17 + (21 X n) where n is the number of priority level QCBs	CDRCKOPT	Beginning of the option fields defined for the terminal referred to by CDRCKOFF	IEDQNM

In summary, the general format of a CKREQ checkpoint record is as follows:

0	1	4	6	8	9	10 (A)	12 (C)	14 (E)
Flag	Link address	Input sequence number	Output sequence number	Key D'18'	Unused	Terminal name offset	QCB message count	Queue-back chain pointer

17 (11)			17 + (21 x n)
Priority QCB disk pointers for the first priority level		Priority QCB disk pointers for the last priority level	Option data fields

PARAMETER LISTS FOR THE MESSAGE HANDLING MACRO EXPANSIONS

The following are the formats of the input parameter lists for the routines called by the message handling macro expansions through the User Interface routine (IEDQUI). The User Interface routine uses the value in the first byte as an index into a VCON table at AVTMSGS in the AVT. Each entry in the VCON table is an address of a message handling routine. If a macro is not listed, its expansion does not use a parameter list.

MACRO		PARAMETER LIST 0	1	2	3	ROUTINE CALLED
CANCELMG		Index to IEDQAR and Bits	Parameter List Length and Logical	X'00'	Mask	IEDQAR
	4	Mask				

Bits 6 Recall Necessary
7 Unconditional Mask

Logical 7 AND

		0	1	2	3	
CHECKPT		Index to IEDQBB	Parameter List Length - X'04'	X'00'	X'00'	IEDQBB

		0	1	2	3	
CODE		Index to IEDQAW	Parameter List Length	X'00'	Status	IEDQAW
	4	Address of the Translation Table				

Status		
	X'80'	Use the Translation Table address in the DCB
	X'40'	A nonstandard Translation Table
	X'20'	Entry from INBUF or OUTBUF
	X'10'	Entry from INMSG or OUTMSG
	X'00'	A standard Translation Table

		0	1	2	3	
		Index to IEDQAI	Parameter List Length - X'04'	Register 15 Offset	Variable Length indicator	IEDQAI
	4	Blank Character	Address of Characters			IEDQAQ

There is no input parameter list for this routine.

		0	1	2	3	
COUNTER		Index to IEDQAE	Parameter List Length - X'04'	Option Field, Offset	Register 15 Offset	IEDQAE

MACRO | PARAMETER LIST | ROUTINE CALLED

CUTOFF

0	1	2	ROUTINE CALLED
Index to IEDQAU	Parameter List Length - X'04'	Requested Cutoff Length	IEDQAU

DATETIME

0	1	ROUTINE CALLED
Index to IEDQAF and Bit	Data Type Flag	IEDQAF

Bit 6 ON — To request the expand buffer function

0	1	2	ROUTINE CALLED
Index to IEDQAC	Parameter List Length	Count of Bytes to be Inserted	IEDQAC

ERRORMSG

	0	1	2	3	ROUTINE CALLED
	Index to IEDQAZ and Bits	Parameter List Length and Logical	Status	Mask	IEDQAZ
4	Mask				
8	Destination Status	Variable Data			
12	Index to IEDQAT	Parameter List Length	Index to IEDQAF	Index to IEDQAO	IEDQAT
16	Count	Address of the Message			
20	Address of the Exit Routine				

Bits 6 Recall is necessary
7 Unconditional Mask

Logical 7 AND

Status X'01' indicates that the IEDQAT parameter list follows

Destination Status and Variable Data
C'S' + AL3 (0) — send to the source
C'D' + AL3 (0) — send to the destination
C'N' + AL3 (destination name) — send to the named destination
C'O' + I(AE) + AL1 (opfield) + AL1 (16) — send to the destination named in the option field

MACRO	PARAMETER LIST	0	1	2	3	ROUTINE CALLED
FORWARD		Index to IEDQA5	Parameter List Length	Status	Index to IEDQBA	IEDQA5
	4	EOA String Length	Address of EOA String			
	8	Address of the Exit Routine				
	12	Variable Data				
	16	Index to IEDQA1	Parameter List Length	X'00'	Length	IEDQA1
	20	Address of the Character String				

Status 0 Destination = Name
1 Destination = Option Field
2 Destination in Buffer
3 Exit specified
4 EOA specified

Variable Data
Index to IEDQAE, Length, Option Offset, X'16' or
Index to IEDQAI, Length, X'16', Address Field Length

		0	1	2	3	
		Index to IEDQAJ	Parameter List Length - X'08'	Length	Register 15 Offset	IEDQAJ
	4	X'40'	Address of the EOA String			

There is no input parameter list for this routine IEDQAV

		0	1	2	3	
HOLD		Index to IEDQAS and Bits	Parameter List Length and Logical	X'00'	Mask	IEDQAS
	4	Mask				
	8	Interval		X'00'	X'00'	

Bits 6 Recall is necessary
7 Unconditional mask

Logical 7 AND

MACRO	PARAMETER LIST 0	1	2	3	ROUTINE CALLED
INBUF	Index to IEDQAE	Parameter List Length - X'04'	Option Field Offset	Register 15 Offset	IEDQAE

	0	1			
INEND	Index to IEDQA4	Parameter List Length - X'02'			IEDQA4

	0	1	2	3	
	Index to IEDQAK	Parameter List Length - X'04'	Index to IEDQAF	Index to IEDQAO	IEDQAO

	0	1	2	3	
INHDR	Index to IEDQAE	Parameter List Length - X'04'	Option Field Offset	Register 15 Offset	IEDQAE

	0	1	2	3	
INITIATE	Index to IEDQAI Bit	Parameter List Length - X'08'	Register 15 Offset	Length	IEDQAI
4	Blank Character	Address of Characters			

Bit 7 No Blank Character

	0	1	2	3	
INMSG	Index to IEDQAE	Parameter List Length - X'04'	Option Field Offset	Register 15 Offset	IEDQAE

	0	1	2	3	
	Index to IEDQAK	Parameter List Length - X'04'	Index to IEDQAF	Index to IEDQAO	IEDQAK

	0	1			
	Index to IEDQA4	Parameter List Length - X'02'			IEDQA4

MACRO	PARAMETER LIST	0	1	2	3	ROUTINE CALLED
LOCK		Index to IEDQAI and Bit	Parameter List Length - X'08'	Register 15 Offset	Variable Length	IEDQAI
	4	Blank	Address of Characters			

Bit 7 No Blank Characters

MACRO		0	1	2	3	ROUTINE CALLED
LOCOPT		Index to IEDQAE	Parameter List Length - X'04'	Option Field Offset	Register Offset	IEDQAE

MACRO		0	1	2	3	ROUTINE CALLED
LOG		Index to IEDQBY and Bits	Parameter List Length - X'08'	X'00'	X'00'	IEDQBY
	4		Address of LOGTYPE Entry			

MACRO		0	1			ROUTINE CALLED
		Index to IEDQBX	Address of DCB			IEDQBX

PARAMETER LIST

MACRO		0	1	2	3	ROUTINE CALLED
MSGEDIT		Index to IEDQAN	Parameter List Length	Index to IEDQAF	Index to IEDQAO	IEDQAN
	4	Index to IEDQAJ	Blank Character	Number of Entries	Reserved	
	8	Reserved	Address of the Characters Table			
	12	Key	Status	Data Description		
	16	"FROM" Delimiter Description		"TO" Delimiter Description		
	20	A Total of 31 Entries				

Key

Status		
	0	Data = Characters
	1	Data = Idles (Reserve Characters)
	2	Data = CONTRACT
	3	TO = Character String
	4	TO = Offset or SCAN
	5	TO = Count
	6	Inclusive FROM
	7	Inclusive TO

Data = REPLACE, TO = Character String:

	0	1	2	3	
	Index to IEDQAP and Bit	Status	Index to IEDQAF	Index to IEDQAO	IEDQAP
4	Insert Data				
8	Index to IEDQAJ	Parameter List Length	X'00'	Register 15 Offset	
12	Blank Character (optional)	Character String Address			
16	"AT" Offset (optional)				

Bit 7 ON—Remove at Scan Pointer
OFF—Remove at Specified Offset

Status See MSGEDIT parameter list for IEDQAN

Insert Data

Characters: Length of Character String (1 byte), Address of Character String (3 bytes)

Idles: Number of Idles (1 byte), Idle Character (1 byte), X'0000'

MACRO

PARAMETER LIST

ROUTINE CALLED

MSGEDIT (Cont.)

Data = REPLACE, TO = Extent or Offset:

	0	1	2	3
	Index to IEDQAP and Bit	Status	Index to IEDQAF	Index to IEDQAO
4	Insert Data			
8	"TO" Extent or Offset		"AT" Offset (optional)	

IEDQAP

Bit See first MSGEDIT parameter list for IEDQAP

Status See MSGEDIT parameter list for IEDQAN

Insert Data
See first MSGEDIT parameter list for IEDQAP

Data = CONTRACT, TO = Character String:

	0	1	2	3
	Index to IEDQAP and Bit	Status	Index to IEDQAF	Index to IEDQAO
4	Index to IEDQAJ	X'08'	X'00'	Register 15 Offset
8	Blank Character (optional)	Character String Address		
12	"AT" Offset (optional)			

IEDQAP

Bit See first MSGEDIT parameter list for IEDQAP

Status See MSGEDIT parameter list for IEDQAN

Data = CONTRACT, TO = Extent or Offset:

	0	1	2	3
	Index to IEDQAP and Bit	Status	Index to IEDQAF	Index to IEDQAO
4	"TO" Extent or Offset		"AT" Offset (optional)	

IEDQAP

Bit See first MSGEDIT parameter list for IEDQAP

Status See parameter list for IEDQAN

PARAMETER LIST

MACRO

ROUTINE CALLED

MSGEDIT (Cont.)

Offset specified in MSGEDIT Macro:

	0	1	2	3	ROUTINE CALLED
	Index to IEDQA2 and Bit	Parameter List Length - X'0A'	Index to IEDQAF	Index to IEDQAO	IEDQA2
4	Insert Data				
8	"AT" Offset				

Bit 7 ON—Data = Idles
OFF—Data = Characters

Insert Data
See first MSGEDIT parameter list for IEDQAP

Insert at Scan Pointer:

	0	1	2	3	ROUTINE CALLED
	Index to IEDQA2 and Bit	Parameter List Length - X'08'	Index to IEDQAF	Index to IEDQAO	IEDQA2
4	Insert Data				

Bit 7 ON—Data = Idles
OFF—Data = Characters

Insert Data
See first MSGEDIT parameter list for IEDQAP

	0	1	2	3	ROUTINE CALLED
	Index to IEDQA8 and Bit	Parameter List Length - X'0C'	Index to IEDQAF	Index to IEDQAO	IEDQA8
4	Insert Data				
8	Extent between Inserts		Index to IEDQAE	Option Field Offset	

Bit 7 ON—Data = Idles
OFF—Data = Characters

Insert Data
See first MSGEDIT parameter list for IEDQAP

PARAMETER LIST

MACRO		0	1	2	3	ROUTINE CALLED
MSGGEN		Index to IEDQBL	Parameter List Length	Status	Mask	IEDQBL
	4		Mask			
	8	Address of Message				
	12	Address of Table				

Status X'80' Use Table in DCB
X'00' Use Table Specified

MACRO	0	1	2	3	ROUTINE CALLED
MSGLIMIT	Index to IEDQAE	Parameter List Length	Option Field Offset	Register 15 Offset	IEDQAE
	Reserved	Limit			IEDQAG

MACRO					ROUTINE CALLED
MSGTYPE	Index to IEDQAI and Bit	Parameter List Length	Register 15 Offset	Length	IEDQAI
	Blank Character		Address of Characters		

Bit 7 No Blank Character

MACRO					ROUTINE CALLED
ORIGIN	Index to IEDQAI and Bit	Parameter List Length	X'00'	Length	IEDQAI
	Blank Characters	Address of Characters			

Bit 7 No Blank Character

MACRO		PARAMETER LIST 0	1	2	3	ROUTINE CALLED
ORIGIN (Cont.)		Index to IEDQA1	Parameter List Length - X'08'	X'00'	X'00'	IEDQA1
	4	Address of Characters (AVTDOUBL)				
		There is no input parameter list for this routine.				IEDQAM
OUTBUF		Index to IEDQAE	Parameter List Length - X'04'	Option Field Offset	Register 15 Offset	IEDQAE
OUTEND		Index to IEDQAK	Parameter List Length - X'04'	Index to IEDQAF	Index to IEDQAO	IEDQAK
		Index to IEDQA4	Parameter List Length - X'02'			IEDQA4
OUTHDR		Index to IEDQAE	Parameter List Length - X'04'	Option Field Offset	Register 15 Offset	IEDQAE
OUTMSG		Index to IEDQAE	Parameter List Length - X'04'	Option Field Offset	Register 15 Offset	IEDQAE
		Index to IEDQAK	Parameter List Length - X'04'	Index to IEDQAF	Index to IEDQAO	IEDQAK
		Index to IEDQA4	Parameter List Length - X'02'			IEDQA4

MACRO	PARAMETER LIST					ROUTINE CALLED
		0	1	2	3	
PATH		Index to IEDQAI and Bit	Parameter List Length - X'08'	Register 15 Offset	Variable Length	IEDQAI
	4	Blank Character	Address of Characters			

Bit 7 No Blank Character

	0	1	2	3	ROUTINE CALLED
	Index to IEDQAE	Parameter List Length - X'04'	Option Field Offset	Register 15 Offset	IEDQAE

MACRO		0	1	2	3	ROUTINE CALLED
PRIORITY		Index to IEDQAI and Bit	Parameter List Length - X'08'	Register 15 Offset	Length	IEDQAI
	4	Blank Character	Address of Characters			

Bit 7 No Blank Character

MACRO		0	1	2	3	ROUTINE CALLED
REDIRECT		Index to IEDQAT and Bits	Parameter List Length and Logical	Status	Mask	IEDQAT
	4	Mask				
	8	Destination Status	Variable Data			

Bits 6 Recall is necessary
7 Unconditional Mask

Logical 7 AND

Status X'01' indicates that the IEDQAT parameter list follows

Destination Status and Variable Data
C'S' + AL3 (0) — send to the source
C'D' + AL3 (0) — send to the destination
C'N' + AL3 (destination name) — send to the named destination
C'O' + Index to IEDQAE + AL1 (option offset) — send to the destination named in the option field

MACRO	PARAMETER LIST					ROUTINE CALLED
		0	1	2	3	
SCREEN		Index to IEDQAI and Bit	Parameter List Length - X'08'	Register 15 Offset	Variable Length	IEDQAI
	4	Blank Character	Address of Characters			

Bit 7 No Blank Character

	0	1	ROUTINE CALLED
	Index to IEDQAY and Bit	Request Code	IEDQAY

Bit 7 ON—indicates that the user specified SCREEN = WDC, WLA, or WRE
OFF—indicates that the user specified SCREEN with any other operand or no operand

Request Code
X'00' SCREEN = WDC, no operand, or not WLA or WRE
X'01' SCREEN = WLA
X'02' SCREEN = WRE

MACRO		0	1	2	3	ROUTINE CALLED
SEQUENCE		Index to IEDQAI and Bit	Parameter List Length - X'08'	X'00'	Variable Length Indicator	IEDQAI
	4	Blank Character	Address of Characters			

Bit 7 No Blank Character

	0	1	ROUTINE CALLED
	Index to IEDQAH	Parameter List Length - X'02'	IEDQAH

	0	1	ROUTINE CALLED
	Index to IEDQAD	Index to IEDQAF	IEDQAD

MACRO		0	1	2	3	ROUTINE CALLED
SETEOF		Index to IEDQAI and Bit	Parameter List Length - X'08'	Register 15 Offset	Variable Length	IEDQAI
	4	Blank Character	Address of Characters			

Bit 7 No Blank Character

MACRO — PARAMETER LIST — ROUTINE CALLED

SETSCAN

	0	1	2	3
	Index to IEDQAJ and Bit	Parameter List Length - X'08'	Status	Register Offset
4	Blank Character	Address of Characters		

IEDQAJ

Bit 7 No Blank Character

Status X'00' Return Scan Pointer
X'FF' Update Scan Pointer

	0	1	2
	Index to IEDQAI and Bit	Parameter List Length - X'06'	Skip Count
4	Blank Character	X'00'	

IEDQAI

Bit 7 No Blank Character

0	1	2
Index to IEDQA0	Parameter List Length - X'04'	Skip Count

IEDQA0

UNLOCK

0	1
Index to IEDQBF	X'00'

IEDQBF

SCB ERROR WORD USAGE BY MODULE

SCBERR1 (Byte 0)

Bit	Bit Indication (On/Off)	Module Action
0	Is/is not an incomplete header	Checked by IEDQAT and IEDQA4. Checked by IEDQBD for IN/OUT message macro instructions. Checked by IEDQCI.
1	Is/is not an invalid origin	Checked the same as bit 0. Set by IEDQAM.
2	TSO is not/is in system	Checked the same as bit 0.
3	Is/is not high sequence	Checked the same as bit 0. Set by IEDQAH.
4	Is/is not low sequence	Checked the same as bit 0. Set by IEDQAH.
5	Message is not/is sent/received	Checked the same as bit 0.
6	Are not/are sufficient buffers	Checked the same as bit 0. Set by IEDQAK.
7	Is/is not a cutoff error	Checked the same as bit 0.
	RVI to a selection device for BSC buffered terminals	Set by Line End Appendage. Checked by the user-coded macros.

SCBERR2 (Byte 1)

Bit	Bit Indication (On/Off)	Module Action
0	Core minimum is/is not exceeded	Set by IEDQBD for AVT SYSER.

		Checked by IEDQBD for IN/OUT message macro instructions. Checked by IEDQCI.
1	Core maximum is/is not exceeded	Set by IEDQBD for AVT SYSER. Checked by IEDQBD for IN/OUT message macro instructions. Checked by IEDQCI.
2	Error is/is not in a dynamic translate operation	Set and checked the same as bit 1.
3	Is/is not automatic line numbering	Set and checked the same as bit 1.
4	TOTE is not/is in the system	Set by IEDQAA. Checked by the error macros.
5	BSC abort sequences are/are not received	Set by Line End Appendage. Checked by the error macros,
6	Forward terminal error	Set by IEDQA4 and IEDQA5. Checked by the INMSG macro.
7	Reserved	

SCBERR3 (Byte 2)

Bit	Bit Indication (On/Off)	Module Action
0	Message is lost/processed	Checked by IEDQBD for IN/OUT message macro instructions. Checked by IEDQCI. Set by IEDQFA and IEDQFQ for a lost message
1	Terminal ID is invalid/valid	Checked by IEDQBD for IN/OUT message macro instructions. Checked and set by

		IEDQBT. Checked by IEDQCI.
2	Terminal is inoperative/operative	Checked by IEDQBD for IN/OUT message macro instructions. Checked by IEDQCI.
3	Simulated attention is/is not received	Checked the same as bit 2.
4	User error has/has not occurred	Checked the same as bit 2.
5	Is/is not format error in BSC message	Checked the same as bit 2.
6	Is/is not hardware attention	Checked the same as bit 2. Set by IGG019RQ.
7	Is/is not unit exception	Checked the same as bit 2. Set by IGE0504G.

SCBERR4 (Byte 3)

Bit	Bit Indication (On/Off)	Module Action
0	Is/is not selection error	Checked by IEDQBD for IN/OUT message macro instructions. Checked by IEDQCI. Set by IGG019RQ.
1	Is/is not error during text transfer	Checked by IEDQAA. Checked by IEDQBD for IN/OUT message macro instructions. Checked by IEDQCI. Set by IGE0004G. Set by IGE0004H. Set by IGE0104G. Set by IGE0104H. Set by IGE0204H. Set by IGG019RQ.
2	Is/is not error in connect/disconnect	Checked by IEDQBD for IN/OUT message macro instructions. Checked by IEDQCI. Set by IGE0304G.

3	Is/is not terminal error	Checked by IEDQBD for IN/OUT message macro instructions. Checked by IEDQCI.
4	Is/is not line error	Checked the same as bit 3.
5	Is/is not error in control unit	Checked the same as bit 3. Set by IGE0104G. Set by IGE0104H.
6	Is/is not channel error	Checked the same as bit 3. Set by IGE0104G. Set by IGE0804G. Set by IGE0804H.
7	Is/is not undefined error	Checked the same as bit 3. Set by IGE0504G. Set by IGE0504H.

LCB STATUS BYTE USAGE BY MODULE

LCBSTAT1

Bit	Bit Indication (On/Off)	Module Action
0	Recall being/not being performed	Checked, turned off, and reset by IEDQBD. Checked by IEDQFA and IEDQCI. Cleared by IEDQAA and IEDQAT.
1	Line is/is not in control mode	Checked by IEDQCI. Cleared by IEDQAA.
2	Operator control is not/is immediate	Set by IEDQHK. Checked and cleared by IEDQAA. Checked by IEDQCI.
3	Is/is not initiate mode	Reset by IEDQBD. Checked by IEDQFA, IEDQHM, and IEDQCI. Checked and cleared by IEDQAA.
4	Is/is not continue/reset operation	Set by IEDQCU. Checked by IEDQCI. Cleared by IEDQAA.
5	Line is/is not free	Checked by IEDQCI and IEDQHK. Cleared by IEDQAA.
6	Line is/is not receiving	Set by IEDQCU. Checked by IEDQGA and IEDQCI. Cleared by IEDQAA.
7	Line is/is not sending	Checked by IEDQAS, IEDQAG, IEDQAN, IEDQAW, IEDQA4, IEDQBD, IEDQCI, IEDQFA, IEDQGA, and IGG019RN. Checked and cleared by IEDQAA.

Notes: If both bits 6 and 7 are off, the line is inoperative. When a stop line function is being performed, IEDQHK set LCBSTAT1 equal to X'00'. Also, IEDQAA and IEDQC2 set LCBSTAT1 to X'00' when TOTE asks fcr ccntrol; IEDQCU, IEDQCV, and IEDQCO test for this condition.

LCBSTAT2

Bit	Bit Indication (On/Off)	Module Action
0	I/O trace is/is not active for this line	Set and checked by IEDQCP. Checked by IEDQCI.
1	Is/is not MSGGEN/startup message	Turned off by IEDQBD. Checked by IEDQCI.
2	EOT from a buffered terminal without/with EOM	Checked by IEDQCI.
3	Send priority switch is/is not set by the send scheduler	Checked by IEDQCI.
4	Negative response to polling is/is not received	Checked by IEDQCI. Set by IEDQHK.
5	Line is/is not BSC	Checked by IGG019RN and IEDQCI.
6	Is/is not a dial LCB	Checked by IEDQAG, IEDQCI, IEDQCO, IEDQCZ, and IEDQHK.
7	Do/do not owe a terminal a response	Checked by IEDQAK and IEDQCI. Checked and cleared by IEDQA4.

FORMATTED TCAM DUMP

A formatted TCAM dump is automatically produced as a part of the OS ABEND/SNAP storage dump when TCAM is resident in the system. ABEND/SNAP storage dumps occur immediately after an abnormal termination, provided that the control program or problem program has issued an ABEND or SNAP macro instruction, or when the operator issues a CANCEL command that requests a dump, and the proper dump data sets have been defined.

The TCAM part of an MFT dump starts after the TRACE TABLE entries, and in an MVT dump, the TCAM part starts after the SAVE AREA TRACE entries. For a complete discussion of the CS portion of the dump, refer to the publication _IBM System/360 OS Programmer's Guide to Debugging_, GC28-6670.

The following discussion of the TCAM part of either the OS MFT or MVT dump is interspersed with sample sections from an ABEND dump. Capital letters represent the headings found in all dumps, and lowercase letters represent information that varies. The lowercase letter used indicates the mode of the information, and the number of letters indicate the length of the information.

- h represents 1/2 byte of hexadecimal information
- d represents 1 byte of decimal information
- c represents a 1-byte character

```
TCAM ADDRESS VECTOR TABLE  hhhhhh

SAVE AREA 1

0000      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0020      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0040      hhhhhhhh hhhhhhhh

SAVE AREA 2

0048                        hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0060      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0080      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh

SAVE AREA 3

0090                                             hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
00A0      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
00C0      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh

SAVE AREA 4

00D8                                                               hhhhhhhh hhhhhhhh
00E0      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0100      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh

DISABLED SAVE AREA

0120      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0140      hhhhhhhh hhhhhhhh
```

TCAM ADDRESS VECTOR TABLE hhhhhh
is the starting address of the TCAM Address Vector Table (AVT), which is generated by the INTRO macro instruction. The formatted dump of the AVT beginning with the first save area, labeled SAVE AREA 1, and ending with the disk queues section, labeled DISK, follows the TCAM ADDRESS VECTOR TABLE hhhhhh heading.

SAVE AREA 1
is the contents of the first save area defined in the AVT. The registers are saved in and restored from this area according to standard linkage conventions. Along the left-hand side of the dump are the relative offsets of this save area from the beginning of the AVT.

SAVE AREA 2
is the contents of the second save area defined in the AVT. The registers are saved in and restored from this area according to standard linkage conventions. Along the left-hand side of the dump are the relative offsets of this save area from the beginning of the AVT.

SAVE AREA 3
is the contents of the third save area defined in the AVT. The registers are saved in and restored from this area according to standard linkage conventions. Along the left-hand side of the dump are the relative offsets of this save area from the beginning of the AVT.

SAVE AREA 4
is the contents of the fourth save area defined in the AVT. The registers are saved in and restored from this area according to standard linkage conventions. Along the left-hand side of the dump are the relative offsets of this save area from the beginning of the AVT.

DISABLED SAVE AREA
is the contents of the fifth save area defined in the AVT. When a disabled TCAM routine gains control from the I/O Supervisor, it saves and restores consecutively the I/C Supervisor's registers 0 through 9 in this save area.

```
TABLE POINTERS

0148                        hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0160      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh
```

TABLE POINTERS
are the address of the first Device Characteristics Table, the address of the TCB of the TCAM MCP, the address of the TCAM Line I/O Trace Table, the Operator Control message identification string, the scrambled password character string, and three work areas used by the internal TCAM logic. The following table shows the different fields, their offsets relative to the beginning of the AVT (which are also given on the left-hand side of the dump), their length, and their contents.

Offset	Contents
+0148	Address of the first Device Characteristics Table entry
+014C	Disabled parameter list
+0150 +0154	Disabled doubleword scratch area
+0158 +015C	Enabled doubleword scratch area
+0160 +0164	The Operator Control message identification character string
+0168 +106C	The scrambled password character string
+0170	Address of the TCB of the TCAM MCP
+0174	Address of the TCAM Line I/O Trace Table

```
DISPATCHER READY QUEUES

0178                                                                 hhhhhhhh hhhhhhhh
0180      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
01A0      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
01C0      hhhhhhhh hhhhhhhh hhhhhhhh
```

DISPATCHER READY QUEUES
gives the contents of the TCAM Dispatcher Ready Queues (one enabled, one disabled) and various other fields in this section of the AVT. The following table shows the different fields, their offsets relative to the beginning of the AVT (which are also given

on the left-hand side of the dump) their length, and their contents.

+0178	Enabled Ready Queue (points to first element to be dispatched)
+017C	First word of the disabled FIFO Ready Queue
+0180	Second word of the disabled FIFO Ready Queue
+0184	Checkpoint work area
+0188	Operator Control work area
+018C +0190 +0194	Executable instructions to save the user's registers, if requested
+0198	Parameter List
+019C	Protection key
+019D	Address of the AVT
+01A0	Address of additional optional parameters
+01A4	Address of the TCAM Dispatcher Subtask Trace Table
+01A8	Address of the Termname Table
+01AC	User exit address in the READY macro expansion
+01B0	Address of the Line End Appendage BSC message scan subroutine (SCAN)
+01B4	Address of the Line I/O Interrupt Trace routine (IGG019Q0)
+01B8 +01BC	Tpost parameter list used by Operator Control
+01C0	Address of Start Parameter List
+01C4	Number of CIBs
+01C5	Number of Checkpoint Requests
+01C6	Number of line units
+01C8	Address of Hold/Release Terminal routine (IEDQAS)

```
TCB POINTERS

01CC                                hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
```

TCB POINTERS

gives the addresses of the TCBs for Checkpoint, Operator Control, On-Line Test, and the FE Common Write task. These tasks are attached tasks of the TCAM MCP. The following table shows the fields containing the addresses and the offsets of the fields relative to the beginning of the AVT.

+01CC	Address of the Checkpoint TCB
+01D0	Address of the Operator Control TCB
+01D4	Address of the On-Line Test TCB
+01D8	Address of the FE Common Write TCB

```
ECBS

01DC                                                                       hhhhhhhh
01E0     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0200     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0220     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0240     hhhhhhhh hhhhhhhh hhhhhhhh
```

ECBS

contains the addresses of some of the internal routines and subtasks of the TCAM MCP, the addresses of certain TCAM tables, the Checkpoint ECB, the On-Line Test ECB, the Operator Control ECB, the ECB used by the TCAM Dispatcher to cause TCAM to enter a wait state when the Ready Queues are empty, and the address of the FE Common Write ECB. The following table gives a list of the different fields, their contents, and their relative offsets from the beginning of the AVT (which are also given on the left-hand side of the dump).

+01DC	Address of the FE Common Write ECB
+01E0	Checkpoint ECB
+01E4	On-Line Test ECB
+01E8	Operator Control ECB
+01EC	ECB used by the Dispatcher to cause TCAM to be in wait state
+01F0	Address of the first Process Entry Control Block
+01F4	Address of the Option Table
+01F8	Address of the I/O Generator in the Activate subtask
+01FC	Address of the user trace exit
+0200	Address of the Cross Reference Table
+0204	Address of the Communications Parameter List
+0208	Address of the User Interface routine (IEDQUI)
+020C	Address of the Return Interface routine (IEDQLM)
+0210	Address of the routine to remove an element from the Time Delay QCB (IEDQHG02)
+0214	Address of the Address Finder routine (IEDQAL)
+0218	Address of the Buffer Association routine (IEDQGD)
+021C	Address of the Transparency CCW Builder routine (IEDQGT)
+0220	Address of the Buffer Step routine (IEDQAX)
+0224	Address of the TCAM Dispatcher (IGG019RB or IGG019RO)
+0228	Address of the Leased Receive Scheduler (IGG019R3)

+022C	Address of the Send Scheduler (IGG019R4)
+0230	Address of the Get Scheduler (IEDQEW)
+0234	Address of the Put Scheduler (IEDQEC)
+0238	Address of the Get FIFO Scheduler (IEDQEZ)
+023C	Address of the Log Scheduler (IEDQBZ)
+0240	Address of the Dial Receive Scheduler (IGG019R1)
+0244	Address of the Buffered Terminal Scheduler (IGG019RD)
+0248	Address of the Retrieve Scheduler (IEDQE7)

```
SPECIAL ELEMENTS

024C                                      hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0260      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0280      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
02A0      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
02C0      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh   hhhhhhhh hhhhhhhh hhhhhhhh
```

SPECIAL ELEMENTS

contains the Interval Checkpoint element, a special element to request removal of the Interval Checkpoint element from the Time Delay Queue, the Incident Checkpoint element, and several address and constant areas used by the internal TCAM logic. The following table gives a list of the different fields, their contents, their size, and their relative offsets from the beginning of the AVT.

+024C Reserved

+0250 Reserved

+0254 Reserved

+0258 Reserved

+025C Reserved

+0260 Reserved

+0264 Reserved

+0268 Reserved

+026C Reserved

+0270 Dummy Line ECB

+0274 Address of the Translation List for IEDQA3

+0278 Address of the World Trade Tone Characters

+027C Address of the Operator Awareness Message Router routine (IEDQNX)

+0280 Address of the I/O Trace Table Handler routine (IGG019Q0)

+0284 Address of the System Delay QCB

+0288 Address of the Stop Line QCB

+028C Special element to cause removal of the Checkpoint element from the Time Delay Queue

+029C
+02A0 Element to request Interval Checkpoint

+02A4 Size of SCB	+02A5 Address of Checkpoint QCB

+02A8 Checkpoint Request element flags	+02A9 Number of Checkpoint Records	+02AA Checkpoint time interval

+02AC Time of day of interrupt	+02AE Offset to Checkpoint QCB	+02AF Open error locator
+02B0 Open module ID having error	+02B2 Type of Open error	+02B3 Checkpoint Time Delay Status

+02B4 Open translate byte	+02B5 Address of Time Delay subroutine (IEDQHG01)

Offset	Field	Offset	Field
+02B8	Offset to Binary Search routine	+02B9	Link field on Time Queue
+02BC	Dummy last element		
+02C0	Address of dummy last element		
+02C4 +02C8	Incident Checkpoint Element		
+02CC	Halfword constant X'0000'	+02CE	Halfword constant X'FFFF'
+02D0	Address of current buffer being processed (by Message Handler)		
+02D4	Address of the 2260 Local Line End Appendage (IGG019R5)		
+02D8	System error flag byte		Address of list of V-type address constants

```
QCB POINTERS

02DC                                                                             hhhhhhhh
02E0        hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0300        hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0320        hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0340        hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0360        hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0380        hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
```

QCB POINTERS

contains the Available Buffer QCB, the Buffer Return QCB, the Checkpoint QCB, the Operator Control QCB, the On-Line Test QCB, the Activate QCB, the Closedown QCB, the QCB to remove the Checkpoint element from the Time Delay Queue, the Disk I/O QCB, the CPB Cleanup QCB, the address of the Start-up Message QCB, the address of the Time Sharing Input QCB, the address of the application program OPEN/CLOSE routine, the address of the first byte of main storage obtained by GETMAIN for the buffer unit pool, a word containing the number of buffer units being used by the main storage message queues data set, and a fullword constant of zeroes. The following table gives a list of the different fields, their contents, their size and their relative offsets from the beginning of the AVT.

+02DC	Queue of available insert blocks
+02E0	Address of the Start-up Message QCB
+02E4	Address of the Time Sharing Input QCB
+02E8	Address of the application program OPEN/CLOSE routine (IEDQEU)
+02EC	Time Delay QCB
+02FC	Reference Time
+02FE	Dummy INEND/OUTEND AVT
+0300	SVC 102 parameter list, used to cause SVC 102 to tpost the Time
+0304	Delay QCB to itself when a timer interrupt occurs
+0308	Time Delay Queue
+030C	Available Buffer QCB
+0318	Buffer Return QCB
+0324	Checkpoint QCB
+0330	Operator Control QCB
+033C	On-Line Test QCB
+0348	Activate QCB
+0354	Closedown QCB
+0360	QCB to remove Checkpoint element from Time Delay Queue
+036C	Disk I/O QCB
+0378	CPB Cleanup QCB
+0384	Address of area obtained by GETMAIN for buffer unit pool
+0388	Number of buffer units being used by main storage message queues
+038C	Fullword constant of zero

```
INTERFACE

0390                                                 hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
03A0     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
03C0     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
03E0     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0400     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
```

INTERFACE

contains a GETMAIN parameter list used to obtain the buffer unit pool, the key length specified for the message queues, the number of lines opened, the number of lines in the system delay, the offset into the Termname Table of the Primary Operator Control terminal, the number of buffer units in the buffer pool, the number of lines serviced by the Start-up Message subtask, the number of seconds of the system delay, the offset into the Termname Table of the dead-letter queue terminal, three flags and several constants used by the internal TCAM logic, the number of restart Checkpoint records, the number of buffers or CPBs on the EXCP or Retry Queue, and the address of the FE patch module used for additional serviceability routines. Also, there is an FE work area, two parameter list pointers, two ECBs, and four flag bytes all used by the FE Common Write subtask. The following table gives a list of the different fields, their contents, their size, and their relative offsets from the beginning of the AVT.

+0390	Address of the FE Patch module (IEDQFE)		
+0394	First parameter list pointer		
+0398	First ECB		
+039C FE flag byte 1	+039D FE flag byte 2	+039E FE flag byte 3	+039F FE flag byte 4
+03A0	Second parameter list pointer		
+03A4	Second ECB		
+03A8	FE work area		
+03AC			
+03B0			
+03B4			
+03B8			
+03BC			
+03C0			
+03C4			
+03C8			
+03CC			
+03D0			
+03D4			
+03D8			
+03DC			
+03E0			
+03E4			
+03E8			

+03EC			
+03F0			
+03F4 GETMAIN parameter list			
+03F8			
+03FC		+03FE Halfword constant of 2	
+0400 Halfword constant of 3		+0402 Halfword constant of 4	
+0404 Halfword constant of 7		+0406 Halfword constant of 16	
+0408 Key length on message queues		+040A Number of lines opened	
+040C Number of lines in system delay		+040E Offset to Primary Operator Control terminal	
+0410 Number of buffer units in buffer unit pool		+0412 Number of lines serviced by Start-up Message subtask	
+0414 Number of seconds of system delay		+0416 Offset to dead-letter terminal	
+0418 BR instruction		+041A Flag byte 1	+041B Flag byte 2
+041C Flag byte 3	+041D Number of Restart Checkpoint Records	+041E Number of buffer or CPBs on EXCP or Retry Queue	

Note: This is the end of the AVT when ENVIRON=TSO has been specified on the INTRO macro instruction.

```
CORE QUEUE
0420     hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh     hhhhhhhh hhhhhhhh
```

CORE QUEUE
contains the address of the Destination Assignment routine, the values specified by the MSMIN=, MSMAX=, and MSUNITS= operands of the INTRO macro instruction, and a queue of buffers and ERBs waiting tc be processed. The follcwinq table gives a list of the different fields, their contents, their size, and their relative offsets from the beqinning of the AVT.

+0420	Address of the Destination Assignment routine (IEDQHM02)
+0424	MSMIM=integer
+0428	MSMAX=integer
+042C	Number of units usable in main storage queues (MSUNITS=integer)
+0430 +0434	Queue of buffers and ERBs waiting to be processed

```
DISK

0438                                                                 hhhhhhhh hhhhhhhh
0440      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0460      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
0480      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
04A0      hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh    hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
04C0      hhhhhhhh hhhhhhhh hhhh
```

DISK

contains the queues and control information for the disk message queues (reusable and nonreusable) for the TCAM MCP. The following table gives a list of the different fields, their contents, their size, and their relative offsets from the beginning of the AVT.

+0438	Address of the Disk EXCP Driver routine (IGG019RC)
+043C	Address of the Reusability subtask (IGG019RP - REUS)
+0440	Address of the Copy subtask QCB (IGG019RP - COPY)
+0444 +0448	Disabled queue of CPBs to be processed by CPB Cleanup
+044C +0450	Enabled queue of CPBs to be processed by CPB Cleanup
+0454 +0458	Queue of CPBs waiting for buffers
+045C +0460	Queue of CPBs being returned to the Reusability subtask by CPB Cleanup
+0464 +0468	Queue of CPBs requesting I/O to be done by the Disk EXCP Driver
+046C	Queue of inactive CPBs, called the CPB Free pool
+0470	Address of the CPB Free pool
+0474	Address of list of IOBs for reusable disk queues
+0478	Address of list of IOBs for non-reusable queues
+047C	Reusable disk queue when Reusability subtask activated
+0480	Address of DEB (reusable disk)
+0484	Number of extents (reusable disk)
+0488	Number of records per track (reusable disk)
+048C	Number of tracks per cylinder (reusable disk)

+0490	Number of records in entire data set (reusable disk)
+0494	Product of number of extents times number of records per track (reusable disk)
+0498	Address of DEB (non-reusable disk)
+049C	Number of extents (non-reusable disk)
+04A0	Number of records per track (non-reusable disk)
+04A4	Number of tracks per cylinder (non-reusable disk)
+04A8	Number of records in entire data set (non-reusable disk)
+04AC	Product of number of extents times number of records per track (non-reusable disk)
+04B0	Absolute record number that is the threshold to cause closedown due to filling of the non-reusable disk queue
+04B4	Non-reusable disk queue
+04B8	Reusable disk queue
+04BC	Non-reusable Threshold Closedown Element
+04C8	CPB = integer

Note: This is the end of the AVT.

```
TNT  hhhhhh  CODE  hhhhhhhh      hhhhhhhh      hhhhhhhh      hhhhhhhh      hhhhhhhh
                   hhhhhhhh      hhhhhhhh      hhhhhhhh      hhhhhhhh      hhhh
             SRCHX hhhh    ENLEN hh      MIDEN hhhhhh   LEN  hhhh
             DCODE hhhhhhhh      hhhhhhhh      hhhhhhhh      hhhhhhhh      hhhhhhhh
                   hhhhhhhh      hhhhhhhh      hhhhhhhh      hhhhhhhh
```

TNT hhhhhh
is the address of the TCAM Termname Table, which contains the names and addresses of all of the Terminal Table entries. (Each of the Terminal Table entries is displayed following this section of the dump.)

CODE
is the executable Termname Table code that converts the invitation list relative position field into the absolute address of the Terminal Table entry. This code is used only by enabled routines.

SRCHX hhhh
is the search extent factor.

ENLEN hh
is the number of bytes in each entry.

MIDEN hhhhhh
is the absolute address of the middle entry.

LEN hhhh
is the total number of entries.

DCODE
is the executable Termname Table code that converts the invitation list relative position field into the absolute address of the Terminal Table entry. This code is used only by disabled routines.

Following the TNT section of the dump are each of the Terminal Table entries along with their Option Table entries (if any exist) and contents. Some additional fields in each of the Terminal Table entries may or may not be present according to the optional parameters specified on the TERMINAL macro instruction. These are discussed where applicable. There are four different types of entries in the Terminal Table. They are single entries, list entries (cascade and distribution), Process entries, and Line entries. The following four sections give an example of each type of entry. Each of the four types of entries has a STATE field. The following table is a list of the bit meanings of this one-byte status field.

BIT(S)	MEANING
0-2	000 = single entry 001 = process entry 010 = list entry (cascade or distribution) 100 = line entry
3	always 0 for a list or process entry always 1 for a single or line entry
4	0 = PUT type process entry (if process entry) 1 = GET type process entry (if process entry, always 1 for other type entries)
5	0 = terminal is not in HOLD mode 1 = terminal is in HOLD mode
6	0 = no option fields used 1 = option fields used
7	0 = not secondary Operator Control terminal 1 = secondary Operator Control terminal

The followinq is an examrle of a sinqle entry.

```
NAME cccccccc
TRM  hhhhhh  STATE/DESTQ hhhhhhhh   IN/OUTSEQ hhhhhhhh   ALTD/DEVFL hhhhhhhh   STAT hhhhhhhh   CHCIN/OPNO/OPTBL hhhhhhhh

             NAME      ADDR   OPTION FIELD
             cccccccc hhhhhh hhhhhhhh
             cccccccc hhhhhh hhhhhhhh

             BUFFSIZE          hhhh
             DIAL DIGITS       hhhhhhh
             ADDR CHAR         hhhhhh
             BLOCK             hhhh
             SUBBLOCK          hh
             TRANS BLOCK       hhhh
             BFDELAY           hhhh
             TIME SHARING      hhhh
```

NAME cccccccc
    is the name in the Termname Table of this Terminal Table entry.

TRM hhhhhh
    is the address of the Terminal Table entry.

STATE/DESTQ hhhhhhhh
    The first byte is the status byte of the Terminal Table entry. The last three bytes contain the address of the Destination QCB for this entry.

IN/OUTSEQ hhhhhhhh
    The first two bytes contain the next expected input sequence number. The second two bytes contain the next output sequence number to be used.

ALTD/DEVFL hhhhhhhh
    The first two bytes contain the offset into the Terminal Table of the alternate destination for this entry. The last two bytes are flaq bytes used by the internal TCAM loqic. The followinq table is a list of the bits and their meaninqs.

BIT(S)	MEANING
0	BUFSIZE= specified
1	Dial diqits present
2	Addressinq characters present
3	BLOCK= specified
4	SUBBLCK= specified
5	TRANSP= specified
6	BFDELAY= specified
7	Time Sharinq field present
8-15	Reserved

STAT hhhhhhhh
    is a word for error statistics.

CHCIN/OPNO/OPTBL hhhhhhhh
    The first byte is the index to the Device Characteristics Table for this entry. The second byte qives the number of option fields for this entry. The next two bytes contain the offset into the Option Table for the option fields for this entry.

NAME   ADDR OPTION FIELDS
cccccccc hhhhhh hhhhhhhh

gives a list of the names, addresses, and contents of each of the option fields for this entry.

BUFSIZE hhhh

is the output buffer size for this entry. This value is given in the dump only when a nonzero value has been specified on the BUFSIZE= operand of the TERMINAL macro.

DIAL DIGITS hhhhhhh

is the telephone number of this terminal. This field is given in the dump only when the CALL= operand of the TERMINAL macro has been specified, except where CALL=NONE was specified.

ADDR CHAR hhhh

is the addressing characters for the terminal as specified on the ADDR= operand of the TERMINAL macro.

BLOCK hhhh

is the number of bytes to be transmitted in each block of data in non-transparent mode for messages sent to this terminal. The value corresponds to the value specified in the BLOCK= operand of the TERMINAL macro and is not given in the dump if the value was not specified.

SUBBLOCK hh

is the number of bytes to be transmitted in each sub-block of data in non-transparent mode for messages sent to this terminal. The value corresponds to the value specified in the SUBBLCK= operand of the TERMINAL macro and is not given in the dump if the value was not specified.

BFDELAY hhhh

is the number of seconds of delay to be used between message blocks being sent to a buffered terminal. This field is given in the dump only if the BFDELAY= operand of the TERMINAL macro has been specified.

TIME SHARING hhhh

is a field used by Time Sharing. In the case that this entry is an IBM 2260 or an IBM 2265, the first byte is the number of lines that can be displayed and the second byte is the number of characters per line. If the terminal is not an IBM 2260 or an IBM 2265, both bytes are zero. This field is given in the dump only when Time Sharing is being used.

The following is an example of a list entry.

```
NAME cccccccc
TRM  hhhhhh  STATE/DESTQ hhhhhhhh  TLISTCNT hhhh

              LIST ENTRIES
                cccccccc
                cccccccc
```

NAME cccccccc
is the name in the Termname Table of this Terminal Table entry.

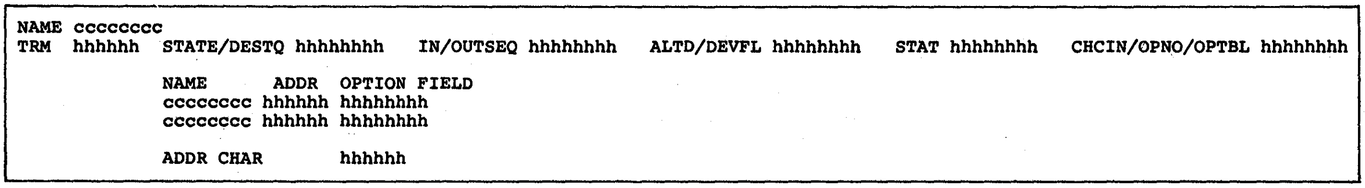
TRM hhhhhh
is the address of the Terminal Table entry.

STATE/DESTQ hhhhhhhh
The first byte is the status byte of the Terminal Table entry. The last three bytes contain the address of the Destination QCB.

TLISTCNT hhhh
is the number of entries in this distribution or cascade list.

LIST ENTRIES
is a list of the names that appear in the cascade or distribution list.

The following is an example of a line entry.



```
NAME cccccccc
TRM  hhhhhh  STATE/DESTQ hhhhhhhh   IN/OUTSEQ hhhhhhhh   ALTD/DEVFL hhhhhhhh   STAT hhhhhhhh   CHCIN/OPNO/OPTBL hhhhhhhh

             NAME      ADDR   OPTION FIELD
             cccccccc hhhhhh hhhhhhhh
             cccccccc hhhhhh hhhhhhhh

             ADDR CHAR        hhhhhh
```

NAME cccccccc
is the name in the Termname Table of this Terminal Table entry.

TRM hhhhhh
is the address of the Terminal Table entry.

STATE/DESTQ hhhhhhhh
The first byte is the status byte of the Terminal Table entry. The last three bytes contain the address of the Destination QCB for this entry.

IN/OUTSEQ hhhhhhhh
the first two bytes contain the next expected input sequence number. The second two bytes contain the next output sequence number to be used.

ALTD/DEVFL hhhhhhhh
The first two bytes contain the offset into the Terminal Table of the alternate destination for this entry. The last two bytes are flag bytes used by the internal TCAM logic. The following table is a list of the bits and their meanings.

BIT(S)	MEANING
0	BUFSIZE= specified
1	Dial digits present
2	Addressing characters present
3	BLOCK=specified
4	SUBBLCK=specified
5	TRANSP=specified
6	BFDELAY=specified
7	Time Sharing field present
8-15	Reserved

STAT hhhhhhhh
is a word for error statistics.

CHCIN/OPNO/OPTBL hhhhhhhh
The first byte is the index to the Device Characteristics Table for this entry. The second byte gives the number of option fields for this entry. The next two bytes contain the offset into the Option table for the option fields for this entry.

NAME ADDR OPTION FIELDS
cccccccc hhhhhh hhhhhhhh
gives a list of the names, addresses, and contents of each of the option fields for this entry.

ADDR CHAR hhhh
is the addressing characters for the terminal as specified on the ADDR= operand of the LINE macro.

The following is an example of a process entry.

```
NAME cccccccc
TRM  hhhhhh   STATE/DESTQ hhhhhhhh   IN/OUTSEQ hhhhhhhh   ALTD/DEVFL hhhhhhhh   STAT hhhhhhhh   CHCIN/OPNO/OPTBL hhhhhhhh

              NAME      ADDR  OPTION FIELD
              cccccccc hhhhhh hhhhhhhh
              cccccccc hhhhhh hhhhhhhh
```

```
TCAM DESTINATION QCB'S

QCB  hhhhhh   DSFLG/ELCHN hhhhhhhh   PRI/LINK     hhhhhhhh   STVTO/STCHN hhhhhhhh   STPRI/SLINK hhhhhhhh
              EOLDT/STAT  hhhhhhhh   SCBOF/INSRC hhhhhhhh   INTVL/MSGCT hhhhhhhh   PRLVL/LKRRN hhhhhhhh
              RELLN/DCBAD hhhhhhhh   FLAG/QBACK   hhhhhhhh

     PRIORITY QCB  hhhhhh
              DNHDR hhhhhh   FHDLZ hhhhhh   FHDTZ hhhhhh   INTFF hhhhhh   INTLF hhhhhh
              FFEFO hhhhhh   LFEFO hhhhhh   CFHDR hhhhhh   PRIPQ hh       CPVHD hhhhhh
```

NAME cccccccc
is the name in the Termname Table of this Terminal Table entry.

TRM hhhhhh
is the address of the Terminal Table entry.

STATE/DESTQ hhhhhhhh
The first byte is the status byte of the Terminal Table entry. The last three bytes contain the address of the Destination QCB for this entry.

IN/OUTSEQ hhhhhhhh

The first two bytes contain the next expected input sequence number. The second two bytes contain the next output sequence number to be used.

ALTD/DEVFL hhhhhhhh

The first two bytes contain the offset into the Terminal Table of the alternate destination for this entry. The last two bytes are flag bytes used by the internal TCAM logic. The following table is a list of the bits and their meanings.

BIT(S)	MEANING
0	BUFSIZE= specified
1	Dial digits present
2	Addressing characters present
3	BLOCK= specified
4	SUBBLCK= specified
5	TRANSP= specified
6	BFDELAY= specified
7	Time Sharing field present
8-15	Reserved

STAT hhhhhhhh

is a word for error statistics.

CHCIN/OPNO/OPTEL hhhhhhhh

the first byte is the index to the Device Characteristics Table for this entry. The second byte gives the number of option fields for this entry. The next two bytes contain the offset into the Option table for the option fields for this entry.

```
NAME      ADDR   OPTION FIELDS
cccccccc hhhhhh hhhhhhhh
```

gives a list of the names, addresses, and contents of each of the option fields for this entry.

TCAM DESTINATION QCB'S

gives the Destination QCBs for all of the Terminal Table entries. These QCBs are used to control the message queuing for the terminals in the TCAM system. Each QCB may service one or more terminals depending upon the type of queuing specified in the TERMINAL macro. Each of these QCBs consists of a Master QCB and one or more priority level QCBs. Priority QCBs are generated due to the LEVEL= operand of the TERMINAL macro. If this operand is omitted, only one priority level QCB is generated and its priority is X'00'. Whether or not the LEVEL= operand is specified, the X'00' priority level QCB is generated.

QCB hhhhhh

is the starting address of the Master QCB.

DSFLG/ELCHN hhhhhhhh

The first byte is a flag byte indicating the type of queuing being used by this QCB. The next three bytes contain the address of the next element in the chain.

PRI/LINK hhhhhhhh

The first byte is the priority of this QCB. The last three bytes contain the address if the next STCB in the chain.

STVTO/STCHN hhhhhhhh

The first byte is the index to the entry in the Subtask Vector Table. The last three bytes contain the STCB chain.

STPRI/SLINK hhhhhhhh

The first byte is the priority of the STCB. The last three bytes contain the address of the next STCB in the chain.

EOLTD/STAT hhhhhhhh

The first two bytes contain the interrupt time used by the Time Delay routine. The third byte is the LOCK relative line number, and the fourth byte is the QCB status byte.

SCBOF/INSRC hhhhhhhh

The first byte is the offset to the proper SCB for the current transmission. The next three bytes contain the address of the first LCB in the source LCB chain.

INTVL/MSGCT hhhhhhhh

The first two bytes contain the value as specified on the CLOCK= or INTVL= operand of the TERMINAL macro. The second two bytes contain the count of the messages on this queue.

PRLVL/LKRRN hhhhhhhh

The first byte is the priority of the highest priority level message in the queue. The last three bytes contain the LOCK relative record number.

RELLN/DCBAD hhhhhhhh

The first byte is the relative line number for the line that this QCB represents. The last three bytes contain the address of the DCB.

FLAG/QBACK hhhhhhhh

The first byte is an additional status byte for the QCB. The last three bytes contain the QBACK message chain.

PRIORITY QCB hhhhhh

is the address of this priority level QCB.

DNHDR hhhhhh

is the disk record number assigned to the next header that is received.

FHDLZ hhhhhh

is the disk record number of the first header placed in the last zone used by this queue.

FHDTZ hhhhhh
is the disk record number of the first header placed in the current zone.

INTFF hhhhhh
is the disk record number of the first held message in this queue (placed in FEFO order).

INTLF hhhhhh
is the disk record number of the last held message in this queue (placed in FEFO order).

FFEFO hhhhhh
is the disk record number of the first message that has not been sent (placed in FEFO order)..

LFEFO hhhhhh
is the disk record number of the last message that has not been sent (placed in FEFO order).

CFHDR hhhhhh
is the main storage queue address of the first header appearing in this queue.

PRIPQ hh
is the priority level of this priority level QCB.

CPVHDR hhhhhh
is the main storage queue address of the last header appearing in this queue.

```
TCAM DCB'S

DCB  hhhhhh  (LINE GROUP)
             DEVICE INTERFACE  hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh
             D/S INTERFACE     hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh
             FOUNDATION        hhhhhhhh   hhhhhhhh   hh
             EXTENSION                                 hhhhhh   hhhhhhhh   hhhhhhhh
             INVITATION LISTS  hhhhhhhh

     LCB  hhhhhh  KEY/QCBA    hhhhhhhh  PRI/LINK    hhhhhhhh  RSKEY/STCBA hhhhhhhh  RSPRI/RSLNK hhhhhhhh
                  EOLTD/TSOB  hhhhhhhh  CHAIN/INSRC hhhhhhhh  SCBO/SCBDA  hhhhhhhh  ISZE/FSBRF  hhhhhhhh
                  FLAGS/SENSE hhhhhhhh  ECBCC/ECBPT hhhhhhhh  FLAG3/CSW   hhhhhhhh              hhhhhhhh
                  SIOCC/START hhhhhhhh  DCBPT       hhhhhhhh  RCQCB       hhhhhhhh  INCAM/ERRCT hhhhhhhh
                  UCBX/RCBFR  hhhhhhhh  RECOF/STATE hhhhhhhh  TSTSW/RECAD hhhhhhhh  ERBKY/ERBQB hhhhhhhh
                  ERBPY/ERBLK hhhhhhhh  ERBST/ERBCH hhhhhhhh  ERBCT/TTCIN hhhhhhhh  MSGFM/SCBA  hhhhhhhh
                  ERMSK/INVPT hhhhhhhh  TPCD        hhhhhhhh              hhhhhhhh              hhhhhhhh
                  SNSV/CSWSV  hhhhhhhh              hhhhhhhh  ERCCW       hhhhhhhh              hhhhhhhh
                              hhhhhhhh              hhhhhhhh              hhhhhhhh              hhhhhhhh
```

TCAM DCB'S
gives the three different types of TCAM DCBs: the Line Group DCBs (along with their related LCBs), the Message Queues DCBs, and the Checkpoint DCB. (Note: The Message Queues DCBs are not given in the dump if the TCAM system does not use disk queuing, and the Checkpoint DCB is not given in the dump if the Checkpoint/Restart facility is not being utilized.)

DCB hhhhhh (LINE GROUP)
is the starting address of this Line Group DCB.

DEVICE INTERFACE

This section is reserved.

D/S INTERFACE

contains the number of buffers assigned initially for input operations, the number of buffers assigned initially for output operations, the address of the Message Handler for this Line Group, the polling delay interval, the Program-Controlled Interruption options, the Data Set organization, the maximum number of buffers to be used at any given time for data transfer for each line in the Line Group, the Open-Base for addressing IOBs, the relative priority of send and receive operations, the address of the Translation Table, the extended IOB index (size of an LCB), and the address of the exit list. The following table shows these fields, their relative offsets from the beginning of the DCB, their contents and their size.

+14 Initial Receive Allocation	Initial Send Allocation	+15 Address of the Message Handler	
+18 Polling delay interval		+19 PCI options -	+1A Data Set organization
+1C Maximum Send or Receive Allocation		+1D Open-Base for addressing IOBs	
+20 Priority of Send/Receive Operations		+21 Address of the Translation Table	
+24 IOB index		+25 Address of the exit list	

For more detailed information on these fields, refer to the publication, IBM System/360 Operating System System Control Blocks, GC28-6628.

FOUNDATION

contains fields that are changed during Open. Before Open, these fields contain the DDNAME character string, the Open flags, the IOS error flags, and the Macro Instruction reference. After Open, they contain the offset of the DD entry from the beginning of the TIOT, the Macro Instruction Reference, the IOS error flags, the address of the DEB, and the Open flags. The following two tables show this area and its contents before and after Open.

Before Open:

+28 DDNAME character string		
+2C		
+30 Open flags	+31 IOS error flags	+32 Macro Instruction Reference

Note: During Open, the IOS error flags field and the Macro Instruction Reference field are relocated and the last three bytes of the last word become part of the EXTENSION section.

After Open:

<table>
<tr><td colspan="2">+28 Offset of DD entry from beginning of the TIOT</td><td colspan="2">+2A Macro Instruction Reference</td></tr>
<tr><td>+2C IOS error flags</td><td colspan="3">+2D Address of the DEB</td></tr>
<tr><td>+30 Open flags</td><td colspan="3"></td></tr>
</table>

For more detailed information on these fields, refer to the publication, <u>IBM System/360 Operating System System Control Blocks</u>, GC28-6628.

EXTENSION

contains the address of the Special Characters Table, the number of invitation lists, the number of units for each buffer, the size of all buffers used by this Line Group, and the number of reserve characters. The following table shows these fields, their relative offsets from the beginning of the DCB, their contents, and their size.

<table>
<tr><td></td><td colspan="3">+31 Address of the Special Characters Table</td></tr>
<tr><td>+34 Number of Invitation Lists</td><td>+35 Number of Units per Buffer</td><td colspan="2">+36 Buffer size</td></tr>
<tr><td colspan="4">+38 Four one-byte reserve values</td></tr>
</table>

For more detailed information on these fields, refer to the publication, <u>IBM System/360 Operating System System Control Blocks</u>, GC28-6628.

INVITATION LISTS
gives the addresses of the different invitation lists for the different lines in the Line Group. Each list is pointed to by a one-word address. These addresses are given in order by relative line number.

Following each Line Group DCB is one or more LCBs (Line Control Blocks) which are used by the internal TCAM logic to perform line management. The LCBs in the dump are given in order by relative line number.

LCB hhhhhh
is the starting address of this LCB.

KEY/QCBA hhhhhhhh
The first byte is the key of this LCB. The next three bytes contain the address of its QCB.

PRI/LINK hhhhhhhh
The first byte is the priority of this LCB. The next three bytes contain the link address to the next element.

RSKEY/STCBA hhhhhh
The first byte is the Receive Scheduler key. The next three bytes contain the address of the first STCB when the LCB is a QCB.

RSPRI/RSLNK hhhhhhhh
The first byte is the Receive Scheduler priority. The next three bytes contain the address of the next item in the chain.

EOLTD/TSOB hhhhhhhh
The first two bytes contain the end of polling list time delay reference time. The third byte is the time delay queue offset to the QCB address (always X'14' for an LCB). The fourth byte is a status byte used by Time Sharing.

CHAIN/INSRC hhhhhhhh
The first byte is a status byte used by TCAM. The next three bytes contain the in-source chain.

SCBO/SCBDA hhhhhhhh
The first byte is the offset to the current SCB (Station Control Block). The next three bytes contain the address of the SCB directory.

ISZE/FSBFR hhhhhhhh
The first byte is the count of reserved idles. The next three bytes contain the address of the first buffer assigned to this line.

FLAGS/SENSE hhhhhhhh
is the start of the IOB contained in the LCB. The first and second bytes are IOS flags. The last two bytes are the sense bytes.

ECBCC/ECBPT hhhhhhhh
The first byte is the ECB completion code. The next three bytes contain the address of the ECB.

FLAG3/CSW
The first byte is an IOS flaq byte. The next seven bytes are the last seven bytes of the CSW.

STOCC/START hhhhhhhh
The first byte is the Start I/O conditicn code, The last three bytes contain the address cf the start of the channel program area.

DCBPT hhhhhhhh
is the address of the DCB for this line.

RCQCB hhhhhhhh
is the address of the QCB to tpost a recalled buffer tc IOS.

INCAM/ERRCT hhhhhhhh
are two half word IOS error counters.

UCBX/RCBFR hhhhhhhh
The first byte is the UCB index. The last three bytes contain the address of a recalled buffer or the last buffer serviced by a PCI.

RECOF/STATE hhhhhhhh
The first two bytes contain the offset into the current block. The last two bytes are the LCB status bytes.

TSTSW/RECAD hhhhhhhh
The first byte is a test-and-set switch. The last three bytes contain the address of the current messaqe block.

ERBKY/ERBQB hhhhhhhh
The first byte is the key of the ERB. The next three bytes contain the address of the QCB to which the ERB is tposted.

ERBPY/ERBLK hhhhhhhh
The first byte is the priority of this ERB. The next three bytes contain the address of the next item in the chain.

ERBST/ERBCH hhhhhhhh
The first byte is the ERB status byte. The next three bytes contain the address of a chain of assiqned buffers.

ERBCT/TTCIN hhhhhhhh
The first two bytes contain the count cf buffers requested by this ERB. The second two bytes contain the index into the Termname Table of the currently connected terminal.

MSGFM/SCBA hhhhhhhh
The first byte is used to control BSC lines. The next three bytes contain the address of the current SCB.

ERMSK/INVPT hhhhhhhh
The first byte is an error recording mask. The next three bytes contain the address of the current entry in the invitation list.

TPCD
is a three-word list of TP operation codes for the CCWs.

SNSV/CSWSV hhhhhhhh hhhhhhhh
the first byte is a save area for the sense byte. The last seven bytes comprise a save area for the CSW.

ERCCW
is a three doubleword area for ERP (Error Recovery Procedure) CCWs.

The following section gives the Checkpoint DCB.

```
DCB  hhhhhh  (CHECKPOINT)
             DEVICE INTERFACE   hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh
             D/S INTERFACE      hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh
             FOUNDATION         hhhhhhhh   hhhhhhhh   hh
             EXTENSION                                  hhhhhh   hhhhhhhh   hhhhhhhh
```

DCB hhhhhh (CHECKPOINT)
is the starting address of the Checkpoint DCB.

DEVICE INTERFACE
This section is reserved.

D/S INTERFACE
contains the Data Set organization, the address of the AVT, and the address of the exit list. The following table shows these fields, their relative offsets from the beginning of the DCB, their contents, and their size.

+14 Reserved	
+18 Reserved	+1B Data Set organization
+1C Reserved	+1D Address of the AVT
+20 Reserved	
+24 Reserved	+25 Address of the exit list

For more detailed information on these fields, refer to the publication, *IBM System/360 Operating System System Control Blocks*, GC28-6628.

FOUNDATION

contains fields that are changed during Open. Before Open, these fields contain the DDNAME character string, the Open flags, the IOS error flags, and the Macro Instruction reference. After Open, they contain the offset of the DD entry from the beginning of the TIOT, the Macro Instruction reference, the IOS error flags, the address of the DEB, and Open flags. The following two tables show this area and its contents before and after Open.

Before Open:

+28 DDNAME character string		
+2C		
+30 Open flags	+31 IOS error flags	+32 Macro Instruction reference

Note: During Open, the IOS error flags field and the Macro Instruction reference field are relocated and the last three bytes become part of the EXTENSION section.

After Open:

+28 Offset of DD entry from beginning of the TIOT	+2A Macro Instruction reference
+2C IOS error flags	+2D Address of the DEB
+30 Open flags	

For more detailed information on these fields, refer to the publication, *IBM System/360 Operating System System Control Blocks*, GC28-6628.

EXTENSION

contains the OPTCD= value of the DCB. The remainder of this area is reserved. The following table shows these fields, their relative offsets from the beginning of the DCB, their contents, and their size.

	+31 Reserved
+34 OPTCD=value	+35 Reserved
+38 Reserved	

For more detailed information on these fields, refer to the publication, IBM System/360 Operating System System Control Blocks, GC28-6628.

The following section gives the Message Queues DCB.

```
DCB  hhhhhh  (MESSAGE QUEUE)
             DEVICE INTERFACE   hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh
             D/S INTERFACE      hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh   hhhhhhhh
             FOUNDATION         hhhhhhhh   hhhhhhhh   hh
             EXTENSION                                  hhhhhh   hhhhhhhh   hhhhhhhh
```

DCB hhhhhh (MESSAGE QUEUES)
is the starting address of the Message Queues DCB.

DEVICE INTERFACE
This section is reserved.

D/S INTERFACE
contains the Data Set organization, the address of the AVT, the threshold value of the percentage of the nonreusable disk message queue records to be used before a flush closedown of the system is initiated, and the address of the exit list. The following table shows these fields, their relative offsets from the beginning of the DCB, their contents, and their size.

+14 Reserved	
+18 Reserved	+1A Data Set organization
+1C Reserved	+1D Address of the AVT
+20	+21 Reserved
+24 Reserved	+25 Address of the exit list

For more detailed information on these fields, refer to the publication, IBM System/360 Operating System System Control Blocks, GC28-6628.

FOUNDATION
contains fields that are changed during Open. Before Open, these fields contain the DDNAME character string, the Open flags, the IOS error flags, and Macro Instruction reference. After Open, they contain the offset of the DD entry from the beginning of the TIOT, the Macro Instruction reference, the IOS error flags, the address of the DEB, and Open flags. The following two tables show this area and its contents before and after Open.

Before Open:

+28 DDNAME character string +2C		
+30 Open flags	+31 IOS error flags	+32 Macro Instruction reference

Note: Durinq Open, the IOS error flaqs field and the Macro Instruction reference field are relocated and the last three bytes become part of the EXTENSION section.

After Open:

+28 Offset of DD entry from beginning of the TIOT	+2A Macro Instruction reference
+2C IOS error flags	+2D Address of the DEB
+30 Open flags	

For more detailed information on these fields, refer to the publication, IBM System/360 Operatinq System System Control Blocks, GC28-6628.

EXTENSION
contains the OPTCD= value of the DCB. The remainder of this area is reserved. The followinq table shows these fields, their relative offsets from the beqinninq cf the DCB, their contents, and their size.

+30	Reserved
+34 OPTCD=value	+35 Reserved
+38 Reserved	

For more detailed information on these fields, refer to the publication, IBM System/360 Operatinq System System Control Blocks, GC28-6628.

TRACE TABLES

SUBTASK TRACE TABLE

The Subtask Trace facility keeps a sequential record in main storage of the subtasks activated by the TCAM Dispatcher. This facility is included in a TCAM MCP when the DTRACE keyword of the INTRO macro is assembled with a nonzero numerical value. This facility is implemented by the IGG019RO TCAM Dispatcher, which places an entry in the Subtask Trace Table each time a subtask is dispatched.

The address of the Subtask Trace Table is stored at AVTPARM+12 in the AVT, and the number of entries in the table is determined by the numerical value coded on the DTRACE keyword. Each entry consists of four words, and there is one four-word Subtask Trace Table Control Block. At INTRO time, TCAM allocates (16n+16) contiguous bytes for the table; n is the integer value coded in the DTRACE operand of the INTRO macro. The table is filled on a wrap-around principle--when all the available entries have been used, the Dispatcher places new entries at the beginning of the table, overlaying the earliest entries.

The Subtask Trace Table Control Block is located in the first four words of the table. Therefore, the actual subtask entries start four words beyond the address in the AVT.

Format of the Subtask Trace Table Control Block

Offset	
0	Address of the next entry in the table
4	Address of the first entry in the table
8	Address of the last entry in the table
12	Reserved

Note: Since the first word of this control block contains the address of the next entry in the table, the last entry made is at this address minus 16.

Format of a Subtask Trace Table Fntry

Offset		
0	Priority of the dispatched element	Address of the dispatched element
4	Entry point address of the dispatched subtask	
8	Flag byte of the dispatched QCB	Address of the dispatched QCB
12	Subtask entry code (MCPL)	Address of the dispatched STCB

LINE INPUT/OUTPUT INTERRUPT TRACE TABLE

The Line Input/Cutput Interrupt Trace facility is loaded by OPEN when the TRACE keyword of the INTRO macrc is assembled with a nonzero numerical value. This facility is implemented by the I/O Interrupt Trace routine (IGG01900), which places an entry in the I/O Interrupt Trace Table each time an I/O interrupt occurs. Interrupts that occur as a result cf retries by TCAM's Error Recovery Procedures are not recorded in the I/O Interrupt Trace Table. The TCAM I/O Interrupt Trace facility is activated for a line by means of a GOTRACE Operator Control Messaqe, and may be deactivated fcr a line by means of a NOTRACE Operator Control Messaqe.

The address of the I/O Interrupt Trace Table is stored in the AVTRACE field cf the Address Vector Table. The number of 32-byte entries in the table is determined by the numerical value coded on the TRACE operand of the INTRO macro. At INTRO execution time, 32n contiquous bytes of main storaqe (where n is the inteqer specified in the "TRACE=" operand) are allocated for the I/O Interrupt Trace Table. When all entries in the table have been filled, the earliest entries are overlaid when new interrupts occur. The user may name (via the "TREXIT=" operand of INTRO) a user subroutine that is passed control when the I/O Interrupt Trace Table is full.

Format of an I/C Interrupt Trace Table Fntry

Byte:		
0	Sense Byte	+1 CSW
+8	Interrupt CCW	+12 TP OP code at Interrupted CCW
+16	First CCW in Channel Program Chain	+20 TP OP code of first CCW
+24	Station Name	+30 Channel Unit Address

CROSS REFERENCE TABLE

The TCAM Cross Reference Table provides a means of locating in a dump certain information associated with each open line. The Cross Reference Table is included in a TCAM MCP when the CROSSRF operand of the INTRO macro is assembled with a nonzero numerical value.

At INTRO execution time, the INTRO GETMAIN routine obtains four words of main storage for each unit specified on the CROSSRF operand. The total amount of main storage reserved is called the Cross Reference Table, and the INTRO GETMAIN routine places the address of this table at AVTCRSRF in the AVT. Each time a line is successfully opened, the IGG01940 Line Group Open routine completes the next available four-word entry in the table.

IGG01940 fills in the first word of the entry with the name of the UCB (the EBCDIC representation of the hardware line address), and fills in the second word with the address of the UCB. IGG01940 places the address of the LCB in the third word, and the address of a Master QCB (Queue Control Block) for this line in the fourth word. If queuing is by line, there is only one Master QCB assigned to the line, and its address is placed in the fourth word. If queuing is by terminal, there is one Master QCB for each station on the line; the fourth word is filled in with the address of the QCB for the station that has its terminal entry appear in the Terminal Table before the terminal entry of any other station on this particular line. If the user opens more lines than entries in the Cross Reference Table, TCAM fills the Table until the space is exhausted; lines opened after space runs out have no Cross Reference entry.

The format of each entry in the table is as follows:

Offset	
0	UCB name
4	UCB address
8	LCB address
12	Address of the Master QCB for this line

TABLE OF MESSAGE ORIGINS

This table lists each of the messages generated by the TCAM executable modules. The originating module names and the message routing codes are included by each message.

Routing Codes:

* This routing code indicates that the message must be routed back to the console that initiated the associated request.

1 MASTER CONSOLE. This routing code is for messages that must be sent to the master console because some action is required by the master console operator, or because the message contains information considered critical to the continued operation of the system.

2 MASTER CONSOLE INFORMATIONAL. This routing code is for informational messages to the master console operator. Informational messages usually require no action from the operator. If they do, that action should be at the operator's discretion.

8 TELEPROCESSING CONTROL. This routing code is for messages relating to teleprocessing.

10 SYSTEM ERROR/MAINTENANCE. This routing code is used for any message that indicates a system error or an incorrectable I/O error, or any message associated with system maintenance.

11 PROGRAMMER INFORMATION. This routing code is for messages of interest to the programmer. The message is sent to an operator console and not to the system output device.

Message	Origin	Routing Code
IED001I TCAM JOB jobname, stepname, procstepname ADDRESS OF AVT address	IEDQOB	2,11
IED002A SPECIFY TCAM PARAMETERS	IEDQOB	1
IED003A INVALID KEYWORD keyword	IEDQOB	1
IED004A REQUIRED PARAMETER MISSING. SPECIFY xxx xxx = the keyword missing S = STARTUP - Cold or Warm start B = LNUNITS - number of line buffers K = KEYLEN - size of each buffer D = CPB - number of CPBs (if disk is being used).	IEDQOB	1

IED005A MSUNITS (M) SPECIFICATION NOT PERMITTED. CONTINUE RESPONSE	IEDQOB	1
IED006A INVALID OPERAND ON KEYWORD. RESPECIFY keyword	IEDQOB	1
IED007I terminal name IS AN ILLEGAL DESTINATION	IEDQOM	11
IED008I TCAM OPEN ERROR xxx=y IN DCB zzz descriptor	IGG01930 IGG01931	11

xxx = 040 The error occurred in opening a line group data set.
041 The error occurred in opening a message queues data set.

when xxx=040 and

y = 1 Insufficient main storage to build a line group DEB.
2 Non-compatible devices specified in the same line group.
3 A UCB for a line in this line group specifies something other than telecommunications or graphics devices.
4 A UCB for a line in this line group specifies an unsupported control unit type.
5 A UCB for a line in this line group specifies a nonstandard terminal type.
6 The device characteristics table generated for this line does not contain the right characteristics for this terminal type as defined by the UCB.
7 Insufficient main storage was available to build a Line Control Block.
8 Insufficient main storage was available to build a Station Control Block.
9 The characteristics of a binary synchronous device do not agree with those values in the UCB for the line.
A The DD cards for this DCB contain no valid UCB addresses.
B The header prefix size and the number of idle characters plus one reserved for receiving operations exceed one logical unit in length.
C No data set has been specified for disk or main storage queuing for a specified terminal.
D There are no lines in the line group (that is, one or more of the lines in the group have not been opened due to some other error).
E A relative line number of zero has been specified.

when xxx=041 and

```
    y = 1  An incorrect-length AVT has been specified to
           support disk message queuing.
        2  The key length specified on the INTRO macro
           and the key length used to format the message
           queues data set are not equal.
        3  Dissimilar disk types have been defined for
           message queuing.
        4  Something other than reusable and nonreusable
           disk message queuing has been specified.
        5  Insufficient main storage was available to
           build a message queues DEB.
        6  Insufficient main storage was available to
           build a message queues IOB.
        7  The message queue was not formatted correctly
           prior to opening the Message Queues Data Set.

  descriptor is a single word describing the type of error.
  zzz - DD statement name

IED009I  CHECKPOINT DISK ALLOCATION
     ERROR - DATA SET NOT OPENED                 IGG01942      11

IED010I  CHECKPOINT - INSUFFICIENT               IEDQNR        11
     CORE ( ENVIRON           )                  IGG01941
          ) INCIDENT          (  xxx
          ) CKREQ name        (
          ( DATA SET NOT OPEN )
     xxx = number of bytes of main storage
           not available for the data set

IED011I  SYSTEM INTERVAL CANNOT BE               IEDQCZ        *
     ALTERED

IED012I TSO SESSION ON LINE xxx COMMAND          IEDQCV        *
     REJECTED

IED013I STOP REQUEST FOR SELF - VARY COMMAND          IEDQCV   *
     COMMAND REJECTED

IED014I  TCAM ALREADY IN SYSTEM                  IEDQOB        2,11

IED015I  TCAM AP OPEN ERROR 043-x yyy zzz
                                                 IGG01933      2,8,11

     x = 1  An application program OPEN has been issued,
            but there is no TCAM MCP active in the system.
         2  The QNAME= parameter of a DD statement for an
            application program is not the name of a
            process entry in the Terminal Table or the
            process entry named is inconsistent with the
            DCB format.
         3  A process entry named on a DD statement for an
            application program is currently being used by
            another DCB.
          4  Insufficient main stroage was available in the
             MCP to build internal control blocks.
          5  Insufficient main storage was avaiable in the
             application program area to build the internal
             control blocks.

      yyy = the name on the DD statement

      zzz = the job name

IED016I   STATION name NOT FOUND                      IEDQCF/G/H/N/  *
                                                      J/O/Q/U/V/X/Z
IED017I   LINE{ddname,rln}NOT OPEN                    IEDQCI/L/P/U/  *
              {address   }                            V/W/X/3

IED018I   command field COMMAND INVALID               IEDQCA/I/L/N/  *
                                                      P/U/V/W/X/Z/3
                                                      IGC0310D
         {termname     }
IED019I  {grpname,rln  }ALREADY STARTED               IEDQCO/Q/U     *
         {address      }

         {termname     }
IED020I  {grpname,rln  }STARTED                       IEDQCO/Q/U     *
         {address      }

IED021I   AUTO POLL STARTED FOR{grpname,rln}              IEDQCW     *
                               {address    }

IED022I   AUTO POLL ALREADY STARTED
          FOR {grpname,rln}                           IEDQCW         *
              {address    }

IED023I   TRACE STARTED FOR {grpname,rln}             IEDQCP         *
                            {address    }

IED024I   TRACE ALREADY STARTED FOR                   IEDQCP         *
      {grpname,rln}
      {address    }

         {termname    }
IED025I  {grpname,rln } ALREADY STOPPED               IEDQCO/Q/V     *
         {address     }

         {terminal    }
IED026I  {grpname,rln } STOPPED                       IEDQCO/Q/V     *
         {address     }

IED027I   AUTO POLL STOPPED FOR{grpname,rln}                IEDQCW   *
                               {address    }

IED028I   AUTO POLL ALREADY STOPPED
       FOR{grpname,rln}                               IEDQCW         *
          {address    }
```

```
IED029I  TRACE STOPPED FOR{grpname,rln }     IEDQCP        *
                          {address     }

IED030I  TRACE ALREADY STOPPED
     FOR{grpname,rln}
        {address    }                        IEDQCP        *

IED031I  statname QUEUE SIZE=integer,
     QUEUETYP=type, STATUS=status,...        IEDQCJ        *

IED032I {grpname,rln} LNSTAT=status,...
        {address    }
     ERR=error,...                           IEDQCI        *

IED033I  statname STATUS=status,...
     INTENSE={sense count}
             {NO         }
     IN-SEQ=integer,OUT-SEQ=integer          IEDQCH        *

IED034I  statname HAS NO opfldname OPTION   IEDQCF        *

IED035I  statname OPTION opfldname=data     IEDQCF        *

IED036I {grpname,rln}ACTIVE= statname...
        {address    }        NONE            IEDQCI        *

IED037I {grpname,rln}INACTIVE= statname...
        {address    }          NONE          IEDQCL        *

                                 {ddname     }
IED038I  statname IS ON LINE{grpname,rln}
                                 {address    }   IEDQCG        *

IED039I  NO STATIONS INTERCEPTED             IEDQCK        *

IED040I  INTERCEPTED STATIONS=statname,...   IEDQCK        *

IED041I  PRIMARY={statname}                  IEDQCN
                 {SYSCON  }                  IEDQCM        *

IED042I {statname} ALREADY PRIMARY           IEDQCN        *
        {SYSCON  }

IED043I  SECONDARY=statname                  IEDQCM        *

IED044I  statname NOT ELIGIBLE FOR PRIMARY   IEDQCN        *

IED045I  SYSTEM INTERVAL ALREADY ACTIVE      IEDQCZ        *

IED046I LINE FOR statname IS OUTPUT ONLY
     STATION                                 IEDQCO        *

IED047I  SYSTEM INTERVAL IS data             IEDQCZ        *
```

IED048I	POLLING DELAY FOR statname IS data	IEDQCZ	*
IED049I	OLT CONTROLS LINE line COMMAND REJECTED	IEDQCU	*
IED050I	statname OPTION opfield MODIFIED	IEDQCF	*
IED051I	statname SET FOR HOLD, SEQ-OUT=integer	IEDQCQ	*
IED052I	statname ALREADY SET FOR HOLD	IEDQCQ	*
IED053I	statname ALREADY RELEASED	IEDQCQ	*
IED054I	statname RELEASED,SEQ-OUT=integer	IEDQCQ	*
IED055I	I/O TRACE CANNOT BE ALTERED	IEDQCP	*
IED056I	termname opfldname DATA FORMAT INVALID	IEDQCF	*
IED057I	address NOT CAPABLE OF AUTO POLL	IEDQCW	*
IED058I	{grpname,rln address statname} SENSECOUNT =count, SETTING=sense	IEDQCX	*
IED059I	{grpname,rln address} LIST STATUS=status	IEDQC3	*
IED060I	statname CANNOT BE HELD	IEDQCQ	*
IED061I	POLLING DELAY FOR statname CANNOT BE ALTERED	IEDQCZ	*
IED062I	statname OPTION opfldname CANNOT ACCEPT SPECIFIED DATA	IEDQCF	*
IED063I	CLOSEDOWN IN PROGRESS -- xxx COMMAND REJECTED	IGC0110D	*
IED064I	LINE addr CONTROL UNIT NOT OPERATIONAL	IGE0204G	8
IED065I	INITIALIZATION ERROR return code	IEDQOA	2,11
IED066I	UNABLE TO OPEN SYSPRINT	IEDQXA	2
IED067I	TCAM INITIALIZATION BEGUN	IEDQXA	2
IED068I	UNABLE TO OPEN IEDODATA	IEDQXA	2

IED069I	INVALID KEYLEN FOR IEDQDATA	IEDQXA	2
IED070I	IEDQDATA DOES NOT SPECIFY CONTIG SPACE IN CYLINDERS	IEDQXA	2
IED071I	UNEQUAL PRIMARY AND SECONDARY EXTENTS ON IEDQDATA	IEDQXA	2
IED072I	I/O ERROR ON IEDQDATA	IEDQXA	2,10
IED073I	I/O ERROR ON SYSPRINT	IEDQXA	2,10
IED074I	TCAM INITIALIZATION COMPLETE	IEDQXA	2
IED075I	END OF EXTENT. RECORD COUNT IS number	IEDQXA	2
IED076I	TCAM NON-REUSABLE DISK THRESHOLD CLOSEDOWN	IGG019RC	2,11
IED077I	termname opfldname DATA CHARACTER INVALID	IEDQCF	*
IED078I	DLQ TERM ERROR	IEDQOM	11
IED079I	ENDING STATUS NOT RECEIVED FROM LINE address - LINE UNAVAILABLE	IGG01948	8
IED080I	START OF TCAM SYSTEM DELAY	IEDQHI	2
IED081I	END OF TCAM SYSTEM DELAY	IEDQHI	2
IED082I	CHECKPOINT DISK ERROR -- DATA SET NOT OPENED	IGG01942	11
IED083I	CHECKPOINT DISK ERROR -- RECOVERY FROM PREVIOUS RECORD	IGG01943	11
IED084I	CHECKPOINT DISK ERROR - RECOVERED	IEDQNQ	11
IED085I	CHECKPOINT DISK ERROR -{CKREQ INCIDENT} RECORD IGNORED	IEDQND IGG01944	11
IED086I	CHECKPOINT DISK ERROR -{ENVIRONMENT CKREQ,name}	IEDQNP	11
IED087I	CHECKPOINT DISK ERROR - CONTROL RECORD	IGG02041 IEDQNQ	11
IED088I	termname ON DIAL LINE - CANNOT BE VARIED	IEDQCO	*

IED089I	LINE ACTIVE - VARY TERMINAL COMMAND REJECTED	IEDQCO	*
IED090I	statname IS NOT SINGLE ENTRY	IEDQCG/O	*
IED091I	LINE FOR statname NOT OPEN	IEDQCG/J/O	*
IED092I	BISYNC ERROR - LINE xxx CANNOT BE STARTED	IEDQCU	*
IED093I	SET SYSTEM INTERVAL COMMAND ACCEPTED	IEDQCZ	*
IED094I	CORE REQUESTED FOR ON-LINE TEST NOT AVAILABLE	IEDQND	11
IED095I	MODIFY OLT REJECTED - OLT NOT ACTIVE	IEDQC2	8,11
IED096I	{ CHECKPOINT OPERATOR CONTROL COMWRITE } NO LONGER ACTIVE	IEDQNA2	2,11
IED097I	TCAM IS CLOSED DOWN	IEDQNA2	2,11,*
IED098I	DCB OPEN FOR MESSAGE PROCESSING PROGRAM - jobname	IEDQCO	2,11
IED099I	ROUTINE LOADED	IEDQC6	8
IED100I	ROUTINE DEACTIVATED	IEDQC6	8
IED101I	RESTART IN PROGRESS	IEDQC6	8
IED102I	INVALID OPERAND	IEDQC6	8
IED103I	ROUTINE IS ACTIVE	IEDQC6	8
IED104I	ROUTINE NOT ACTIVE	IEDQC6	8
IED105I	RETURN CODE=xxxx	IEDQC6	8
IED106I	MULTIPLE REQUEST	IEDQC6	8
IED107I	COMWRITE NOT ACTIVE	IEDQC6	8
IED109I	ROUTINE NOT DELETED	IEDQC6	8

TABLE OF CROSS-REFERENCES BETWEEN TCAM MODULES

Module Name	Entry Points	Entered From	External Routines	Exit Points	Exits To
IEDAYA	IEDAYA IEDAYA+12	IEDAYX IEDAYF	IEDQTNT	BZ after	DSPDISP in IGG019RB "ENTRY@12" or IGG019RO
				B after "ENTRY@12"	DSPCHAIN in IGG019RB or IGG019RO
				B after "RETURN"	DSPCHAIN in IGG019RB or IGG019RO
				BR after "RETURN2"	IEDAYM
IEDAYC	IEDAYC	IEDQUI (CARRIAGE)	IEDQTNT	BR after "EXIT"	IEDQLM (to MH)
IEDAYD	IEDAYD	IGG019RO or IGG019RB	IEDAYZ	B after "AYD200"	DSPBYPAS in IGG019RO or IGG019RB
IEDAYE	IEDAYE	IEDAYO IEDAYM	IEDQTNT, IEDAYS, QTIP SVC 101	BR after "EXIT"	IEDAYM, IEDAYO
IEDAYF	IEDAYF	IGG019RO or IGG019RB	IOHALT IEDQHG, DSPPRIOR in IGG019RO IGG019RB	B at "EXIT"	DSPDISP in IGG019RB or IGG019RO
				BR after "POSTSUB"	DSPCHAIN in IGG019RB or IGG019RO
IEDAYH	IEDAYH	IEDAYX	IEDQTNT	B after "SAYONARA"	DSPCHAIN in IGG019RB or IGG019RO
	IEDAYH+12	IGG019RO	IEDQTNT	B after "SAYONARA"	DSPCHAIN in IGG019RB or IGG019RO
IEDAYI	IEDAYI	IGG019RB or IGG019RO	IEDQTNT, QTIP SVC	B after "TSIN2050"	DSPCHAIN in IGG019RB or IGG019RO
IEDAYL	IEDAYL	IEDQUI (LOGON)	IEDQUI (IEDQAE) IEDQTNT	BR after "BYEBYE"	IEDQLM (Time Sharing Message Handler)
IEDAYM	IEDAYM, AYM000	IEDQBD, IEDAYS, IGG019RO or IGG019RB	IEDQTNT, IEDQUI (IEDQAE), IEDAYF	B after "AYM020" B after "AYM309"	DSPCHAIN in IGG019RB or IGG019RO

Module Name	Entry Points	Entered From	External Routines	Exit Points	Exits To
IEDAYO	IEDAYO IEDAYO002	IGG019RB or IGG019RO	IEDAYE, IEDAYM, IEDQFA, IEDQFQ, QTIP SVC	B after "CHAIN" B after "POST" B after "DISP"	DSBYPAS, DSPPOST, or DSPDISP in IGG019RB or IGG019RO
IEDAYP	IEDQAA01	IGG019RO or IGG019RB	IEDQUI (IEDQAI or IEDQAW), IEDQTNT	B after "POST"	DSPPOST in IGG019RB or IGG019RO
				BR after "MBHEXIT"	MH
IEDAYS	IEDAYS, IEDAYS2, IEDAYS3	IEDQUI IEDAYE IGG019RO or IGG019RB	IEDQTNT, QTIP/SVC 101	B after "AYS104"	IEDQLM (to MH)
				BNO after "AYSAYS3"	DSPDISP in IGG019RO or IGG019RB
				B after "AYS401"	IEDAYM
IEDAYT	IEDAYT0, IEDAYT1, IEDAYT2	STAE Exit Address	IGC102, TCABEND (SVC 94)	RETURN after "IEDAYT0", after "RETURN", and after "IEDAYT2"	ABEND/STAE Interface routine in OS Supervisor
IEDAYX	IEDAYX	IEDQBD	None	BR after "AYX050"	IEDAYA IEDAYH
IEDAYY	IEDAYY	IEDAYGP, IKJGGF94, IEDAYII	DSPPOSTP		DSPDISP in IGG019RO or IGG019RB
IEDAYZ	AYZ000	IGG019R3	IEDQTNT	BR after "AYZ042" B after "AYZ015"	IGG019R3
	AYZ100	IGG019R1	IEDQHG, DSPPOSTP	BCR after "AYZ100"	IGG019R1
				BR after "AYZ135" B after "AYZ125"	DSPPOST in IGG019RB or IGG019RO
	AYZ200	IGG019RO	DSPDISP	B after "AYZ060"	DSPDISP in IGG019RB or IGG019RO, IGG019R3
	AYZ300	QEVENT in IGG019R3	IEDQKA, EXCP	BR after "AYZ240" BR after "AYZ258" BR after "AYZ270" B after "AYZ280"	

Module Name	Entry Points	Entered From	External Routines	Exit Points	Exits To
	AYZ400	IGG019R4	DSPUNAVR IOHALT (SVC 33)	BCR after "AYZ300" BCR after "AYZ325" BCR after "AYZ330" BCR after "AYZ390" BR after "AYZ400" B after "AYZ495" BCR after "AYZ410"	 IGG019R4 DSPUNAV in IGG019RB or IGG019RO
	AYZ410	IEDAYD		BR after "AYZ460"	IEDAYD
	AYZ500	IGG019R4		BR after "AYZ500"	IGG019R4
	AYZ600	IEDQKA		B after "AYZ510" BR after "AYZ515" B after "AYZ665" BCR after "AYZ692" B after "AYZ045" B after "AYZ530" BR after "AYZ085"	IEDQKA DSPCHAIN in IGG019RB or IGG019RO DSPBYPAS in IGG019RB or IGG019RO IEDAYM
IEDQAA	IEDQAA01	IGG019RB or IGG019RO	IEDQTNT, IEDQUI (IEDQAW, IEDQAI)	B after "SETOTT" BR after "EXIT"	DSPPOST IEDQLM (STARTMH)
IEDQAC	IEDQAC01	IEDQUI (DATETIME)	IEDQAL, TIME IEDQAX	BR after "RETURN"	IEDQLM (DATETIME)
IEDQAD	IEDQAD01	IEDQUI (SEQUENCE)	IEDQUI (IEDQAF)	BR after "EXIT"	IEDQLM (SEQUENCE)
IEDQAE	IEDQAE	IEDQUI (INBUF, INHDR, INMSG, LOCOPT, MSGLIMIT, OUTBUF, OUTHDR, OUTMSG, or PATH) IEDQUI (IEDQAZ, IEDQA5, IEDQA7, IEDQA8 IEDQBT, IEDQYL)	IEDQTNT	BR after "EXIT"	IEDQLM (INBUF, INHDR, INMSG, LOCOPT, MSGLIMIT,OUTBUF, OUTHDR,OUTMSG or PATH) IEDQLM (IEDQAZ, IEDQA5,IEDQA7, IEDQA8,IEDQBT,IEDQYL)
IEDQAF	IEDQAF01	IEDQUI(DATETIME or SEQUENCE, IEDQAD, IEDQAK, IEDQAN, IEDQAO, IEDQAP, IEDQAT, IEDQA2, IEDQA8)	IEDQAL	BR after "EXIT"	IEDQLM(DATETIME or SEQUENCE, IEDQAD IEDQAK, IEDQAN, IEDQAP, IEDQAT, IEDQA2, IEDQA8)

Module Name	Entry Points	Entered From	External Routines	Exit Points	Exits To
IEDQAG	IEDQAG01	MSGLIMIT	None	BR after "ZERORTN"	MSGLIMIT
IEDQAH	IEDQAH01	IEDQUI (SEQUENCE)	IEDQTNT.	BR after 'STORE'	IEDQLM (SEQUENCE)
IEDQAI	IEDQAI01	IEDQUI(CODE,INITIATE, LOCK, MSGTYPE, ORIGIN, PATH, PRIORITY, SCREEN, SEQUENCE, SETEOF, SETSCAN, or UNLOCK, IEDQAA IEDQAN, IEDQA4, IEDQA5)	IEDQAL, IEDQAX	BR after "EXIT"	IEDQLM(CODE, INITIATE, LOCK, MSGTYPE, ORIGIN PATH, PRIORITY, SCREEN, SEQUENCE, SETEOF, SETSCAN, or UNLOCK, IEDQAA, IEDQAN, IEDQA4, IEDQA5)
IEDQAJ	IEDQAJ01	IEDQUI (FORWARD or SETSCAN) IEDQUI (IEDQAN or IEDQAP)	IEDQAL, IEDQAX	BR after "RETURN"	IEDQLM (FORWARD or SETSCAN) IEDQLM (IEDQAN or IEDQAP)
IEDQAK	IEDQAK01	IEDQUI (OUTMSG or OUTEND)	IEDQTNT, IEDQUI (IEDQAF, IEDQAO), IEDQAL	BR after "EXIT2"	IEDQA4
IEDQAL	ADDRCOMP	IEDQAC IEDQAF IEDQAI IEDQAJ, IEDQAO IEDQAK, IEDQAN IEDQAW IEDQA0, IEDQA3, IEDQA4 IEDQA5	None	BCR and BR after "ENTRLOOP"	IEDQAC IEDQAF IEDQAI IEDQAJ, IEDQAK, IEDQAN, IEDQAO, IEDQAW, IEDQA0,IEDQA3,IEDQA4, IEDQA5
IEDQAM	IEDQAM01	ORIGIN	IEDQTNT	BR after "SETBIT3"	ORIGIN
IEDQAN	IEDQAN01	IEDQUI (MSGEDIT)	IEDQTNT, IEDQAL, IEDQUI (IEDQAF, IEDQAI, IEDQAJ, IEDQAO)	BR after "ABCOMP"	IEDQLM (MSGEDIT)
IEDQAO	IEDQAO01	IEDQUI (IEDQAN IEDQAP, IEDQA2 IEDQA8)	IEDQAL, IEDQBW	BR BR after "INSERT"	IEDQLM (IEDQAN, IEDQAP,IEDQA2, IEDQA8) IEDQUI (IEDQAF)
IEDQAP	IEDQAP01	IEDQUI(MSGEDIT)	IEDQUI (IEDQAF IEDQAJ,	BR after "EXIT2"	IEDQLM (MSGEDIT)

Module Name	Entry Points	Entered From	External Routines	Exit Points	Exits To
			IEDQAO)		
IEDQAO	IEDQAO01	IEDQUI (CODE)	IEDQTNT	BCR	IEDQLM (CODE)
				B	"DSPPOST" in IGG019RB or IGG019RO
IEDQAR	IEDQAR	IEDQBD	IEDQTNT	B after "RETURN"	"DSPCHAIN" in IGG019RB or IGG019RO
IEDQAS	IEDQAS IEDQAS01 GETCPB LCBRTN	IEDQBD IGG019RB or IGG019RO IGG019RB or IGG019RO IGG019RB or IGG019RO	IEDQTNT DSPPOSTR, IEDQHG02, IEDQHG01,		DSPDISP or DSPCHAIN in IGG019RB or IGG019RO IEDQFQ
IEDQAT	IEDQAT01 STCBAT+2	IEDQAZ IGG019RB or IGG019RO	IEDQUI (IEDQAF)	BR after "POSTSUB"	DSPCHAIN in IGG019RB or IGG019RO
IEDQATTN	IEDATTN	I/O Supervisor	None		IGG019R5 IOS
IEDQAU	IEDQAU CUTFFQCB+12	IEDQUI (CUTOFF) IGG019RB or IGG019RO	None EXCP		IEDQLM (CUTOFF) IGG019RB or IGG019RO
IEDQAV	IEDQAV01	IEDQUI (FORWARD)	IEDQTNT	BR after "LOADPFX"	IEDQLM (FORWARD)
		IEDQUI(IEDQAZ,IEDQA5)		BCR after "LINKINT"	IEDQLM (IEDQAZ, IEDQA5)
IEDQAW	IEDQAW01	IEDQUI (CODE)	IEDQAL, IEDQA3	BR after "EXIT"	IEDQLM (CODE)
		IEDQUI (IEDQAA)			IEDQLM (IEDQAA)
IEDQAX	SCAN	IEDQAC IEDQAI IEDQAJ IEDQA4	None	BCR, BL, and B to return to caller	IEDQAC IEDQAI IEDQAJ IEDQA4
IEDQAY	IEDQAY01	IEDQUI (SCREEN)	IEDQTNT	BR after "EXIT"	IEDQLM (SCREEN)
IEDQAZ	IEDQAZ01	IEDQBD	IEDQUI (IEDQAF, IEDQA1)	BR after "ERRMSG"	IEDQAT
			IEDQAV	B after "MOVEQCB"	DSPCHAIN in IGG019RB or IGG019RO
IEDQA0	IEDQA001	IEDQUI (SETSCAN)	IEDQAI	BR at "RETURN"	IEDQLM (SETSCAN)

Module Name	Entry Points	Entered From	External Routines	Exit Points	Exits To
IEDQA1	IEDQA101	IEDQUI (ORIGIN) IEDQUI(IEDQAZ,IEDQAS, IEDQE1,IEDQE2,IEDQE3,IGG01946, IGG019RI,IGG019RJ)	None	BR at "RETURN"	IEDQLM (ORIGIN) IEDQLM(IEDQAZ,IEDQAS, IEDQE1,IEDQE2, IEDQE3,IGG01946, IGG019RI,IGG019RJ)
IEDQA2	IEDQA201	IEDQUI (MSGEDIT)	IEDQUI(IEDQAF, IEDQAO)	BR after "ABEXIT"	IEDQLM (MSGEDIT)
IEDQA3	IEDQA3	IEDQAW	IEDQAL IEDQUI (IEDQAE)	BR at "NOXLTI"	IEDQAW
IEDQA4	IEDQA401	IEDQUI (INEND, INMSG, OUTEND, or OUTMSG)	IEDQAL, IEDQAX, IEDQTNT, DSPPOSTR	BCR after "SELEXIT" B after "EXIT"	IEDQGD IEDQGT DSPPOST in IGG019RB or IGG019RO
IEDQA5	IEDQA501	IEDQUI (FORWARD) IEDQUI(IEDQBA)	IEDQUI (IEDQAE, IEDQAI, IEDQA1) IEDQAL IEDQAV	BR after "EXIT"	IEDQLM (FORWARD) IEDQLM(IEDQBA)
IEDQA6	IEDQA601	IEDQUI (MSGFORM)	IEDQTNT	BR after "EXITOFF"	IEDQLM (MSGFORM)
IEDQA7	IEDQA701	COUNTER	IEDQUI (IEDQAE)	BR after "ERROR"	COUNTER
IEDQA8	IEDQA801	IEDQUI (MSGEDIT)	IEDQUI (IEDQAE IEDQAF, IEDQAO)	BR after "EXIT"	IEDQLM (MSGEDIT)
IEDQBA	IEDQBA01	IGG019RB or IGG019RO	IEDQUI (IEDQA5)	B after "POSTBFR" B after "EOAENTER" B after "MBHLINK"	DSPPOST or DSPCHAIN in IGG019RB or IGG019RO
IEDQBB	IEDQBB	IEDQUI (CHECKPT) IEDQBD	None		IEDQLM (CHECKPT) DSPPOST in IGG019RB or IGG019RO
IEDQBC	IEDQBC	IGG019RB or IGG019RO	IEDQTNT		DSPCHAIN in IGG019RB or IGG019RO

Module Name	Entry Points	Entered From	External Routines	Exit Points	Exits To
IEDQBD	IEDQBD01	IGG019RB or IGG019RO	IEDQTNT		DSPCHAIN in IGG019RB or IGG019RO IEDQNX
	IEDQBD02	IGG019RB or IGG019RO	IEDQTNT		DSPCHAIN in IGG019RB or IGG019RO
IEDQBE	IEDQBE	IEDQUI (LOCK)	IEDQTNT		IEDQLM (LOCK)
IEDQBF	IEDQBF	IEDQUI (UNLOCK)	None		IEDQLM(UNLOCK)
IEDQBG	IEDQBG	IGG019RB or IGG019RO	IEDQTNT		DSPPOST in IGG019RB IGG019RO
IEDQBL	IEDQBL	IEDQBD	None		DSPCHAIN in IGG019RB or IGG019RO
IEDQBT	IEDQBT	IGG019RB or IGG019RO	IEDQUI (IEDQAE)		DSPBYPAS or DSPCHAIN in IGG019RB or IGG019RO
IEDQBW	IEDQBW	IEDQAO	None		IEDQAO
IEDQBX	IEDQBX	IEDQUI (LOG)	WRITE CHECK GETMAIN	BR after 'EXIT'	IEDQLM (LOG)
IEDQBY	IEDQBY	IEDQBD	None		DSPCHAIN in IGG019RB or IGG019RO
IEDQBZ	IEDQBZ	IGG019RB or IGG019RO	GETMAIN CHECK WRITE DSPPOSTR DSPUNAVR		DSPBYPAS or DSPPOST in IGG019RB or IGG019RO
IEDQCA	IEDQCA	OS Task Management	IGC0010D		OS Task Management
	IEDQCA02	Transient Operator Control Routines	IGC0010D		Transient Operator Control Routines
IEDQCF	IEDQCF	IGC0110D	IEDQC5		IGC0110D
IEDQCG	IEDQCG	IGC0110D	None		IGC0110D
IEDQCH	IEDQCH	IGC0110D	None		IGC0110D
IEDQCI	IEDQCI	IGC0110D	None		IGC0110D
IEDQCJ	IEDQCJ	IGC0110D	None		IGC0110D
IEDQCK	IEDQCK	IGC0110D	None		IGC0110D

Module Name	Entry Points	Entered From	External Routines	Exit Points	Exits To
IEDQCL	IEDQCL	IGC0110D	None		IGC0110D
IEDQCM	IEDQCM	IGC0110D	None		IGC0110D
IEDQCN	IEDQCN	IGC0110D	IEDQNX		IGC0110D
IEDQCO	IEDQCO	IGC0110D	None		IGC0110D
IEDQCP	IEDQCP	IGC0110D	None		IGC0110D
IEDQCQ	IEDQCQ	IGC0110D	IGC102 WAIT		IGC0110D
IEDQCU	IEDQCU	IGC0110D	IGC102 EXCP		IGC0110D
IEDQCV	IEDQCV	IGC0110D	IGC102 WAIT		IGC0110D
IEDQCW	IEDQCW	IGC0110D	None		IGC0110D
IEDQCX	IEDQCX	IGC0110D	None		IGC0110D
IEDQCZ	IEDQCZ	IGC0110D	IGC102		IGC0110D
IEDQC0	IEDQC0	IGG019RB or IGG019RO IGC0110D	IGC102		IGG019RB or IGG019RO
IEDQC1	IEDQC1	IGC0110D	IGC102		IGC0110D
IEDQC2	IEDQC2	IGC0110D	IGC102 WAIT		IGC0110D
IEDQC3	IEDQC3	IGC0110D	None		IGC0110D
IEDQC6	IEDQC6	IGC0110D	IEDQCA02 Service Aid Routines LOAD DELETE		IGC0110D
IEDQEC	IEDQEC	IGG019RB or IGG019RO	DSPPOSTR in IGG019RB or IGG019RO	B after "REQUEST" B after "NORECDEL"	DSPCHAIN in IGG019RB or IGG019RO DSPDISP in IGG019RB or IGG019RO

Module Name	Entry Points	Entered From	External Routines	Exit Points	Exits To
			IGC102	B after "ECBPOST"	DSPDISP in IGG019RB or IGG019RO
				B after "SAVIT"	DSPDISP in IGG019RB or IGG019RO
IEDQES	IEDQES	RETRIEVE	IEDQUI (IEDQA1) IGC102		RETRIEVE
IEDQET	IEDQET	ICHNG, RELEASEM, MRELEASE, STARTLN, STOPLN, MCPCLOSE, or CLOSEMC	IGC102 IEDQE6		Next instruction in the application program
IEDQEU	IEDQEU	IGG019RB or IGG019RO	IGC102 GETMAIN FREEMAIN		IGG019RB or IGG019RC
IEDQEW	IEDQEW	IGG019RB or IGG019RO	IGC102		IGG019RB or IGG019RC
IEDQEZ	IEDQEZ	IGG019RB or IGG019RO	None		IGG019RB or IGG019RO
IEDQE1	IEDQE1	TCOPY	IEDQUI (IEDQA1)		TCOPY
IEDQE2	IEDQE2	QCOPY	IEDQUI (IEDQA1)	RETURN after "EXIT"	Next sequential instruction or macro expansion
IEDQE3	IEDQE3	TCHNG	IEDQE6, IEDQUI (IEDQA1), IEDQNB, IGC102	RETURN after "EXIT"	TCHNG
IEDQE4	IEDQF4	ICOPY	None	RETURN after "EXIT"	ICOPY
IEDQE6	IEDQE6	OS Link (IEDQOA) IEDQE3 IEDQET	None		IEDQOA IEDQE3
IEDQE7	IEDQE7	IGG019RB or IGG019RO	IGC102		DSPDISP in IGG019RB or IGG019RO
IEDQFA	IEDQFA	IGG019RB or IGG019RO IEDQFO			IGG019RB or IGG019RO IGG019RC
IEDQFA1	IEDQFA1	IGG019RO or IGG019RO IEDQFO			IGG019RB or IGG019RO IGG019RC

Module Name	Entry Points	Entered From	External Routines	Exit Points	Exits To
IEDQFA2	IEDQFA2	IGG019RO or IGG019RB IEDQFO			IGG019RB or IGG019RO IGG019RC
IEDQGA	IEDQGA	IGG019RB or IGG019RO	DSPLIFOR		IGG019RB or IGG019RO
IEDQGA	IEDQGB	IGG019RB or IGG019RO	DSPPOSTR DSPLIFOR		IGG019RB or IGG019RO
IEDQGA	IEDQGD	IGG019RB or IGG019RO	EXCP		IGG019RB or IGG019RO
IEDQGT	IEDQGT	OUTBUF	None		OUTBUF
IEDQHG	IEDQHG	IGG019RB or IGG019RO	TIME STIMER		IGG019RB or IGG019RO
	IEDQHG01	TCAM Subtasks	IGC102		Calling Subtask
	IEDQHG02	TCAM Subtasks	DSPPOSTR DSPPRIOR		Calling Subtask
	IEDQHG03	IGG019RB or IGG019RO			IGG019RB or IGG019RO
	TIMEEXIT	OS Interrupt Routine			
IEDQHI	IEDQHI	IGG019RB or IGG019RO	IEDQHG01 IEDQHG02 IOHALT DSPPRIOR WTO		IGG019RB or IGG019RO
IEDQHK	IEDQHK	IGG019RB or IGG019RO	None		IGG019RB or IGG019RO
IEDQHM	IEDQHM	IGG019RB or IGG019RO	IEDQTNT		IGG019RB or IGG019RO
	IEDQHM02	IGG019RD IGG019RP	Scheduler Subroutine		IGG019RP
IEDQHM1	IEDQHM1	IGG019RB or IGG019RO	IEDQTNT		IGG019RB or IGG019RO
	IEDQHM02	IGG019RD IGG019RP	Scheduler Subroutine		IGG019RP
IEDQHM2	IEDQHM2	IGG019RB or IGG019RO	IEDQTNT		IGG019RB or IGG019RO
	IEDQHM02	IGG019RD IGG019RP	Scheduler Subroutine		IGG019RP
IEDQKA	IEDQKA	IGG019RB or IGG019RO	EXCP		IGG019RB or IGG019RO
	IEDQKA02	IGG019RO	IEDQTNT		
IEDQKB	IEDQKB IEDQKA02	IGG019RB or IGG019RO	EXCP, IEDQTNT		IGG019RB or IGG019RO
IEDQKC	IEDQKC IEDQKA02	IGG019RB or IGG019RO	EXCP, IEDQTNT		IGG019RB or IGG019RO
IEDQKD	IEDQKD IEDQKA02	IGG019RB or IGG019RO	EXCP, IEDQTNT		IGG019RB or IGG019RO

Module Name	Entry Pcints	Entered From	External Routines	Exit Points	Exits To
IEDQKE	IEDQKE IEDQKA02	IGG019RB or IGG019RO	EXCP IEDQTNT		IGG019RB or IGG019RO
IEDQLM	IEDQLM	Message handling	None	Last executable instruction	Message handling macro expansions and routines
IEDQNA	IEDQNA	IGG019RB or IGG019RO	IEDQNA2	RETURN after "QNA00"	User code following READY
	IEDQNA3	OS Termination routine	IEDQNA2	RETURN after "QNA40'	OS Termination routine
IEDQNA2	IEDQNA2	IEDQNA	OS Task Manage-ment, DETACH, DSPPOSTR	BR after "QNA60" or BR after "QNA90"	IEDQNA
IEDQNB	IEDQNB	CKREQ	IEDQTNT, IGC102	RETURN after "QNB30"	CKREQ
IEDQNB	IEDQNB02	IEDQE3	IGC102, IEDQTNT	RETURN after "QNB30"	IEDQE3
IEDQND	IEDQND	READY	ATTACH GETMAIN FREEMAIN IEDQTNT EXCP LOAD POST WTO WAIT IECPCNVT	BR after 'QND84'	READY
IEDQNF	IEDQNF	OS Task Management	IEDQNG, IEDQNH, IEDQNJ, IEDQNK, IEDQNM, IEDQNO, IEDQNP, IEDQNQ, IEDQNR, IEDQNS, OS Contents Supervisor, OS Task Management	RETURN after "IEDQNF90"	OS Task Management

Module Name	Entry Points	Entered From	External Routines	Exit Points	Exits To
IEDQNG	IEDQNG	IEDQNF	IEDQTNT, OS Contents Supervisor	BR after "IEDQNG40" BR after "IEDQNG50" BR after "IEDQNG60" BR after "IEDQNG70"	IEDQNF IEDQNF IEDQNF IEDQNF
IEDQNH	IEDQNH	IEDQNF	IEDQTNT, OS Contents Supervisor	BR after "IEDQNH50" BR after "IEDQNH60" BR after "IEDQNH70"	IEDQNF IEDQNF IEDQNF
IEDQNJ	IEDQNJ	IEDQNF	None	BR after "QNJ25", "QNJ30", or "QNJ40"	IEDQNF
IEDQNK	IEDQNK	IEDQNF	OS Contents Supervisor	BR after "QNK30" BR after "QNK81"	IEDQNF IEDQNF
IEDQNM	IEDQNM	IEDQNF	OS Contents Supervisor IEDQTNT	BR after "QNM60" BR after "QNM65"	IEDQNF IEDQNF
IEDQNO	IEDQNO	IEDQNF	OS Contents Supervisor	BNO after "QNO20" BZ after "QNO50" BE after "QNO60"	IEDQNF register 14 IEDQNP register 14 plus 4 IEDQNF register 14 plus 4
IEDQNP	IEDQNP	IEDQNF	TIME OS DISK Address Convert Routine, OS Data Management	B after "QNP60" B after "QNP98"	IEDQNF register 14 plus 4 IEDQNF register 14
IEDQNQ	IEDQNQ	IEDQNF	IGC102		IEDQNF plus 4 IEDQNF plus 0
IEDQNR	IEDQNR	IEDQNF	OS Data Management	BNZ after "QNR00"	IEDQNF plus 8

Module Name	Entry Points	Entered From	External Routines	Exit Points	Exits To
				BR after "QNR40"	IEDQNF plus 0
IEDQNS	IEDQNS	IEDQNF	IGC102	BO after "QNS00"	IEDQNF plus 4
			IEDQHG03		
				B after "QNS20"	IEDQNF plus 4
IEDQNX	IEDQNX	IEDQBD	IEDQTNT, IEDQUI	BR after "QNX40"	DSPCHAIN in IGG019RB or IGG019RO
IEDQOA	IEDQOA	INTRO	LINK (IEDQOB, IEDQOG, IEDQOM, IEDQOS, IEDQE6), WTO	RETURN after "EXIT"	INTRO
IEDQOB	IEDQOB	OS Link (IEDQOA)	WTO WAIT	RETURN at "GETMAIN"	IEDQOA
IEDQOG	IEDQOG	OS Link (IEDQOA)	GETMAIN	RETURN after "RTEND"	IEDQOA
IEDQOM	IEDQOM	OS Link (IEDQOA)	GETMAIN, WTO	RETURN after "RTEND"	IEDQOA
IEDQOS	IEDQOS	OS Link (IEDQOA)	ATTACH, EXTRACT LOAD		IEDQOA
IEDQTNT	IEDQTNT	IEDQAE IEDQAG IEDQAH IEDQAM IEDQAS IEDQAV IEDQAY IEDQA4 IEDQA6 IEDQBC IEDQBD IEDQBG IEDQHM IEDQNB IEDQNG IEDQNH IGG019R1 IGG019R4 IEDQNM IEDQNX IEDQKA	None		IEDQAE IEDQAG IEDQAH IEDQAM IEDQAS IEDQAV IEDQAY IEDQA4 IEDQA6 IEDQBC IEDQBD IEDQBG IEDQHM IEDQNB IEDQNG IEDQNH IGG019R1 IGG019R4 IEDQNM IEDQNX IEDQKA

Module Name	Entry Points	Entered From	External Routines	Exit Points	Exits To
		IEDQKB IEDQKC IEDQKD IEDQKE IGE0004G IGE0004H IGE0504G IGE0504H IGG019R0 IGG019Q2 IGG019Q3 IGG019Q4 IGG019Q5 IGG019RP			IEDQKB IEDQKC IEDQKD IEDQKE IGE0004G IGE0004H IGE0504G IGE0504H IGG019R0 IGG019Q2 IGG019Q3 IGG019Q4 IGG019Q5 IGG019RP
IEDQUI	IEDQUI01	Message handling macro expansions and routines	None	BR after "ROUTADDR"	Message handling routines
IEDQXA	IEDQXA	OS Task Management	SVC 64 BSAM WTO, GETMAIN, FREEMAIN, SVC 31	RETURN after "EXIT"	OS Task Management
IGC0010D	IGC0010D	IEDQCA, IGC0210D, or IGC0410D	WAIT		IEDQCA, IGC0110D, IGC0210D, or IGC0410D
IGC0110D	IGC0110D	IGC0010D, IGC0210D, IGC0310D, or IGC0410D	QEDIT, WTO, LOAD, DELETE, Operator Control Processing routines		IGC0010D, IGC0210D, IGC0310D, or IGC0410D
IGC0210D	IGC0210D	IGC0110D or IGC0410D	IGC0010D		IGC0110D or IGC0310D
IGC0310D	IGC0310D	IGC0110D, IGC0210D, or IGC0410D	IGC102, WAIT, WTO, DELETE, QEDIT		IGC0110D, IGC0410D, or IGC0510D
IGC0410D	IGC0410D	IGC0110D, IGC0310D, or IGC0510D	IGC102, IGC0010D, DELETE		IGC0110D, IGC0210D, or IGC0310D
IGC0510D	IGC0510D	IGC0310D	IGC102, WAIT		IGC0410D
IGC102	IGC102	SVC 102 from any Routine in the System	POST, IKJTSI00 STATUS, TESTDSP		Calling routine

Module Name	Entry Points	Entered From	External Routines	Exit Points	Exits To
IGC1303D	IGC1303D	SVC 34	GETMAIN, FREEMAIN, QEDIT		IGC0503D Address in register 14
IGE0004G	IGE0004G	OS I/O Supervisor	IEDQTNT		IGE0104G IGE0204G IGE0304G IGE0404G IGE0504G IGE0604G IGE0804G IGE0904G IGG019R0 IGE0025F
IGE0104G	IGE0104G	IGE0004G	None		IGE0504G IGG019R0
IGE0204G	IGE0204G	IGE0004G IGE0004H	WTO		IGE0504G IGG019R0
IGE0304G	IGE0304G	IGE0004G	None		IGE0504G
IGE0404G	IGE0404G	IGE0004G IGE004H	None		IGE0504G IGG019R0
IGE0504G	IGE0504G	IGE0004G IGE0104G IGE0204G IGE0304G IGE0404G IGE0604G	IEDQTNT		OS Message Writer
IGE0604G	IGE0604G	IGE0004G	None		IGE0504G
IGE0804G	IGE0804G	IGE0004G	None		IGE0804G
IGE0904G	IGE0904G	IGE0004G	IGE0025F		IGG019R0
IGE0004H	IGE0004H	OS I/O Supervisor	IEDQTNT		IGE0104H IGE0204G IGE0204H IGE0304G IGE0404G IGE0404H IGE0504H IGE0804H IGG019R0
IGE0104H	IGE0104H	IGE0004H	None		IGE0504H IGG019R0
IGE0204H	IGE0204H	IGE0004H	None		IGE0404H IGE0504H

Module Name	Entry Points	Entered From	External Routines	Exit Points	Exits To
IGE0404H	IGE0404H	IGE0004H IGE0204H	None		IGE0504H IGG019R0
IGE0504H	IGE0504H	IGE0004H IGE0104H IGE0204H IGE0404H IGE0804H	IEDQTNT		IGE0025F
IGE0804H	IGE0804H	IGE0004H	None		IGE0504H
IGG019Q0	IGG019Q0	IGG019R0	None		IGG019R0 or User Trace Exit routine
IGG019Q1	IGG019Q1	IGG019RB or IGG019RO	None		IGG019RB or IGG019RO
IGG019Q2	IGG019Q2	I/O Supervisor	POST IEDQTNT, IEDQKA02, TESTDSP		I/O Supervisor
IGG019Q3	IGG019Q3	I/O Supervisor	POST IEDQTNT, IEDQKA02, TESTDSP		I/O Supervisor
IGG019Q4	IGG019Q4	I/O Supervisor	POST IEDQTNT, IEDQKA02, TESTDSP		I/O Supervisor
IGG019Q5	IGG019Q5	I/O Supervisor	POST IEDQTNT, IEDQKA02, TESTDSP		I/O Supervisor
IGG019Q6	IGG019Q6	IGG019RB or IGG019RO	IOHALT, EXCP		IGG019RB or IGG019RO
IGG019Q7	IGG019Q7	IGG019RB or IGG019RO	IEDQTNT, IOHALT, EXCP		IGG019RB or IGG019RC
IGG019Q8	IGG019Q8 IGG019Q8+4 IGG019Q8+8 IGG019Q8+12 IGG019Q8+16	IGG01945	IGG019RC, WAIT		IGG01945
IGG019RA	IGG019RA	OS I/O Supervisor	None	ECR	OS I/O Supervisor return address

Module Name	Entry Points	Entered From	External Routines	Exit Points	Exits To
				B	OS I/O Supervisor return address plus 8
IGG019RB or IGG019RO	DSPBYPAS DSPCHAIN DSPDLETE DSPDISP IGG019RB+16 or IGG019RO+16 DSPLIFO DSPLIFOR DSPLIST DSPPOST DSPPOSTR DSPPRIO DSPPRIOR DSPWAIT DSPTSTQ DSPTSTR DSPUNAV DSPUNAVR	Any TCAM Subtask	WAIT POST DELETE		Any TCAM Subtask
IGG019RC	IGG019RC	IEDQFA	OS I/O Supervisor	B at "EXIT"	DSPDISP in IGG019RB or IGG019RO
		IGG019Q8			IGG019Q8
			OS Data Management		
IGG019RD	IGG019RD	IGG019RB or IGG019RO IEDQHM	DSPPOSTR, IEDQHG01, IEDQTNT		DSPBYPAS, DSPPOST, or DSPDISP in IGG019RB or
IGG019RF	IGG019RF	IEDQFA	OS I/O Supervisor	B at "EXIT"	DSPDISP in IGG019RB or IGG019RO

Module Name	Entry Points	Entered From	External Routines	Exit Points	Exits To
IGG019RG	IGG019RG	GET/READ	WAIT, IGC102		GET/READ
IGG019RH	IGG019RH	GET	WAIT, IGC102		GET
IGG019RI	IGG019RI	PUT/WRITE	IEDQUI (IEDQA1), WAIT IGC102		PUT/WRITE
IGG019RJ	IGG019RJ	PUT	IEDQUI (IEDQA1), WAIT IGC102		PUT
IGG019RK	IGG019RK	OS I/O Supervisor	POST "EXIT"	BR after	I/O Supervisor
IGG019RL	IGG019RL	CHECK	IGG019RG OS WAIT		CHECK
IGG019RM	IGG019RM	POINT	IEDQUI		Next sequential application program instruction
IGG019RN	I/O Supervisor		IGG019R0, IEDQTNT		OS Post Routine (I/O Supervisor)
IGG019RO - see IGG019RB					
IGG019RP	IGG019RP	IEDQFA IGG019RB or IGG019RO	IEDQHM02, IEDQHM03, IEDQTNT, IEDQHG02	BR15 at "EXIT"	IEDQFA02
IGG019RQ	IGG019RQ	Rollout/Rollin Routine IEAQRORI	POST		Rollout/Rollin Routine IEAQRORI
IGG019R0	IGG019R0 SCAN	I/O Supervisor IGG019RN	POST IEDQTNT, IEDQKA02, TESTDSP IGG019Q0		I/O Supervisor
IGG019R1	IGG019R1	IGG019RB or IGG019RO	IEDQTNT, IEDQHG01, IGG019RB, TIME, EXCP		IGG019RB or IGG019RO
IGG019R2	IGG019R2	OS I/O Supervisor	POST	BR after "EXIT"	OS I/O Supervisor

Module Name	Entry Points	Entered From	External Routines	Exit Points	Exits To
IGG019R3	IGG019R3	IGG019RB or IGG019RO	DSPUNAVR,	B after "ID"	DSPBYPAS in IGG019RB or IGG019RO
	QEVENT	IGG019RB or IGG019RO	IEDAYZ, POST	B after "ACTIV"	DSPPOST in IGG019RB or IGG019RO
				BNZ after "ENDOLIST"	DSPPOST in IGG019RB or IGG019RO
				B after "LCBPOST"	DSPPOST in IGG019RB or IGG019RO
				B after "FREELINE"	DSPPOST in IGG019RB or IGG019RO
IGG019R4	IGG019R4	IGG019RB or IGG019RO	IEDQTNT, IGGR19RB IOHALT, IEDAYZ, EXCP		IGG019RB or IGG019RO
IGG019R5	IGG019R5	IEDQATTN	OS Post, TESTDSP		I/O Supervisor
IGG019R6	IGG019R6	IGG019RB or IGG019RO	IGG019RC, IEDQHM02 IEDQTNT OS WAIT		DSPPOST or DSPDLETE in IGG019RB or IGG019RO
IGG01930	IGG01930	System Open Monitor or another open executor	GETMAIN	XCTL macro after "XCTLRTN"	Another open executor-IGG01931 IGG01933
IGG01931	IGG01931	System Open Monitor or another open executor (from IGG01930)	GETMAIN	XCTL macro after "XCTLRTN"	Another open executor - IGG01934 IGG01933
IGG01933	IGG01933	Any TCAM Open Executor		OS WTO OS SYNCH	ABEND; any TCAM open executor
IGG01934	IGG01934	System Open Monitor or another open executor (from IGG01931)	LOAD	XCTL macro after "XCTLRTN"	Another open executor - IGG01941 or IGG01935
IGG01935	IGG01935	System Open Monitor or another open executor	GETMAIN	XCTL macro	Another open executor - IGG01936 IGG01933
IGG01936	IGG01936	System Open Monitor or another open executor (from IGG01935)	GETMAIN	XCTL macro	Another open executor - IGG01937 IGG01933

Module Name	Entry Points	Entered From	External Routines	Exit Points	Exits To
IGG01937	IGG01937	System Open Monitor or another open executor (from IGG01936)	None	XCTL macro	Another open executor - IGG01938
IGG01938	IGG01938	System Open Monitor or another open executor (from IGG01937)	None	XCTL macro	Another open executor - IGG01939
IGG01939	IGG01939	System Open Monitor or another open executor (from IGG01938)	LOAD EXCP	XCTL macro	Another open executor IGG01940
IGG01940	IGG01940	System Open Monitor or another open executor (from IGG01939)	LOAD	XCTL macro	Another open executor-IGG01948
IGG01941	IGG01941	IGG01934	GETMAIN WTO LOAD EXCP	XCTL macro at "QMA70"	IGG01949 for Cold Restart or IGG01943 for Warm Restart IGG0190S
IGG01942	IGG01942	IGG01949	IECPCNVT WTO, EXCP, WAIT	XCTL macro at "QMB89"	IGG0190S
IGG01943	IGG01943	IGG01941	IECPCNVT EXCP, WAIT, WTO	XCTL after "QME879"	IGG01944
IGG01944	IGG01944	IGG01943	IECPCNVT, IEDQTNT, EXCP WAIT, WTO	XCTL after "QMG87"	IGG01945 IGG0190S
IGG01945	IGG01945	IGG01944	IGG019Q8, LOAD, IEDQTNT	XCTL after "QMJ87"	Next Module in the System Where-to-Go Table
IGG01946	IGG01946	System Open Executor IGG01933	IEDQUI (IEDQA1), IGC102		IGG01947
IGG01947	IGG01947	IGG01946	IEDQNB05		Next entry in system Where-to-Go Table
IGG01948	IGG01948	IGG01940	TIME, WTO		IGG0190S

Module Name	Entry Points	Entered From	External Routines	Exit Points	Exits To
IGG01949	IGG01949	IGG01941	None	XCTL after "QMM75"	IGG01942
IGG02030	IGG02030	System I/O Support module	System Loader	XCTL macro	Another Close executor
IGG02035	IGG02035	System Close Module	System Loader	XCTL macro	Another Close executor
IGG02036	IGG02036	System Close Module	System Loader	XCTL macro	Another Close executor
IGG02041	IGG02041	System Close Module	IECPCNVT		Another Close executor
IGG02046	IGG02046	System Close Executor	IGC102 IEDQNB05	XCTL after "QL750"	Next entry in the system Where-to-Go Table - IGG02047
IGG02047	IGG02047	IGG02046	IGC102	XCTL	Next entry in the system Where-to-Go Table

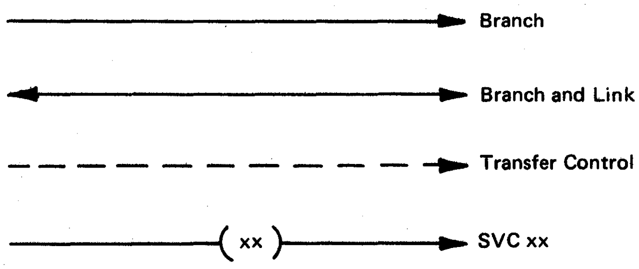
TCAM LINKAGES BETWEEN MACRO EXPANSION AND MODULES

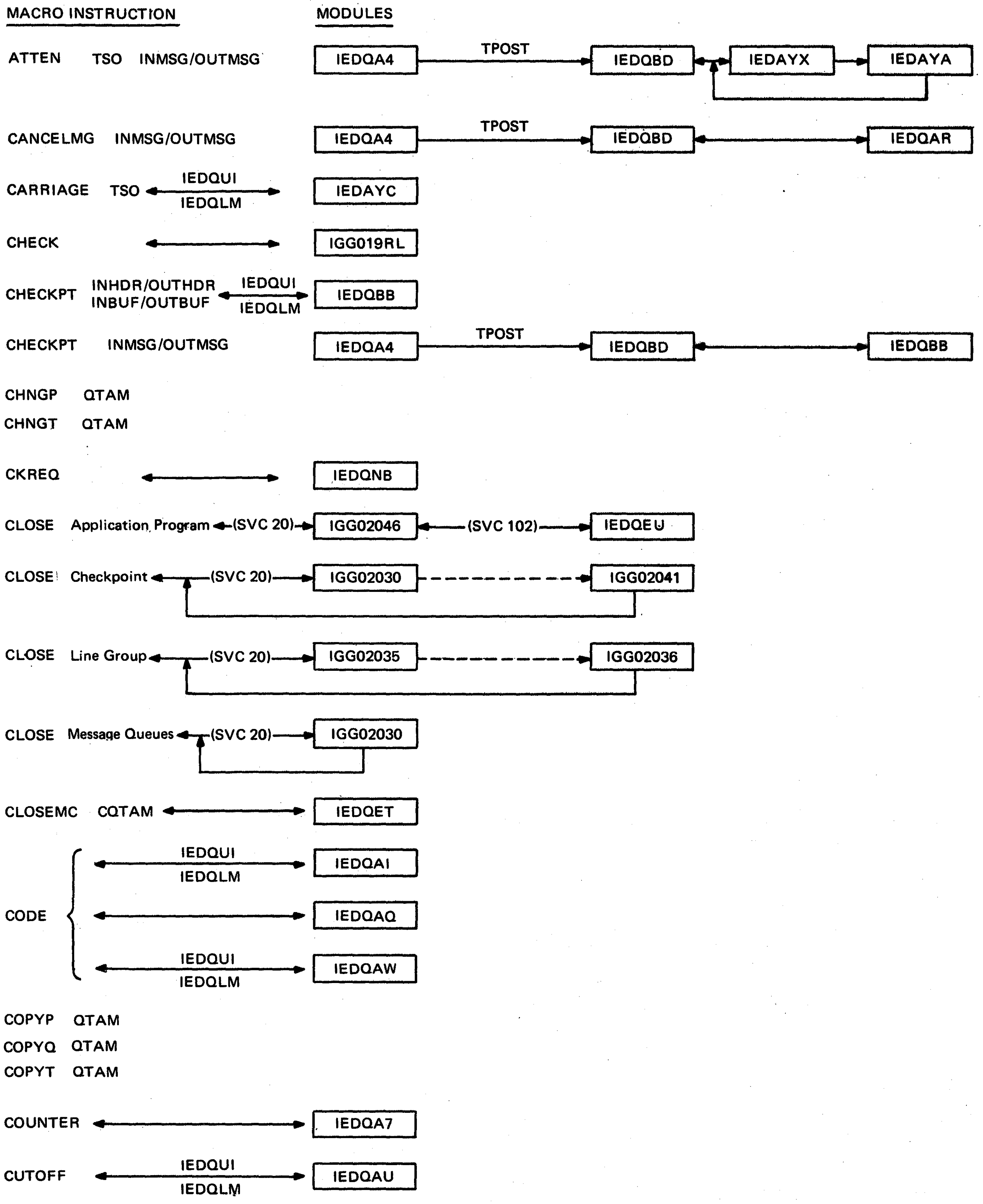
This chart depicts the linkages between the macro expansion and the modules they call.

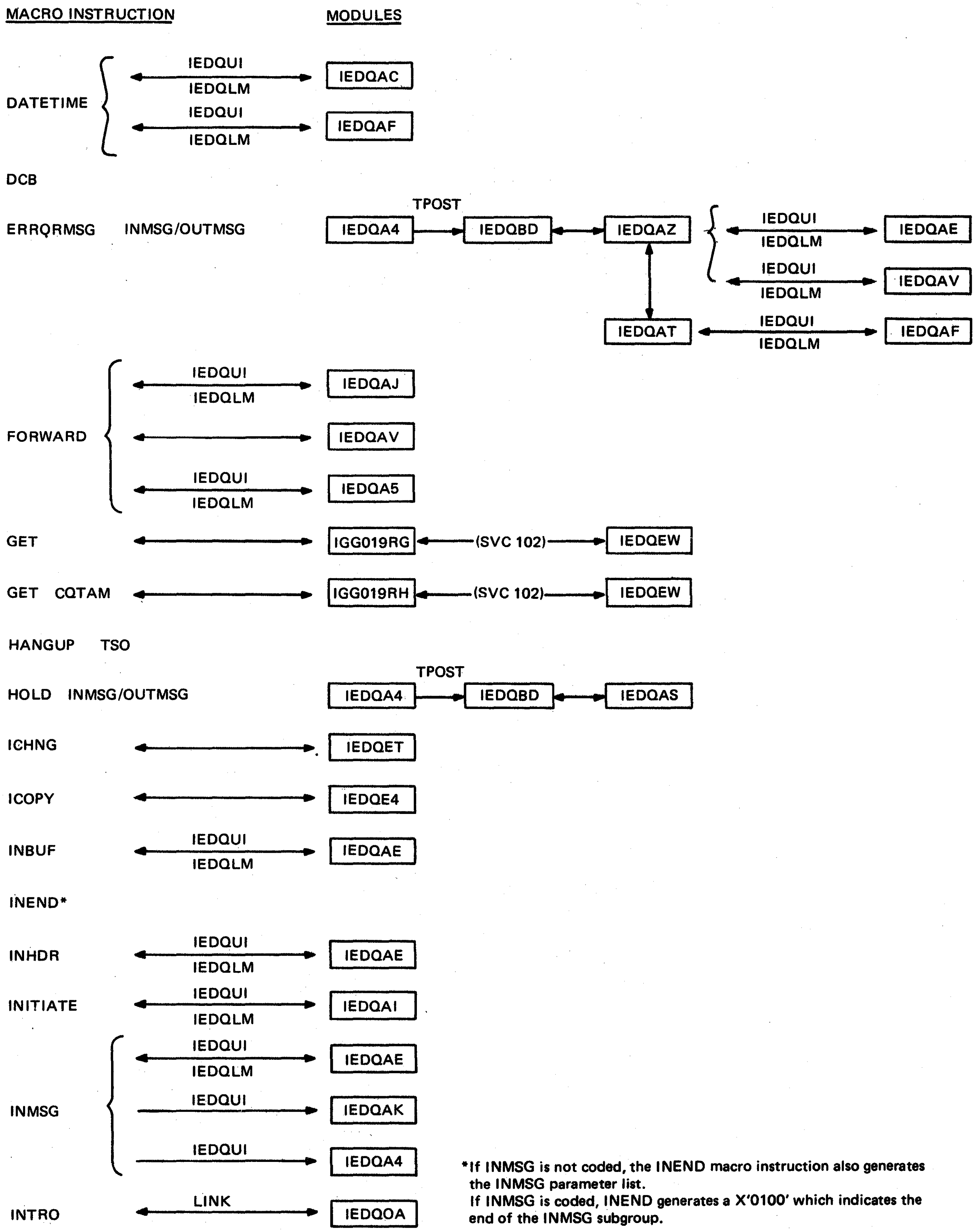
Macros indicated as part of the INMSG/OUTMSG subgroup which are followed by no branch to IEDQA4 are macros which are coded within an INMSG or OUTMSG subgroup. These macro expansions generate a parameter list indicating the routine to be executed. Buffer Disposition (IEDQBD), which is tposted by IEDQA4, examines the parameter list and branches to the proper routine.

Macros indicated as QTAM macros and followed by no linkage are those which if assembled with the QSTART macro instruction generate as NO OPs.

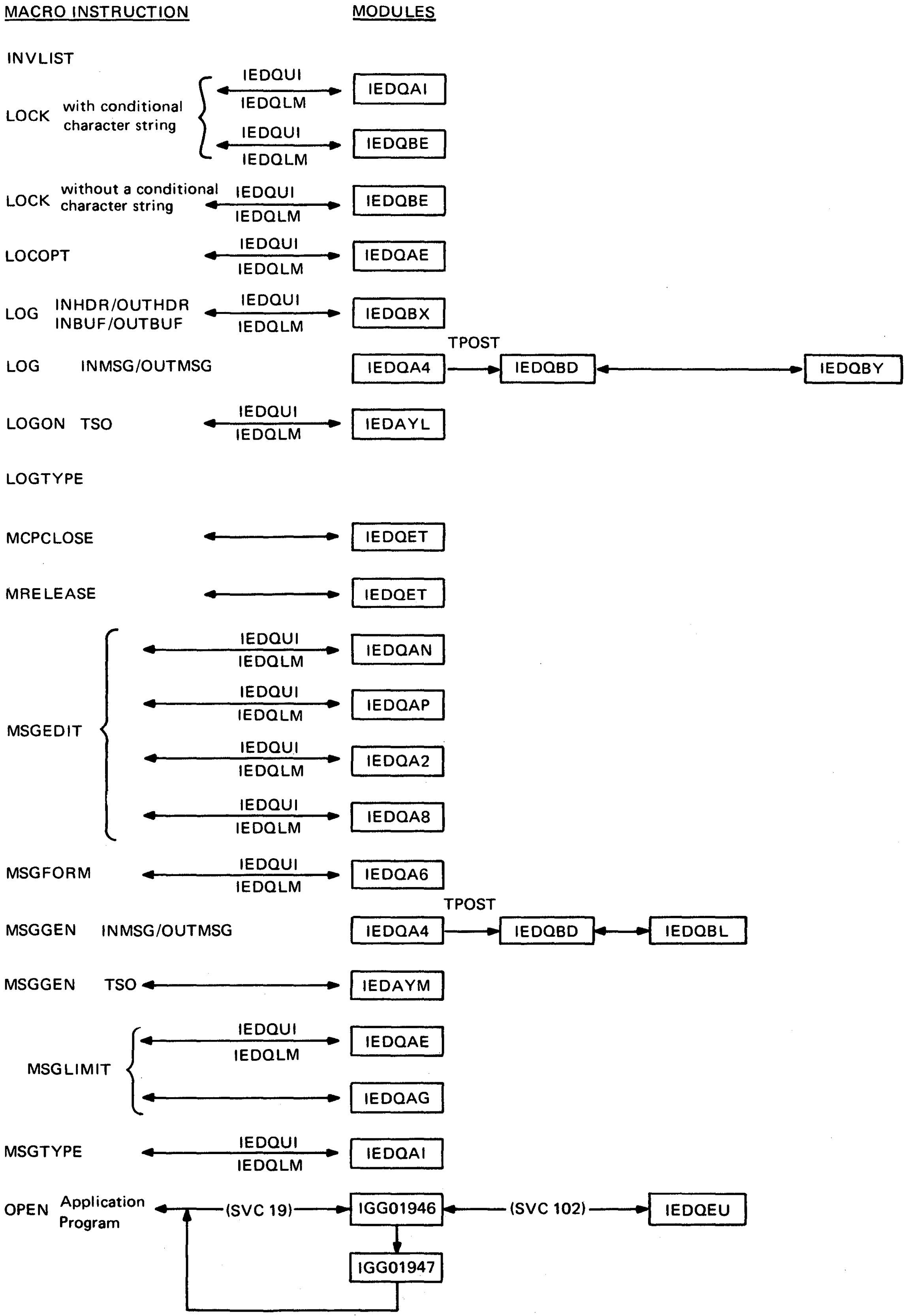
Macros for which no linkage is indicated are those which generate no code or generate only in-line code and do not link to any modules.

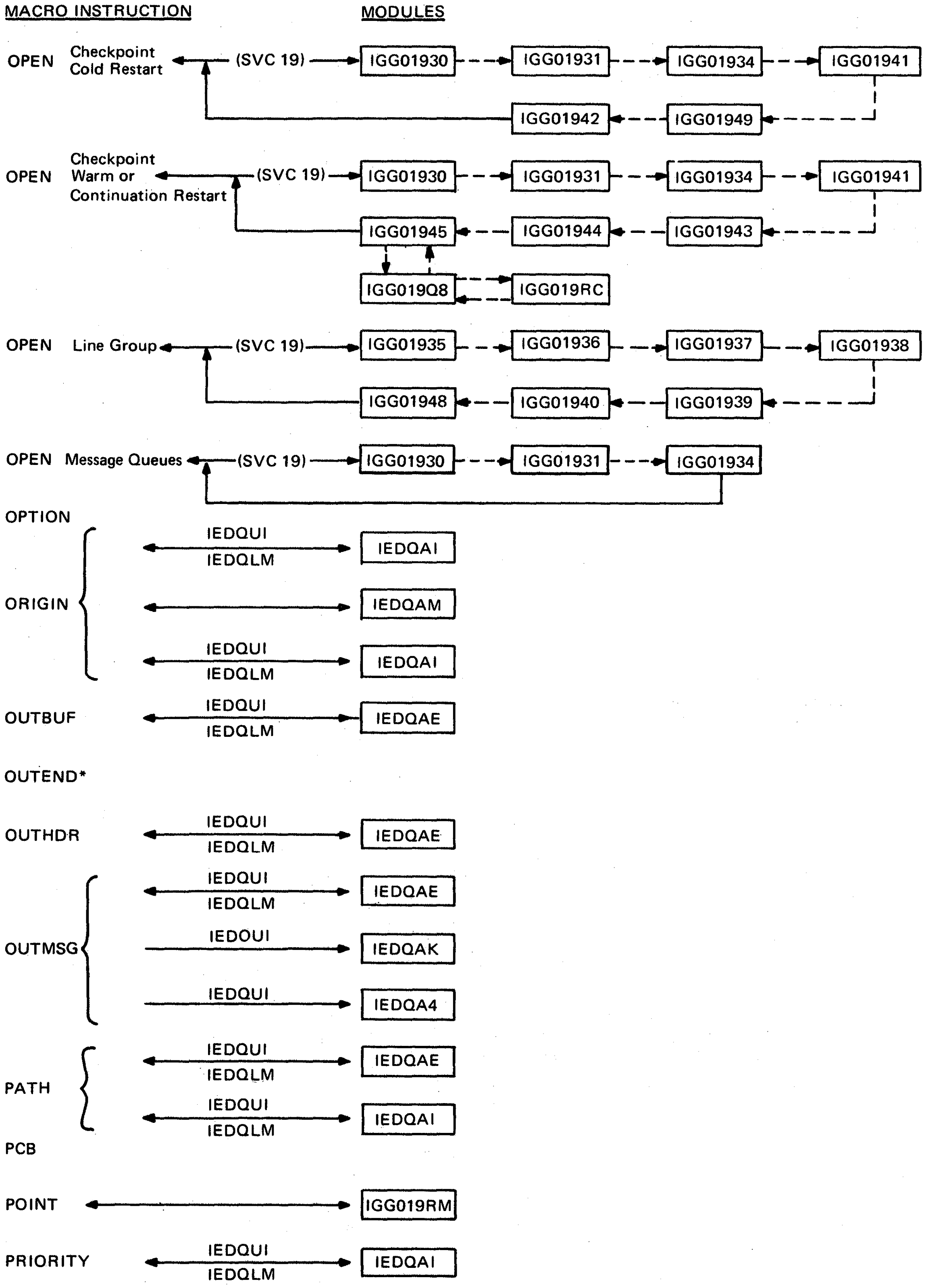



MACRO INSTRUCTION
MODULES
ATTEN TSO INMSG/OUTMSG
IEDQA4
TPOST
IEDQBD
IEDAYX
IEDAYA
CANCELMG INMSG/OUTMSG
IEDQA4
TPOST
IEDQBD
IEDQAR
CARRIAGE TSO
IEDQUI
IEDQLM
IEDAYC
CHECK
IGG019RL
CHECKPT INHDR/OUTHDR INBUF/OUTBUF
IEDQUI
IEDQLM
IEDQBB
CHECKPT INMSG/OUTMSG
IEDQA4
TPOST
IEDQBD
IEDQBB
CHNGP QTAM
CHNGT QTAM
CKREQ
IEDQNB
CLOSE Application Program
(SVC 20)
IGG02046
(SVC 102)
IEDQEU
CLOSE Checkpoint
(SVC 20)
IGG02030
IGG02041
CLOSE Line Group
(SVC 20)
IGG02035
IGG02036
CLOSE Message Queues
(SVC 20)
IGG02030
CLOSEMC CQTAM
IEDQET
CODE
IEDQUI
IEDQLM
IEDQAI
IEDQAQ
IEDQUI
IEDQLM
IEDQAW
COPYP QTAM
COPYQ QTAM
COPYT QTAM
COUNTER
IEDQA7
CUTOFF
IEDQUI
IEDQLM
IEDQAU

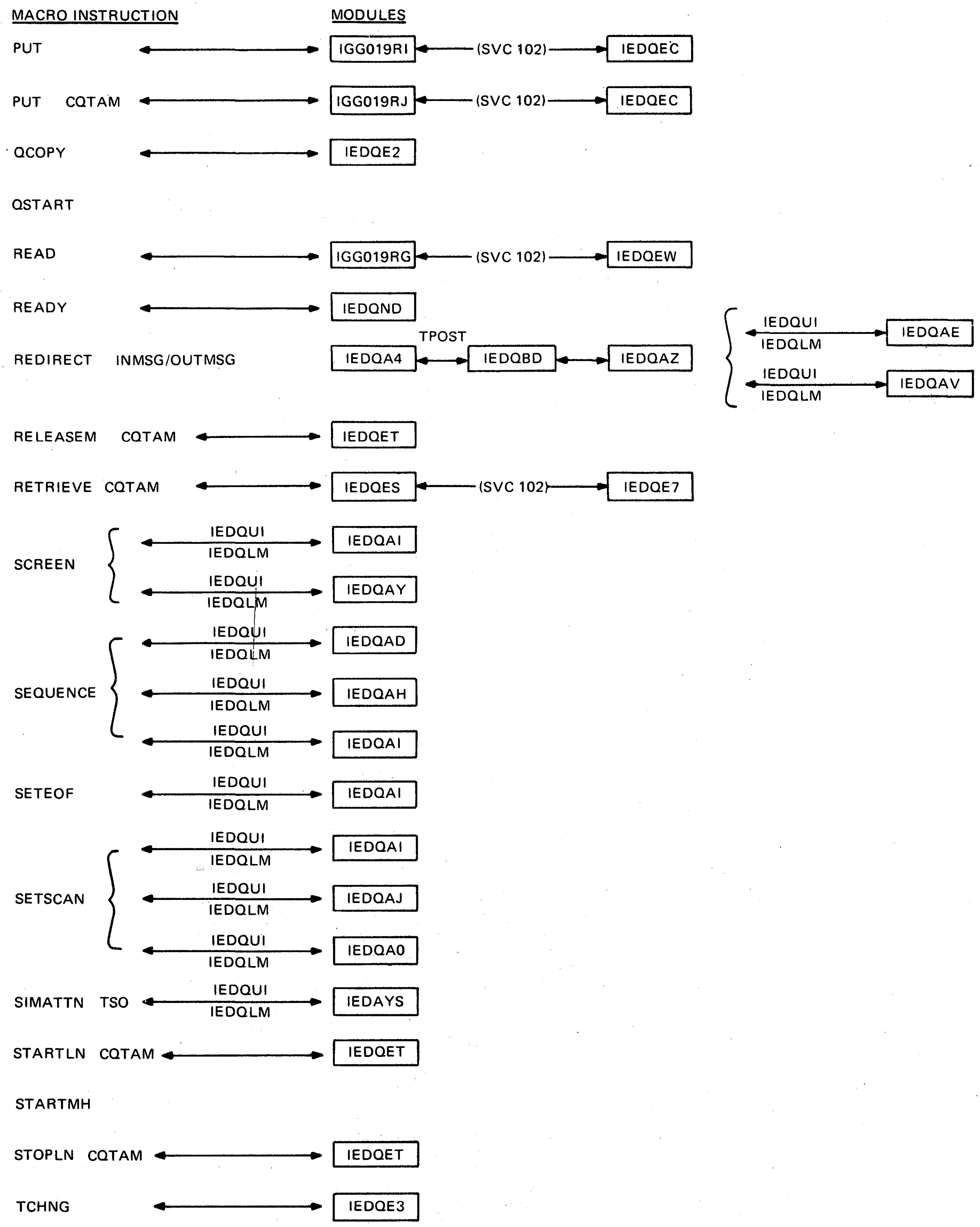


*If INMSG is not coded, the INEND macro instruction also generates the INMSG parameter list.
If INMSG is coded, INEND generates a X'0100' which indicates the end of the INMSG subgroup.

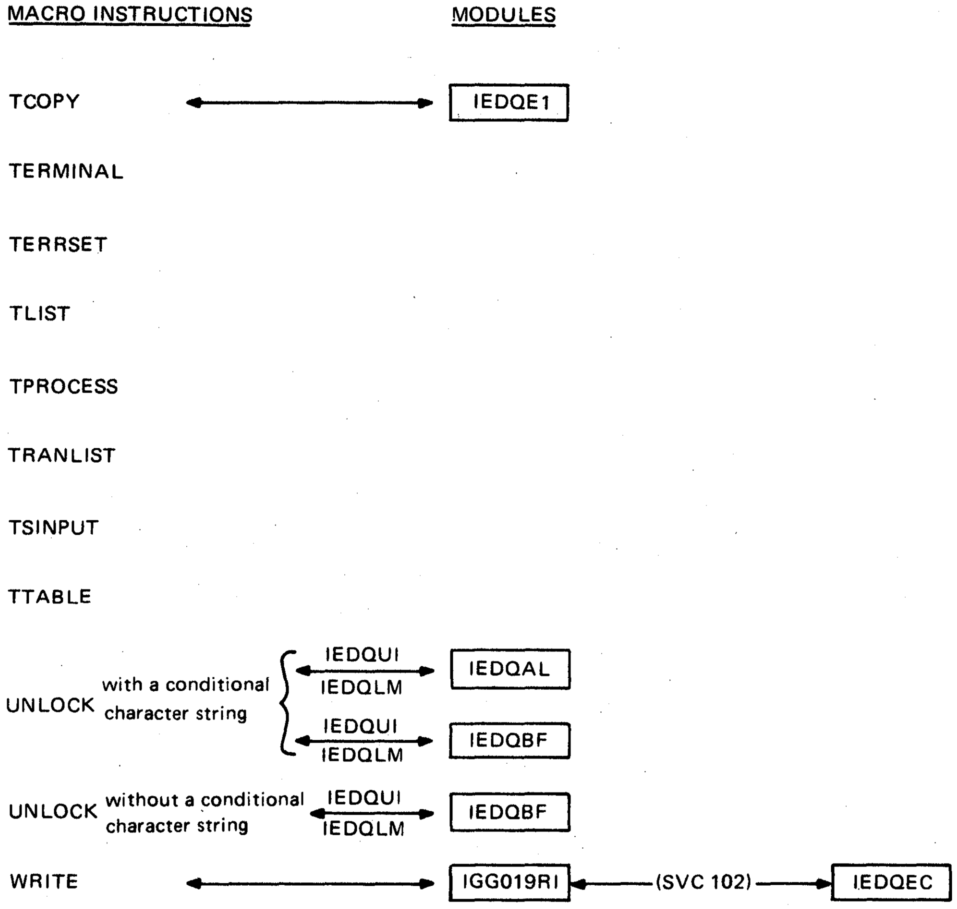

MACRO INSTRUCTION
MODULES
INVLIST
LOCK with conditional character string
IEDQUI
IEDQLM
IEDQAI
IEDQUI
IEDQLM
IEDQBE
LOCK without a conditional character string
IEDQUI
IEDQLM
IEDQBE
LOCOPT
IEDQUI
IEDQLM
IEDQAE
LOG INHDR/OUTHDR INBUF/OUTBUF
IEDQUI
IEDQLM
IEDQBX
LOG INMSG/OUTMSG
IEDQA4
TPOST
IEDQBD
IEDQBY
LOGON TSO
IEDQUI
IEDQLM
IEDAYL
LOGTYPE
MCPCLOSE
IEDQET
MRELEASE
IEDQET
MSGEDIT
IEDQUI
IEDQLM
IEDQAN
IEDQUI
IEDQLM
IEDQAP
IEDQUI
IEDQLM
IEDQA2
IEDQUI
IEDQLM
IEDQA8
MSGFORM
IEDQUI
IEDQLM
IEDQA6
MSGGEN INMSG/OUTMSG
IEDQA4
TPOST
IEDQBD
IEDQBL
MSGGEN TSO
IEDAYM
MSGLIMIT
IEDQUI
IEDQLM
IEDQAE
IEDQAG
MSGTYPE
IEDQUI
IEDQLM
IEDQAI
OPEN Application Program
(SVC 19)
IGG01946
(SVC 102)
IEDQEU
IGG01947



*If OUTMSG is not coded, the OUTEND macro instruction also generates the OUTMSG parameter list.
If OUTMSG is coded, OUTEND generates a X'0100' which indicates the end of the OUTMSG subgroup.



MACRO INSTRUCTION		MODULES				
PUT		IGG019RI	(SVC 102)	IEDQEC		
PUT CQTAM		IGG019RJ	(SVC 102)	IEDQEC		
QCOPY		IEDQE2				
QSTART						
READ		IGG019RG	(SVC 102)	IEDQEW		
READY		IEDQND				
REDIRECT INMSG/OUTMSG		IEDQA4	TPOST IEDQBD	IEDQAZ	IEDQUI IEDQLM	IEDQAE
					IEDQUI IEDQLM	IEDQAV
RELEASEM CQTAM		IEDQET				
RETRIEVE CQTAM		IEDQES	(SVC 102)	IEDQE7		
SCREEN	IEDQUI IEDQLM	IEDQAI				
	IEDQUI IEDQLM	IEDQAY				
SEQUENCE	IEDQUI IEDQLM	IEDQAD				
	IEDQUI IEDQLM	IEDQAH				
	IEDQUI IEDQLM	IEDQAI				
SETEOF	IEDQUI IEDQLM	IEDQAI				
SETSCAN	IEDQUI IEDQLM	IEDQAI				
	IEDQUI IEDQLM	IEDQAJ				
	IEDQUI IEDQLM	IEDQA0				
SIMATTN TSO	IEDQUI IEDQLM	IEDAYS				
STARTLN CQTAM		IEDQET				
STARTMH						
STOPLN CQTAM		IEDQET				
TCHNG		IEDQE3				



MACRO INSTRUCTIONS

MODULES

TCOPY

IEDQE1

TERMINAL

TERRSET

TLIST

TPROCESS

TRANLIST

TSINPUT

TTABLE

UNLOCK with a conditional character string

IEDQUI

IEDQLM

IEDQAL

IEDQUI

IEDQLM

IEDQBF

UNLOCK without a conditional character string

IEDQUI

IEDQLM

IEDQBF

WRITE

IGG019RI

(SVC 102)

IEDQEC

TABLE OF REGISTER USAGE BY MODULE

This table gives the general register usage for each of the executable TCAM modules. When a register contains input data for a module, that register is labeled I. A module work register is labeled W, and a label of O represents a register in which the module is passing output data.

Module Name	Entry Point	Register	I,W, or O	Use
IEDAYA	IEDAYA	0	I	Parameter list address
	IEDAYA+12	1	I	LCB address if entry is at IEDAYA+12
			O	ERB address
		2	W	Work register
		3	I	SCB address
		4	I	LCB address if entry is at IEDAYA
		5	I	TJB address
		6	I	TSB address
		7	I	QCB address
		8	I	RCB address
			O	Address of the message
		9	I	AVT address
		10	I	Time Sharing CVT address
		11	W	Work register
		12	I	Base register; entry point address to IEDAYA
			O	Entry point address to IEDAYM
		13	I	Save area address
		14	I	Return address
		15	I	Entry point address to IEDAYA+12
IEDAYC	IEDAYC	0	W	Work register
		1	I	Index to the Termname Table
		2	I	Data index
		3	I	DCB address
		3	O	SCB address
		4	I,O	LCB address from IEDQUI
		5	I	Data count for carriage position
		6	I	Current buffer address
		7	I	Data pointer
		8	I	Unit index
		9	W	Work register
		10	O	QCB address
		11	I,O	Address of the first buffer
		12	I	Entry point address
		13	I	AVT address
		14	I	Return address
		15	O	Entry point address in IEDQLM
IEDAYD	IEDAYD	1	I	Address of Time-Sharing QCB or TCAM buffer
			W	Address of Terminal Table entry
		2	W,O	Work register; on exit to IEDQTSI, return address in IEDAYD
		3	I	STCB address
			O	Return address in IEDAYD on exit to IEDQTSI; On exit to IEDAYZ - return address in Destination Scheduler (IEDQHM)
		4	O	LCB address
		6	I	Address of TS Scheduler
		7	I,O	Destination QCB address
		10	O	DCB address
		11	I,O	Dispatcher address
		12	W	Base register for IEDAYD
		13	I,O	Address of AVTSAVE2
		15	I	Entry point address
IEDAYF	IEDAYF	0	I	Message length (from IEDAYO); length of the data area in the SCB (from IEDAYM)

		1	I	TCAM buffer address (from IEDAYO); the SCB address (from IEDAYM)
			O	Length of the data moved
		2	I	TIOC buffer address (from IEDAYO); address of the message (from IEDAYM)
		3	I	SCB address
		4	I	LCB address
		5	O	Destination QCB address
		6	I	Address of TIOC buffer
		7	O	Address of TCAM buffer
		8	W	Work register
			I	TSB address (from IEDAYO)
		9	W	Work register
		10	I	DCB address
		11	I	Dispatcher address (from IEDAYM)
		12	I	Base register
		13	I	Address of AVTSAVE2
		14	I,O	Return address
		15	I	Entry point address
			O	Return code register: X'00' - Successful completion X'0C' - Partial line moved X'10' - End of line reached
IEDAYF	IEDAYF	1	I	LCB or ERB address
			O	Address of the LCB to be tposted, as the first element in the chain to the DSPCHAIN function; LCB address to the Time Delay subtask; address of the Receive Scheduler key - to DSPPRIOR
		2	O	UCB address to IOHALT SVC
		3	O	SCB address to DSPCHAIN
		4	O	LCB address to DSPDISP & DSPCHAIN
		5	O	Address of QCB to which the LCB is to be tposted - to DSPCHAIN
		7	O	Address of first STCB in the LCB STCB chain - to DSPPRIOR
		8	O	DEB address
		10	O	DCB address
		11	I	Dispatcher address
		12	I	Base register for IEDAYF
		13	I	Address of AVTSAVE2
		14	O	Return address in IEDAYF - to IEDQHG; entry point address to DSPCHAIN in the Dispatcher
		15	I	Entry point address
			O	Entry point address of the Time Delay Removal subroutine in the Time Delay subtask
IEDAYH	IEDAYH	0	W	Work register
		1	I	Address of the tpost list, contains either the buffers to be tposted to the TSINPUT or zero
			O	ERB address
		2	I	TJB address
		3	I	SCB address
		4	I	LCB address
		5	I	CVT address
		6	I	Time Sharing CVT address
		7	I	QCB address
		8	I	Buffer address
		9	W	Work register
		10	I	TSB address
		11	I,O	TCAM Dispatcher address
		12	I	Entry point address, base register
		13	I,O	Address of AVTSAVE2
		14	I	Return address
	IEDAYH+12	1	I	LCB address
			O	ERB address
		7	I	Address of IEDAYH

		11	I,O	TCAM Dispatcher address
		13	I,O	Address of AVTSAVE2
		15	I	Entry point address
IEDAYI	IEDAYI+2	0	I,W	Index register; work register
		1	I	Address of the TCAM buffer or of the TSI
			O	Address of the chain of elements to be tposted, or zero
		2	I	Pointer to the beginning of the data scanned
		3	I,W	SCB address; pointer to the end of the data scanned
		4	I,W	LCB address; pointer to the basic unit being scanned
		5	I	Address of the Terminal Table
		6	I	Address of the TCAM buffer
		7	I	Address of the TSI
		8	W	Work register
		9	I	TSB address
		10	I	TIOCRPT address
		11	I	Address of the TCAM Dispatcher
		12	I	Base register
		13	I	Address of AVTSAVE2
		14	I,W	Return address; offset to the data in the TCAM buffer
		15	I	Entry point address
IEDAYL	IEDAYL	1	I	Parameter list address
		2	I	DCB address; Time Sharing CVT address
		3	I	SCB adress
		4	I	LCB address
		5	I	Parameter list address
		6	I	Buffer prefix address
		7	I	QUCB address
		8	I	Terminal entry address
		9	I	AVT address
		10	W	Work register
		11	W	Work register
		12	I,O	Entry point address; base register
		13	I	Save area address
		14	W	Work register
		15	O	Return code: X'00' - Initialization is performed for IEDQAE X'04' - Initialization is not performed for IEDQAE
IEDAYM	IEDAYM	0	O	On linking to IEDAYE - size in the SCB of message data
		1	I	Address of tpost list or zero (from IEDQB1) Zero - from IEDAYS; ERB address from IGG019RB
			W	For passing to IEDAYE - address in the SCB to move data to High-order byte: X'05' - Left justify message. X'06' - Do not justify message.
			O	Address of the ERB to be tposted
		2	W	For passing to IEDAYE - address of the message, the first byte of which contains length
		3	I,O	SCB address
		4	I,O	LCB address
		5	W	Work register; translate table address
		6	O	Address of the routine to which to tpost the ERB (IEDQKA or IEDQBD)
		7	I	Address of the QCB removed from the time delay queue (from IEDAYS)
		8	I	Address of macro expansion (MSGGEN macro) from IEDQBD; address of the Simulated Attention Mode (from IEDAYS)
		9	W	"Insert character" register
		11	I,O	Dispatcher address
		12	I	Base address of IEDAYM; Entry point address (from IEDQBD)

		13	I	Address of AVTSAVE2 (from IEDAYS)
		14	O	Return address (after linking to IEDAYE)
		15	O	Entry point in IEDAYE
IEDAYM	AYMCO	0	O	On linking tc IEDAYE - size, in the SCB, of message data
		1	I	Address of ERB
			I	On return frcm IEDAYE - length of the data moved
			O	For passing to IEDAYE - address in the SCB to move data to High-order byte: X'05' - Left justify message X'06' - Do not justify message
		2	O	For passing to IEDAYE - address of the message, the first byte of which contains the length
		3	W	SCB address
		4	W	ICB address
		11	I	Dispatcher address
		13	I	Address of AVTSAVE2
		14	O	Return address (after linking to IEDAYE)
		15	I	Address of the entry point
			O	Entry point in IEDAYE
IEDAYC	IFDAYO	0	W	Work register
		1	I	ERB address or TCAM buffer address
			W	Work register
		2	W	Work register
		3	I	STCB address
			W	SCB address
		4	W	LCB address
		5	W	TJB address
		6	W	TSB address or buffer address
		7	I	Disk I/O QCB address
		8	W	TSB address
		9	W	TSB address
		10	W	TICCFPT address
		11	I	TCAM Dispatcher address
		13	W	Save area address for external routines
		14	W	Return address for external routines
		15	I	Entry pcint address
			W	Return code register fcr external routines
	IFDAYC02	1	I	Basic unit address or QCB address
		3	I	STCB address
		7	I	CPB Cleanup QCB address
		11	I	TCAM Dispatcher address
		13	I	Address of AVTSAVE2
		15	I	Entry pcint address
IEDAYR	IEDQAA01	0	W	Work register
			O	On exit to an uncompleted routine, a negative value to indicate a multiple-buffer header
		1	I	Current buffer address
			O	To IEDQAI and IEDQAW, parameter list address; to Dispatcher, the address of the buffer to be tposted
		2	W	Work register
			O	Increment value of 4096; used for multiple base register support
		3	O	SCB address
		4	O	LCB address
		5	W	UCB address
		6	W	Current buffer address
		7	I	Address of the STARTMH QCB
			O	Address of the new QCB
		8	W	SCT address
		9	W	Work register

		10	W	DCB address
		11	W	Dispatcher address
		12	O	Base register for the message handler
		13	I	Address of AVTSAVE2
		14	W	Return address from the external routines
		15	I	Entry point address into the STARTMH subtask
		Condition Code	O	Entry point address in the new MH for the LOGON exit; On return to MH, the condition code: 4 - Multiple-buffer-header re-entry 1 - Normal receive processing 8 - Normal send processing
IEDAYS	IEDAYS	0	W	TS job identification
		1	I	On return from IEDQTNT, the address of the terminal entry
		2	W	Work register, used for the counter of characters in the buffer for device-dependent processing
		3	I	SCB address
		4	I	LCB address
		5	W	Address of the terminal entry
		6	I	Current TCAM buffer address
		7	W	Destination QCB address
		8	W	Work register, used for device-dependent processing
		9	I	AVT address
		10	W	TSB address
		11	I	Dispatcher address; Entry point address
		12	W	Base register for IEDAYS
		13	I	Address of AVTSAVE2
		14	I	Return address
		15	W	Work register, used for device-dependent processing
			O	On return via the Return Interface routine (IEDQLM), contains the return address
IEDAYS	IEDAYS2	3	I,W	SCB address
		4	I,W	LCB address
		9	I	AVT address
		14	I	Return address in IEDAYE
		15	I	Entry point address
IEDAYS	IEDAYS3	1	I	Address of the QCB removed from the time delay queue and posted
			O	On exit to the MSGGEN routine, set to zero
		2	W	Work register
		3	W	SCB address
		4	W	LCB address
		7	W	QCB address
		8	W	Work register
		9	O	On exit to the MSGGEN routine, set to zero
		11	I	Dispatcher address
		12	I	Base register
			O	Address of entry point in the TSO MSGGEN routine (IEDAYM)
		13	I	Address of AVTSAVE2
		15	I	Entry point address
IEDAYT	IEDAYTO	0	W	SVC entry code for TCABEND
		12	W	Base register
		13	I	Register save area
		14	I	Return address
		15	O	Return code

IEDAYT	IEDAYT1	1	I	Pointer to the work area that contains the address of the ABEND routine
			W	Address of the parameter list for IGC102
		2	W	CVT address
		3	W	AVT address
		4	W	Address of the ABEND routine
		5	W	Work register, used to determine what routine issued an ABEND
		6	W	Address of the QCB for the ABEND routine
		7	W	Work register
		8	W	Address of the first element in the chain
		12	I	Base register
		13	I	Address of the save area
		14	I	Return address
		15	O	Return code
IEDAYT	IEDAYT2	0	W	SVC entry code for TCABEND
		1	W	Address of parameter list for IGC102
		2	W	CVT address
		3	W	AVT address
		7	W	Work register
		12	W	Base register
		13	I	Address of the save area
		14	I	Return address
		15	O	Return code
IEDAYX	IEDAYX	1	I	Address of the tpost list, or zero
		3	I	SCB address
		4	I	LCB address
		8	I	Address of the macro expansion
		9	W	Insert character register
		11	I	Address of the Dispatcher
		12	I	Entry point address
		13	I	Address of AVTSAVE2
IEDAYY	IEDAYY	1	I	Address of the QCB to tpost to the ready queue
		3	W	Work register
		4	W	Address of the previous element on the time delay queue
		5	W	Address of the current element on the time delay queue
		6	I	Address of the TSINPUT QCB
		11	I	Address of the Dispatcher
		12	I	Base address
		13	I	Address of AVTSAVE2
		14	O	Return address in IEDAYY
		15	I	Entry point address
IEDAYZ	AYZ000	0	O	Terminal index or line entry, to IEDQTNT; address of the QCB or LCB to be removed or inserted in the time delay queue, to IEDQHG.
		1	W	Work register
		2	W	Work register
		3	I	Return address in the Leased Receive Scheduler
		4	I	LCB address
		5	W	Work register
		6	I	Base register
			O	Cleared to zero
		7	W	Work register
			O	Destination QCB address
		8	I	Address of the current invitation list entry
		9	I	Address of the invitation list
		10	I	DCB address
		11	I	Address of the Dispatcher
		12	W	Work register
		13	I	Address of AVTSAVE2
		14	O	SCB address to Dispatcher POST

				functicn (return address)
		15	I	Base address of the Leased Receive Scheduler
			O	Entry pcint address
IEDAYZ	AYZ100	0	W	Work register
		1	W	Work register
		2	I	Return address in the Dial Receive Scheduler
		3	I	STCB address
		4	I	LCB address
		5	W	Work register
		6	I	Base register
		7	O	Destination QCB address
		8	W	Work register
		9	W	Work register
		10	I	DCB address
		11	I	Address of the Dispatcher
		12	I	Base address of the Dial Receive Scheduler
		13	I	Address of AVTSAVE2
		14	O	SCB address
		15	W	Work register
IEDAYZ	AYZ200	0	W	Wcrk register
		1	I	Address of the request queue element
			O	Terminal index or line entry, to IEDQTNT; address of the QCB or LCB to be removed from cr inserted in the time delay queue, to IEDQHG.
		2	I	LCB address
		3	I	Return address in Line End Appendage
		4	I	Seccnd base register in Line End Appendage
		5	W	Work register
		6	I	Base register
		7	W	Work register
		8	O	Destination QCB address; Address of the last buffer serviced
		9	W	Wcrk register
		10	C	Cleared to zero
		11	O	DCB address
		12	I	Address of the Write Poll Characters CCW
		13	I	Address of AVTSAVE2
		14	I	Return address to IOS
			O	Return address to IEDAYZ; SCB address to Dispatcher
		15	I	Base address of Line End Appendage
			O	Entry pcint address
IEDAYZ	AYZ300	0	W	Work register
		1	O	Terminal index or line entry
		2	O	LCB address
		3	I	Return address in the QEVENT routine
		4	I	LCB address
		5	O	Return address in IEDAYZ to IEDQKA
		6	I	Base register
		7	I	Address of the Destination QCB
		8	O	Address cf the first buffer to DSBUNAV
		9	W	Work register
		10	I	DCB address
		11	W	Wcrk register
		12	O	Entry pcint address to IEDQKA
		13	I	Address of AVTSAVE2
		14	O	Return address in IEDAYZ
		15	I	Base address of the QEVENT routine
			O	Entry pcint address

IEDAYZ	AYZ400	0	W	Work register
		1	O	Terminal or line index
		2	W	Work register
		3	I	Return address in the Send Scheduler
			O	LCB address to DSPUNAV
		4	I	LCB address
		5	W	Work register
		6	I	Base register
		7	I	Address of the Destination QCB
		8	O	Address of the first buffer to DSPUNAV
		9	W	Work register
		10	I	DCB address
		11	I	Address of the Dispatcher
		12	W	Work register
		13	I	Address of AVTSAVE2
		14	I	Base address of the Send Scheduler
			O	Entry point to the Dispatcher Dispatch function unless a work break was requested and could not be issued, to IGG019BH; return address in IEDAYZ, to IEDQTNT
		15	I	Base address of the Send Scheduler
			O	Entry point address
IEDAYZ	AYZ410	0	W	Work register
		1	O	Terminal or line index
		2	I	Return address in the Time Sharing Destination Scheduler
		3	I	Return address in the Time Sharing Destination Scheduler
		4	I	LCB address
		5	W	Work register
		6	I	Base register
		7	I	Address of the Destination QCB
		8	O	Address of the first buffer
		9	W	Work register
		10	I	DCB address
		11	I	TCAM Dispatcher address
		12	W	Work register
		13	I	Address of AVTSAVE2
		14	O	Return address in IEDAYZ
		15	O	Entry point address
IEDAYZ	AYZ500	0	W	Work register
		1	O	Terminal or line index, to IEDQTNT; LCB address, to the Dispatcher BYPASS function
		2	I	Return address in the Send Scheduler
		3	I	Address of the next STCB
		4	I	LCB address
		5	W	Work register
		6	I	Base register
		7	I,O	Address of the Destination QCB
		8	W	Work register
		9	I	SCB address
		10	W	Work register
		11	I	Address of the Dispatcher
		12	I	Base address of the Send Scheduler
		13	I	Address of AVTSAVE2
		14	O	Return address in IEDAYZ
		15	O	Entry point address
IEDAYZ	AYZ600	0	W	Work register
		1	O	Terminal or line index, to IEDQTNT
		2	I,O	LCB address
		3	I	Return address in Activate-I/O Generator
		4	I	DCB address
		5	W	Work register

		6	I	Base register
		7	O	Destination QCB address
		8	W	Work register
		9	W	Work register
		10	W	Work register
		11	I	Address of the Dispatcher
		12	W	Work register
		13	I	Address of AVTSAVE2
		14	O	Return address in IEDAYZ - to IEDQTNT; SCB address
		15	I	Base address of Activate-I/O Generator
			O	Entry point address
IEDQAA	IEDQAA01	0	W	Work register
			O	On exit to an uncompleted routine, the input parameter list address
		2	W	Work register
		3	W	SCB address
		4	W	LCB address
		6	W	Current buffer address
		7	I	STARTMH QCB address
		8	W	SCT address
		10	W	DCB address
		11	I,O	TCAM Dispatcher address
		12	W	Base register
			O	Base register for the message handler routine
		13	I,O	Calling routine save area address and base for AVT
		14	I	Return address
		15	I	Entry point address into the STARTMH subtask
			W	Link register
			O	Return code: X'01' - Normal receive processing X'04' - Multiple-buffer header re-entry X'08' - Normal send processing
IEDQAC	IEDQAC01	0	W	Parameter register for time
		1	I	Input parameter list address
			W	Parameter register for date
		2	W	Work register
		3	W	Work register
		4	W	Work register
		5	W	Scan pointer address
		6	I	Current buffer address
		7	W	Scan pointer offset
		8	W	Index register
		9	I	AVT address
		10	W	Local return register
		12	I	Base register
		13	I,O	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code: X'00' - Successful completion
IEDQAD	IEDQAD01	1	I	Parameter list address
		2	W	Work register
		3	I	SCB address
		6	I	Current buffer address
		7	W	Translate Table address
		8	W	Work register
		9	I	AVT address
		10	W	Local return register
		12	I	Base register
		14	W	Return register
		15	W	Entry point address
			O	Return code: X'00' - Successful completion X'04' - Insufficient reserve characters specified

IEDQAE	IEDQAE	1	I	Parameter list address
		2	W	Work register
		3	W	Parameter list address
		4	I	LCB address
		5	W	Work register
		7	W	Work register
		8	W	Terminal table entry address
		9	I	AVT address
		12	I,O	Base register
		13	I,O	Calling routine save area address
		14	I	Return address
		15	W	Entry point address
			O	Return code: - If register 15 is the return register - Option field address - Successful completion X'00' - Error exit - If register 15 is not the return register - X'00' - Successful completion X'04' - Error exit
IEDQAF	IEDQAF01	0	W	Work register
		1	I	Input parameter list address
		2	W	Work register
		3	I	SCB address
			W	Insert data address and insert character
		4	I	LCB address
			W	Current unit address
		5	W	Insert address
		6	I	Current buffer address
		7	W	Data offset
		8	I	Shift limit
		9	I	AVT address
		10	W	Local return register
		11	W	Insert data length
		12	I	Base register
		13	I,O	Calling routine save area address
		14	W	Return address
		15	W	Entry point address
			O	Return code: X'00' - Successful completion X'04' - For expand buffer, insufficent reserve characters present for the expansion
IEDQAG	IEDQAG01	0	W	Work register
		1	I	Parameter list address
		13	I	Base register for AVT
		14	I,O	Return address
		15	I	Base register
			O	Return code: X'00' - successful completion
IEDQAH	IEDQAH01	0	W	Digit
		1	I	Input parameter list address
			W	Terminal Table entry address
		2	W	Return register
		3	I	SCB address
		4	I	LCB address
		6	I	Current buffer address
		7	W	Clear mask
		9	I	AVT address
		11	W	Count
		12	I	Base register
		13	I,O	Calling routine save area address
		14	W	Index register and return address
		15	W	Sequence number
			O	Return code: X'00' - Successful completion X'04' - Sequence number is low X'08' - Sequence number is high X'0C' - Origin is not specified

IEDQAI	IEDQAI01	0	W	Work register
		1	I	Parameter list address
		2	W	Blank character
		3	I	SCB address
		4	W	Current unit address
		5	W	Scan pointer address
		6	I	Current buffer address
		7	W	Scan pointer offset
		8	W	Compare characters address
		9	I	AVT address
		10	W	Length parameter
		11	W	End-of-unit address
		12	I	Base register
		13	I,O	Calling routine save area address
		14	I	Return address
		15	W	Entry point address
			O	Return code: X'00' - Successful completion X'04' - Skip incomplete Negative - Multiple-buffer-header processing, scan incomplete
IEDQAJ	IEDQAJ01	0	W	Work register
		1	I	Parameter list address
		2	W	Blank character
		3	I	SCB address
		4	I	Current unit address
		5	W	Scan pointer address
		6	I	Current buffer address
		7	W	Scan pointer offset
		8	W	Character string address
		9	I	AVT address
		10	W	Compare register
		11	W	End-of-unit address
		12	I	Base register
		13	I,O	Calling routine save area address
		14	I	Return address
		15	W	Entry point address
			O	Return code: Successful completion - X'00' or, if register 15 is the return register, the offset to the last byte of the character string Unsuccessful completion - - Parameter return register specified - X'04' or, if register 15 is the return register, X'00' - No parameter return register specified - a negative value for multiple-buffer-header processing
IEDQAK	IEDQAK01	0	W	Work register
		1	I	Parameter list address
		2	W	SCT address
		3	I	SCB address
		4	I	LCB address
		5	W	Data offset
		6	I	Current buffer address
		7	W	SCT address
		8	W	Shift limit
		9	I	AVT address
		10	W	Work register
		11	W	Size of data
		12	I	Base register
		13	I,O	Calling routine save area address
		14	I	Return address
		15	W	Entry point address
IEDQAI	ADDPCOMP	1	I	Parameter register
		2	W	Current unit address
		4	O	Current unit address

	5	I	Cffset to the item of which the address is sought
		O	Address of the item sought
	6	I	Current tuffer address
	9	I	AVT address
	11	O	End-cf-unit address
	14	I,O	Return address
	15	I	Base register
IEDQAM IEDQAM01	0	W	Work register
	1	I	Parameter list address
	13	I	AVT tase register
	14	I	Return address
	15	I	Link register
		O	Return code: X'00' - Successful completion X'04' - Origin field is invalid
IEDQAN IEDQAN01	0	W	Work register
	1	I	Address of parameter list
		W	Work register
	2	W	Wcrk register
	3	I	SCB address
	4	I	ICB address
		W	Character string address
	5	W	Data address
	6	I	Current tuffer address
	7	W	Parameter list address
	8	W	Shift limit
	9	I	AVT address
	10	W	Current sutparameter list address
	11	W	Execute register
	12	I	Base register
	13	I,O	Calling routine save area address
	14	I	Return address
	15	W	Entry pcint address
		O	Return code: X'00' - Successful completion X'04' - Empty tuffers needed for insert operaticn not available
IEDQAO IEDQAO01	0	W	Work register
	1	I	Parameter list address
	2	W	Work register
	3	I	SCB address
	4	I	LCB address
		W	Unit address
	5	W	Data address
	6	I	Address cf tuffer currently being processed
	7	W	Compare address
	8	I	New unit address
	9	I	AVT address
	10	W	Conditicn code register
	11	W	TCAM Dispatcher address and execute register
	12	I	Base register
	13	I,O	Calling routine save area address
	14	W	Return address
	15	W	Entry pcint address
		O	Return code: X'C0' - Successful completion X'04' - A buffer unit was not available, thus an unsuccessful insert operation
IEDQAP IEDQAP01	0	W	Wcrk register
	1	I	Address of parameter list
	2	W	Work register
	3	I	SCB address
	4	I	LCB address
	5	W	Scan pointer address
	6	I	Current tuffer address
	7	W	'At' address

		8	W	Subparameter list address and shift limit
		9	I	AVT address
		11	W	Prefix length
		12	I	Base register
		13	I,O	Calling routine save area address
		14	W	Return address
		15	W	'To' offset
			O	Return code: X'00' - Successful completion X'04' - The 'to' character string is not found in the buffer currently being processed; thus, the remove operation was unsuccessful.
IEDQAQ	IEDQAQ01	0	W	Work register
		1	W	Work register
		11	O	On exit to tpost, address of the TCAM Dispatcher
		13	I	AVT base register
		15	W	Entry point address
IEDQAR	IEDQAR	1	I	Address of a list of elements to be tposted
		3	I	SCB address
		4	I	LCB address
		11	I	TCAM Dispatcher address
		12	I	Entry point address
		13	I	Calling routine save area address
IEDQAS	IEDQAS IEDQAS01	1 3	I I	Address of a list of elements to be tposted SCB address
		4	I	LCB address
		6	I	Recalled header buffer
		11	I	TCAM Dispatcher address
		12	I	Entry point address for IEDQAS
		13	I	Calling routine save area address
		15	I	Entry point address for IEDQAS01
IEDQAT	IEDQAT01	0	W	Work register
		1	I	Post list address
		2	W	Work register
		3	I	SCB address
		4	I	LCB address
		5	W	Error message address
		6	I	Current buffer address
		7	I	Error message length
		8	I	Error message parameter list address
		9	W	Priority
		10	W	Post list address
		11	I	Dispatcher address
		12	I	Base register
		13	I,O	Base register for AVT addressability
		14	W	Return address
		15	W	Entry point address
IEDQATTN	IEDATTN	7	I	UCB address
		13	W	AVT address
IEDQAU	IEDQAU	1	I	Address of macro-generated parameter list
		3	I	SCB address
		4	I	LCB address
		6	I	Address of the current buffer
		12	I	Entry point address and base register
		13	I	Save area address and base register for AVT addressability

CUTFFQCB+12	1	I	ICB address
	7	I	Cutoff QCB address
	11	I	Address of the TCAM Dispatcher
	13	I	Save area address and base register for AVT addressability
IEDQAV IEDQAV01	0	W	Work register
	1	I	Macro-generated parameter list address
		W	Work register
	13	I	Base address for the AVT
	14	I,O	Return address
	15	I	Base register
		O	Return code: X'00' - Successful completion X'C4' - Buffer prefix destination key is not available; thus, the terminal entry is not found.
IEDQAW IEDQAW01	0	W	Key length
	1	I	Parameter list address
	2	W	Work register
	3	I	SCB address
	4	I	ICB address
	5	W	Address of start of translation
	6	I	Current buffer address
	8	W	Translation Table address
	9	I	AVT address
	10	W	DCB address and data length
	11	W	Execute register
	12	I	Base register
	13	I,O	Calling routine save area address
	14	I	Return address
	15	W	Entry point address
IEDQAX SCAN	0	W	Work register
	4	I	Current unit address
	5	I	Offset into buffer
		O	Address in buffer
	6	I	Current buffer address
	9	I	AVT address
	11	O	End-of-unit address
	13	I,O	Calling routine save area address
	14	I,O	Return address
	15	I	Base register
IEDQAY IEDQAY01	0	W	Work register
	1	I	Address of the input parameter list
	2	W	Work register
	4	I	ICB address
	6	I	Current buffer address
	7	W	UCB address
	8	W	Parameter list address
	9	I	AVT address
	12	I	Base register
	13	I,O	Calling routine save area address
	14	W	Return address
	15	W	Entry point address
		O	Return indication: New function byte - successful completion All zeros - the destination is not a screen device
IEDQAZ IEDQAZ01	0	W	Work register
	1	I	Address of additional element to be tposted
		W	Subparameter list address
	2	W	Work register
	3	I	SCB address
	6	I	Buffer address
	7	W	Tpost element address
		I	Parameter list address

		8	I	Address of macro-generated parameter list
			W	Subparameter list address
		9	W	Local link register
		11	I	TCAM Dispatcher address
		12	I	Base register
		13	I	Base for AVT addressability
		14	W	Return address
		15	W	Entry point address
IEDQA0	IEDQA001	1	I	Input parameter list address
		3	W	End-of-prefix address
		4	I	LCB address
		5	W	Scan pointer address
		6	I	Current buffer address
		7	W	Scan pointer offset
		8	W	Blank character
		9	I	AVT address
		10	W	Skip length
		12	I	Base register
		13	I	Calling routine save area address
		14	W	Return address
		15	W	Entry point address
			O	Return code: X'00' - Successful completion X'04' - Skip ran into the buffer prefix
IEDQA1	IEDQA101	0	W	Work register
		1	I	Input parameter list address
		2		Compare characters address
		3	I	SCB address
		4	W	Termname Table address
		5	W	Compare length
		7	W	Search extent register
		8	W	Address of the last entry
		9	I	AVT address
		10	W	Address of the current entry in the table
		11	W	Length of a table entry
		12	I	Base register
		13	I,O	Calling routine save area address
		15	O	Return register: X'00' - No matching entry found Ordinal index (key) to matching Termname Table entry - Successful completion
IEDQA2	IEDQA201	0	W	Work register
		1	I	Address of the input parameter list
		2	W	Work register
		3	I	SCB address
		4	I	LCB address
		5	W	Data offset
		6	I	Current buffer address
		7	W	Residual count
		8	W	New unit address and shift limit
		9	I	AVT address
		10	W	Local return address
		12	I	Base register
		13	I,O	Calling routine save area address
		14	W	Return address
		15	W	Entry point address
			O	Return code: X'00' - Successful completion X'04' - No logically empty buffer units are available, thus an unsuccessful insert operation
IEDQA3	IEDQA3	1	W	Work register
		2	W	Work register
		3	I	SCB address
			W	String pointer

	4	I	ICB address
	5	W	Buffer address of data
	6	W	Address of buffer containing data
	7	W	Work register
	8	I	Address of TRANLIST
		O	Address of the translation table
	9	I	AVT base
	10	W	Data size index
	11	W	Execute register
	12	I	Routine base register
	13	I	Save area address
	14	I	Return address
	15	O	Return code register
			X'00' Successful completion
			X'04' No control string match found
			X'08' No option field found
IEDQA4 IEDQA401	0	W	Work register
	1	I	Parameter list address
		W	Work register
	2	W	Work register
	3	I	SCB address
	4	I	ICB address
	5	W	Scan address
	6	I	Current buffer address
	7	W	SCT address and scan offset
	9	I	AVT address
	11	W	TCAM Dispatcher address
	12	I	Base register
	13	I,O	Calling routine save area address
	14	W	Return address
	15	W	Entry point address
IEDQA5 IEDQA501	0	W	Work register
	1	I	Parameter list address
		W	Subparameter list address
	2	W	Parameter list address
	3	I	SCB address
	4	I	ICB address
	5	W	Work register
	6	I	Current buffer address
	8	W	Subparameter list address
	9	I	AVT address
	10	W	Local return address
	11	W	Execute register
	12	I	Base register
	13	I,O	Calling routine save area address
	14	W	Return address
	15	W	Entry point address
		O	Return code:
			X'00' - Successful completion
			X'04' - on entry from multiple routing - the destination name is incomplete in the current buffer
			X'04' - on entry from the FORWARD macro expansion - a valid destination cannot be found
			X'08' - on entry from Multiple Routing - an EOA character string is detected.
IEDQA6 IEDQA601	0	W	Work register
	1	I	Address of parameter list
		W	Terminal Table entry address
	2	W	Device dependent area address
	3	I	SCB address
	4	I	ICB address
	5	W	Block extent
	6	I	Current buffer address
	7	W	Subblock extent
	8	W	Parameter list address
	9	I	AVT address

		10	W	Local return address
		12	I	Base register
		13	I,O	Calling routine save area address
		14	W	Return address
		15	W	Entry point address
			O	Return code: X'00' - Successful completion X'04' - The line control intervals are not in the input parameter list or terminal entry, or the line control characters are not valid for the buffer destination.
IEDQA7	IEDQA7C1	1	W	Work register
		13	I,O	Calling routine save area address
			W	Base for AVT addressability
		14	W	Return address
		15	W	Entry point address and option field address
			O	Return code: X'00' - Successful completion X'FF' - The buffer is zero-length, the counter option field is not found, or the counter option field is not changed.
IEDQA8	IEDQA801	0	W	Work register
		1	I	Address of the input parameter list
		2	W	Extent between inserts
		3	W	Data flag
		4	I	LCB address
		5	W	Scan register
		6	I	Current buffer address
		7	W	Parameter list address
		8	W	Shift limit
		9	I	AVT address
		10	W	Option field address
		12	I	Base register
		13	I,O	Calling routine save area address
		14	W	Return address
		15	W	Entry point address
			O	Return code: X'00' - Successful completion X'04' - The option field is not found; thus, the insertion operation is not successful
IEDQBA	IEDQBA01	0	W	Work register address to be posted
		1	I	Address of the element to be tposted
			W	Parameter list address
		2	W	Work register
		3	W	SCB address
		4	W	LCB address
		5	W	Recalled buffer count
		6	W	Current buffer address
		7	I	Multiple Routing QCB address
		8	W	Header buffer address
		11	I	TCAM Dispatcher address
		12	W	Base register
		13	I,O	Calling routine save area address
			W	Base for AVT addressability
		14	W	Return address
		15	W	Entry point register
IEDQBB	IEDQBB	0	I	Entry code: X'00' - entry from IEDQBD Nonzero - entry from IEDQUI
		1	O	ERB address
		2	W	Address of IEDQBD
		3	I	SCB address
		4	I	LCB address
		9	I	AVT address
		11	I	TCAM Dispatcher address

		12	I	Routine base register
		13	I	Calling routine save area address
		15	O	Return code: X'00' - Successful completion X'04' - Checkpoint not in system
IEDQBC	IEDQBC	1	I	Address of the buffer or ERB
		2	W	0 or LCB address
		3	W	STCB address
		4	W	LCB address
		6	W	Buffer address
		7	I	IEDQBC QCB address
		9	W	Work register
		10	W	Work register
		11	I	TCAM Dispatcher address
		13	I	Calling routine save area address
		14	W	IEDQTNT return address
		15	I	Entry point address
			W	Termname Table address
IEDQBD	IEDQBD	0	W	Work register = 0
		1	I	Element address (buffer, ERB, or LCB)
		2	W	Work register
		3	W	SCB address
		4	W	LCB address
		5	W	Work register
		6	W	Buffer address
		7	I	QCB address for IEDQBD
		8	O	Parameter list for the Incoming/Outgoing Message Delimiter routine
		9	W	Work register
		10	W	Subroutine link register
		11	I	TCAM Dispatcher address
		12	O	Incoming/Outgoing Message Delimiter routine base register
		13	I	Save area address for the calling routine
		14	W	Subroutine link register
		15	I	Base register
IEDQBE	IEDQBE	1	I	Input parameter list address
		2	W	Work register
		3	I	SCB address
		4	I	LCB address
		5	W	Buffer Return QCB address
		6	I	Buffer address
		9	I	AVT address
		14	W	IEDQLM address and subroutine return address
		15	O	Return code: X'00' - Successful completion X'04' - The source was not identified X'08' - The destination is not an open process entry, the buffer has a length of zero, or the input buffer is not a header buffer X'12' - The station is already locked
IEDQBF	IEDQBF	3	I	SCB address
		4	I	LCB address
		9	I	AVT address
		14	O	Return address (IEDQLM)
		15	O	Return code: X'00' - Successful completion X'04' - Terminal is not locked
IEDQBG	IEDQBG	1	I	Input buffer address
		2	W	Work register
		3	W	Work register
		4	W	Work register
		5	W	Work register
		6	W	Buffer address
		7	I	QCB address for IEDQBG

		8	W	Work register
		9	W	Work register
		11	I	TCAM Dispatcher address
		13	I	Save area address for the calling routine
		14	W	Subroutine return register
		15	W	Termname Table address
IEDQBL	IEDQBL	1	I	Chain of elements to be tposted
		2	W	DCB address
		3	I	SCB address
		4	I	LCB address
		5	W	Work register
		8	I	Input parameter list address for MSGGEN
		9	W	Length of MSGGEN
		10	W	Work register
		11	I	TCAM Dispatcher address
		12	I	Base register
		13	I	Save area address for the calling routine
		14	W	Work register
		15	W	Translation Table address
IEDQBT	IEDQBT	1	I	Input element address (buffer or ERB)
		2	W	Work register
		3	I	Base register
		4	W	LCB address
		5	W	Work register
		6	W	Buffer address
		7	I	STARTMH QCB address
		8	W	SCB address
		9	W	Work register
		10	W	Subroutine return register
		11	I	TCAM Dispatcher address
		12	W	Work register
		13	I	Save area address for the calling routine
		14	W	Subroutine return register
		15	W	Work register
IEDQBW	IEDQBW	3	I	SCB address
		4	W	LCB address
		6	I	Buffer address
		7	W	Buffer Return QCB address
		11	I	TCAM Dispatcher address
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
IEDQBW	IEDQBW01	1	I	Address of the buffer just returned
		3	W	SCB address
		4	W	LCB address
		7	W	Buffer Return QCB address
		11	I	TCAM Dispatcher address
		13	I	Calling routine save area address
		15	I	Entry point address
IEDQBX	IEDQBX	1	I	Input parameter list address
		6	I	Current buffer address
		12	I	Entry point address
		14	I	Return address
		15	O	Return code: X'00' - Successful completion X'04' - DCB not open
IEDQBY	IEDQBY	1	I	Address of element chain to be tposted
		3	I	SCB address
		6	I	Recalled header address
		7	I	Destination QCB address
		8	I	Input parameter list address
		11	I	TCAM Dispatcher base address
		12	I	Entry point address

IEDQBZ	IEDQBZ	1	I	Address of the buffer, LCB or ERB
		2	I	DCB address
		3	I	SCB address
			W	Address of the buffer to write
		4	I	LCB address
		5	W	Number of units to write for this buffer
		6	I	Current buffer address
		7	W	QCB address
		8	W	Number of writes that can be issued
		10	W	DCB address
		11	I	TCAM Dispatcher address
		12	I	Entry point address
		13	I	Calling routine save area address
IEDQCA	IEDQCA01	0	O	Entry code
		1	I	AVT address
			O	Operator Control AVT address
		2	W	Operator Control AVT address
		7	W	Work register
		12	W	Routine base register
		13	I	Save area address
			O	Operator Control save area address
		15	I	Entry point address
	IEDQCA02	0	O	Entry code
		1	O	Operator Control AVT address
		14	I	Return address
		15	I	Entry point address
IEDQCF	IEDQCF	1	I	Operator Control AVT address
			O	Address of appropriate response message
		2	W	Operator Control AVT address
		3	W	Input buffer address or CIB address
		4	W	AVT address
		5	W	Option address
		6	W	Termname Table address
		7	W	Terminal entry address
		12	W	Routine base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code:
				X'00' - Successful completion
				X'04' - Unsuccessful completion
IEDQCG	IEDQCG	1	I	Operator Control AVT address
			O	Address of appropriate response message
		2	W	Operator Control AVT address
		4	W	AVT address
		5	W	Termname Table address
		6	W	Terminal entry address
		7	W	QCB address
		8	W	DCB address
		9	W	DEB address
		10	W	UCB address
		12	W	Routine base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code:
				X'00' - Successful completion
				X'04' - Unsuccessful completion
IEDQCH	IEDQCH	1	I	Operator Control AVT address
			O	Address of appropriate response message
		2	W	Operator Control AVT address
		4	W	AVT address
		5	W	Termname Table address
		6	W	Terminal entry address
		10	W	QCB address
		12	W	Routine base register

		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code: X'00' - Successful completion X'04' - Unsuccessful completion
IEDQCI	IEDQCI	1	I	Operator Control AVT address
			O	Address of appropriate response message
		2	W	Operator Control AVT address
		3	W	Input buffer address or CIB address
		4	W	AVT address
		6	W	DEB address
		7	W	DCB address
		9	W	LCB address
		11	W	UCB address
		12	W	Routine base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code: X'00' - Successful completion X'04' - Unsuccessful completion
IEDQCJ	IEDQCJ	1	I	Operator Control AVT address
			O	Address of appropriate response message
		2	W	Operator Control AVT address
		4	W	AVT address
		5	W	Termname Table address
		6	W	Terminal entry address
		7	W	OCB address
		11	W	LCB address
		12	W	Routine base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code: X'00' - Successful completion X'04' - Unsuccessful completion
IEDQCK	IEDQCK	1	I	Operator Control AVT address
			O	Address of appropriate response message
		2	W	Operator Control AVT address
		4	W	AVT address
		5	W	Termname Table address
		10	W	Terminal entry address
		12	W	Routine base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code: X'00' - Successful completion X'04' - Unsuccessful completion
IEDQCL	IEDQCL	1	I	Operator Control AVT address
			O	Address of appropriate response message
		2	W	Operator Control AVT address
		4	W	AVT address
		5	W	DEB address
		7	W	DCB address
		8	W	UCB address
		9	W	Invitation List address
		12	W	Routine base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code: X'00' - Successful completion X'04' - Unsuccessful completion

IEDQCM	IEDQCM	1	I	Operator Control AVT address
			O	Address of the appropriate response message
		2	W	Operator Control AVT address
		4	W	AVT address
		5	W	Termname Table address
		6	W	Terminal entry address
		12	W	Routine base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code:
				X'00' - Successful completion
				X'04' - Unsuccessful completion
IEDQCN	IEDQCN	1	I	Operator Control AVT address
			O	Address of appropriate response message
		2	W	Operator Control AVT address
		4	W	AVT address
		5	W	Termname Table address
		6	W	Terminal entry address
		12	W	Routine base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code:
				X'00' - Successful completion
				X'04' - Unsuccessful completion
IEDQCO	IEDQCO	1	I	Operator Control AVT address
			O	Address of appropriate response message
		2	W	Operator Control AVT address
		4	W	AVT address
		5	W	Termname Table address
		6	W	Terminal entry address
		7	W	QCB address
		8	W	ICB address
		9	W	DCB address
		12	W	Routine base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code:
				X'00' - Successful completion
				X'04' - Unsuccessful completion
IEDQCP	IEDQCP	1	I	Operator Control AVT address
			O	Address of appropriate response message
		2	W	Operator Control AVT address
		4	W	AVT address
		6	W	DEB address
		7	W	DCB address
		9	W	ICB address
		11	W	UCB address
		12	W	Routine base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code:
				X'00' - Successful completion
				X'04' - Unsuccessful completion
IEDQCQ	IEDQCQ	1	I	Operator Control AVT address
			O	Address of appropriate response message
		2	W	Operator Control AVT address
		4	W	AVT address
		5	W	Termname entry address
		6	W	Terminal entry address
		10	W	QCB address

		11	W	DCB address
		12	W	Routine base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code:
				X'00' - Successful completion
				X'04' - Unsuccessful completion
IEDQCU	IEDQCU	1	I	Operator Control AVT address
			O	Address of appropriate response message
		2	W	Operator Control AVT address
		4	W	AVT address
		6	W	DEB address
		7	W	DCB address
		9	W	LCB address
		10	W	UCB address
		12	W	Routine base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code:
				X'00' - Successful completion
				X'04' - Unsuccessful completion
IEDQCV	IEDQCV	1	I	Operator Control AVT address
			O	Address of appropriate response message
		2	W	Operator Control AVT address
		4	W	AVT address
		6	W	DEB address
		7	W	DCB address
		9	W	LCB address
		10	W	UCB address
		12	W	Routine base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code:
				X'00' - Successful completion
				X'04' - Unsuccessful completion
				X'14' - Closedown is in progress
IEDQCW	IEDQCW	1	I	Operator Control AVT address
			O	Address of appropriate response message
		2	W	Operator Control AVT address
		4	W	AVT address
		5	W	UCB address
		7	W	DEB address
		8	W	DCB address
		12	W	Routine base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code:
				X'00' - Successful completion
				X'04' - Unsuccessful completion
IEDQCX	IEDQCX	1	I	Operator Control AVT address
			O	Address of appropriate response message
		2	W	Operator Control AVT address
		4	W	AVT address
		5	W	Termname Table address or DEB address
		6	W	Terminal entry address or LCB address
		9	W	DCB address
		10	W	UCB address
		12	W	Routine base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code:
				X'00' - Successful completion
				X'04' - Unsuccessful completion

IEDQCZ	IEDQCZ	1	I	Operator Control AVT address
			O	Address of appropriate response message
		2	W	Operator Control AVT address
		4	W	AVT address
		7	W	Termname Table address
		8	W	Terminal entry address
		9	W	QCB address
		10	W	DCB address
		11	W	ICB address
		12	W	Routine base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code: X'00' - Successful completion X'04' - Unsuccessful completion
IEDQC0	IEDQC0	14	I	Return address
		15	I	Entry point address
			O	Return code: X'00' - Processing complete X'04' - Invalid command X'03' - Canceled command X'16' - Load IEDQCV
IEDQC1	IEDQC1	1	I	Operator Control AVT address
			O	Address of appropriate response message
		2	W	Operator Control AVT address
		4	W	AVT address
		5	W	ICB address
		6	W	DCB address
		12	W	Routine base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code: X'00' - Successful completion X'04' - Unsuccessful completion
IEDQC2	IEDQC2	1	I	Operator Control AVT address
			O	Address of an error message if TOTE is not active
		14	I	Return address
		15	I	Entry point address
			O	Return code: X'04' - TOTE is not active X'08' - Command was canceled X'14' - Command is queued for TOTE
IEDQC3	IEDQC3	1	I	Operator Control AVT address
			O	Address of appropriate response message
		2	W	Operator Control AVT address
		4	W	AVT address
		5	W	DEB address
		7	W	DCB address
		8	W	UCB address
		10	W	ICB address
		12	W	Routine base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code: X'00' - Successful completion X'04' - Unsuccessful completion
IEDQC6	IEDQC6	0	O	On exit to service aid routine - indicates load or delete function
		1	I	Operator Control AVT address
		2	W	Operator Control AVT base register
		3	W	Work register
		4	W	AVT base register

		5	W	Work register
		6	W	Local return register
		7	W	Work register
		8	W	Index register
		12	I	Routine base register
		13	I	Calling routine save area address
		14	I	Return register
		15	I	Entry point address
			O	On return from the service aid routine, return code: X'00' - Good return X'04' - Verify error X'08' or higher - not defined Negative - complement of the address of the response message
IEDQEC	IEDQEC	1	I	Address of an ERB that points to an empty buffer or the address of a special element tposted to the Put Scheduler STCB
			O	Address of an ERB to be tposted to the Buffer Request routine or a full buffer to be tposted to an MH
		11	I	TCAM Dispatcher address
		13	I	TCAM Dispatcher save area address
		15	I	Entry point address
IEDQES	IEDQES	0	I	User work area address (contains terminal name)
			W	Work register
		1	I	Output sequence number (complement of input sequence number), or relative record address for subsequent retrieval
		2	W	Work register
		3	W	User's work area address
		4	W	Buffer address
		5	W	Process Control Block address
		6	W	Access method work area address
		7	W	Data Extent Block address
		8	W	Task Control Block address
		9	W	Terminal Table entry address
		10	W	Termname Table address
		11	W	Address Vector Table address
		12	I	Base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code register: X'00' - Successful completion X'04' - Invalid sequence number or address
IEDQET	IEDQET	1	I	CIB address
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code set by the Operator Control task, except when equal to X'01'. If equal to X'01', it is set by this routine to indicate that there is no active MCP in the system.
IEDQEU	IEDQEU	1	I	Address of special element Word 3 points to the process entry Word 4 points to the Executor ECB
		11	I	TCAM Dispatcher address
		13	I	Register save area address
		15	I	Entry point address

IEDQEW	IEDQEW	1	I	Address of the input element: - ERB containing a pointer to a full buffer or a return code indicating a logical read error. - A special "empty buffer" element, or - A special retrieve element.
			O	Address of an ERB or a chain of empty buffers
		11	I	TCAM Dispatcher address
		13	I	TCAM Dispatcher save area address
		15	I	Entry point address
IEDQEZ	IEDQEZ	1	I	Address of a full buffer or of a retrieve element.
			O	If the input was a retrieved element - the address of a Read-ahead QCB; if the input was a buffer - zero.
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code: X'04' - Input was a retrieve element X'00' - Input was a buffer
IEDQE1	IEDQE1	0	I	Termname Table entry address
		1	I	Application program work area address
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code: X'00' - Successful completion X'08' - TCAM is not in the system X'0C' - No open DCB in the application program X'20' - Invalid terminal name
IEDQE2	IEDQE2	0	I	Termname Table entry address
		1	I	Application program work area address
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code: X'00' - Successful completion X'04' - Invalid terminal name X'08' - TCAM is not in the system X'0C' - No open DCB in the application program X'20' - Invalid terminal type
IEDQE3	IEDQE3	1	I	Address of a three-word input parameter list: - Word 1 - address of terminal name - Word 2 - address of application program work area - Word 3 - address of unscrambled password, if specified
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code: X'00' - Successful completion X'04' - Invalid or missing password X'08' - TCAM is not in the system X'0C' - No open DCB in the application program X'20' - Invalid terminal name
IEDQE4	IEDQE4	0	I	Application program work area address
		1	I	Byte 0: relative line number Bytes 1-3: Address of ddname
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code: X'00' - Successful completion

				X'04' - Invalid relative line number
				X'0C' - TCAM is not in the system
				X'20' - Invalid DDNAME for line group DCB
IEDQE6	IEDQE6	0	O	First half of scrambled password
		1	I	Address of character string to be scrambled
			O	Second half of scrambled password
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
IEDQE7	IEDQE7	1	I	Address of a retrieve element, an ERB, or a Buffer Return element
		2	W	Special element
		3	W	Work register
		4	W	Terminal entry address
		5	W	Buffer prefix address
		6	W	Destination QCB address
		7	W	Queue Control Block address
		8	W	Dummy Line Control Block address
		9	W	Dummy Station Control Block address
		10	W	Process Control Block address
		11	W	TCAM Dispatcher address
		12	I	Base register
		13	W	TCAM Address Vector Table address
		14	I	Return address
		15	I	Entry point address
IEDQFA	IEDQFA	0	I	Address of the last element tposted
	IEDQFC	1	I	Address of the element tposted
		2	I	CPB unit address; address of QCB to tpost to: buffer unit address
		3	I	SCB address
		4	I	LCB address
		5	W	Work register
		6	I	Current buffer address
		7	W	Destination QCB address
		8	W	Address of last unit of the buffer; number of units in the buffer
		9	W	Address of priority Destination QCB
		10	W	DCB address; value of "address" for the disk record
		11	I	TCAM Dispatcher address
		12	I	Base register
		13	I	Calling routine save area address
		14	W	Work register
		15	W	CPB address
IEDQFA1	IEDQFA1	0	I	Address of the last element tposted
	IEDQFC	1	I	Address of the element tposted
		2	I	CPB unit address; address of QCB to tpost to: buffer unit address
		3	I	SCB address
		4	I	LCB address
		5	W	Work register
		6	I	Current buffer address
		7	W	Destination QCB address
		8	W	Address of last unit of the buffers; number of units in the buffer
		9	W	Address of priority Destination QCB
		10	W	DCB address; value of "address" for the disk record
		11	I	TCAM Dispatcher address
		12	I	Base register
		13	I	Calling routine save area address
		14	W	Work register
		15	W	CPB address

IEDOFA2	TFDQFA2	0	I	Address of the last element tposted
	IFDQFQ	1	I	Address of the element tposted
		2	I	CPB unit address; address of QCB to tpost to: buffer unit address
		3	I	SCB address
		4	I	ICB address
		5	W	Work register
		6	T	Current buffer address
		7	W	Destination QCB address
		8	W	Address of last unit of the buffer; number of units in the buffer
		9	W	Address of priority Destination QCB
		10	W	DCB address; value of "address" for the disk record
		11	T	TCAM Dispatcher address
		12	I	Base register
		13	I	Calling routine save area address
		14	W	Work register
		15	W	CPB address
IEDQGA	IEDQGA	0	W	Work register
		1	I	ERB address
		2	W	DCB unit count
		3	W	Work register
		4	W	ICB address
		5	W	Count of available buffers
		6	W	Buffer address
		7	I	QCB address
		8	W	Count reserved for disk
		9	W	Work register
		10	W	DCB address
		11	W	ERB buffer request count
		12	I	Base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
IEDQGA	IEDQGB	0	W	Work register
		1	I	Buffer address
		2	W	Address of the last unit of the last buffer
		3	W	Address of the last unit of the current buffer
		4	W	ICB address
		6	W	Buffer address
		7	I	QCB address
		8	W	Count reserved for disk
		9	W	Work register
		10	W	DCB address
		11	I	Dispatcher base register
		12	I	Base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
IEDQGA	IEDQGD	0	W	Work register
		1	I	AVTPARM address; AVTPARM contains the buffer address, AVTPARM+4 contains the AVT address
		3	W	Work register
		4	W	LCB address
		5	W	Address of the Read/Write idle loop used
		6	W	Buffer address
		7	I	QCB address
		8	W	Count reserved for disk
		9	W	Work register
		10	W	DCB address
		11	W	Address of the unused Read/Write idle loop
		12	I	Base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address

IEDQGT	IEDQGT	0	W	Work register
		1	W	Next unit address
		2	W	DCB address
		3	W	SCB address
		4	W	ICB address
		5	W	Buffer length
		6	I	Buffer address
		7	W	Number of bytes remaining in the unit
		8	W	Work register
		9	W	Count of bytes remaining in the ETB size
		10	W	Work register
		11	W	Work register
		12	W	Work register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
IEDQHG	IEDQHG	0	W	Divide and multiply register - even
		1	I	Input element address
			W	Divide and multiply register - odd; Work register
		2	W	Address of the element on the time delay queue
		3	W	Address of the element on the time delay queue
		4	W	Address of the time delay element being processed
		5	W	Time of day
		6	W	Time interval-even
		7	W	Time interval-odd
		8	W	Save area address
		9	I	AVT address
		10	W	Exit switch: X'00' - Return by BR 14 X'04' - Exit to DSPDISP X'08' - Exit to DSPPCST
		11	I	TCAM Dispatcher address
		12	I	Routine base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
IEDQHG	IEDQHG01	0	W	Divide and multiply register
		1	I	Time delay request element address
		2	W	Work register
		3	W	Address of the element on the time delay queue
		4	W	Address of the time delay element in the AVT
		5	W	Time of day
		6	W	Time interval; even multiply and divide register
		7	W	Time interval; odd multiply and divide register
		8	W	Save area address
		10	W	Exit switch: has the address of "RETURN" - Return by BR 14 "DSPDISP" - Exit to DSPDISP in the TCAM Dispatcher "POSTEXIT" - Exit to DSPPOST in the TCAM Dispatcher
		11	W	TCAM Dispatcher address
		12	W	Routine base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
IEDQHG	IEDQHG02	0	W	Divide and multiply register - even
		1	I	Input element address
			W	Divide and multiply register - odd
		2	W	Address of the element on the time delay queue
		3	W	Address of the element on the time delay queue
		4	W	Address of the time delay element being processed
		5	W	Time of day
		6	W	Time interval: even divide and multiply register

		7	W	Time interval; odd divide and multiply register
		8	W	Save area address
		10	W	Exit switch: has the address of "RETURN" - Return by BR 14 "DSPDISP" - Exit to DSPDISP in the TCAM Dispatcher "POSTEXIT" - Exit to DSPPOST in the TCAM Dispatcher
		11	W	TCAM Dispatcher address
		12	W	Routine base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
IEDQHG	IEDQHG03	0	W	Divide and multiply register - even
		1	I	Input element address
			W	Divide and multiply register-odd
		2	W	Address of the element of the time delay queue
		3	W	Address of the element on the time delay queue
		4	W	Address of the time delay element to be processed
		5	W	Time of day
		6	W	Address of special element requesting removal of an element from the time delay queue
		8	W	Calling routine save area address
		10	W	Exit switch: has the address of "RETURN - Return by BR "DSPDISP" - Exit to DSPDISP in the TCAM Dispatcher "POSTEXIT" - Exit to DSPPOST in the TCAM Dispatcher
		11	I	TCAM Dispatcher address
		12	W	Routine base register
		14	W	Work register
		15	I	Entry point address
IEDQHI	IEDQHI	0	W	Work register
		1	I	Input element - the system delay request element, an ICB, or the System Delay QCB
		3	W	TCB address
		4	W	ICB address
		5	W	DEB address
		6	W	Subroutine return address
		7	W	System Delay QCB address
		8	W	Branch switch
		10	W	DCB address
		11	W	TCAM Dispatcher address
		12	I	Routine base register
		13	I	AVT base register
		14	I	Return address
		15	W	Work register
IEDQHK	IEDQHK	1	I	Stopline request address or LCB address
		0	O	Address of the ICB to be tposted to the Operator Control queue
		3	W	ICB address
		6	W	DEB address
		7	W	Routine base register
		8	W	DCB address
		11	I	TCAM Dispatcher address
		12	W	TCB address
		13	I	Save area address for the AVT
		15	I	Entry point address
IEDQHM	IEDQHM	0	W	Work register
	IEDQHM02	1	I	Current buffer address
			W	Parameter to the Termname Table
		2	W	Address of the AVT "address" value being used
			O	Address of the QCB to tpost to
		3	I	Address of the STCB just activated
			W	SCB address
		4	W	ICB address
		5	W	Address of the buffer formed from the main storage units
		6	W	Current buffer address
		7	W	Master QCB address

		8	W	Priority QCB address
		9	W	Work register
		10	W	Work register
		11	W	Work register
		12	I	Base register
		13	I	AVT address
		14	I	Return address
			W	Number of main storage units to get
			O	Return address
		15	W	Work register
IEDQHM1	IEDQHM1	0	W	Work register
	IEDQHM02	1	I	Current buffer address
			W	Parameter to the Termname Table
		2	W	Address of the AVT "address" value being used
			O	Address of the QCB to bpost to
		3	I	Address of the STCB just activated
			W	SCB address
		4	W	ICB address
		5	W	Address of the buffer formed from the main storage units
		6	W	Current buffer address
		7	W	Master QCB address
		8	W	Priority QCB address
		9	W	Work register
		10	W	Work register
		11	W	Work register
		12	I	Base register
		13	I	AVT address
		14	I	Return address
			W	Number of main storage units to get
			O	Return address
		15	W	Work register
IEDQHM2	IEDQHM2	0	W	Work register
	IEDQHM02	1	I	Current buffer address
			W	Parameter to the Termname Table
		2	W	Address of the AVT "address" value being used
			O	Address of the QCB to bpost to
		3	I	Address of the STCB just activated
			W	SCB address
		4	W	ICB address
		5	W	Address of the buffer formed from the main storage units
		6	W	Current buffer address
		7	W	Master QCB address
		8	W	Priority QCB address
		9	W	Work register
		10	W	Work register
		11	W	Work register
		12	I	Base register
		13	I	AVT address
		14	I	Return address
			W	Number of main storage units to get
			O	Return address
		15	W	Work register
IEDQKA	IEDQKA	1	W	Parameter register
		2	I	ICB address
		3	W	Address of the current TP field in the ICB
		4	W	DCB address
		5	W	Work register
		6	W	Work register
		7	W	Work register
		8	W	Terminal entry address
		9	W	Work register
		10	W	CCW address
		11	I	Dispatcher base - points to model CCWs
		12	I	Base register
		13	I	AVT address
		14	I	Internal linkage register
		15	W	Work register

IEDQKB	IEDQKB	1	W	Parameter register
		2	I	LCB address
		3	W	Address of the current TP field in the LCB
		4	W	DCB address
		5	W	Work register
		6	W	Work register
		7	W	Work register
		8	W	Terminal entry address
		9	W	Work register
		10	W	CCW address
		11	I	Dispatcher base - points to model CCWs
		12	I	Base register
		13	I	AVT address
		14	I	Internal linkage register
		15	W	Work register
IEDQKC	IEDQKC	1	W	Parameter register
		2	I	LCB address
		3	W	Address of the current TP field in the LCB
		4	W	DCB address
		5	W	Work register
		6	W	Work register
		7	W	Work register
		8	W	Terminal entry address
		9	W	Work register
		10	W	CCW address
		11	I	Dispatcher base - points to model CCWs
		12	I	Base register
		13	I	AVT address
		14	I	Internal linkage register
		15	W	Work register
IEDQKD	IEDQKD	1	W	Parameter register
		2	I	LCB address
		3	W	Address of the current TP field in the LCB
		4	W	DCB address
		5	W	Work register
		6	W	Work register
		7	W	Work register
		8	W	Terminal entry address
		9	W	Work register
		10	W	CCW address
		11	I	Dispatcher base - points to model CCWs
		12	I	Base register
		13	I	AVT address
		14	I	Internal linkage register
		15	W	Work register
IEDQKE	IEDQKE	1	W	Parameter register
		2	I	LCB address
		3	W	Address of the current TP field in the LCB
		4	W	DCB address
		5	W	Work register
		6	W	Work register
		7	W	Work register
		8	W	Terminal entry address
		9	W	Work register
		10	W	CCW address
		11	I	Dispatcher base - points to model CCWs
		12	I	Base register
		13	I	AVT address
		14	I	Internal linkage register
		15	W	Work register
IEDQLM	IEDQLM	13	I	Calling routine save area address
		14	I	Return address
IEDQNA	IEDQNA3	1	I	TCB address of the terminating task
	IEDQNA	13	I	Calling routine save area address

IEDQNA2	IEDQNA2	1	I	AVTSAVE2 address, if from IEDQNA; negative TCB address, if from IEDQNA3
			O	If there is disk activity, the address of the Closedown Completion QCB
IEDQNB	IEDQNB01	13	I	Save area address
		14	I	Return address
		15	I	Entry point address
	IEDQNB02	1	I	Termname offset of the terminal specified in the TCHNG macro
		13	I	Save area address
		14	I	Return address
		15	I	Entry point address
	IEDQNB05	1	I	Address of OPEN or CLOSE macro DCB
		2	W	Checkpoint work area address
		5	W	DEB chain address
		6	W	PCB address
		9	W	AVT address
		10	W	Current TCB address
		11	W	Access method work area address
		12	I	Base register
		13	I	Save area address
		14	O	Return address
		15	I	IEDQNB05 address
			O	Return code
IEDQND	IEDQND	0	W	Work register
		1	W	Work register
		2	W	Address of the checkpoint work area
		3	W	Work register
		4	W	Work register
		5	W	Work register
		6	W	Work register
		7	W	Work register
		8	W	Work register
		9	W	Work register
		10	W	Work register
		11	W	Address of the checkpoint record
		12	W	Routine base register
		13	I	AVT address
		14	I	Return address
		15	I	Entry point address
IEDQNF	IEDQNF	2	I	Checkpoint work area address
		9	I	AVT address
		12	I	Base register
		14	I	Address of IEDQNF branch table
		15	O	Entry point address of the loaded module
IEDQNG	IEDQNG	2	I	Checkpoint work area address
		3	I	Address of the request element this module is to process
		4	O	Disk record address
		9	I	AVT address
		13	I	IEDQNF base register
		14	I	Return address
		15	I	Entry point address
			O	Offset to the next module to gain control: X'40' - Checkpoint Queue Manager X'58' - No Available Core routine X'6C' - No Incident Records routine
IEDQNH	IEDQNH	2	I	Checkpoint work area address
		3	I	Address of the request element this module is to process

		4	O	Disk record address
		9	I	AVT address
		12	I	IEDQNF base register
		14	I	Return address
		15	I	Entry pcint address
			O	Offset to the next module to gain control: X'40' - Checkpoint Queue Manager X'58' - No Available Ccre routine X'60' - No Incident Records routine
IEDONJ	IEDQNJ	2	I	Checkpoint work area address
		3	I	Address of the request element this module is tc process
		4	O	Disk record address
		9	I	AVT address
		12	I	IEDQNF base register
		14	I	Return address
		15	I	Entry pcint address
			O	Offset to the next module to gain control: X'40' - Checkpcint Queue Manager X'58' - No Available Core routine X'60' - No Incident Records routine
IEDONK	IEDQNK	2	I	Checkpoint work area address
		3	I	Address of the request element this module is to process
		4	O	Disk record address
		9	I	AVT address
		12	I	IEDQNF base register
		14	I	Return address
		15	I	Entry pcint address
			O	Offset to the mext module to gain control: X'40' - Checkpcint Queue Manager X'48' - Checkpcint Disk I/O routine X'58' - No Available Core routine
IEDONM	IEDQNM	2	I	Checkpcint wcrk area address
		3	I	Address of the request element this module
		4	O	Disk record address
		9	I	AVT address
		12	I	IEDQNF base register
		14	I	Return address
		15	I	Entry pcint address
			O	Offset to the next mcdule to gain control: X'40' - Checkpcint Queue Manager X'58' - No Available Ccre routine
IEDQNO	IEDQNC	2	I	Checkpcint wcrk area address
		3	I	Address of the last request element for which a disk reccrd was built
		4	I	Address of the last disk record built
		9	I	AVT address
		12	I	IEDQNF base register
		14	I	Return address
		15	I	Entry point address
IEDQNP	IEDQNP	2	I	Checkpoint work area address
		9	I	AVT address
		12	I	IEDQNF base register
		14	I	Return address
		15	I	Entry pcint address
			O	Return code: X'30' - CKREQ is incomplete X'38' - Environment Checkpoint is incomplete
IEDQNQ	IEDQNQ	2	I	Checkpoint work area address
		9	I	AVT address
		12	I	IEDQNF base register
		14	I	Return address
		15	I	Entry point address

IEDQNR	IEDQNR	2	I	Checkpoint work area address
		3	I	Address of the request element this module is to process
		9	I	AVT address
		12	I	IEDQNF base register
		14	I	Return address
		15	I	Entry point address
			O	If there are no outstanding GETMAIN records, the offset to the Notification and Disposition routine (X'50')
IEDQNS	IEDQNS	2	I	Checkpoint work area address
		9	I	AVT address
		12	I	IEDQNF base register
		14	I	Return address
		15	I	Entry point address
IEDQNX	IEDQNX	1	I	Address of chain of elements to be tposted to the ready queue
			O	Address of chain of elements to be tposted to the TCAM Dispatcher
		2	I	Address of QCB for IEDQBD02 routine
		3	I	SCB address
		4	I	LCB address
		5	I	LCB address
		9	I	Priority to be put in the LCB for the tpost to IEDQBD02
		11	I	TCAM Dispatcher address
		12	I	IEDQNX address
		13	I	AVTSAVE2 address
IEDQCA	IEDQOA	1	I	AVT address
		2	W	AVTSAVE3 address
		3	W	Error code
		9	I	AVT address
		12	I	Base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code: X'00' - Successful completion X'04' - TCAM already in the system X'08' - Insufficient main storage to satisfy GETMAIN request for permanent storage X'0C' - Insufficient main storage to satisfy GETMAIN request for temporary storage X'10' - Terminal definition error X'14' - Primary operator control terminal improperly defined
IEDQOB	IEDQOB	1	I	Input parameter; AVT address
			W	CVT address
		2	W	Response area address; start of keyword address
		3	W	Return address for subroutines
		4	W	Address of the AVT field to be modified
		5	W	Address of the first character beyond the operand
		6	W	Binary copy of the decimal character keyword
		7	W	Number of bytes in the keyword
		8	W	Work register
		9	W	Index to the list of keywords
		10	W	Incrementor of the index through the keyword table (Value = 2)
		11	W	Stopping point in the keyword table search loop
		12	W	Base register
		13	W	AVT base register
		14	I	Return address to IEDQCA
			W	Return address from an internal subroutine
		15	I	Entry point address
			O	Return code: X'00' - Successful completion X'04' - TCAM is already in the system (CVT word not = zero)

IEDQOG	IEDQOG	1	I	AVT address
		13	I	Calling routine save area address
		15	O	Return code: X'00' - Successful completion X'08' - Insufficient main storage to satisfy the GETMAIN request
IEDQOM	IEDQOM	0	W	Work register
		1	I	AVT address
			W	Address of current Termname Table entry
		2	W	Length in bytes of a full Termname Table entry
		3	W	Length of a Termname Table entry minus one
		4	W	Length of a Termname Table entry minus three
		5	W	Address of the last entry in the Termname Table
		6	W	Even numbered register for divide operations; address of the next offset entry
		7	W	Total number of entries in the Termname Table
		8	W	Address of the first entry in the Termname Table
		9	W	AVT base register
		10	W	Address of the end of the offset entries
		11	W	Work register
		12	I	Routine base register
		13	I	Calling routine save area address
			W	Address of the beginning of the Termname Table
		14	W	Address of the current offset entry
		15	W	Address of the next Termname Table entry
			O	Return code: X'00' - Successful completion X'12' - Insufficient main storage to satisfy the GETMAIN request X'16' - Terminal definition error X'20' - Primary operator control terminal definition error
IDEQOS	IEDQOS	1	I	Parameter register
		2	I	ECB address
		3	I	AVT address
		9	I	AVT base address
		12	I	Routine base register
		13	I	Save area address
		14	I	Return address
		15	W	Work register
IEDQTNT	IEDQTNT	0	W	Work register
		1	I	Ordinal index to a Termname Table entry
			O	Termname Table address of an entry
		14	I	Return address
		15	I	Entry point address
IEDQUI	IEDQUI	0	W	Work register
		1	I	Address of the input parameter list for the routine to be executed; the first two bytes of this list contain the index to the address of the appropriate routine in the MH VCON table.
		3	W	SCB address
		4	W	LCB address
		6	W	Current buffer address
		9	W	AVT address
		12	I,O	Base register
		13	I,O	Calling routine save area address
		15	W	Index byte
IEDQXA	IEDQXA	0	W	Parameter list address for GETMAIN or FREEMAIN
		1	I	Parameter list address for OPEN, CLOSE, WRITE, CHECK, FEOV, WTO
		2	W	Address of the message for WRITE and WTO

		3	W	Return code
		4	W	Return address from internal subroutines
		5	W	WRITE return code
		6	W	Volume counter
		7	W	Record counter
		8	I	JFCB address
		9	O	Message text address
		10	W	Length of the message, block size
		11	W	Work area buffer address
		12	W	SYSPRINT or dat DCBs address
		13	I	Routine base register; save area address
		14	I	Return address
			W	Scratch register
		15	I	Entry point address
			W	Scratch register
			O	Return code:
				X'00' - Successful completion
				X'04' - I/O error on SYSPRINT
				X'08' - KEYLEN omitted or invalid
				X'0C' - I/O error on IEDQDATA
				X'10' - Illegal SPACE parameters
				X'14' - Unable to open
IGC0010D	IGC0010D	0	I	Entry Code
(Entry code=1)		1	I,W	Operator control AVT address; work register
		2	W	Operator control AVT address
		4	W	AVT address
		7	W	Operator control save area address
		11	O	Entry code
		12	W	Routine base register
		13	I,O	Save area address
IGC0010D	IGC0010D	0	I,O	Entry code
(Entry code=2)		1	I	Operator control AVT address
		2	W	Operator control AVT address
		3	W	CIB address
		5	I	Return address
		6	W	Compare register
		12	W	Routine base register
		14	I	Return address
IGC0010D	IGC0010D	0	I	Entry Code
(Entry Code=3)		1	I	Operator control AVT address
		5	I	Return address
		12	W	Routine base register
		14	I	Return register
		15	O	Return code
				X'00' No errors detected
				X'02' Errors detected
IGC0010D	IGC0010D	0	I,W	Entry code; work register
(Entry code=4)		1	I	Operator control AVT address
		2	W	Operator control AVT address
		3	W	CIB address
		5	W	Scan pointer
		6	W	Work register
		7	W	Work register
		8	W	Buffer address
		12	W	Routine base register
		14	I	Return address
		15	O	Length scanned
IGC0110D	IGC0110D	2	I	Operator control AVT address
(Entry Code=1)		4	W	AVT address
		7	I	Buffer address
		11	I,O	Entry code
		12	W	Routine base register
		14	I	Return address

IGC0110D	IGC0110D	0	W	Module address
(Entry code=2)		1	W	Module index
		2	I	Operator control AVT address
		5	W	Index register
		6	W	Length of the entry
		7	I	Buffer address
		9	W	Pointer to the verb table
		11	I	Entry code
		13	W	Save area address
		14	I,W	Return address; BAL return
		15	I	Return code
			C	Exit address
IGC0110D	IGC0110D	0	O	Parameter address
(Entry code=3)		1	O	Parameter address
		2	I	Operator control AVT address
		3	W	CIB address
		5	W	Index register
		6	W	Length of the entry
		7	I,W	Buffer address
		9	W	Pointer to the verb table
		11	I,O	Entry code
		12	W	Routine base register
		14	I	Return address
IGC0110D	IGC0110D	2	I	Operator control AVT address
(Entry code=4)		3	W	CIB address
		4	W	AVT address
		5	W	Index register
		6	W	Length of the entry
		7	I,W	Buffer address
		9	W	Pointer to the verb table
		11	I,O	Entry code
		12	W	Routine base register
		13	I	Save area address
		14	I,W	Return address; BAL return
IGC0210D	IGC0210D	2	I,W	Operator control AVT address
(Entry code=1)		4	W	AVT address
		5	W	BAL (link) address
		7	W	TCB address
		9	W	Branch and link return address
		11	I,O	Entry code
		12	W	Routine base register
IGC0210D	IGC0210D	0	O	Entry code
(Entry code=2)		1	C	Operator control AVT address
		2	I,W	Operator control AVT address
		5	W	BAL (link) address
		9	W	Option Table address
		11	I,O	Entry code
		12	W	Routine base register
		14	I,W,O	Return address
		15	I	Length of the field
IGC0310D	IGC0310D	1	I,W	Address of the message
(Entry code=1)		2	I,O	Operator control AVT address
		4	I,O	AVT address
		7	W	Work register
		11	I,O	Entry code
		12	W	Routine base register
IGC0310D	IGC0310D	1	W	Parameter for DELETE, WTO, AQCTL
(Entry code=2)		2	I,O	Operator control AVT address
		3	I	CIB address
		4	I,O	AVT address
		6	W	Work register
		7	W	Work register
		11	I,O	Entry code
		12	W	Routine base register
		15	W	Save area address

IGC0310D	IGC0310D	1	I,O	Address of the message
(Entry code=3)		2	I,O	Operatcr ccntrol AVT address
		4	I,O	AVT address
		7	W	Work address
		11	I,O	Entry code
		12	W	Routine base register
IGC0410D	IGC0410D	0	O	Entry code
(Entry code=1)		1	O	Operatcr ccntrol AVT address
		2	I,O	Cperator ccntrcl AVT address
		3	W	Scan pcinter
		4	I,O	AVT address
		5	I	Return address
			W	Scan offset
		6	W	Work register
		8	W	Buffer address
		9	W	Prefix address
		10	W	Work register
		11	I,O	Entry code
		12	W	Rcutine base register
		14	I	Return address
			W	BAL address
IGC0410D	IGC0410D	2	I,O	Operator control AVT address
(Entry code=2)		4	I,O	AVT address
		11	I,O	Entry code
		12	W	Routine base register
IGC0410D	IGC041CD	0	W	Wcrk register
(Entry code=3)		1	W	Termname entry length; SCB address
		2	I,O	Operator ccntrol AVT address
		3	W	Scan pcinter
		4	I,O	AVT address
		6	W	Termname table address
		7	W	ICB address
		8	W	Pointer to prefix
		10	W	Work register
		11	I,O	Entry code
		12	W	Routine base register
IGC0510D	IGC0510D	0	W	Work register
		1	W	Wcrk register, parameter
		3	I	Buffer address
			W	ICB address; prefix address
		4	I,O	AVT address
		5	W	ERB address
		6	W	Keylength
		7	W	Number of units in the buffer
		8	W	Prefix address
		9	W	Address of the message; length of the message
		10	W	Wcrk register
		11	O	Entry code
		12	W	Rcutine base, register
		15	I	Entry pcint address
IGC102	IGC102	1	I	Input parameter list address
		10	W	IEACFT01 address; Completion code
		11	W	Complement of the ECB address
		13	W	TJID for the ECB to be tposted in the lcw-crder sixteen bits
		14	I	Return address
		15	O	Return code: X'00' - Successful completion X'04' - No active TCAM MCP in the system
IGC1303D	IGC1303D	2	I	Extended save area address
		14	I	Return address
IGE0004G	IGE0004G	1	I	12 Star address
		3	W	UCB address

		4	W	LCB address
		10	W	CCW address
		11	W	AVT base register.
		12	W	SCB address
		13	W	Linkage for next module load
		14	W	XCTL register
		15	I	Base register
IGE0104G	IGE0104G	1	I	12 Star address
		3	W	UCB address
		4	W	LCB address
		5	W	SCB address
		6	W	CCW address
		10	W	DCB address
		11	W	AVT base register
		13	W	Linkage for next module load
		14	W	XCTL register
		15	I	Base register
IGE0204G	IGE0204G	1	I	12 Star address
		3	W	UCB address
		4	W	LCB address
		13	W	Linkage for next module load
		15	I	Base register
IGE0304G	IGE0304G	1	I	12 Star address
		3	W	UCB address
		4	W	LCB address
		5	W	SCB address
		6	W	CCW address
		11	W	AVT base register
		13	W	Linkage for next module load
		14	W	XCTL register
		15	I	Base register
IGE0404G	IGE0404G	1	I	12 Star address
		2	W	SCB address
		3	W	UCB address
		4	W	LCB address
		5	W	DCB address
		6	W	CCW address
		11	W	AVT base register
		13	W	Linkage for next module load
		14	W	XCTL register
		15	I	Base register
IGE0504G	IGE0504G	1	I	12 Star address
		2	W	LCB base register
		3	W	UCB address
		4	W	DCB address
		5	W	CCW base register
		11	W	AVT base register
		13	W	Linkage for next module load
		14	W	XCTL register
		15	I	Base register
IGE0604G	IGE0604G	1	I	12 Star address
		3	W	UCB address
		4	W	LCB address
		5	W	SCB address
		6	W	CCW address
		11	W	AVT base register
		13	W	Linkage for next module load
		14	W	XCTL register
		15	I	Base register
IGE0804G	IGE0804G	1	I	12 Star address
		13	W	Linkage for next module load
		14	W	XCTL register
		15	I	Base register

IGE0904G	IGE0904G	1	I	12 Star address
		2	W	LCB address
		12	W	AVT address
		13	W	Linkage for next module load
		15	I	Base register
IGE0004H	IGE0004H	1	I	12 Star address
		2	W	CCW base register
		3	W	UCB address
		4	W	LCB address
		5	W	DCB address
		11	W	AVT base register
		12	W	SCB address
		13	W	Linkage for next module load
		14	W	XCTL register
		15	I	Base register
IGE0104H	IGE0104H	1	I	12 Star address
		2	W	LCB base register
		3	W	SCB base register
		4	W	DCB base register
		5	W	CCW base register
		11	W	AVT base register
		13	W	Linkage for next module load
		14	W	XCTL register
		15	I	Base register
IGE0204H	IGE0204H	1	I	12 Star address
		2	W	LCB base register
		4	W	DCB base register
		5	W	CCW base register
		10	W	SCB address
		11	W	AVT base register
		13	W	Linkage for next module load
		14	W	XCTL register
		15	I	Base register
IGE0404H	IGE0404H	1	I	12 Star address
		2	W	LCB base register
		3	W	SCB base register
		4	W	DCB base register
		5	W	CCW base register
		11	W	AVT base register
		13	W	Linkage for next module load
		14	W	XCTL register
		15	I	Base register
IGE0504H	IGE0504H	1	I	12 Star address
		2	W	LCB base register
		3	W	UCB base register
		4	W	DCB base register
		5	W	CCW base register
		11	W	AVT base register
		14	W	XCTL register
		15	I	Base register
IGE0804H	IGE0804H	1	I	RQE address
		2	W	LCB base register
		3	W	UCB address
		12	W	CCW base register
		13	W	Linkage for next module load
		14	W	XCTL register
		15	I	Base register
IGG019Q0	IGG019Q0	1	I	IO Supervisor register
		2	I	LCB address
		3	W	DEB address
		4	I	DCB address
		5	W	Address of the Current I/O Interrupt Trace Table entry

		6	W	I/O Trace control words address
		7	W	Current line's UCB address
		8	W	Work register
		9	W	Work register
		10	W	Work register
		11	I	Return address in IGG019R0
		12	I	IGG019Q0 base register
		13	I	AVT address
		14	I	SCB address
		15	I	IGG019R0 base register
IGG019Q1	IGG019Q1	1	I	Attention element or LCB address
		4	O	LCB address
		10	O	DCB address
IGG019Q2	IGG019Q2	2	I	LCB address
		4	I	DCB address
		6	O	Prefix address
		12	O	Interrupted CCW address
		13	O	AVT address
		14	O	SCB address
IGG019Q3	IGG019Q3	2	I	LCB address
		4	I	DCB address
		6	O	Prefix address
		12	O	Interrupted CCW address
		13	O	AVT address
		14	O	SCB address
IGG019Q4	IGG019Q4	2	I	LCB address
		4	I	DCB address
		6	O	Prefix address
		12	O	Interrupted CCW address
		13	O	AVT address
		14	O	SCB address
IGG019Q5	IGG019Q5	2	I	LCB address
		4	I	DCB address
		6	O	Prefix address
		12	O	Interrupted CCW address
		13	O	AVT address
		14	O	SCB address
IGG019Q6	IGG019Q6	1	I	Buffer, LCB, or QCB address
		4	O	LCB address
		7	O	QCB address
IGG019Q7	IGG019Q7	1	I	Buffer, LCB, or QCB address
		4	O	LCB address
		7	O	QCB address
IGG019Q8	IGG019Q8	2	I	Checkpoint work area address
	IGG019Q8+4	12	I	Base register
	IGG019Q8+8	14	I	Return address
	IGG019Q8+12			
	IGG019Q8+16			
IGG019RA	IGG019RA	1	I	Address of the request element that this module is to process
		2	I	IOB address
		3	I	DEB address
		4	I	DCB address
		7	I	UCB address
		14	I	Return address
		15	I	Entry point address
IGG019RB	DSPBYPAS	1	I	Address of the element to pass
			O	Address of the last dispatched RCB
		3	I	Address of the STCB controlling the subtask
			O	Address of the last dispatched STCB
		7	I	Address of the QCB controlling the subtask

			O	Address of the last dispatched QCB
		11	I	Address of the TCAM Dispatcher (IGG019RB)
		12	I	AVTSAVE2 address
		14	O	Return address
		15	O	Entry point of the subtask to be dispatched
IGG019RB	DSPCHAIN	1	I	Address of the first item in a chain to be tposted
			O	Address of the last dispatched RCB
		3	O	Address of the last dispatcher STCB
		7	O	Address of the last dispatched QCB
		11	I	Address of the TCAM Dispatcher (IGG019RB)
		12	I	AVTSAVE2 address
		14	O	Return address
		15	O	Entry point of the subtask to be dispatched
IGG019RB	DSPDISP	1	O	Address of the last dispatched RCB
		3	O	Address of the last dispatched STCB
		7	O	Address of the last dispatched QCB
		11	I	Address of the TCAM Dispatcher (IGG019RB)
		12	I	AVTSAVE2 address
		14	O	Return address
		15	O	Entry point of the subtask to be dispatched
IGG019RB	DSPDLETE	1	I	Address of the first item in a chain to be tposted
			O	Address of the last dispatched RCB
		3	O	Address of the last dispatched STCB
		7	O	Address of the last dispatched QCB
		11	I	Address of the TCAM Dispatcher (IGG019RB)
		12	I	AVTSAVE2 address
		14	O	Return address
		15	O	Entry point of the subtask to be dispatched
IGG019RB	DSPLIFO DSPLIFOB	1	I	Address of an item
			O	Address of the last dispatched RCB
		3	O	Address of the last dispatched STCB
		7	I	Address of the chain to receive the item
			O	Address of the last dispatched QCB
		11	I	Address of the TCAM Dispatcher (IGG019RB)
		12	I	AVTSAVE2 address
		14	O	Return address
		15	O	Entry point of the subtask to be dispatched
IGG019RB	DSPLIST	1	I	Address of a list of items to be tposted
			O	Address of the last dispatched RCB
		3	O	Address of the last dispatched STCB
		7	O	Address of the last dispatched QCB
		11	I	Address of the TCAM Dispatcher (IGG019RB)
		12	I	AVTSAVE2 address
		14	O	Return address
		15	O	Entry point of the subtask to be dispatched
IGG019RB	DSPPOST DSPPOSTP	1	I	Address of an RCB to be tposted
			O	Address of the last dispatched RCB
		3	O	Address of the last dispatched STCB
		7	O	Address of the last dispatched QCB
		11	I	Address of the TCAM Dispatcher (IGG019RB)
		12	I	AVTSAVE2 address
		14	O	Return address
		15	O	Entry point of the subtask to be dispatched
IGG019RB	DSPPRIO DSPPRIOB	1	I	Address of an item
			O	Address of the last dispatched RCB
		3	O	Address of the last dispatched STCB
		7	I	Address of the chain to receive the item
			O	Address of the last dispatched QCB
		11	I	Address of the TCAM Dispatcher (IGG019RB)
		12	I	AVTSAVE2 address
		14	O	Return address
		15	O	Entry point of the subtask to be dispatched

IGG019RB	DSPTSTQ	1	O	Address of the last dispatched RCB
	DSPTSTQR	3	I	Address of the desired QCB
			O	Address of the last dispatched STCB
		7	I	Address of the QCB that currently has the STCB at the top of its STCB chain
			O	Address of the last dispatched QCB
		11	I	Address of the TCAM Dispatcher (IGG019RB)
		12	I	AVTSAVE2 address
		14	O	Return address
		15	O	Entry point of the subtask to be dispatched
IGG019RB	DSPUNAV	1	O	Address of the last dispatched RCB
	DSPUNAVR	3	I	Address of the desired QCB
			O	Address of the last dispatched STCB
		7	I	Address of the QCB that currently has the STCB at the top of its STCB chain
			O	Address of the last dispatched QCB
		11	I	Address of the TCAM Dispatcher (IGG019RB)
		12	I	AVTSAVE2 address
		14	O	Return address
		15	O	Entry point of the subtask to be dispatched
IGG019RB	DSPWAIT	1	O	Address of the last dispatched RCB
		3	I	Address of the QCB from which an RCB is to be obtained
			O	Address of the last dispatched STCB
		7	I	Address of the QCB that contains the STCB to receive the element
			O	Address of the last dispatched QCB
		11	I	Address of the TCAM Dispatcher (IGG019RB)
		12	I	AVTSAVE2 address
		14	O	Return address
		15	O	Entry point of the subtask to be dispatched
IGG019RC	IGG019RC	0	W	Cylinder ID of current arm position
		1	W	ICB address; work register
			O	Nonreusable disk threshold closedown element
		2	W	Cylinder ID of the next CPB on the new queue
		4	W	Address of the previous CPB on the new queue
		5	W	DEB address
		6	W	Subroutine return address
		7	W	CPB address
		8	W	Number of extents
		9	W	Work register
		10	W	Address of the next CPB on the new queue; modified CPB address
		11	W	IOB address
			O	Dispatcher address
		12	W	Base register
		13	I	AVTSAVE2 address
		14	W	IOB size; subroutine return address
		15	I	Entry point address
			W	Work register
IGG019RC	IEDQFP	0	W	Threshold value; even divide register
		1	W	Absolute disk record number; odd divide register
			O	Abend code for Abend; message address for WTO
		2	W	Even divide register
		3	W	Odd divide register
		6	W	CPB address
		7	W	DEB address
		8	W	Number of extents
		9	W	Number of records per track
		10	W	Number of tracks per cylinder
		11	W	Number of records in the data set
		12	W	Product of the number of volumes x number of records per track

		13	I	AVTSAVE2 address
		14	I	IGG019RC return address
		15	I	Entry point address; base register
IGG019RD	IGG019RD	1	I	ICB or buffer address
		3	I	STCB address
		7	I	QCB address
		11	I	TCAM Dispatcher address
		13	I	AVTSAVE2 address
		15	I	Entry point address
IGG019RE	IGG019RE	0	W	Work register
		1	W	Parameter list address
		5	W	DEB address
		7	W	CPB address
		11	W	IOB address
			O	TCAM Dispatcher address
		12	W	Routine base register
		13	I,O	Address of AVTSAVE2
		14	W	IOB size; subroutine return register
		15	I	Entry point address
			W	Work register
IGG019RF	IEDQFP - see IGG019RC IEDQFP			
IGG019RG	IGG019RG	0	I	GET: the address of the application program work area.
		1	I	GET: the address of the input DCB READ: the address of the DECB
		13	I	Register save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code from a QSAM operation: X'00' - Successful completion X'04' - SETECF condition with no EODAD specified X'08' - Work area overflow
IGG019RH	IGG019RH	0	I	Address of the application program work area
		1	I	DCB address
		13	I	Register save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code: X'00' - Successful completion X'04' - Work area overflow
IGG019RI	IGG019RI	0	I	Address of the application program PUT/WRITE work area (not applicable for locate mode)
		1	I	PUT: the address of the PUT DCB (QSAM) WRITE: the address of the DECB (BSAM)
			O	If Locate mode - PUT/WRITE work area address
		13	I	Register save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code for a PUT (QSAM) operation: X'CC' - Successful completion X'0C' - Invalid terminal name or Terminal Table offset X'08' - Message segments are not in the proper sequence
IGG019RJ	IGG019RJ	0	I	Address of the application program PUT/WRITE work area
		1	I	PUT DCB address
		13	I	Register save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code: X'00' - Successful completion

				X'40' - Message segments or records not in proper sequence
IGG019RK	IGG019RK	1	I	12 star address
		2	I	IOB address
		6	W	Temporary AVT base
		8	I,O	IOS register
		9	W	AVT address
		11	I	CPB Cleanup QCB address; Multiprocessor CVT address
		11	W	ECB address for OS Post
		12	W	Base register: temporary storage of the TCB address during the OS Post
		14	I	Return address
			W	TCB address; Return address from OS Post
		15	I	Entry point address
			W	Work register; internal subroutine entry register
IGG019RL	IGG019RL	1	I	DECB address
		13	I	User-provided register save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code: X'00' - successful completion X'04' - SETECF without EODAD X'08' - work area overflow (READ) or sequence error (WRITE) X'0C' - invalid destination
IGG019RM	IGG019RM	0	I	Address of the 11-byte input retrieve data area: Bytes 0-7 - Terminal name (left adjusted and padded with blanks) to initiate retrieval; blank to terminate retrieval Bytes 8-9 - Message sequence number Byte 10 - Sequence number type (I for input, O for output)
		1	I	DCB address
		2	W	Work register
		3	W	Termname Table address
		4	W	AVT address
		5	W	DCB address
		6	W	DEB address
		7	W	Access method work area address
		12	I	Routine base register
		13	I	Calling routine save area address
		14	I	Return address
		15	I	Entry point address
			O	Return code: X'00' - Successful completion X'04' - Invalid sequence number type specified X'08' - Invalid terminal name specified
IGG019RN	IGG019RN	0	I	Reserved register
		1	I	12 star address
		2	I	IOB address
		3	I	DEB address
		4	I	DCB address
		5	I	Reserved register
		6	I	Reserved register
		7	I	Reserved register
		8	I	Reserved register
		9	I	Current buffer address
		10	I	AVT address
		11	W	Work register
		12	I	Base register
		13	W	Work register
		14	I	IOS return address
		15	W	Work register

IGG019RO - see IGG019RB

IGG019RP	REUS	0	W	Work register
		1	W	Work register
		2	W	Work register
		3	W	SCB address
		4	W	LCB address
		5	W	Termname Table entry address
		6	W	Buffer address
		7	W	QCB address
		8	W	Work register
		9	W	Unit work area address
		10	W	DCB address
		11	W	CPB address
			O	TCAM Dispatcher address
		12	I	Routine base register
			O	Address of IEDQFA+2
		13	I	Calling routine save area address
		14	W	Work register
		15	I	Entry point address
			O	Address of IEDQFA02
IGG019RP	COPY	0	W	Work register
		1	I	Address of the first unit of the buffer to be copied
		2	W	Work register
		3	W	SCB address
		4	W	LCB address
		5	W	Termname Table entry address
		6	W	Buffer address
		7	W	QCB address
		9	W	Unit work area base address
		10	W	DCB address
		11	W	CPB address
			O	Dispatcher address
		12	I	Routine base register
			O	Address of IEDQFA+2
		13	I,O	Calling routine save area address
		14	W	Work register
		15	I	Entry point address
			O	Address of IEDQFA02
IGG019RQ	IGG019RQ	0	W	Work register
		1	I	Parameter register
		2	I	DEB base register
		3	I	CVT base register
		4	I	TCB address
		9	I	AVT address
		10	I	Process entry work area address
		11	W	Work register
		12	I	Routine base register
		13	W	Work register
		14	I	Return address
		15	I	CVT address
IGG019R0	IGG019R0	2	I	LCB address
		4	I	DCB address
		6	O	Prefix address
		12	O	Interrupted CCW address
		13	O	AVT address
		14	O	SCB address
IGG019R1	IGG019R1	1	I	LCB address or QCB address
		4	O	LCB address
		7	O	QCB address
		10	O	LCB address

IGG019R2	IGG019R2	1	I	12 star address
		2	I	IOB address
		6	W	Temporary AVT base address
		8	I,O	IOS register
		9	I,O	IOS register
		10	W	AVT address
		11	W	CPB Cleanup QCB addres: CPB address
		12	W	Base register ECB address; Temporary storage of the TCP address during the OS Post
		13	I	Calling routine save area address
		14	I	Return address
			W	Work register
		15	I	Entry pcint address
			W	Work register
IGG019R3	IGG019R3	1	I	LCB address
		4	O	LCB address
		10	O	DCB address
IGG019R4	IGG019R4	1	I	Buffer address,LCB address,or QCB address
		4	O	LCB address
		7	O	QCB address
IGG019R5	IGG019R5	3	O	DEB address
		6	O	DCB address
		7	I	UCB address
		9	O	LCB address
		13	I	AVT address
IGG019R6	IGG019R6	1	I	LCB address
		2	W	IOB address from the DCB
		3	W	IOB address in the LCB
		4	W	Termname Table address
		5	W	Length of the terminal name
		6	W	Terminal name address
		7	W	Terminal entry address
		8	W	QCB address
		9	W	LCB address
		10	W	DCB/SCB address
		11	W	TCAM Dispatcher address
		12	I	Routine base register
		13	I	Calling routine save area address
		15	I	Entry point address
IGG01930	IGG01930	0	W	Work register
		1	W	Work register
		2	W	Current DCB address
		3	W	TIOT address
		4	W	DCB work area address
		5	I	Address of the first entry in the DCB parameter list
		6	I	Address of the Where-to-Go Table
		7	I	Address of the current entry in the DCB parameter list
			O	Updated to the next entry in the DCB parameter list
		8	I	Address of the current entry in the Where-to-Go Table
			O	Updated to the next entry in the Where-to-Go Table
		9	W	AVT address
		10	W	Current UCB address
		11	W	DEB address
		12	I	Routine base register
		13	W	Work register
		14	W	Work register
		15	W	Work register

IGG01931	IGGC1931	0	W	Wcrk register
		1	W	Work register
		2	W	Current DCB address
		3	W	Work register
		4	W	DCB wcrk area address
		5	I	Address of the first entry in the DCB parameter list
		6	I	Address of the Where-to-Go Table
		7	I	Address of the current entry in the DCE parameter list
			O	Updated to the next entry in the DCB parameter list
		8	I	Address of the current entry in the Where-to-Go Table
			O	Updated to the next entry in the Where-to-Go Tatle
		9	W	AVT address
		10	W	Work register
		11	W	Work register
		12	I	Routine base register
		13	W	Work register
		14	W	Work register
		15	W	Work register
IGG01933	IGG01933	0	O	Error code to user-specified error routine
		1	O	Opticn code to user-specified error routine
		2	I	Address of the current DCB
		3	W	Work register
		4	I	Address of the DCB Open Work Area
		5	I	Address cf the first entry in the DCE parameter list
		6	I	Address of the Where-to-Go Table
		7	I	Address of the current entry in the DCB parameter list
			O	If register 15 = 0, updated to the next entry in the DCB parameter list If register 15 = 1, the address of the DCB parameter list entry for the module that detected the error If register 15 is greater than 1, destroyed
		8	I	Address of the current entry in the Where-to-Go Table
			O	If register 15 = 0, updated to the next entry in the Where-to-Go Table
		9	I	AVT address
		10	W	Work register
		11	I	Address of the current DEB
		12	I	Routine base address
		13	W	Work register
		14	W	Work register
		15	I	Return Code: X'00' - Ignore the data set in error X'01' - Continue processing with the error causing limited capabilities X'02 - ABEND the TCAM job
IGG01934	IGG01934	0	W	Work register
		1	W	Work register
		2	I	Current DCB address
		3	I	Routine base register
		4	I	DCB work area address
		5	I	Address of the first entry in the DCB parameter list
		6	I	Address of the Where-to-Go Table
		7	I	Address of the current entry in the DCB parameter list
			O	Updated to the next entry in the DCB parameter list
		8	I	Address of the current entry in the Where-tc-Go Tatle
			O	Updated to the next entry in the Where-to-Gc Table

		9	I	AVT address
		10	I	Current UCB address
		11	I	DEB address
		12	I	TIOT address
		13	W	Work register
		14	W	Work register
		15	W	Work register
IGG01935	IGG01935	0	W	Work register
		1	W	Work register
		2	W	Current DCB address
		3	W	TIOT address
		4	W	DCB work area address
		5	W	DCB parameter list address
		6	I	Where-to-Go Table address
		7	I	Address of the current entry in the DCB parameter list
			O	Updated to the next entry in the DCB parameter list
		8	I	Address of the current entry in the Where-to-Go Table
			O	Updated to the next entry in the Where-to-Go Table
		9	W	AVT address
		10	W	Current UCB address
		11	W	DEB address
		12	I	Routine base register
		13	W	Work register
		14	W	Work register
		15	W	Work register
IGG01936	IGG01936	0	W	Work register
		1	W	Work register
		2	I	Address of the current DCB
		3	I	TIOT address
		4	I	Address of the DCB work area
		5	I	DCB parameter list address
		6	I	Where-to-Go Table address
		7	I	Address of the current entry in the DCB parameter list
			O	Updated to the next entry in the DCB parameter list
		8	I	Address of the current entry in the Where-to-Go Table
			O	Updated to the next entry in the Where-to-Go Table
		9	I	AVT address
		10	O	Total number of CCWs required for each device
		11	I	DEB address
		12	I	Routine base register
		13	W	Work register
		14	W	Work register
		15	W	Work register
IGG01937	IGG01937	0	W	Work register
		1	W	Work register
		2	I	Address of the current DCB
		3	I	TIOT Address
		4	I	Address of the DCB work area
		5	I	DCB parameter list address
		6	I	Where-to-Go Table address
		7	I	Address of the current entry in the DCB parameter list
			O	Updated to the next entry in the DCB parameter list
		8	I	Address of the current entry in the Where-to-Go Table
			O	Updated to the next entry in the Where-to-Go Table
		9	I	AVT address
		10	I	Address of the current UCB
		11	I	DEB address
		12	I	Routine base register
		13	W	Work register

		14	W	Work register
		15	W	Work register
IGG01938	IGG01938	0	W	Work register
		1	W	Work register
		2	I	Address of the current DCB
		3	W	Work register
		4	I	Address of the DCB work area
		5	I	DCB parameter list address
		6	I	Where-to-Go Table address
		7	I	Address of the current entry in the DCB parameter list
			O	Updated to the next current entry in the DCB parameter list
		8	I	Address of the current entry in the Where-to-Go Table
			O	Updated for the next entry in the Where-to-Go Table
		9	I	AVT address
		10	W	Work register
		11	W	Work register
		12	I	Routine base register
		13	W	Work register
		14	W	Work register
		15	W	Work register
IGG01939	IGG01939	0	W	Work register
		1	W	Work register
		2	I	Address of the current DCB
		3	W	Work register
		4	I	Address of the DCB work area
		5	I	DCB parameter list address
		6	I	Where-to-Go Table address
		7	I	Address of the current entry in the DCB parameter list
			O	Updated to the next entry in the DCB parameter list
		8	I	Address of the current entry in the Where-to-Go Table
			O	Updated to the next entry in the Where-to-Go Table
		9	I	AVT address
		10	W	Work register
		11	W	Work register
		12	I	Routine base register
		13	W	Work register
		14	W	Work register
		15	W	Work register
IGG01940	IGG01940	0	W	Work register
		1	W	Work register
		2	I	Address of the current DCB
		3	W	Work register
		4	I	Address of the DCB work area
		5	I	DCB parameter list address
		6	I	Where-to-Go Table address
		7	I	Address of the current entry in the DCB parameter list
			O	Updated to the next entry in the DCB parameter list
		8	I	Address of the current entry in the Where-to-Go Table
			O	Updated to the next entry in the Where-to-Go Table
		9	I	AVT address
		10	W	Work register
		11	W	Work register
		12	I	Routine base register
		13	W	Work register
		14	W	Work register
		15	W	Work register

IGG01941	IGG01941	2	O	Checkpoint work area address
		5	I	Address of the first entry in the DCB parameter list
		6	I	Address of the Where-to-Go Table
		7	I	Address of the current entry in the DCB parameter list
			O	Updated to the next entry in the DCB parameter list
		8	I	Address of the current entry in the Where-To-Go Table
			O	Updated to the next entry in the Where-To-Go Table
		12	I	Base register
IGG01942	IGG01942	2	I	Checkpoint work area address
		5	I	Address of the first entry in the DCB parameter list
		6	I	Address of the Where-to-Go Table
		7	I	Address of the current entry in the DCB parameter list
			O	Updated to the next entry in the DCB parameter list
		8	I	Address of the current entry in the Where-to-Go Table
			O	Updated to the next entry in the Where-to-Go Table
		12	I	Base register
IGG01943	IGG01943	2	I	Checkpoint work area address
		5	I	Address of the first entry in the DCB parameter list
		6	I	Address of the Where-to-Go Table
		7	I	Address of the current entry in the DCB parameter list
			O	Updated to the next entry in the DCB parameter list
		8	I	Address of the current entry in the Where-to-Go Table
			O	Updated to the next entry in the Where-to-Go Table
		12	I	Base register
IGG01944	IGG01944	2	I	Checkpoint work area address
		5	I	Address of the first entry in the DCB parameter list
		6	I	Address of the Where-to-Go Table
		7	I	Address of the current entry in the DCB parameter list
			O	Updated to the next entry in the DCB parameter list
		8	I	Address of the current entry in the Where-to-Go Table
			O	Updated to the next entry in the Where-to-Go Table
		12	I	Base register
IGG01945	IGG01945	2	I	Checkpoint work area address
		5	I	Address of the first entry in the DCB parameter list
		6	I	Address of the Where-to-Go Table
		7	I	Address of the current entry in the DCB parameter list
			O	Updated to the next entry in the DCB parameter list
		8	I	Address of the current entry in the Where-to-Go Table
			O	Updated to the next entry in the Where-to-Go Table
		12	I	Base register

IGG01946	IGG01946	5	I	Address of the first entry in the DCB parameter list
		6	I	Address of the Where-to-Go Table
		7	I	Address of the current entry in the DCB parameter list
			O	Updated to the next entry in the DCB parameter list
		8	I	Address of the current entry in the Where-to-Go Table
			O	Updated to the next entry in the Where-to-Go Table
IGG01947	IGG01947	5	I	Address of the first entry in the DCB parameter list
		6	I	Address of the Where-to-Go Table
		7	I	Address of the current entry in the DCB parameter list
			O	Updated to the next entry in the DCB parameter list
		8	I	Address of the current entry in the Where-to-Go Table
			O	Updated to the next entry in the Where-to-Go Table
IGG01948	IGG01948	0	W	Work register
		1	W	Work register
		2	I	Address of the current DCB
		3	W	Work register
		4	I	Address of the DCB work area
		5	I	Address of the first entry in the DCB parameter list
		6	I	Address of the Where-to-Go Table
		7	I	Address of the current entry in the DCB parameter list
			O	Updated to the next entry in the DCB parameter list
		8	I	Address of the current entry in the Where-to-Go list
			O	Updated to the next entry in the Where-to-Go Table
		9	I	AVT address
		10	W	Work register
		11	W	Work register
		12	I	Routine base register
		13	W	Work register
		14	W	Work register
		15	W	Work register
IGG01949	IGG01949	2	I	Checkpoint work area address
		5	I	Address of the first entry in the DCB parameter list
		6	I	Where-to-Go Table address
		7	I	Address of the current entry in the DCB parameter list
			O	Updated to the next entry in the DCB parameter list
		8	I	Address of the current entry in the Where-to-Go Table
			O	Updated to the next entry in the Where-to-Go Table
IGG02030	IGG02030	0	W	Work register
		1	W	Work register
		2	I	Current DCB address
		3	I	TIOT address
		4	I	DCB work area address
		5	I	Address of the first entry in the DCB parameter list
		6	I	Address of the Where-to-Go Table

		7	I	Address of the current entry in the DCB parameter list
			O	Updated to the next entry in the DCB parameter list
		8	I	Address of the current entry in the Where-to-Go Table
			O	Updated to the next entry in the Where-to-Go Table
		9	I	AVT address
		10	I	Current UCB address
		11	I	DEB address
		12	I	Routine base register
		13	W	Work register
		14	W	Work register
		15	W	Work register
IGG02035	IGG02035	0	W	Work register
		1	W	Work register
		2	I	DCB base register
		3	I	TCB base register
		4	I	Termname Table entry DSECT base
		5	I	Address of the first entry in the DCB parameter list
		6	I	Address of the Where-to-Go Table
		7	I	Address of the current entry in the DCB parameter list
			O	Updated to the next entry in the DCB parameter list
		8	I	Address of the current entry in the Where-to-Go Table
			O	Updated to the next entry in the Where-to-Go Table
		9	I	AVT base address
		10	I	Termname table base address
		11	I	DEB base address
		12	I	Routine base address
		13	I	PCB DSECT base register
		14	W	Work register
		15	W	Work register
IGG02036	IGG02036	0	W	Work register
		1	W	Work register
		2	I	Current DCB address
		3	I	TIOT address
		4	I	DCB work area address
		5	I	Address of the first entry in the DCB parameter list
		6	I	Address of the Where-to-Go Table
		7	I	Address of the current entry in the DCB parameter list
			O	Updated to the next entry in the DCB parameter list
		8	I	Address of the current entry in the Where-to-Go Table
			O	Updated to the next entry in the Where-to-Go Table
		9	I	AVT address
		10	I	Current UCB address
		11	I	Routine base address
		12	W	Work register
		13	W	Work register
		14	W	Work register
		15	W	Work register
IGG02041	IGG02041	5	I	Address of the first entry in the DCB parameter list
		2	I	Checkpoint work area address
		6	I	Address of the Where-to-Go Table
		7	I	Address of the current entry in the DCB parameter list

			O	Updated to the next entry in the DCB parameter list
		8	I	Address of the current entry in the Where-to-Go Table
			O	Updated to the next entry in the Where-to-Go Table
		12	I	Base register
IGG02046	IGG02046	5	I	Address of the first entry in the DCB parameter list
		6	I	Address of the Where-to-Go Table
		7	I	Address of the current entry in the DCB parameter list
			O	Updated to the next entry in the DCB parameter list
		8	I	Address of the current entry in the Where-to-Go Table
			O	Updated to the next entry in the Where-to-Go Table
IGG02047	IGG02047	3	W	DCB address
		4	W	DEB address
		5	I,O	Address of the first entry in the DCB parameter list
			W	TCB address
		6	I,O	Address of the system Where-to-Go Table
			W	TCB address
		7	I	Address of the current entry in the DCB parameter list
			O	Address of the next entry in the DCB parameter list
		8	I	Address of the current entry in the system Where-to-Go Table
			O	Address of the next entry in the system Where-to-Go Table
		9	W	SCB address
		10	W	AVT address
		12	I	Base register
		13	W	Open/Close work area address

APPENDIX A: LIST OF TCAM MODULES BY LIBRARY

This appendix identifies the modules that comprise TCAM. The modules are organized by the libraries in which the modules reside. The modules in each library are in alphabetical order by name. For those modules that represent macro instruction implementing routines, the mnemonic operation code for the macro is included in parentheses.

All resident TCAM modules are in SYS1.TELCMLIB. Transient modules reside in SYS1.LINKLIB, and all Open, Close, Get, and Put modules are in SYS1.SVCLIB. The system nucleus modules are in SYS1.NUCLEUS. The TCAM module IEDQTNT is not stored in a library; rather, it is assembled as part of the Termname Table. TCAM macros are in SYS1.MACLIB.

SYS1.LINKLIB

IEDOCF	Modify Options Routine
IEDOCG	Copy Line Information Routine
IEDOCH	Copy Terminal Information Routine
IEDOCI	Copy LCB Information Routine
IEDOCJ	Copy QCB Information Routine
IEDOCK	Copy Held Terminals Routine
IEDOCL	Copy Invitation List Entry Routine
IEDOCM	Copy Operator Control Terminal Routine
IEDOCN	Change Control Terminal Routine
IEDOCO	Change Terminal Routine
IEDOCP	Alter Trace Status Routine
IEDOCQ	Stop/Resume Terminal Transmission Routine
IEDOCU	Start Line Routine
IEDOCV	Stop Line Routine
IEDOCW	Modify Poll Routine
IEDOCX	Modify Intense Routine
IEDOCZ	Change Interval Type Routine
IEDOC0	MCP Closedown Processing Routine

IEDQC1	ICHNG Processing Routine
IEDQC2	On-Line Test Interface Routine
IEDQC3	Copy Invitation List Status Routine
IEDQC5	Nonexecutable Work Area for Operator Control
IEDQC6	Debug Service Aid Router
IEDQE6	Password Scramble Routine
IEDQHI	System Delay Routine
IEDQNA2	Nonresident Closedown Completion Routine
IEDQNB	Application Program/Checkpoint Interface Routine
IEDQND	Ready Routine (READY)
IEDQNF	Checkpoint Executor
IEDQNG	Build Incident Record for MH Routine
IEDQNH	Build Incident Record for ICHNG Routine
IEDQNJ	Incident Checkpoint for Operator Control Routine
IEDQNK	Environment Checkpoint Routine
IEDQNM	Build CKREQ Disk Record Routine
IEDQNO	Checkpoint Queue Manager
IEDQNP	Checkpoint Disk I/O Routine
IEDQNQ	Checkpoint Notification and Disposition Routine
IEDQNR	Checkpoint - No Available Core Routine
IEDQNS	Checkpoint - No Incident Records Routine
IEDQNX	Operator Awareness Message Router
IEDQOA	Link Routine
IEDQOB	WTOR Interpreter Routine
IEDQOG	INTRO GETMAIN Routine
IEDQOM	Termname Table Sort Routine
IEDQOS	Attach Routine
IEDQXA	Disk Message Queue Initializer

SYS1.MACLIB

ATTEN	Activates the TSO/TCAM attention processing routine
CANCELMG	Cancels messages
CARRIAGE	Processes characters that move the carriage
CHECKPT	Takes an Incident Checkpoint record of the option fields
CHNGP	Modifies an invitation list
CHNGT	Places specified data in a Terminal Table entry
CKREQ	Checkpoints the MCP
CLOSEMC	Closes down the telecommunications system
CODE	Translates the data in the buffer currently being handled
COPYP	Examines the contents of an invitation list
COPYQ	Examines the contents of a QCB
COPYT	Examines the contents of a Terminal Table entry
COUNTER	Maintains a count of complete messages or of message segments received from or sent to a terminal
CUTOFF	Specifies the maximum allowable incoming message length
DATETIME	Inserts the date and time in an incoming or outgoing message header
ERRORMSG	Sends an error message when an error occurs
FORWARD	Queues messages for specified destinations
HANGUP	Checks for I/O errors
HOLD	Suspends transmission to a terminal
ICHNG	Modifies an invitation list
ICOPY	Examines the contents of an invitation list
IEDQCHAR	Internal assembly macro to check character strings
IEDQCHT	Internal assembly macro to determine device characteristics

IEDOCKO Internal assembly macro to perform validity checking on terminal operands

IEDOFEA Internal assembly macro for the FE serviceability modules

IEDOGCH Internal assembly macro to generate device dependent fields for a terminal entry

IEDQMASK Internal assembly macro to analyze mask operands

IEDOSCAN Internal assembly macro to search for a character string

IEDOTO Internal assembly macro to generate the option fields specified by a TERMINAL macro

IEDQTO Internal assembly macro to generate QCBs

IEDOTT Internal assembly macro to generate a Termname Table entry

IEDQVCON Internal assembly macro to provide proper branching addresses for all the macros

INBUF Identifies a subgroup that handles incoming message buffers

INEND Identifies the end of the MH incoming group

INHDR Identifies the beginning of an inheader subgroup

INITIATE Sends message segments immediately to their destination

INMSG Identifies the beginning of an MH inmessage subgroup

INTRO Creates the AVT

INVLIST Generates the invitation list for a line

INVLIST1 Internal assembly macro to generate an invitation list

INVLIST2 Internal assembly macro to generate an invitation list

INVLIST3 Internal assembly macro to generate an invitation list

LINEGRP TSO MCP generation macro

LISTTA TSO MCP generation macro

LOCK Locks one terminal on a line to an application program

LOCOPT	Locates a field in the Option Table
LOG	Logs complete messages or message segments
LOGON	Performs logon procedures
LOGTYPE	Initializes for using the TCAM logging facility
MCPCLOSE	Initiates closedown of the telecommunications system
MRELEASE	Releases messages queued for a destination
MSGEDIT	Inserts specified characters into specific locations in a message
MSGFORM	Inserts EOT line control characters in outgoing messages
MSGGEN	Generates an unqueued message
MSGLIMIT	Limits the number of messages during a single transmission sequence
MSGTYPE	Controls the path of a header through an MH
OPTION	Defines the Option Table
ORIGIN	Checks the validity of the origin field in a message header
OUTBUF	Identifies a subgroup that handles outgoing message buffers
OUTEND	Identifies the end of any MH outgoing group
OUTHDR	Identifies the beginning of an outheader subgroup
OUTMSG	Identifies the beginning of an MH outmessage subgroup
PATH	Dynamically varies the path of a message through an MH
PCB	Generates a Process Control Block in an MCP to interface with an application program
PRIORITY	Specifies priority handling for messages
PROCESS	Interfaces between the MCP and an application program
QCOPY	Examines the contents of a QCB
QSTART	Differentiates between a QTAM and a TCAM application program
READY	Initializes and activates the MCP

REDIRECT Queues a message for an additional destination

RELEASEM Releases messages queued for a destination

RETRIEVE Retrieves a message for reprocessing

RTAUTOPT Resumes automatic prompting (after a null line).

SCREEN Modifies the Write operations for display terminals

SEQUENCE Checks the input sequence number of an incoming message

SETEOF Indicates an EOF message

SETSCAN Proves the scan pointer forward or backward or returns the address of the last character of a specific character string

SGIEC3TP Moves BTAM, QTAM, and TCAM modules into SYS1.SVCLIB at system generation time

SGIEC5TP Moves BTAM, QTAM, and TCAM modules into SYS1.LINKLIB, SYS1.SVCLIB, and SYS1.TELCMLIB at system generation time

SGIEC2PT Generates UCBs at system generation time

SGIEC519 Moves the proper macros into SYS1.MACLIB at system generation time

SIMATTN Handles a simulated attention string or code

SPAUTOPT Stops automatic prompting

STARTLN Activates a line or line group

STARTMH Establishes addressability for an MH routine

STATTN Sets up a simulated attention string or code, time or lines

STAUTOCP Starts automatic character prompting

STAUTOLN Starts automatic line numbering

STBREAK Allows the user to specify the presence of the reverse break feature

STCC Allows the user to specify line and character deletion characters

STCLEAR Specifies the character string used to clear the 2260 screen

STCOM	Specifies whether to allow other TSO stations to send the user messages
STOPLN	Deactivates a line or line group
STSIZE	Specifies the length of a line or the length of and the number of lines for a 2260
STTIMEOU	Specifies whether a 1050 has the timeout suppression feature
TCHNG	Places specified data in a Terminal Table entry
TCLEARQ	Allows the user to clear the TSO input or output queue.
TCOPY	Examines the contents of a Terminal Table entry
TERMINAL	Creates a single or group entry in the Terminal Table
TERRSET	Sets a bit in the Error Record
TGET	Transfers a line of input from a TSO terminal to the user's data area.
TLIST	Defines a cascade-list or distribution-list entry in the Terminal Table
TPROCESS	Interfaces between the MCP and an application program
TPUT	Transfers a line of output from the user's data area to a TSO terminal
TRANLIST	Generates a control table for use by the Dynamic Translation routine (IEDQA3)
TSINPUT	Generates a QCB for the TSO subtask and creates an extension of the AVT for TSO support
TSOMCP	TSO MCP generation macro
TSOMH	TSO MCP generation macro
TTABLE	Defines the Terminal Table
UNLOCK	Removes a terminal from extended lock mode

SYS1.NUCLEUS

IEDQATTN	Attention Routine
IGC102	AOCTL SVC 102 Routine

SYS1.SVCLIB

IGC1303D	TCAM Command Scheduler - SVC 34
IGC0010D	Operator Control Control Module - Load 0
IGC0110D	Operator Control Control Module - Load 1
IGC0210D	Operator Control Control Module - Load 2
IGC0310D	Operator Control Control Module - Load 3
IGC0410D	Operator Control Control Module - Load 4
IGC0510D	Operator Control Control Module - Load 5
IGE0004G	Start/Stop ERP Control Module
IGE0104G	Read/Write Unit Check and Unit Exception ERP Module
IGE0204G	Non-operational Control Unit Module
IGE0304G	Unit Check for Non-read, Non-write, and Non-poll CCWs ERP Module
IGE0404G	Auto Poll and Read Response to Poll Unit Check and Unit Exception ERP Module
IGE0504G	Error Post and Second Level CCW Return Module
IGE0604G	Unit Check and Unit Exception on Read/Write CCWs for Audio and 2260 Local Devices ERP Module
IGE0804G	Start/Stop Channel Check Module
IGE0904G	Closedown Terminal Statistics Recording Module
IGE0004H	BSC ERP Control Module
IGE0104H	BSC Read/Write Equipment Check, Lost Data, Intervention Required, and Unit Exception ERP Module
IGE0204H	BSC Read/Write Data Check, Overrun, and Command Reject ERP Module
IGE0404H	BSC Second Level CCW Return Module
IGE0504H	BSC Error Post Module
IGE0804H	BSC Channel Check ERP Module
IGG01900	Line I/O Interrupt Trace Routine

IGG019Q1	Local Receive Scheduler
IGG019Q2	Line End Appendage for BSC Lines
IGG019Q3	Line End Appendage for Start/Stop Lines
IGG019Q4	Line End Appendage for Leased and Start/Stop Lines and No TSO
IGG019Q5	Line End Appendage for a QTAM Compatible System
IGG019Q6	Send Scheduler for Leased Lines and No TSO
IGG019Q7	Send Scheduler with No TSO
IGG019Q8	Checkpoint Continuation Restart Subroutine
IGG019RA	Checkpoint Disk End Appendage
IGG019RB	TCAM Dispatcher
IGG019RC	EXCP Driver
IGG019RD	Buffered Terminal Scheduler
IGG019RF	EXCP Drive for a Single CPB
IGG019RG	GET/READ Routine
IGG019RH	Get Compatible Routine
IGG019RI	PUT/WRITE Routine
IGG019RJ	Put Compatible Routine
IGG019RK	Disk End Appendage for a Single CPB
IGG019RL	Check Routine (CHECK)
IGG019RM	Point Routine (POINT)
IGG019RN	PCI Appendage
IGG019RO	TCAM Dispatcher with Subtask Trace
IGG019RP	Reusability-Copy Subtask
IGG019RQ	Post Pending Routine
IGG019RR	IBM 1030, 1050, 1060, 2740, 2741 Special Characters Table
IGG019RS	IBM 2260 Remote Special Characters Table

IGG019RT	AT&T 115A or Western Union 83B3 Special Characters Table
IGG019RU	AT&T TWX, with Odd Parity Special Characters Table
IGG019RV	IBM 2260 Local Special Characters Table
IGG019RW	World Trade Teletype Adapter (WTTA) Special Characters Table
IGG019RX	AT&T TWX, with Even Parity Special Characters Table
IGG019RY	Audio Special Characters Table
IGG019R0	Line End Appendage
IGG019R1	Dial Receive Scheduler
IGG019R2	Disk End Appendage
IGG019R3	Leased Receive Scheduler
IGG019R4	Send Scheduler
IGG019R5	Attention Handler
IGG019R6	Startup Message Routine
IGG019R7	BSC EBCDIC Code Special Characters Table
IGG019R8	BSC USASCII Code Special Characters Table
IGG019R9	BSC 6-bit Code Special Characters Table
IGG01930	Disk Message Queues Open - 1
IGG01931	Disk Message Queues Open - 2
IGG01933	Open Error Handler
IGG01934	Disk Message Queues Open - 3
IGG01935	Line Group Open - 1
IGG01936	Line Group Open - 2
IGG01937	Line Group Open - 3
IGG01938	Line Group Open - 4
IGG01939	Line Group Open - 5
IGG01940	Line Group Open - 6
IGG01941	Checkpoint Open Routine

IGG01942	Checkpoint Disk Initialization Routine
IGG01943	Checkpoint/Restart from Environment Record Routine
IGG01944	Checkpoint/Restart from Incident and CKREQ Records Routine
IGG01945	Checkpoint Continuation Restart Routine
IGG01946	GET/PUT and READ/WRITE Cpen Executor - 1
IGG01947	GET/PUT and READ/WRITE Open Executor - 2
IGG01948	Line Group Open - 7
IGG02030	Disk Message Queues Close Routine
IGG02035	Line Group Close Routine - 1
IGG02036	Line Group Close Routine - 2
IGG02041	Checkpoint Close Routine
IGG02046	GET/PUT and READ/WRITE Close Executor - 1
IGG02047	GET/PUT and READ/WRITE Close Executor - 2

SYS1.TELCMLIB

IEDAYA	TSC Attention Routine
IEDAYC	TSO Carriage Subroutine
IEDAYD	Time Sharing Destination Scheduler
IEDAYE	TSO TICC Edit Routine
IEDAYF	TSO IOHALT Routine
IEDQYH	TSO Hangup Routine
IEDAYI	TSINPUT Routine
IEDAYL	TSO Logon Routine
IEDAYM	TSO Message Generation Routine
IEDAYO	TSOUTPUT Routine
IEDAYR	STARTMH Subtask for TCAM-TSO Mixed
IEDAYS	TSO Simulated Attention Routine
IEDAYT	TSO Abend Interface Routine

IEDAYX	TSO INMSG/OUTMSG Linker Routine
IEDAYY	TSO Asynchronous Time Delay Removal Routine
IEDAYZ	Time Sharing Scheduler
IEDQAA	STARTMH Subtask (STARTMH)
IEDQAC	Date and Time Provision Routine (DATETIME)
IEDQAD	Output Sequence Number Provision Routine
IEDQAE	Locate Option Field Address Routine (LOCOPT)
IEDQAF	Insert Data Routine
IEDQAG	Message Limit Routine
IEDQAH	Input Sequence Number Insertion Routine
IEDQAI	Skip Forward and Scan Routine
IEDQAJ	Skip to Character Set Routine
IEDQAK	Line Control Insertion Routine
IEDQAL	Address Finder Routine
IEDQAM	Origin Routine
IEDQAN	Multiple Insert/Remove Routine
IEDQAO	Unit Request Interface Routine
IEDQAP	Remove at Offset Routine
IEDQAQ	Operator Control Interface Routine
IEDQAR	Cancel Message Routine
IEDQAS	Hold/Release Terminal Routine
IEDQAT	Create an Error Message Routine
IEDQAU	Cutoff Message Transmission Routine
IEDQAV	Lookup Terminal Entry Routine
IEDQAW	Translate Buffer Routine
IEDQAX	Buffer Step Routine
IEDQAY	Screen Routine
IEDQAZ	Redirect a Message Routine

IEDQA0	Skip Backward Routine
IEDQA1	Binary Search Routine
IEDQA2	Insert at Offset Routine
IEDQA3	Dynamic Translation Routine
IEDQA4	Incominq/Outqoinq Messaqe Delimiter Routine
IEDQA5	Forward Routine
IEDQA6	Line Control Initialization Routine
IEDQA7	Counter Routine
IEDQA8	Multiple Insert at Offset Routine
IEDQBA	Multiple Routinq Subtask
IEDQBB	Checkpoint Request Routine
IEDQBC	Distribution List Subtask
IEDQBD	Buffer Disposition Subtask
IEDQBE	Lock Routine
IEDQBF	Unlock Routine
IEDQBG	Cascade List Subtask
IEDQBL	Messaqe Generation Routine (MSGGEN)
IEDQBT	EOB/ETB Handlinq Subtask
IEDQBW	Unit Request Routine
IEDQBX	Loq Seqment Routine
IEDQBY	Loq Messaqe Routine
IEDQBZ	Loq Scheduler
IEDQCA	Resident Operator Control Module
IEDQEC	Put Scheduler
IEDQES	Retrieve Service Routine
IEDQET	Operator Control/Application Proqram Interface Routine
IEDQEU	Open/Close Subtask
IEDQEW	Get Scheduler

IEDQEZ	Get Scheduler FIFO Routine
IEDQE1	TCOPY Service Routine (TCOPY)
IEDQE2	QCOPY Service Routine (QCOPY)
IEDQE3	TCHNG Service Routine (TCHNG)
IEDQE4	ICOPY Service Routine (ICOPY)
IEDQE7	Retrieve Scheduler
IEDQFA	CPB Initialization Module
IEDQFA1	CPB Initialization-Main Storage Queuing Only
IEDQFA2	CPB Initialization-Disk Queuing Only
IEDQGA	Buffer Management Module
IEDQGT	Transparent Transmission CCW Building Routine
IEDQHG	Time Delay Subtask
IEDQHK	Stop Line I/O Subtask
IEDQHM	Destination Scheduler
IEDQHM1	Destination Scheduler-Main Storage Queuing Only
IEDQHM2	Destination Scheduler-Disk Queuing Only
IEDQKA	Activate-I/O Generator Subtask
IEDQKB	Activate-I/O Generator Subtask for BSC Lines
IEDQKC	Activate-I/O Generator Subtask for Start/Stop Lines
IEDQKD	Activate-I/O Generator Subtask for Leased and Start/Stop Lines and No TSO
IEDQKE	Activate-I/O Generator Subtask for a QTAM Compatible System
IEDQLM	Return Interface Routine
IEDQNA	Resident Closedown Completion Routine
IEDQUI	User Interface Routine
IEDQ10	IBM 1030 Translate Table
IEDQ11	IBM 1050 Translate Table
IEDQ12	IBM 1050 Folded Translate Table

IEDQ13	IBM 1060 Translate Table
IEDQ14	IBM 2260 Translate Table
IEDQ15	Alias for IEDQ14
IEDQ16	IBM 2740 Translate Table
IEDQ17	IBM 2740 Folded Translate Table
IEDQ18	World Trade Teletype Adapter (WTTA), ITA2 Translate Table
IEDQ19	World Trade Teletype Adapter (WTTA), ZSC3 Translate Table
IEDQ20	AT&T 115A or Western Union 83B3 Translate Table
IEDQ21	AT&T TWX, with Parity Translate Table
IEDQ22	AT&T TWX, without Parity Translate Table
IEDQ23	IBM 2780, 6-bit Code Translate Table
IEDQ24	USASCII Code Translate Table
IEDQ25	Dummy Table (EBCDIC to EBCDIC)
IEDQ26	IBM 2741, BCD Code Translate Table
IEDQ27	IBM 2741, EBCD Code Translate Table
IEDQ28	IBM 2741, Correspondence Code Translate Table

TCAM QUEUES

Checkpoint Disk I/O queue - Checkpoint disk records wait on this queue to be written to disk. The records are queued in FIFO order. The first word of the record is the link field, Each time an environment checkpoint record is put on the checkpcint disk I/O queue, the IEDQNO routine scans the queue. If there are any incident checkpoint disk records on the queue, the IEDQNO routine removes them and frees them up. Since the information in the incident checkpoint record is included in each environment record, it is not necessary to write bcth records to disk. The Checkpoint Executor routine (IEDQNF) looks at the queue when a record is put on the queue, and gives control to the Checkpoint Disk I/O routine (IEDQNP).

Communication queue - This is a queue of command input blocks in FIFO order, chained by the first word in each CIB. The communication queue is used to queue command input blocks containing operator control commands from the console. An SVC 34 from the Command Scheduler places the CIBs on the queue, and the SVC 34 routine removes them. The second wcrd of the queue is the communication ECB.

Copy Buffer queue - When a message is to be copied from one queue medium to another, the first buffer of the message is tposted to COPY, which hangs the buffer on the copy buffer queue, pointed to by the AVTCOPY field. This field also points to the Copy QCB whose first two words are used as a FIFC queue of buffers. Each message stays on the Copy Buffer queue until a CPB is available to be used tc copy the message. One CPB is used per message. The use of this queue ensures that messages will be copied in the order that the copy operation was requested as CPBs become available. Buffers are chained by their second word, There is a zero in the second word of the last buffer.

CPB Free Pool queue - The AVTFCPB field contains the address of the first of a chain of CPBs that are not busy. They are chained by CPBNEXT, with a zero in the last one. This is not a FIFO queue (as is other CPB queues) but a LIFO (last in, first out) queue. If the user specifies too many CPBs (INTRO CPB=integer), the CPBs at the end of this free pool chain will never have been used. The user should look at a TCAM dump for unused CPBs and be able to specify a smaller number next time, thereby saving main storage.

Disabled Ready queue - The disabled ready queue is a FIFO queue that contains elements passed frcm application programs and attached tasks for processing by the MCP. The contents of this queue are merged into the enabled ready queue by the TCAM Dispatcher.

Disk End queue - There are two disk end queues. The address of the first is at AVTDKAPQ. This queue is used to pass CPBs from the Disk End Appendage to the CPB Cleanup rcutine. The address of the second queue is at AVTDKENQ. This queue is used as an alternate in the disabled/enabled interface to pass CPBs from the Disk End Appendage to the CPB Cleanup routine. If the AVTBPLKN bit is on, the Disk End Appendage cannot put a CPB in the disk end queue pointed to by AVTDKAPQ, but must place it in the queue pointed to by AVTDKENÇ.

Enabled Ready queue - see Ready queue.

EXCP queue - This is a chain of CPBs for the one cylinder, in one extent of a disk message queues data set, that is currently ready for I/O execution. CPBs are ordered on this chain by FIFO order. CPB Initialization waits cn this queue for the I/O to complete so it can build a new CPB and do another EXCP.

EXCP Driver Input queue - This is a chain of CPBs that the EXCP Driver processes until it is empty. Only Read or Write CCW op codes and the buffer unit address are in the channel program. The disk address is an absolute disk address in the same form as when taken from the CPBRADDR or CPBNADDR field. An indication of reusability or nonreusability is in the CPBFLAG. The EXCP Driver removes the CPBs in FIFO order, hangs each one on the New queue by cylinder, and then completes the channel program. No EXCP is issued until the input queue is emptied. A doubleword queue pointer is in the AVTINCPQ field.

FEFO queue - first-ended-first-out - A FEFO message queue is ordered so that the message that ends first will be sent out first regardless of the order in which the messages started being received.

FIFO queue - A FIFO queue is any queue of elements that is managed on a first-in-first-out basis. When an element is placed on the queue, it is placed in the order in which it was received and the first element on the queue is the first to be removed.

Hold queue - A hold queue is a FEFO-ordered queue that is a part of the priority level QCB for each Destination QCB. If a terminal is intercepted (held), its messages are placed in this queue while messages for other terminals on this Destination QCB are sent.

New queue - A queue on the IOB chain of CPBs being built by EXCP Driver. The CPBs are sorted on this queue by absolute cylinder number and are in FIFO order for any cylinder group. The CPBs are placed on the queue one at a time from the input queue by the EXCP Driver. They are removed by cylinder group and are placed on the Retry queue.

No-buffer queue - This is a FIFO-ordered queue of CPBs for read operations when no buffers are in the buffer unit pool. This is an internal queue used by IEDQFA and IEDQFQ. The elements are linked by the CPBNEXT field.

No-CPB queue - This is queue of buffers and ERBs waiting for CPBs. The queue is located in the AVT and serves as a place to keep elements until CPBs are built for them.

Operator Control queue - This is FIFO queue of buffers, dummy CPBs from application programs and TOTE, stopped LCBs, and dummy ERBs with buffers associated. The second word of the queue is the Operator Control ECB. The queue is used as a communication link between the TCAM MCP and Operator Control. All commands other than those from the console are placed on this queue, as well as elements (LCBs, ERBs) requested by Operator Control.

Ready queue - This is a priority-FIFO ordered queue of TCAM elements that are to be processed by the TCAM subtasks.

Retry queue - This is a chain of CPBs for one cylinder in an extent of the disk message queues data set. These CPBs are next in line for I/O execution after the CPBs on the EXCP queue are processed. When the Disk End Appendage receives control after the CPBs on the EXCP queue are finished, it requests IOS to do a "retry" after moving the CPBs on this queue to the EXCP queue. This last move avoids an extra EXCP and permits the channel to begin work on the new disk channel program faster.

REUS CPB queue - When the CPB Cleanup routine finds a CPB belonging to the Reusability-Copy subtask (CPBFLAG, CPBREUSN bit on), it puts the CPB on the REUS CPB queue at AVTREUSQ, a doubleword FIFO queue of CPBs being returned to the Reusability-Copy subtask. Each CPB has a 4-bit field in the low-order 4 bits of CPBFLAG identifying the type of action the CPB is to do. Since the CPB may be doing I/O for the Reusability-Copy subtask, there may be several CPBs on the queue. The CPB Cleanup routine calls the Reusability-Copy subtask if either the reusability CPB queue is not empty, the reusability 'first time' switch is set (AVTRUFTN bit in AVTBIT2), or the 'copy wants control' bit is set (AVTCOPYN in AVTBIT2). Once Reusability-Copy gets control, it continues to process CPBs from the reusability CPB queue until that queue is empty.

System Delay queue - This is a chain of LCBs. The System Delay subtask (IEDQHI) waits on the queue until all the LCBs are on the queue and then begins the system delay interval. When a system delay is requested, the Leased Receive Scheduler and the Buffered Terminal Scheduler tpost LCBs to the system delay queue, rather than continue I/O on the lines. When the count of LCBs is the same as the number of LCBs received by the System Delay subtask, a time request is posted to the Time Delay subtask (IEDQHG). After the interval is complete, each LCB is removed and tposted to itself to resume line activity.

Time Delay queue - This is a relative time of interrupt ordered chain of elements that are requesting a system STIMER interrupt. The elements are chained by the eighth word in the element. The Time Delay QCB is always the last element in the queue. The purpose of this queue is to inform the routine tposting the element when a specified time has elapsed.

TCAM QCBS

Buffer Disposition QCB - The address of the Buffer Disposition subtask (IEDQBD) is the first address in the list pointed to by the AVTMSGS field of the AVT. The Buffer Disposition QCB comprises the first three words of the routine. The Incoming/Outgoing Message Delimiter routine (IEDQA4) tposts the last segment of the incoming message to the QCB, and the Line End Appendage routine (IGG019R0) tposts the last segment of the outgoing message to the QCB to execute the INMSG and OUTMSG macro instructions. The Line End Appendage routine tposts the ICB to the QCB when the routine reaches the end of the polling list to clean up the line.

Buffer Request QCB - The Buffer Request QCB address is located in the AVTBFREB field in the AVT. The Receive Schedulers (IGG019R1 and IGG019R3) tpost to the QCB ERBs requesting buffers for receiving operations . Buffer units are chained from the first word of the QCB to form the buffer unit pool.

Buffer Return QCB - The Buffer Return QCB address is located in the AVTBFRTB field in the AVT. Routines which are no longer using buffers tpost them to the QCB to be returned to the buffer pool.

Checkpoint QCB - The Checkpoint QCB address is located in the AVTCKPTB field in the AVT. This is a special type of QCB for attached tasks, and the QCB is also the STCB. An ECB is in the second word of the QCB. The Checkpoint Executor (IEDQNF) waits on the ECB. The TCAM Dispatcher posts the ECB when it puts a request element on the chain. The Checkpoint QCB is never tposted to itself. However, when a checkpoint request element is tposted to the QCB, the Checkpoint Executor is given control.

Closedown Completion Element QCB - the QCB address is located in the AVTCLOSB field in the AVT. The MCP Closedown Processing routine (IEDQC0) and the Checkpoint Notification and Disposition routine (IEDQNQ) tpost the QCB to itself to give control to the Resident Closedown Completion routine (IEDQNA). The QCB is used as an element with the lowest priority of any element in the system. It is the only element ever tposted to the QCB.

CPB Cleanup QCB - The address of the CPB Cleanup QCB is located in the AVTCPBCB field in the AVT. The Disk End Appendage (IGG019R2), upon completion of an I/O operation, chains the completed CPBs on the AVTDKAPQ queue and tposts the QCB to itself to activate the CPB Cleanup routine (IEDQFQ) in CPB Initialization (IEDQFA).

Cutoff QCB - The Cutoff QCB is located within the Cutoff routine (IEDQAU). The Cutoff routine places the address of the QCB in the first word of the LCB. Line End Appendage (IGG019R0) tposts the LCB being cutoff to the QCB when a channel program check occurs or when the read skip or write break sequence initiated by the Cutoff routine completes.

Delete from Time Delay QCB - The address of the Delete from Time Delay QCB is in the AVTCPRMB field of the AVT. Attached tasks tpost a special four-word element to this QCB. The element defines another element and requests the Time Delay subtask (entry point IEDQHG03) to search the time delay queue for a particular element. If the Time Delay subtask finds the element on the time delay queue, it removes that element. After this process, the subtask tposts the four-word element back to the requestor to indicate the completion of the request.

Destination QCB - A pointer to a specific Destination QCB is in each Terminal entry. This pointer does not change, but, as messages are received or sent, the SCB points to the Destination QCB involved. For dial or buffered terminals, the Time Delay subtask (IEDQHG) tposts the QCB to itself at the end of a time delay. Routines tpost full buffers to be queued to the Destination QCB. The Destination Scheduler (IEDQHM) is always the last subtask represented on the STCB chain of a Destination QCB. A Destination QCB is made up of a master QCB, which contains the Send Scheduler STCB for this QCB and other information pertinent to the entire QCB; and one or more priority level QCBs, which contain all the queuing pointers for messages for the particular priority level.

Disk I/O QCB - The Disk I/O QCB address is located in the AVTDSIOB field in the AVT. Buffers requesting a write on disk or the servicing of a bit are tposted to the Disk I/O QCB for processing by CPB Initialization. The schedulers tpost to this QCB ERBs requesting full buffers to send.

Log Destination QCB - There is a pointer to a Log Destination QCB in every logtype Terminal Table entry. When a log message is specified, a LOGTYPE macro must be specified in the Terminal Table to generate a terminal entry, an LCB, and an SCB. The Log Message routine (IEDQBY) tposts a duplicate header to the Log Destination QCB after the complete message is received or sent.

Master QCB - The basic format of a Destination QCB. This QCB contains ten words of destination-specific data.

Multiple Routing QCB - The Multiple Routing QCB is in the list of VCON pointed to by the AVTMSGS field in the AVT. The FORWARD parameter list has the index to it. Elements chained on the QCB are either IEDQFA recalled buffers or the IEDQFA ERB for the line.

On-Line Test QCB - The address of the On-Line Test QCB is in the AVTOLTQB field of the AVT. Test request messages (messages requesting TOTE to run an on-line test through TCAM) are tposted to this QCB.

Operator Control QCB - The address of the Operator Control QCB is located in the AVTOPCOB field in the AVT. This is a special QCB for attached tasks, and the second word of the QCB is an ECB. When the Dispatcher receives an element for this QCB at the top of the ready queue, the ECB is posted complete. The Translation Test routine (IEDQA3) tposts buffers containing operator commands to the QCB. The Application Program/Operator Control Interface routine (IEDQNB) tposts dummy CIBs from application programs to the QCB. The Buffer Management module-Buffer Request routine (IEDQGA) tposts dummy ERBs containing requested buffers to the QCB. The Stop Line I/O subtask (IEDQHK) tposts stopped LCBs to the Operator Control QCB.

PCB QCB - The PCB QCB is located in words 2 through 4 of the PCB. This QCB is used in support of QTAM compatible RETRIEVE. The Dispatcher dispatches the Retrieve Scheduler (IEDQE7) from this QCB. The element chain contains retrieved buffers.

Priority QCB - Priority QCBs follow the Master QCB and are logically a part of the Master Destination QCB. IEDQHM queues messages on one of the Priority QCBs that is associated with the Master Destination QCB to which the message was tposted. The Send Scheduler (IGG019R4) sends messages queued on the highest Priority QCB first.

Put Process QCB - The address of the Put Process QCB is in a process entry in the Terminal Table. This QCB provides compatability and symmetry so that all terminal entries will look alike to TCAM modules.

QCB for IEDQBD02 - The QCB is located within the IEDQBD02 Buffer Disposition subtask (IEDQBD02 entry point). The subtask (IEDQBD) tposts the LCB to this QCB when an INMSG/OUTMSG subgroup has been executed.

Read-ahead QCB - The address of the Read-ahead QCB is in the DEBQCBAD field of the application program data extent block, the location of which is within the process entry work area PERAQCB. The element chain contains buffers processed by the application program output message handler, but not processed by the GET/READ logic. The Dispatcher uses this QCB to dispatch the Get Scheduler (IEDQEW).

Recall QCB - The address of this QCB is in the LCBRCQCB field of the LCB. This is a pointer to the QCB of the subtask wishing control to be passed to it with a recalled buffer. The ERB is tposted to the QCB indicated in LCBRCQCB.

STARTMH QCB - The address of the STARTMH QCB is in the DCBMH field of the DCB for the line group. Buffers are tposted to this QCB by Line End Appendage and PCI Appendage on input when they are filled. On output, the buffers are tposted to the QCB by Line End Appendage after a positive response to addressing. When buffers are tposted to the QCB, IEDQAA receives control unless EOB checking is requested, in which case IEDQBT receives control.

QCB for the Stop Line I/O subtask - The address of this QCB is in the AVTHK field in the AVT. The Stop Line routine (IEDQCK) tposts stop line reouests to this QCB. The various schedulers tpost LCBs to it.

System Delay QCB - The System Delay QCB is located in the first three words of the System Delay subtask (IEDQHI). The address of the subtask is in the AVTHI field of the AVT. The System Delay subtask tposts the QCB to the Time Delay subtask (IEDQHG) to start a wait. At the end of the wait, the Time Delay subtask tposts the QCB to itself to activate the System Delay subtask.

Time Delay QCB - The Time Delay QCB is the last element on the time delay queue. The AQCTL SVC 102 routine (IGC102) tposts the QCB to itself as a result of the STIMER exit routine. This QCB is used by the STIMER exit routine to activate the Time Delay subtask (IEDQHG).

TSINPUT QCB - The address of this QCB is in the AVTTSOPT field of the AVT. The QCB is tposted to the TSINPUT routine (IEDAYI) to remove the system WAIT and unlock the keyboard.

APPENDIX C. LIST OF RELATIVE PRORITIES IN TCAM

TCAM routines apply relative priorities to elements through the use of the TPRIOR macro. The names and values presented in this table are established by this internal macro.

Name	Value	Use in an ERB	Routines Using
PRIINTRQ	E4	to request full buffers from Disk I/O	Send Scheduler Receive Scheduler Get Scheduler Put Scheduler Create an Error Message Routine
PRIFSPCI	E8	to request empty buffers from Buffer Request QCB; to request full buffers from Disk I/O	PCI Appendage-on first PCI only Multiple Routing Subtask
PRISBPCI	E0	to request empty buffers from Buffer Request QCB; to request full buffers from Disk I/O	PCI Appendage-all PCIs except the first
PRIDSKRQ	EC	to request an empty unit by chaining the ERB on the Buffer Return QCB	CPB Cleanup
PRIACTIV	E4	in tposting ERB to the Activate QCB to request building of an initial contact program and EXCP for the line	CPB Cleanup Buffer Request Buffer Return
PRIDKEOB	E0	to enable EOB to recall; to tpost to EOB Handling after an EOB error; must be lower priority than PRIMHBFR	CPB Cleanup CPB Initialization
PRIRECAL	E0	to request from Disk I/O a copy of the header	All routines requesting recalled headers Multiple Routing Subtask
PRIRCQCB	E0	to return the ERB to any routine specified in LCBRCQCB	CPB Cleanup - after recall Create an Error Message Routine

PRIAPERB	D0	to request full buffers	Application Program
PRIEDISP	E0	to tpost ERB to itself on send operations when an error occurs before EOM; must be lower than PRIMHBFR	Buffer Disposition
PRIMHBFR	E4	to have a buffer processed by MH	PCI Appendage CPB Cleanup Line End Appendage-receive, last buffer only
PRIUREQ	E8	to request an empty unit for insert function in MH; must be higher than PRIMHBFR	Unit Request
PRIAPBFR	DC	to tpost a buffer to an application program	Incoming/Outgoing Message Delimiter Routine
PRILNEND	E4	to have Buffer Disposition finish processing macros and cleanup the line	Line End Appendage-send, last buffer only
PRIRCBFR	E0	to return a duplicate header to a specified routine	CPB Cleanup Destination Scheduler
PRIBFRTB	E4	to return a buffer or unit to the buffer unit pool	Incoming/Outgoing Message Delimiter Routine PCI Appendage CPB Cleanup Destination Scheduler Multiple Routing Subtask
PRIDSKBF	FC	to give a unit to CPB Cleanup	Buffer Return
PRICOPY	E0	to have a message copied to a different data set	Destination Scheduler
PRIDESTO	E4	to put a buffer on a message queue	Incoming/Outgoing Message Delimiter Routine Multiple Routing Subtask Create an Error Message Routine
PRIDKWRT	E4	to have a full buffer written on disk	Destination Scheduler

PRIDKSRV	EC	to have a message flagged serviced	Buffer Cleanup
PRIDKCNC	E0	to have a message canceled in the message queue	Cancel Message
PRIDKINT	E0	to have a message intercepted	Hold/Release Terminal Routine
PRICKPLN	EC	to tpost the LCB to Checkpoint requesting a checkpoint	Buffer Disposition
PRIMULTR	E0	to tpost the LCB to the Multiple Router routine to continue	Buffer Disposition TLIST
PRIOPCTL	DC	to tpost an operator control buffer	Message Handling Routine Operator Control Interface Routine
PRIDSPLB	E4	to tpost last buffer of message to Buffer Disposition QCB; must be lower than any PCI tpost of an ERB	Incoming/Outgoing Message Delimiter Routine
PRIONLT	DC	to request On-Line Test	STARTMH Subtask
PRILAEND	E4	to start error processing	Line End Appendage
PRIMHUNT	E8	to tpost a unit to MH; must be greater than PRIMHBFR	Unit Request
PRIRELSE	E0	to release a subtask from Time Delay or Operator Control	Operator Control Hold/Release Terminal
PRICPBCL	E8	to Post CPB Cleanup complete	Disk End Appendage
PRICKPT	DC	to request a complete checkpoint	Reusability-Copy subtask MCPCLOSE Time Delay subtask
PRILNFRE	E8	to free a line; must get to Destination Scheduler before line is free	Buffer Disposition Put Scheduler Send Scheduler

PRICLSDN	10	to request closedown; must be lowest priority	
PRIAPCKP	DC	to request an incident checkpoint	Application Program
PRIOPCKP	DC	to request an incident checkpoint	Operator Control
PRILNCL	EC	to clean up buffers and to free a line	Line End Appendage
		to tpost a line to Buffer Disposition	INEND OUTEND
PRILOGLB	E0	to tpost the LOG LCB to itself	LOG Scheduler
PRISSOLT	DC	tposted to Operator Control to request Startline/Stopline to return an element from the time delay queue	On-Line Test Time Delay
PRIATTN	DC	to tpost the attention element for local devices	Attention Handler
PRISYSDL	DC	to initiate system delay	Operator Control
PRILCBDL	20	to indicate to Environment Checkpoint that an LCB is on the System Delay queue to Destination	System Delay Subtask Environment Checkpoint Routine

Appendix D: TCAM CHANNEL PROGRAMS AND TP OPERATION CODES

The format of the TCAM channel command word (CCW) is as follows:

Offset	0 7	8 31
	Command Code	Data Address

32 36	37 39	40 47	48 63
Flags	000	Reserved	Count

The TCAM Channel programs are generated by IO Generator Module (IEDOKA). Channel programs are listed by operation types within communication line types. The description of each channel program begins with a graphic representation of the model channel program according to the following categories:

1. Operation - The command code with a brief description of the information that is being transferred.

2. Address - The data address that is set in the CCW before execution.

   - Buffer refers to the buffer CCW address.
   - Table refers to the appropriate location in the Special Characters Table.
   - List refers to the applicable invitation or addressing list entry.
   - LCB refers to the Line Control Block.
   - Entry refers to addressing characters, dial digits, etc., in a terminal entry.
   - Idles refers to an idles loop that is used to process data.

3. Flags - The flags that are set in the generated CCW are: Chain Command (CC), Chain Data (CD), and Suppress Length Indication (SLI).

4. Count - The data count that is set in the generated CCW before execution.

A Tp OP code differentiates among the types of CCWs on which interrupts can cccur. In TCAM, the Activate-I/O Generator subtask builds a string of Tp OP codes for any given channel program in the LCB. There is one Tp OP Code for each CCW. These codes are retrieved and used by Line End Appendage. An even-valued Tp OP code represents a text or non-text CCW for which an interrupt is anticipated. An odd-valued Tp OP code represents a CCW for which no interrupt is anticipated. The following is a list of the TCAM Tp OP codes:

Name	Value	Description
TPWREOT	X'01' .	Write EOT For SELECTION
TPOPEN	X'02' .	Open TP OP Code
TPWRPOLL	X'03' .	Write Polling Characters
TPRDRESP	X'04' .	Read Response to Polling
TPWRPAD	X'05' .	Write Pads
TPENABLE	X'06' .	Enable on Dial Line
TPWRAD	X'07' .	Write Addressing Sequence
TPRDRSPD	X'08' .	Read Response to Addressing
TPWREOA	X'09' .	Write EOA Sequence
TPRDRPEB	X'0A' .	Read Response to EOB/ETB
TPWRCPU	X'0B' .	Write CPU ID
TPRDENQ	X'0C' .	Read ENQ
TPWRENQ	X'0D' .	Write ENQ
TPRSPENQ	X;0E' .	Read Response to ENQ
TPWRDLET	X'0F' .	Write DLE EOT
TPRDID	X'10' .	Read ID
TPNULL	X'11' .	Non Read Write CCW's for which no Interrupt is anticipated
TPBREAK	X'12' .	WRITE BREAK TP-CD
TPENOAD	X'13' .	WRITE ENQ after Selection Response
TPRDLC	X'14' .	Read LCOUT
TPWRACK	X'15' .	Write Response Prior to Text

TPWRAKNK	X'16' .	Write Response
TPWRTONE	X'17' .	Write Tone WTTA BSC
TPRDIDNQ	X'18' .	BSC Read ID ENQ
TPRDIDAK	X'1A' .	BSC Read ID ACK
TPRESET	X'1C' .	Abort for Send/Receive
TPTWXID	X'1E' .	Read TWX ID
TPBUFEOT	X'20' .	Buffered Terminal Reset after Block
TPCLOSE	X'22' .	Close SDR Recording
TPRSPAD	X'24' .	Write Reset after Selection
TPRDSKIP	X'51' .	Read Skip Loop
TPWRIDLE	X'53' .	Write Idles Loop
TPDLESTX	X'57' .	Write DLE STX
TPDLEETX	X'59' .	Write DLE ETB (ETX)
TPENQRSP	X'5B' .	Write ENQ in Response to Text
TPTEXT	X'FF' .	Text CCW

The first two CCWs in Read Initial channel programs are the following:

	Operation	Address	Flags	Tp Code	Count
A	Read	Skip	CC,SLI	51	1
	TIC	A			

These CCWs are executed whenever a buffer is not available. The initial contact CCWs are constructed in the channel program area plus 16 (3rd CCW).

When an idle character is defined for a device, the first two CCWs in Write Initial channel programs are the following:

	Operation	Address	Flags	Tp Code	Count
A	Write	Idles	CC,SLI	53	3
	TIC	A			

CHANNEL PROGRAMS FOR THE AT&T 83B3 SELECTIVE CALLING STATION LINES

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	01	
Write polling characters	List	CC,SLI	03	
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program places the line in control mode by sending the EOT sequence, polls the terminal, and then reads the response. The read response command has a data count of 2. Thus, when there is a one-byte positive response, the response is followed by data. This reduces the count to zero and causes data chaining to read the rest of the data until an EOB or EOT is received or the count is zero. A negative response causes channel end and device end with unit exception and wrong length indicated. Line End Appendage finds the polling restart Tp code, reinitializes for the next terminal to be polled, and returns control to IOS for execution of the CCWs.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	01	3
Write addressing characters	T entry	CC,SLI	07	2
Read response	LCB		08	9
Write EOA sequence	Table	CD	09	
TIC	Idles			

The write initial channel program places the line in control mode, addresses a terminal, and reads the response. An interrupt is taken on the read response, after which buffers are tposted to the outgoing MH. Restart is made at the write EOA sequence, which TICs to the idles loop and from there writes data.

CHANNEL PROGRAMS FOR WESTERN UNION PLAN 115A OUTSTATION

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	07	3
Write polling	List	CC,SLI	03	
characters				2
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program places the line in control mode by sending the EOT sequence, polls the terminal, and then reads the response. The read response command has a data count of 2. Thus, when there is a one-byte positive response, the response is followed by data. This reduces the count to zero and causes data chaining to read the rest of the data until an EOB or EOT is received or the count is zero. A negative response causes channel end and device end with unit exception and wrong length indicated. Line End Appendage finds the polling restart Tp code, reinitializes for the next terminal to be polled, and returns control to IOS for execution of the CCWs.

Write Initial Channel Prcgram

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	01	3
Write addressing characters	T entry	CC,SLI	07	2
Read response	LCB		08	9
Write EOA sequence	Table	CD	09	4
TIC	Idles			

The write initial channel program places the line in control mode, addresses a terminal, and reads the response. An interrupt is taken on the read response, after which buffers are tposted to the outgoing MH. Restart is made at the write EOA sequence, which TICs to the idles loop and from there writes data.

CHANNEL PROGRAMS FOR IBM 1030 LINES

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	01	3
Write polling characters	List	CC,SLI	03	1
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program places the line in control mode by sending the EOT sequence, polls the terminal, and then reads the response. The read response command has a data count of 2. Thus, when there is a one-byte positive response, the response is followed by data. This reduces the count to zero and causes data chaining to read the rest of the data until an EOB or ECT is received or the count is zero. A negative response causes channel end and device end with unit exception and wrong length indicated. Line End Appendage finds the polling restart Tp code, reinitializes for the next terminal to be polled, and returns control to IOS for execution of the CCWs.

Read Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Write positive (ACK) or negative (NAK) response	Table		16	1

The read continue channel program sends a positive or negative response to the previcus message block to indicate a response from TCAM.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	01	3
Write addressing characters	T entry	CC,SLI	07	2
Read response	LCB		08	9
Write EOA sequence	Table	CD	09	1
TIC	Idles			

The write initial channel program places the line in control mode, addresses a terminal, and reads the response. An interrupt is taken on the read response, after which buffers are tposted to the outgoing MH. Restart is made at the write EOA sequence, which TICs to the idles loop and from there writes data.

Write Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Read response	LCB	CC,SLI	0A	9
TIC	Buffer			

The write continue channel program reads the response to the last message block. If the response is positive, chaining takes place to the next write text command.

CHANNEL PROGRAMS FOR IBM 1050 LEASED LINES

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	01	3
Write polling characters	List	CC,SLI	03	2
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program places the line in control mode by sending the EOT sequence, polls the terminal, and then reads the response. The read response command has a data count of 2. Thus, when there is a one-byte positive response, the response is followed by data. This reduces the count to zero and causes data chaining to read the rest of the data until an EOB or EOT is received or the count is zero. A negative response causes channel end and device end with unit exception and wrong length indicated. Line End Appendage finds the polling restart Tp code, reinitializes for the next terminal to be polled, and returns control to IOS for execution of the CCWs.

Read Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Write positive or negative response	Table	CC,SLI	16	1
TIC	Buffer			

The read continue channel program writes the appropriate response to a block of data and then chains to read data.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	01	3
Write addressing characters	T entry	CC,SLI	07	2
Read response	LCB		08	9
Write EOA sequence	Table	CD	09	1
TIC	Idles			

The write initial channel program places the line in control mode, addresses a terminal, and reads the response. An interrupt is taken on the read response, after which buffers are tposted to the outgoing MH. Restart is made at the write EOA sequence, which TICs to the idles loop and from there writes data.

Write Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Read response	LCB	CC,SLI	0A	9
TIC	Buffer			

The write continue channel program reads the response to the last message block. If the response is positive, chaining takes place to the next write text command.

Write Conversational Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOA	Table	CD,SLI	09	1
TIC	Idles			

The write conversational channel program writes end-of-address and then chains to write data.

CHANNEL PROGRAMS FOR IBM 1050 DIAL

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Disable		CC,SLI	11	1
Enable		SLI	06	1
Write EOT sequence		CD	01	3
Write polling characters	List	CC,SLI	03	2
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program disables and then enables the line adapter so that a remote terminal may dial the CPU. An interrupt is taken on the enable so that TCAM can set internal switches. Fifteen pad characters are sent by the CPU, followed by an EOT sequence; this places the terminal in control mode. Two polling characters are sent and then a read response that specifies a data count of two, with wrong length indication not suppressed, while the length of the response character is one byte. The effect of this technique is as follows:

1. Positive response. The response character and the first byte of the message are read under control of the Read Response CCW. This reduces the data count to zero and causes data chaining to take place. The second and subsequent bytes of the message are read under control of the address and count fields of the Read Data CCW. Execution continues in the channel with an interrupt occurring only at receipt of an EOB or EOT.

2. Negative response. This response causes channel end and device end with unit exception and wrong length record indicated.

The read initial channel program then transfers-in-channel (TICs) to the address in the buffer CCW to read data.

Read Continue Channel Prcqram

Operation	Address	Flags	Tp Code	Count
Write positive (ACK) or negative (NAK) response	Table	CC,SLI	16	1
TIC	Buffer			

The read continue channel program sends a positive or negative response to the previous message block and continues reading data.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Disable		CC,SLI	11	1
Dial	T entry	CC,SLI	11	X
Write pad characters	Table	CD,SLI	05	15
Write EOT sequence	Table	CD,SLI	01	3
Write addressing characters	T entry	CC,SLI	07	2
Read response to address	LCB	SLI	C8	9
Write EOA	Table	CD,SLI	09	1
TIC	Idles			

The write initial channel program disables the line and then dials a terminal. When the remote terminal answers, the CPU sends the pad characters and the EOT sequence, which places the terminal in control mode. The address characters select the component, which responds to the addressing. End-of-address terminates addressing and then the write initial channel program TICs to the idles loop and from there to a write data. The X count value depends on the number of dial digits specified in the terminal entry.

Write Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Read response	LCB	CC,SLI	0A	9
TIC	Buffer			

The write continue channel program reads the response to the last message block. If the response is positive, chaining takes place to the next write text command.

Write Conversational Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOA	Table	CD,SLI	09	1
TIC	Idles			

The write conversational channel program writes End-of-Address character and then TICs to a write idles loop to write data.

CHANNEL PROGRAMS FOR IBM 1060 TERMINALS

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	01	3
Write polling characters	List	CC,SLI	03	2
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program places the line in control mode by sending the EOT sequence, polls the terminal, and then reads the response. The read response command has a data count of 2. Thus, when there is a one-byte positive response, the response is followed by data. This reduces the count to zero and causes data chaining to read the rest of the data until an EOB or EOT is received or the count is zero. A negative response causes channel end and device end with unit exception and wrong length indicated. Line End Appendage finds the polling restart Tp code, reinitializes for the next terminal to be polled, and returns control to IOS for execution of the CCWs.

Read Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Write positive (ACK) or negative (NAK) response	Table	SLI	16	1

The read continue channel program sends a positive or negative response to the previous message block and continues reading data to the previous block.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	01	3
Write addressing characters	T entry	CC,SLI	07	2
Read response	LCB		08	9
Write EOA sequence	Table	CD	09	1
TIC	Idles			

The write initial channel program places the line in control mode, addresses a terminal, and reads the response. An interrupt is taken on the read response, after which buffers are tposted to the outgoing MH. Restart is made at the write EOA sequence, which TICs to the idles loop and from there writes data.

Write Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Read response	LCB	CC,SLI	0A	9
TIC	Buffer			

The write continue channel program reads the response to the last message block. If the response is positive, chaining takes place to the next write text command.

CHANNEL PROGRAMS FOR IBM 2741 LEASED

Read Initial Channel Program

Operation	Address	Flags	TP Code	Count
Write EOT sequence	Table	CC,SLI	07	3
Prepare		CC,SLI	11	1
Sense	LCB	CC,SLI	11	1
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program sends a write EOT sequence and then prepares the control unit to receive a message from a terminal. The sense operation informs the CPU of the status of the terminal through the read response. The read initial channel program then TICs to read data.

Write Initial Channel Program

Operation	Address	Flags	TP Code	Count
Write EOA sequence	Table	CD,SLI	09	1
Write Idle characters	Table	CD,SLI	05	15
TIC	Idles			

The write initial channel program sends an EOA sequence to set up the terminal and writes 15 idle characters on the line. The program then TICs to a write command.

CHANNEL PROGRAM FOR IBM 2741 DIAL

Read Initial Channel Program

Operation	Address	Flags	TP Code	Count
Disable	LCB	CC,SLI	11	1
Enable	LCB	SLI	06	1
Prepare	LCB	CC,SLI	11	1
Sense	LCB	CC,SLI	11	1
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program disables and then enables the line to receive a call. TCAM takes an interrupt on the enable to set internal switches. The prepare command conditions the control unit to receive a message. Read response reads the response from the terminal and then chains to read data via the TIC.

Note: The write initial channel program for 2741 Dial is the same as for 2741 Leased. TCAM, however, does not dial a 2741; the user calls to establish the connection.

CHANNEL PROGRAMS FOR IBM 2740 COMMUNICATION LINES

IBM 2740 BASIC CHANNEL PROGRAM

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	07	3
Prepare		CC,SLI	11	1
Sense	LCB	CC,SLI	11	1
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program sends a write EOT sequence and then prepares the ccntrol unit to receive a message from a terminal. The sense operation informs the CPU of the status of the terminal through the read response. The read initial program then TICs to read data.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CD,SLI	01	3
Write EOA sequence	Table	CD,SLI	09	1
Write idle characters	Table	CD,SLI	05	15
TIC	Idles			

The write initial channel program sends an EOT and EOA sequence for setting up the terminal. It then writes 15 idle characters and transfers-in-channel to a write command.

IBM 2740 WITH CHECKING

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	01	3
Prepare		CC,SLI	11	1
Sense	LCB	CC,SLI	11	1
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program sends a write EOT sequence, then prepares the ccntrol unit to receive a message from a terminal. The sense operation informs the CPU of the status of the terminal through the read response. The read initial program then TICs to read data.

Read Continue Channel Prcgram

Operation	Address	Flags	Tp Code	Count
Write circle Y or circle N	Table	CC,SLI	16	1
TIC	Buffer			

The read continue channel program is initiated after a read initial operation. The program writes the response character and then TICs to read data.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CD,SLI	01	3
Write EOA sequence	Table	CD,SLI	09	1
Write idle characters	Table	CD,SLI	05	15
TIC	Idles			

The write initial channel program sends an EOT and EOA sequence for setting up the terminal. It then writes 15 idle characters and transfers-in-channel to a write command.

Write Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Read response	LCB	CC,SLI	0A	9
TIC	Buffer			

The write continue channel program reads the response after a write initial operation and then TICs to a write text command in the buffer.

Write Conversaticnal Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOA	Table	CD,SLI	09	1
TIC	Idles			

The write conversational channel program writes End-of-Address character and then TICs to a write idles loop to write data.

IBM 2740 WITH DIAL

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Disable		CC,SLI	11	1
Enable		SLI	06	1
Prepare		CC,SLI	11	1
Sense	LCB	CC,SLI	11	1
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program disables and then enables the line to receive a call. TCAM takes an interrupt on the enable to set internal switches. The prepare command conditions the control unit to receive a message. Read response reads the response from the terminal and then chains to read data via the TIC.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Disable		CC,SLI	11	1
Dial	T entry	CC,SLI	11	X
Write pad characters	Table	CD,SLI	05	15
Write EOT sequence	Table	CD,SLI	01	3
Write EOA plus idles	Table	CD,SLI	09	16
TIC	Idles			

The write initial channel program disables the line and then dials the specified terminal. The channel program sends 15 pad characters before the EOT sequence. An EOA character plus 15 idle characters are sent and then the program TICs to write text. The X count value depends on the number of dial characters specified in the terminal entry.

IBM 2740 WITH DIAL AND CHECKING

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Disable		CC,SLI	11	1
Enable		SLI	06	1
Prepare		CC,SLI	11	1
Sense	LCB	CC,SLI	11	1
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program disables and then enables the line to receive a call. The prepare command conditions the control unit to receive a message. Read response reads the terminal's response and then chains to read data via the TIC.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Disable		CC,SLI	11	1
Dial	T entry	CC,SLI	11	X
Write pad characters	Table	CD,SLI	05	15
Write EOT sequence	Table	CD,SLI	01	3
Write EOA plus idles	Table	CD,SLI	09	16
TIC	Idles			

The write initial channel program disables the line and then dials the specified terminal. The channel program sends 15 pad characters before the EOT sequence. An EOA character plus 15 idle characters are sent and then the program TICs to write text. X represents the number of dial digits for the terminal.

IBM 2740 WITH DIAL AND TRANSMIT CONTROL

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Disable		CC,SLI	11	1
Enable		SLI	C6	1
Write EOT sequence		CD	01	3
Write polling characters	List	CC,SLI	03	2
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program disables and then enables the line adapter so that a remote terminal may dial the CPU. After the enable TCAM waits for an interrupt from the terminal, after which the channel program resumes. Fifteen pad characters are sent by the CPU, followed by an EOT sequence; this places the terminal in control mode. Two polling characters are sent and then a read response that specifies a data count of two. The effect of this technique is as follows:

1. Positive response. The response character and the first byte of the message are read under control of the Read Response CCW. This reduces the data count to zero and causes data chaining to take place. The second and subsequent bytes of the message are read under control of the address and count fields of the Read Data CCW. Execution continues in the channel with an interrupt occurring only at receipt of an EOB or EOT.

2. Negative response. This response causes channel end and device end with unit exception and wrong length record indicated.

The read initial channel program then transfers-in-channel (TICs) to the address in the buffer CCW to read data.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Disable		CC,SLI	11	1
Dial	T entry	CC,SLI	11	X
Write pad characters	Table	CD,SLI	05	15
Write EOT sequence	Table	CD,SLI	01	3
Write EOA plus idles	Table	CD,SLI	09	16
TIC	Idles			

The write initial channel program disables the line and then dials the specified terminal. The channel program sends 15 pad characters before the EOT sequence. An EOA character plus 15 idle characters are sent and then the program TICs to write text. X represents the number of dial digits for the terminal.

IBM 2740 WITH DIAL, TRANSMIT CONTROL, AND CHECKING

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Disable		CC,SLI	11	1
Enable		SLI	06	1
Write EOT sequence		CD	01	3
Write polling characters	List	CC,SLI	03	2
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program disables and then enables the line adapter so that a remote terminal may dial the CPU. After the enable, TCAM waits for an interrupt from the terminal, after which the channel program resumes. Fifteen pad characters are sent by the CPU, followed by an EOT sequence; this places the terminal in control mode. Two polling characters are sent and then a read response that specifies a data count of two. The effect of this technique is as follows:

1. Positive response. The response character and the first byte of the message are read under control of the Read Response CCW. This reduces the data count to zero and causes data chaining to take place. The second and subsequent bytes of the message are read under control of the address and count fields of the Read Data CCW. Execution continues in the channel with an interrupt occurring only at receipt of an EOB or EOT.

2. Negative response. This response causes channel end and device end with unit exception and wrong length record indicated.

The read initial channel program then transfers-in-channel (TICs) to the address in the buffer CCW to read data.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Disable		CC,SLI	11	1
Dial	T entry	CC,SLI	11	X
Write pad characters	Table	CD,SLI	05	15
Write EOT sequence	Table	CD,SLI	01	3
Write EOA plus idles	Table	CD,SLI	09	16
TIC	Idles			

The write initial channel program disables the line and then dials the specified terminal. The channel program sends 15 pad characters before the EOT sequence. An EOA character plus 15 idle characters are sent and then the program TICs to write text. X represents the number of dial digits for the terminal.

IBM 2740 (DIAL WITH A CONNECTION)

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	01	3
Prepare		CC,SLI	11	1
Sense	LCB	CC,SLI	11	1
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program sends a write EOT sequence, then prepares the control unit to receive a message from a terminal. The sense operation informs the CPU the status of the terminal through the read response. The read initial program then TICs to read data.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CD,SLI	01	3
Write EOA sequence	Table	CD,SLI	09	1
Write idle characters	Table	CD,SLI	05	15
TIC	Idles			

The write initial channel program sends an EOT and EOA sequence for setting up the terminal. It then writes 15 idle characters and transfers-in-channel to a write command.

IBM 2740 WITH CHECKING (DIAL WITH A CONNECTION)

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	11	3
Prepare		CC,SLI	11	1
Sense	LCB	CC,SLI	11	1
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program sends a write EOT sequence, then prepares the control unit to receive a message from a terminal. The sense operation informs the CPU the status of the terminal through the read response. The read initial program then TICs to read data.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CD,SLI	01	3
Write EOA sequence	Table	CD,SLI	09	1
Write idle characters	Table	CD,SLI	05	15
TIC	Idles			

The write initial channel program sends an EOT and EOA sequence for setting up the terminal. It then writes 15 idle characters and transfers-in-channel to a write command.

IBM 2740 WITH STATION CONTROL

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	01	3
Write polling characters	List	CC,SLI	03	2
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program places the line in control mode by sending the EOT sequence, polls the terminal, and then reads the response. The read response command has a data count of 2. Thus, when there is a one-byte positive response, the response is followed by data. This reduces the count to zero and causes data chaining to read the rest of the data until an EOB or EOT is received or the count is zero. A negative response causes channel end and device end with unit exception and wrong length indicated. Line End Appendage finds the polling restart Tp code, reinitializes for the next terminal to be polled, and returns control to IOS for execution of the CCWs.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	01	3
Write addressing characters	T entry	CC,SLI	07	2
Read response	LCB		08	9
Write EOA sequence	Table	CD	09	1
TIC	Idles			

The write initial channel program places the line in control mode, addresses a terminal, and reads the respcnse. An interrupt is taken on the read response, after which buffers are tposted to the outgoing MH. Restart is made at the write EOA sequence, which TICs to the idles loop and from there writes data.

IBM 2740 WITH STATION CONTROL AND CHECKING

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	01	3
Write polling characters	List	CC,SLI	03	2
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program places the line in control mode by sending the EOT sequence, polls the terminal, and then reads the response. The read response command has a data count of 2. Thus, when there is a one-byte positive response, the response is followed by data. This reduces the count to zero and causes data chaining to read the rest of the data until an EOB or EOT is received or the count is zero. A negative response causes channel end and device end with unit exception and wrong length indicated. Line End Appendage finds the polling restart Tp code, reinitializes for the next terminal to be polled, and returns control tc IOS for execution of the CCWs.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	01	3
Write addressing characters	T entry	CC,SLI	07	2
Read response	LCB		08	9
Write EOA sequence	Table	CD	09	1
TIC	Idles			

The write initial channel program places the line in control mode, addresses a terminal, and reads the response. An interrupt is taken on the read respcnse, after which buffers are tposted to the outgoing MH. Restart is made at the write EOA sequence, which TICs to the idles loop and from there writes data.

IBM 2740 WITH TRANSMIT CCNTROL (DIAL WITH A CONNECTION)

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	01	3
Write polling characters	List	CC,SLI	03	2
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program places the line in control mode by sending the EOT sequence, polls the terminal, and then reads the response. The read response command has a data count of 2. Thus, when there is a one-byte positive response, the response is followed by data. This reduces the count to zero and causes data chaining to read the rest of the data until an EOB or ECT is received or the count is zero. A negative response causes channel end and device end with unit exception and wrong length indicated. Line End Appendage finds the polling restart Tp code, reinitializes for the next terminal to be polled, and returns control to IOS for execution of the CCWs.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	01	3
Write EOA sequence	Table	CD,SLI	09	1
Write idle characters	Table	CD,SLI	05	15
TIC	Idles			

The write initial channel program sends an EOT and EOA sequence for setting up the terminal. It then writes 15 idle characters and TICs to a write command.

IBM 2740 WITH TRANSMIT CONTROL AND CHECKING (DIAL WITH A CONNECTION)

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	01	3
Write polling characters	List	CC,SLI	03	2
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program places the line in control mode by sending the EOT sequence, polls the terminal, and then reads the response. The read response ccmmand has a data count of 2. Thus, when there is a one-byte positive response, the response is followed by data. This reduces the count to zero and causes data chaining to read the rest of the data until an EOB or EOT is received or the count is zero. A negative response causes channel end and device end with unit exception and wrong length indicated. Line End Appendage finds the polling restart Tp code, reinitializes for the next terminal to be polled, and returns control to IOS for execution of the CCWs.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	01	3
Write EOA sequence	Table	CD	09	
TIC	Idles			

The write initial channel program places the line in control mode. The program then issues the write EOA sequence, TICs to the idles loop, and from there writes data.

CHANNEL PROGRAMS FOR WORLD TRADE TELEGRAPH

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Prepare		CC,SLI	11	1
Sense	LCB	CC,SLI	11	1
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program prepares the control unit to receive a message from a terminal. The sense operation informs the CPU of the status of the terminal through the read response. The read initial program then TICs to a read text ccmmand in the buffer.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CD,SLI	01	3
Write mark characters	Table	CD,SLI	05	21
Write	WRU	CC,SLI	07	1
Read response	LCB		08	24
Write EOA sequence	Table		09	1
TIC	Idles			

The write initial channel program writes an EOT sequence, sends 21 mark characters to condition the line, and writes a WRU on the line, and reads the response. An interrupt is taken on the read response, after which the buffers are tposted to outgoing MH. Restart is at the Write EOA sequence, which TICs to the idles loop and writes data.

IBM 2260 REMOTE CHANNEL PROGRAMS

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	01	3
Write polling characters	List	CC,SLI	03	3
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program places the line in control mode by sending the EOT sequence, polls the terminal, and then reads the response. The read response command has a data count of 2. Thus, when there is a one-byte positive response, the response is followed by data. This reduces the count to zero and causes data chaining to read the rest of the data until an EOB or EOT is received or the count is zero. A negative response causes channel end and device end with unit exception and wrong length indicated. Line End Appendage finds the polling restart Tp code, reinitializes for the next terminal to be polled, and returns control to IOS for execution of the CCWs.

Read Continue Channel Program

Operation	Address	Flags	TP Code	Count
Write positive (ACK) or negative (NAK) response	Table	CC,SLI	16	1
TIC	Buffer			

The read continue channel program sends a positive or negative response to the previous message block and continues reading data.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write EOT sequence	Table	CC,SLI	01	3
Write address	T entry	CC,SLI	07	2
Read response	LCB		08	9

The write initial channel program writes an EOT sequence followed by an address. After the read response, the buffers are tposted to MH and data is transferred to the line by EXCP.

Write Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Read response	LCB	CC,SLI	0A	9
TIC	Buffer			

The write continue channel program reads the response to the last message block. If the response is positive, chaining takes place to the next write text command.

IBM 2260 LOCAL CHANNEL PROGRAMS

In local mode the channel programs simply read data or write data, as the case may be.

CHANNEL PROGRAMS FOR IBM 7770 (DIAL)

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Disable		CC,SLI	11	1
Enable		SLI	06	1
Write CPU ID (if ID is specified)	T entry	CC,SLI	0B	X
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program disables and then enables the line. The CPU ID is written if this is specified, and then the program chains to a read response. The X count value is the length of the CPU ID specified in the invitation list.

Write Initial Channel Program

This program simply writes data to the 7770.

CHANNEL PROGRAMS FOR TTY MODELS 33 AND 35 TWX LINES

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Disable		CC,SLI	11	1
Enable		SLI	06	1
Write CPU ID	T entry	CC,SLI	0B	X
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program disables the line and sets the enable latch within the line adapter. This permits the terminal to dial the CPU. The write CPU ID command writes the CPU identification, which is assigned by the invitation list for the line. A read response command is then issued, followed by a TIC to a read text in the buffer. X is the length of the CPU ID specified in the invitation list.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Disable		CC,SLI	11	1
Dial	T entry	CC,SLI	11	X
Read ID	LCB	SLI	IE	Y

The write initial channel program disables and then dials the specified terminal. If the identification received is valid, the program restarts on the idles loop and writes data. If the ID is invalid, the channel program is terminated. X represents the number of dial digits for the terminal and Y represents the length of the CPU ID specified in the invitation list.

CHANNEL PROGRAMS EMPLOYING THE AUTO POLL FEATURE

The devices that use this feature are the following:

IBM 1030
IBM 1050 (nonswitched)
IBM 1060
IBM 2740 (with station control)
IBM 2740 (with station control and checking)
BSC Multipoint

	Operation	Address	Flags	Tp Code	Count
	Write EOT sequence	Table	CC,SLI	01	X
	Poll	List	CC,SLI	11	Y
	TIC	A			
	TIC	B			
A	Poll	List	CC,SLI	11	Z
	TIC	A			
B	Read	Buffer	CD,SLI	04	2
	TIC	Buffer			

This feature employs the read initial type of channel program. First, a write EOT sequence command is sent, followed by a poll of the addresses in the invitation list. If no positive responses are returned, the program TICs to a poll of another list. If there are positive responses, the read initial program TICs to read response command, and from there chains to a read text in the buffer. X represents the number of EOTs dependent on the type of terminal (1 for BSC, 3 for all others), Y represents the position in the invitation list, and Z is the length in bytes of the invitation list.

CHANNEL PROGRAMS FOR IBM BSC MULTIPOINT LINES

Read Initial Channel Program

Operation	Address	Flags	TP Code	Count
Write EOT sequence	Table	CC,SLI	01	3
Write polling characters	List	CC,SLI	03	2
Read response	Buffer	CD,SLI	04	2
TIC	Buffer			

The read initial channel program places the line in control mode by sending the EOT sequence, polls the terminal, and then reads the response. The read response command has a data count of 2. This reduces the count to zero and causes data chaining to read the rest of the data until an ETB or ETX is received or the count is zero. A negative response causes channel end and device end with unit exception and wrong length indicated. Line End Appendage finds the polling restart Tp code, reinitializes for the next terminal to be polled, and returns control to IOS for execution of the CCWs.

Read Continue Channel Program

Operation	Address	Flags	TP Code	Count
Write ACK or NAK response	Table	CC,SLI	16	2
TIC	Buffer			

The read continue channel programs writes the appropriate response to a block of data and then chains to read data.

Write Initial Channel Program

Operation	Address	Flags	TP Code	Count
Write EOT sequence	Table	CC,SII	01	3
Write addressing characters	T entry	CC,SII	07	
Read response	LCB		08	9

The write initial channel program places the line in control mode, addresses a terminal, reads the response (ACK-1), and then begins transmission of data.

Write Continue Channel Program

Operation	Address	Flags	TP Code	Count
Read response	LCB	CC,SII	0A	9
TIC	Buffer			

The write continue channel program reads the response to the last message block. If the response is positive, chaining takes place to the next write text command.

CHANNEL PROGRAMS FOR BSC DEVICES (BINARY SYNCHRONOUS COMMUNICATION)

The devices supported under BSC channel programs are the following:

IBM 2770
IBM 2780
IBM 1130 Computing System
IBM System/360, all models 20 and higher

CHANNEL PROGRAMS FOR S/360 to S/360 POINT TO POINT

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Prepare		CC,SLI	11	1
Read inquiry	LCB		0C	11
Write ACK-0	Table	CC,SLI	15	2
TIC	Buffer			

The read initial channel program prepares the control unit to receive an inquiry signal, which is read by the read command. The program then writes an ACK-0 and TICs to a read command in the buffer.

Read Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Write ACK or NAK	Table	CC,SLI	16	2
TIC	Buffer			

The read continue channel program writes a response (ACK or NAK) and TICs to a read data command in the buffer.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write inquiry	Table	CC,SLI	0D	1
Read response	LCB	SLI	08	2

The write initial channel program writes an inquiry, reads the response (ACK-0), and then begins transmission of data.

Write Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Read response	LCB	SLI		9

The write continue channel program checks the response to the last block of data (ACK-0, ACK-1, RVI) and restarts on a write data command.

CHANNEL PROGRAMS FOR S/360 TO 1130 POINT TO POINT

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Prepare		CC,SLI	11	1
Read inquiry	LCB		0C	11
Write ACK-0	Table	CC,SLI	15	2
TIC	Buffer			

The read initial channel program prepares the control unit to receive an inquiry signal, which is read by the read command. The program then writes an ACK-0 and TICs to a read command in the buffer.

Read Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Write ACK or NAK	Table	CC,SLI	16	2
TIC	Buffer			

The read continue channel program writes a response (ACK or NAK) and TICs to a read data command in the buffer.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write inquiry	Table	CC,SLI		1
Read response	LCB	SLI	08	2

The write initial channel program writes an inquiry, reads the response (ACK-0), and then begins transmission of data.

Write Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Read response	LCB	SLI	0A	9

The write continue channel program checks the response to the last block of data (ACK-0, ACK-1, RVI) and restarts on a write data command.

CHANNEL PROGRAMS FOR S/360 TO 2770 POINT TO POINT

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Prepare		CC,SLI	11	1
Read inquiry	LCB		0C	11
Write ACK-0	Table	CC,SLI	15	2
TIC	Buffer			

The read initial channel program prepares the control unit to receive an inquiry signal, which is read by the read command. The program then writes an ACK-0 and TICs to a read command in the buffer.

Read Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Write ACK or NAK	Table	CC,SLI	16	2
TIC	Buffer			

The read continue channel program writes a response (ACK or NAK) and TICs to a read data command in the buffer.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write inquiry	Table	CC,SLI	0D	1
Read response	LCB	SLI	0E	2
Write escape sequence (STX,ESC or DC,ETB)	T entry	CC,SLI	07	X
Read response	LCB	SLI	08	2

The write initial channel program writes an inquiry, reads the response to that inquiry (ACK-0), writes an excape sequence, reads the response (ACK-1), and then begins transmission of data, X represents the length of the addressing sequence specified in the terminal entry.

Write Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Read response	LCB	SLI		9

The write continue channel program checks the response to the last block of data (ACK-0, ACK-1, RVI) and restarts on a write data command.

CHANNEL PROGRAMS FOR S/360 TO 2780 POINT TO POINT

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Prepare		CC,SLI	11	1
Read inquiry	LCB		0C	11
Write ACK-0	Table	CC,SLI	15	2
TIC	Buffer			

The read initial channel program prepares the control unit to receive an inquiry signal, which is read by the read command, The program then writes an ACK-0 and TICs to a read command in the buffer.

Read Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Write ACK or NAK	Table	CC,SLI	16	2
TIC	Buffer			

The read continue channel program writes a response (ACK or NAK) and TICs to a read data command in the buffer.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Write inquiry	Table	CC,SLI	0D	1
Read response	LCB	SLI	0E	2
Write escape sequence STX,ESC or DC,ETB	T entry	CC,SLI	07	X
Read response	LCB	SLI	08	2

The write initial channel program writes an inquiry, reads the response (ACK-0), writes the escape sequence, reads the response to the escape sequence (ACK-1)', and then begins transmission of data. X represents the length of the addressing sequence specified in the terminal entry.

Write Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Read response	LCB	SLI		9

The write continue channel program checks the response to the last block of data (ACK-0, ACK-1, RVI) and restarts on a write data command.

CHANNEL PROGRAMS FOR S/360 to S/360 DIAL

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Disable		CC,SLI	11	1
Enable		CC,SLI	06	1
Read ID inquiry	LCB	SLI	18	16
Write ID (if ID is specified)	List	CD,SLI	0B	X
Write ACK-0	Table	CC,SLI	15	2
TIC	Buffer			

The read initial channel program disables the line and enables the control unit. The program then reads the inquiry (and writes the CPU ID, if specified). It then writes an ACK-0 and chains to a read text command in the buffer. X represents the length in bytes of the user-specified ID in the invitation list.

Read Initial Channel Program with Connection Established

Operation	Address	Flags	Tp Code	Count
Read inquiry	LCB		0C	17
Write ACK-0	Table	CC,SLI	15	
TIC	Buffer			

The read initial channel program reads the inquiry, writes an ACK-0, and then chains to a read data command.

Read Initial Channel Program - CPU Yields the Right to Transmit

Operation	Address	Flags	Tp Code	Count
Write EOT	Table	CC,SLI	01	1
Read inquiry	LCB		0C	17
Write ACK-0	Table	CC,SLI	15	2
TIC	Buffer			

The read initial channel program writes an EOT character and then reads the inquiry from the station. The read initial channel program then writes an ACK-0 and continues to read data from the station.

Read Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Write ACK or NAK	Table	CC,SLI	16	2
TIC	Buffer			

The read continue channel program writes a response (ACK or NAK) and TICs to a read data command in the buffer.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Disable		CC,SLI	11	1
Dial	T entry	CC,SLI	11	X
Write CPU ID (if ID is specified)	List	CD,SLI	0B	Y
Write inquiry	Table	CC,SLI	0D	1
Read ID ACK-0	LCB	SLI	1A	17

The write initial channel program disables the line and dials the station. The program writes the CPU ID, if specified, and writes an ENQ character. The response is read and the ID is checked. The buffers are tposted to MH, and the channel program restarts at a write command. X represents the number of dial digits for a terminal, and Y is the length of the CPU ID.

Write Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Read response	LCB	SLI	0A	9

The write continue channel program checks the response to the last block of data (ACK-0, ACK-1, RVI) and restarts on a write data command.

CHANNEL PROGRAMS FOR S/360 TO 1130 DIAL

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Disable		CC,SLI	11	1
Enable		CC,SLI	06	1
Read ID inquiry	LCB	SLI	18	16
Write ID (if ID is specified)	List	CD,SLI	0B	X
Write ACK-0	Table	CC,SLI	15	2
TIC	Buffer			

The read initial channel program disables the line and enables the control unit. The program then reads the inquiry (and writes the CPU ID, if specified). It then writes an ACK-0 and chains to a read text command in the buffer. X is the length of the CPU ID.

Read Initial Channel Program with Connection Established

Operation	Address	Flags	Tp Code	Count
Read inquiry	LCB		0C	17
Write ACK-0	Table	CC,SLI	15	2
TIC	Buffer			

The read initial channel program reads the inquiry, writes an ACK-0, and then chains tc a read data command.

Read Initial Channel Program - CPU Yields the Right to Transmit

Operation	Address	Flags	Tp Code	Count
Write EOT	Table	CC,SLI	01	1
Read inquiry	LCB		0C	17
Write ACK-0	Table	CC,SLI	15	2
TIC	Buffer			

The read initial channel program writes an EOT character and then reads the inquiry from the station. The read initial channel program then writes an ACK-0 and continues to read data from the station.

Read Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Write ACK or NAK	Table	CC,SLI	16	2
TIC	Buffer			

The read continue channel program writes a response (ACK or NAK) and TICs to a read data command in the buffer.

Write Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Read response	LCB	SLI	0A	9

The write continue channel program checks the response to the last block of data (ACK-0, ACK-1, RVI) and restarts on a write data command.

CHANNEL PROGRAMS FOR S/360 TO IBM 2770 DIAL

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Disable		CC,SLI	11	1
Enable		CC,SLI	06	1
Read ID inquiry	LCB	SLI	18	16
Write ID (if ID is specified)	List	CD,SLI		X
Write ACK-0	Table	CC,SLI	15	2
TIC	Buffer			

The read initial channel program disables the line and enables the control unit. The program then reads the inquiry (and writes the CPU ID, if specified). It then writes an ACK-0 and chains tc a read text command in the buffer. X is the length of the CPU ID.

Read Initial Channel Program with Connecticn Established

Operation	Address	Flags	Tp Code	Count
Read inquiry	LCB		0C	17
Write ACK-0	Table	CC,SLI	15	
TIC	Buffer			

The read initial channel prcgram reads the inquiry, writes an ACK-0, and then chains tc a read data command.

Read Initial Channel Program - CPU Yields the Right to Transmit

Operation	Address	Flags	Tp Code	Count
Write EOT	Table	CC,SLI	01	1
Read inquiry	LCB		0C	17
Write ACK-0	Table	CC,SLI	15	2
TIC	Buffer			

The read initial channel program writes an EOT character and then reads the inquiry from the station. The read initial channel program then writes an ACK-0 and continues to read data from the station.

Read Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Write ACK or NAK	Table	CC,SLI	16	2
TIC	Buffer			

The read continue channel program writes a response (ACK or NAK) and TICs to a read data command in the buffer.

Write Initial Channel Prcgram

Operation	Address	Flags	Tp Code	Count
Disable		CC,SLI	11	1
Dial digits	T entry	CC,SLI	11	X
Write CPU ID (if ID is specified)	List	CD,SLI	0B	Y
Write inquiry	Table	CC,SLI	0D	1
Read ID ACK-0	LCB	SLI	1A	17
Write escape sequence	T entry	CC,SLI	07	Z
Read ACK-1	LCB		08	9

The write initial channel program disables the line and dials the station. The program writes the CPU ID, if specified, and writes an ENQ character. The response is checked. The buffers are tposted to

MH, and the channel program restarts at the write escape sequence. The ACK-1 is read by the program and then the program chains to a write command. X represents the number of dial digits for a terminal, Y is the length of the CPU ID, and Z is a device dependent variable.

Write Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Read response	LCB	SLI		9

The write continue channel program checks the response to the last block of data (ACK-0, ACK-1, RVI) and restarts on a write data command.

CHANNEL PROGRAMS FOR S/360 TO IBM 2780 DIAL

Read Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Disable		CC,SLI	11	1
Enable		CC,SLI	06	1
Read ID inquiry	LCB	SLI	18	16
Write ID (if ID is specified)	List	CD,SLI		X
Write ACK-0	Table	CC,SLI	15	2
TIC	Buffer			

The read initial channel program disables the line and enables the control unit. The program then reads the inquiry (and writes the CPU ID, if specified). It then writes an ACK-0 and chains to a read text command in the buffer. X is the length of the CPU ID.

Read Initial Channel Program with Connection Established

Operation	Address	Flags	Tp Code	Count
Read inquiry	LCB		0C	17
Write ACK-0	Table	CC,SLI	15	2
TIC	Buffer			

The read initial channel program reads the inquiry, writes an ACK-0, and then chains tc a read data command.

Read Initial Channel Program - CPU Yields the Right to Transmit

Operation	Address	Flags	Tp Code	Count
Write EOT	Table	CC,SLI	01	1
Read inquiry	LCB		0C	17
Write ACK-0	Table	CC,SLI	15	2
TIC	Buffer			

The read initial channel program writes an EOT character and then reads the inquiry from the staticn. The read initial channel program then writes an ACK-0 and continues to read data from the station.

Read Continue Channel Program

Operation	Address	Flags	Tp Code	Count
Write ACK or NAK	Table	CC,SLI	16	2
TIC	Buffer			

The read continue channel program writes a response (ACK or NAK) and TICs to a read data ccmmand in the buffer.

Write Initial Channel Program

Operation	Address	Flags	Tp Code	Count
Disable		CC,SLI	11	1
Dial digits	T entry	CC,SLI	11	X
Write CPU ID (if ID is specified)	List	CD,SLI	0B	Y
Write inquiry	Table	CC,SLI	0D	1
Read ID ACK-0	LCB	SLI	1A	9
Write escape sequence	T entry	CC,SLI	07	Z
Read ACK-1	LCB		08	

The write initial channel program disables the line and dials the station. The program writes the CPU ID, if specified, and writes an ENQ character. The response is checked. The buffers are tposted to MH, and the channel program restarts at the write escape sequence. The ACK-1 is read by the program and then the program chains to a write command. X represents the number of dial digits for the terminal: Y represents the length of the CPU ID specified in the invitation list; and Z represents the length of the addressing sequence in the terminal entry.

SPECIAL CHANNEL PROGRAMS

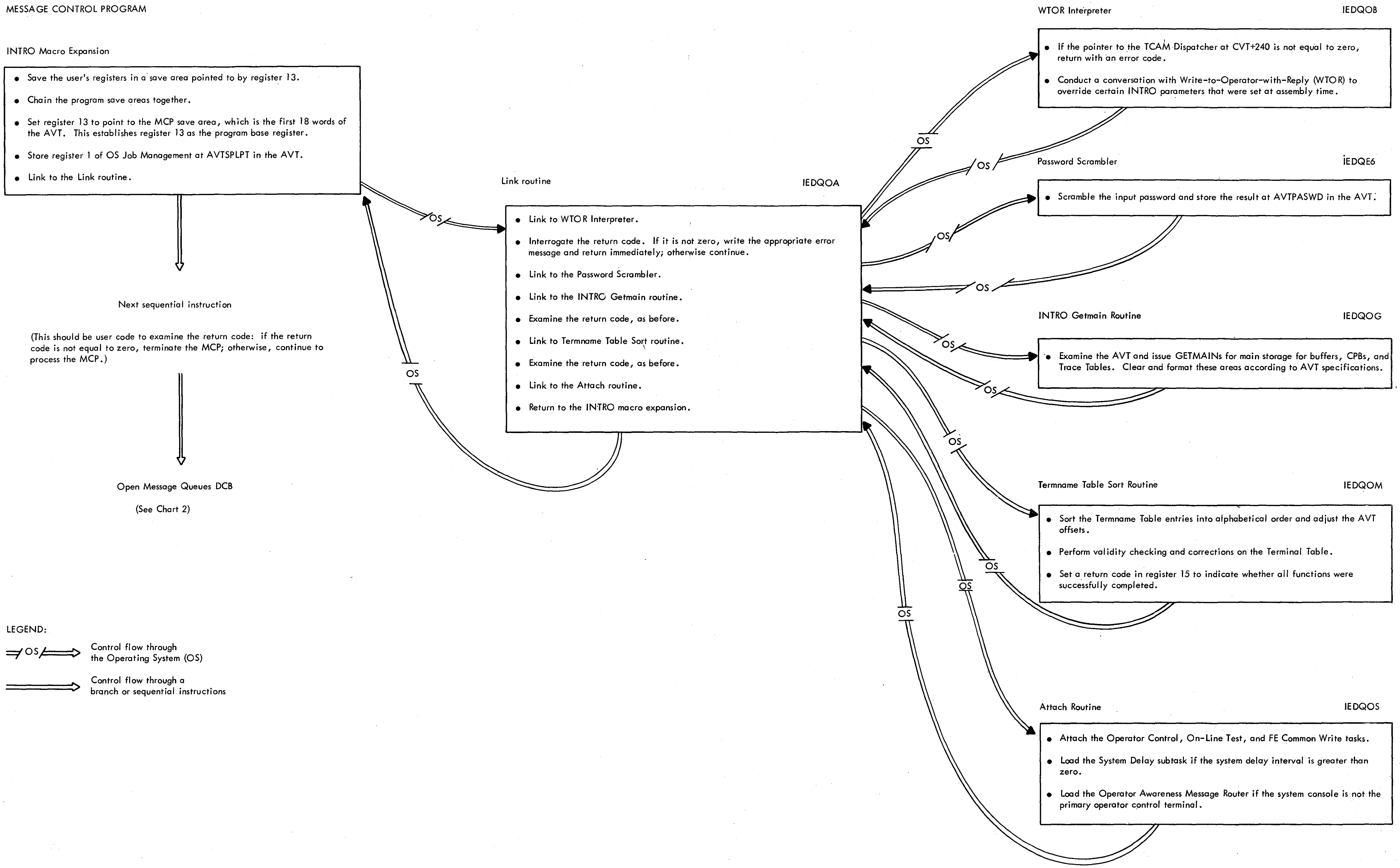
In BSC on a read continue operation, when a temporary time delay (TTD) sequence (STX ENQ) is received the channel program is as follows:

Operation	Address	Flags	Tp Code	Count
Write NAK	Table	CC,SLI	16	2
TIC	Buffer			

When, in response to a text request, TCAM receives two RVIs in succession, a WACK character (except for buffered terminals), or an invalid response, TCAM generates the following channel program to correct the problem.

Operation	Address	Flags	TP Code	Count
Write ENQ	Table	CC,SLI	5B	1
Read Response	LCB		0A	9

For two RVIs or an invalid response, TCAM retries this channel program seven times. For a WACK character, TCAM performs no retry operation.



MESSAGE CONTROL PROGRAM

INTRO Macro Expansion

- Save the user's registers in a save area pointed to by register 13.
- Chain the program save areas together.
- Set register 13 to point to the MCP save area, which is the first 18 words of the AVT. This establishes register 13 as the program base register.
- Store register 1 of OS Job Management at AVTSPLPT in the AVT.
- Link to the Link routine.

Next sequential instruction

(This should be user code to examine the return code: if the return code is not equal to zero, terminate the MCP; otherwise, continue to process the MCP.)

Open Message Queues DCB

(See Chart 2)

Link routine IEDQOA

- Link to WTOR Interpreter.
- Interrogate the return code. If it is not zero, write the appropriate error message and return immediately; otherwise continue.
- Link to the Password Scrambler.
- Link to the INTRO Getmain routine.
- Examine the return code, as before.
- Link to Termname Table Sort routine.
- Examine the return code, as before.
- Link to the Attach routine.
- Return to the INTRO macro expansion.

WTOR Interpreter IEDQOB

- If the pointer to the TCAM Dispatcher at CVT+240 is not equal to zero, return with an error code.
- Conduct a conversation with Write-to-Operator-with-Reply (WTOR) to override certain INTRO parameters that were set at assembly time.

Password Scrambler IEDQE6

- Scramble the input password and store the result at AVTPASWD in the AVT.

INTRO Getmain Routine IEDQOG

- Examine the AVT and issue GETMAINs for main storage for buffers, CPBs, and Trace Tables. Clear and format these areas according to AVT specifications.

Termname Table Sort Routine IEDQOM

- Sort the Termname Table entries into alphabetical order and adjust the AVT offsets.
- Perform validity checking and corrections on the Terminal Table.
- Set a return code in register 15 to indicate whether all functions were successfully completed.

Attach Routine IEDQOS

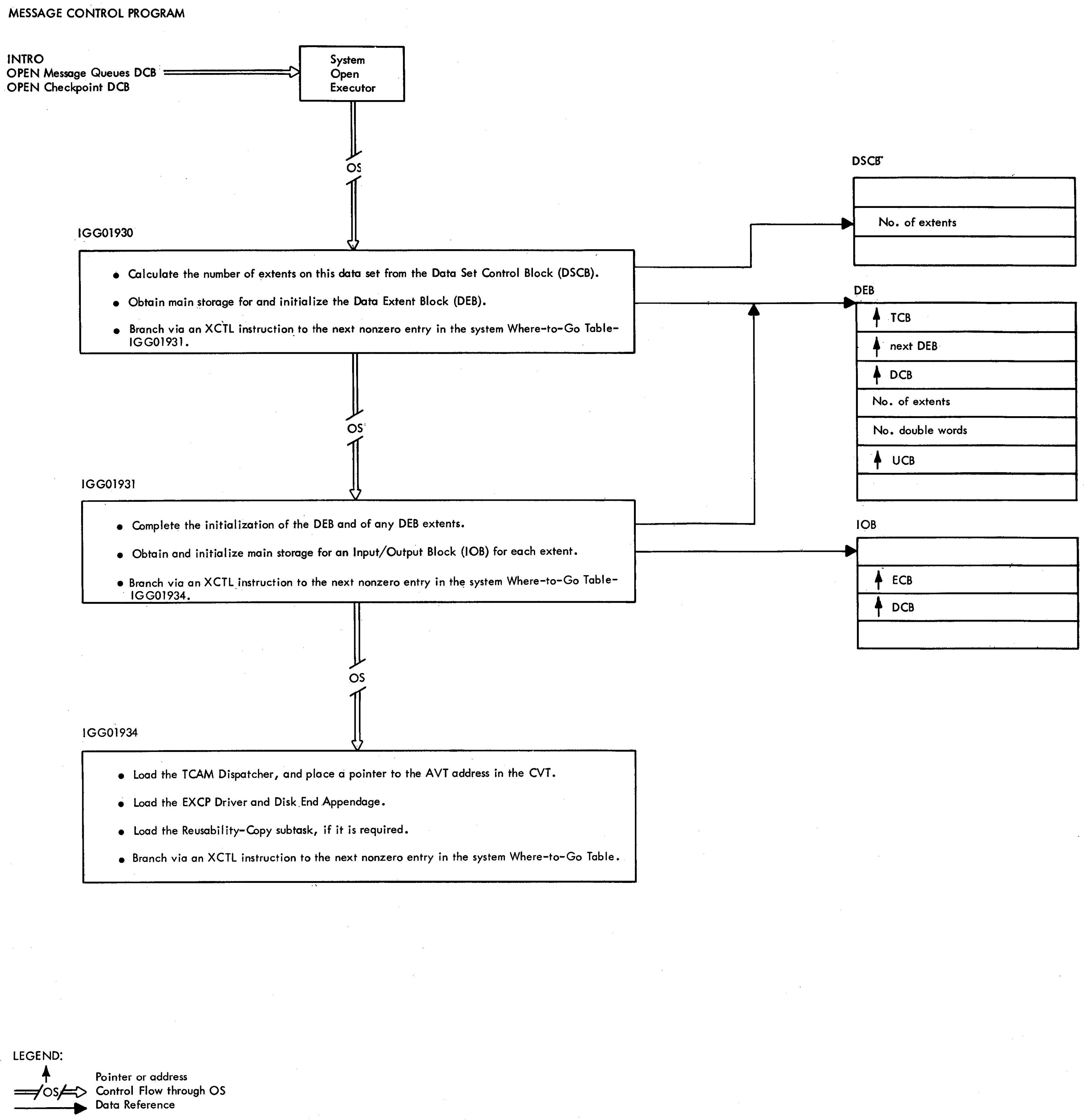
- Attach the Operator Control, On-Line Test, and FE Common Write tasks.
- Load the System Delay subtask if the system delay interval is greater than zero.
- Load the Operator Awareness Message Router if the system console is not the primary operator control terminal.

LEGEND:

Control flow through the Operating System (OS)

Control flow through a branch or sequential instructions

Chart 1. Initialization of a Message Control Program



IGG01930

- Calculate the number of extents on this data set from the Data Set Control Block (DSCB).
- Obtain main storage for and initialize the Data Extent Block (DEB).
- Branch via an XCTL instruction to the next nonzero entry in the system Where-to-Go Table-IGG01931.

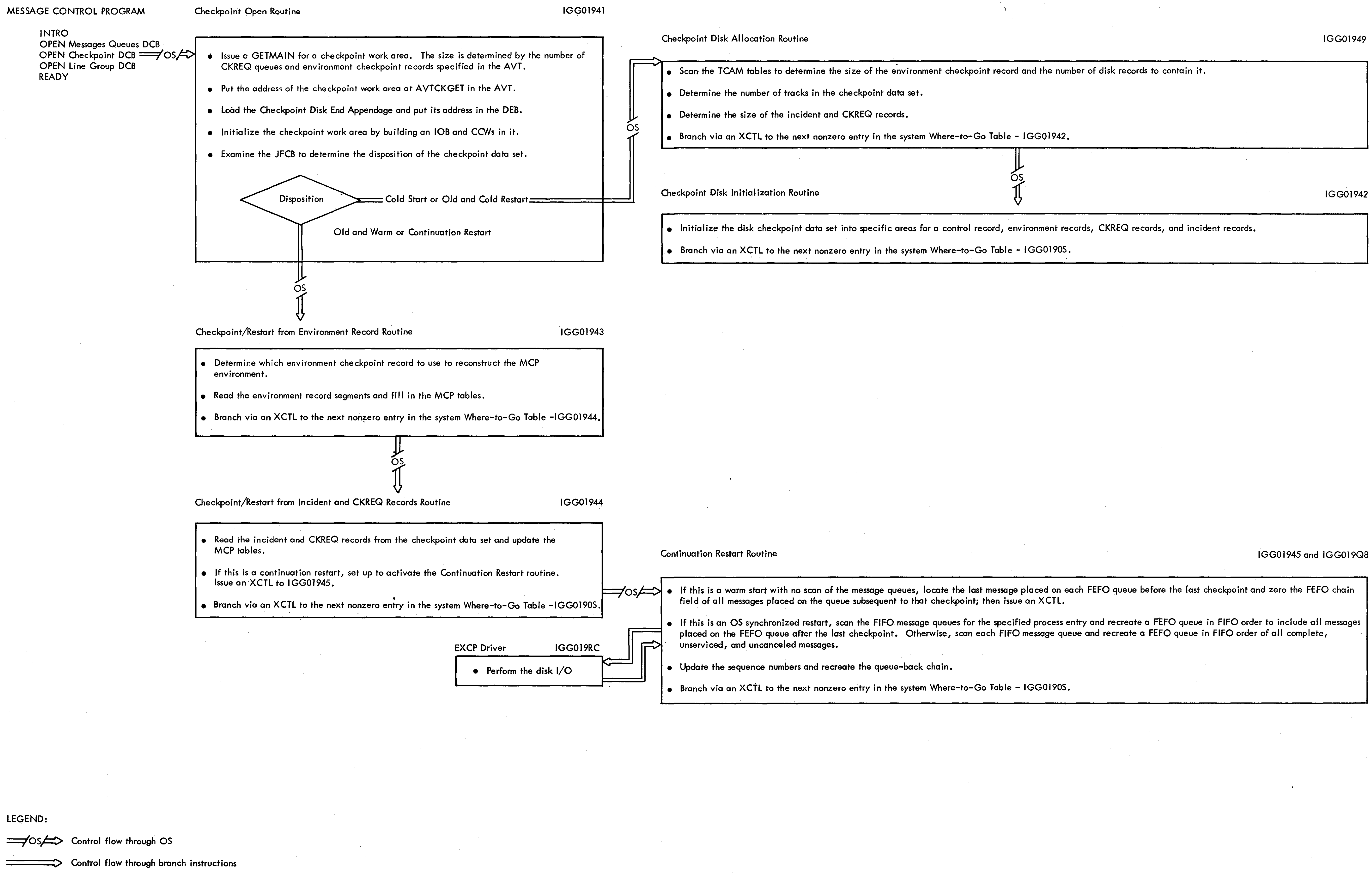
IGG01931

- Complete the initialization of the DEB and of any DEB extents.
- Obtain and initialize main storage for an Input/Output Block (IOB) for each extent.
- Branch via an XCTL instruction to the next nonzero entry in the system Where-to-Go Table-IGG01934.

IGG01934

- Load the TCAM Dispatcher, and place a pointer to the AVT address in the CVT.
- Load the EXCP Driver and Disk End Appendage.
- Load the Reusability-Copy subtask, if it is required.
- Branch via an XCTL instruction to the next nonzero entry in the system Where-to-Go Table.

Chart 2. Functions of Open Disk Message Queues Data Set



MESSAGE CONTROL PROGRAM

INTRO
OPEN Messages Queues DCB
OPEN Checkpoint DCB
OPEN Line Group DCB
READY

Checkpoint Open Routine IGG01941

- Issue a GETMAIN for a checkpoint work area. The size is determined by the number of CKREQ queues and environment checkpoint records specified in the AVT.
- Put the address of the checkpoint work area at AVTCKGET in the AVT.
- Load the Checkpoint Disk End Appendage and put its address in the DEB.
- Initialize the checkpoint work area by building an IOB and CCWs in it.
- Examine the JFCB to determine the disposition of the checkpoint data set.

Disposition
Cold Start or Old and Cold Restart
Old and Warm or Continuation Restart
OS

Checkpoint Disk Allocation Routine IGG01949

- Scan the TCAM tables to determine the size of the environment checkpoint record and the number of disk records to contain it.
- Determine the number of tracks in the checkpoint data set.
- Determine the size of the incident and CKREQ records.
- Branch via an XCTL to the next nonzero entry in the system Where-to-Go Table - IGG01942.

Checkpoint Disk Initialization Routine IGG01942

- Initialize the disk checkpoint data set into specific areas for a control record, environment records, CKREQ records, and incident records.
- Branch via an XCTL to the next nonzero entry in the system Where-to-Go Table - IGG0190S.

Checkpoint/Restart from Environment Record Routine IGG01943

- Determine which environment checkpoint record to use to reconstruct the MCP environment.
- Read the environment record segments and fill in the MCP tables.
- Branch via an XCTL to the next nonzero entry in the system Where-to-Go Table -IGG01944.

Checkpoint/Restart from Incident and CKREQ Records Routine IGG01944

- Read the incident and CKREQ records from the checkpoint data set and update the MCP tables.
- If this is a continuation restart, set up to activate the Continuation Restart routine. Issue an XCTL to IGG01945.
- Branch via an XCTL to the next nonzero entry in the system Where-to-Go Table -IGG0190S.

EXCP Driver IGG019RC

- Perform the disk I/O

Continuation Restart Routine IGG01945 and IGG019Q8

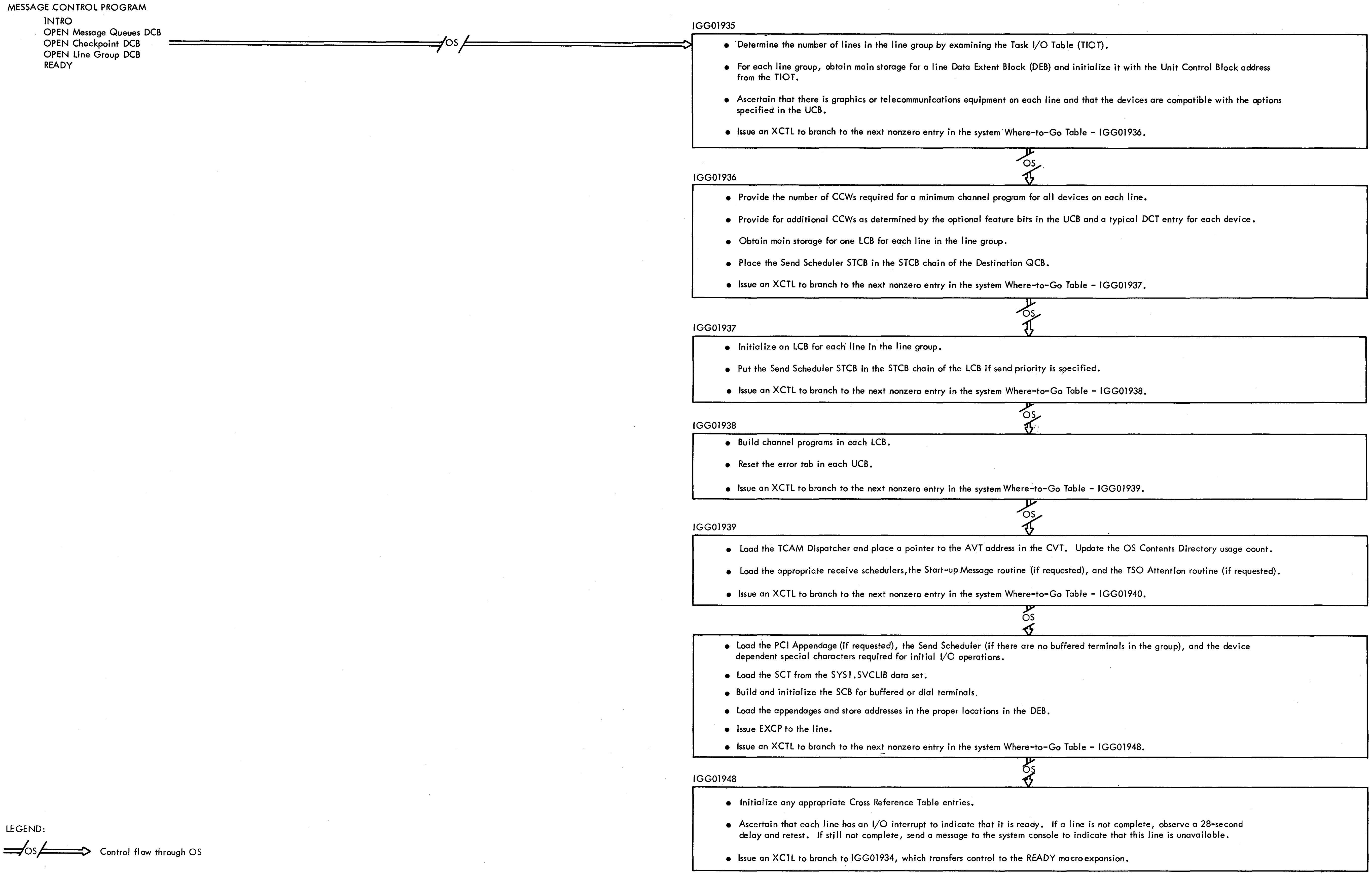
- If this is a warm start with no scan of the message queues, locate the last message placed on each FEFO queue before the last checkpoint and zero the FEFO chain field of all messages placed on the queue subsequent to that checkpoint; then issue an XCTL.
- If this is an OS synchronized restart, scan the FIFO message queues for the specified process entry and recreate a FEFO queue in FIFO order to include all messages placed on the FEFO queue after the last checkpoint. Otherwise, scan each FIFO message queue and recreate a FEFO queue in FIFO order of all complete, unserviced, and uncanceled messages.
- Update the sequence numbers and recreate the queue-back chain.
- Branch via an XCTL to the next nonzero entry in the system Where-to-Go Table - IGG0190S.

LEGEND:

Control flow through OS

Control flow through branch instructions

Chart 3. Functions of Checkpoint Open



MESSAGE CONTROL PROGRAM

INTRO
OPEN Message Queues DCB
OPEN Checkpoint DCB
OPEN Line Group DCB
READY

IGG01935

- Determine the number of lines in the line group by examining the Task I/O Table (TIOT).
- For each line group, obtain main storage for a line Data Extent Block (DEB) and initialize it with the Unit Control Block address from the TIOT.
- Ascertain that there is graphics or telecommunications equipment on each line and that the devices are compatible with the options specified in the UCB.
- Issue an XCTL to branch to the next nonzero entry in the system Where-to-Go Table - IGG01936.

IGG01936

- Provide the number of CCWs required for a minimum channel program for all devices on each line.
- Provide for additional CCWs as determined by the optional feature bits in the UCB and a typical DCT entry for each device.
- Obtain main storage for one LCB for each line in the line group.
- Place the Send Scheduler STCB in the STCB chain of the Destination QCB.
- Issue an XCTL to branch to the next nonzero entry in the system Where-to-Go Table - IGG01937.

IGG01937

- Initialize an LCB for each line in the line group.
- Put the Send Scheduler STCB in the STCB chain of the LCB if send priority is specified.
- Issue an XCTL to branch to the next nonzero entry in the system Where-to-Go Table - IGG01938.

IGG01938

- Build channel programs in each LCB.
- Reset the error tab in each UCB.
- Issue an XCTL to branch to the next nonzero entry in the system Where-to-Go Table - IGG01939.

IGG01939

- Load the TCAM Dispatcher and place a pointer to the AVT address in the CVT. Update the OS Contents Directory usage count.
- Load the appropriate receive schedulers, the Start-up Message routine (if requested), and the TSO Attention routine (if requested).
- Issue an XCTL to branch to the next nonzero entry in the system Where-to-Go Table - IGG01940.

- Load the PCI Appendage (if requested), the Send Scheduler (if there are no buffered terminals in the group), and the device dependent special characters required for initial I/O operations.
- Load the SCT from the SYS1.SVCLIB data set.
- Build and initialize the SCB for buffered or dial terminals.
- Load the appendages and store addresses in the proper locations in the DEB.
- Issue EXCP to the line.
- Issue an XCTL to branch to the next nonzero entry in the system Where-to-Go Table - IGG01948.

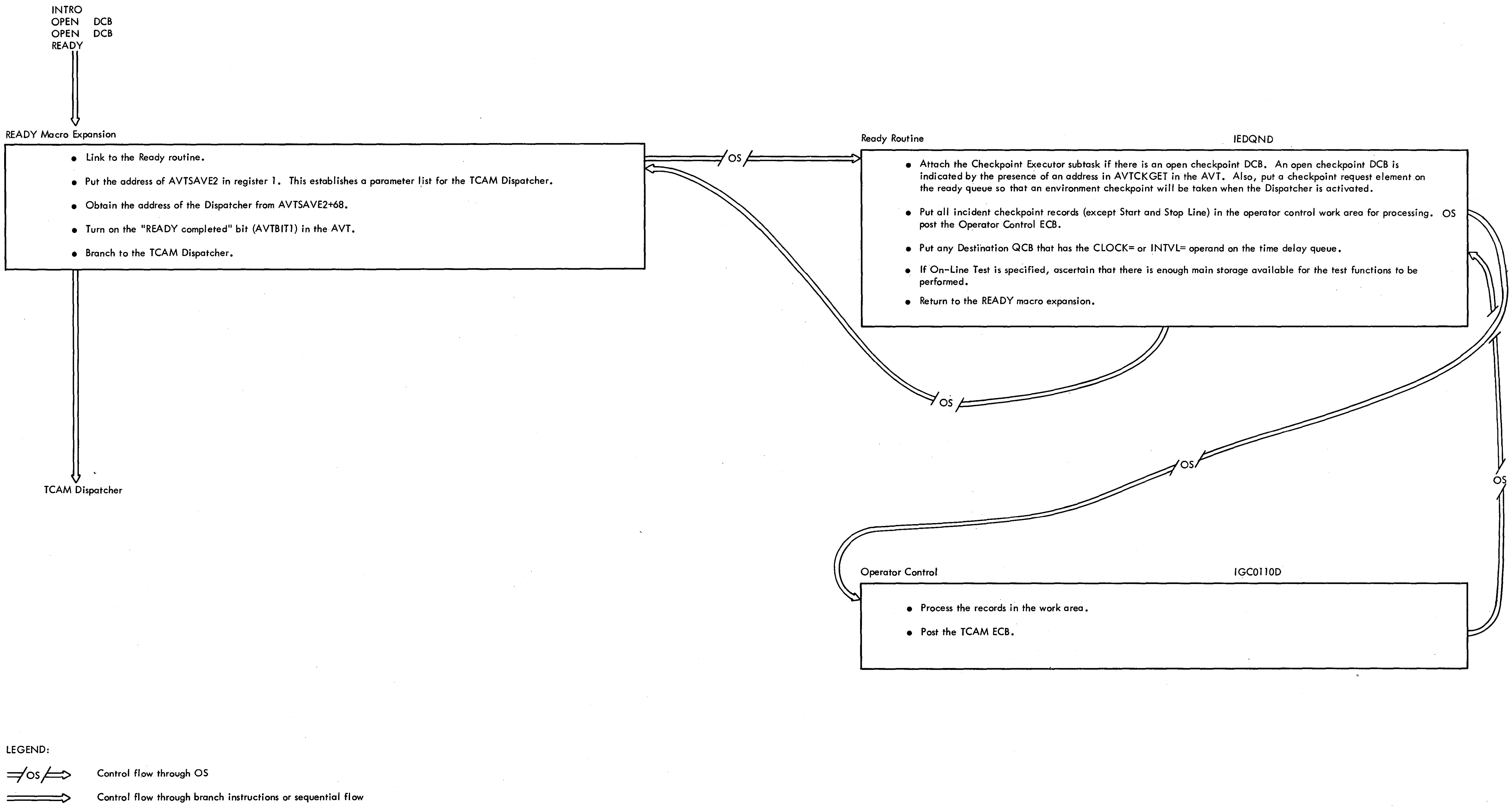
IGG01948

- Initialize any appropriate Cross Reference Table entries.
- Ascertain that each line has an I/O interrupt to indicate that it is ready. If a line is not complete, observe a 28-second delay and retest. If still not complete, send a message to the system console to indicate that this line is unavailable.
- Issue an XCTL to branch to IGG01934, which transfers control to the READY macro expansion.

LEGEND:

OS — Control flow through OS

Chart 4. Functions of Line Group Open



READY Macro Expansion

- Link to the Ready routine.
- Put the address of AVTSAVE2 in register 1. This establishes a parameter list for the TCAM Dispatcher.
- Obtain the address of the Dispatcher from AVTSAVE2+68.
- Turn on the "READY completed" bit (AVTBIT1) in the AVT.
- Branch to the TCAM Dispatcher.

Ready Routine IEDQND

- Attach the Checkpoint Executor subtask if there is an open checkpoint DCB. An open checkpoint DCB is indicated by the presence of an address in AVTCKGET in the AVT. Also, put a checkpoint request element on the ready queue so that an environment checkpoint will be taken when the Dispatcher is activated.
- Put all incident checkpoint records (except Start and Stop Line) in the operator control work area for processing. OS post the Operator Control ECB.
- Put any Destination QCB that has the CLOCK= or INTVL= operand on the time delay queue.
- If On-Line Test is specified, ascertain that there is enough main storage available for the test functions to be performed.
- Return to the READY macro expansion.

Operator Control IGC0110D

- Process the records in the work area.
- Post the TCAM ECB.

LEGEND:

OS Control flow through OS

Control flow through branch instructions or sequential flow

Chart 5. Functions of the READY Macro Expansion and Routine

Subtask A

- Perform a tpost by putting the address of the new QCB in the RCB of the element.
- Branch to the TCAM Dispatcher at DSPPOST.

TCAM Dispatcher

- Put the address of the element into the ready queue chain by priority.
- Merge the disabled onto the enabled ready queue by priority, if there are any elements on the disabled ready queue.
- Remove Element X from the ready queue by placing its address in register 1.
- Update the ready queue by placing the link field of Element X in the ready queue.
- Examine the element pointed to by register 1.
  a. If the QCB points to an STCB that has an MCPL of X'00', there is no work to be done; therefore, issue an OS WAIT.
  b. If the element represents an attached task (MCPL = X'02'), post the ECB for that task complete so that the task can vie for control when a TCAM WAIT is issued.
  c. If the MCPL is neither X'00' nor X'02', get the entry point for the highest priority subtask represented on the STCB chain of the QCB. Branch to that subtask.

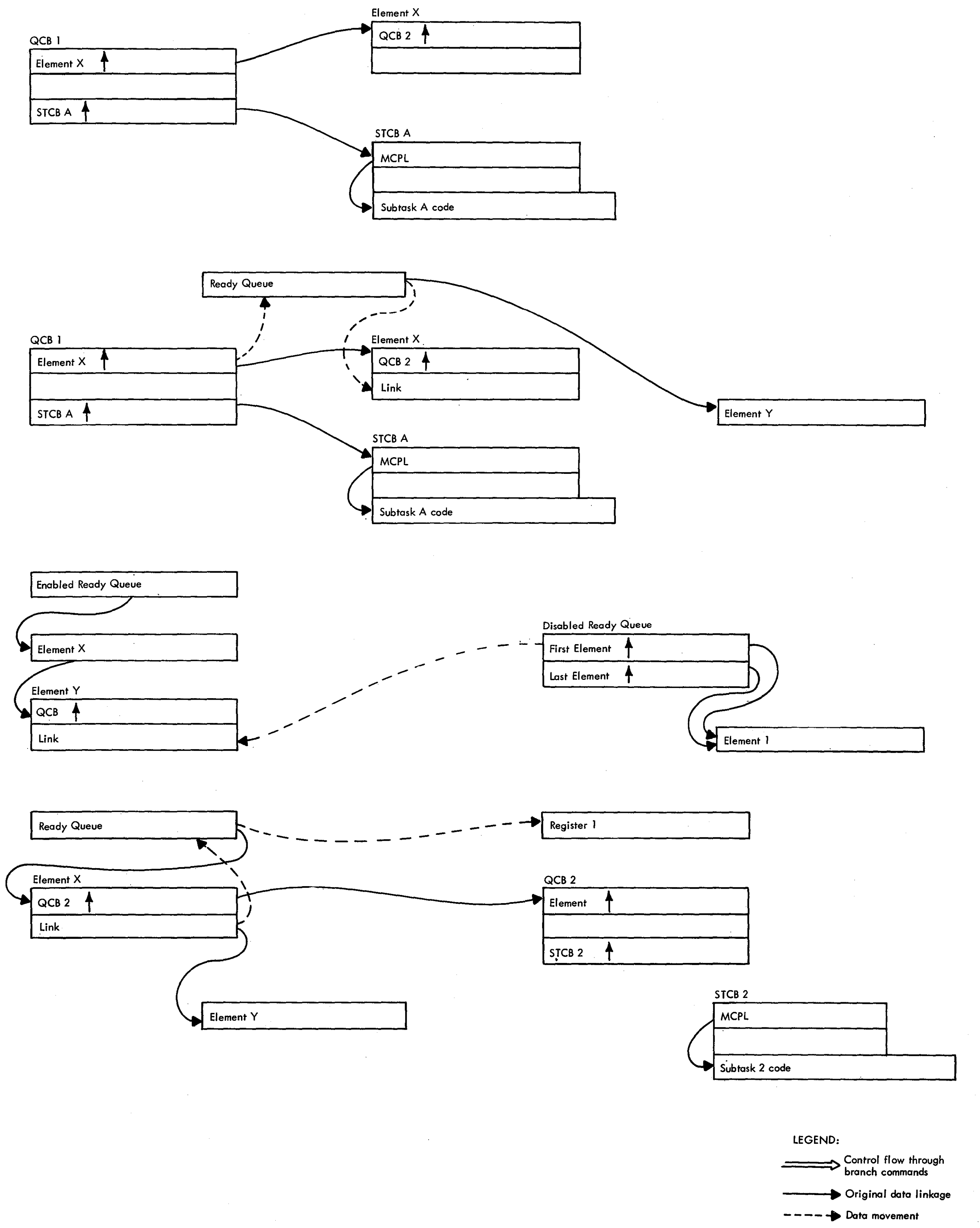
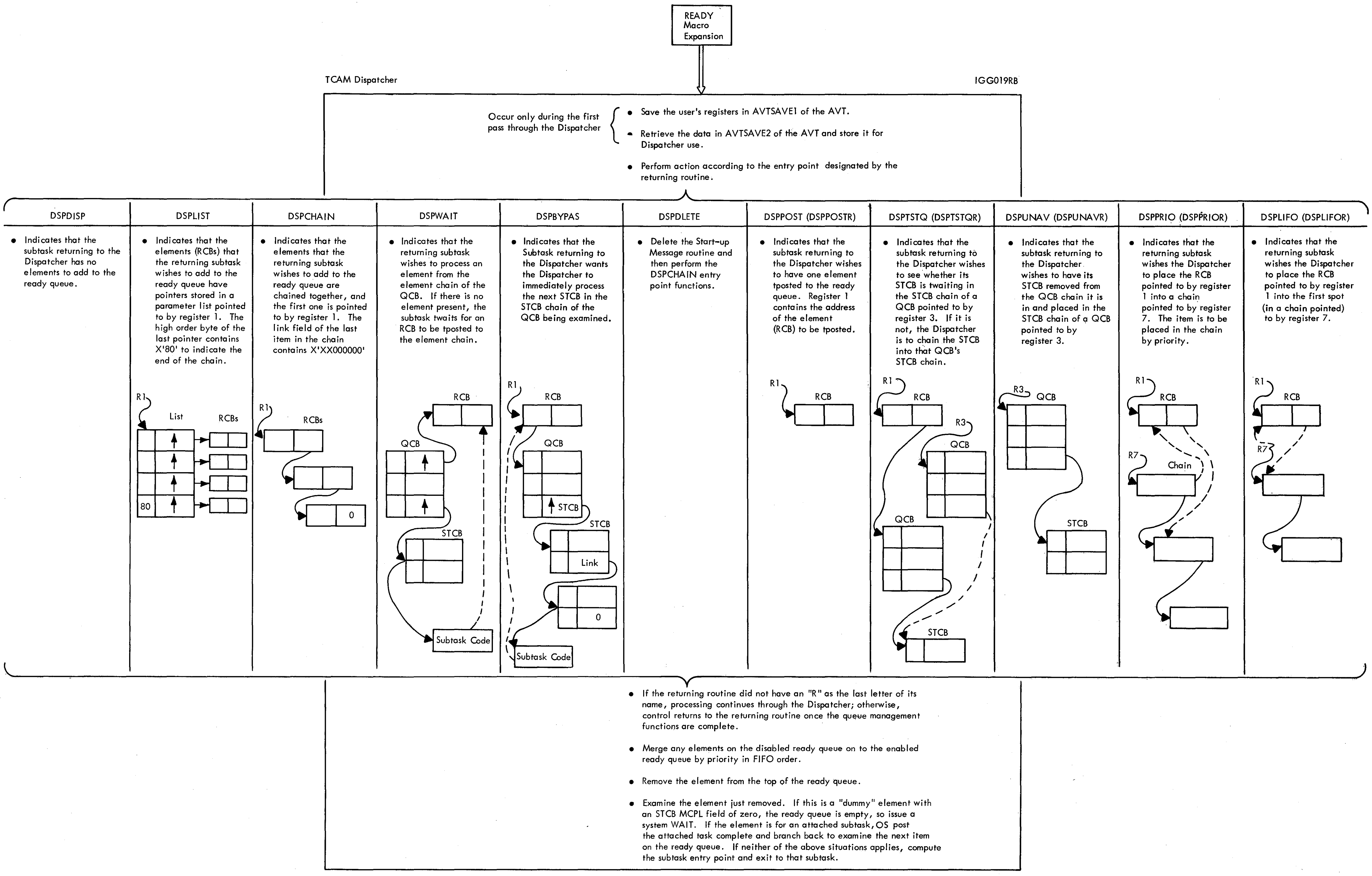


Chart 6. Summary of the Dispatching Functions of the TCAM Dispatcher



TCAM Dispatcher IGG019RB

Occur only during the first pass through the Dispatcher:

- Save the user's registers in AVTSAVE1 of the AVT.
- Retrieve the data in AVTSAVE2 of the AVT and store it for Dispatcher use.

- Perform action according to the entry point designated by the returning routine.

DSPDISP

- Indicates that the subtask returning to the Dispatcher has no elements to add to the ready queue.

DSPLIST

- Indicates that the elements (RCBs) that the returning subtask wishes to add to the ready queue have pointers stored in a parameter list pointed to by register 1. The high order byte of the last pointer contains X'80' to indicate the end of the chain.

DSPCHAIN

- Indicates that the elements that the returning subtask wishes to add to the ready queue are chained together, and the first one is pointed to by register 1. The link field of the last item in the chain contains X'XX000000'

DSPWAIT

- Indicates that the returning subtask wishes to process an element from the element chain of the QCB. If there is no element present, the subtask twaits for an RCB to be tposted to the element chain.

DSPBYPAS

- Indicates that the Subtask returning to the Dispatcher wants the Dispatcher to immediately process the next STCB in the STCB chain of the QCB being examined.

DSPDLETE

- Delete the Start-up Message routine and then perform the DSPCHAIN entry point functions.

DSPPOST (DSPPOSTR)

- Indicates that the subtask returning to the Dispatcher wishes to have one element tposted to the ready queue. Register 1 contains the address of the element (RCB) to be tposted.

DSPTSTQ (DSPTSTQR)

- Indicates that the subtask returning tö the Dispatcher wishes to see whether its STCB is twaiting in the STCB chain of a QCB pointed to by register 3. If it is not, the Dispatcher is to chain the STCB into that QCB's STCB chain.

DSPUNAV (DSPUNAVR)

- Indicates that the subtask returning to the Dispatcher wishes to have its STCB removed from the QCB chain it is in and placed in the STCB chain of a QCB pointed to by register 3.

DSPPRIO (DSPPRIOR)

- Indicates that the returning subtask wishes the Dispatcher to place the RCB pointed to by register 1 into a chain pointed to by register 7. The item is to be placed in the chain by priority.

DSPLIFO (DSPLIFOR)

- Indicates that the returning subtask wishes the Dispatcher to place the RCB pointed to by register 1 into the first spot (in a chain pointed) to by register 7.

- If the returning routine did not have an "R" as the last letter of its name, processing continues through the Dispatcher; otherwise, control returns to the returning routine once the queue management functions are complete.
- Merge any elements on the disabled ready queue on to the enabled ready queue by priority in FIFO order.
- Remove the element from the top of the ready queue.
- Examine the element just removed. If this is a "dummy" element with an STCB MCPL field of zero, the ready queue is empty, so issue a system WAIT. If the element is for an attached subtask, OS post the attached task complete and branch back to examine the next item on the ready queue. If neither of the above situations applies, compute the subtask entry point and exit to that subtask.

Chart 7. Summary of the Queuing Functions of the TCAM Dispatcher

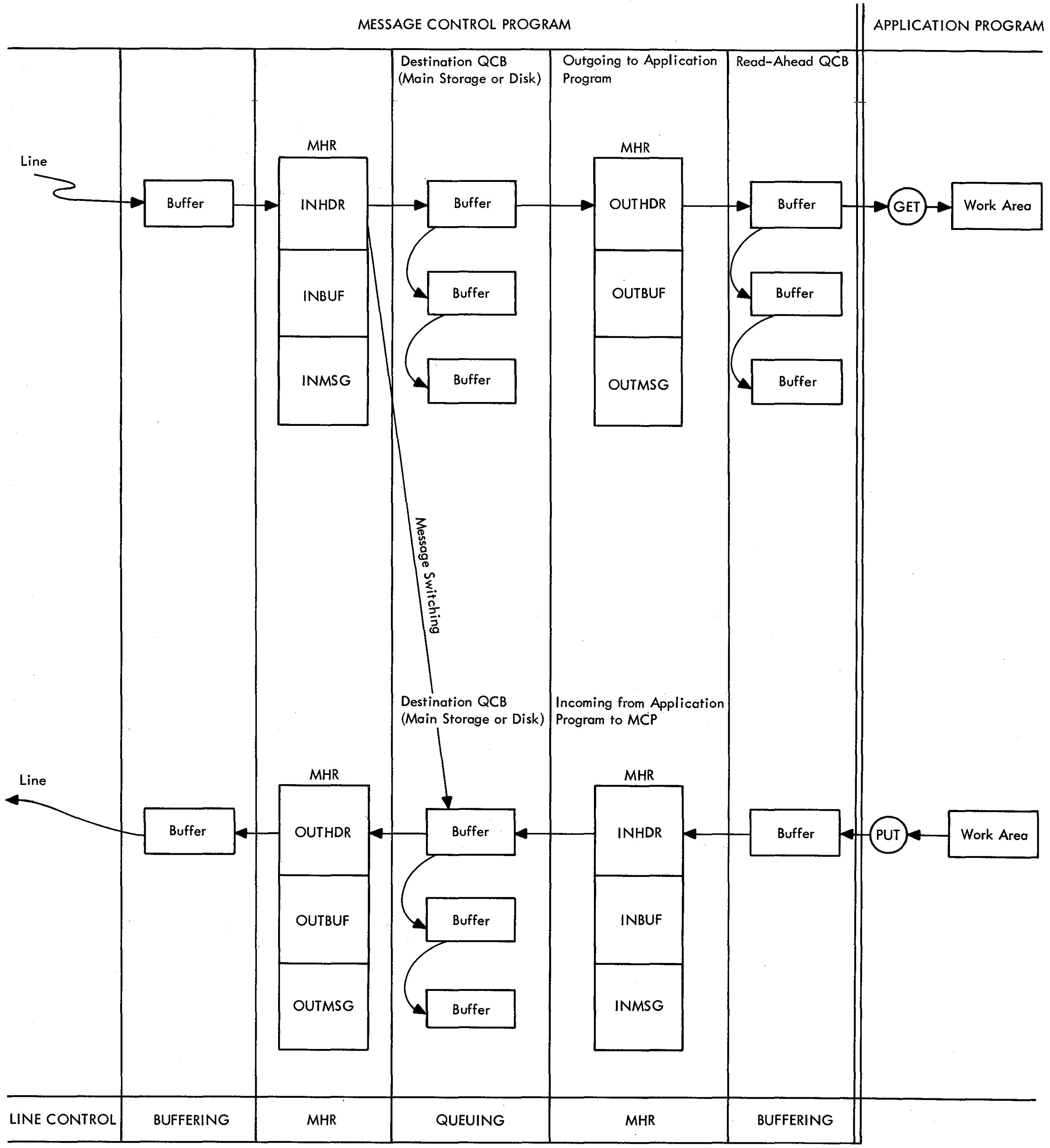
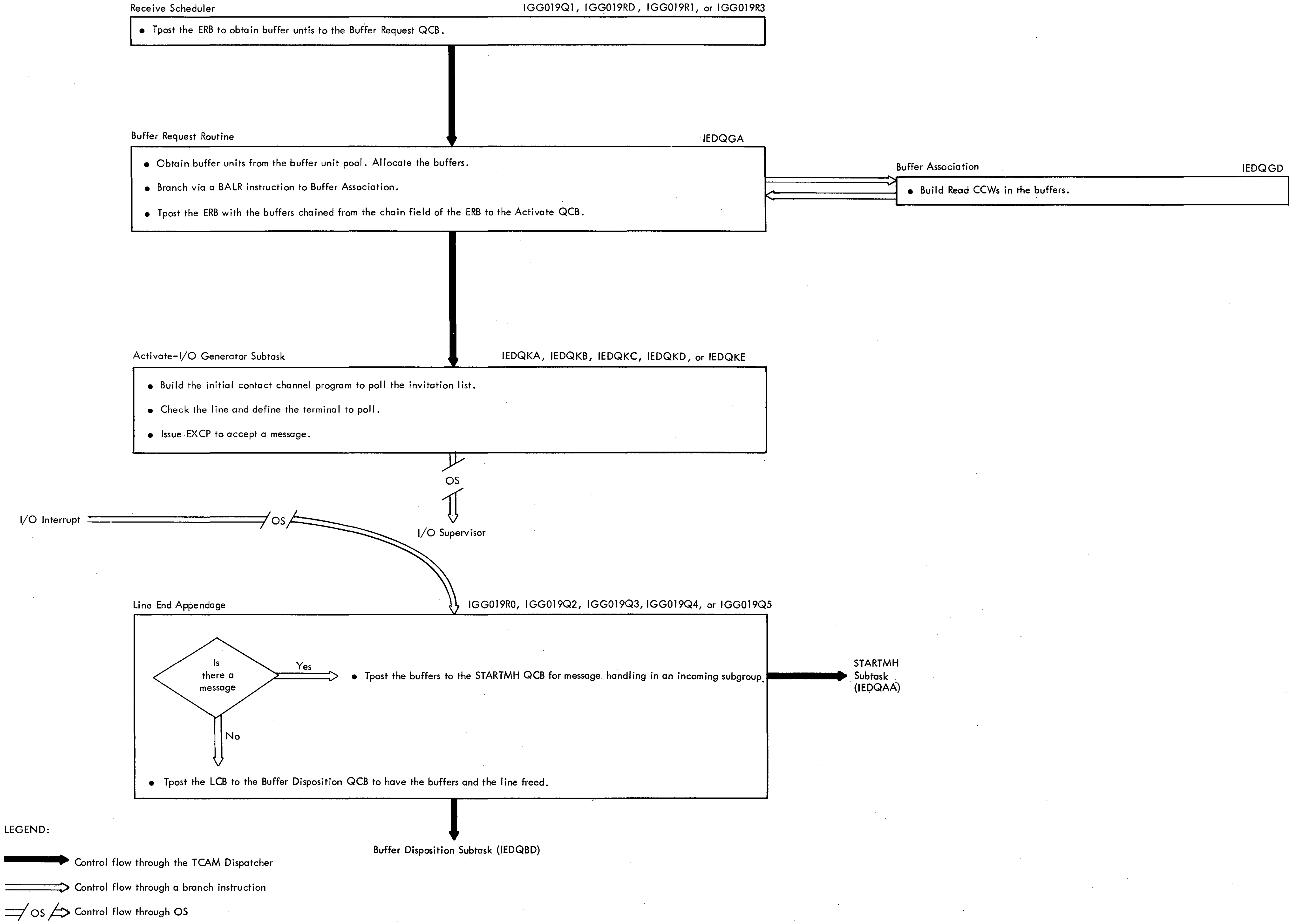


Chart 8. TCAM Message Flow



Receive Scheduler IGG019Q1, IGG019RD, IGG019R1, or IGG019R3

- Tpost the ERB to obtain buffer untis to the Buffer Request QCB.

Buffer Request Routine IEDQGA

- Obtain buffer units from the buffer unit pool. Allocate the buffers.
- Branch via a BALR instruction to Buffer Association.
- Tpost the ERB with the buffers chained from the chain field of the ERB to the Activate QCB.

Buffer Association IEDQGD

- Build Read CCWs in the buffers.

Activate-I/O Generator Subtask IEDQKA, IEDQKB, IEDQKC, IEDQKD, or IEDQKE

- Build the initial contact channel program to poll the invitation list.
- Check the line and define the terminal to poll.
- Issue EXCP to accept a message.

Line End Appendage IGG019R0, IGG019Q2, IGG019Q3, IGG019Q4, or IGG019Q5

- Tpost the buffers to the STARTMH QCB for message handling in an incoming subgroup.
- Tpost the LCB to the Buffer Disposition QCB to have the buffers and the line freed.

LEGEND:

Control flow through the TCAM Dispatcher
Control flow through a branch instruction
OS
Control flow through OS

Chart 9. Functional Flow in a Receive Operation

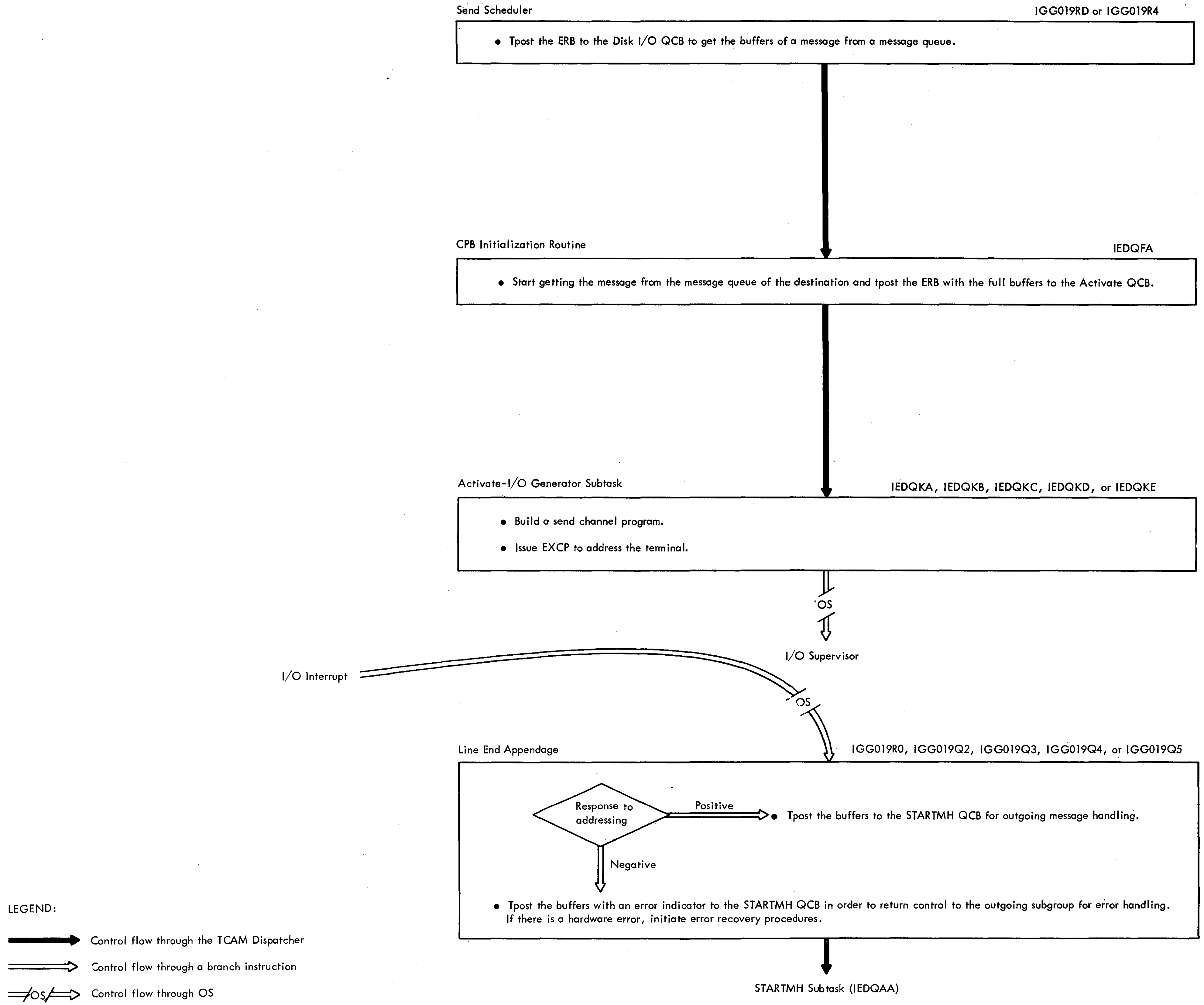
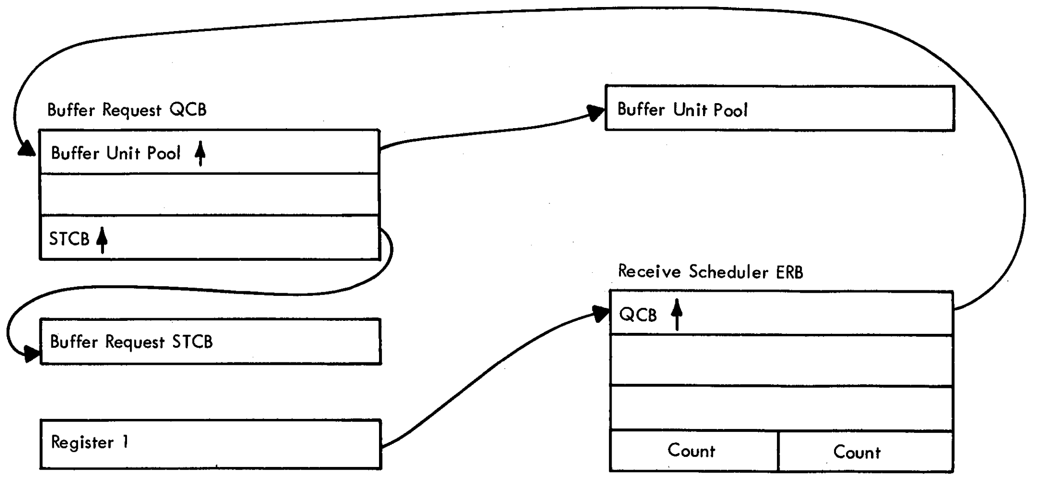

Send Scheduler
IGG019RD or IGG019R4
• Tpost the ERB to the Disk I/O QCB to get the buffers of a message from a message queue.
CPB Initialization Routine
IEDQFA
• Start getting the message from the message queue of the destination and tpost the ERB with the full buffers to the Activate QCB.
Activate-I/O Generator Subtask
IEDQKA, IEDQKB, IEDQKC, IEDQKD, or IEDQKE
• Build a send channel program.
• Issue EXCP to address the terminal.
OS
I/O Supervisor
I/O Interrupt
OS
Line End Appendage
IGG019R0, IGG019Q2, IGG019Q3, IGG019Q4, or IGG019Q5
Response to addressing
Positive
• Tpost the buffers to the STARTMH QCB for outgoing message handling.
Negative
• Tpost the buffers with an error indicator to the STARTMH QCB in order to return control to the outgoing subgroup for error handling. If there is a hardware error, initiate error recovery procedures.
STARTMH Subtask (IEDQAA)
LEGEND:
Control flow through the TCAM Dispatcher
Control flow through a branch instruction
OS
Control flow through OS

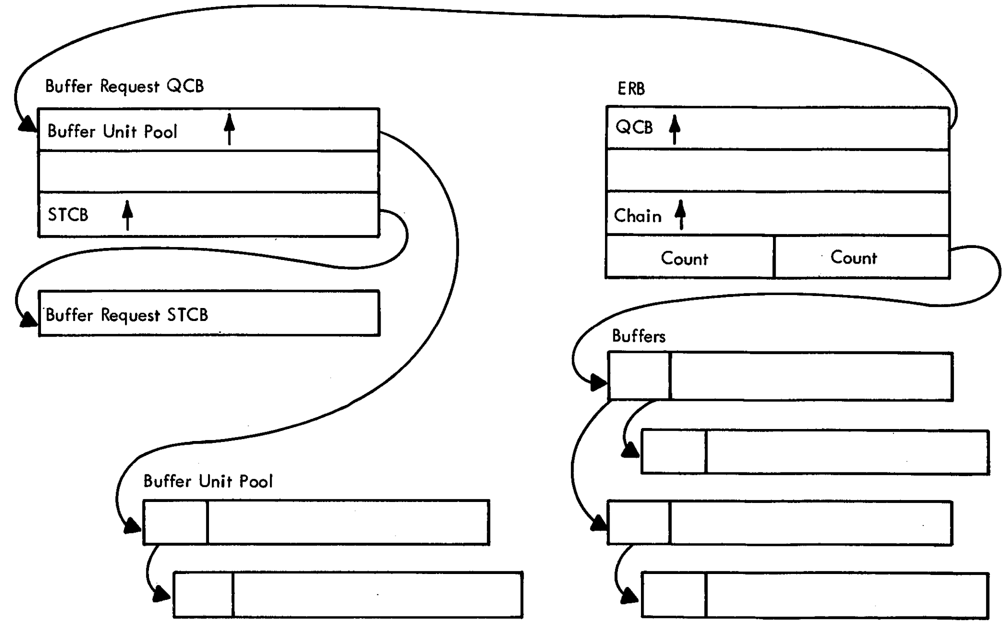
Chart 10. Functional Flow in a Send Operation

Line Group Receive Operation

Receive Scheduler

- Get the ERB from the LCB.
- Tpost the ERB to the Buffer Request QCB.



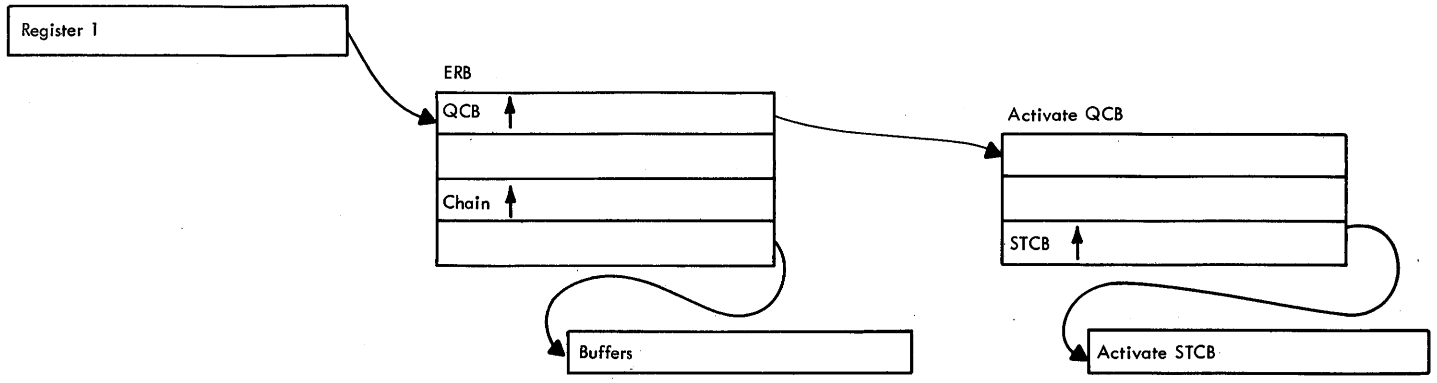


Buffer Request Routine IEDQGA

- Check the DCB to get the number of units in one buffer.
- Get the units for the required number of buffers from the buffer unit pool. (If the units are not available, chain the ERB by priority into the element chain of the Buffer Return QCB and exit. Otherwise, continue.)
- Chain the units off the chain pointer in the third word of the ERB.
- Reconstruct the buffer unit pool chain. (See Figure 25.)
- Branch via a BAL instruction to Buffer Association.
- Tpost the ERB to the Activate QCB. This is notification that the request has been satisfied and the system is ready to receive.

Buffer Association IEDQGD

- Build a Read CCW and a TIC in each 12-byte unit control area and chain the units together by the TIC commands. (See Figure 26.)
- Return via a BR 14 instruction.



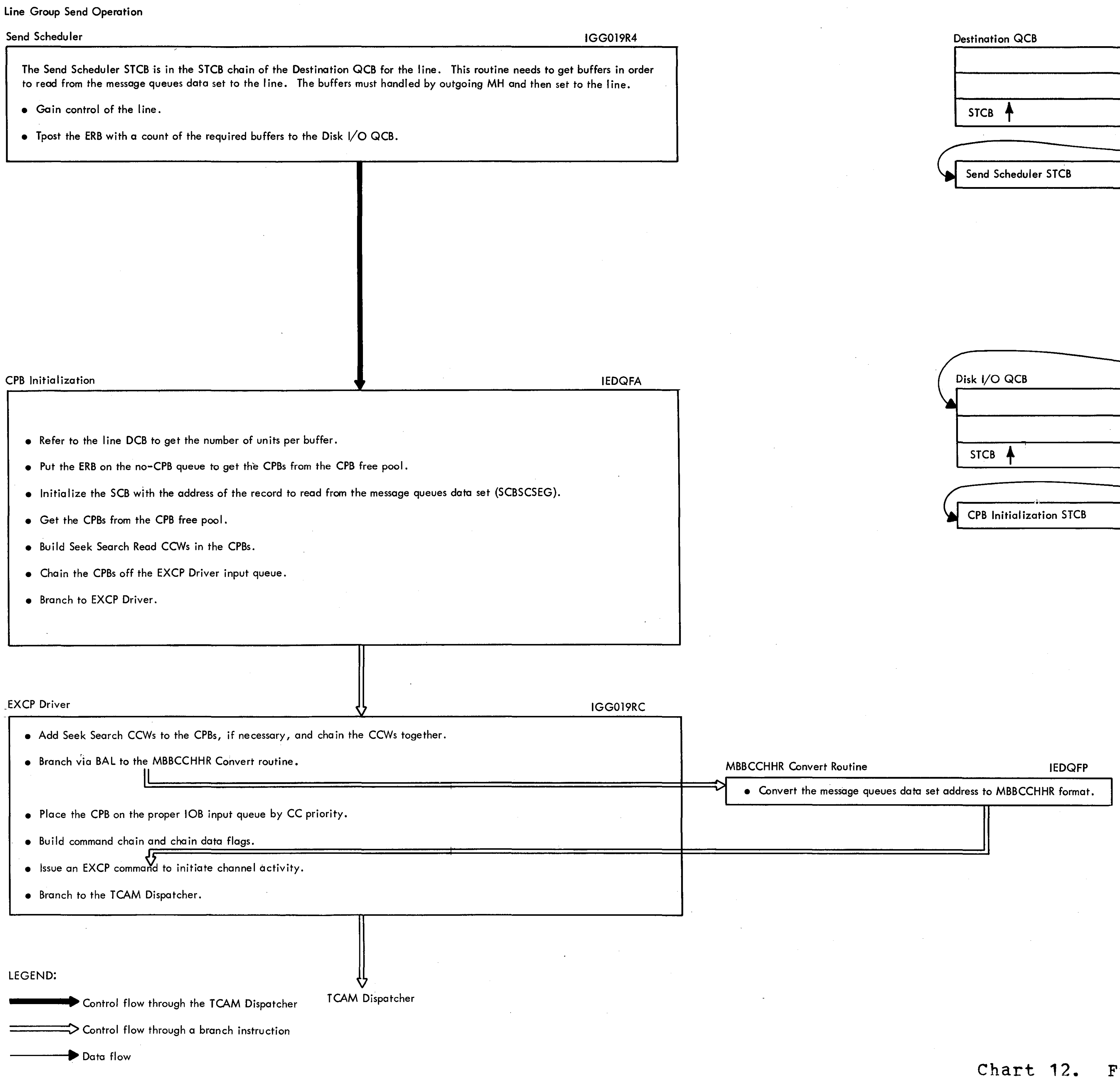
Activate Subtask IEDQKA

- Establish initial contact with the line for a receive operation.

LEGEND:

➡ Control flow through the TCAM Dispatcher
⇨ Control flow through a branch instruction
→ Data flow

Chart 11. Functions of an Initial Buffer Request in a Receive Operation



Line Group Send Operation

Send Scheduler IGG019R4

The Send Scheduler STCB is in the STCB chain of the Destination QCB for the line. This routine needs to get buffers in order to read from the message queues data set to the line. The buffers must handled by outgoing MH and then set to the line.

- Gain control of the line.
- Tpost the ERB with a count of the required buffers to the Disk I/O QCB.

CPB Initialization IEDQFA

- Refer to the line DCB to get the number of units per buffer.
- Put the ERB on the no-CPB queue to get the CPBs from the CPB free pool.
- Initialize the SCB with the address of the record to read from the message queues data set (SCBSCSEG).
- Get the CPBs from the CPB free pool.
- Build Seek Search Read CCWs in the CPBs.
- Chain the CPBs off the EXCP Driver input queue.
- Branch to EXCP Driver.

EXCP Driver IGG019RC

- Add Seek Search CCWs to the CPBs, if necessary, and chain the CCWs together.
- Branch via BAL to the MBBCCHHR Convert routine.
- Place the CPB on the proper IOB input queue by CC priority.
- Build command chain and chain data flags.
- Issue an EXCP command to initiate channel activity.
- Branch to the TCAM Dispatcher.

MBBCCHHR Convert Routine IEDQFP

- Convert the message queues data set address to MBBCCHHR format.

TCAM Dispatcher

LEGEND:

Control flow through the TCAM Dispatcher
Control flow through a branch instruction
Data flow

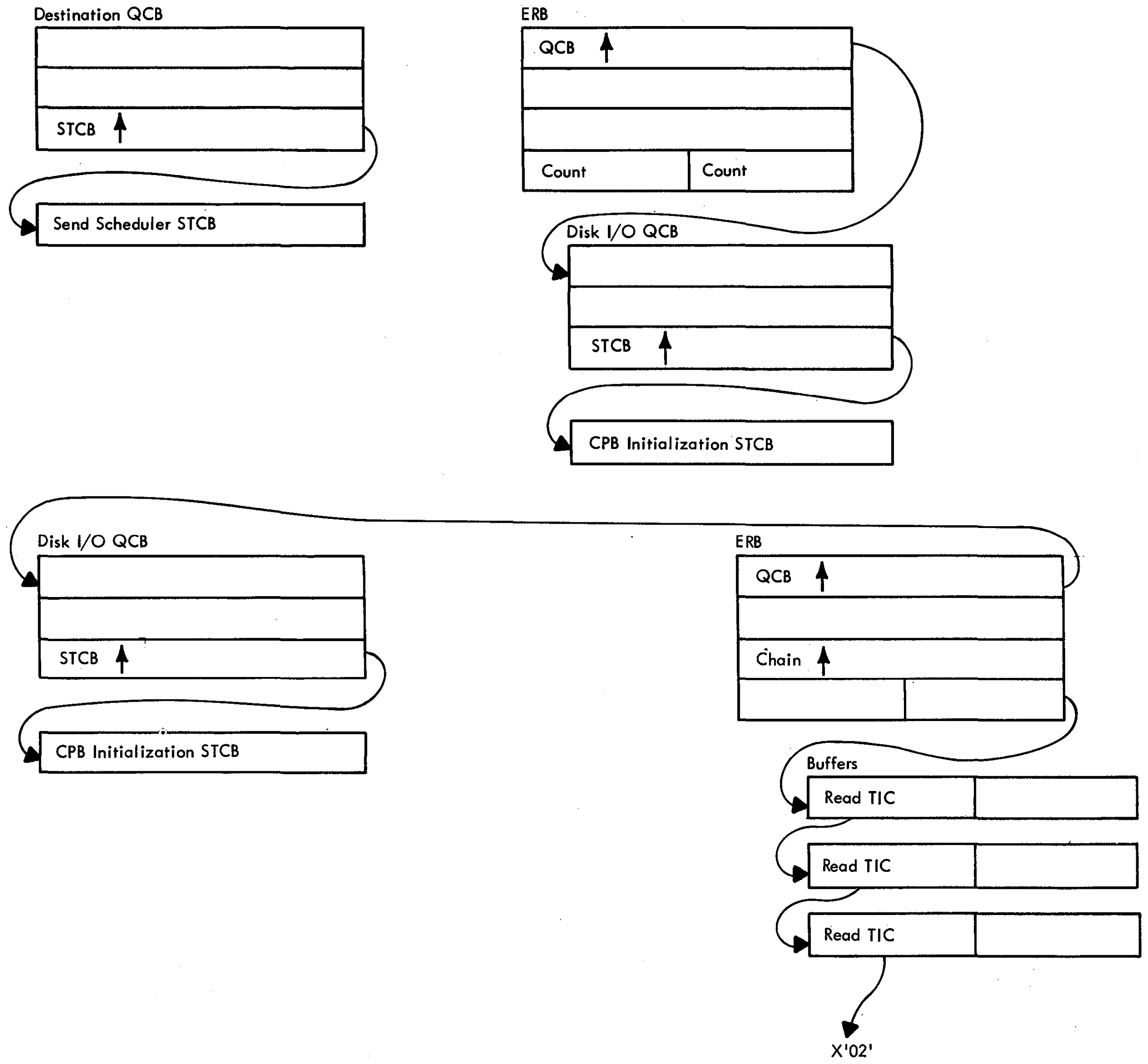
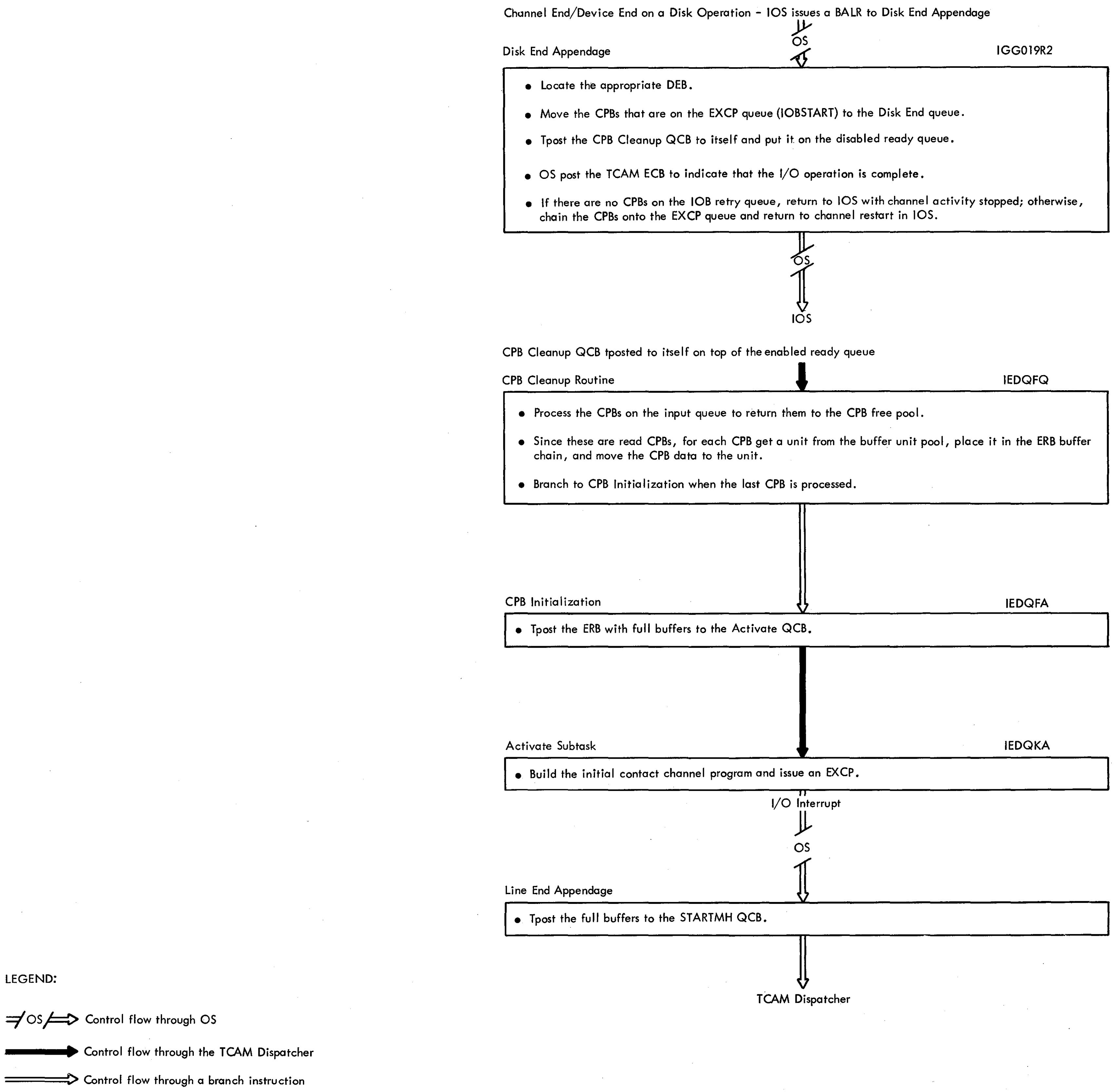


Chart 12. Functions of an Initial Buffer Request in a Send Operation (Part 1 of 2)



Channel End/Device End on a Disk Operation - IOS issues a BALR to Disk End Appendage

OS

Disk End Appendage IGG019R2

- Locate the appropriate DEB.
- Move the CPBs that are on the EXCP queue (IOBSTART) to the Disk End queue.
- Tpost the CPB Cleanup QCB to itself and put it on the disabled ready queue.
- OS post the TCAM ECB to indicate that the I/O operation is complete.
- If there are no CPBs on the IOB retry queue, return to IOS with channel activity stopped; otherwise, chain the CPBs onto the EXCP queue and return to channel restart in IOS.

OS

IOS

CPB Cleanup QCB tposted to itself on top of the enabled ready queue

CPB Cleanup Routine IEDQFQ

- Process the CPBs on the input queue to return them to the CPB free pool.
- Since these are read CPBs, for each CPB get a unit from the buffer unit pool, place it in the ERB buffer chain, and move the CPB data to the unit.
- Branch to CPB Initialization when the last CPB is processed.

CPB Initialization IEDQFA

- Tpost the ERB with full buffers to the Activate QCB.

Activate Subtask IEDQKA

- Build the initial contact channel program and issue an EXCP.

I/O Interrupt

OS

Line End Appendage

- Tpost the full buffers to the STARTMH QCB.

TCAM Dispatcher

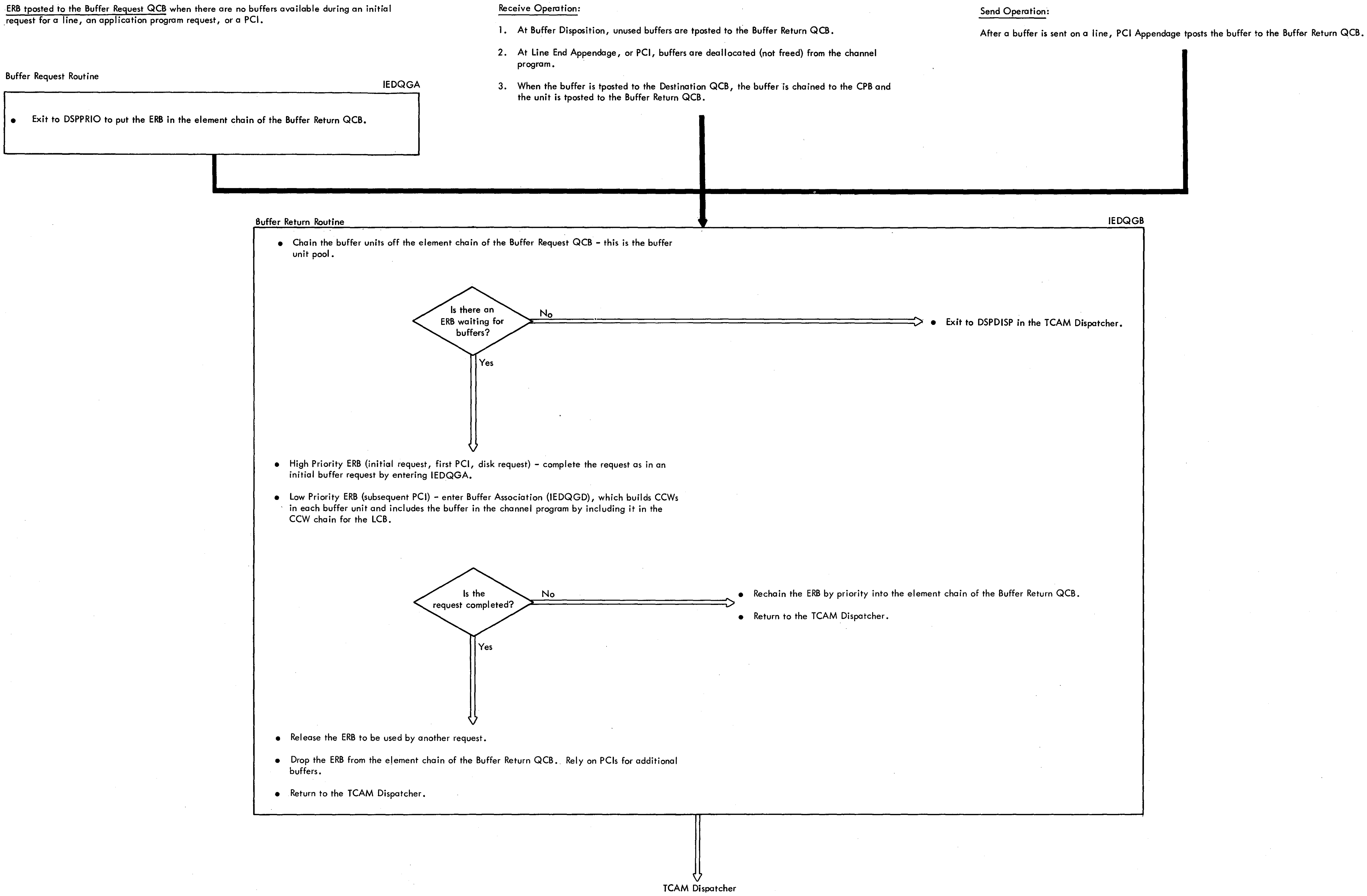
LEGEND:

OS Control flow through OS

Control flow through the TCAM Dispatcher

Control flow through a branch instruction

Chart 12. Functions of an Initial Buffer Pequest in a Send Oneration (Part 2 of 2)



ERB tposted to the Buffer Request QCB when there are no buffers available during an initial request for a line, an application program request, or a PCI.

Buffer Request Routine

IEDQGA

- Exit to DSPPRIO to put the ERB in the element chain of the Buffer Return QCB.

Receive Operation:

1. At Buffer Disposition, unused buffers are tposted to the Buffer Return QCB.
2. At Line End Appendage, or PCI, buffers are deallocated (not freed) from the channel program.
3. When the buffer is tposted to the Destination QCB, the buffer is chained to the CPB and the unit is tposted to the Buffer Return QCB.

Send Operation:

After a buffer is sent on a line, PCI Appendage tposts the buffer to the Buffer Return QCB.

Buffer Return Routine

IEDQGB

- Chain the buffer units off the element chain of the Buffer Request QCB - this is the buffer unit pool.

- Exit to DSPDISP in the TCAM Dispatcher.

- High Priority ERB (initial request, first PCI, disk request) - complete the request as in an initial buffer request by entering IEDQGA.
- Low Priority ERB (subsequent PCI) - enter Buffer Association (IEDQGD), which builds CCWs in each buffer unit and includes the buffer in the channel program by including it in the CCW chain for the LCB.

- Rechain the ERB by priority into the element chain of the Buffer Return QCB.
- Return to the TCAM Dispatcher.

- Release the ERB to be used by another request.
- Drop the ERB from the element chain of the Buffer Return QCB. Rely on PCIs for additional buffers.
- Return to the TCAM Dispatcher.

TCAM Dispatcher

Chart 13. Functions of Buffer Return

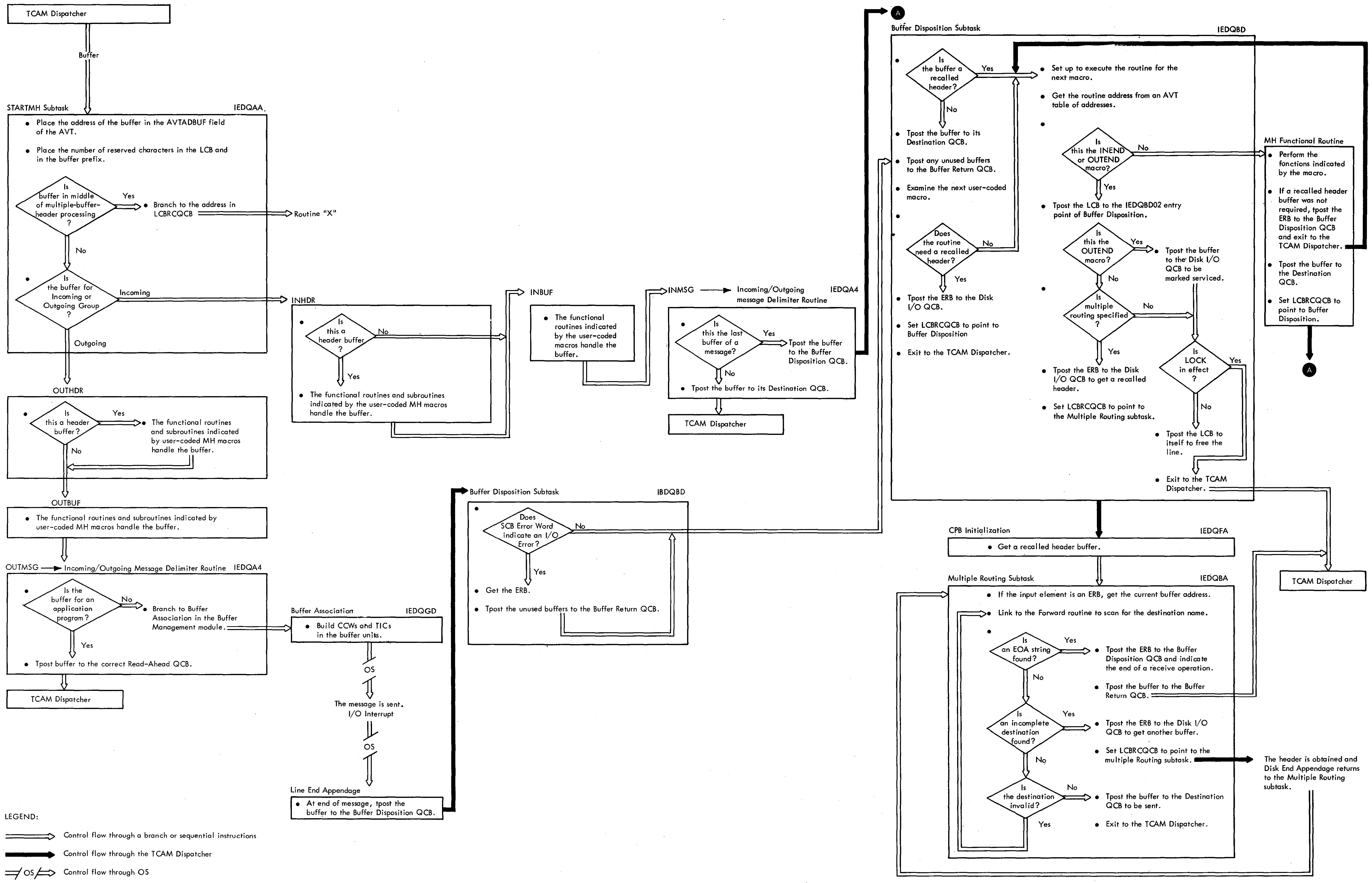
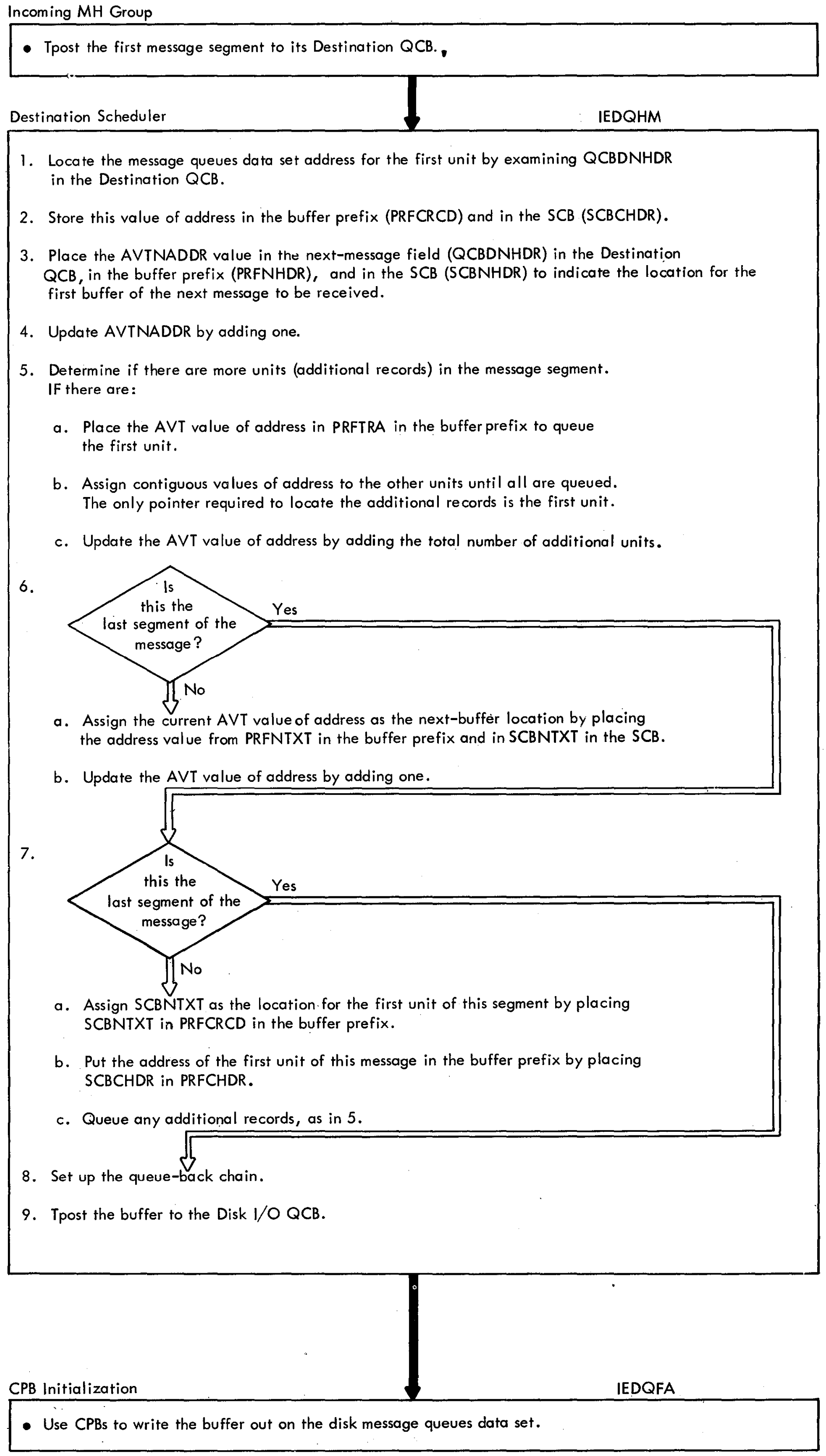

TCAM Dispatcher
Buffer
STARTMH Subtask
IEDQAA
Place the address of the buffer in the AVTADBUF field of the AVT.
Place the number of reserved characters in the LCB and in the buffer prefix.
Is buffer in middle of multiple-buffer-header processing ?
Yes
Branch to the address in LCBRCQCB
Routine "X"
No
Is the buffer for Incoming or Outgoing Group ?
Incoming
Outgoing
INHDR
Is this a header buffer ?
No
Yes
The functional routines and subroutines indicated by the user-coded MH macros handle the buffer.
INBUF
The functional routines indicated by the user-coded macros handle the buffer.
INMSG
Incoming/Outgoing message Delimiter Routine
IEDQA4
Is this the last buffer of a message?
Yes
Tpost the buffer to the Buffer Disposition QCB.
No
Tpost the buffer to its Destination QCB.
TCAM Dispatcher
OUTHDR
Is this a header buffer?
Yes
The functional routines and subroutines indicated by user-coded MH macros handle the buffer.
No
OUTBUF
The functional routines and subroutines indicated by user-coded MH macros handle the buffer.
OUTMSG
Incoming/Outgoing Message Delimiter Routine
IEDQA4
Is the buffer for an application program?
No
Branch to Buffer Association in the Buffer Management module.
Yes
Tpost buffer to the correct Read-Ahead QCB.
TCAM Dispatcher
Buffer Association
IEDQGD
Build CCWs and TICs in the buffer units.
OS
The message is sent.
I/O Interrupt
OS
Line End Appendage
At end of message, tpost the buffer to the Buffer Disposition QCB.
Buffer Disposition Subtask
IBDQBD
Does SCB Error Word indicate an I/O Error?
No
Yes
Get the ERB.
Tpost the unused buffers to the Buffer Return QCB.
A
Buffer Disposition Subtask
IEDQBD
Is the buffer a recalled header?
Yes
No
Tpost the buffer to its Destination QCB.
Tpost any unused buffers to the Buffer Return QCB.
Examine the next user-coded macro.
Does the routine need a recalled header?
No
Yes
Tpost the ERB to the Disk I/O QCB.
Set LCBRCQCB to point to Buffer Disposition
Exit to the TCAM Dispatcher.
Set up to execute the routine for the next macro.
Get the routine address from an AVT table of addresses.
Is this the INEND or OUTEND macro?
No
Yes
Tpost the LCB to the IEDQBD02 entry point of Buffer Disposition.
Is this the OUTEND macro?
Yes
Tpost the buffer to the Disk I/O QCB to be marked serviced.
No
Is multiple routing specified ?
No
Yes
Tpost the ERB to the Disk I/O QCB to get a recalled header.
Set LCBRCQCB to point to the Multiple Routing subtask.
Is LOCK in effect ?
Yes
No
Tpost the LCB to itself to free the line.
Exit to the TCAM Dispatcher.
MH Functional Routine
Perform the fonctions indicated by the macro.
If a recalled header buffer was not required, tpost the ERB to the Buffer Disposition QCB and exit to the TCAM Dispatcher.
Tpost the buffer to the Destination QCB.
Set LCBRCQCB to point to Buffer Disposition.
A
CPB Initialization
IEDQFA
Get a recalled header buffer.
Multiple Routing Subtask
IEDQBA
If the input element is an ERB, get the current buffer address.
Link to the Forward routine to scan for the destination name.
Is an EOA string found?
Yes
Tpost the ERB to the Buffer Disposition QCB and indicate the end of a receive operation.
Tpost the buffer to the Buffer Return QCB.
No
Is an incomplete destination found?
Yes
Tpost the ERB to the Disk I/O QCB to get another buffer.
Set LCBRCQCB to point to the multiple Routing subtask.
The header is obtained and Disk End Appendage returns to the Multiple Routing subtask.
No
Is the destination invalid?
No
Tpost the buffer to the Destination QCB to be sent.
Exit to the TCAM Dispatcher.
Yes
TCAM Dispatcher
LEGEND:
Control flow through a branch or sequential instructions
Control flow through the TCAM Dispatcher
OS
Control flow through OS

Chart 14. Flow of Buffers through a Message Handler



Incoming MH Group

- Tpost the first message segment to its Destination QCB.

Destination Scheduler IEDQHM

1. Locate the message queues data set address for the first unit by examining QCBDNHDR in the Destination QCB.
2. Store this value of address in the buffer prefix (PRFCRCD) and in the SCB (SCBCHDR).
3. Place the AVTNADDR value in the next-message field (QCBDNHDR) in the Destination QCB, in the buffer prefix (PRFNHDR), and in the SCB (SCBNHDR) to indicate the location for the first buffer of the next message to be received.
4. Update AVTNADDR by adding one.
5. Determine if there are more units (additional records) in the message segment. IF there are:
   - a. Place the AVT value of address in PRFTRA in the buffer prefix to queue the first unit.
   - b. Assign contiguous values of address to the other units until all are queued. The only pointer required to locate the additional records is the first unit.
   - c. Update the AVT value of address by adding the total number of additional units.
6. Is this the last segment of the message? Yes / No
   - a. Assign the current AVT value of address as the next-buffer location by placing the address value from PRFNTXT in the buffer prefix and in SCBNTXT in the SCB.
   - b. Update the AVT value of address by adding one.
7. Is this the last segment of the message? Yes / No
   - a. Assign SCBNTXT as the location for the first unit of this segment by placing SCBNTXT in PRFCRCD in the buffer prefix.
   - b. Put the address of the first unit of this message in the buffer prefix by placing SCBCHDR in PRFCHDR.
   - c. Queue any additional records, as in 5.
8. Set up the queue-back chain.
9. Tpost the buffer to the Disk I/O QCB.

CPB Initialization IEDQFA

- Use CPBs to write the buffer out on the disk message queues data set.

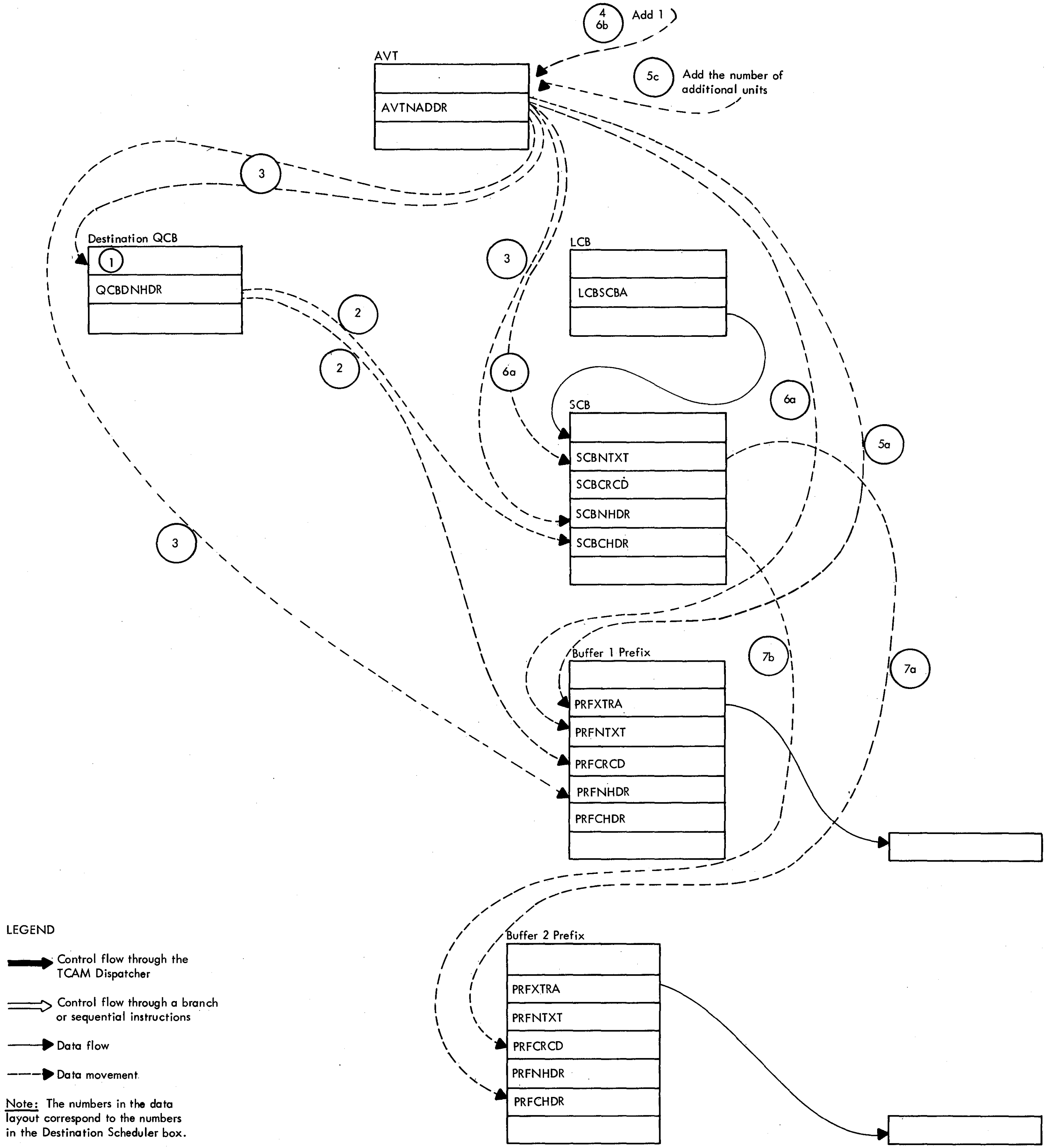
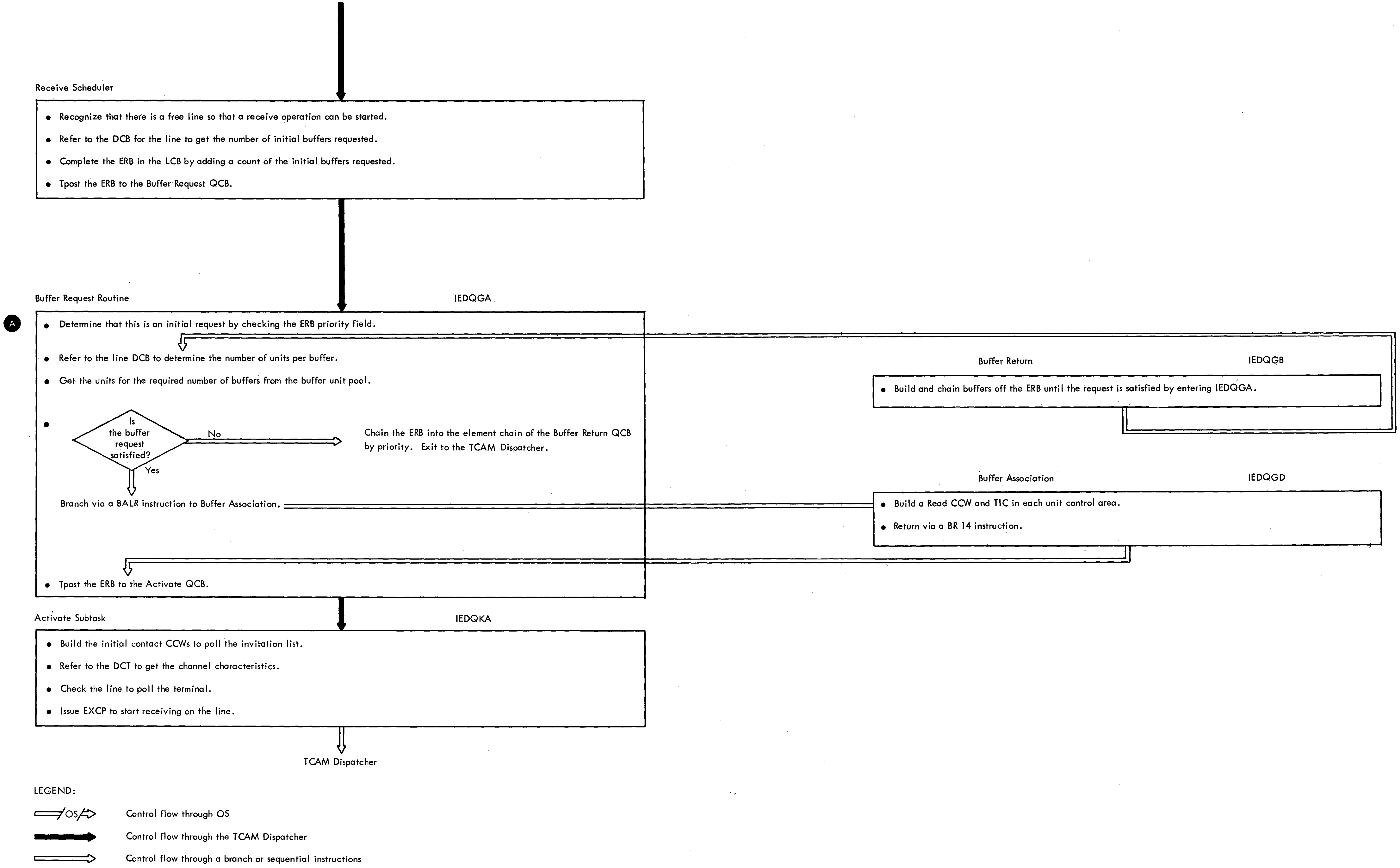


Chart 15. Nonreusable Disk Queuing Functions of the Destination Scheduler Routine

RECEIVE START - an LCB tposted to itself on top of the ready queue indicates that a line is free. At open time, the Receive Scheduler STCB has the highest priority on the STCB chain of the LCB.



Receive Scheduler

- Recognize that there is a free line so that a receive operation can be started.
- Refer to the DCB for the line to get the number of initial buffers requested.
- Complete the ERB in the LCB by adding a count of the initial buffers requested.
- Tpost the ERB to the Buffer Request QCB.

Buffer Request Routine IEDQGA

- Determine that this is an initial request by checking the ERB priority field.
- Refer to the line DCB to determine the number of units per buffer.
- Get the units for the required number of buffers from the buffer unit pool.
- Is the buffer request satisfied? No: Chain the ERB into the element chain of the Buffer Return QCB by priority. Exit to the TCAM Dispatcher. Yes: Branch via a BALR instruction to Buffer Association.
- Tpost the ERB to the Activate QCB.

Buffer Return IEDQGB

- Build and chain buffers off the ERB until the request is satisfied by entering IEDQGA.

Buffer Association IEDQGD

- Build a Read CCW and TIC in each unit control area.
- Return via a BR 14 instruction.

Activate Subtask IEDQKA

- Build the initial contact CCWs to poll the invitation list.
- Refer to the DCT to get the channel characteristics.
- Check the line to poll the terminal.
- Issue EXCP to start receiving on the line.

TCAM Dispatcher

LEGEND:

═/OS/▷	Control flow through OS
➡	Control flow through the TCAM Dispatcher
⇨	Control flow through a branch or sequential instructions

Chart 16. Functional Flow when Receiving from a Line (Part 1 of 4)

Note: PCI Appendage is serviced first if channel end and PCI occur at the same time. If the response to poll is positive, PCI Appendage gains control. Line End Appendage gains control from a negative response, that is, when an EOT is received.

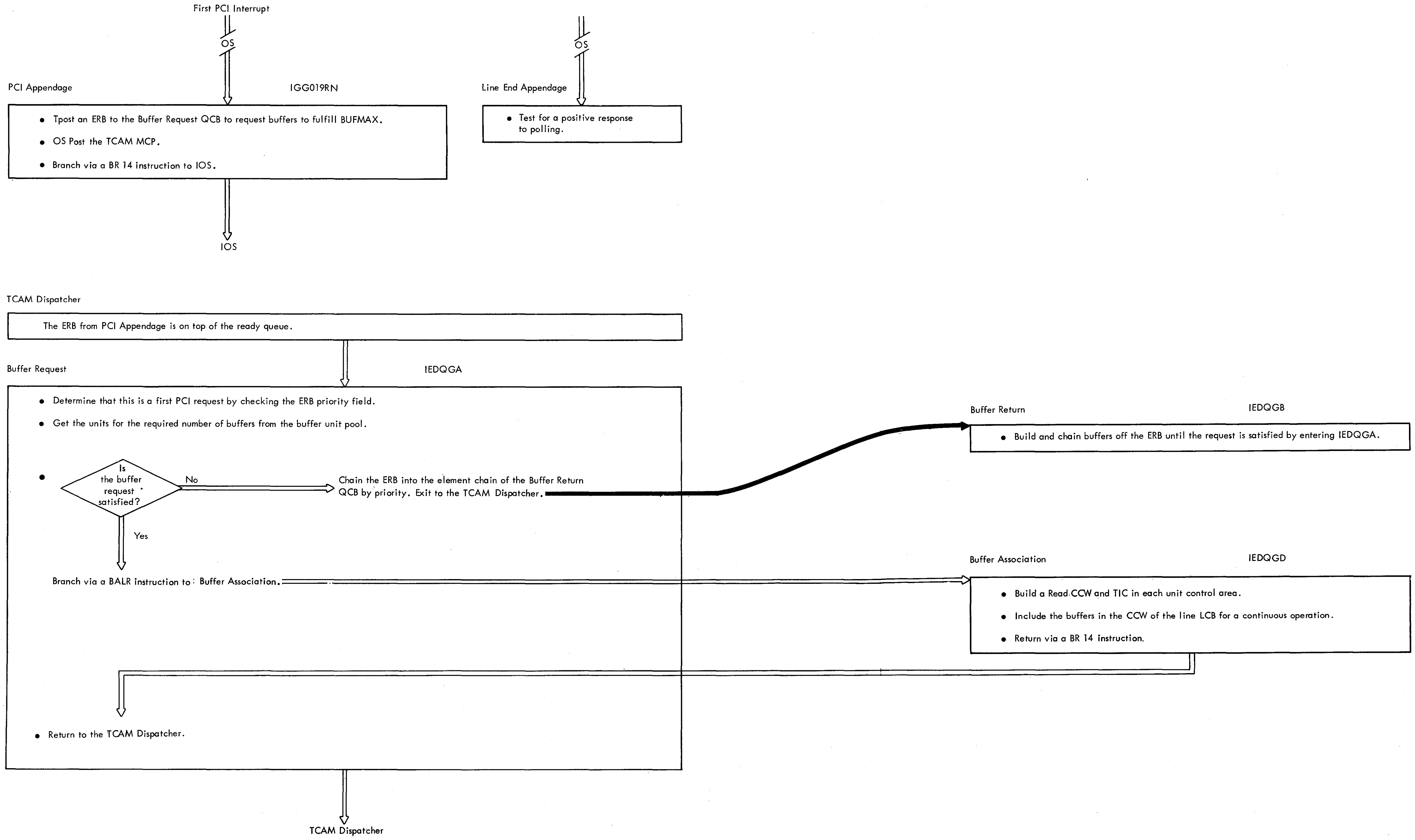
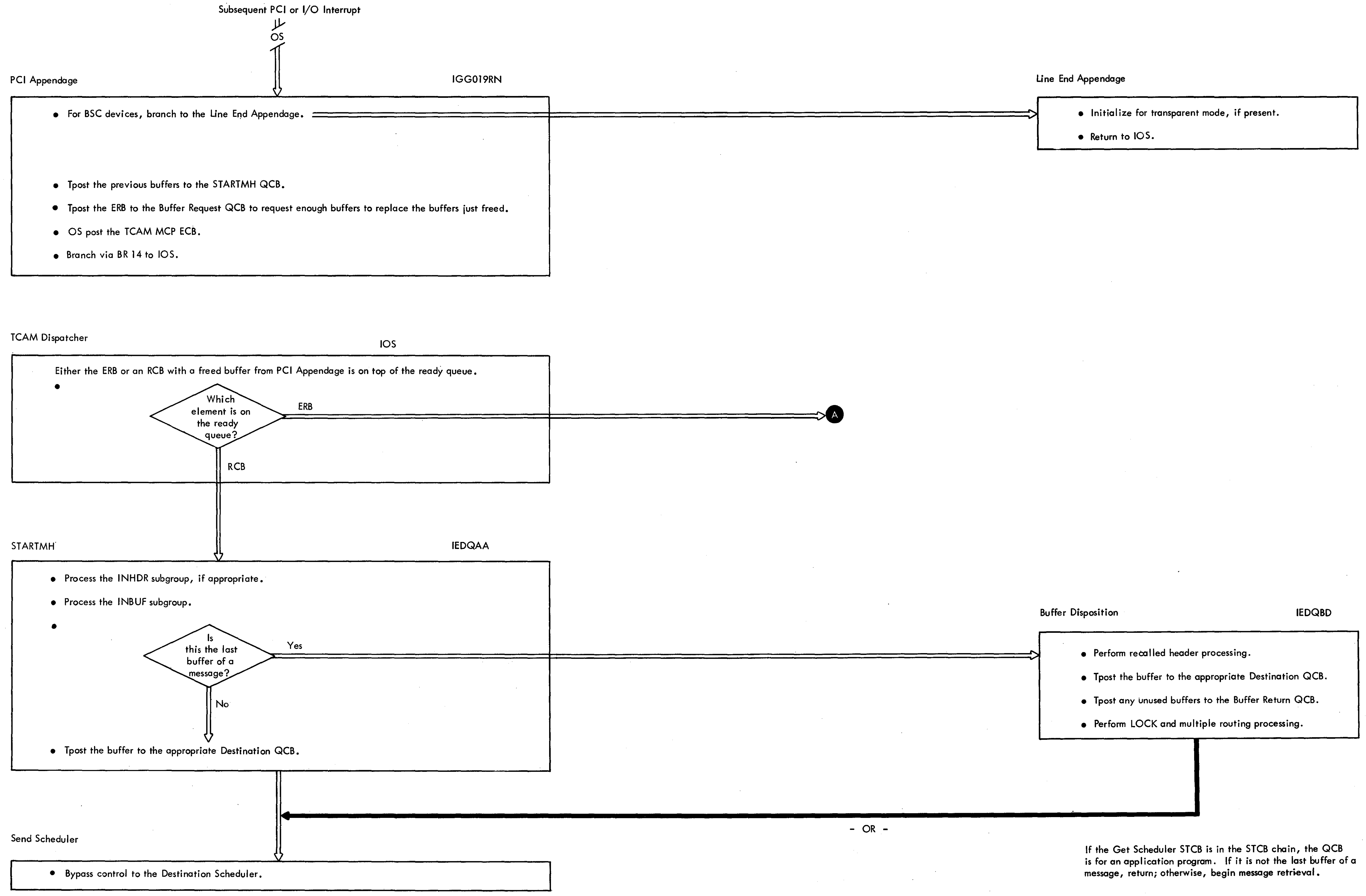

First PCI Interrupt
OS
PCI Appendage
IGG019RN
• Tpost an ERB to the Buffer Request QCB to request buffers to fulfill BUFMAX.
• OS Post the TCAM MCP.
• Branch via a BR 14 instruction to IOS.
IOS
OS
Line End Appendage
• Test for a positive response to polling.
TCAM Dispatcher
The ERB from PCI Appendage is on top of the ready queue.
Buffer Request
IEDQGA
• Determine that this is a first PCI request by checking the ERB priority field.
• Get the units for the required number of buffers from the buffer unit pool.
Is the buffer request satisfied?
No
Chain the ERB into the element chain of the Buffer Return QCB by priority. Exit to the TCAM Dispatcher.
Yes
Branch via a BALR instruction to Buffer Association.
• Return to the TCAM Dispatcher.
TCAM Dispatcher
Buffer Return
IEDQGB
• Build and chain buffers off the ERB until the request is satisfied by entering IEDQGA.
Buffer Association
IEDQGD
• Build a Read CCW and TIC in each unit control area.
• Include the buffers in the CCW of the line LCB for a continuous operation.
• Return via a BR 14 instruction.

Chart 16. Functional Flow when Receivinq from a Line (Part 2 of 4)



PCI Appendage IGG019RN

- For BSC devices, branch to the Line End Appendage.
- Tpost the previous buffers to the STARTMH QCB.
- Tpost the ERB to the Buffer Request QCB to request enough buffers to replace the buffers just freed.
- OS post the TCAM MCP ECB.
- Branch via BR 14 to IOS.

Line End Appendage

- Initialize for transparent mode, if present.
- Return to IOS.

TCAM Dispatcher IOS

Either the ERB or an RCB with a freed buffer from PCI Appendage is on top of the ready queue.

- Which element is on the ready queue?

STARTMH IEDQAA

- Process the INHDR subgroup, if appropriate.
- Process the INBUF subgroup.
- Is this the last buffer of a message?
- Tpost the buffer to the appropriate Destination QCB.

Buffer Disposition IEDQBD

- Perform recalled header processing.
- Tpost the buffer to the appropriate Destination QCB.
- Tpost any unused buffers to the Buffer Return QCB.
- Perform LOCK and multiple routing processing.

If the Get Scheduler STCB is in the STCB chain, the QCB is for an application program. If it is not the last buffer of a message, return; otherwise, begin message retrieval.

Send Scheduler

- Bypass control to the Destination Scheduler.

Chart 16. Functional Flow when Receivinq from a Line (Part 3 of 4)

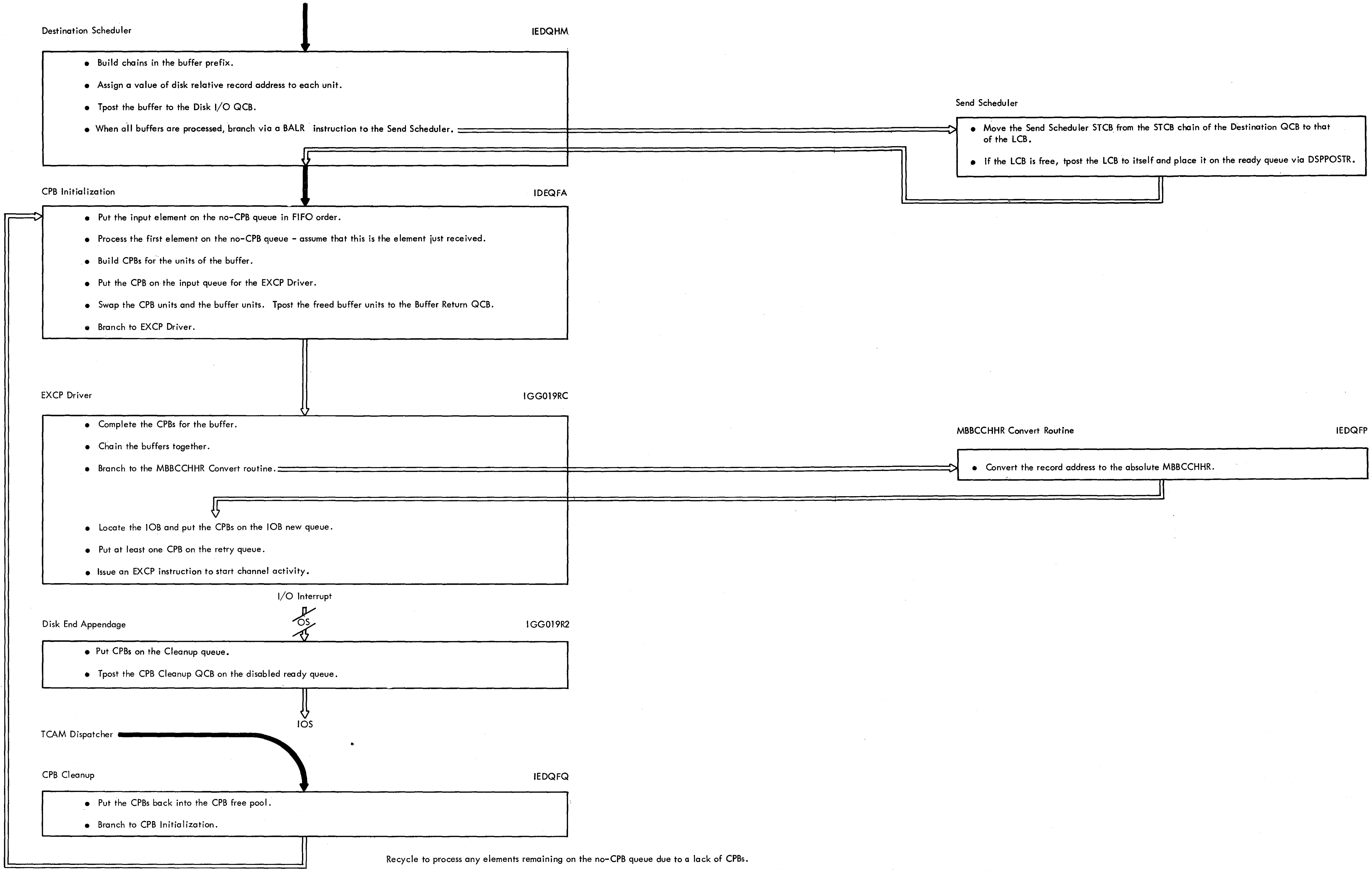
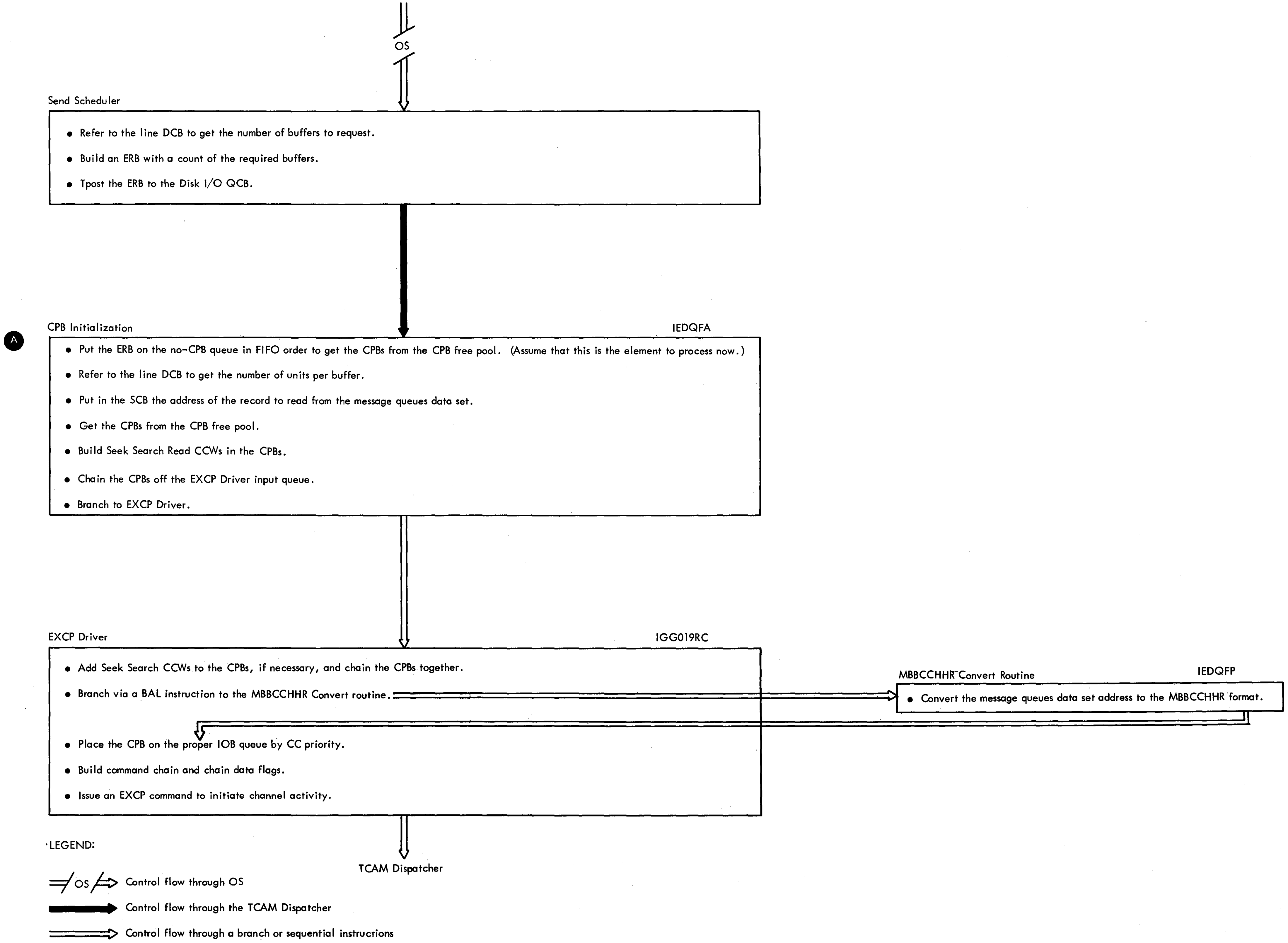

Destination Scheduler
IEDQHM
• Build chains in the buffer prefix.
• Assign a value of disk relative record address to each unit.
• Tpost the buffer to the Disk I/O QCB.
• When all buffers are processed, branch via a BALR instruction to the Send Scheduler.
Send Scheduler
• Move the Send Scheduler STCB from the STCB chain of the Destination QCB to that of the LCB.
• If the LCB is free, tpost the LCB to itself and place it on the ready queue via DSPPOSTR.
CPB Initialization
IDEQFA
• Put the input element on the no-CPB queue in FIFO order.
• Process the first element on the no-CPB queue - assume that this is the element just received.
• Build CPBs for the units of the buffer.
• Put the CPB on the input queue for the EXCP Driver.
• Swap the CPB units and the buffer units. Tpost the freed buffer units to the Buffer Return QCB.
• Branch to EXCP Driver.
EXCP Driver
IGG019RC
• Complete the CPBs for the buffer.
• Chain the buffers together.
• Branch to the MBBCCHHR Convert routine.
MBBCCHHR Convert Routine
IEDQFP
• Convert the record address to the absolute MBBCCHHR.
• Locate the IOB and put the CPBs on the IOB new queue.
• Put at least one CPB on the retry queue.
• Issue an EXCP instruction to start channel activity.
I/O Interrupt
OS
Disk End Appendage
IGG019R2
• Put CPBs on the Cleanup queue.
• Tpost the CPB Cleanup QCB on the disabled ready queue.
IOS
TCAM Dispatcher
CPB Cleanup
IEDQFQ
• Put the CPBs back into the CPB free pool.
• Branch to CPB Initialization.
Recycle to process any elements remaining on the no-CPB queue due to a lack of CPBs.

Chart 16. Functional Flow when Receiving from a Line (Part 4 of 4)

SEND START - an LCB tposted to itself on top of the ready queue indicates that a line is free. A send operation can be initiated when the Send Scheduler STCB has top priority in the STCB chain of the LCB. At open time the Send Scheduler STCB is on the STCB chain of the Destination QCB to await a full message. When the Receive Scheduler has no messages to receive, the Send Scheduler STCB is moved to the STCB chain of the LCB. It remains on the LCB to send messages until there is no message free to send. At this time, the Send Scheduler moves its STCB to the STCB chain of the Destination QCB.



Send Scheduler

- Refer to the line DCB to get the number of buffers to request.
- Build an ERB with a count of the required buffers.
- Tpost the ERB to the Disk I/O QCB.

CPB Initialization IEDQFA

- Put the ERB on the no-CPB queue in FIFO order to get the CPBs from the CPB free pool. (Assume that this is the element to process now.)
- Refer to the line DCB to get the number of units per buffer.
- Put in the SCB the address of the record to read from the message queues data set.
- Get the CPBs from the CPB free pool.
- Build Seek Search Read CCWs in the CPBs.
- Chain the CPBs off the EXCP Driver input queue.
- Branch to EXCP Driver.

EXCP Driver IGG019RC

- Add Seek Search CCWs to the CPBs, if necessary, and chain the CPBs together.
- Branch via a BAL instruction to the MBBCCHHR Convert routine.
- Place the CPB on the proper IOB queue by CC priority.
- Build command chain and chain data flags.
- Issue an EXCP command to initiate channel activity.

MBBCCHHR Convert Routine IEDQFP

- Convert the message queues data set address to the MBBCCHHR format.

TCAM Dispatcher

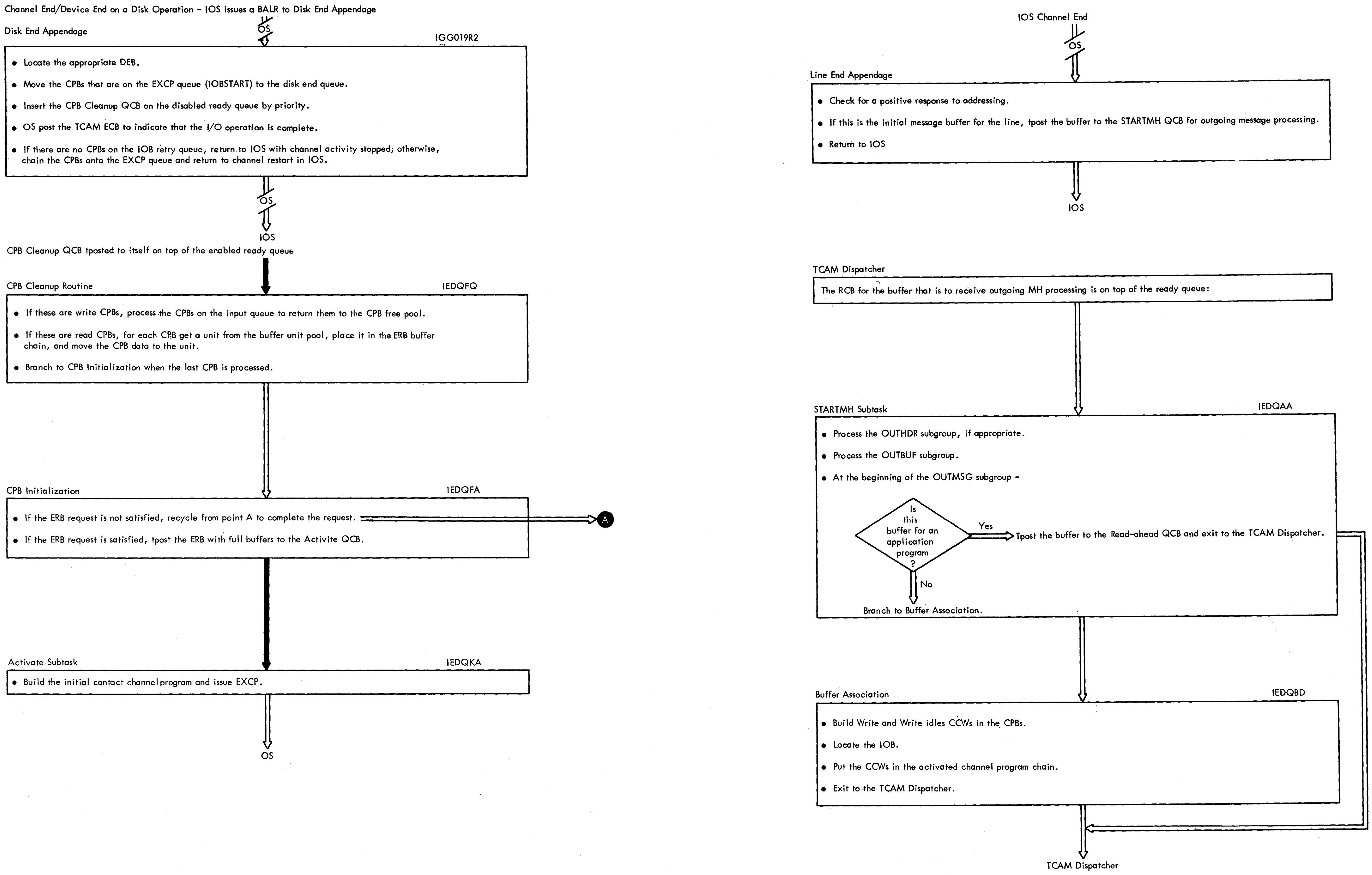
LEGEND:

Control flow through OS

Control flow through the TCAM Dispatcher

Control flow through a branch or sequential instructions

Chart 17. Functional Flow when Sending to a Line (Part 1 of 3)



Channel End/Device End on a Disk Operation - IOS issues a BALR to Disk End Appendage

Disk End Appendage — IGG019R2

- Locate the appropriate DEB.
- Move the CPBs that are on the EXCP queue (IOBSTART) to the disk end queue.
- Insert the CPB Cleanup QCB on the disabled ready queue by priority.
- OS post the TCAM ECB to indicate that the I/O operation is complete.
- If there are no CPBs on the IOB retry queue, return to IOS with channel activity stopped; otherwise, chain the CPBs onto the EXCP queue and return to channel restart in IOS.

CPB Cleanup QCB tposted to itself on top of the enabled ready queue

CPB Cleanup Routine — IEDQFQ

- If these are write CPBs, process the CPBs on the input queue to return them to the CPB free pool.
- If these are read CPBs, for each CPB get a unit from the buffer unit pool, place it in the ERB buffer chain, and move the CPB data to the unit.
- Branch to CPB Initialization when the last CPB is processed.

CPB Initialization — IEDQFA

- If the ERB request is not satisfied, recycle from point A to complete the request.
- If the ERB request is satisfied, tpost the ERB with full buffers to the Activite QCB.

Activate Subtask — IEDQKA

- Build the initial contact channel program and issue EXCP.

IOS Channel End

Line End Appendage

- Check for a positive response to addressing.
- If this is the initial message buffer for the line, tpost the buffer to the STARTMH QCB for outgoing message processing.
- Return to IOS

TCAM Dispatcher

The RCB for the buffer that is to receive outgoing MH processing is on top of the ready queue:

STARTMH Subtask — IEDQAA

- Process the OUTHDR subgroup, if appropriate.
- Process the OUTBUF subgroup.
- At the beginning of the OUTMSG subgroup -

Is this buffer for an application program?

Yes — Tpost the buffer to the Read-ahead QCB and exit to the TCAM Dispatcher.

No — Branch to Buffer Association.

Buffer Association — IEDQBD

- Build Write and Write idles CCWs in the CPBs.
- Locate the IOB.
- Put the CCWs in the activated channel program chain.
- Exit to the TCAM Dispatcher.

Chart 17. Functional Flow when Sending to a Line (Part 2 of 3)

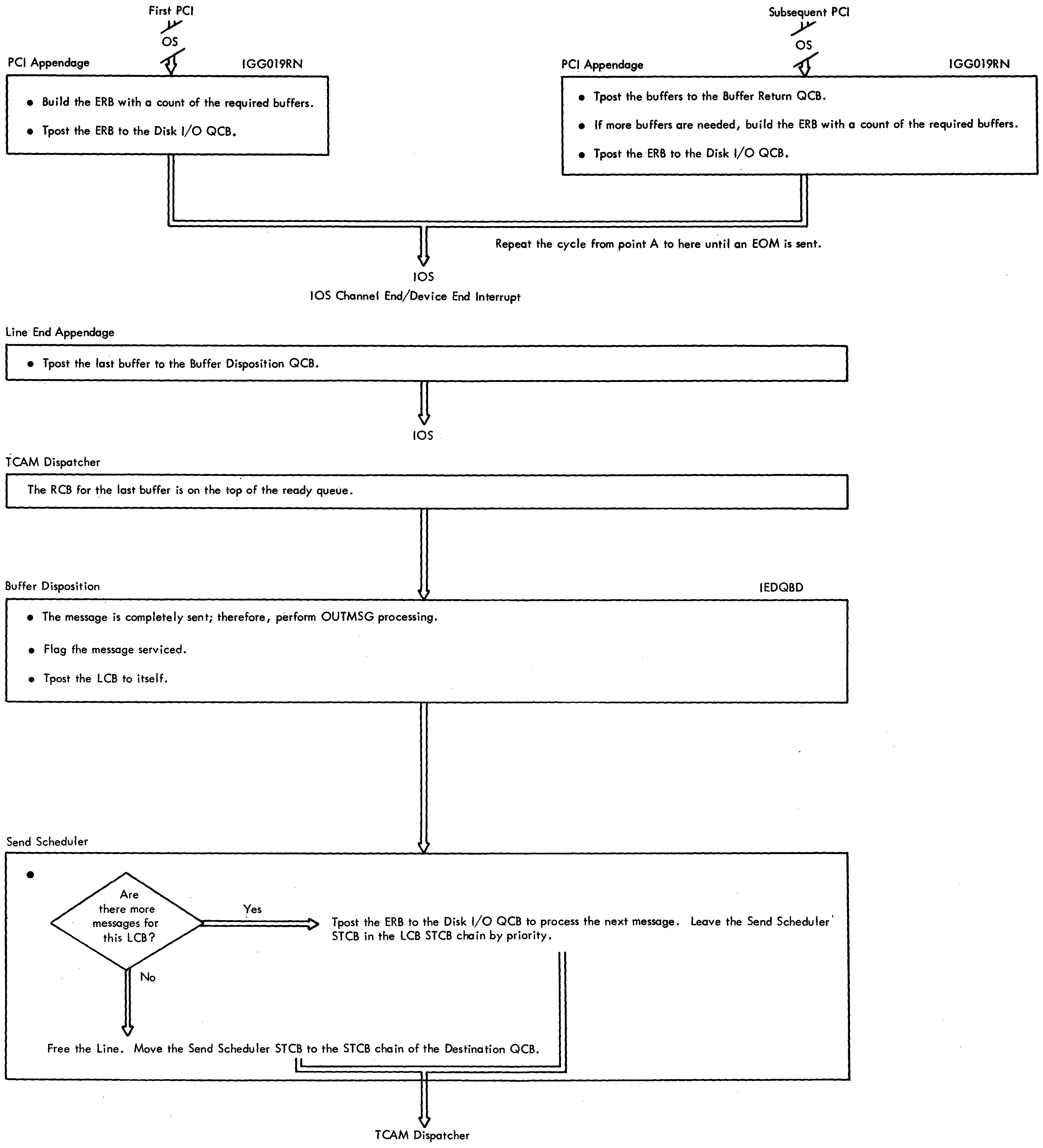

First PCI
OS
PCI Appendage
IGG019RN
• Build the ERB with a count of the required buffers.
• Tpost the ERB to the Disk I/O QCB.
Subsequent PCI
OS
PCI Appendage
IGG019RN
• Tpost the buffers to the Buffer Return QCB.
• If more buffers are needed, build the ERB with a count of the required buffers.
• Tpost the ERB to the Disk I/O QCB.
Repeat the cycle from point A to here until an EOM is sent.
IOS
IOS Channel End/Device End Interrupt
Line End Appendage
• Tpost the last buffer to the Buffer Disposition QCB.
IOS
TCAM Dispatcher
The RCB for the last buffer is on the top of the ready queue.
Buffer Disposition
IEDQBD
• The message is completely sent; therefore, perform OUTMSG processing.
• Flag the message serviced.
• Tpost the LCB to itself.
Send Scheduler
•
Are there more messages for this LCB?
Yes
Tpost the ERB to the Disk I/O QCB to process the next message. Leave the Send Scheduler STCB in the LCB STCB chain by priority.
No
Free the Line. Move the Send Scheduler STCB to the STCB chain of the Destination QCB.
TCAM Dispatcher

Chart 17. Functional Flow when Sending to a Line (Part 3 of 3)

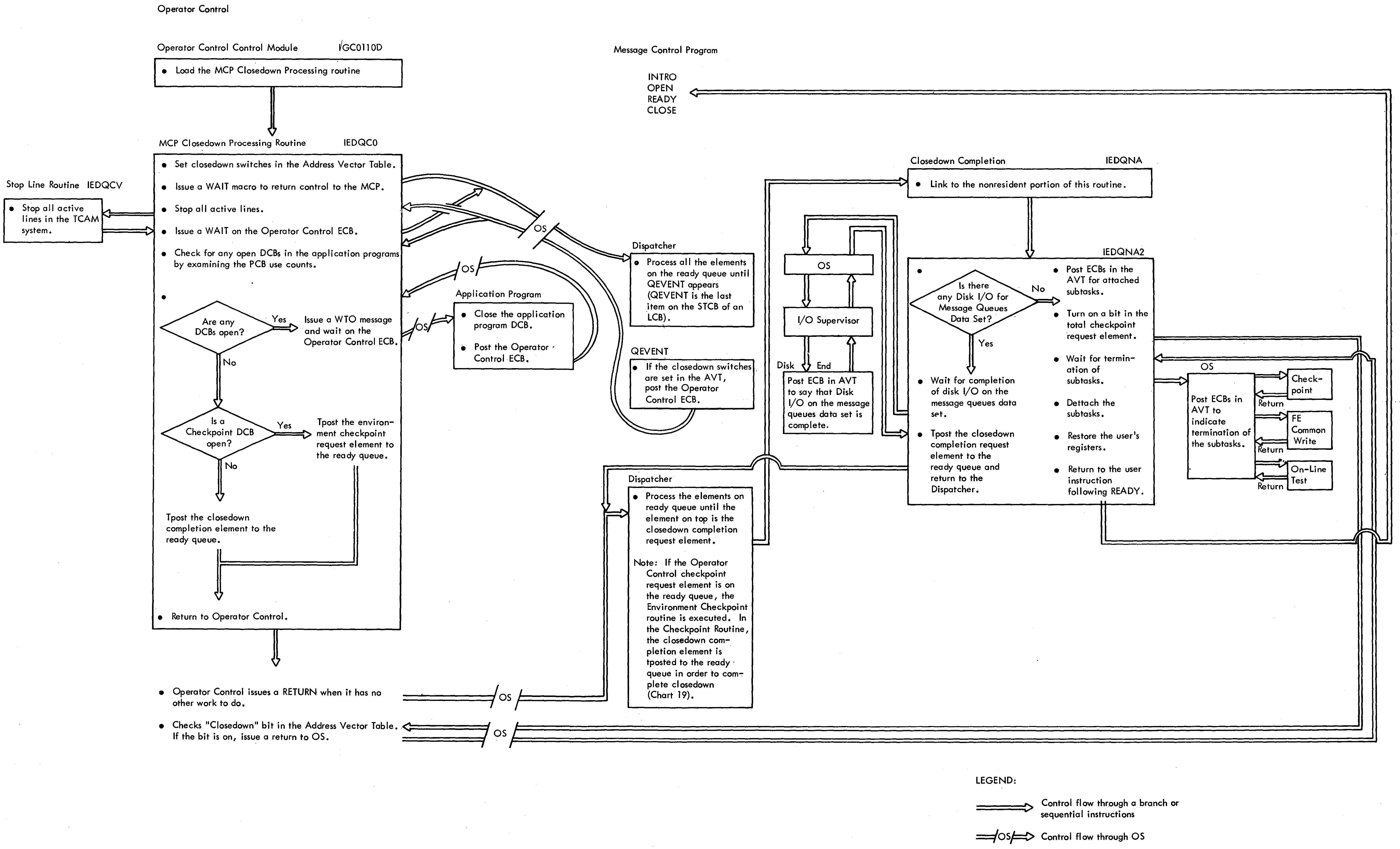

Operator Control
Operator Control Control Module IGC0110D
• Load the MCP Closedown Processing routine
MCP Closedown Processing Routine IEDQC0
• Set closedown switches in the Address Vector Table.
• Issue a WAIT macro to return control to the MCP.
• Stop all active lines.
• Issue a WAIT on the Operator Control ECB.
• Check for any open DCBs in the application programs by examining the PCB use counts.
•
Are any DCBs open?
Yes
Issue a WTO message and wait on the Operator Control ECB.
No
Is a Checkpoint DCB open?
Yes
Tpost the environment checkpoint request element to the ready queue.
No
Tpost the closedown completion element to the ready queue.
• Return to Operator Control.
• Operator Control issues a RETURN when it has no other work to do.
• Checks "Closedown" bit in the Address Vector Table. If the bit is on, issue a return to OS.
Stop Line Routine IEDQCV
• Stop all active lines in the TCAM system.
OS
Application Program
• Close the application program DCB.
• Post the Operator Control ECB.
Message Control Program
INTRO
OPEN
READY
CLOSE
Dispatcher
• Process all the elements on the ready queue until QEVENT appears (QEVENT is the last item on the STCB of an LCB).
QEVENT
• If the closedown switches are set in the AVT, post the Operator Control ECB.
Dispatcher
• Process the elements on ready queue until the element on top is the closedown completion request element.
Note: If the Operator Control checkpoint request element is on the ready queue, the Environment Checkpoint routine is executed. In the Checkpoint Routine, the closedown completion element is tposted to the ready queue in order to complete closedown (Chart 19).
OS
I/O Supervisor
Disk
End
Post ECB in AVT to say that Disk I/O on the message queues data set is complete.
Closedown Completion IEDQNA
• Link to the nonresident portion of this routine.
IEDQNA2
•
Is there any Disk I/O for Message Queues Data Set?
No
Yes
• Wait for completion of disk I/O on the message queues data set.
• Tpost the closedown completion request element to the ready queue and return to the Dispatcher.
• Post ECBs in the AVT for attached subtasks.
• Turn on a bit in the total checkpoint request element.
• Wait for termination of subtasks.
• Dettach the subtasks.
• Restore the user's registers.
• Return to the user instruction following READY.
OS
Post ECBs in AVT to indicate termination of the subtasks.
Checkpoint
Return
FE Common Write
Return
On-Line Test
Return
LEGEND:
Control flow through a branch or sequential instructions
OS Control flow through OS

Chart 18. Functional Flow for MCPCLOSE and Closedown Completion

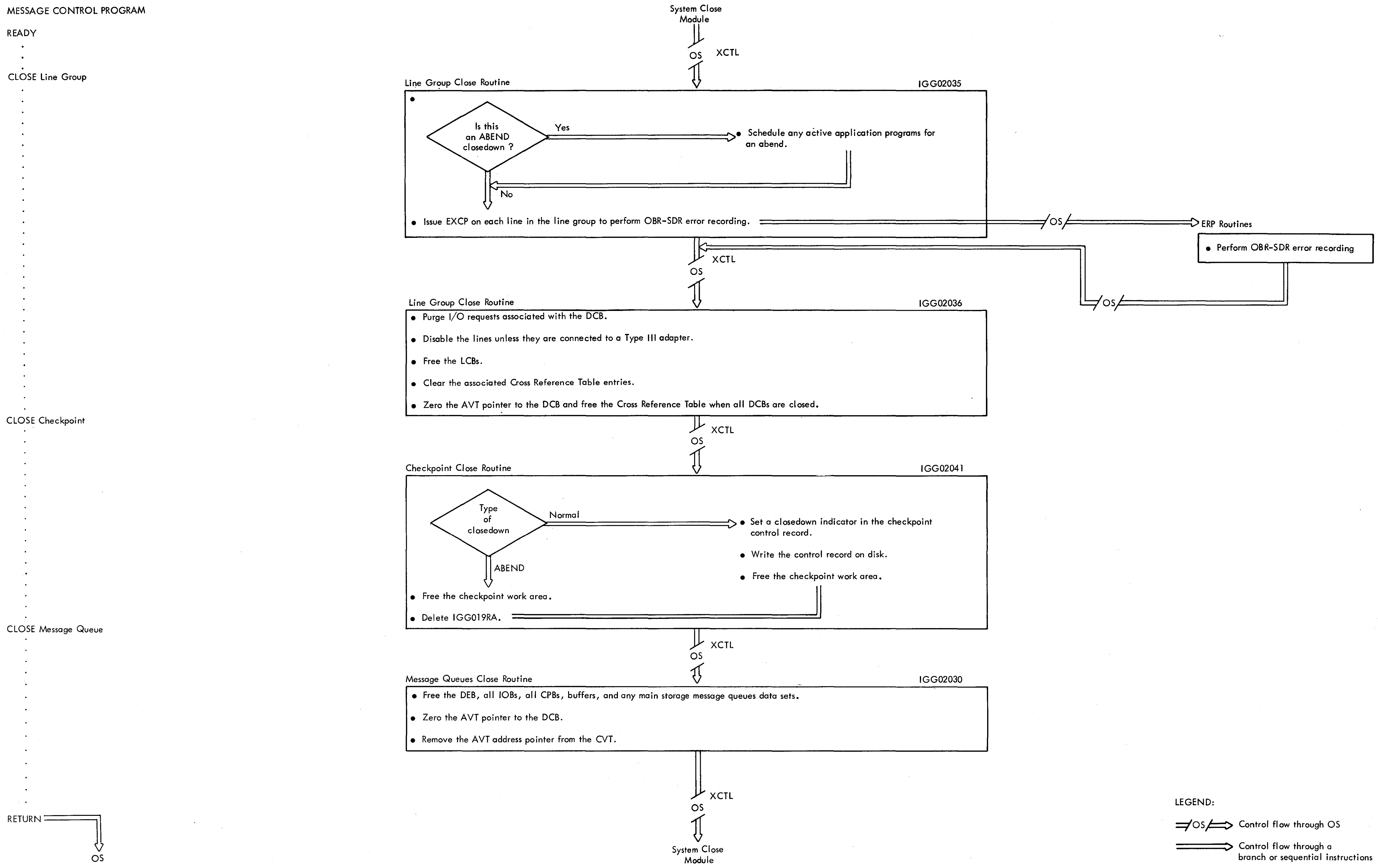
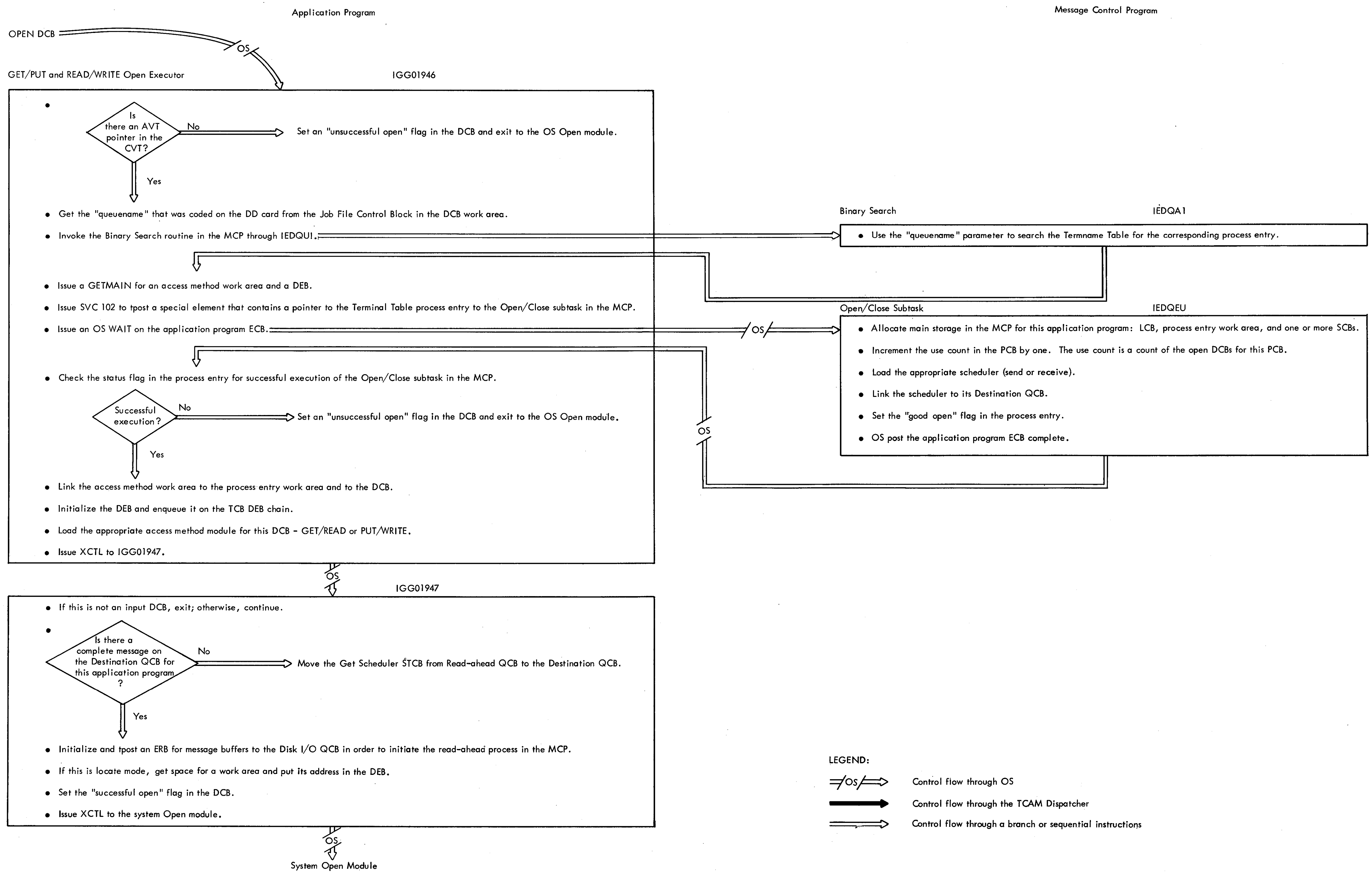
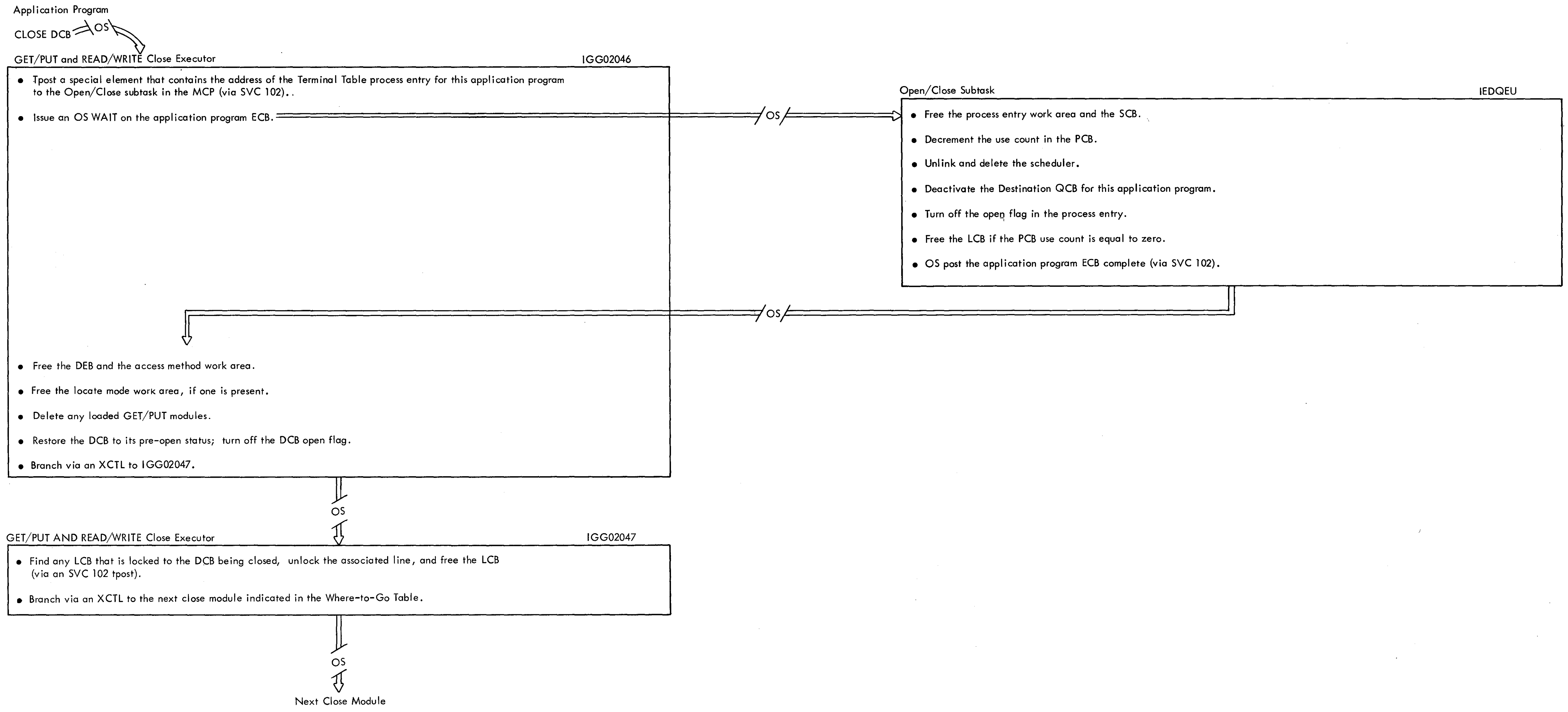

MESSAGE CONTROL PROGRAM
READY
CLOSE Line Group
CLOSE Checkpoint
CLOSE Message Queue
RETURN
OS
System Close Module
OS
XCTL
Line Group Close Routine
IGG02035
Is this an ABEND closedown ?
Yes
• Schedule any active application programs for an abend.
No
• Issue EXCP on each line in the line group to perform OBR-SDR error recording.
OS
ERP Routines
• Perform OBR-SDR error recording
OS
XCTL
OS
Line Group Close Routine
IGG02036
• Purge I/O requests associated with the DCB.
• Disable the lines unless they are connected to a Type III adapter.
• Free the LCBs.
• Clear the associated Cross Reference Table entries.
• Zero the AVT pointer to the DCB and free the Cross Reference Table when all DCBs are closed.
XCTL
OS
Checkpoint Close Routine
IGG02041
Type of closedown
Normal
• Set a closedown indicator in the checkpoint control record.
• Write the control record on disk.
• Free the checkpoint work area.
ABEND
• Free the checkpoint work area.
• Delete IGG019RA.
XCTL
OS
Message Queues Close Routine
IGG02030
• Free the DEB, all IOBs, all CPBs, buffers, and any main storage message queues data sets.
• Zero the AVT pointer to the DCB.
• Remove the AVT address pointer from the CVT.
XCTL
OS
System Close Module
LEGEND:
OS Control flow through OS
Control flow through a branch or sequential instructions

Chart 19. Functional Flow of the DCB Closedown Procedure



Application Program
Message Control Program
OPEN DCB
OS
GET/PUT and READ/WRITE Open Executor
IGG01946
Is there an AVT pointer in the CVT?
No
Yes
Set an "unsuccessful open" flag in the DCB and exit to the OS Open module.
Get the "queuename" that was coded on the DD card from the Job File Control Block in the DCB work area.
Invoke the Binary Search routine in the MCP through IEDQUI.
Binary Search
IEDQA1
Use the "queuename" parameter to search the Termname Table for the corresponding process entry.
Issue a GETMAIN for an access method work area and a DEB.
Issue SVC 102 to tpost a special element that contains a pointer to the Terminal Table process entry to the Open/Close subtask in the MCP.
Issue an OS WAIT on the application program ECB.
OS
Open/Close Subtask
IEDQEU
Allocate main storage in the MCP for this application program: LCB, process entry work area, and one or more SCBs.
Increment the use count in the PCB by one. The use count is a count of the open DCBs for this PCB.
Load the appropriate scheduler (send or receive).
Link the scheduler to its Destination QCB.
Set the "good open" flag in the process entry.
OS post the application program ECB complete.
OS
Check the status flag in the process entry for successful execution of the Open/Close subtask in the MCP.
Successful execution?
No
Yes
Set an "unsuccessful open" flag in the DCB and exit to the OS Open module.
Link the access method work area to the process entry work area and to the DCB.
Initialize the DEB and enqueue it on the TCB DEB chain.
Load the appropriate access method module for this DCB - GET/READ or PUT/WRITE.
Issue XCTL to IGG01947.
OS
IGG01947
If this is not an input DCB, exit; otherwise, continue.
Is there a complete message on the Destination QCB for this application program ?
No
Yes
Move the Get Scheduler $TCB from Read-ahead QCB to the Destination QCB.
Initialize and tpost an ERB for message buffers to the Disk I/O QCB in order to initiate the read-ahead process in the MCP.
If this is locate mode, get space for a work area and put its address in the DEB.
Set the "successful open" flag in the DCB.
Issue XCTL to the system Open module.
OS
System Open Module
LEGEND:
OS
Control flow through OS
Control flow through the TCAM Dispatcher
Control flow through a branch or sequential instructions

Chart 20. Initialization Functions in an Application Program



Application Program

CLOSE DCB

GET/PUT and READ/WRITE Close Executor IGG02046

- Tpost a special element that contains the address of the Terminal Table process entry for this application program to the Open/Close subtask in the MCP (via SVC 102)..
- Issue an OS WAIT on the application program ECB.
- Free the DEB and the access method work area.
- Free the locate mode work area, if one is present.
- Delete any loaded GET/PUT modules.
- Restore the DCB to its pre-open status; turn off the DCB open flag.
- Branch via an XCTL to IGG02047.

Open/Close Subtask IEDQEU

- Free the process entry work area and the SCB.
- Decrement the use count in the PCB.
- Unlink and delete the scheduler.
- Deactivate the Destination QCB for this application program.
- Turn off the open flag in the process entry.
- Free the LCB if the PCB use count is equal to zero.
- OS post the application program ECB complete (via SVC 102).

GET/PUT AND READ/WRITE Close Executor IGG02047

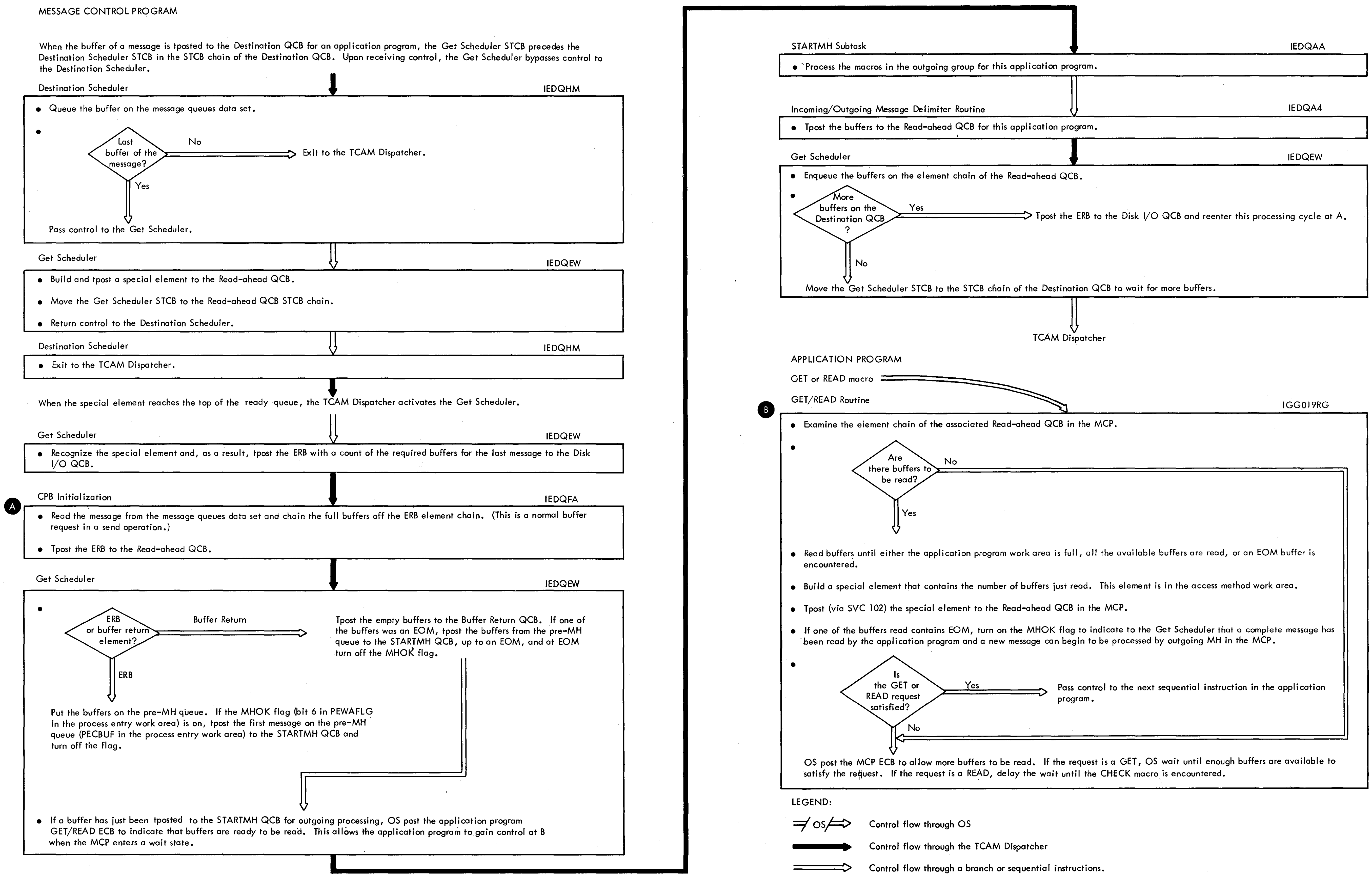
- Find any LCB that is locked to the DCB being closed, unlock the associated line, and free the LCB (via an SVC 102 tpost).
- Branch via an XCTL to the next close module indicated in the Where-to-Go Table.

Next Close Module

LEGEND:

=/OS/⇨ Control Flow through OS

Chart 21. Termination Functions in an Application Program



MESSAGE CONTROL PROGRAM

When the buffer of a message is tposted to the Destination QCB for an application program, the Get Scheduler STCB precedes the Destination Scheduler STCB in the STCB chain of the Destination QCB. Upon receiving control, the Get Scheduler bypasses control to the Destination Scheduler.

Destination Scheduler IEDQHM

- Queue the buffer on the message queues data set.
- Last buffer of the message? No: Exit to the TCAM Dispatcher. Yes: Pass control to the Get Scheduler.

Get Scheduler IEDQEW

- Build and tpost a special element to the Read-ahead QCB.
- Move the Get Scheduler STCB to the Read-ahead QCB STCB chain.
- Return control to the Destination Scheduler.

Destination Scheduler IEDQHM

- Exit to the TCAM Dispatcher.

When the special element reaches the top of the ready queue, the TCAM Dispatcher activates the Get Scheduler.

Get Scheduler IEDQEW

- Recognize the special element and, as a result, tpost the ERB with a count of the required buffers for the last message to the Disk I/O QCB.

A CPB Initialization IEDQFA

- Read the message from the message queues data set and chain the full buffers off the ERB element chain. (This is a normal buffer request in a send operation.)
- Tpost the ERB to the Read-ahead QCB.

Get Scheduler IEDQEW

- ERB or buffer return element? Buffer Return: Tpost the empty buffers to the Buffer Return QCB. If one of the buffers was an EOM, tpost the buffers from the pre-MH queue to the STARTMH QCB, up to an EOM, and at EOM turn off the MHOK flag. ERB: Put the buffers on the pre-MH queue. If the MHOK flag (bit 6 in PEWAFLG in the process entry work area) is on, tpost the first message on the pre-MH queue (PECBUF in the process entry work area) to the STARTMH QCB and turn off the flag.
- If a buffer has just been tposted to the STARTMH QCB for outgoing processing, OS post the application program GET/READ ECB to indicate that buffers are ready to be read. This allows the application program to gain control at B when the MCP enters a wait state.

STARTMH Subtask IEDQAA

- Process the macros in the outgoing group for this application program.

Incoming/Outgoing Message Delimiter Routine IEDQA4

- Tpost the buffers to the Read-ahead QCB for this application program.

Get Scheduler IEDQEW

- Enqueue the buffers on the element chain of the Read-ahead QCB.
- More buffers on the Destination QCB ? Yes: Tpost the ERB to the Disk I/O QCB and reenter this processing cycle at A. No: Move the Get Scheduler STCB to the STCB chain of the Destination QCB to wait for more buffers.

TCAM Dispatcher

APPLICATION PROGRAM

GET or READ macro

B GET/READ Routine IGG019RG

- Examine the element chain of the associated Read-ahead QCB in the MCP.
- Are there buffers to be read? Yes / No
- Read buffers until either the application program work area is full, all the available buffers are read, or an EOM buffer is encountered.
- Build a special element that contains the number of buffers just read. This element is in the access method work area.
- Tpost (via SVC 102) the special element to the Read-ahead QCB in the MCP.
- If one of the buffers read contains EOM, turn on the MHOK flag to indicate to the Get Scheduler that a complete message has been read by the application program and a new message can begin to be processed by outgoing MH in the MCP.
- Is the GET or READ request satisfied? Yes: Pass control to the next sequential instruction in the application program. No: OS post the MCP ECB to allow more buffers to be read. If the request is a GET, OS wait until enough buffers are available to satisfy the request. If the request is a READ, delay the wait until the CHECK macro is encountered.

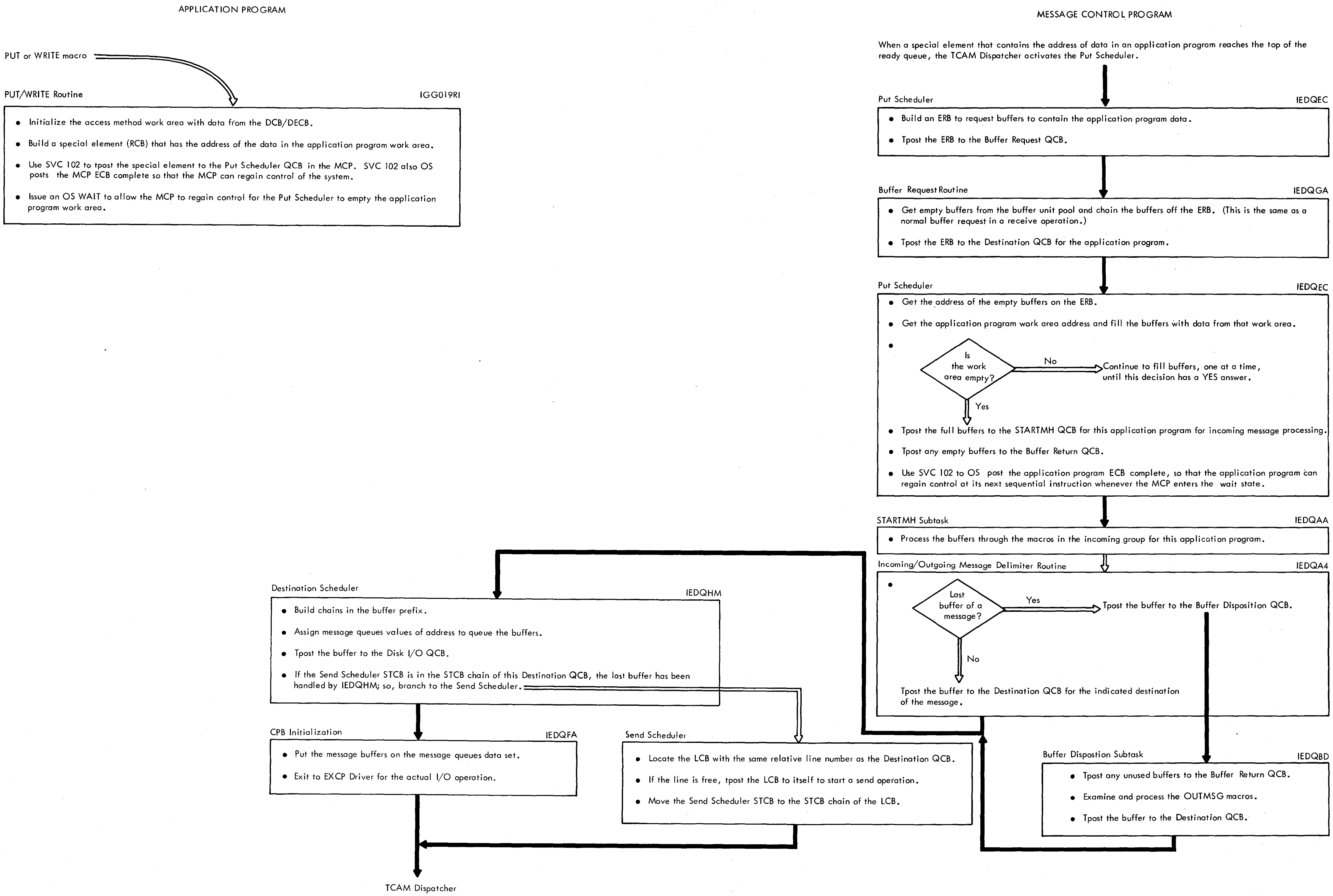
LEGEND:

OS: Control flow through OS

Control flow through the TCAM Dispatcher

Control flow through a branch or sequential instructions.

Chart 22. Functional Flow of How Data is Passed from the MCP to an Application Program



APPLICATION PROGRAM

PUT or WRITE macro

PUT/WRITE Routine IGG019RI

- Initialize the access method work area with data from the DCB/DECB.
- Build a special element (RCB) that has the address of the data in the application program work area.
- Use SVC 102 to tpost the special element to the Put Scheduler QCB in the MCP. SVC 102 also OS posts the MCP ECB complete so that the MCP can regain control of the system.
- Issue an OS WAIT to allow the MCP to regain control for the Put Scheduler to empty the application program work area.

MESSAGE CONTROL PROGRAM

When a special element that contains the address of data in an application program reaches the top of the ready queue, the TCAM Dispatcher activates the Put Scheduler.

Put Scheduler IEDQEC

- Build an ERB to request buffers to contain the application program data.
- Tpost the ERB to the Buffer Request QCB.

Buffer Request Routine IEDQGA

- Get empty buffers from the buffer unit pool and chain the buffers off the ERB. (This is the same as a normal buffer request in a receive operation.)
- Tpost the ERB to the Destination QCB for the application program.

Put Scheduler IEDQEC

- Get the address of the empty buffers on the ERB.
- Get the application program work area address and fill the buffers with data from that work area.
- Is the work area empty? No → Continue to fill buffers, one at a time, until this decision has a YES answer. Yes ↓
- Tpost the full buffers to the STARTMH QCB for this application program for incoming message processing.
- Tpost any empty buffers to the Buffer Return QCB.
- Use SVC 102 to OS post the application program ECB complete, so that the application program can regain control at its next sequential instruction whenever the MCP enters the wait state.

STARTMH Subtask IEDQAA

- Process the buffers through the macros in the incoming group for this application program.

Incoming/Outgoing Message Delimiter Routine IEDQA4

- Last buffer of a message? Yes → Tpost the buffer to the Buffer Disposition QCB. No ↓

Tpost the buffer to the Destination QCB for the indicated destination of the message.

Buffer Dispostion Subtask IEDQBD

- Tpost any unused buffers to the Buffer Return QCB.
- Examine and process the OUTMSG macros.
- Tpost the buffer to the Destination QCB.

Destination Scheduler IEDQHM

- Build chains in the buffer prefix.
- Assign message queues values of address to queue the buffers.
- Tpost the buffer to the Disk I/O QCB.
- If the Send Scheduler STCB is in the STCB chain of this Destination QCB, the last buffer has been handled by IEDQHM; so, branch to the Send Scheduler.

CPB Initialization IEDQFA

- Put the message buffers on the message queues data set.
- Exit to EXCP Driver for the actual I/O operation.

Send Scheduler

- Locate the LCB with the same relative line number as the Destination QCB.
- If the line is free, tpost the LCB to itself to start a send operation.
- Move the Send Scheduler STCB to the STCB chain of the LCB.

TCAM Dispatcher

Chart 23. Functional Flow of How Data is Passed from an Application Program to the MCP

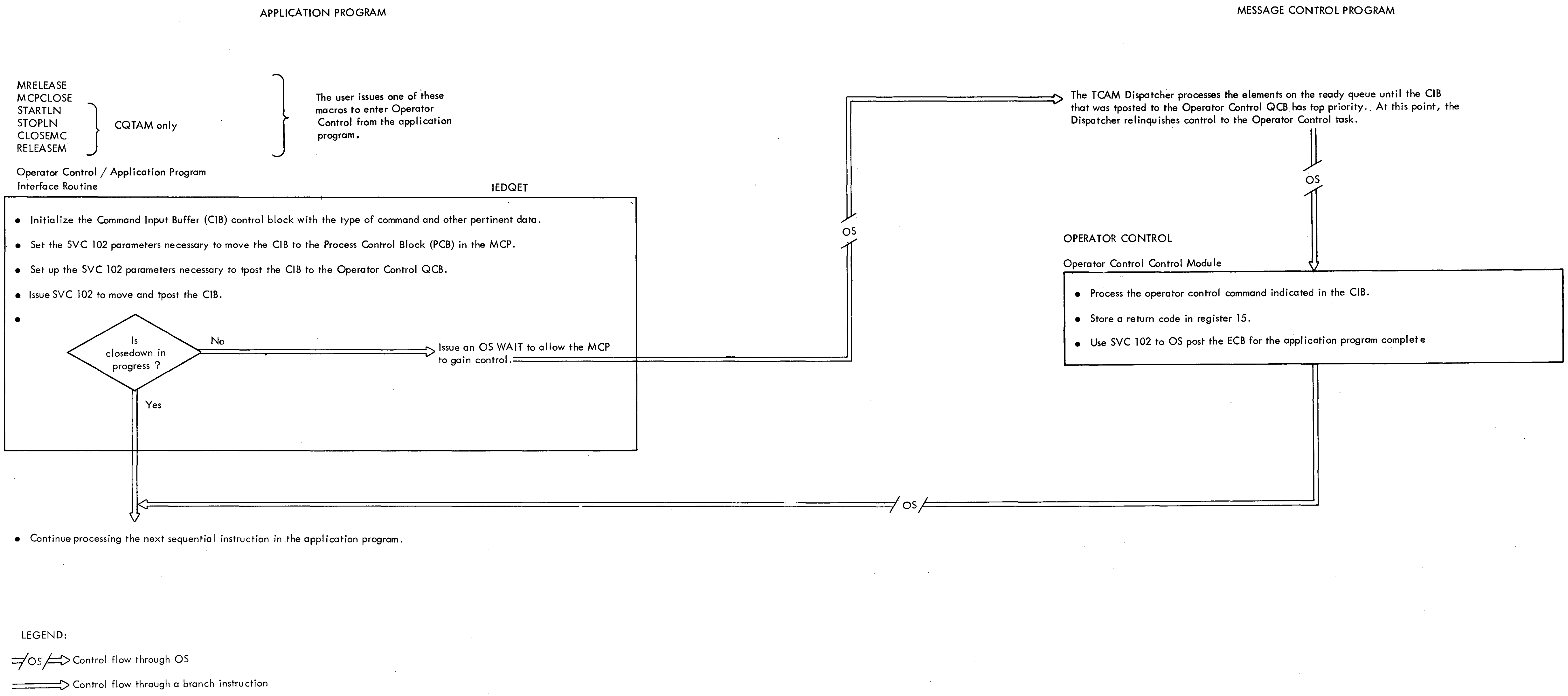

APPLICATION PROGRAM
MESSAGE CONTROL PROGRAM
MRELEASE
MCPCLOSE
STARTLN
STOPLN
CLOSEMC
RELEASEM
CQTAM only
The user issues one of these macros to enter Operator Control from the application program.
Operator Control / Application Program Interface Routine
IEDQET
• Initialize the Command Input Buffer (CIB) control block with the type of command and other pertinent data.
• Set the SVC 102 parameters necessary to move the CIB to the Process Control Block (PCB) in the MCP.
• Set up the SVC 102 parameters necessary to tpost the CIB to the Operator Control QCB.
• Issue SVC 102 to move and tpost the CIB.
•
Is closedown in progress ?
No
Issue an OS WAIT to allow the MCP to gain control.
Yes
OS
The TCAM Dispatcher processes the elements on the ready queue until the CIB that was tposted to the Operator Control QCB has top priority. At this point, the Dispatcher relinquishes control to the Operator Control task.
OS
OPERATOR CONTROL
Operator Control Control Module
• Process the operator control command indicated in the CIB.
• Store a return code in register 15.
• Use SVC 102 to OS post the ECB for the application program complete
OS
• Continue processing the next sequential instruction in the application program.
LEGEND:
OS Control flow through OS
Control flow through a branch instruction

Chart 24. Application Program Interface with Operator Control

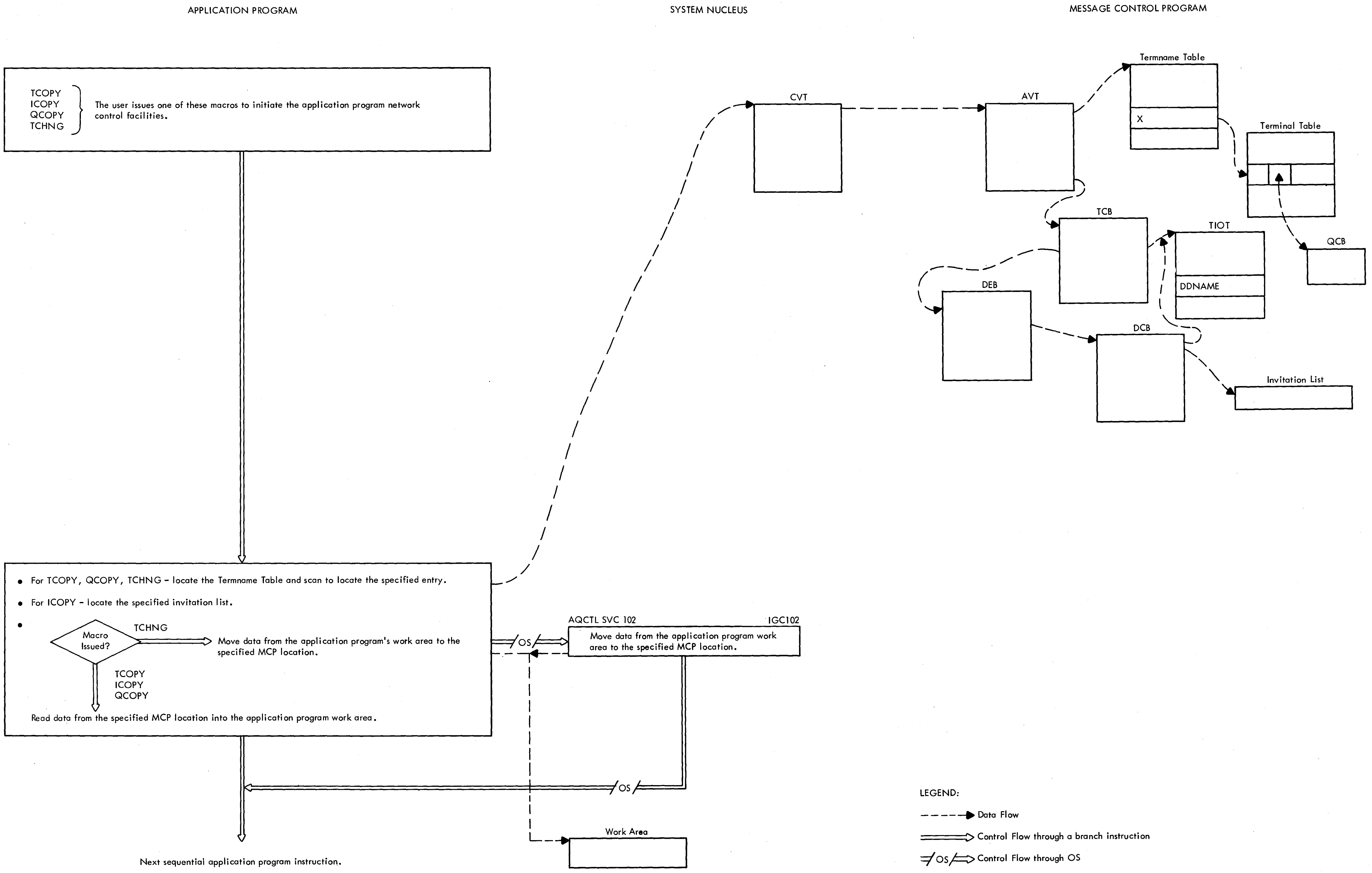

APPLICATION PROGRAM
SYSTEM NUCLEUS
MESSAGE CONTROL PROGRAM
TCOPY
ICOPY
QCOPY
TCHNG
The user issues one of these macros to initiate the application program network control facilities.
CVT
AVT
Termname Table
X
Terminal Table
TCB
TIOT
QCB
DEB
DDNAME
DCB
Invitation List
• For TCOPY, QCOPY, TCHNG – locate the Termname Table and scan to locate the specified entry.
• For ICOPY – locate the specified invitation list.
Macro Issued?
TCHNG
Move data from the application program's work area to the specified MCP location.
TCOPY
ICOPY
QCOPY
Read data from the specified MCP location into the application program work area.
OS
AQCTL SVC 102
IGC102
Move data from the application program work area to the specified MCP location.
OS
Work Area
Next sequential application program instruction.
LEGEND:
Data Flow
Control Flow through a branch instruction
OS
Control Flow through OS

Chart 25. Application Program Network Control Facilities

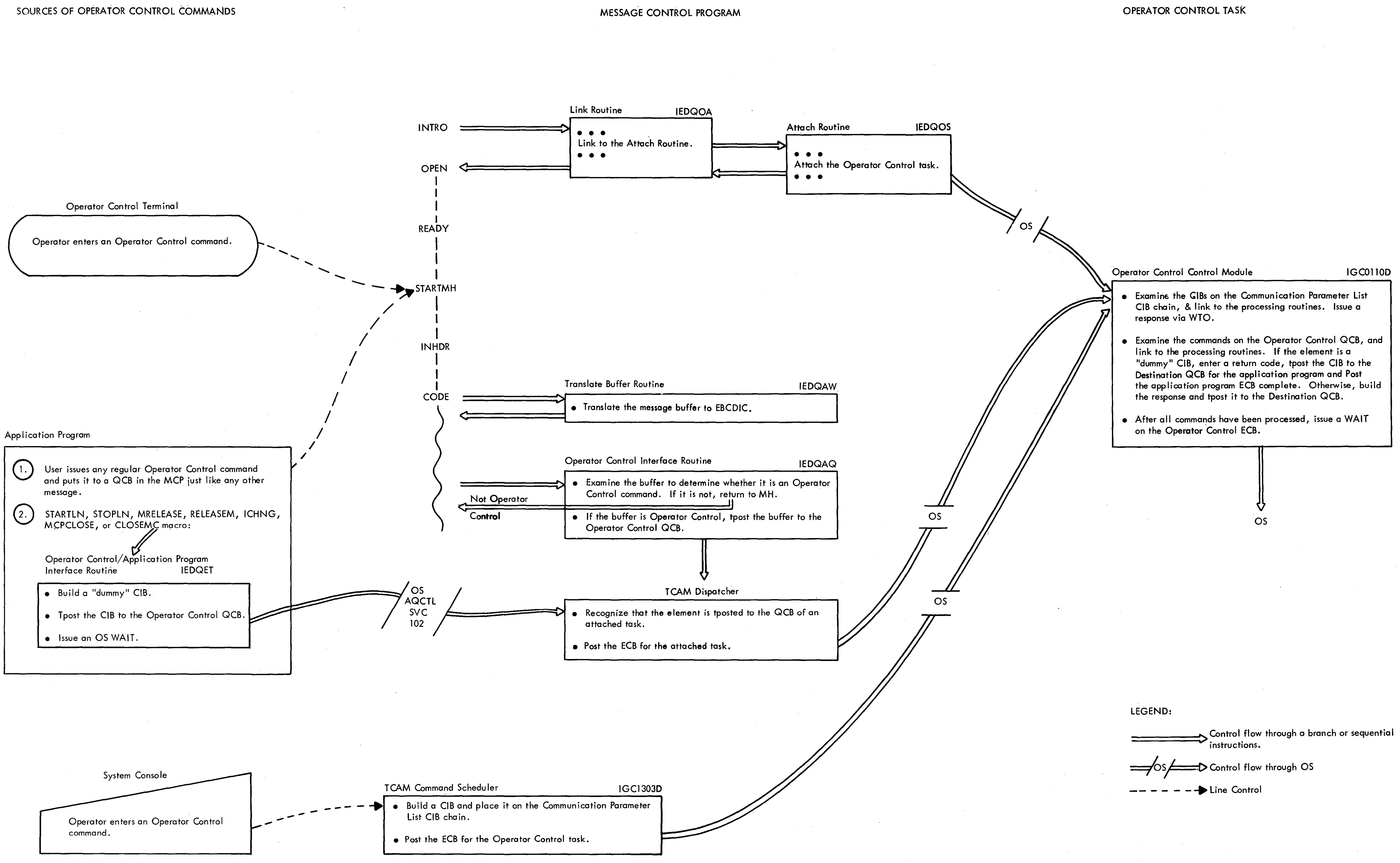
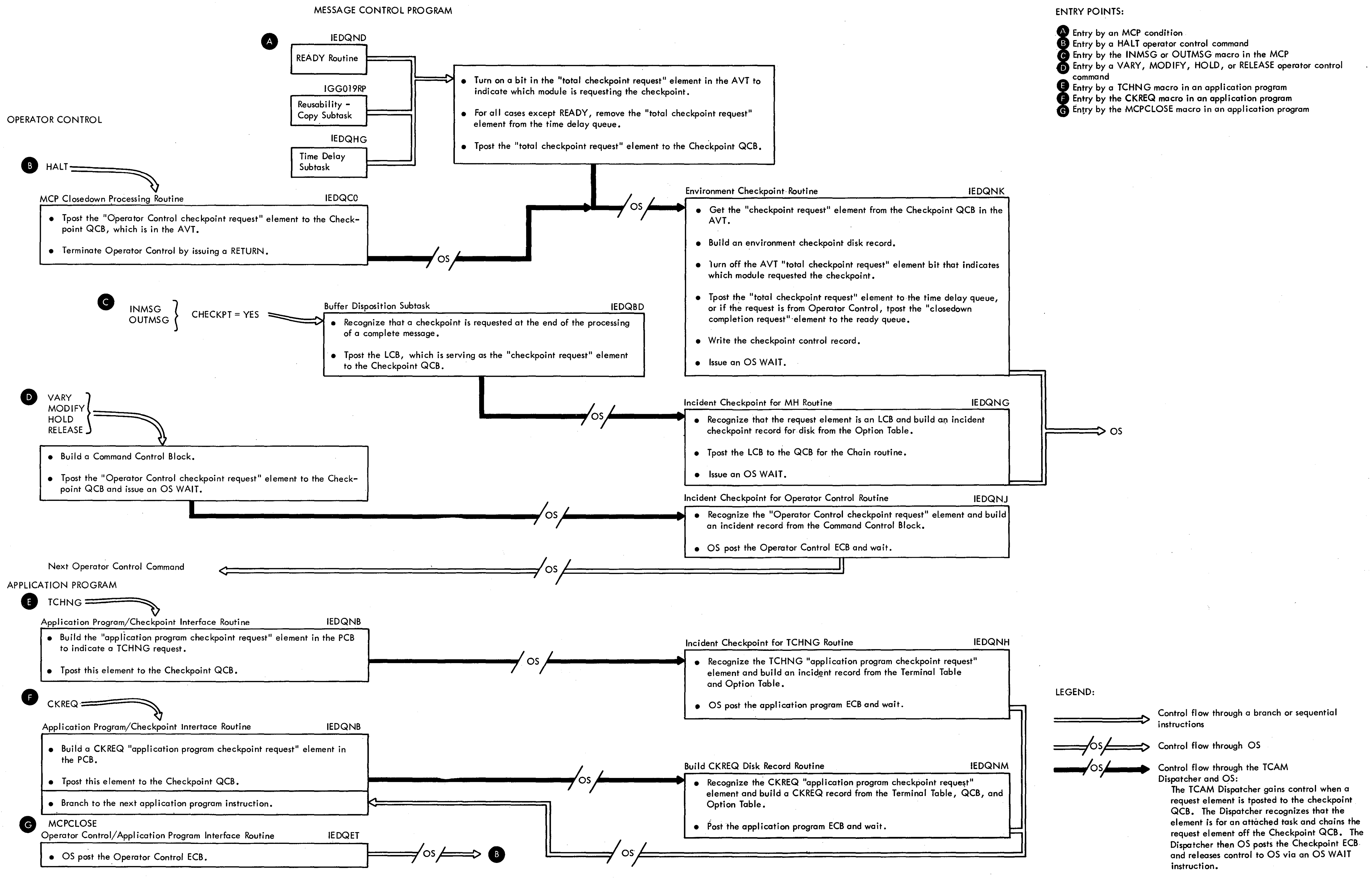

SOURCES OF OPERATOR CONTROL COMMANDS
MESSAGE CONTROL PROGRAM
OPERATOR CONTROL TASK
Link Routine IEDQOA
• • •
Link to the Attach Routine.
• • •
Attach Routine IEDQOS
• • •
Attach the Operator Control task.
• • •
INTRO
OPEN
READY
STARTMH
INHDR
CODE
OS
Operator Control Terminal
Operator enters an Operator Control command.
Operator Control Control Module IGC0110D
• Examine the CIBs on the Communication Parameter List CIB chain, & link to the processing routines. Issue a response via WTO.
• Examine the commands on the Operator Control QCB, and link to the processing routines. If the element is a "dummy" CIB, enter a return code, tpost the CIB to the Destination QCB for the application program and Post the application program ECB complete. Otherwise, build the response and tpost it to the Destination QCB.
• After all commands have been processed, issue a WAIT on the Operator Control ECB.
OS
Translate Buffer Routine IEDQAW
• Translate the message buffer to EBCDIC.
Application Program
1. User issues any regular Operator Control command and puts it to a QCB in the MCP just like any other message.
2. STARTLN, STOPLN, MRELEASE, RELEASEM, ICHNG, MCPCLOSE, or CLOSEMC macro:
Operator Control/Application Program Interface Routine IEDQET
• Build a "dummy" CIB.
• Tpost the CIB to the Operator Control QCB.
• Issue an OS WAIT.
Operator Control Interface Routine IEDQAQ
• Examine the buffer to determine whether it is an Operator Control command. If it is not, return to MH.
• If the buffer is Operator Control, tpost the buffer to the Operator Control QCB.
Not Operator Control
OS
OS
OS AQCTL SVC 102
TCAM Dispatcher
• Recognize that the element is tposted to the QCB of an attached task.
• Post the ECB for the attached task.
System Console
Operator enters an Operator Control command.
TCAM Command Scheduler IGC1303D
• Build a CIB and place it on the Communication Parameter List CIB chain.
• Post the ECB for the Operator Control task.
LEGEND:
Control flow through a branch or sequential instructions.
OS
Control flow through OS
Line Control

Chart 26. Functional Flow of Operator Control



ENTRY POINTS:

A Entry by an MCP condition
B Entry by a HALT operator control command
C Entry by the INMSG or OUTMSG macro in the MCP
D Entry by a VARY, MODIFY, HOLD, or RELEASE operator control command
E Entry by a TCHNG macro in an application program
F Entry by the CKREQ macro in an application program
G Entry by the MCPCLOSE macro in an application program

- Turn on a bit in the "total checkpoint request" element in the AVT to indicate which module is requesting the checkpoint.
- For all cases except READY, remove the "total checkpoint request" element from the time delay queue.
- Tpost the "total checkpoint request" element to the Checkpoint QCB.

MCP Closedown Processing Routine IEDQCO

- Tpost the "Operator Control checkpoint request" element to the Checkpoint QCB, which is in the AVT.
- Terminate Operator Control by issuing a RETURN.

Environment Checkpoint Routine IEDQNK

- Get the "checkpoint request" element from the Checkpoint QCB in the AVT.
- Build an environment checkpoint disk record.
- Turn off the AVT "total checkpoint request" element bit that indicates which module requested the checkpoint.
- Tpost the "total checkpoint request" element to the time delay queue, or if the request is from Operator Control, tpost the "closedown completion request" element to the ready queue.
- Write the checkpoint control record.
- Issue an OS WAIT.

Buffer Disposition Subtask IEDQBD

- Recognize that a checkpoint is requested at the end of the processing of a complete message.
- Tpost the LCB, which is serving as the "checkpoint request" element to the Checkpoint QCB.

Incident Checkpoint for MH Routine IEDQNG

- Recognize that the request element is an LCB and build an incident checkpoint record for disk from the Option Table.
- Tpost the LCB to the QCB for the Chain routine.
- Issue an OS WAIT.

- Build a Command Control Block.
- Tpost the "Operator Control checkpoint request" element to the Checkpoint QCB and issue an OS WAIT.

Incident Checkpoint for Operator Control Routine IEDQNJ

- Recognize the "Operator Control checkpoint request" element and build an incident record from the Command Control Block.
- OS post the Operator Control ECB and wait.

Application Program/Checkpoint Interface Routine IEDQNB

- Build the "application program checkpoint request" element in the PCB to indicate a TCHNG request.
- Tpost this element to the Checkpoint QCB.

Incident Checkpoint for TCHNG Routine IEDQNH

- Recognize the TCHNG "application program checkpoint request" element and build an incident record from the Terminal Table and Option Table.
- OS post the application program ECB and wait.

Application Program/Checkpoint Interface Routine IEDQNB

- Build a CKREQ "application program checkpoint request" element in the PCB.
- Tpost this element to the Checkpoint QCB.
- Branch to the next application program instruction.

Build CKREQ Disk Record Routine IEDQNM

- Recognize the CKREQ "application program checkpoint request" element and build a CKREQ record from the Terminal Table, QCB, and Option Table.
- Post the application program ECB and wait.

Operator Control/Application Program Interface Routine IEDQET

- OS post the Operator Control ECB.

LEGEND:

Control flow through a branch or sequential instructions

Control flow through OS

Control flow through the TCAM Dispatcher and OS:
The TCAM Dispatcher gains control when a request element is tposted to the checkpoint QCB. The Dispatcher recognizes that the element is for an attached task and chains the request element off the Checkpoint QCB. The Dispatcher then OS posts the Checkpoint ECB and releases control to OS via an OS WAIT instruction.

Chart 27. Functional Flow of the Checkpoint Routines

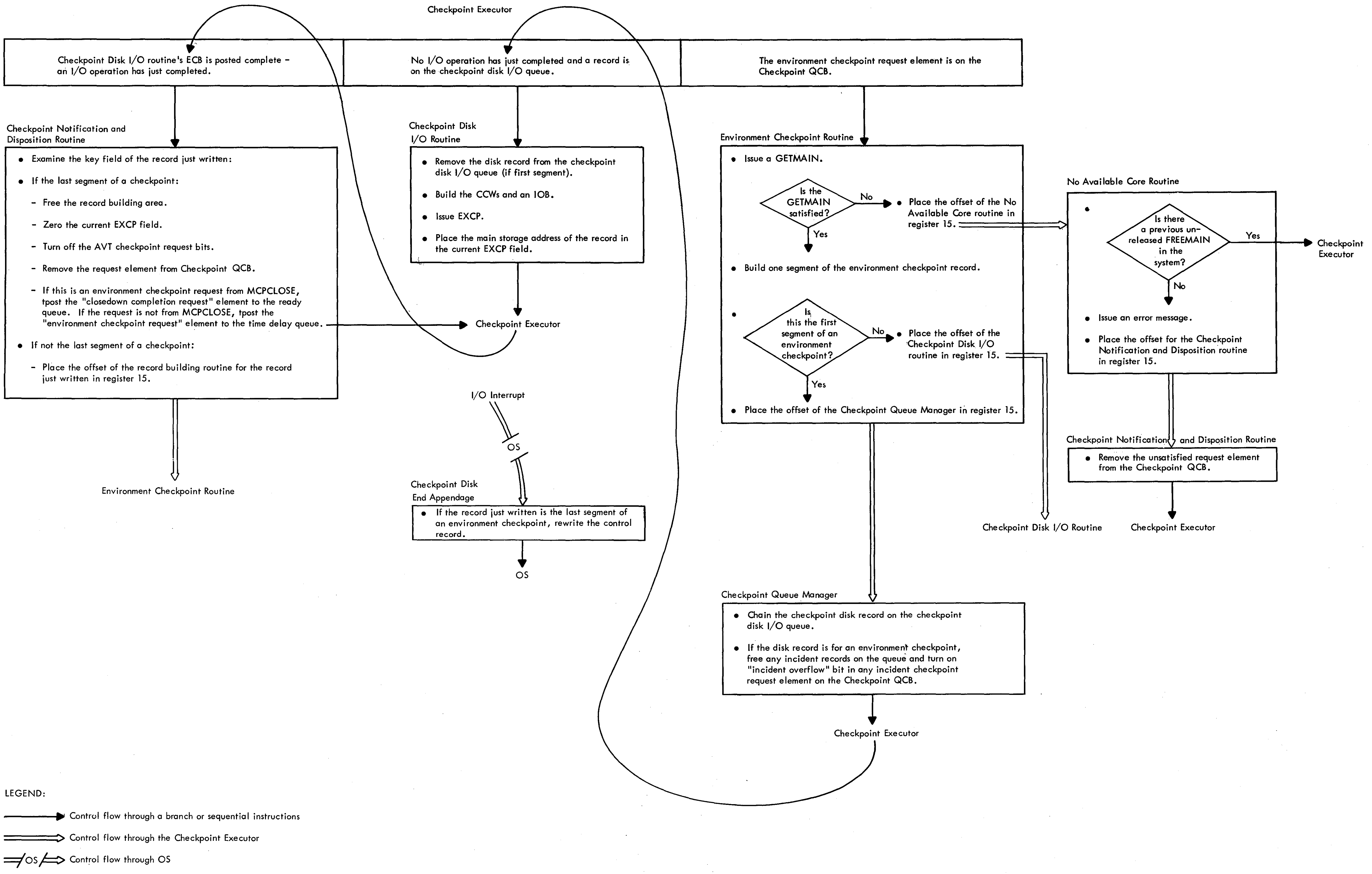

Checkpoint Executor
Checkpoint Disk I/O routine's ECB is posted complete - an I/O operation has just completed.
No I/O operation has just completed and a record is on the checkpoint disk I/O queue.
The environment checkpoint request element is on the Checkpoint QCB.
Checkpoint Notification and Disposition Routine
• Examine the key field of the record just written:
• If the last segment of a checkpoint:
- Free the record building area.
- Zero the current EXCP field.
- Turn off the AVT checkpoint request bits.
- Remove the request element from Checkpoint QCB.
- If this is an environment checkpoint request from MCPCLOSE, tpost the "closedown completion request" element to the ready queue. If the request is not from MCPCLOSE, tpost the "environment checkpoint request" element to the time delay queue.
• If not the last segment of a checkpoint:
- Place the offset of the record building routine for the record just written in register 15.
Environment Checkpoint Routine
Checkpoint Disk I/O Routine
• Remove the disk record from the checkpoint disk I/O queue (if first segment).
• Build the CCWs and an IOB.
• Issue EXCP.
• Place the main storage address of the record in the current EXCP field.
Checkpoint Executor
I/O Interrupt
OS
Checkpoint Disk End Appendage
• If the record just written is the last segment of an environment checkpoint, rewrite the control record.
OS
Environment Checkpoint Routine
• Issue a GETMAIN.
Is the GETMAIN satisfied?
No
• Place the offset of the No Available Core routine in register 15.
Yes
• Build one segment of the environment checkpoint record.
•
Is this the first segment of an environment checkpoint?
No
• Place the offset of the Checkpoint Disk I/O routine in register 15.
Yes
• Place the offset of the Checkpoint Queue Manager in register 15.
No Available Core Routine
•
Is there a previous unreleased FREEMAIN in the system?
Yes
Checkpoint Executor
No
• Issue an error message.
• Place the offset for the Checkpoint Notification and Disposition routine in register 15.
Checkpoint Notification and Disposition Routine
• Remove the unsatisfied request element from the Checkpoint QCB.
Checkpoint Disk I/O Routine
Checkpoint Executor
Checkpoint Queue Manager
• Chain the checkpoint disk record on the checkpoint disk I/O queue.
• If the disk record is for an environment checkpoint, free any incident records on the queue and turn on "incident overflow" bit in any incident checkpoint request element on the Checkpoint QCB.
Checkpoint Executor
LEGEND:
Control flow through a branch or sequential instructions
Control flow through the Checkpoint Executor
OS Control flow through OS

Chart 28. Control Flow of the Environment Checkpoint Routines

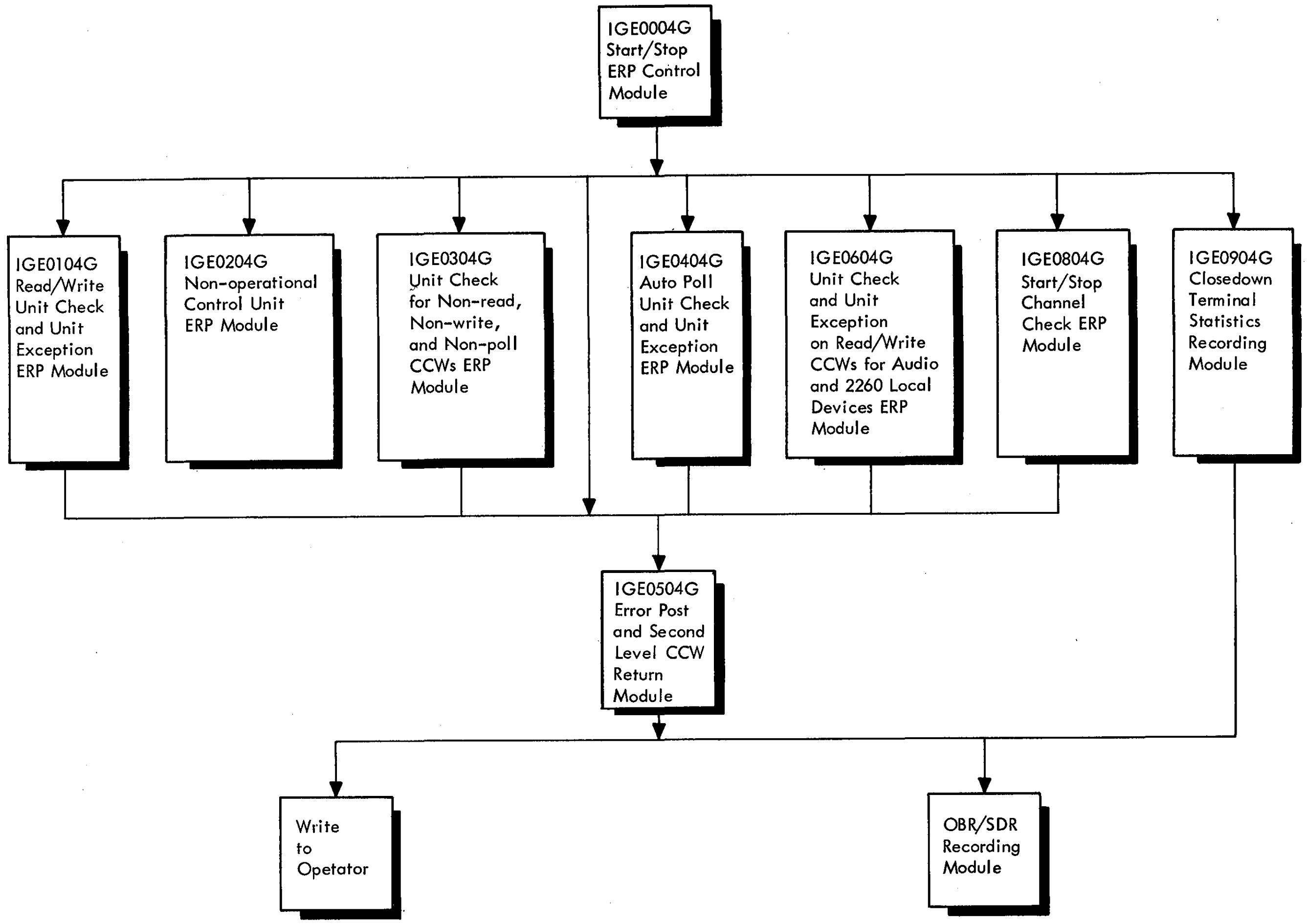

IGE0004G Start/Stop ERP Control Module
IGE0104G Read/Write Unit Check and Unit Exception ERP Module
IGE0204G Non-operational Control Unit ERP Module
IGE0304G Unit Check for Non-read, Non-write, and Non-poll CCWs ERP Module
IGE0404G Auto Poll Unit Check and Unit Exception ERP Module
IGE0604G Unit Check and Unit Exception on Read/Write CCWs for Audio and 2260 Local Devices ERP Module
IGE0804G Start/Stop Channel Check ERP Module
IGE0904G Closedown Terminal Statistics Recording Module
IGE0504G Error Post and Second Level CCW Return Module
Write to Opetator
OBR/SDR Recording Module

Chart 29. Linkage among the Start-Stop ERP Modules

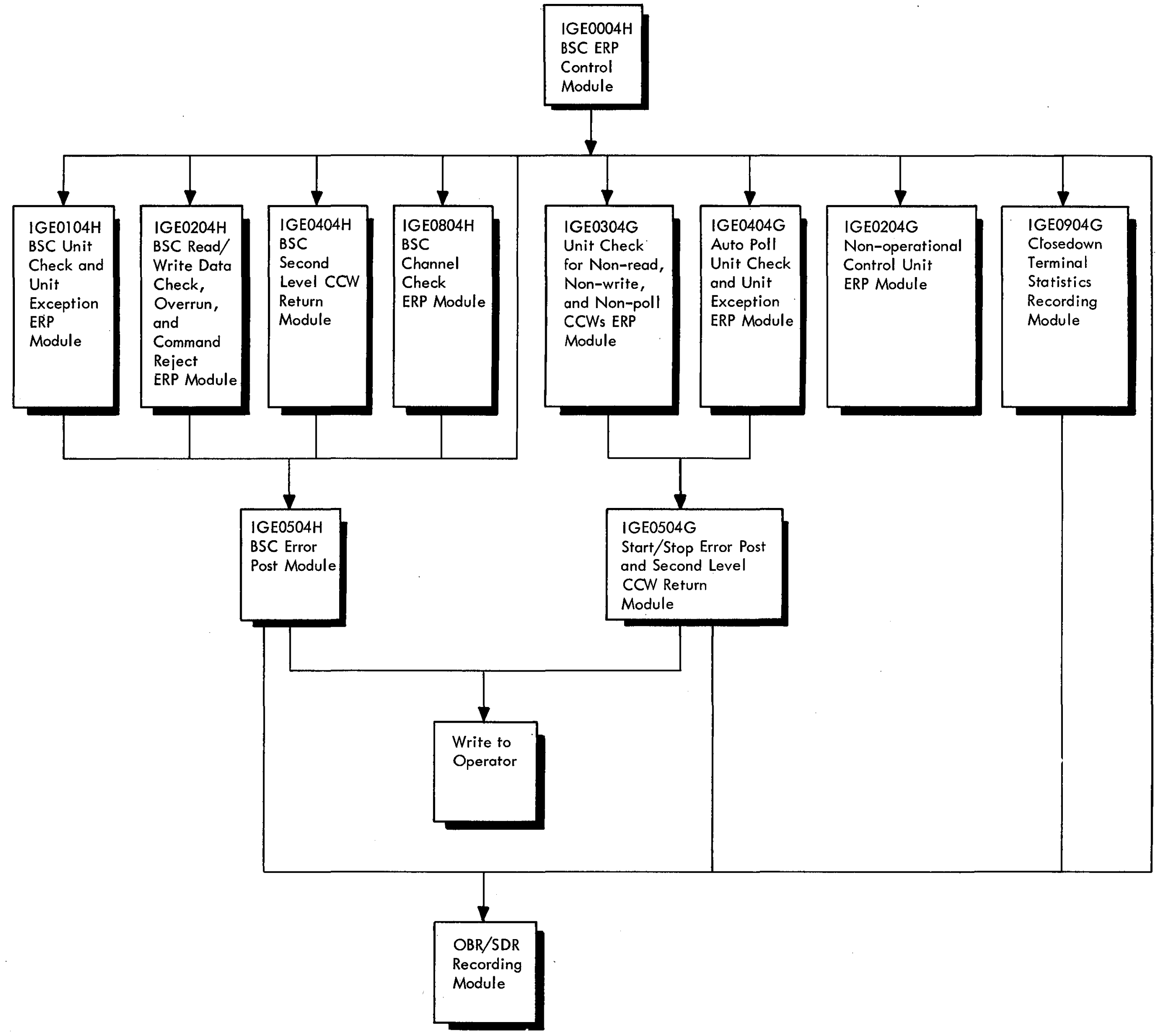

IGE0004H BSC ERP Control Module
IGE0104H BSC Unit Check and Unit Exception ERP Module
IGE0204H BSC Read/ Write Data Check, Overrun, and Command Reject ERP Module
IGE0404H BSC Second Level CCW Return Module
IGE0804H BSC Channel Check ERP Module
IGE0304G Unit Check for Non-read, Non-write, and Non-poll CCWs ERP Module
IGE0404G Auto Poll Unit Check and Unit Exception ERP Module
IGE0204G Non-operational Control Unit ERP Module
IGE0904G Closedown Terminal Statistics Recording Module
IGE0504H BSC Error Post Module
IGE0504G Start/Stop Error Post and Second Level CCW Return Module
Write to Operator
OBR/SDR Recording Module

Chart 30. Linkage among the BSC ERP Modules

Access method (ACSMETH) work area: a storage space in an application program. This work area contains data necessary for the interface between the application program and the MCP.

Additional records: the units, other than the first unit, of a buffer that is being placed on a message queue.

Address: in reference to disk queuing, this term refers to the disk relative record number used to queue a unit of a message segment. In reference to main storage queuing, the address is the actual location of a unit of a message segment.

Address Vector Table (AVT): in TCAM , a local constant area.

Application program: a program that processes the text portions of messages. Application programs run asynchronously with the message control program, and are usually located in another partition or region.

Attached task: a unit of work that is created by another task by means of the ATTACH macro and that competes independently for control of the CPU. In TCAM, Checkpoint, Operator Control, On-Line Test, and FE Common Write are attached tasks.

Auto Poll: a hardware feature or a TCU that processes an invitation list, polling the terminals in order and handling negative responses to polling without interrupting the CPU. At the end of the list, polling is resumed at the beginning of the list.

Binary synchronous (BSC) line: a line on which the data transmission or character synchronism is controlled by timing signals generated at the sending and receiving stations.

Block: that portion of a message terminated by line control characters, EOB, ETB, ETX, or EOT.

Buffer: a main storage area into which a message segment is read or from which a message segment is written. A buffer is a temporary holding area that is used to compensate for a difference in the rate of flow of information between input/output devices and the CPU. The size of the buffers is designated by the user. In TCAM, a buffer is made up of one or more units, in which the total number of bytes satisfy the user-specified buffer size.

Buffer prefix: a control area at the beginning of each buffer. The first buffer of a message in TCAM has a 30-byte prefix, while the prefix for each subsequent buffer of a message is 23 bytes long. A buffer prefix is contained within the userspecified buffer length and TCAM places control information in the prefix.

Buffer unit: see unit.

Buffer unit pool: a chain of all the buffer units that are not currently being used by TCAM. This chain is the element chain of the Buffer Request QCB. At assembly time, the buffer unit pool contains the number of units that is equal to the sum of the integers specified by the LNUNITS and MSUNITS operands of the INTRO macro.

Buffered terminal: a terminal that has a hardware buffer. When the user specifies the BFDELAY operand of a TERMINAL macro for a buffered terminal, the MCP sends messages to that terminal, segment-by-segment. After a segment is sent, the MCP pauses to allow the terminal to empty its buffer. During this pause, the MCP may send message segments to the other stations on the line.

Calling: a procedure by which a first party attempts to establish a connection with a second party through a central exchange. Also termed dialing.

Cascade entry: a Terminal Table block of information that is associated with a cascade list.

Cascade list: a list of pointers to single, group, or process entries. TCAM queues message for the first valid entry with the fewest messages queued for it in the list.

Channel Command Word (CCW): a doubleword definition of an operation to be performed by the I/O channel. One or more CCWs constitute a channel program.

Channel Program Block (CPB): a control area that contains an I/O channel program and a pointer to the buffer that it is to process. A CPB is used in the transfer of data between buffer units and disk message queues. The CPB operand of the INTRO macro specifies the number of CPBs in the TCAM system.

Checkpoint data set: a set of checkpoint records that are maintained and stored on a direct access storage device. When the Checkpoint/Restart facility is used, TCAM uses these records to restructure the MCP environment after closedown or system failure.

Checkpoint request record: a checkpoint taken in response to the issuance of a CKREQ macro in an application program. This record contains the status of a single destination queue for the application program issuing the macro. During restart TCAM uses the latest checkpoint request record for a message queue to start sending the application program the message following the last message sent at the time that the checkpoint request record was written.

Checkpoint/Restart: a TCAM facility that records the status of the telecommunications network at designated intervals or following certain events. Following system failure or closedown, this facility uses the records it has taken to restart the system without loss of messages.

CKREQ checkpoint record: see Checkpcint request record.

Cold start: a TCAM MCP start-up in which TCAM reinitializes the system.

Command Input Buffer (CIB): a communication parameter list that is used by Operator Ccntrol to process a command. It describes the command sent from the console and ccntains the ccmmand code, the console identification, and the data in the command.

Communication Parameter List: the interface between TCAM Operator Control and the Master Scheduler fcr commands entered from the system console.

Communication Vector Table (CVT): a part cf the resident nucleus in the Operating System (OS) that provides the means whereby nonresident routines may refer to information in the nucleus of the control program.

Compatible QTAM: the ability to operate a QTAM application program under a TCAM MCP.

Component: a point in a communications system at which data can enter or leave; an input/output device. A compcnent is always attached to a terminal control unit.

Continuation restart: a TCAM MCP restart that follcws termination of the MCP due to system failure. The TCAM Checkpoint/Restart facility restores the MCP environment as nearly as pcssible to its condition before system failure.

Control record (checkpoint): a disk checkpcint data set record that contains information about the format of the checkpcint data set.

CPB free pool: a chain of the CPBs that are not currently in use by the TCAM MCP.

Cross-partition data movement: the situation in which data is moved from one partition or region in main storage tc another.

Data Control Block (DCB): an area of main storage that serves as a logical connector between the problem program and a data set. The data control blcck can also be used to provide control information for any transfer of data. In TCAM a DCB must be specified for each TCAM data set except a main-storage message queues data set; a DCB macro is used to create a DCB.

Data Event Control Block (DECB): a control block that contains information abcut an input or an cutput operation requested by a READ or WRITE macro instructicn.

Data Extent Block (DEB): a control block that describes the extents of the data set with which it is associated.

Data Set Control Block (DSCB): a collection of information that describes the attributes of a data set in direct-access storage.

Dead-letter queue: the destination queue for the station or application program named by the DLQ operand of the INTRO macro. If TCAM detects an invalid destination in a message header and no user exit is specified in the FORWARD macro, TCAM sends the message to the dead-letter queue.

Delimiter macro instruction: a TCAM macro that classifies and identifies sequences of functional macro instructions and directs control to the appropriate sequence of functional macro instructions.

Destination: a place to which a message being handled by a TCAM Message Handler is to be sent. A destination may be either a station defined by a TERMINAL macro, a group of stations defined by a TLIST macro, or an application program defined by a TPROCESS macro. One or more destinations may be specified in a message header, or a single destination may be specified for all messages handled by an Inheader Subgroup.

Destination offset: a two-byte index to the Termname Table entry of a destination or station.

Destination queue: a chain of message segments that are to be sent to a specific terminal, group of terminals, or application program. TCAM builds this queue after the message segments are processed by the Incoming Group of an MH.

Device Characteristics Table (DCT): a collection of entries that describes the characteristics of the terminals (or devices) in the system.

Dial-out queue: a chain of QCBs each of which is for a dial terminal to which a message has been tposted when the line for the terminal is unavailable.

Disabled ready queue: a chain of elements, in FIFO order, that are to be processed by TCAM and that are from the disabled appendages.

Disk queuing: the process of maintaining the TCAM message queues on a direct access storage device.

Dispatching: the process of providing a routine with an element and giving the routine control to process the element.

Distribution entry: a Terminal Table block of information that is associated with a distribution list.

Distribution list: a group of terminals, each of which is to receive any message directed to the group.

Duplicate-header message: a message that is identical to the one sent previously, as in multiple routing.

Element: an individual part of a system resource; for example, a buffer.

Element Request Block (ERB): a control area that is used to make requests for buffers for a line group.

Enabled ready queue: a chain of elements in priority-FIFO order that are to be processed by TCAM and that are from the enabled TCAM modules.

Enabling a line: the process of conditioning the transmission control unit to accept incoming calls on a line.

End of address (EOA) character:

1. A control character or characters transmitted on a line to indicate the end of non-text characters (for example, addressing characters).
2. A TCAM character that must be placed in a message if the system is to accomodate routing of that message to several destinations; the character must immediately follow the last destination code in the message header, and must also be specified by the EOA operand of the FORWARD macro for the message.

Environment checkpoint record: a record that contains information on the total TCAM operating environment at a single point in time. At restart time, TCAM updates the environment record with the contents of more recent incident checkpoint records in order to reconstruct the MCP environment as it existed before closedown or system failure.

Error Recovery Procedure (ERP): a set of TCAM routines that attempt to recover from transmission errors.

Event Control Block (ECB): the communication medium between the various components of the control program, as well as between processing programs and the control program. An ECB is the subject of WAIT and POST macro instructions.

EXCP: execute channel program.

EXCP queue: a chain of one CPB for the cylinder that is currently ready for I/O operations in one extent of a disk message queues data set.

FEFO: first-ended-first-out.

FEFO queuing: a situation in which messages that ended (EOT received) first are sent before messages that began transmission first. TCAM provides FEFO queuing within priority groups. That is, TCAM sends higher-priority messages before lower-priority messages. When two messages on a queue have equal priority, TCAM sends the message, the last segment of which was received first.

FIFO: first-in-first-out.

FIFO queuing: the situation in which equal-priority messages on a destination queue are sent out in the order in which their first segments arrived on the queue.

First buffer prefix: a 30-byte control area at the beginning of the first buffer of a message.

Flush closedown: a TCAM MCP closedown during which incoming message traffic is suspended and queued outgoing messages are sent before closedown completed. That is, unsent messages are "flushed" from the message queues.

Functional macro instructions: TCAM macros that perform the specific operations required for messages directed to the Message Handler (see delimiter macro instructions).

Functional routine: a routine that is associated with a specific MH macro and that is activated from the expansion of that macro.

Functional subroutine: a routine or subroutine that is activated by a functional routine.

"Good Morning" message: a user-generated message (through an exit specified in the READY macro) to be sent to all or to selected terminals (user-determined) for cold starts. This message is used to notify a terminal operator or operators that the TCAM system is up and running.

Group entry: a Terminal Table block of information that is associated with a group of terminals that have the group-code hardware feature.

Header buffer: a buffer that contains all or any part of a message header.

Held terminal: a terminal that cannot accept messages because of the effect of a HOLD macro.

Hold queue: a chain of messages sent to a terminal or terminals that are not currently accepting messages because a HOLD macro was issued in MH for this terminal.

Idle character: a character that is transmitted on a line and that does not print or punch at the output component of the accepting terminal.

Incident checkpoint record: a disk checkpoint data set record of a specific event or incident during TCAM operation. An incident record logs a change in station or line status or in the contents of an option field. At restart time, TCAM uses incident records to update the information in the environment checkpoint record. These checkpoint records are written as a result of operator control commands or TCHNG, ICHNG, or CHECKPT macros.

Incoming group: that portion of a Message Handler that is designed to handle messages arriving fcr processing by the message control program (see Outgoing Group).

Incoming message: a message that is being transmitted from a station to the computer.

Initiate mode: causes message segments to be sent from a destination queue to the proper destinaticn as soon as possible after they are placed on the queue. Normally the segments are not sent until a full message is on the queue. Initiate mode is provided by a functional macro instruction in an incoming MH.

Input: of or related to a message transmission that involves entering data at a terminal or receiving data at the computer.

In-source chain: a chain off the QCBINSRC field of a Destination QCB of all source LCBs that are currently sending initiate mode messages to the associated station.

Invitation: the process in which the computer makes contact with a terminal in order to give the terminal the cpportunity to transmit a message (if it has one ready). Polling and enabling are forms of invitation.

Invitation characters: see Polling characters.

Invitation List: a sequence of polling characters or identification sequences associated with the terminals on a line. The order in which the characters are specified determines the crder in which the terminals are invited to enter a message.

I/O Block (IOB): the communicaticn medium between a routine that requests an I/O operaticn and the I/O Supervisor. All the information required by the I/O Supervisor to execute an I/O operation is contained in the IOB, or is pointed to by the IOB.

I/O Supervisor (IOS): an Operating System task that controls all the I/O operations in the system.

Job Control Language (JCL): a collecticn of statements used to identify a job and its requirements to the Cperating System.

Job File Control Block (JFCB): a system control block constructed by job management routines to contain informaticn about a specific data set in the system. There is one JFCB fcr each DCB that is opened. The information in the JFCB may be modified during open time.

Line control: the scheme cf operating procedures and control signals by which a telecommunications system is controlled.

Line Control Block (LCB): an area of main storage that contains control informaticn for operations on a line. TCAM maintains one LCB for each line in the system.

Line group: a set of one or more communications lines of the same type, over which terminals with similar characteristics can communicate with the computer.

Locate mode: the manner in which a record is given to (or taken from) the user's program where TCAM provides the work area for a work unit and passes the address of the area to the user in register 1, and the user may work cn the work unit in place.

Log: a collection of messages that prcvides a history of message traffic; either message segments or complete messages can be logged.

Logging: the process of recording messages on a storage medium to maintain a histcry of message traffic for accounting or other purposes.

Logical buffer: see buffer.

Logtype entry: a Terminal Table block of informaticn that is associated with a queue for logging ccmplete messages.

Macro expansion: the assembler-generated output from an instruction in a source language.

Main-storage queuing: a situation in which TCAM message queues are maintained in main storage.

Master QCB: the basic format cf a Destination QCB - 40 bytes of destination specific data.

MCPL: an STCB entry code field that identifies the type of STCB; therefore, the method necessary to activate the corresponding subtask.

Message: a combination of letters, digits, and symbols whose termination point is marked by:

1. an end of transmission character (EOT) for start-stop devices;
2. an end of transmission sequence (ETX EOT) for BSC devices; or
3. if end-of-block checking is specified by the CONV operand of the STARTMH macro, by an ETX, ETB, or EOB character.

Message Contrcl Program (MCP): a series of TCAM routines that identify the telecommunications network to the System/360 Operating System, establish line ccntrcl, and handle and route messages.

Message data: transmitted characters that are recorded as part of a message. A message data area is the area in a buffer that receives message data.

Message Handler (MH): a sequence of user-specified macro instructions that examine and process the control information in message headers, and perform functions necessary to prepare message segments for forwarding to their destinations. One Message Handler is required for each line group that has special message-handling requirements.

Message header: the part of a message containing control information, such as the destination code (as distinct from the text of the message).

Message log data set: a set of messages or message segments that are maintained on secondary storage for accounting or other purposes.

Message queue: a chain of messages for a destination (a line, terminal, application program, or logging medium).

Message queues data set: a collection of the chains of messages for the destinations that are designated to be queued in the same manner; that is, on reusable disk, on nonreusable disk, or in main storage.

Message retrieval function: allows the user to retrieve a previously sent message by specifying a combination of the message destination and the input (or output) sequence number of the message. The sequence number is assigned by the SEQUENCE macro.

Message segment: that portion of a message that fits in the message data area of a buffer.

Message switching: a telecommunications application in which a message is received at a central location, possibly stored until the appropriate time, and then transmitted to its destination.

MFT: multiprogramming with a fixed number of tasks.

Module: a program unit (with one or more entry points) that is discrete and identifiable with respect to compiling, combining with other units, and loading.

Modulo: the remainder after any division has been performed. In TCAM the absolute record number modulo the total number of records is equal to the relative record number.

Multidrop terminal: a terminal on a multipoint line

Multiple routing: the method of sending a message where more than one destination is specified in the header of the message.

Multiple-buffer header: a message header that occupies more than one buffer.

MVT: multiprogramming with a variable number of tasks.

Network control: the management of a series of points interconnected by communications channels.

Next-buffer location: the value of address (disk relative record number) to be used for the first unit of the next buffer of the message that is currently being placed on the related message queue.

Next-message location: the value of address (disk relative record number) to be used for the first unit of the first buffer of the next message received for the related message queue.

New queue: a chain of CPBs for all cylinders in an extent of a disk message queues data set other than the cylinder currently ready for I/O and the cylinder just after it.

No-buffer queue: the chain of Channel Program Blocks (CPBs) for READ operations when no buffers are in the buffer pool.

No-CPB queue: the chain of elements that are to be processed by CPB initialization.

Nonreusable disk queuing: the situation in which each reccrd of a disk message queues data set may be used only cnce.

Operator-awareness message: an unsolicited status message from the TCAM system to the primary operator contrcl terminal operator. This is to make the operator aware of scme potential problem in TCAM.

Operator Control: a TCAM facility that allows the system operator to issue commands to examine or alter the status of his telecommunicaticns network.

Operator Control Address Vector Table (AVT): an MCP area that contains parameters for the Operator Ccntrcl control module.

Option Table: a collection of informaticn provided by the user in OPTION macro instructions.

Outgoing group: that portion of the message handler that processes messages beinc sent from the message ccntrol prcgram to any of the lines, line grcups, or application programs (see Incoming Group).

Outgoing message: a message that is being sent from the message control program to its destination.

Output: of or related to a message transmission that involves accepting data at a terminal or sending data from the computer.

Path switch: an cption field setting used as a switch to indicate the order of or the ccnditional execution of MH macros.

Polling: a flexible, systematic, centrally controlled method of permitting terminals tc transmit without ccntending for the line. The computer contacts terminals according to the crder specified in the invitation list; each terminal contacted is invited to send messages.

Polling characters: a set of characters peculiar to a terminal and the polling operaticn; response to these characters indicates to the computer whether the terminal has a message to enter.

Post: a signal of the completion of an event in the Operating System.

Priority QCB: an area in which the queuing data for a given priority level of a message for a Destination QCB is stored.

Process Control Block (PCB): an MCP storage area for data that is necessary for communication between the MCP and an application program.

Process entry: a Terminal Table block of information that is associated with an application program.

Process entry work area: a work area in the MCP. It contains data pertinent to the presence of an application interface with the MCP.

Program-Controlled Interruption (PCI): an interruption caused by the channel when starting the execution of a CCW with the PCI flag set. This interruption is used in TCAM to notify TCAM of pending or completed data transfer between TCAM and a particular terminal or terminals. This knowledge is used for buffer and data handling.

Purge I/O: an SVC issued at close time to remove all traffic from teleprocessing lines.

Queue: a chain of items waiting for service by the system.

Queue-back chain: a time sequential record of the sending and receiving message traffic for the terminal or terminals of a specific Destination QCB.

Queue Control Block (QCB): a storage area used to associate elements with appropriate subtasks.

Quick closedown: a TCAM MCP closedown that entails stopping the message traffic on each line as soon as any message being sent or received at the time of the closedown request is transmitted.

Read-ahead QCB: the queue control block used by TCAM as an intermediate step between the Destination QCB for an application program and an application program request for data. In TCAM it is used to anticipate a request for data and to avoid the overhead required in retrieving the data from the Destination QCB at the time of the request.

Ready queue: a chain of elements that represent the work to be performed in the TCAM system.

Recall: a method of retrieving a particular message or a part of a message in order to reprocess it or to redirect it.

Recalled-header buffer: the first buffer of header information for a recalled message

Region Control Task (RCT): a TSO task that determines which task is to occupy a particular TSO region. There is one RCT for each region. The RCT is activated by the TSIP SVC.

Relative line number (rln): number of a line in the line group relative to all others in the line group.

Resource: any system facility that is required by a job or task; for example, main storage, I/O devices, data sets, buffer pool.

Resource Control Block (RCB): an eight-byte prefix to an element.

Restart: to restructure the execution of a routine or system, using the data recorded at a checkpoint.

"Restart in Progress" message: a user-generated message (through an exit specified on the READY macro) to be sent to all or selected terminals (user-determined). This message is used to notify a terminal operator or operators that the TCAM system is up and running.

Retrieve mode: the method of operation used during the time the GET Scheduler is recalling buffers to satisfy a retrieve request by the application program.

Retry: an error recovery procedure in which the current block of data is re-sent a prescribed number of times or until accepted.

Retry queue: a chain of one CPB for the cylinder on which to have I/O in an extent of a disk message queues data set after the CPBs on the EXCP queue are processed.

Reusable disk queuing: a situation in which messages are queued to a wrapped message queues data set; that is, serviced messages are overlaid by new messages entering the system.

Rollout/Rollin (RORI): an optional feature of the MVT control program configuration that enables an additional region (or regions) of main storage to be temporarily reassigned from one job step to another.

Routine: an ordered set of instructions with a single entry point.

Save area: a block of main storage that is used to hold certain data (for example, register values) while the location in which the data was originally stored is used for other purposes.

Secondary destination: a destination to which a message is to be sent if the primary destination is unable to accept the message.

Segment: that portion of a message contained in a single buffer.

Selection: the process by which the computer makes contact with a terminal in order to send it a message (includes addressing and, for switched lines, calling).

Sequential Access Method (SAM): a program that performs I/O operations on a data set one record at a time from beginning to end.

Single entry: a Terminal Table block of information that is associated with one terminal or terminal component.

Source offset: the index value intc the Termname Table for the source terminal.

Special Characters Table (SCT): a cclection of entries that contain the special characters required for device I/O for each terminal (or device) in the system.

Start-stop line: a line on which each character of data transmission is preceded by a special control signal that indicates the beginning of the sequence of data bits for a character. Each character is followed by ancther control signal that indicates the end of the data bit sequence.

Station: a computer or a terminal.

Station Control Block (SCB): a logical extension of the QCB for each station. The SCB contains informaticn used by TCAM to control buffering.

Subsequent buffer prefix: a 23-byte contrcl area at the beginning of each buffer of a message after the first buffer.

Subtask Control Block (STCB): a storage area used to contain the information necessary to activate a particular routine.

Subtask: a task that is created by ancther task by means of the ATTACH macro instruction.

Task Control Block (TCB): The consolidation of control information related to a task.

Task I/O Table (TIOT): a control block ccnstructed by job management to provide I/O suppcrt routines (OPEN, CIOSE, EOV) with pointers to JFCBs and allccated devices.

TCAM/TSO buffer: a buffer residing in the TCAM region in which the PRFTSBUF bit in the Buffer Prefix is cn indicating that the buffer contains a TSO message.

Telecommunicaticns: any transmissicn or reception of signals, writing, sounds, or intelligence cf any nature, by wire, radio, visual methods, or electromagnetic systems. Often used interchangeably with "communications." Synonym: teleprocessing.

Telecommunicaticns Access Method (TCAM): a high-level access method that controls data transfer between main stcrage and remote stations.

Terminal: a point in a system at which data can enter, leave, or enter and leave. A terminal can also be a control unit to which cne or more input/output devices can be attached (see Ccmpcnent).

Terminal entry: a single block of device-dependent information in the Terminal Table on a terminal, grcup of terminals, or application program.

Terminal I/O Coordinator (TIOC) : the interface between the TSO subsystem and the version of TCAM that supports TSO.

Terminal Table: an ordered collection of information consisting of blocks of device-dependent informaticn cn each terminal from which a message can originate, and on each terminal, group of terminals, and application prcgram to which a message can te sent.

Termname Table: a table that contains the name of all the terminals in the system in collating sequence.

Text: that part of the message of concern to the party ultimately receiving the message (that is, the message exclusive of the header or control informaticn).

Text buffer: a tuffer that contains no part of a message header.

TIC: Transfer in Channel.

Time delay: a halt of a specific operaticn for a pre-specified amount of time.

Time sharing: a method of using a computing system that allows a number of users tc execute programs ccncurrently and to interact with them during execution.

Time Sharing Option (TSO): an optional configuration of the Operating System providing conversational time sharing from remote terminals.

TIOC buffer: a tuffer residing in the TSO region.

Tpost: the technique in TCAM by which an element is passed from one queue to ancther. The TCAM routines specify the element and the queues and the TCAM Dispatcher actually performs the acticn.

Translation Table: a collection of the information necessary to convert data frcm one transmission code to anather.

Transparent mcde: a mcde of BSC transmission in which all the data, including normally restricted data-link ccntrol characters, are transmitted cnly as specific bit patterns. Control characters that are to be effective as such are preceded by a DLE character.

Twait: the TCAM technique in which a sultask waits for an element to process by having the STCB for that subtask placed in the STCB chain of the OCB to which the needed element will be tposted.

Unit: the basic tuilding blocks from which TCAM tuffers are constructed. All units in a specific TCAM system are the same length; the user specifies this length in the KEYLEN operand of the INTRO macrc.

Unit control area: a twelve-byte ccntrol area prefixed to each TCAM buffer unit.

Unit Control Block (UCB): a system control block that describes the characteristics of the device to the I/O Supervisor and is used by the job scheduler during allocation of the device.

Use count: in a PCB, a count of the open DCBs associated with this PCB.

VCON: V-type address constant used to reserve storage for the address of an external symbol that is used for effecting branches to other programs.

Warm start: a restart in which TCAM reconstructs the environment that existed before closedown.

Write-to-Operator (WTO): an optional user-coded service whereby a message may be written to the system console operator informing him of errors and unusual system conditions that may need correcting.

Write-to-Operator with Reply (WTOR): an optional user-coded service whereby a message may be written to the system console operator informing him of errors and unusual system conditions that may need correcting. The operator must key in a response to this message.

Zone boundary: the middle of a quarter of a reusable disk message queues data set.

23-byte prefix: see subsequent buffer prefix.

30-byte prefix: see first buffer prefix.

INDEX

READER'S COMMENT FORM

IBM System/360 Operating System
Telecommunications Access Method (TCAM)
Program Logic Manual

GY30-2029-0

- How did you use this publication?

As a reference source □
As a classroom text □
As a self-study text □

- Based on your own experience, rate this publication . . .

As a reference source:	Very Good	Good	Fair	Poor	Very Poor
As a text:	Very Good	Good	Fair	Poor	Very Poor

- What is your occupation?

- We would appreciate your other comments; please give specific page and line references where appropriate. If you wish a reply, be sure to include your name and address.

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

IBM System/360 OS TCAM PLM (S360-30) Printed in U. S. A. GY30-2029-0

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)